

CC31xx & CC32xx UniFlash

Introduction

[Return to CC31xx & CC32xx Home Page](#)



Scope

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

Overview

The Uniflash utility allows the end-users to communicate with the Simple Link device via a standard UART interface. For UART connection, please refer to **Annex A: UART connection**.

The Uniflash utility enables the following functionalities:

- GUI interface
- Flushing of files (including read back verification option)
- Service pack flushing
- Storage format
- Versioning reading for boot loader 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

Upcoming for future releases:

- Reading system files
- Compatibility with secured file system implemented with TI's Simple Link CC3x00 devices

Installation

You can download the latest UniFlash here ^[1]. Please read the Release Notes for more information about this release.

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.

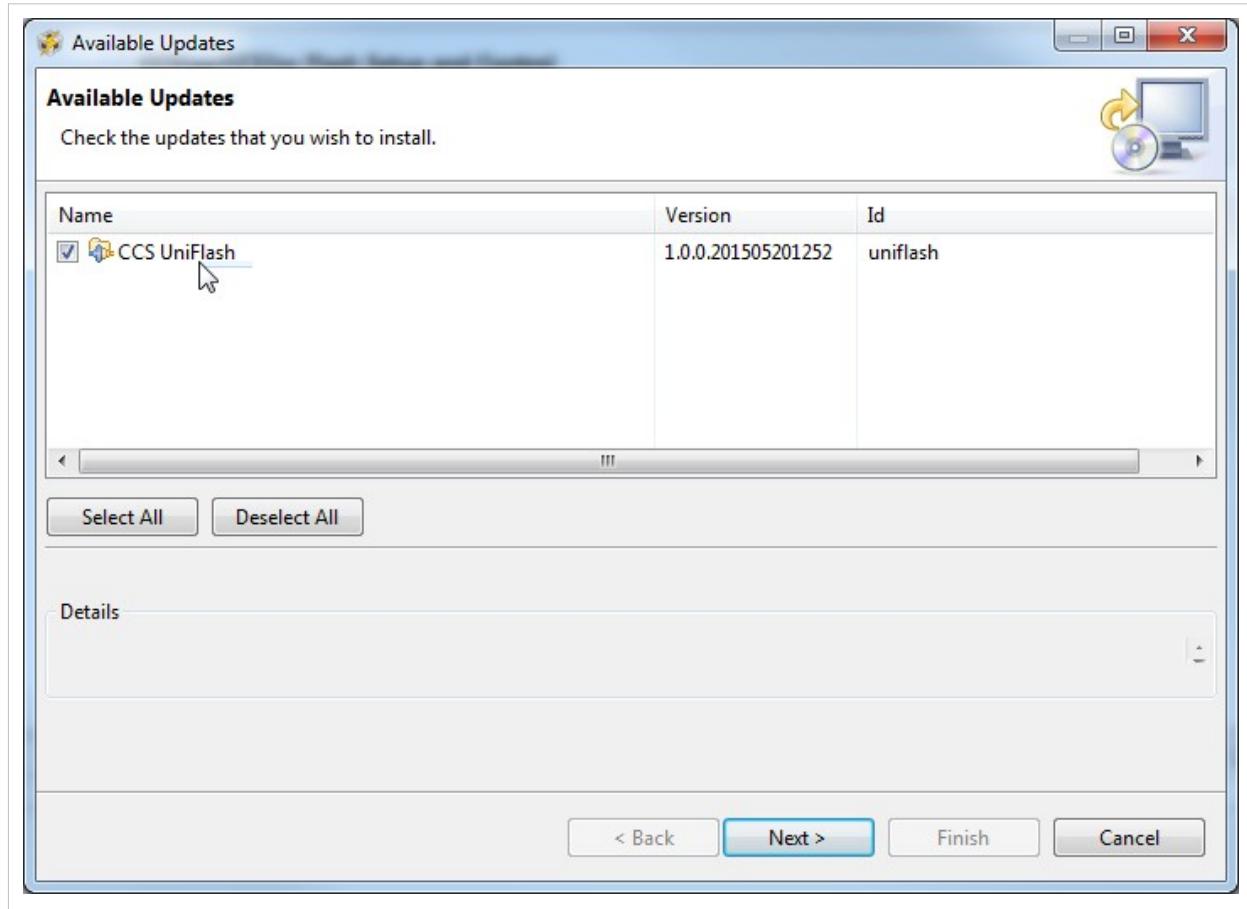
Note: up to version 3.2.0.000123, support for CC3xxx SimpleLink™ devices was not part of the Uniflash main stream where all other platforms are supported. However, now that Uniflash has been upgraded to v3.3.x, CC3xxx SimpleLink™ is also included as one of the supported chipsets. To prevent confusion,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

SW upgrade

Existing users may choose to upgrade Uniflash rather than install it from scratch. Uniflash can be upgraded via the '*help->check for update*' option.

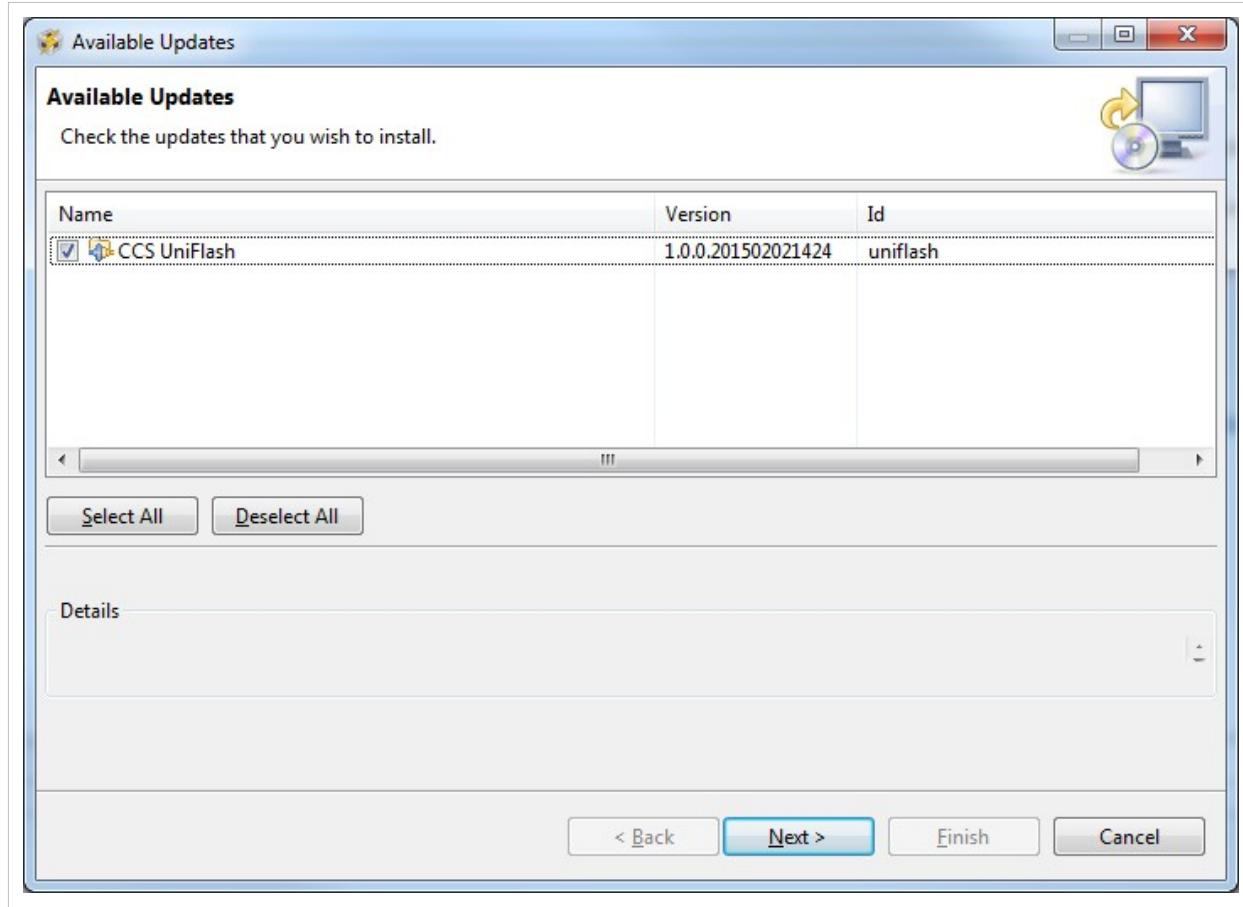
3.2.0.000123 to 3.2.1.00000 upgrade

Upon checking for update, an available update should appear on the Window. Just click the **Next** button several times and accept the terms of the license agreement. Clicking **Finish** would fetch the latest 3.2.1.00000 version and install it instead of 3.2.0.000123.



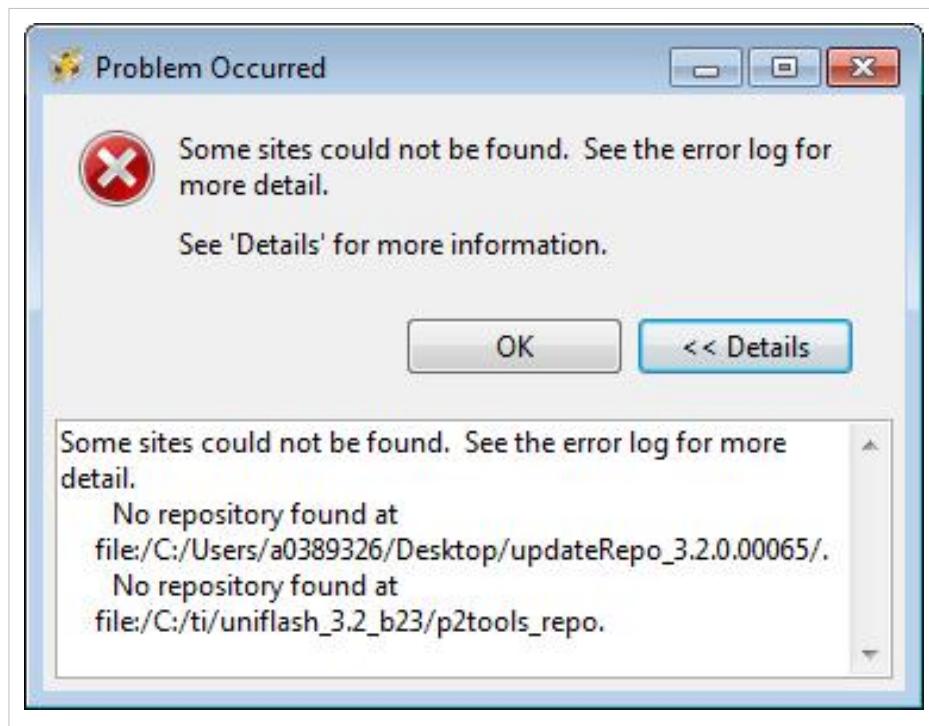
3.2.0.00065 to 3.2.0.00123 upgrade

Upon checking for update, an available update should appear on the Window. Just click the **Next** button several times and accept the terms of the license agreement. Clicking **Finish** would fetch the latest 3.2.0.000123 version and install it instead of 3.2.0.00065.

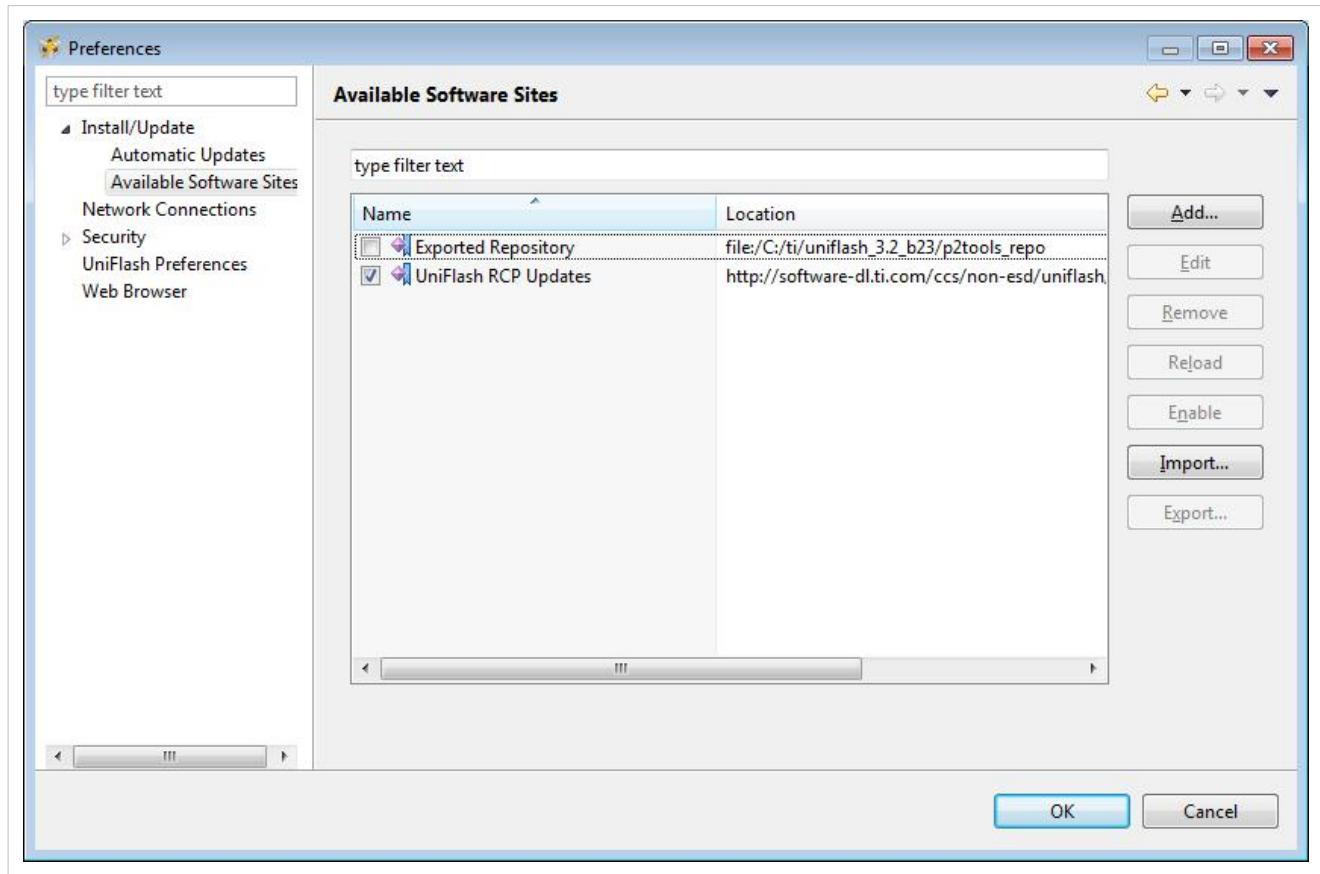


3.2.0.00065 upgrade

There is an update error for 3.2.0.00023 users only. If users check for updates, they will see the below error:



This error can be safely ignored. But if users want to prevent the error from happening, they can go to '**Window->Preference**', '**Install/Update->Available Software Sites**' tab, and unselect the '**Exported Repository**' entry.



Linux

UniFlash is a 32-bit application. Therefore, if you are on a 64-bit Linux OS distribution, you will 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 CCS are similar, please read the Linux Installation Instructions for CCS, search for your distribution from the supported list, and find the recommended packages to install.

Limitations

Note: Linux Uniflash is a Beta version and not an official release

Current 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 (i.e. with 'Verify' flag checked)

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.

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. Please talk to your network administrator on instructions for doing this.

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 we 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, we shipped a version of the required C++ library 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 be loaded and UniFlash GUI will be started.

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

COM Port

When using UniFlash (in both the GUI and CLI), you will 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, just enter 2 as the COM port value instead of 'ttyUSB2'.

Porting between Windows and Linux

UniFlash Configuration/Session files should be compatible between Windows and Linux, as long as the file paths (ie; 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. Please read the Relative Path Support section for more information on this topic.

Prerequisites

Software

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

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

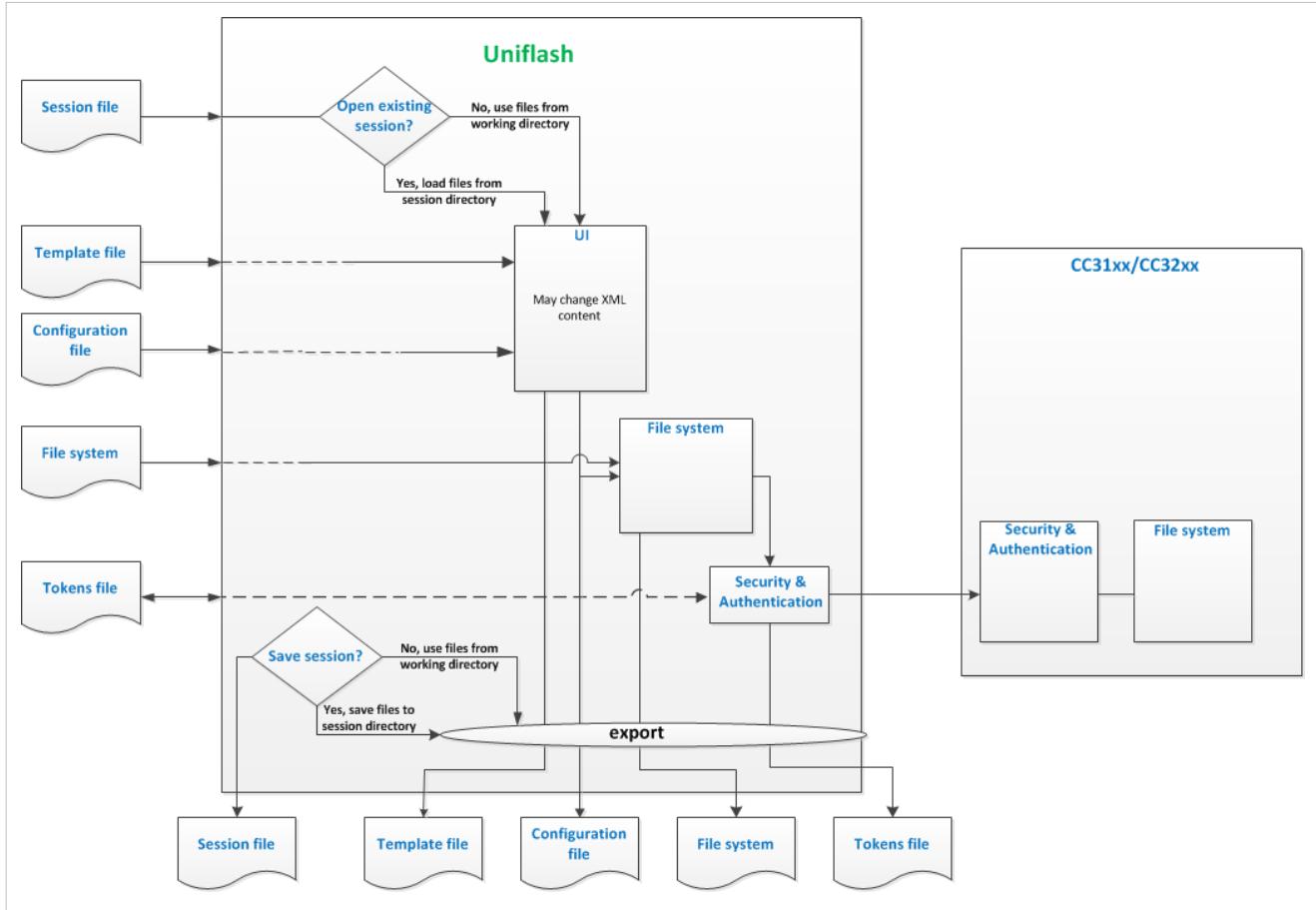
Uniflash Block Diagram

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:

1. ***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, etc). The template file is formatted as an XML file
2. ***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.
3. ***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, e.g. 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
4. ***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
5. ***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.
6. ***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, etc.

The following diagram illustrates the structure:



Session support

Overview

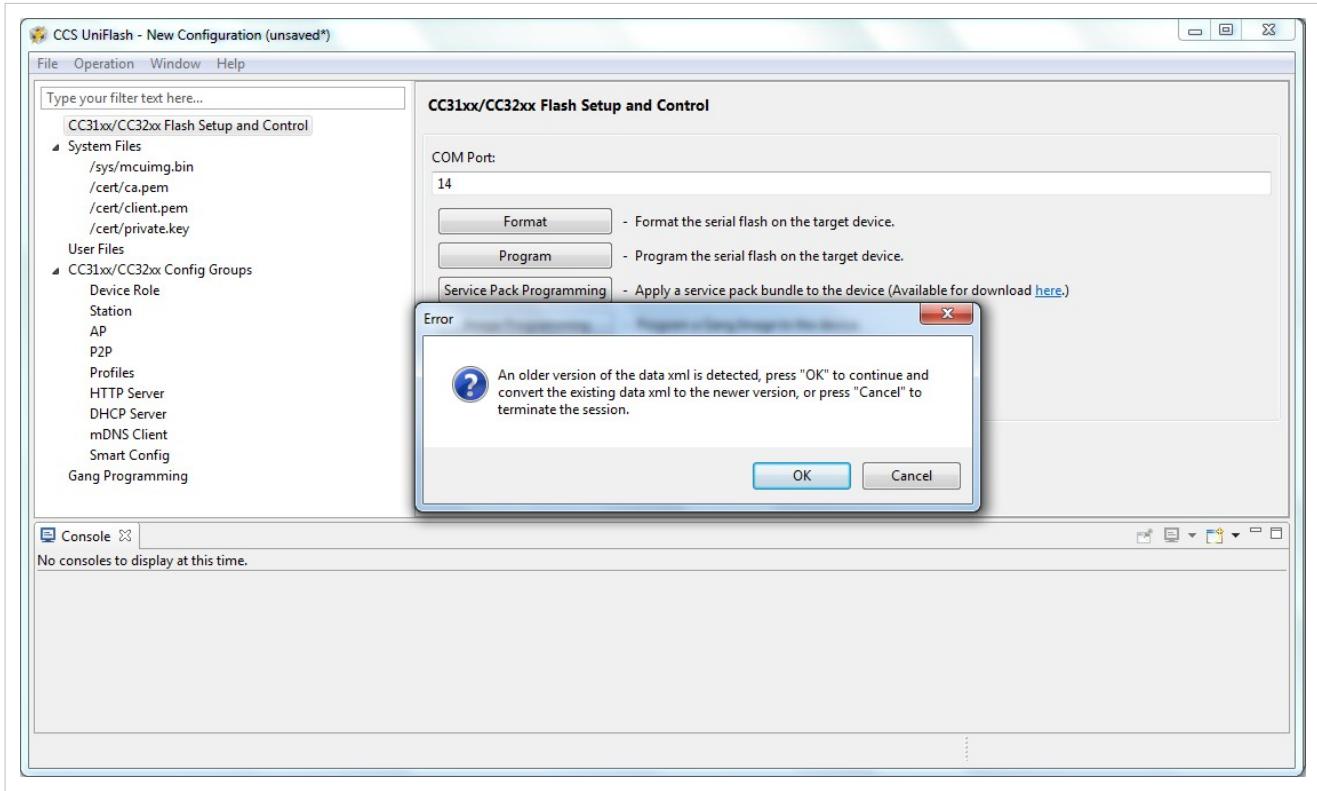
Sessions can be viewed as working directory where all device specific files are located. It is important to note that each device keeps its own files and thus each device would most likely have a separate session.

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.

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.



Creating a session

Session content

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, etc.

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
 - dhcpsrv.cfg
 - httpsrv.cfg

- mdns.cfg
- pref.net

Session files files are saved as *.usf files (UniFlash Session File), and includes information about the interface, device, and the session we 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, e.g. 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
```

Creation procedure

There are two ways to create a session. One 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.

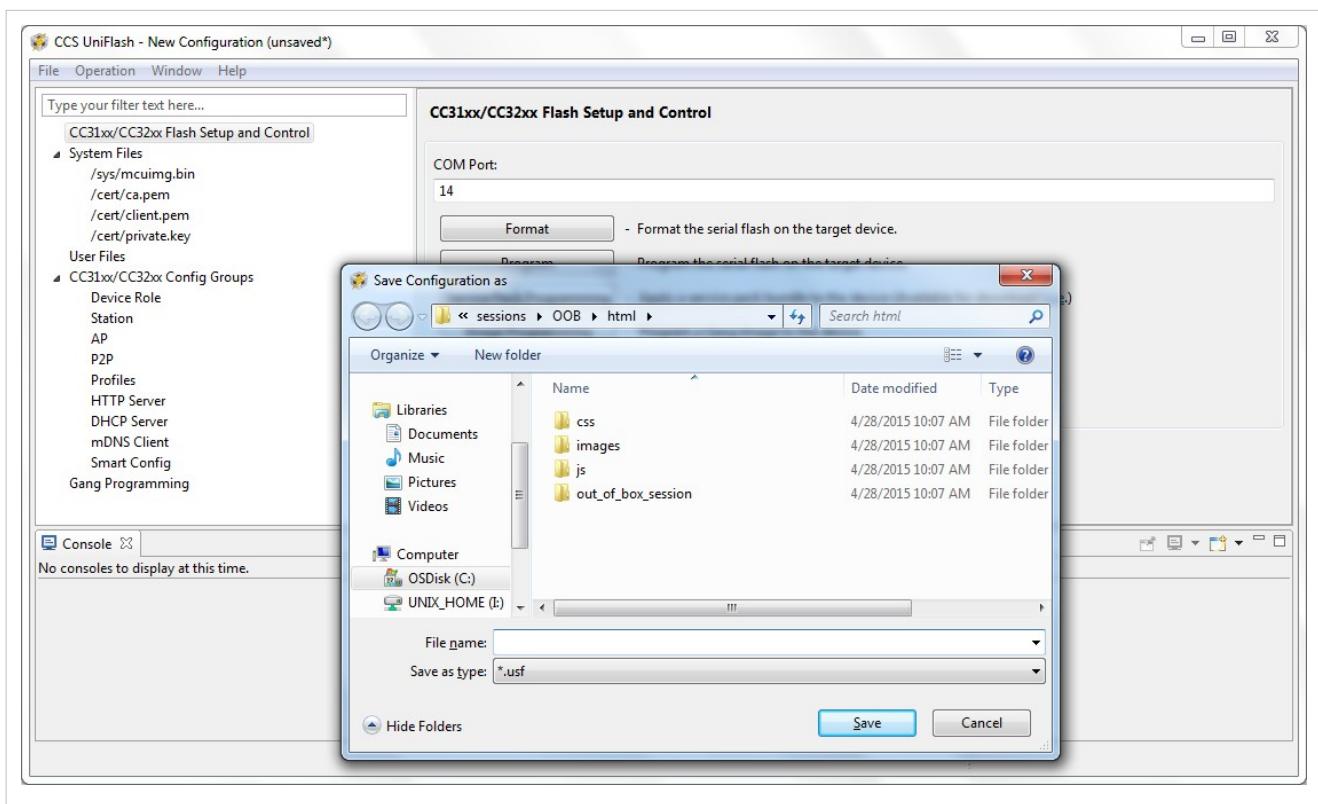
Relative path support

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).

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 (ie; 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.

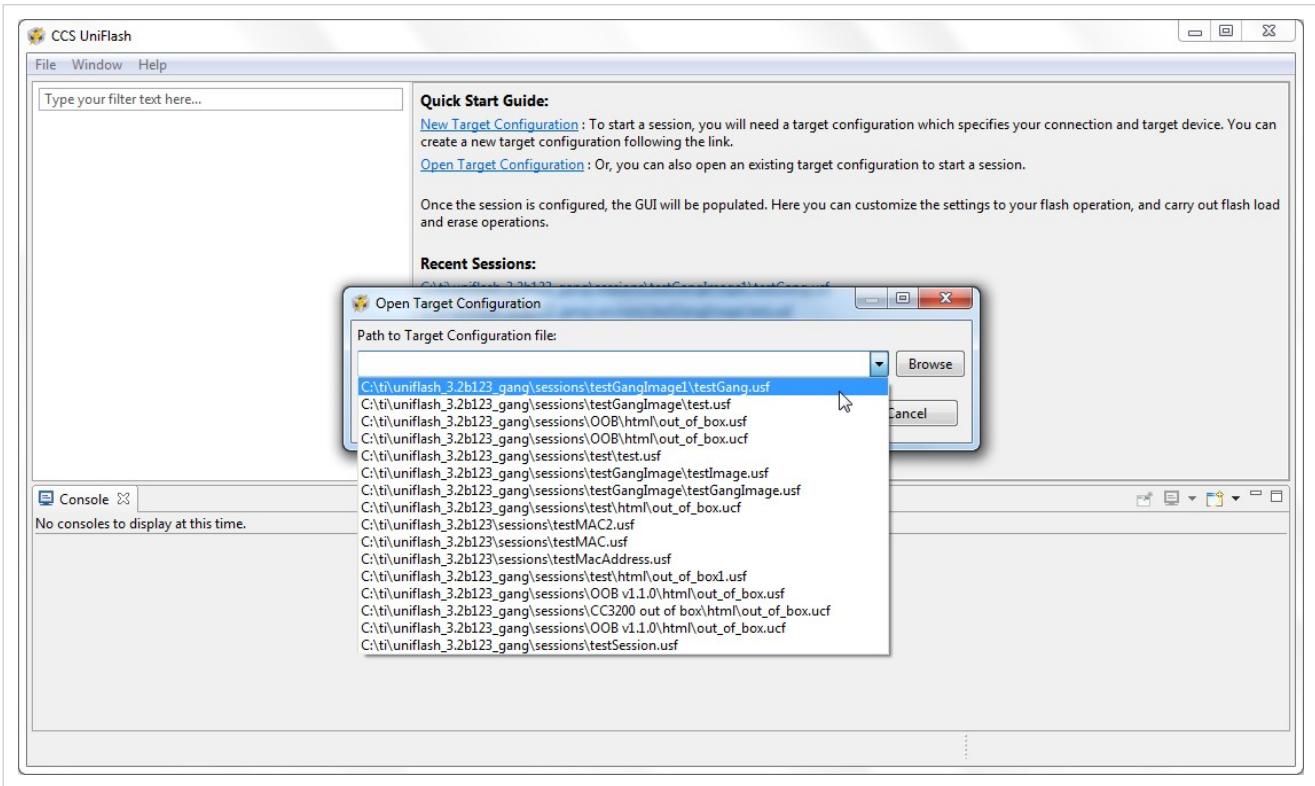
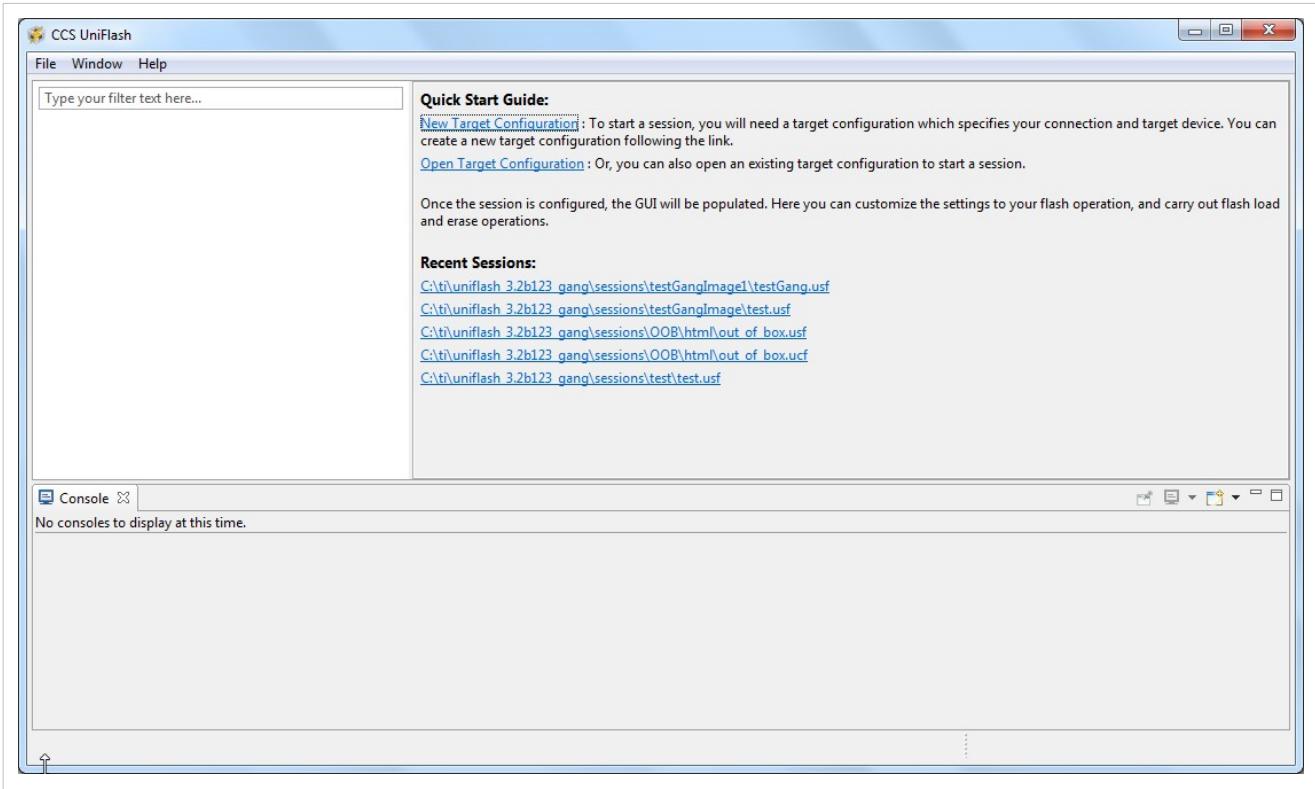
Note that if users do not save their configuration before closing UniFlash, their changes will be lost (although the actual files are still there in the temporary location).



Loading a session

There are two ways to load a session. One 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.



Secured File System support

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

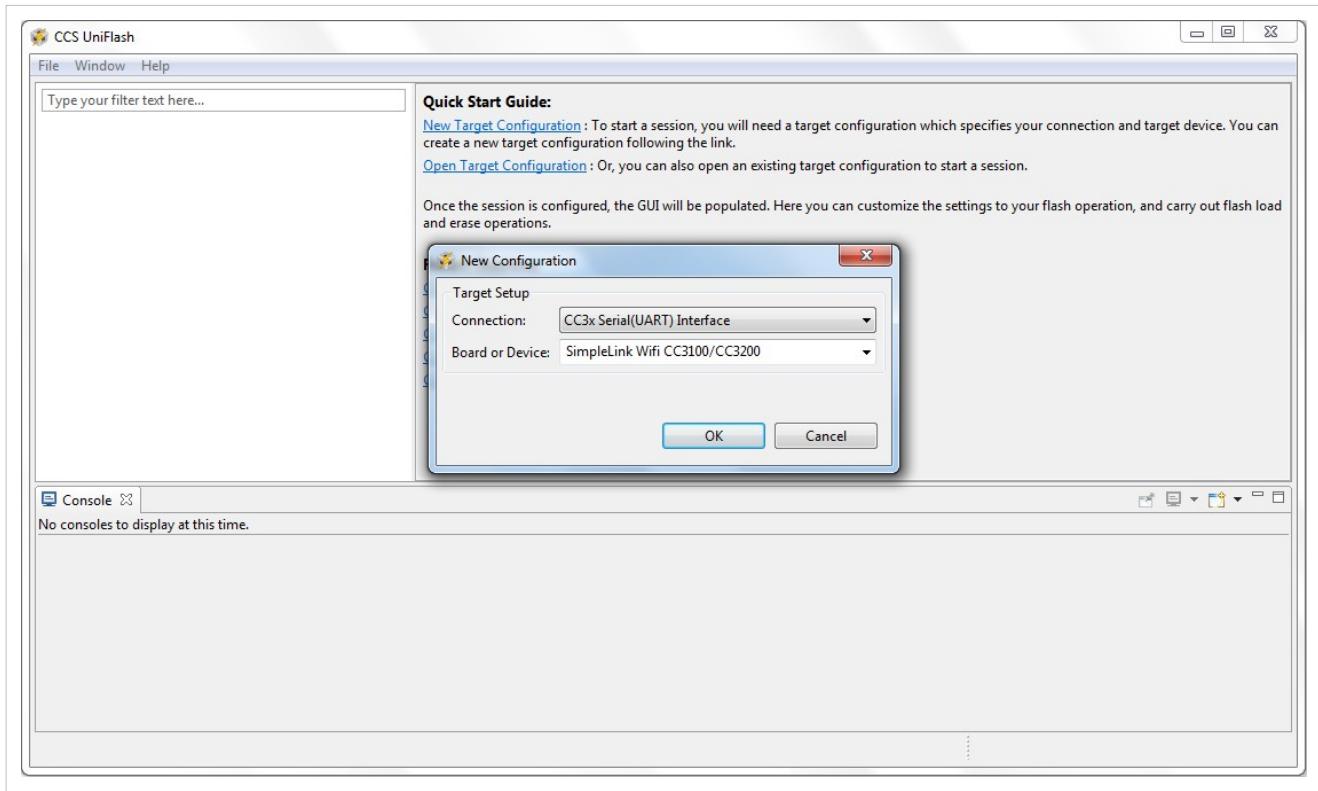
Note: Uniflash version v3.2.0.00123 or earlier had the option to format the device as secured (in case it is not) and also create secured files.

However, secured file system support is deprecated from newer Uniflash versions. The support for secured file system would be added again once new devices fully support this feature.

GUI Interface

Main Screen

Upon running the Uniflash, the user is required to choose the target setup. The target setup may be configured via the top bar by *file->New Configuration* or 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**.



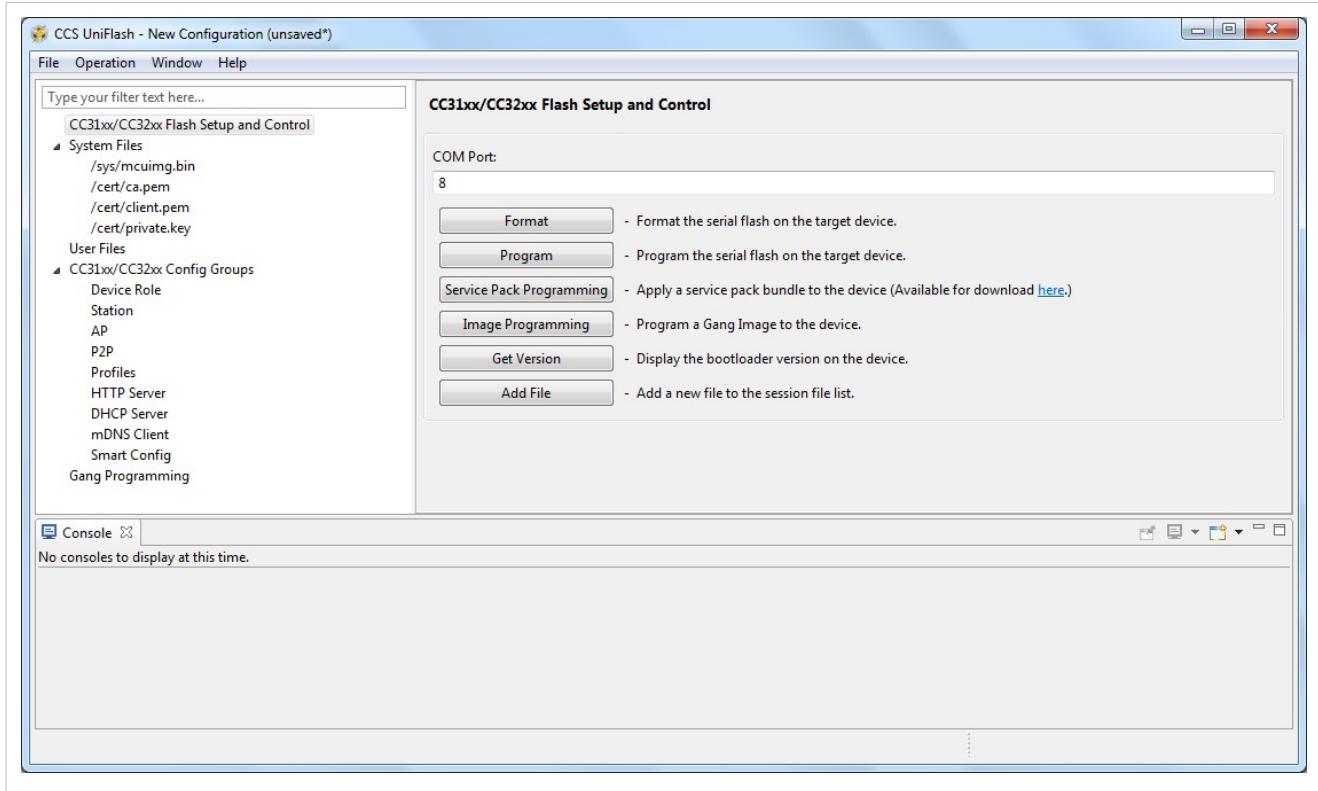
Loading the target configuration provides the Uniflash main screen.

The main screen is divided into 3 main sub screens.

- *Uniflash main view*: this is the upper left section on Main screen 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 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.



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.

The following table describes all possible permutations:

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 3 rd COM port

In addition, 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').

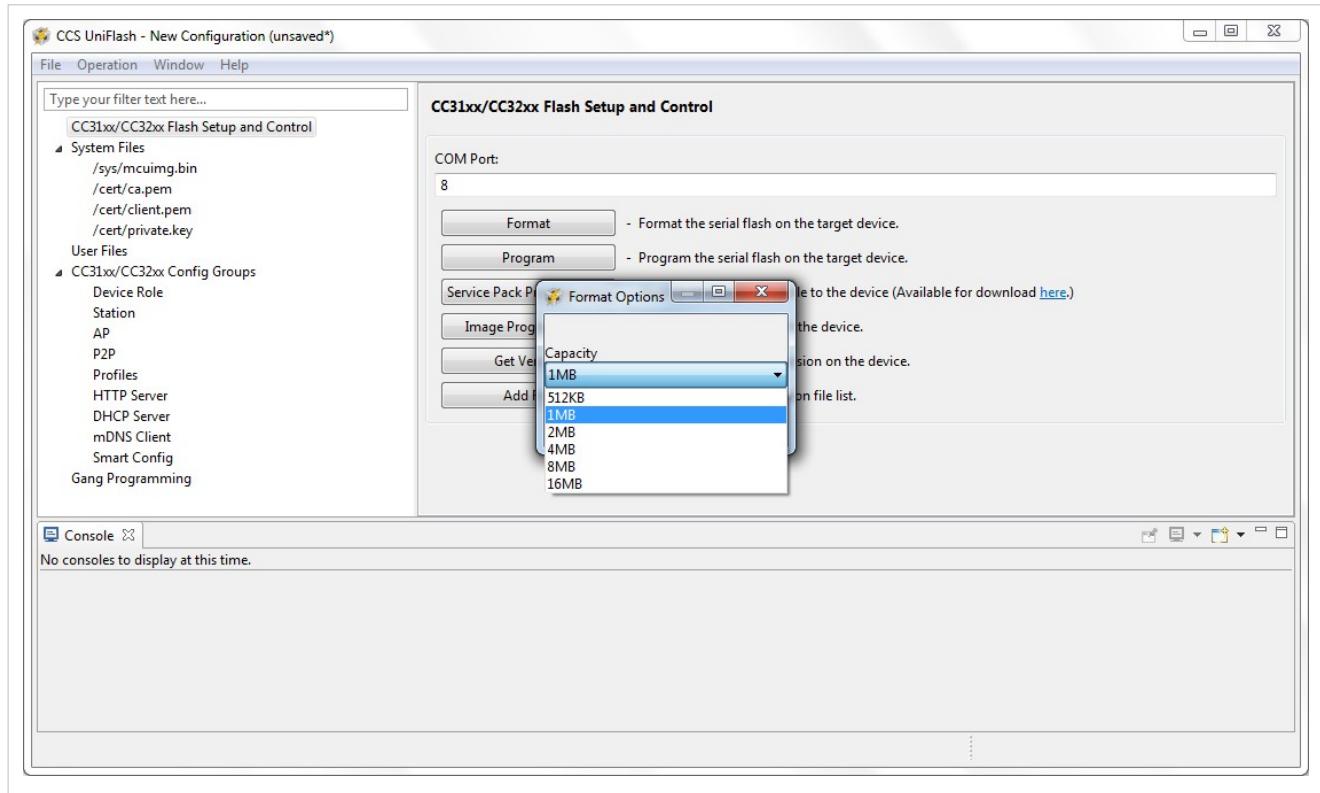
Format

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 to 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.



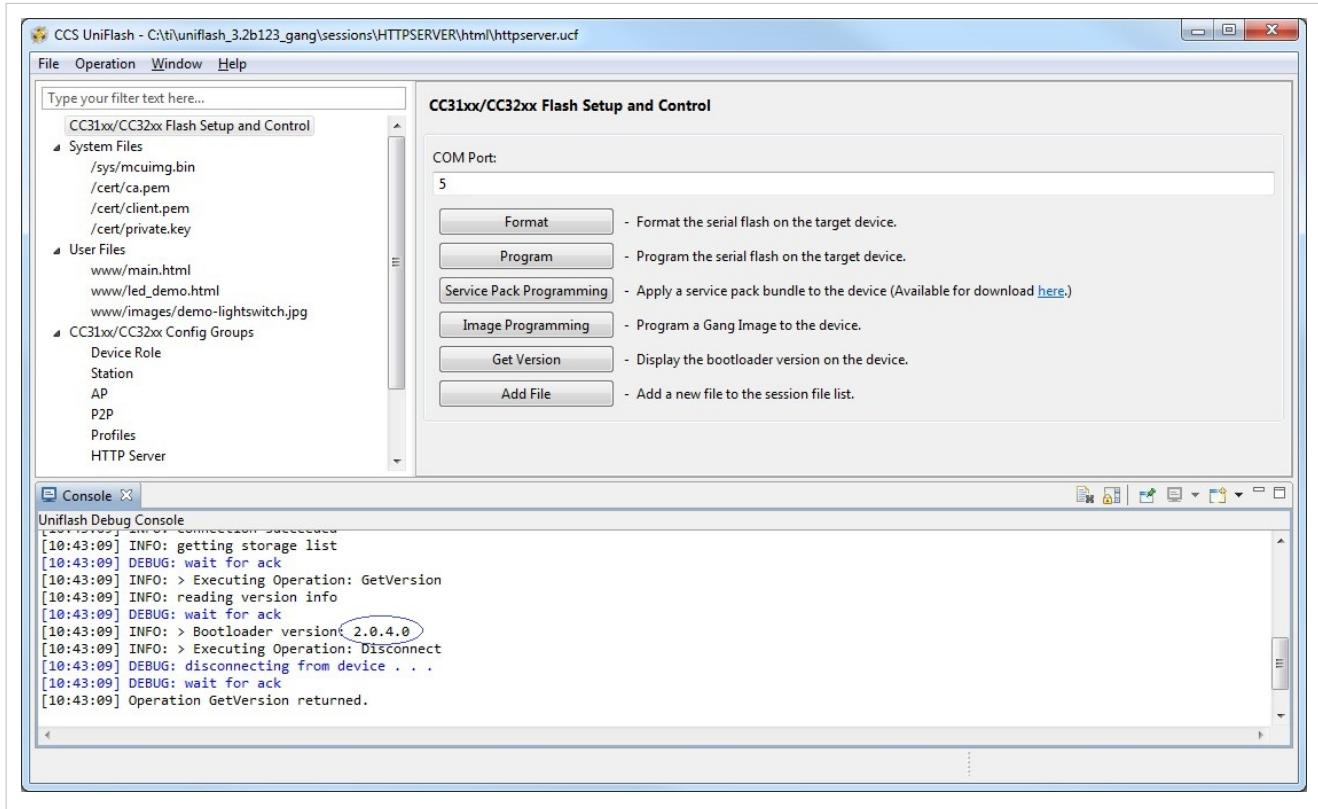
Get Version

The version that can be fetched from the device is the bootloader version (as Uniflash interacts with the boot loader section). Additionally, the chipset version is fetched. The bootloader version is composed of 4 numbers, separated by periods.

The syntax is:

<SoftwareMajor.SoftwareMinor.SoftwareVersion.SoftwareSubVersion>

Please note that the boot loader version fetched is the one that is located in ROM.



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

File Programming

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):

1. each file with the '**erase**' argument set to True is erased from the device
2. each file with the '**update**' argument set to True is flashed to the device
3. 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.

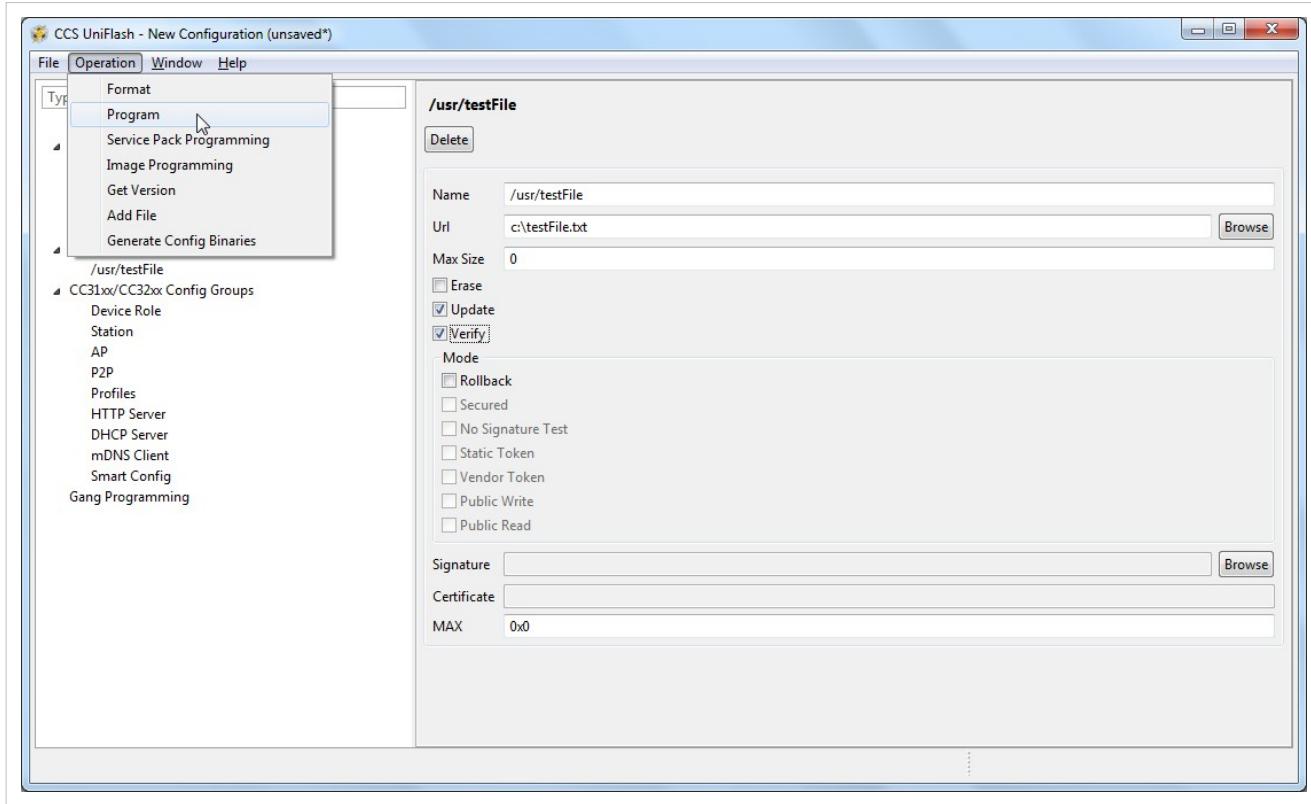
Uniflash does not do conversions for binary-to-text encoded files (e.g. hex, Base64).

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



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

Secured file programming

See **Secured File System support** paragraph. 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 greyed out.

Service Pack Programming

General

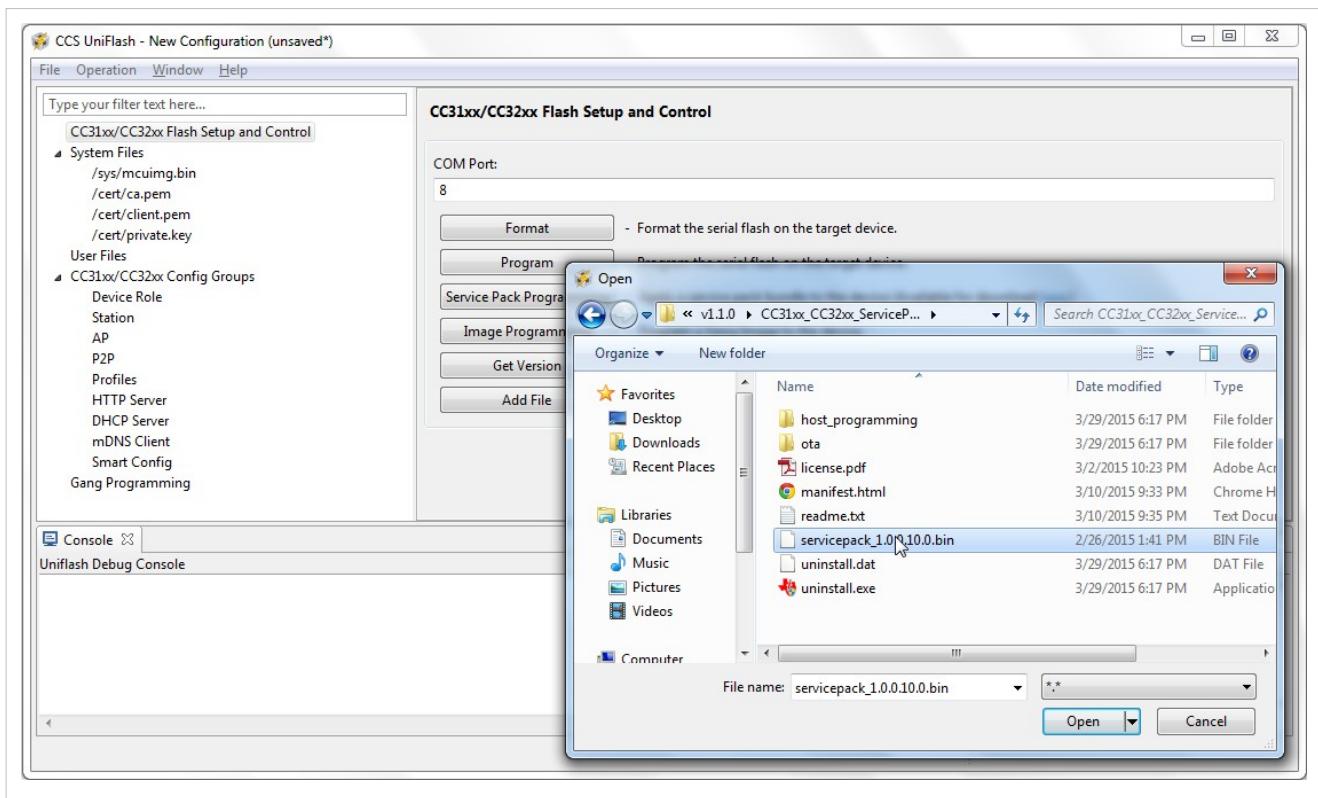
Service pack is a common name for the image required for updating the device (either upgrade or downgrade the device). The service pack is a single file containing images for all flavors of the device, regardless if it is CC3100/CC3200 or ROM/Flash device. The detection is done automatically by Uniflash.

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

Procedure

Please follow the ordered steps below for successful update procedure:

- 1.Format the device prior to flashing. This step is essential and should be performed at least once. With next service pack updates, format operation would not be required. Please follow Format paragraph for more details.
- 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.Floating should begin and the version flashed is printed just before flashing is started



Adding a new file to the device

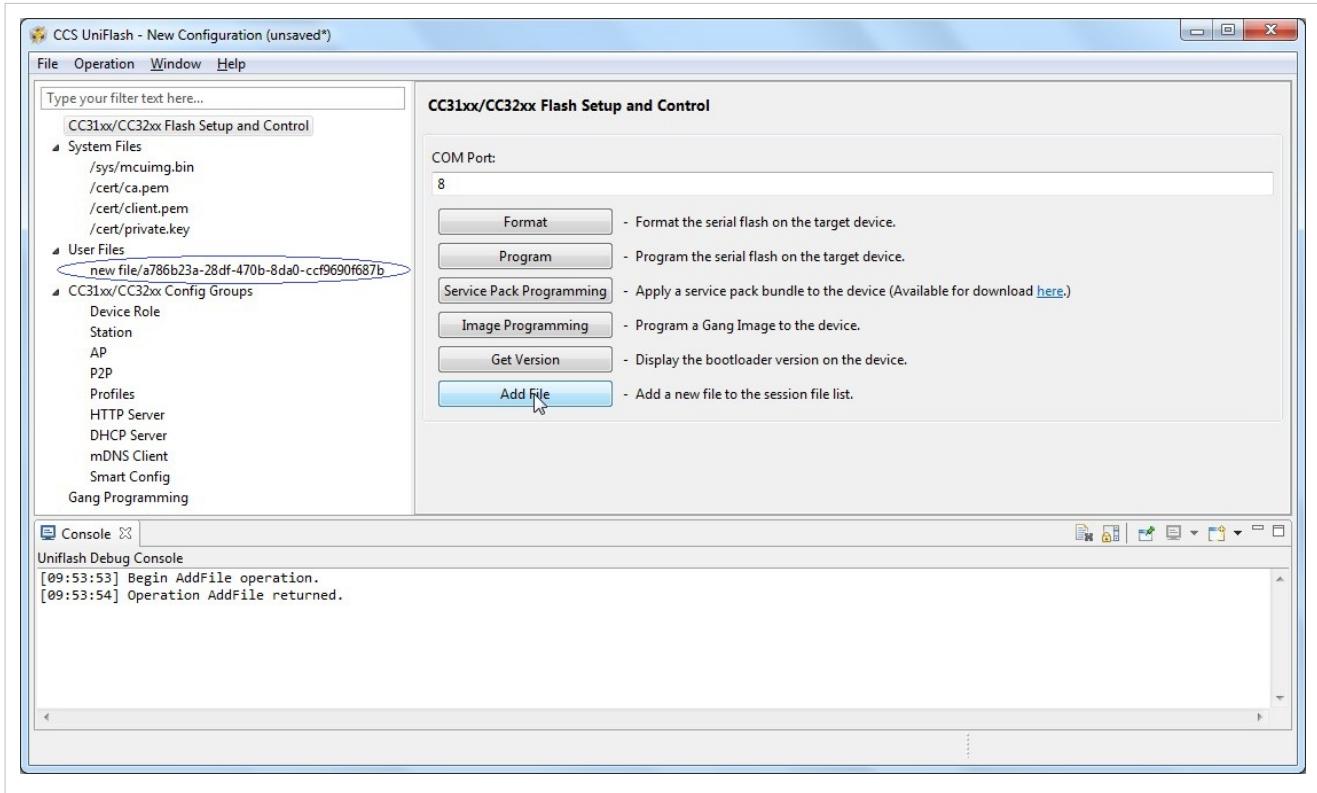
Note: for web pages, please refer to the Programmer's guide, HTTP Server paragraph.

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.

The Programmer's guide can be found under:

CC31xx_Programmers_Guide.

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.



Erasing existing file from the device

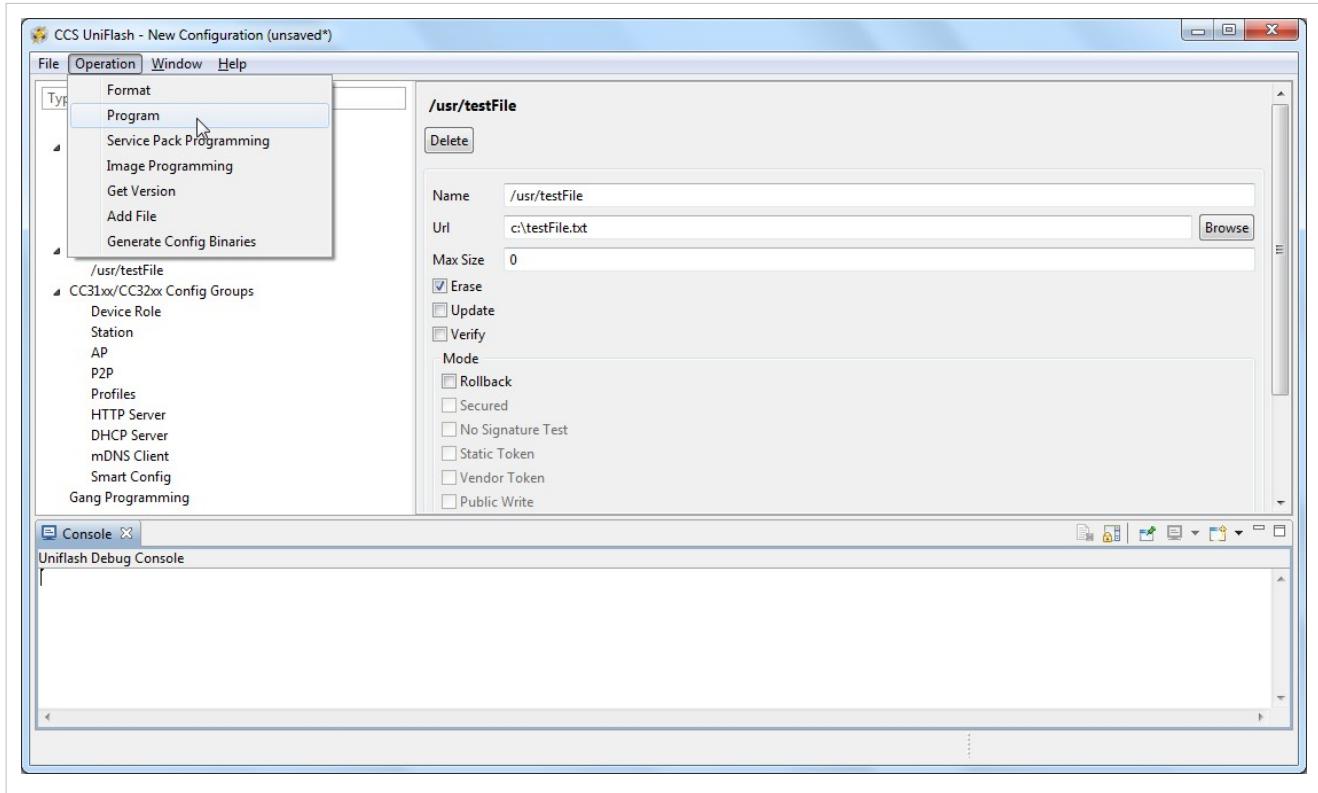
Uniflash enables the user to remove any file via the GUI using the '**erase**' parameter under each file.

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.

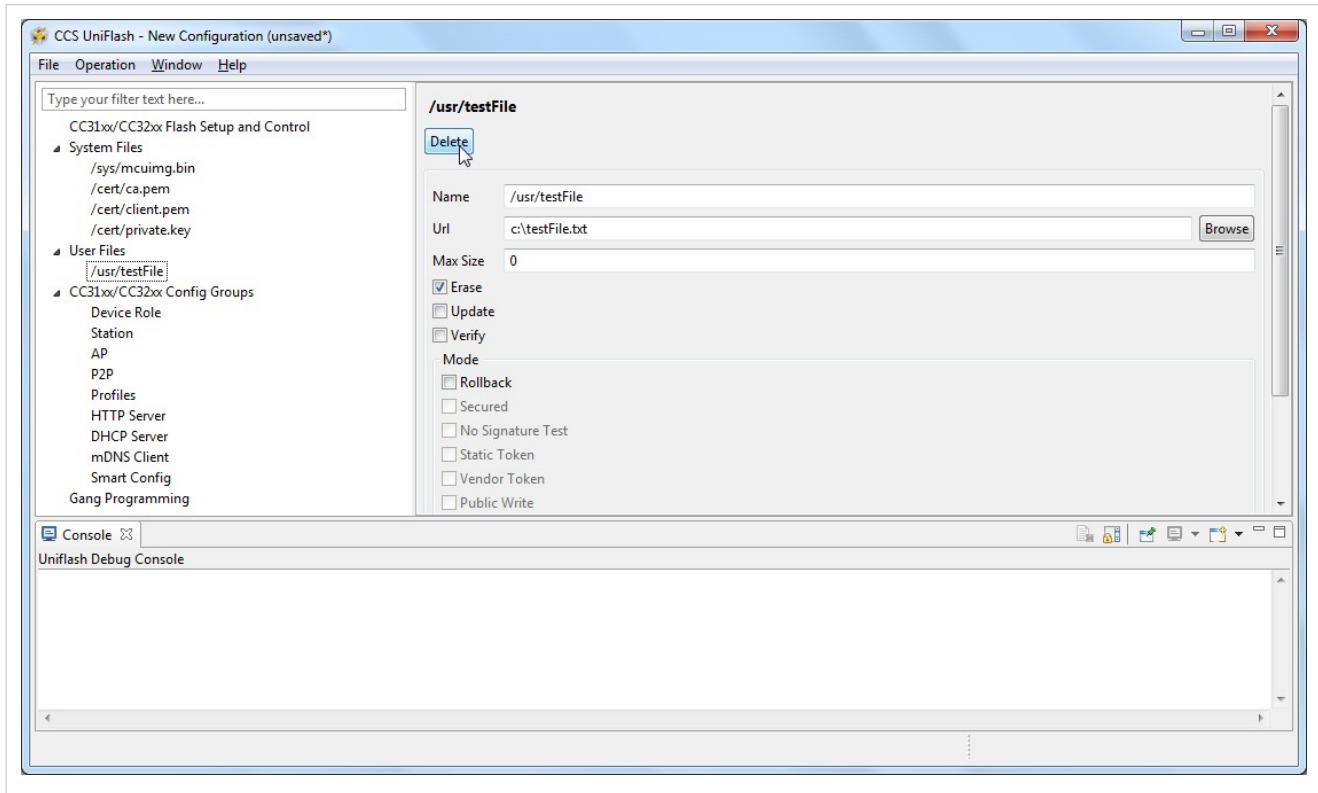
Please note that erasing a file from the device does not delete it from the GUI.

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



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).



CC3200 support

Overview

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

For better understanding:

- Current exiting 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**.

Configuration and programing

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

Future secured CC3200:

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

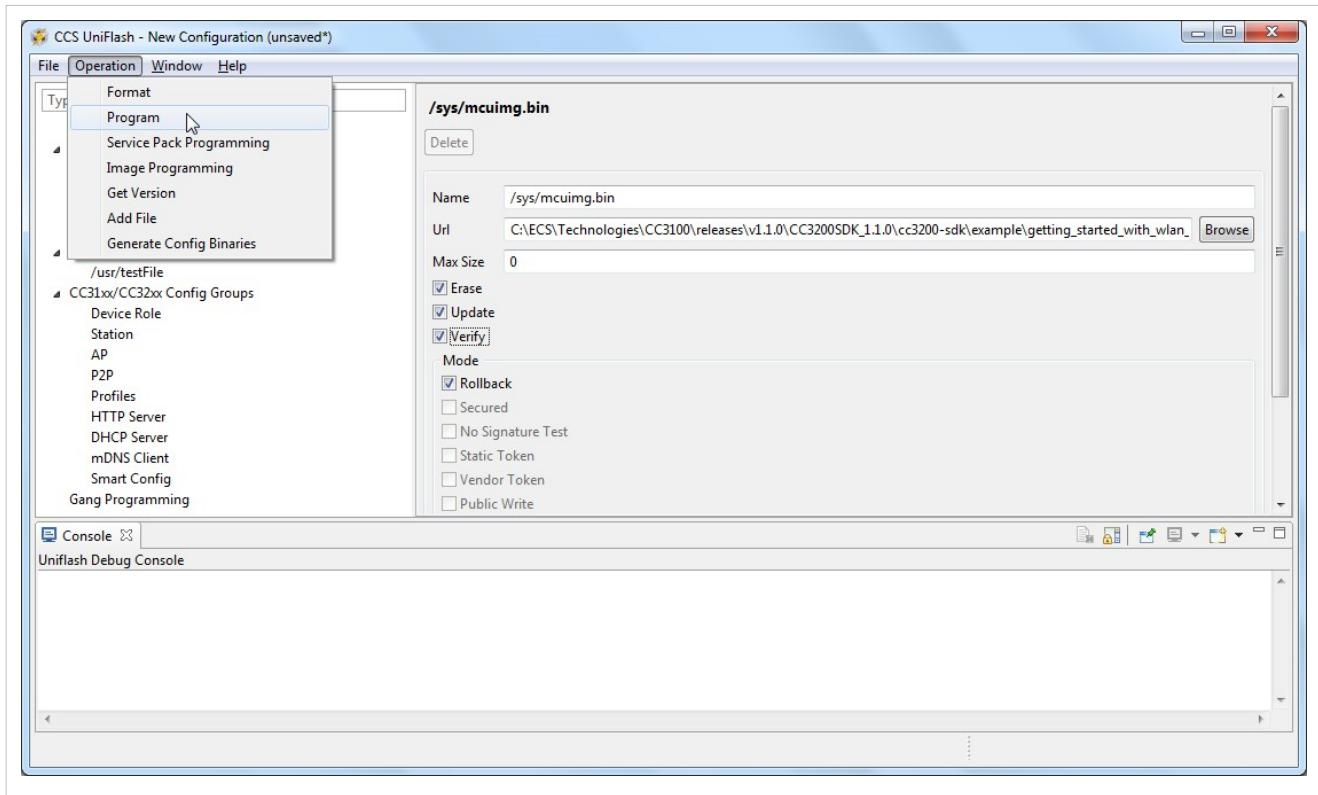
2. User can create a mcuimg file as secured and it will work

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. Please 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.



Command Line support

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.

Programming

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, i.e. several operations can be invoked serially in a single command.

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 (ie; 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 (ie; 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>

```

Available options

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

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

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

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

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

Argument	imageVerify
Description	image verification after programming. Linux only
Options	[true, false]
Default	false
Required?	Yes, can also be set on the *.usf

Available operations

Argument	getVersion
Description	Getting the boot loader version of the device

Argument	format
Description	Serial flash format

Argument	program
Description	Programming of any element except from service pack

Argument	servicePackUpdate
Description	Service pack programming

Argument	imageProgramming
Description	Image programming

Legend

The following table summarizes all 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 boot loader 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

Programming example

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

The example is for Windows but applies to Linux as well.

```
>>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/httpsrv.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>
```

Configuration file support

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 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.

Configuration groups – parameters' description

Device Role

The following table lists all parameters applicable for **Device Role** group:

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

Station

The following tables list all parameters applicable for **Station** group:

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

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

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

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

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

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

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

Parameter	Station IP Address
Description	IP address
Options	Valid IP address separated by '.', e.g. 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.
Dependencies	Applies only when <i>Station Addressing Scheme</i> is set to <i>Static</i>

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

Parameter	Station Default Gateway
Description	Default gateway
Options	Valid IP address separated by '.', e.g. 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.
Dependencies	Applies only when <i>Station Addressing Scheme</i> is set to <i>Static</i>

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

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

AP (Access Point)

The following table lists all parameters applicable for **AP** group:

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

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

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

Parameter	WEP Key Format
Description	WEP key format in AP role
Options	Either <i>Hexadecimal</i> or <i>ASCII</i>
Dependencies	Applies only when <i>Security Type</i> is set to <i>WEP</i>

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

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

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

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

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

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

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

Parameter	Beacon Interval
Description	Beacon Interval is the interval between AP beacons notifications
Options	It is represented in TU's where each TU (Time Unit) is 1024 uSec
Dependencies	None

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

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

Parameter	AP IP Address
Description	IP address
Options	Valid IP address separated by '.', e.g. 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.
Dependencies	Applies only when AP Addressing Scheme is set to <i>Static</i>

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

Parameter	AP Default Gateway
Description	Default gateway
Options	Valid IP address separated by '.', e.g. 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.
Dependencies	Applies only when AP Addressing Scheme is set to <i>Static</i>

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

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

P2P (Peer 2 Peer)

The following table lists all parameters applicable for **P2P** group:

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

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

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

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

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

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

Parameter	Peer2Peer Listen Channels
Description	The device listen channel during Peer2Peer discovery phase
Options	Should be one of the social channels, 1, 6 or 11
Dependencies	None

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

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

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

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

Parameter	Station IP Address
Description	IP address
Options	Valid IP address separated by '.', e.g. 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.
Dependencies	Applies only when <i>Station Addressing Scheme</i> is set to <i>Static</i>

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

Parameter	Station Default Gateway
Description	Default gateway
Options	Valid IP address separated by '.', e.g. 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.
Dependencies	Applies only when <i>Station Addressing Scheme</i> is set to <i>Static</i>

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

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

Parameter	AP IP Address
Description	IP address
Options	Valid IP address separated by '.', e.g. 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.
Dependencies	Applies only when <i>AP Addressing Scheme</i> is set to <i>Static</i>

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

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

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

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

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

Profiles

The following table lists all parameters applicable for **Profiles** group:

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

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

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

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

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

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

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

Parameter	Profile x Priority
Description	Profile x priority
Options	Number between 0-255
Dependencies	None

Parameter	Profile x BSSID set
Description	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
Options	<i>State, Enable or Disable</i>
Dependencies	None

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

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

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

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

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

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

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

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

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

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

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

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

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

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

HTTP Server

The following table lists all parameters applicable for **HTTP Server** group:

Parameter	HTTP Server Port
Description	HTTP server TCP port
Options	Any valid TCP port number
Dependencies	None

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

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

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

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

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

DHCP Server

The following table lists all parameters applicable for **DHCP Server** group:

Parameter	DHCP Server Lease Time
Description	DHCP Lease time
Options	Time in seconds
Dependencies	None

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

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

mDNS client

The following table lists all parameters applicable for **mDNS client** group:

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

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

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

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

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

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

Smart Config

The following table lists all parameters applicable for *Smart Config* group:

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

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

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

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

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

Configuration groups – GUI interface

Device Role

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.

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.

Generation of binary templates

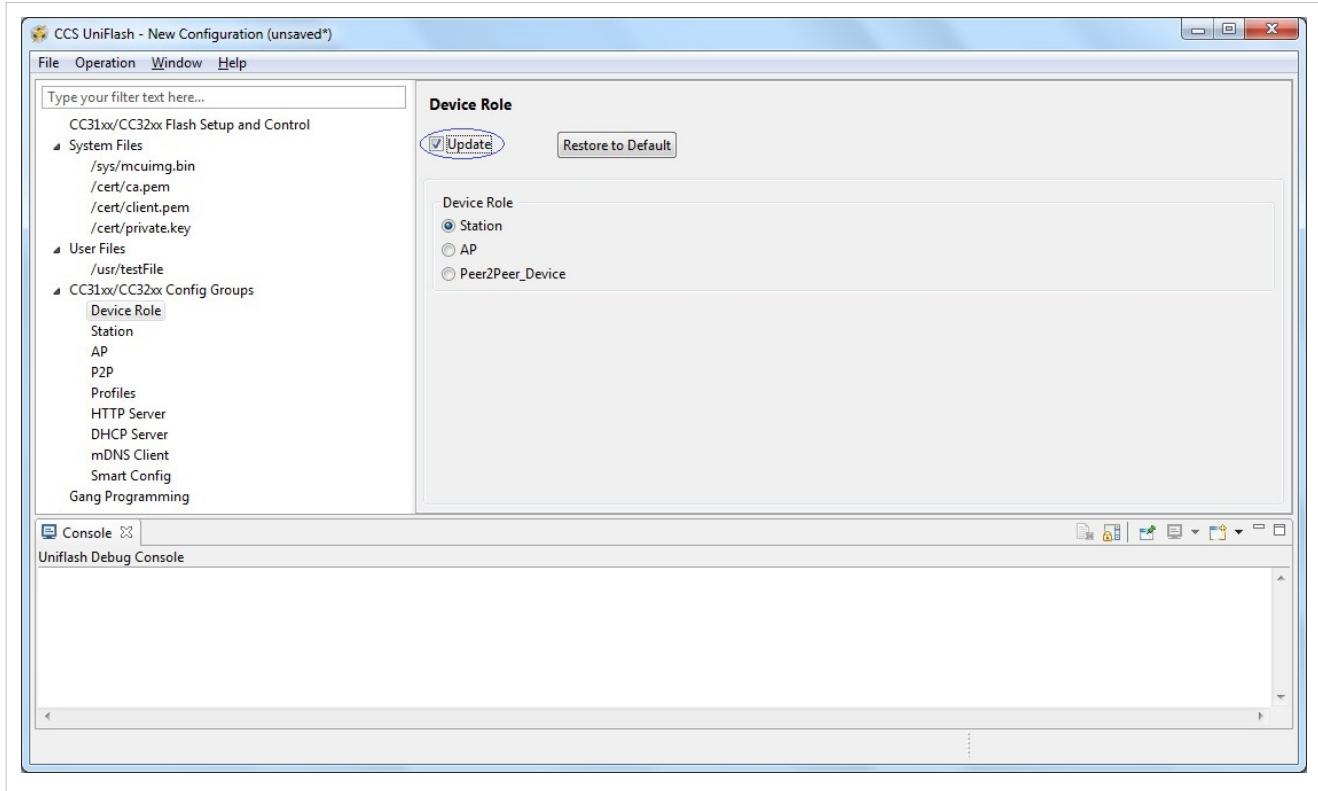
Unlike programming which generates the binary templates and program those to the target device, it is possible just to generate those binary templates offline. This option can be used for debug purposes but also to make an offline session that would be used as part of Image Programming. The latest is 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**.

Configuration

Following is a screen shot of all **Device Role** parameters. In this screen, the following parameters are defined:

- **DeviceRole** is set to **Station**



Station

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.

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.

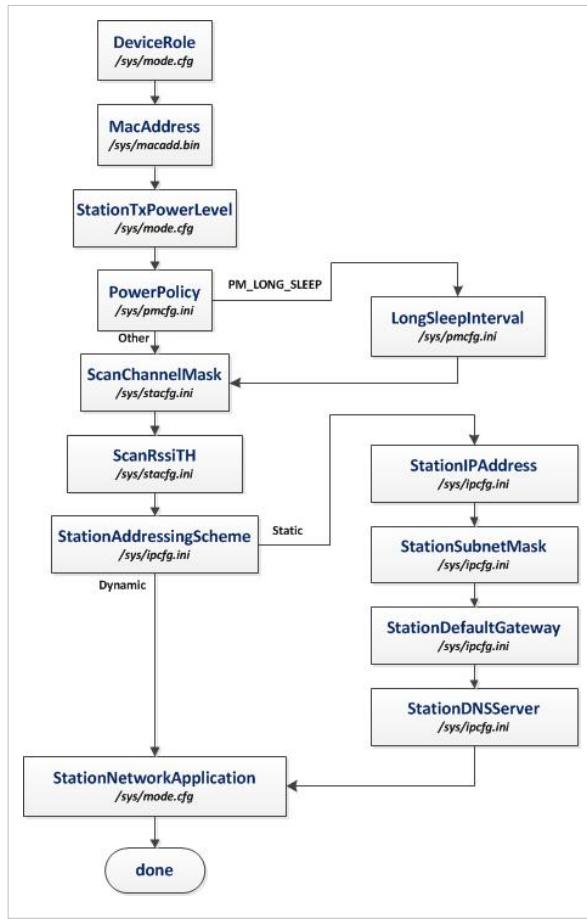
Generation of binary templates

Unlike programming which generates the binary templates and program 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**.

Flow

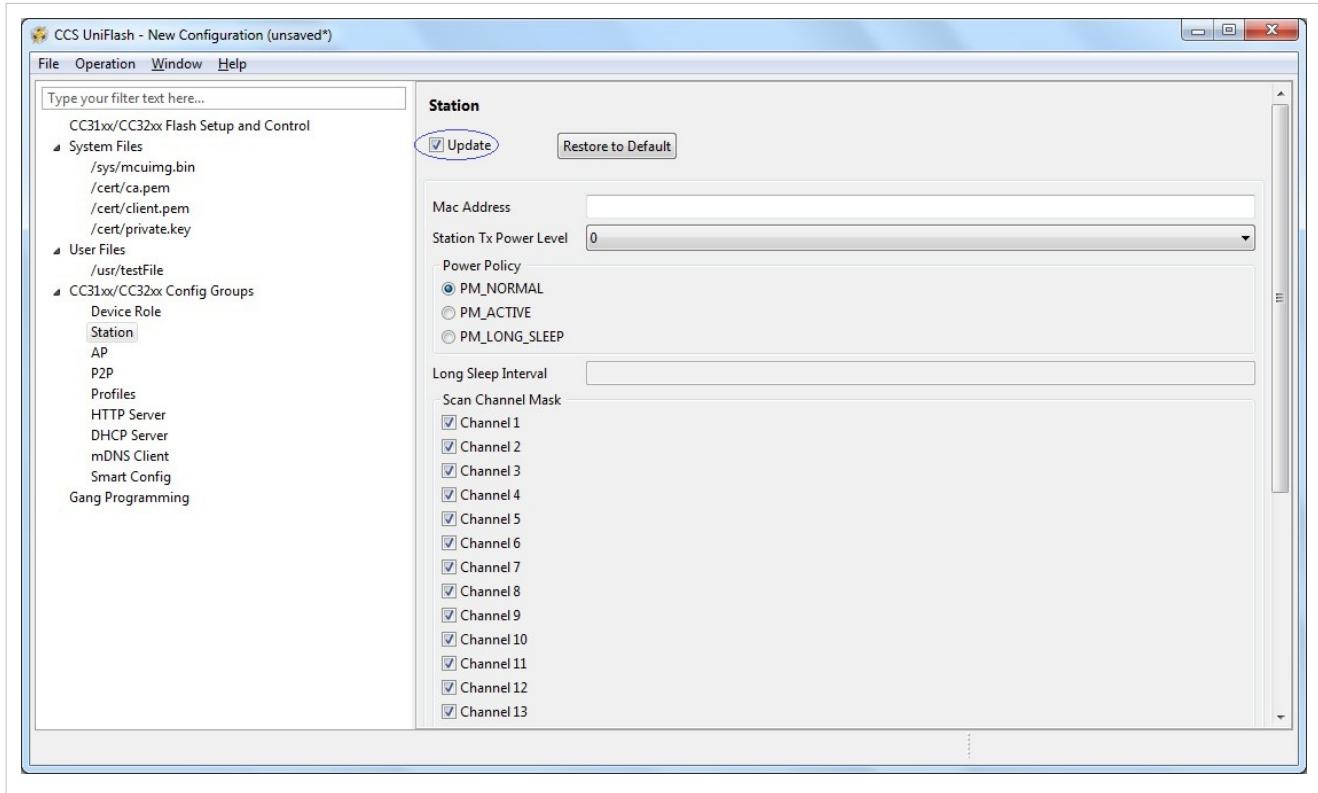
The following figure illustrates the parameters' flow during configuration. It is highly advised to follow this flow but it is not mandatory. The user may choose to go through configuration steps differently.



Configuration

Following is a screen shot of all **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, i.e. 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 -95dBm. All received beacons/probe requests stronger than this level would be received by the device.
- **Station Addressing Scheme** is set to default **Dynamic**, i.e. 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.



AP (Access Point)

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.

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.

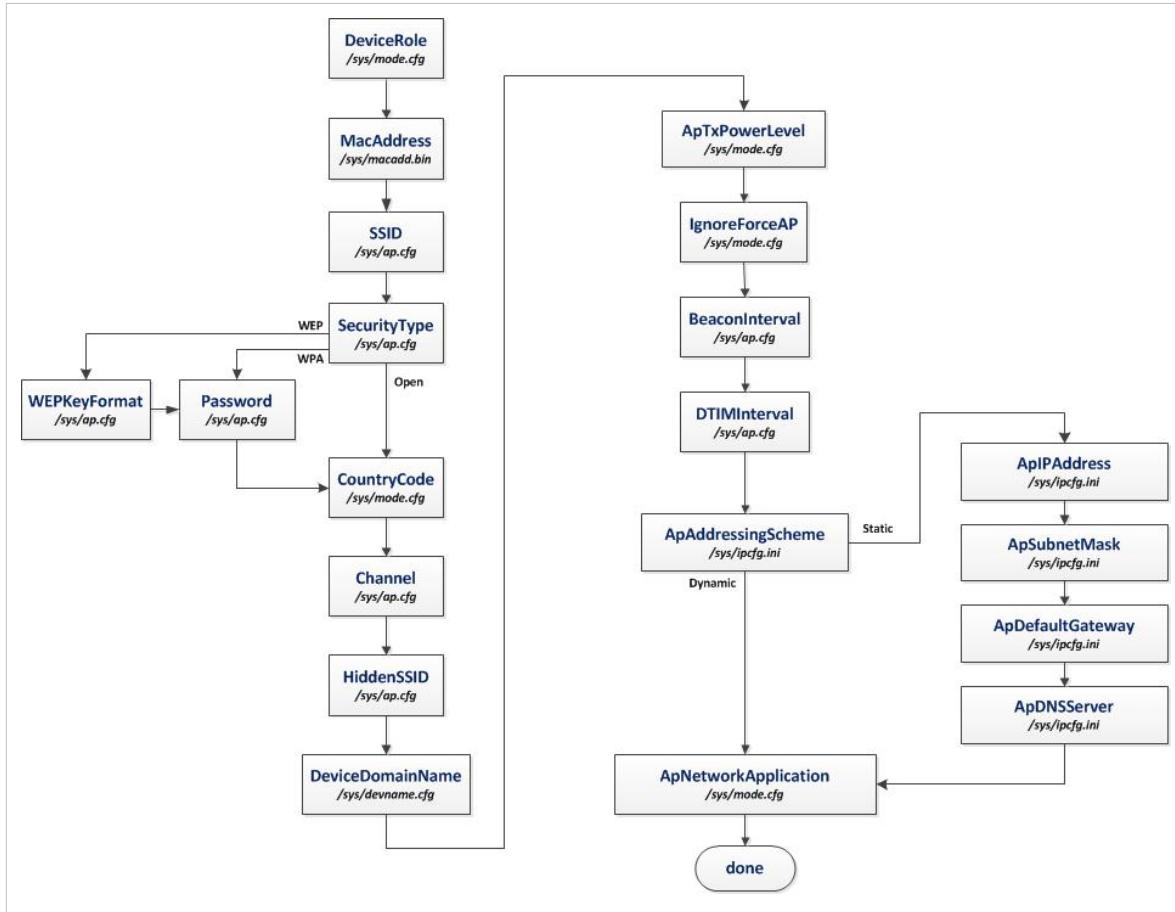
Generation of binary templates

Unlike programming which generates the binary templates and program 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**.

Flow

The following figure 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.

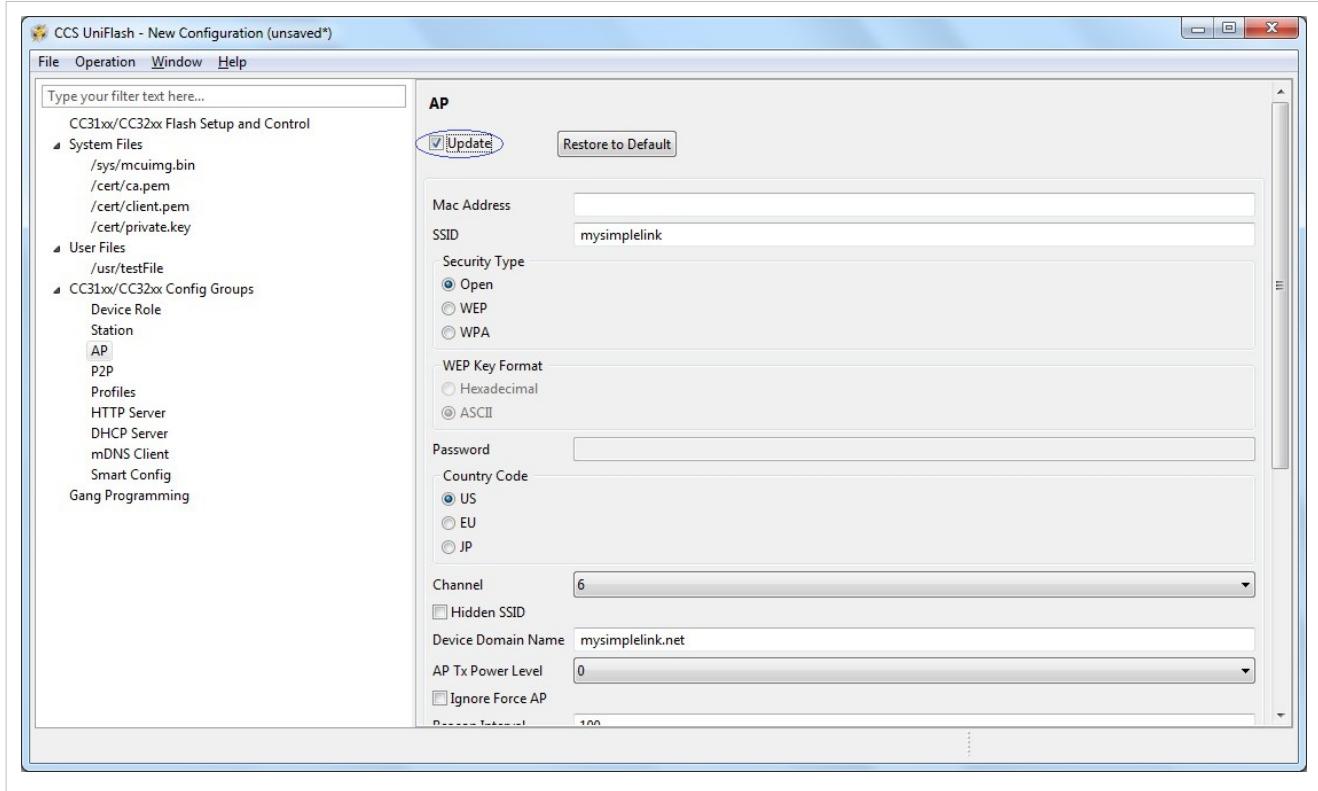


Configuration

Following is a screen shot of all **AP** 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**, i.e. 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**, i.e. 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.
- **Ap Network Application** is set to default where **HTTP Server**, **DHCP Server** and **mDNS Client** application are enabled.



P2P (Peer 2 Peer)

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.

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.

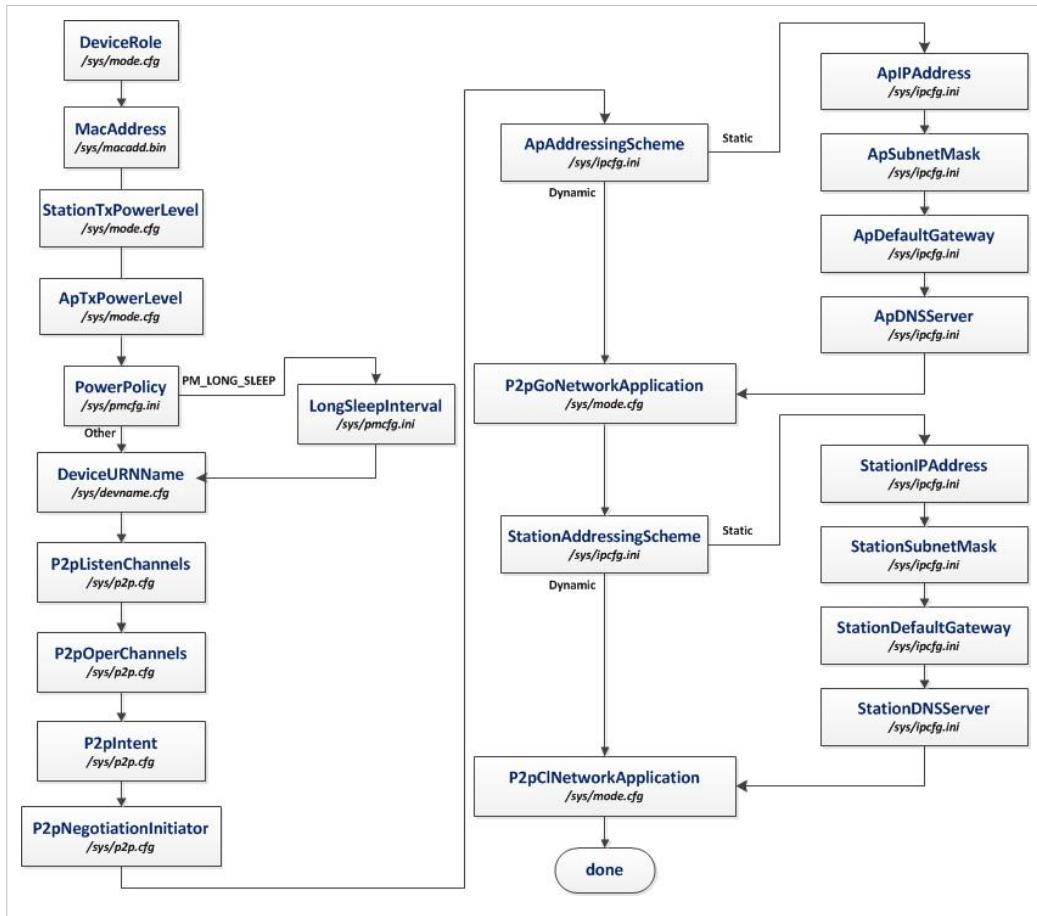
Generation of binary templates

Unlike programming which generates the binary templates and program 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**.

Flow

The following figure 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.

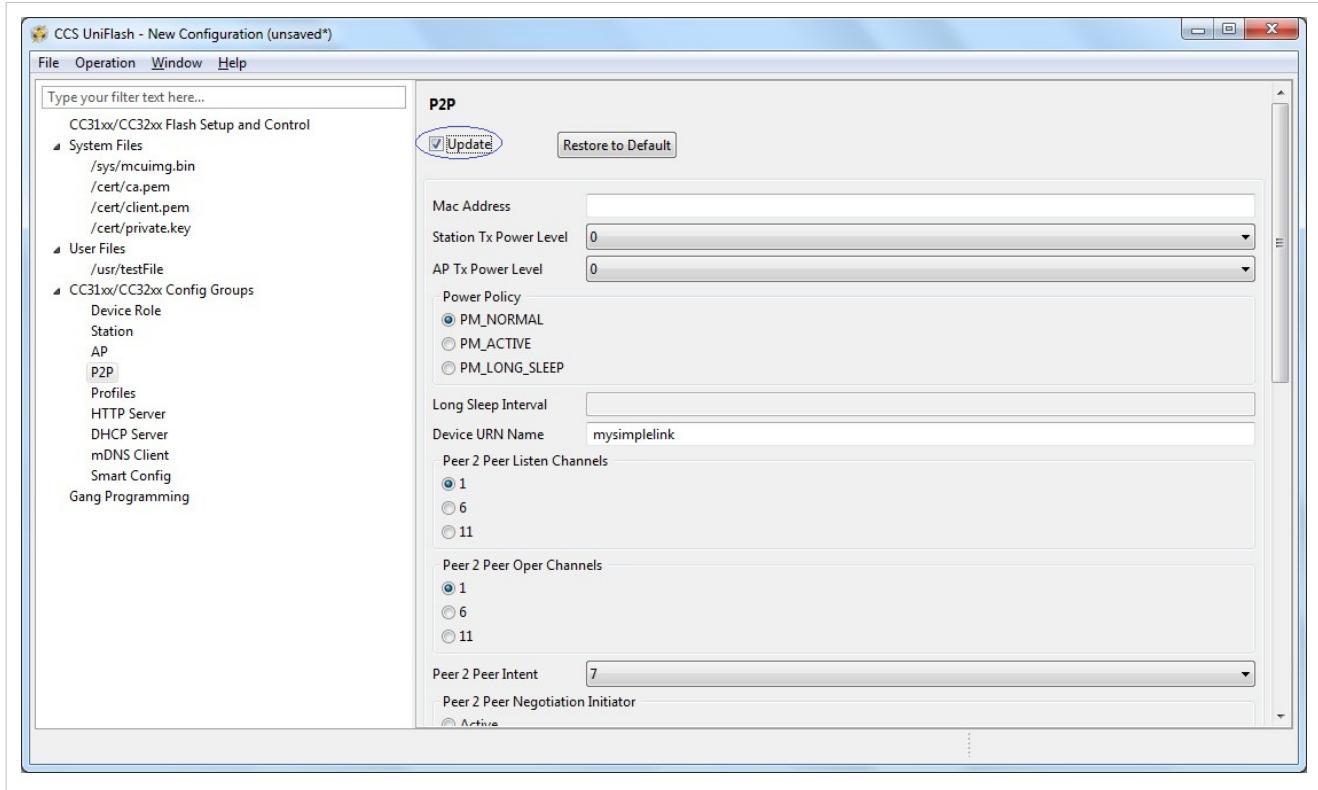


Configuration

Following is a screen shot of all **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**, i.e. transmit at maximum power.
- **AP Tx Power Level** is set to default **0**, i.e. 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**, i.e. 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**, i.e. 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.



HTTP Server

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.

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.

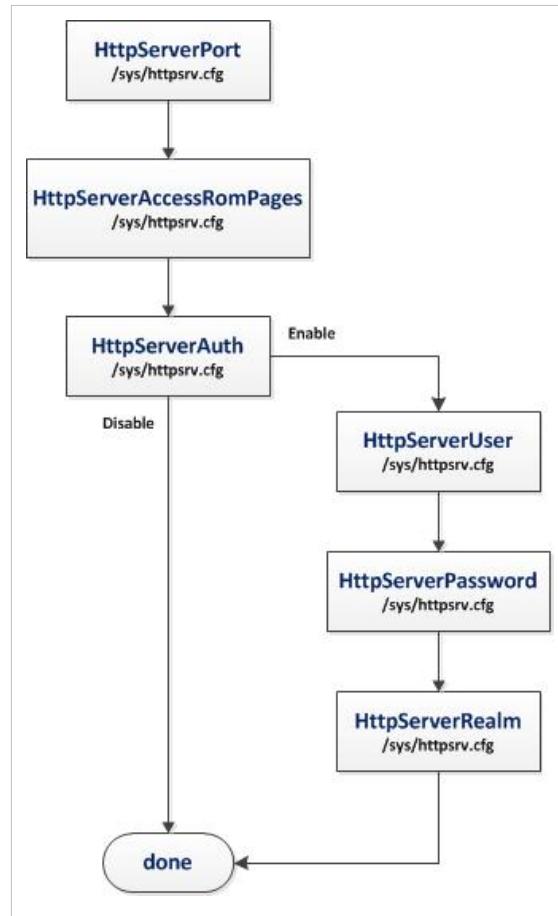
Generation of binary templates

Unlike programming which generates the binary templates and program 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**.

Flow

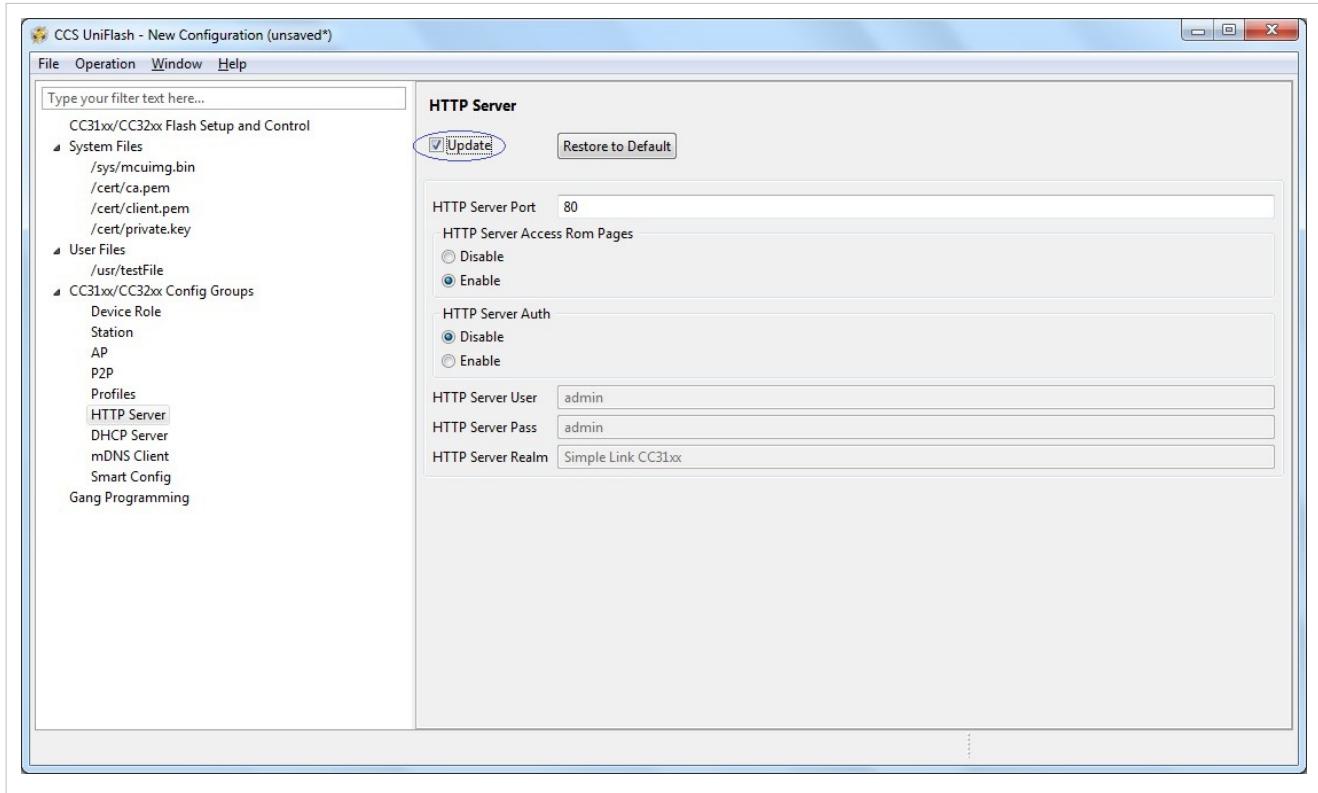
The following figure illustrates the parameters' flow during configuration. It is highly advised to follow this flow but it is not mandatory. The user may choose to go through configuration steps differently.



Configuration

Following is a screen shot 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**.



DHCP Server

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.

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.

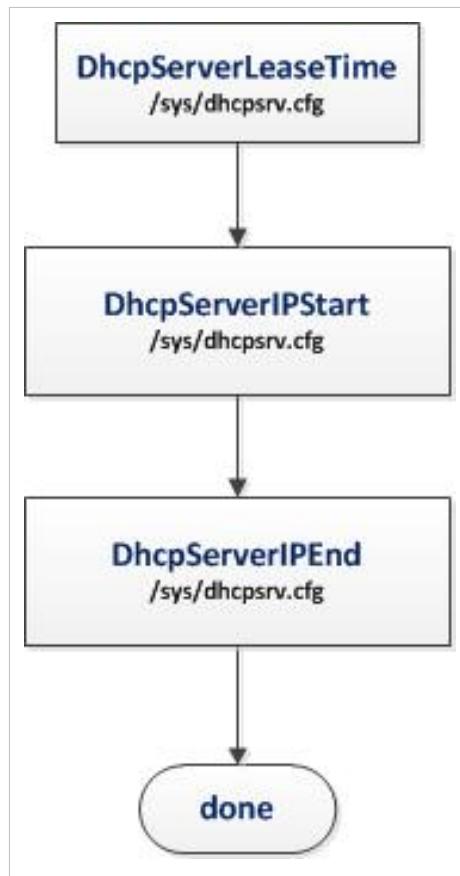
Generation of binary templates

Unlike programming which generates the binary templates and program 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**.

Flow

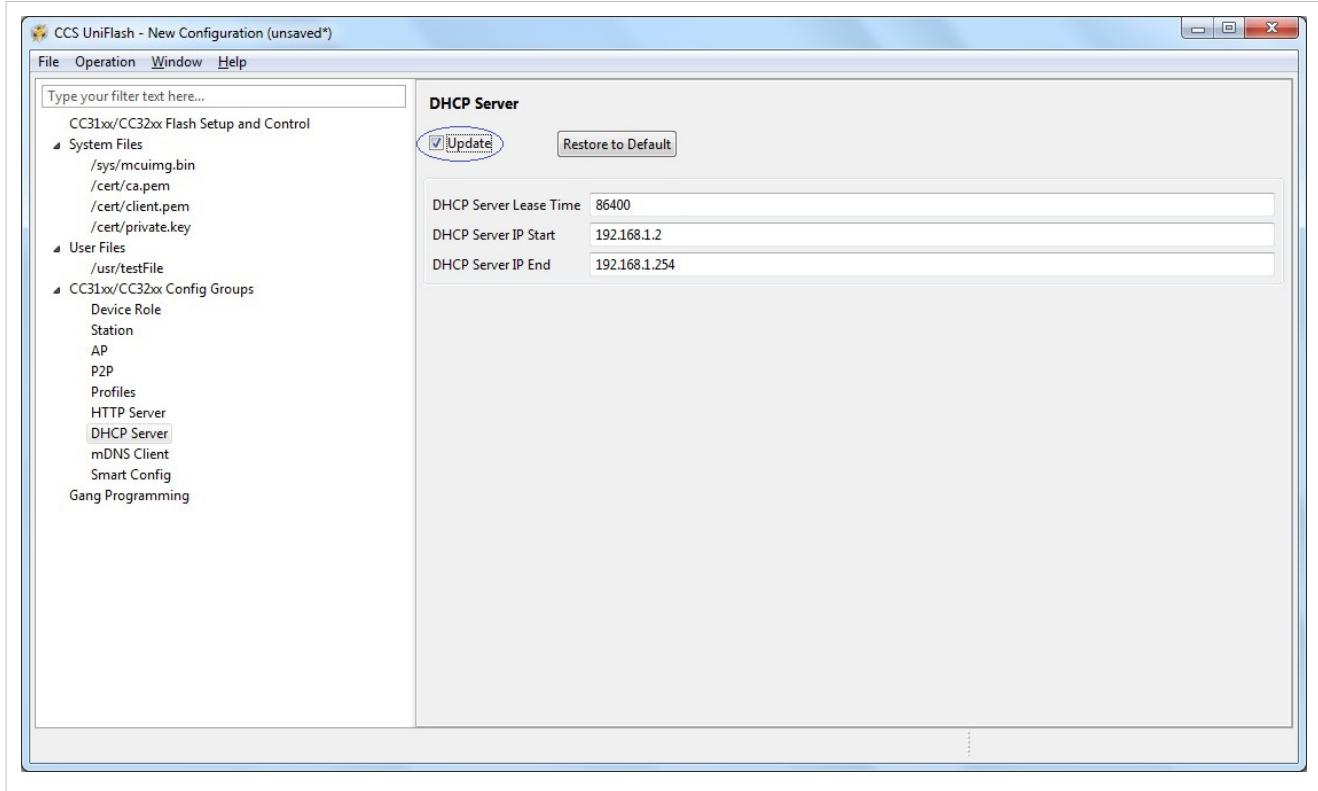
The following figure illustrates the parameters' flow during configuration. It is highly advised to follow this flow but it is not mandatory. The user may choose to go through configuration steps differently.



Configuration

Following is a screen shot 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.



mDNS Client

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.

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.

Generation of binary templates

Unlike programming which generates the binary templates and program 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).

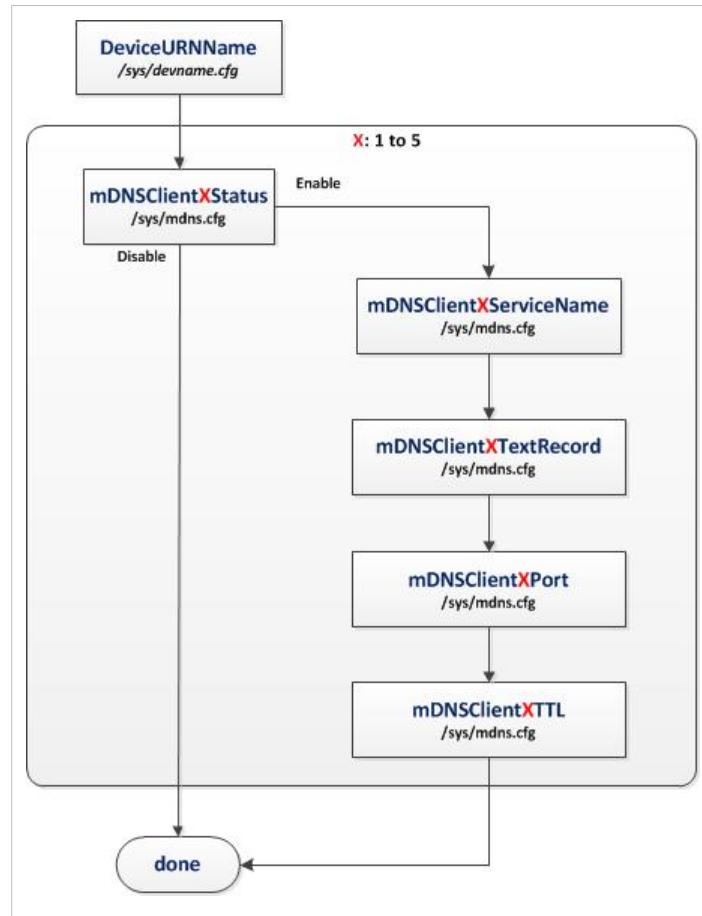
The configuration binaries can be generated via the top bar by clicking **Operation->Generate Config Binaries**.

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.

Flow

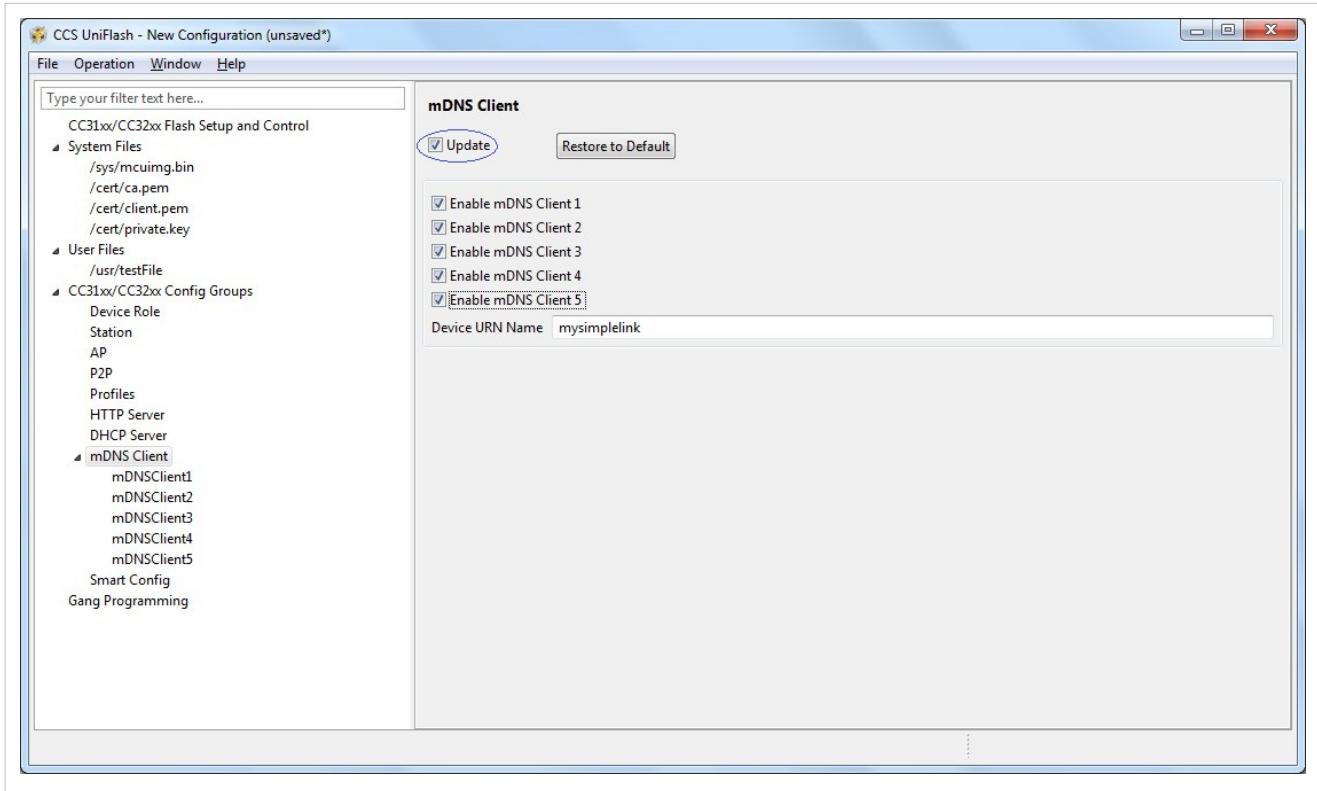
The following figure illustrates the parameters' flow during configuration. It is highly advised to follow this flow but it is not mandatory. The user may choose to go through configuration steps differently.



Configuration

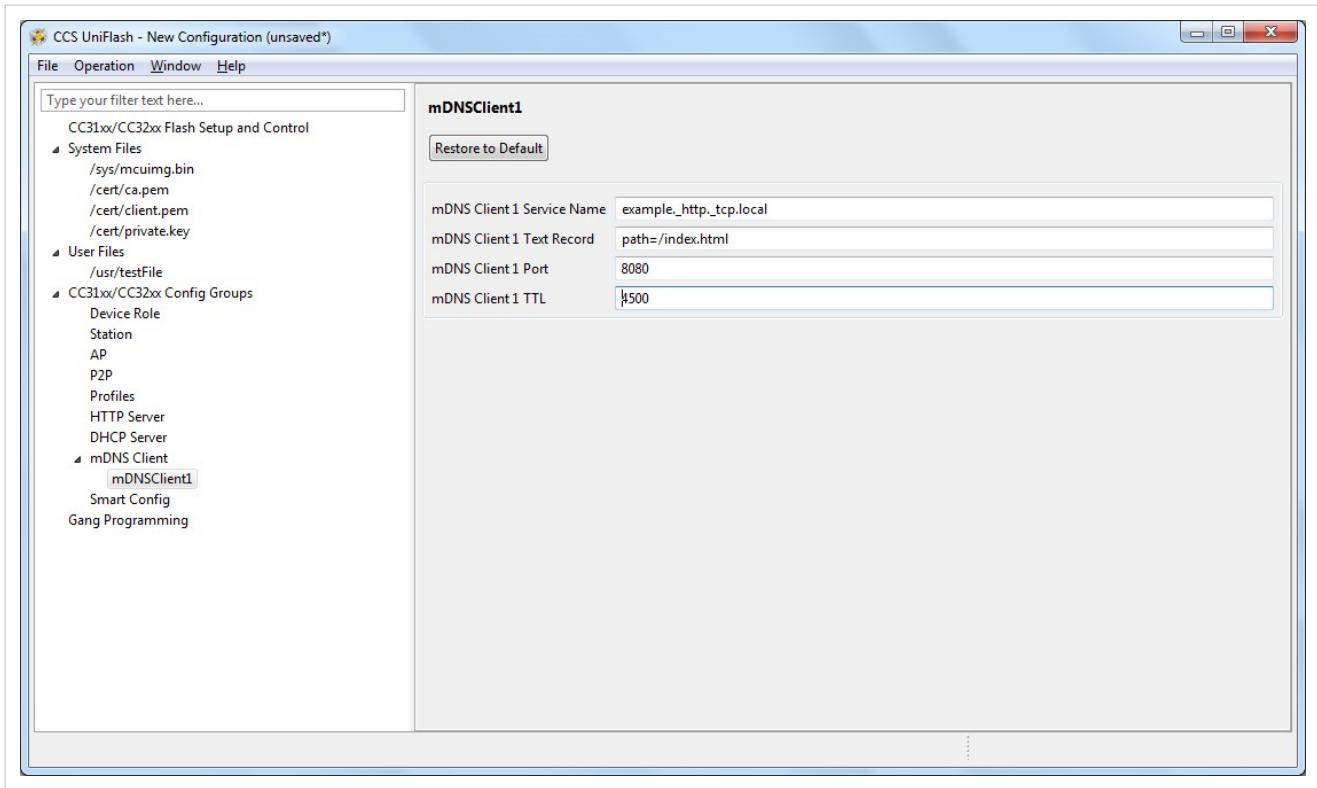
Following is a screen shot of all **mDNS Client** parameters. Some parameters have default values in case the user decides to skip configuration. However, mDNS services are mostly user specific and thus require user configuration. There are two screens relevant, the main screen and per service screen. 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**.



In the per 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**.



Profiles

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.

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.

Generation of binary templates

Unlike programming which generates the binary templates and program 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).

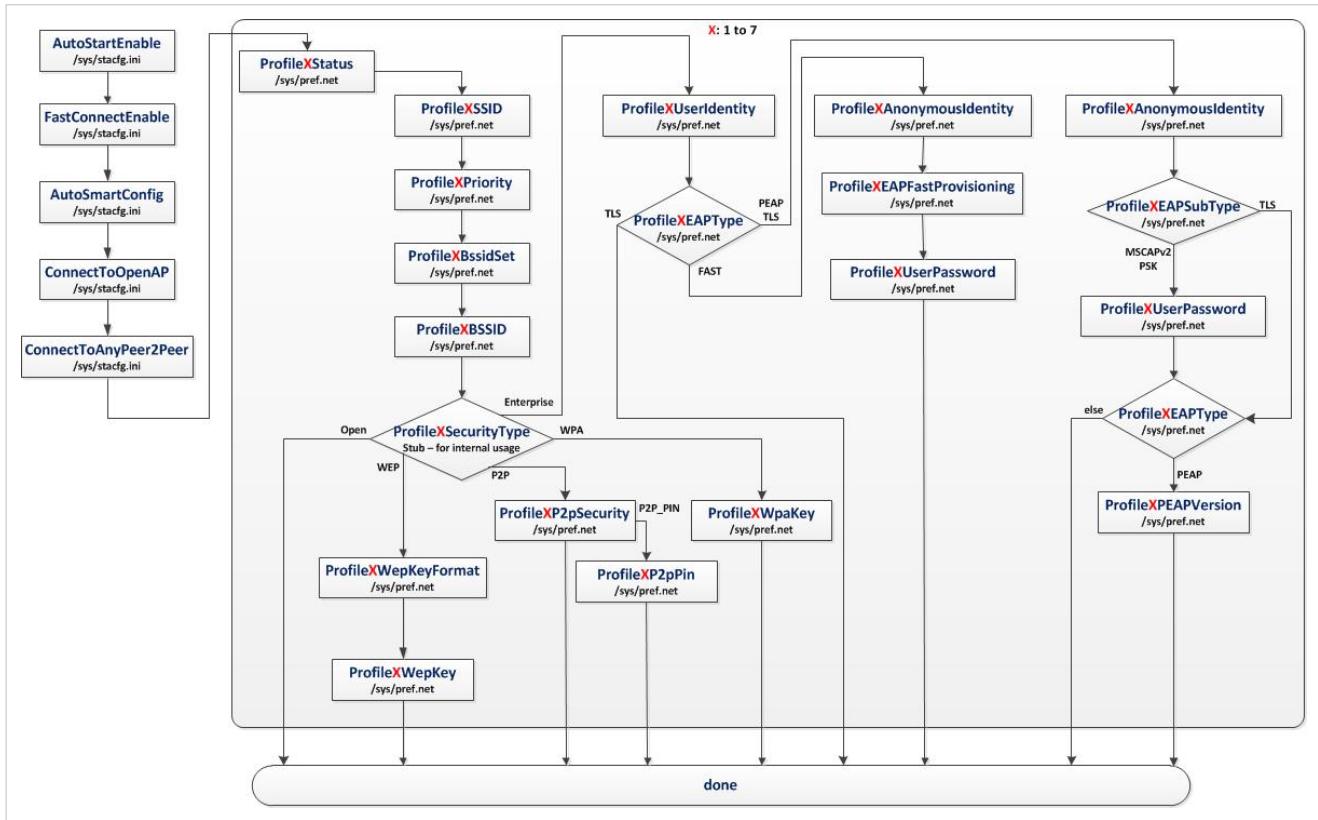
The configuration binaries can be generated via the top bar by clicking **Operation->Generate Config Binaries**.

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.

Flow

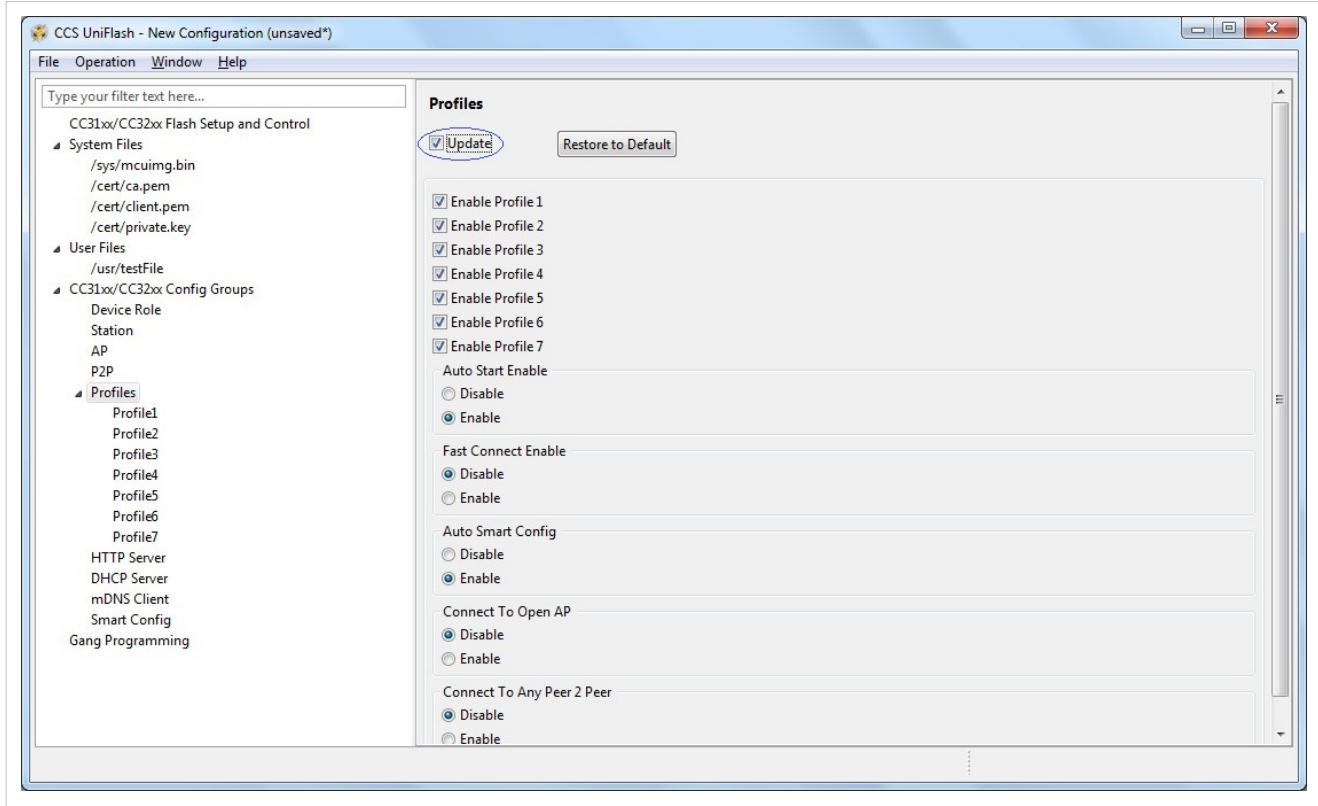
The following figure illustrates the parameters' flow during configuration. It is highly advised to follow this flow but it is not mandatory. The user may choose to go through configuration steps differently.



Configuration

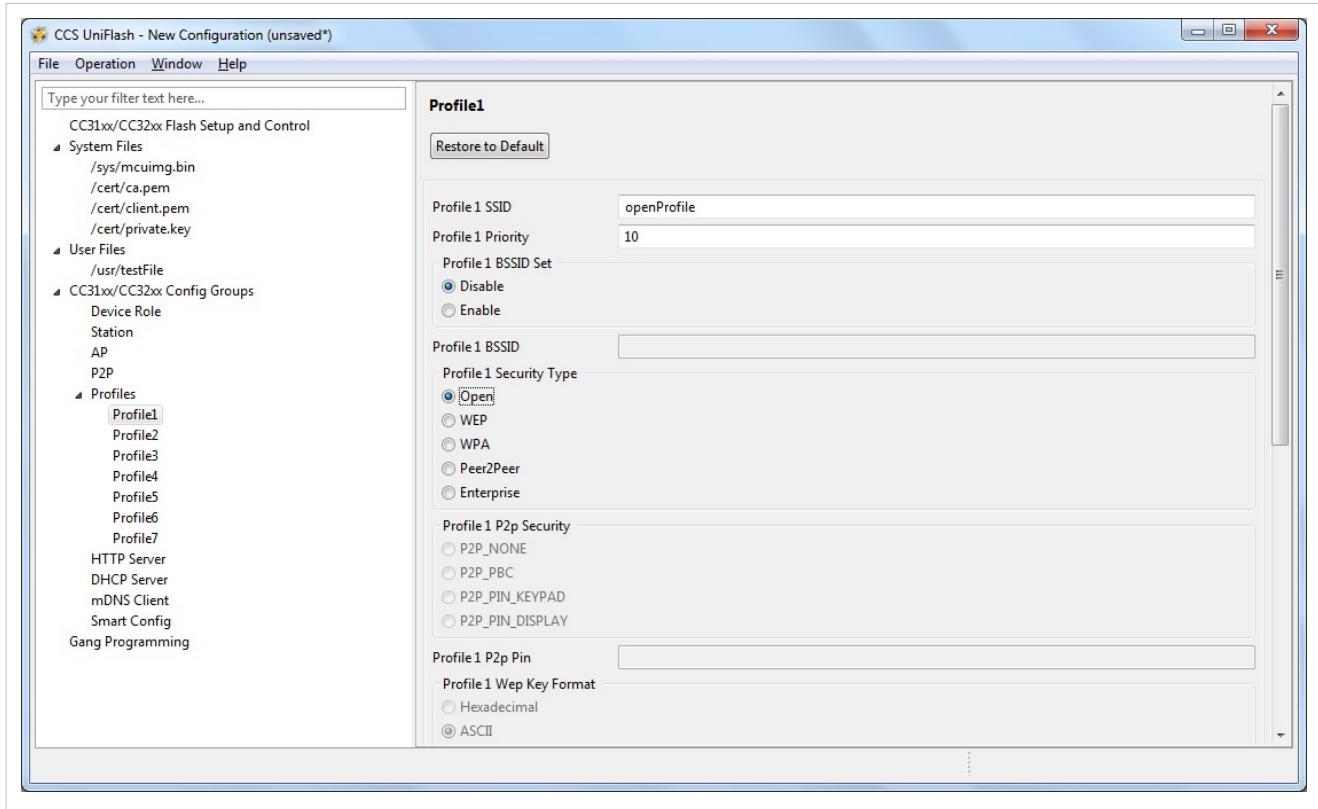
Following is a screen shot of all **Profiles** parameters. Some parameters have default values in case the user decides to skip configuration. However, profiles are mostly user specific and thus require user configuration. There are two screens relevant, the main screen and per profile screen. In the 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**.



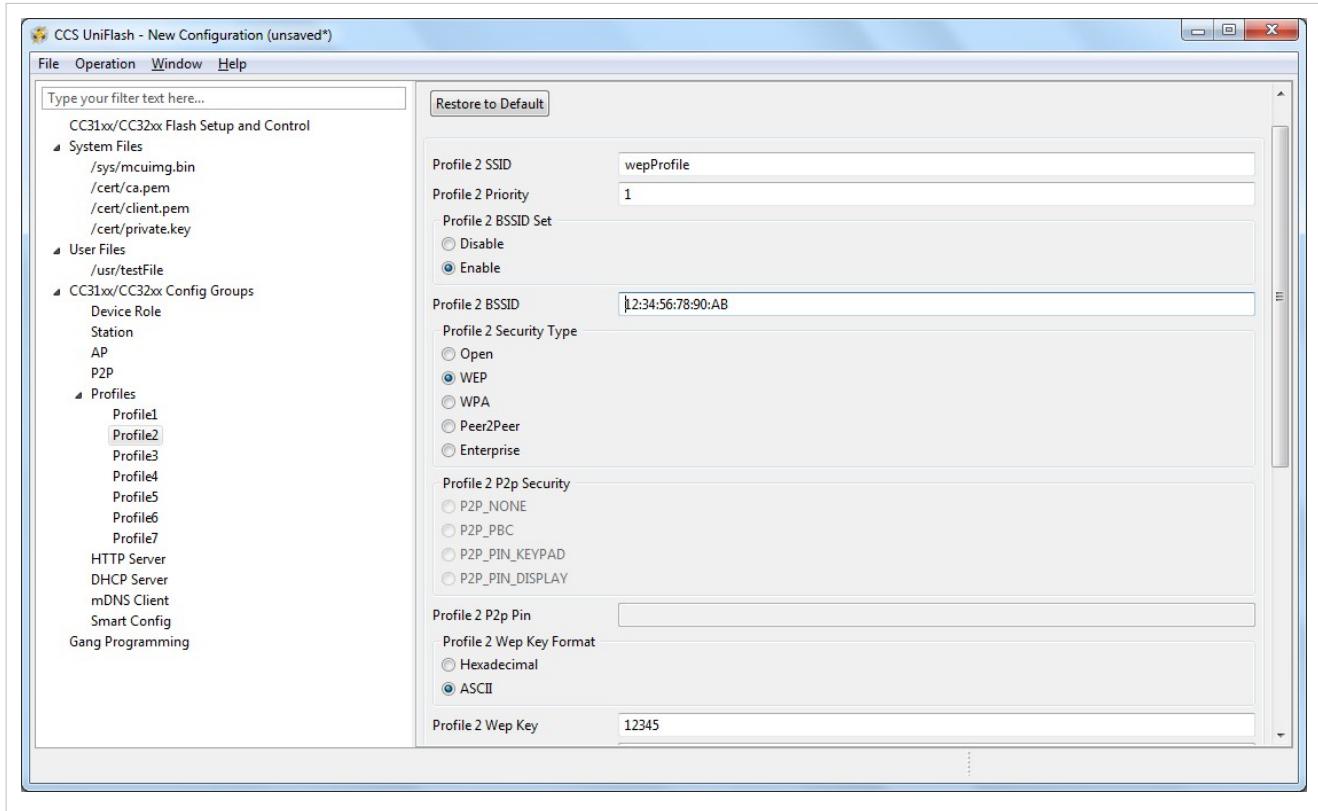
In the per profile screen, several example profiles are described. For Open security profile, the following parameters are defined:

- **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.



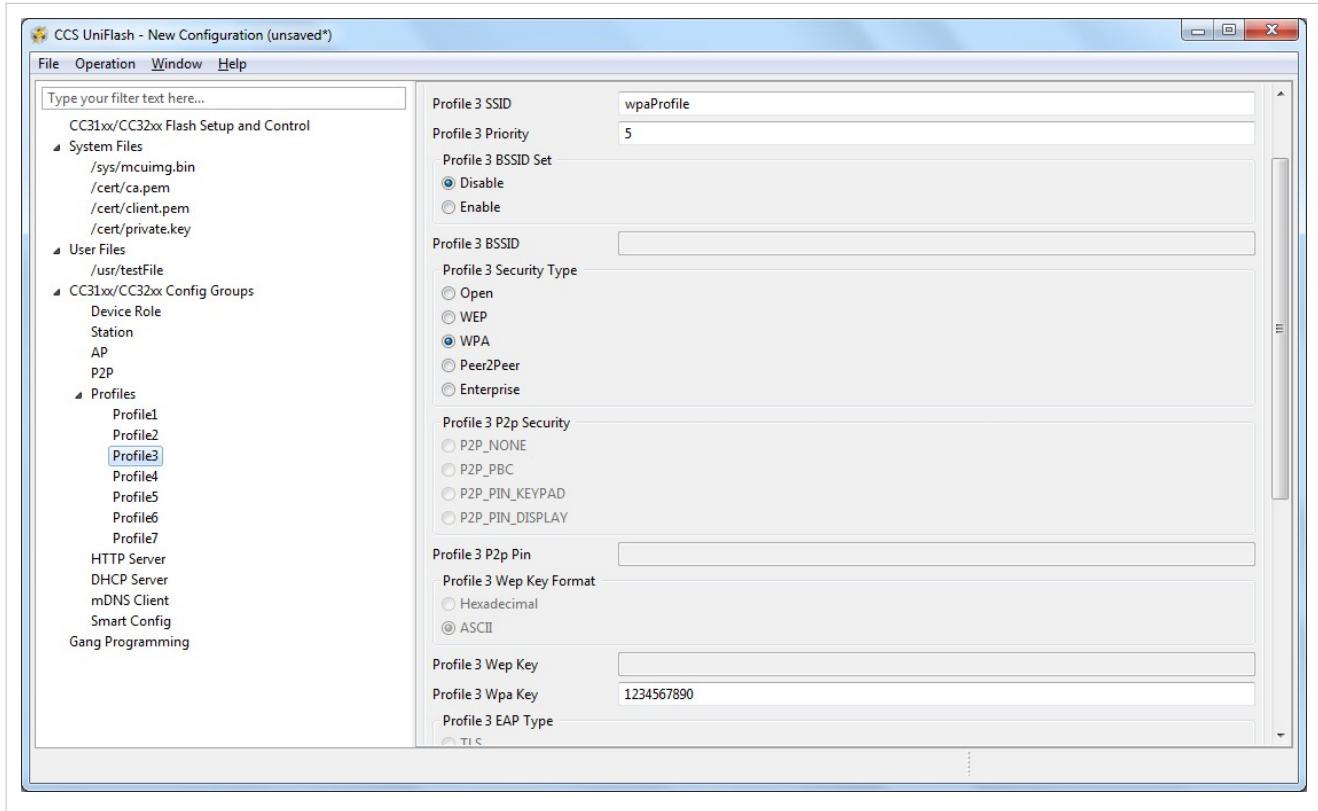
For WEP security profile, the following parameters are defined:

- **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.



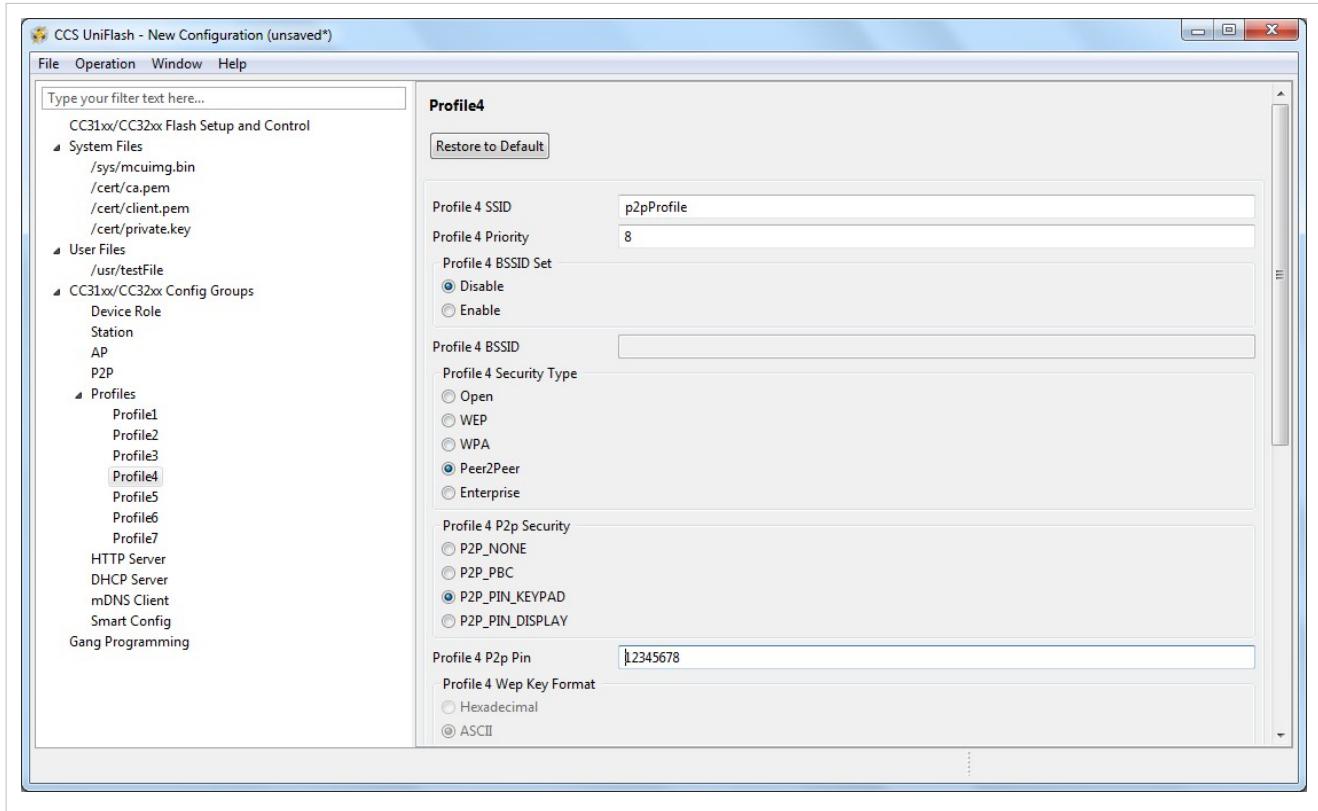
For WPA security profile, the following parameters are defined:

- **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.



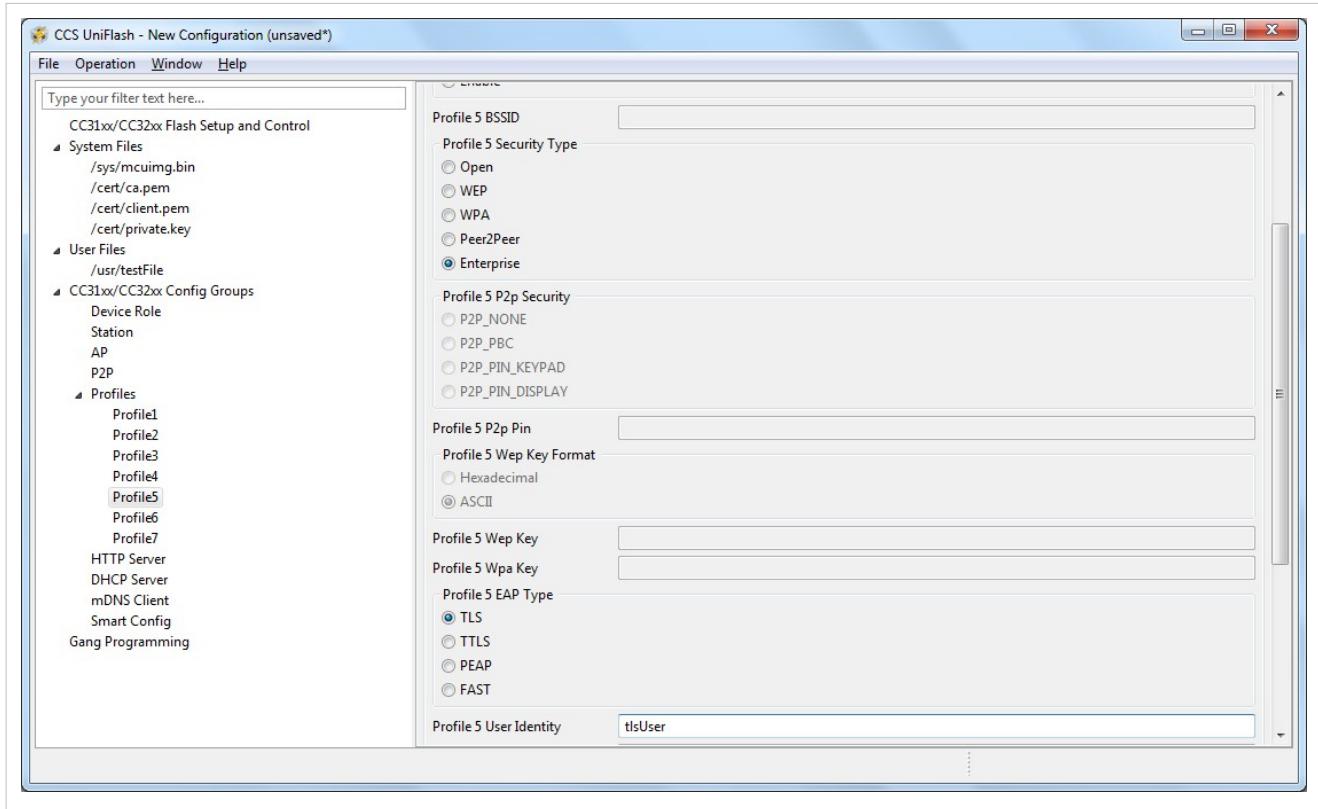
For Peer 2 Peer security profile, the following parameters are defined:

- **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.



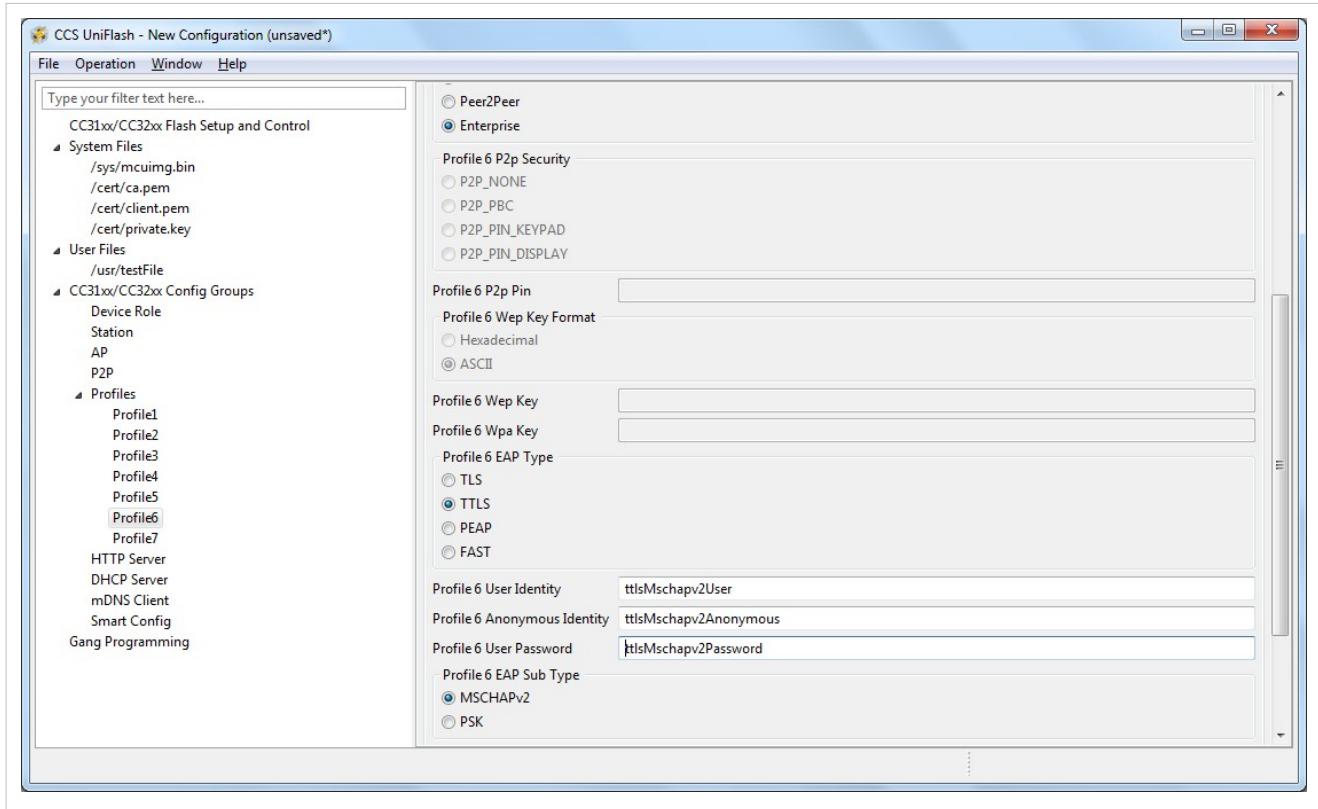
For Enterprise security with TLS profile, the following parameters are defined:

- **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.



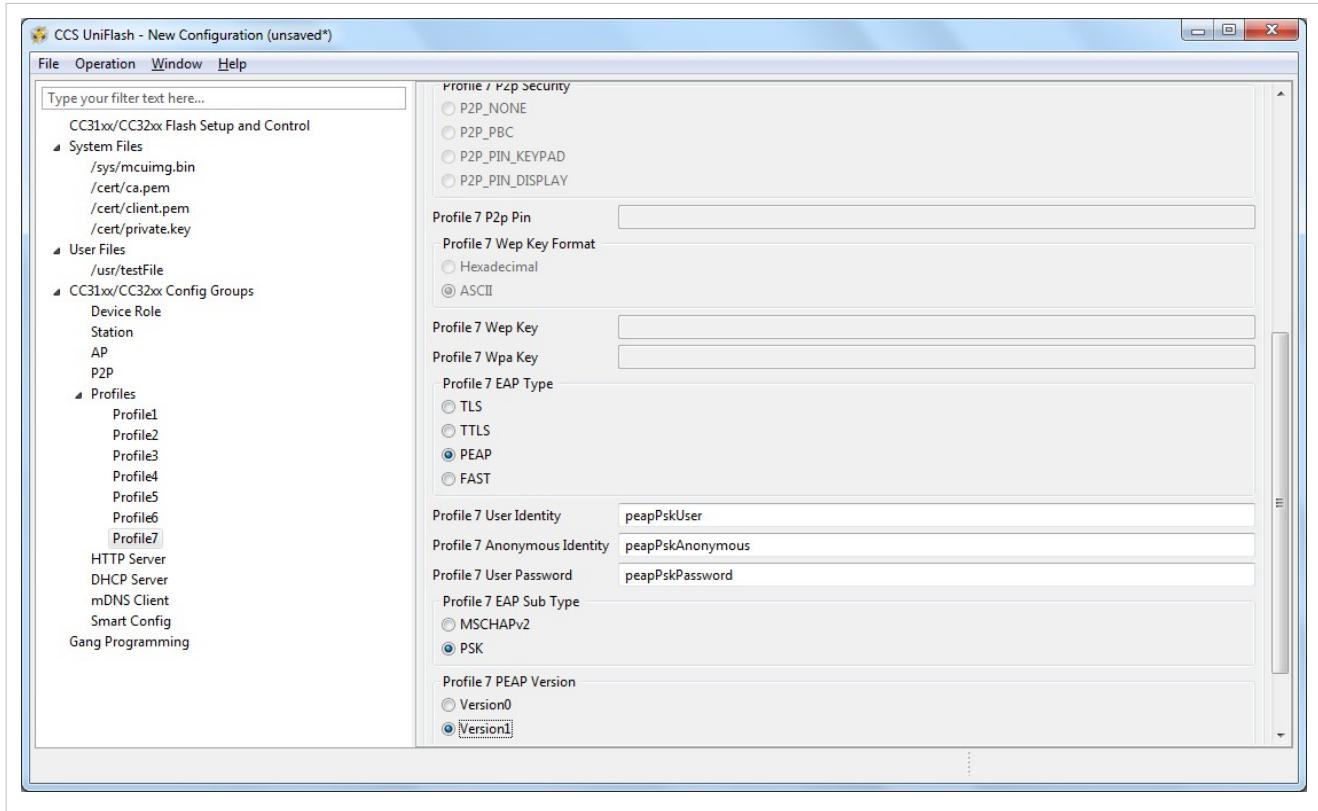
For Enterprise security with TTLS/Mschapv2 profile, the following parameters are defined:

- **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 rest of parameters is not relevant for a TTLS/Mschapv2 Enterprise security profile.



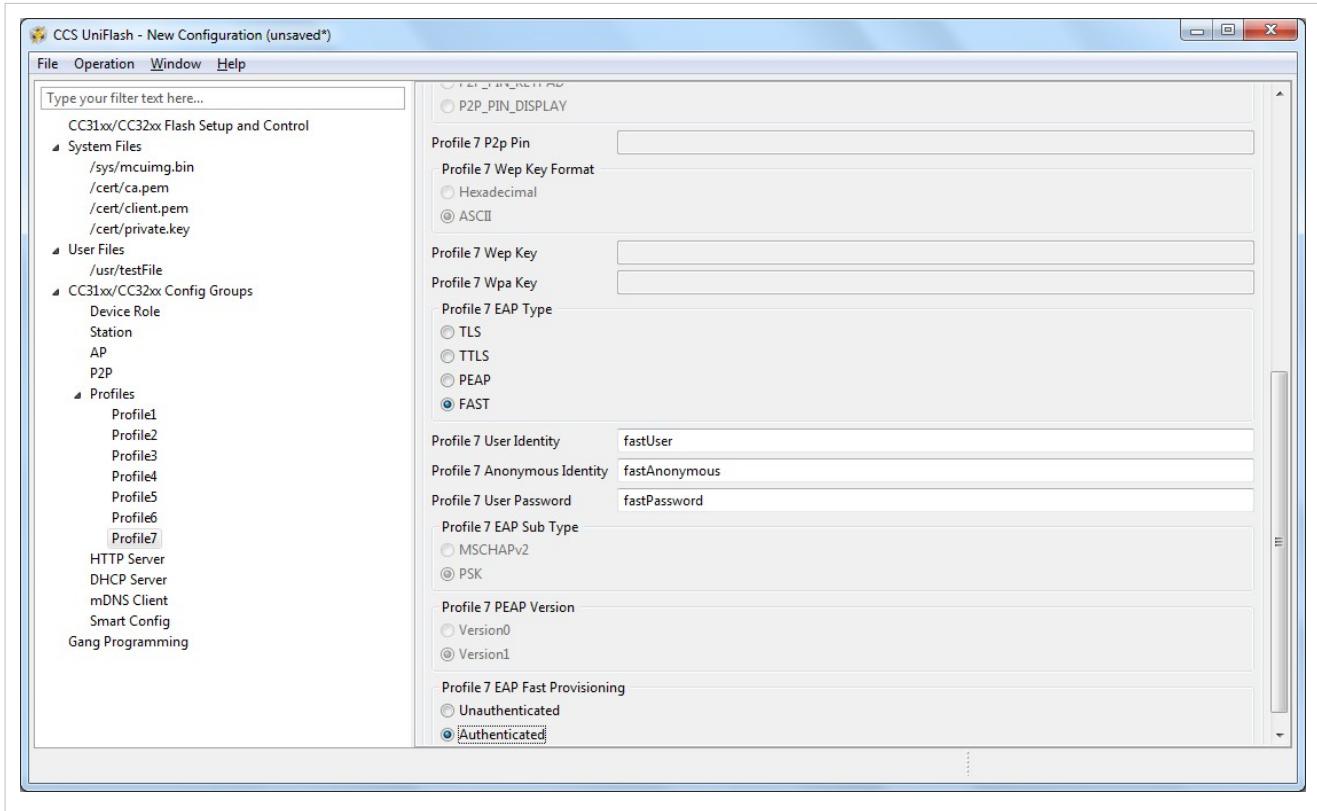
For Enterprise security with PEAP/PSK profile, the following parameters are defined:

- **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 rest of parameters is not relevant for a PEAP/PSK Enterprise security profile.



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 rest of parameters is not relevant for an EAP_FAST Enterprise security profile.



Smart Config

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.

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.

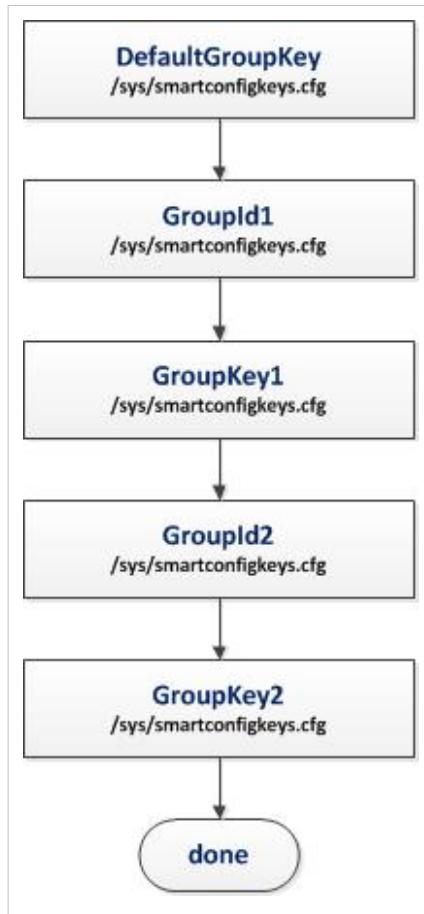
Generation of binary templates

Unlike programming which generates the binary templates and program 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**.

Flow

The following figure 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.



Configuration

Following is a screen shot of all **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.

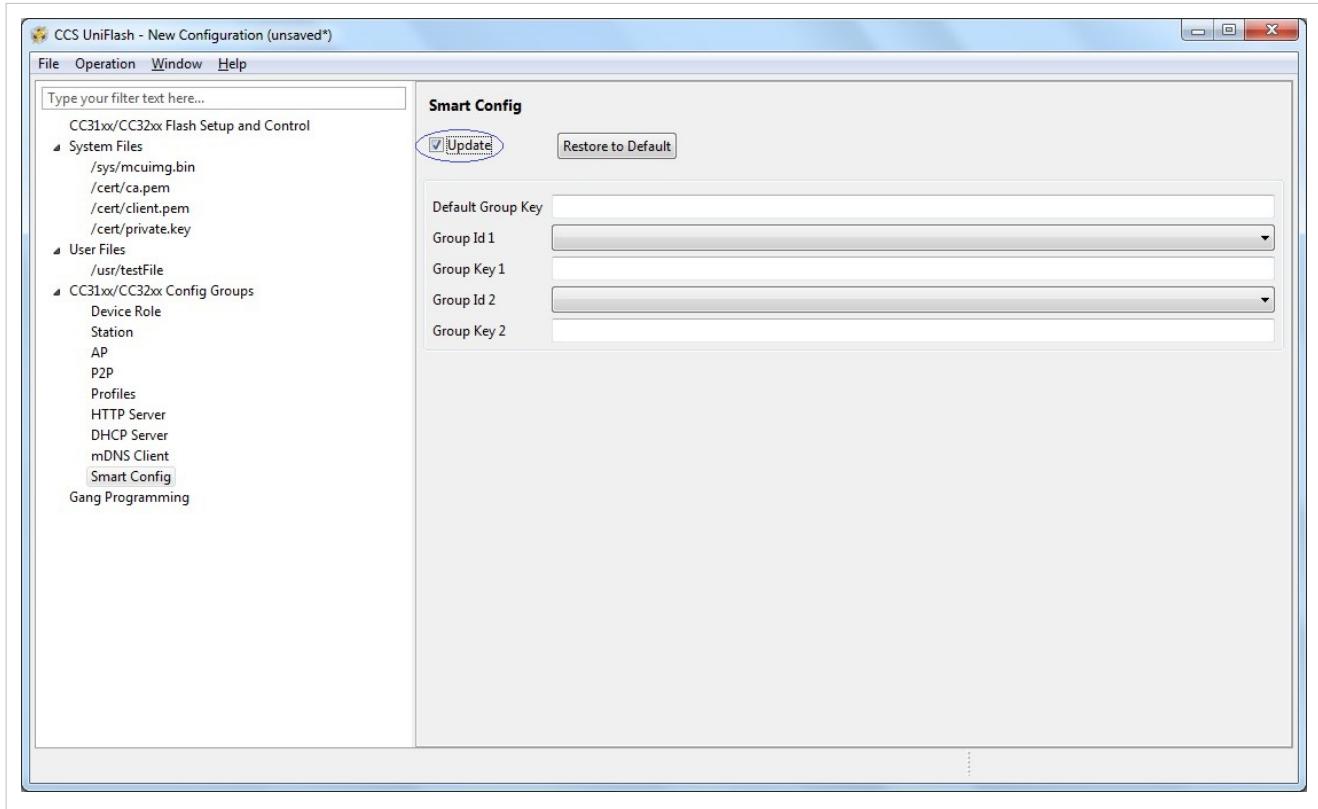


Image Creation and Programming

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 may either get programmed using the Uniflash (and thus via the UART lines) or using any off-the-shelf SPI programmers directly via the serial flash SPI lines.

Basic concept and operation

The image stored on the serial flash is extracted by the SimpleLinkTM 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 SimpleLinkTM 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 (i.e. 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
 1. Program the serial flash devices with the image using any off-the-shelf SPI programmer before assembling them on the target boards
 2. Assemble the programmed serial flash devices on the target boards
 3. If the serial flash is already assembled on the target board

1. 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 SimpleLinkTM 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 will have to replace the serial flash or fix the image and retry.

Image Creation – supported commands

Overview

This chapter 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.

General Settings

This is not a command but a general settings.

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

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 allows 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.

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

Write Service Pack command

This command writes the service pack of the networking sub system.

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

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

Write File command

The command creates and writes a file.

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

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

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

Attribute	Max File Size
Description	Maximum size of the file. 0 indicates the original file size
Options	Text box
Default	0
Dependencies	None

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

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.

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

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

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

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

Image Creation – GUI interface

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, i.e. 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.

Image Programming buttons

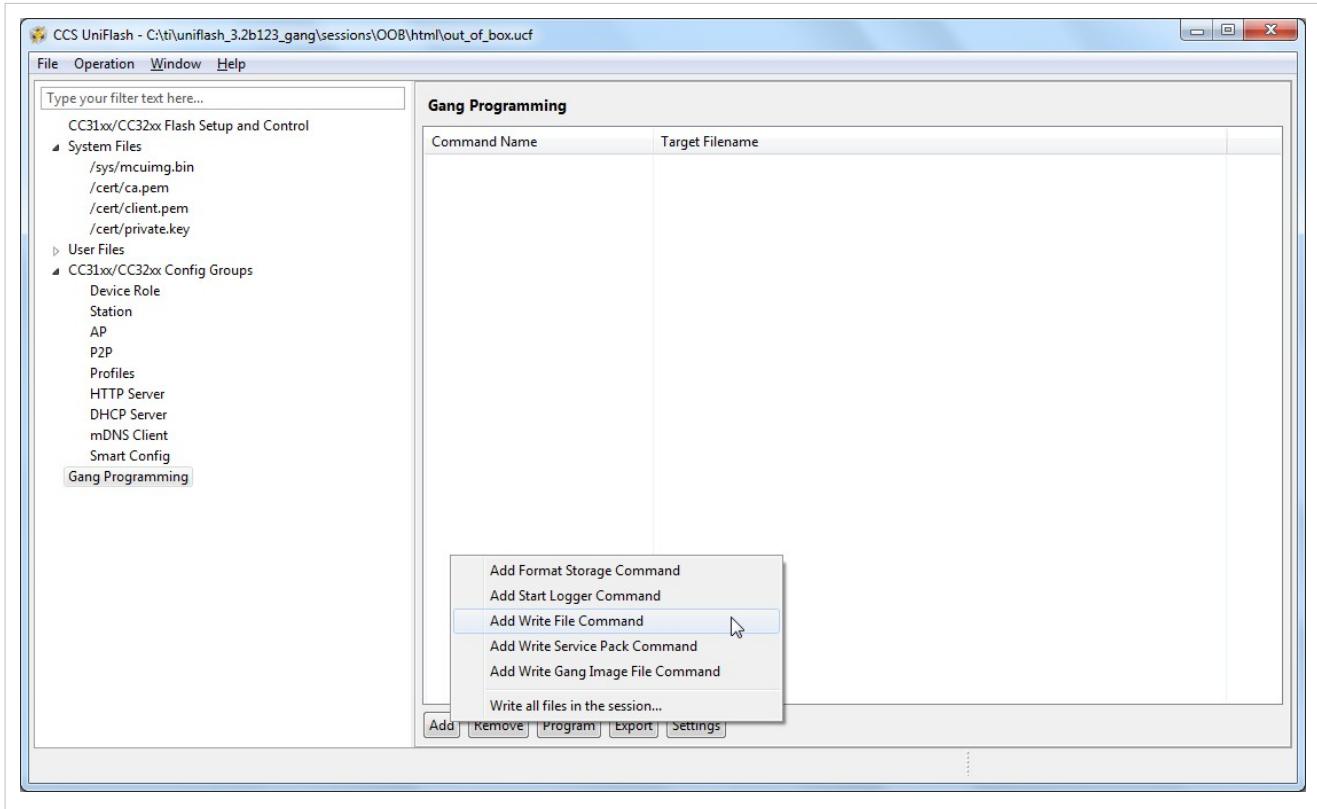
Note: Tooltip support. Hovering over the buttons pops a short description window.

Add

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**.

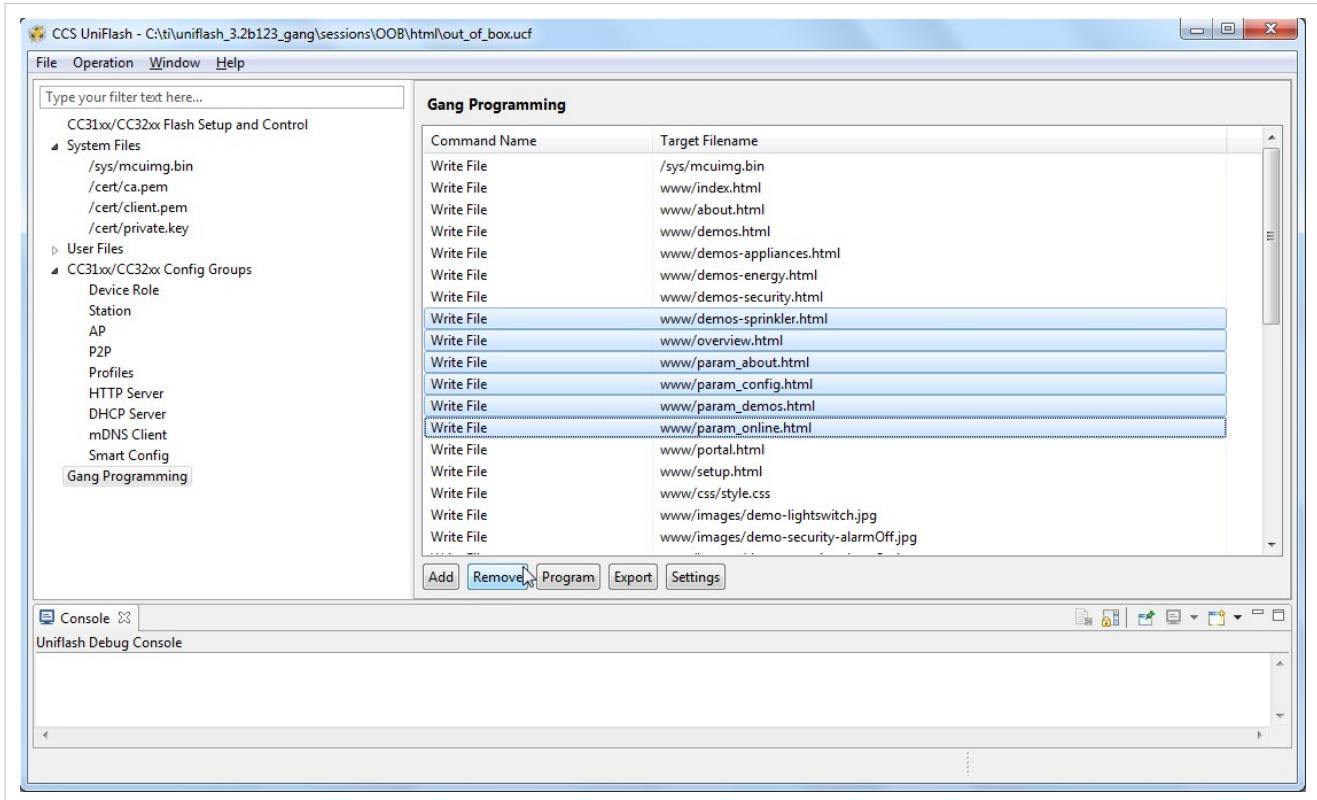
All options are shown in the following figure.



Remove

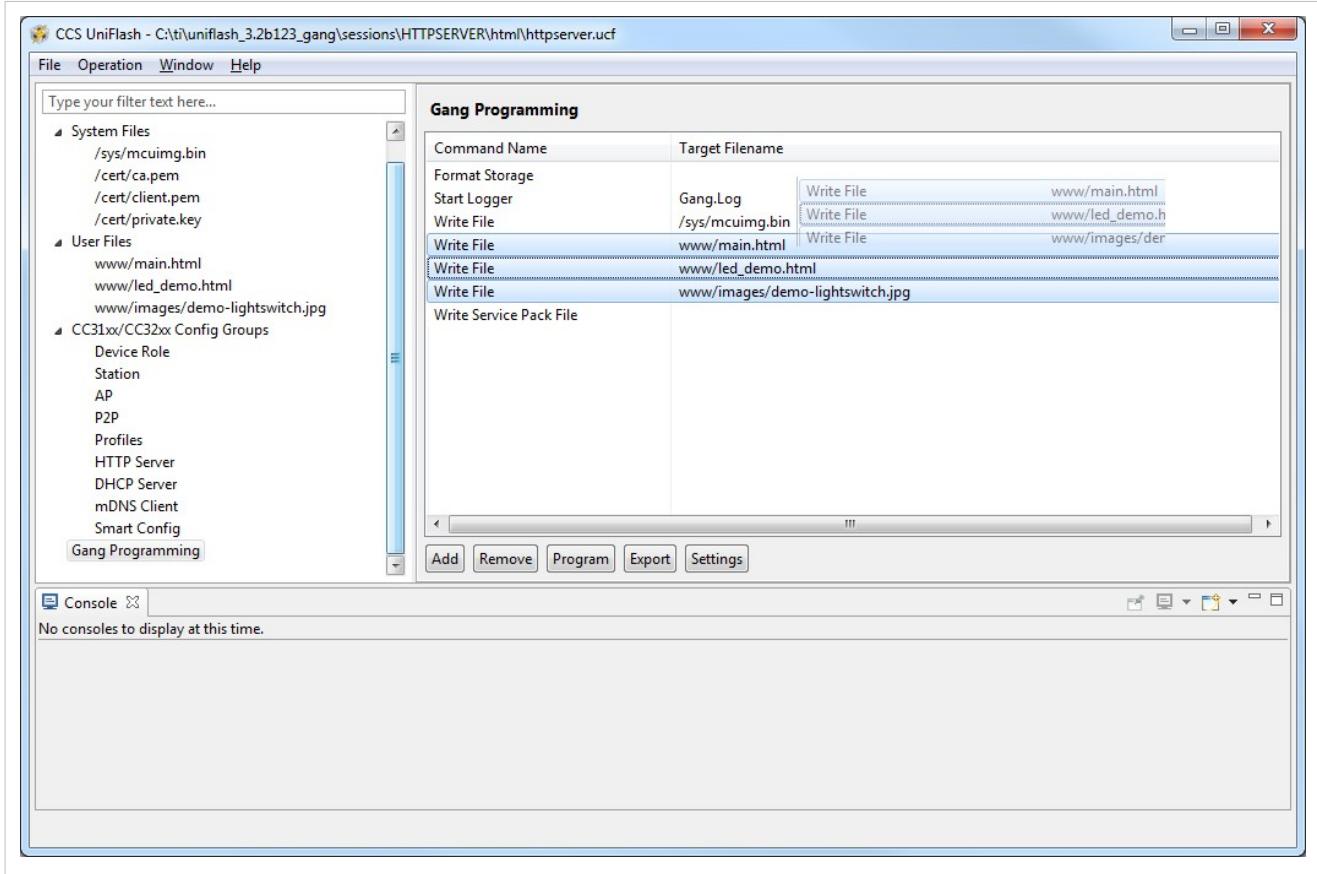
Remove button enables the user to remove any command from the image. Also multiple commands can be removed using multiple selections as shown in the following figure.

Also note it is possible to remove commands by selecting items and pressing the *delete* button on the keyboard.



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.

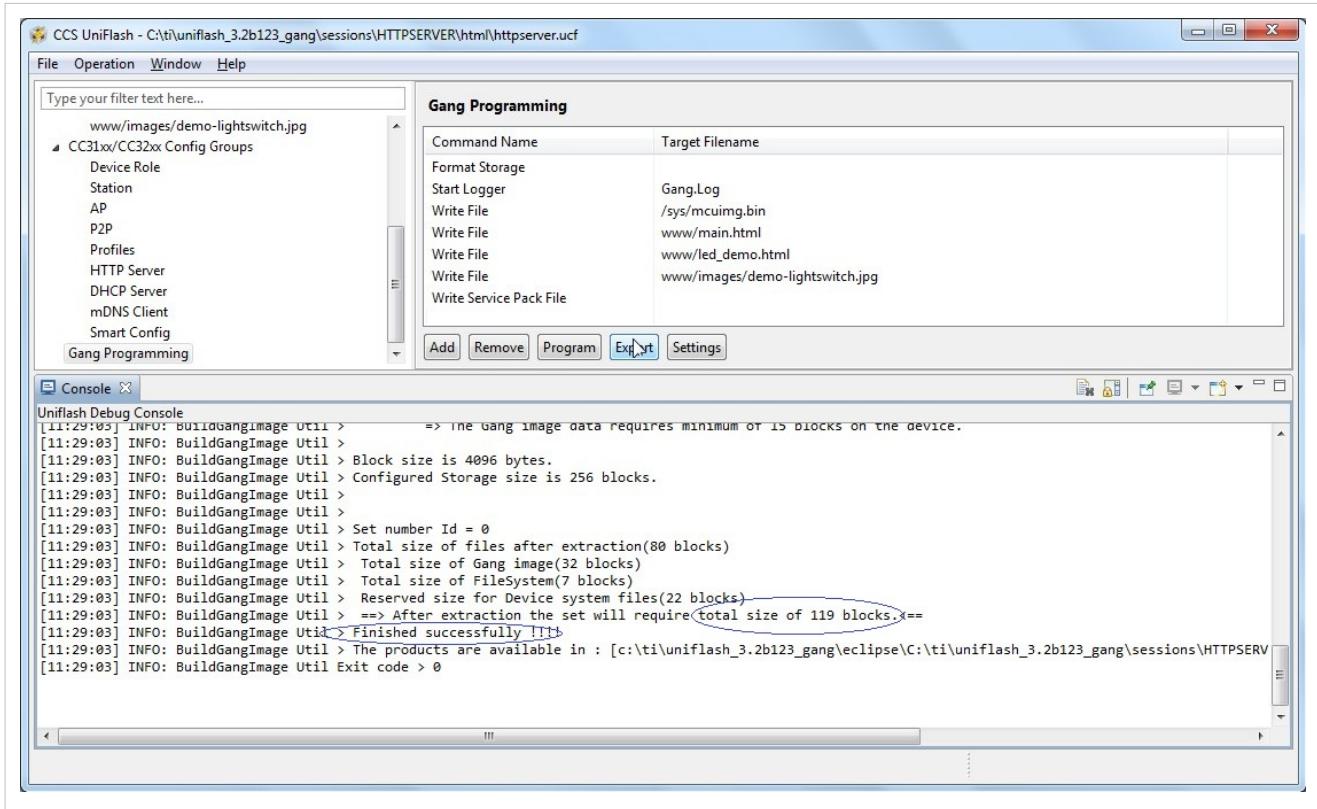


Program

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

Export

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.



Settings

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

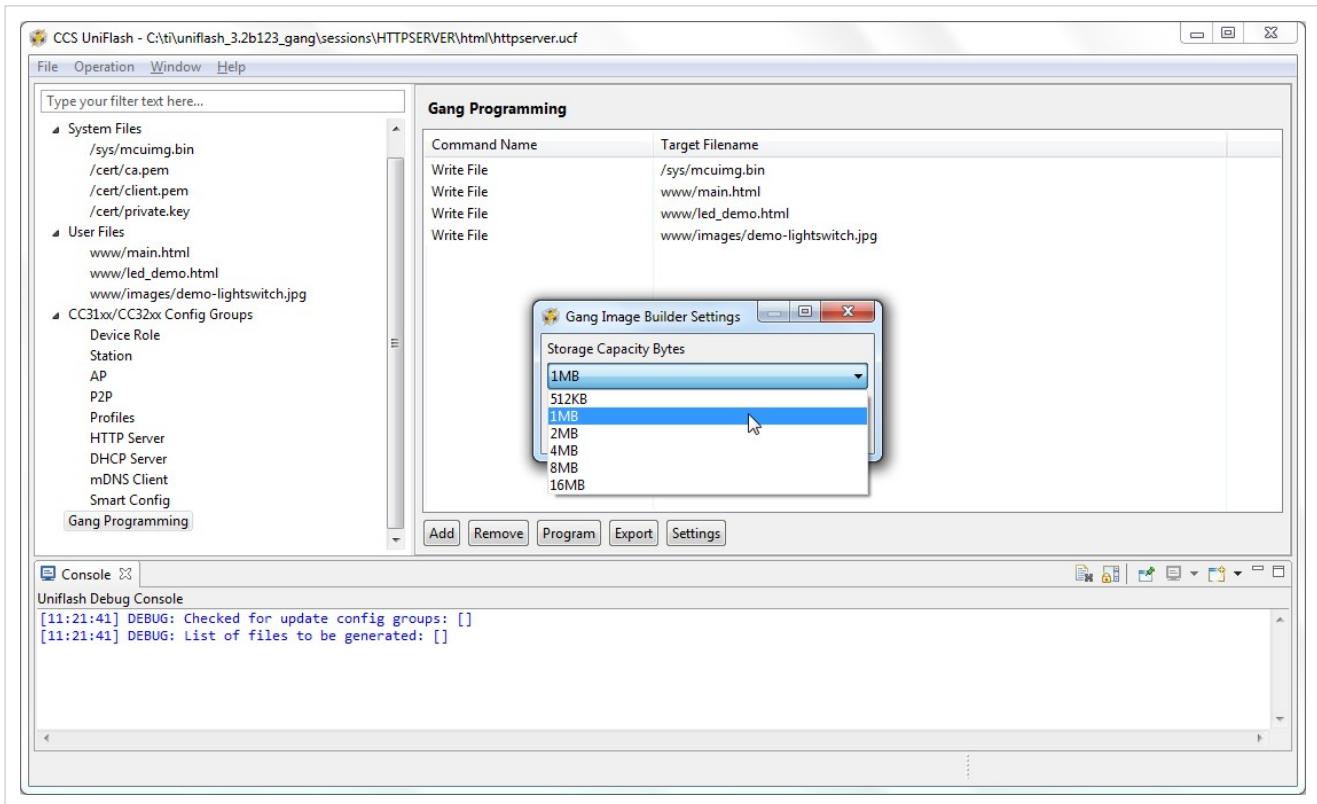
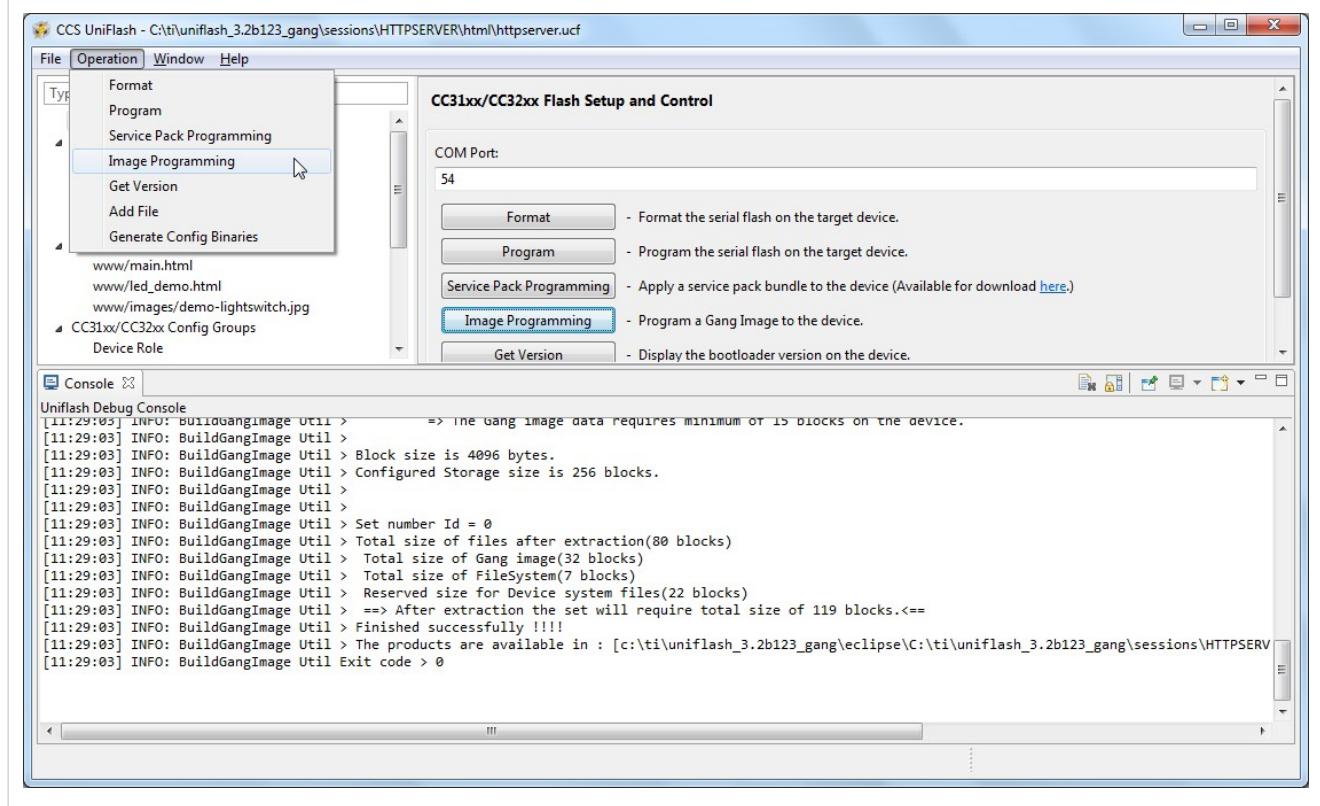


Image Programming to target

Image programming of the device is applied upon clicking the 'Image Programming' 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.

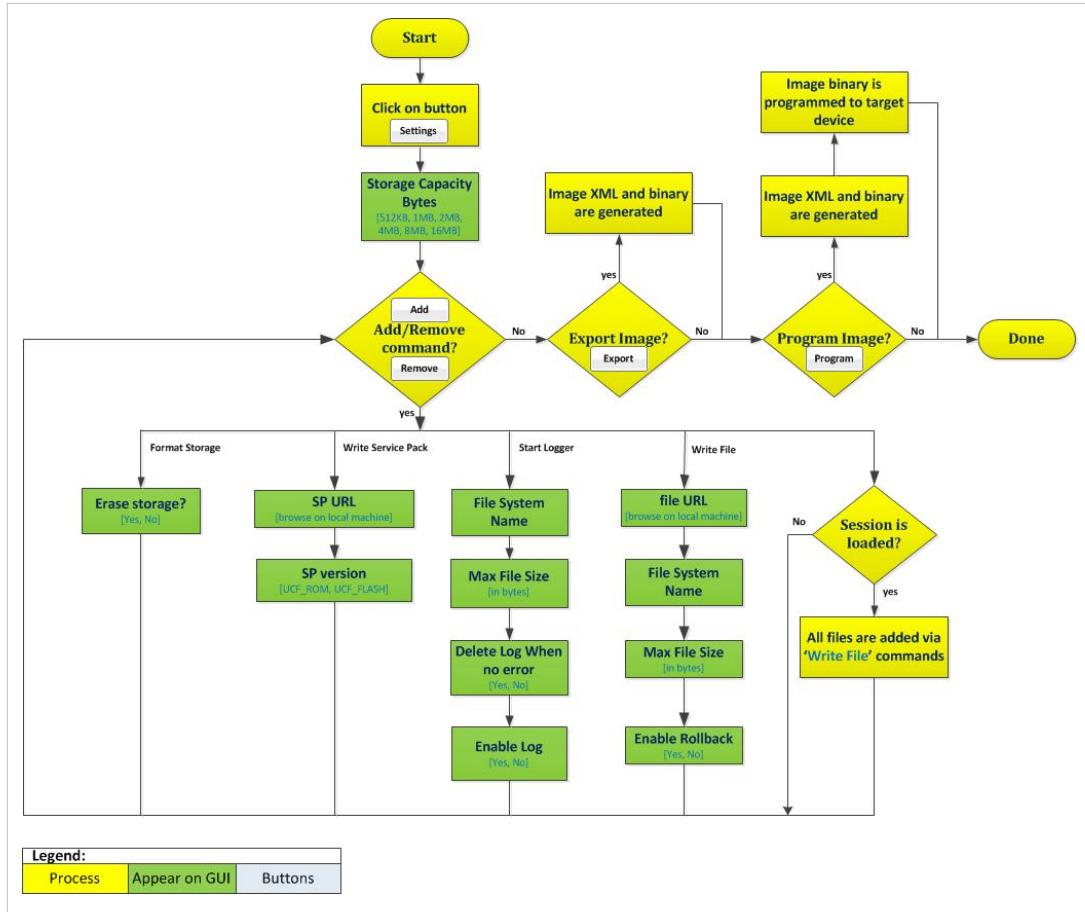


How to build an image guide

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

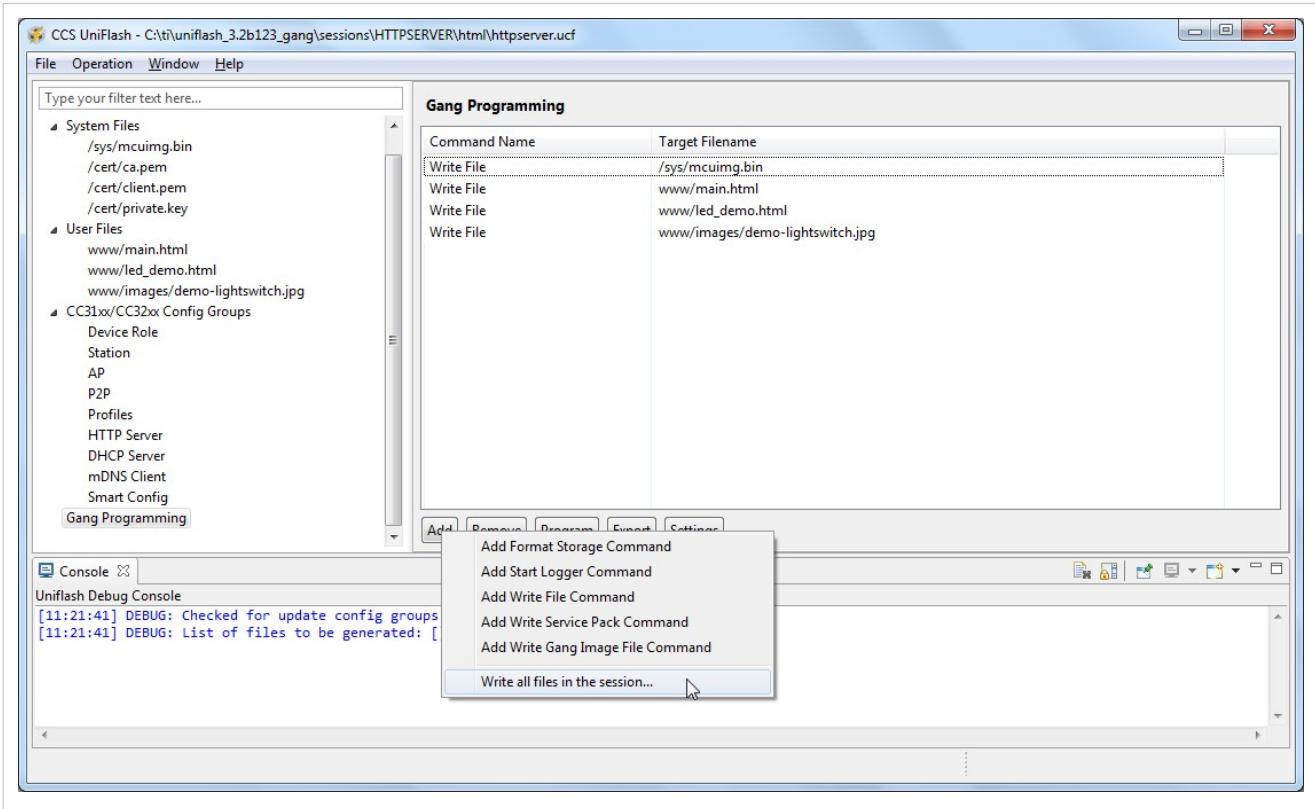
- Capacity of the serial flash
- Location of all user files
- Desired setup of all configuration files
- Latest Service Pack binary file

Following is a figure describing the flow a user must follow in order to generate an image.

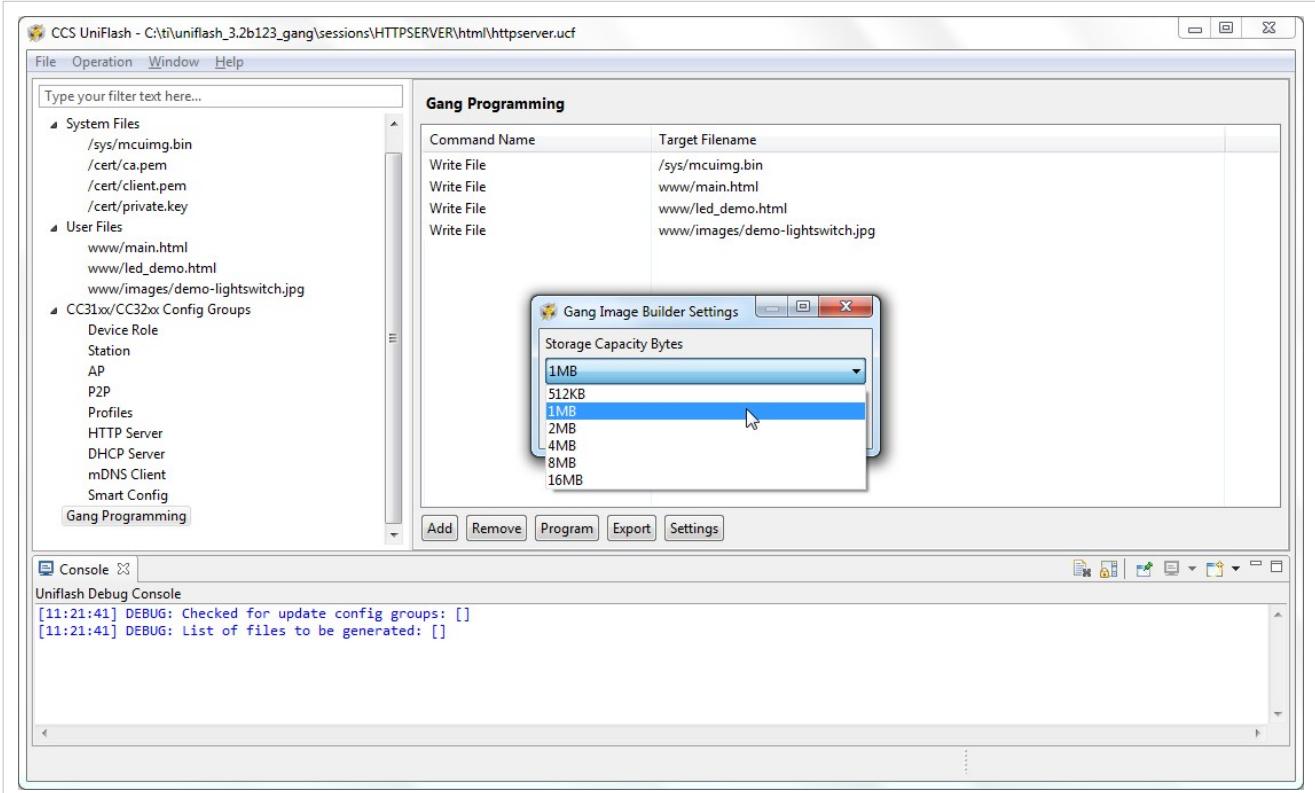


The recommended steps to build the image:

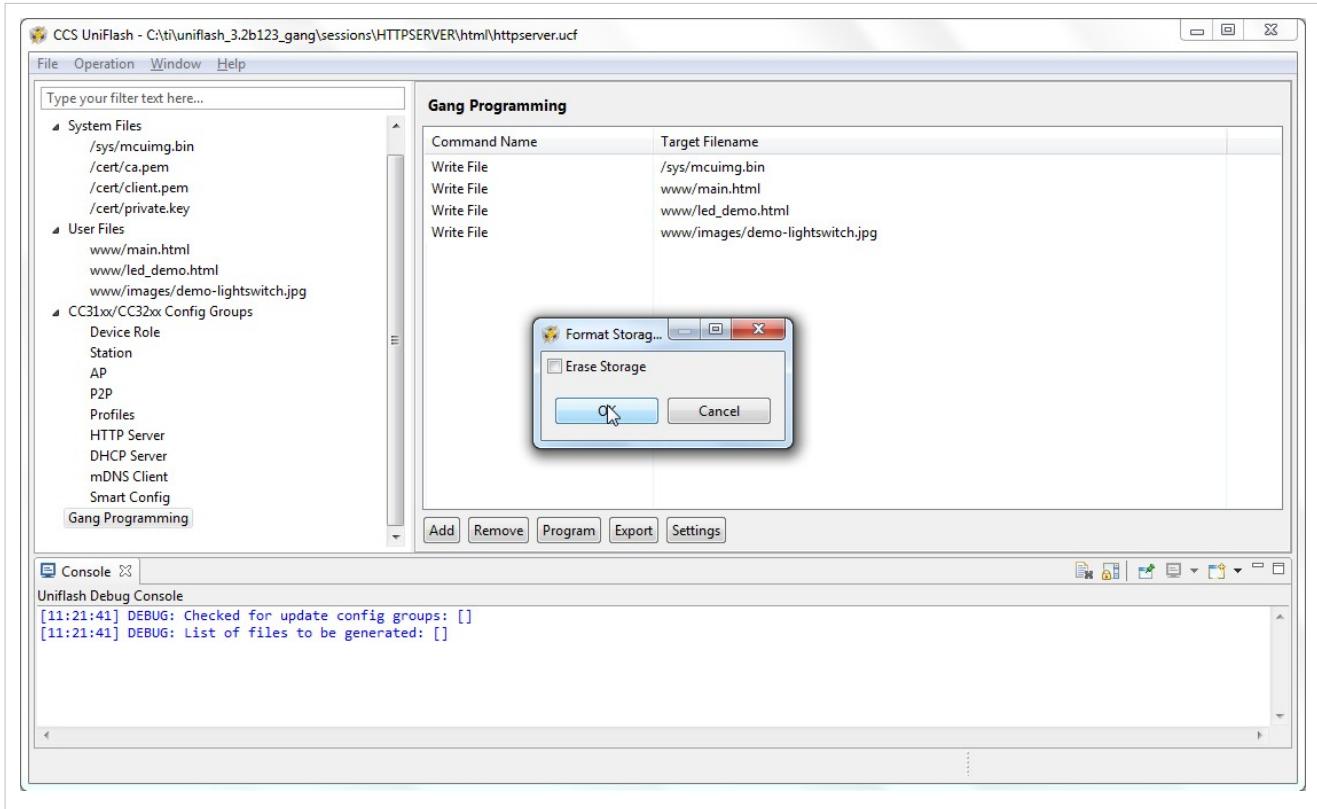
1. Build a session and include the following files:
 1. System files: by configuring any system file from the System Files group
 2. Configuration files: by configuring the desired Configuration Groups and checking their respective Update checkbox
 3. User files: by manually adding files via the Add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



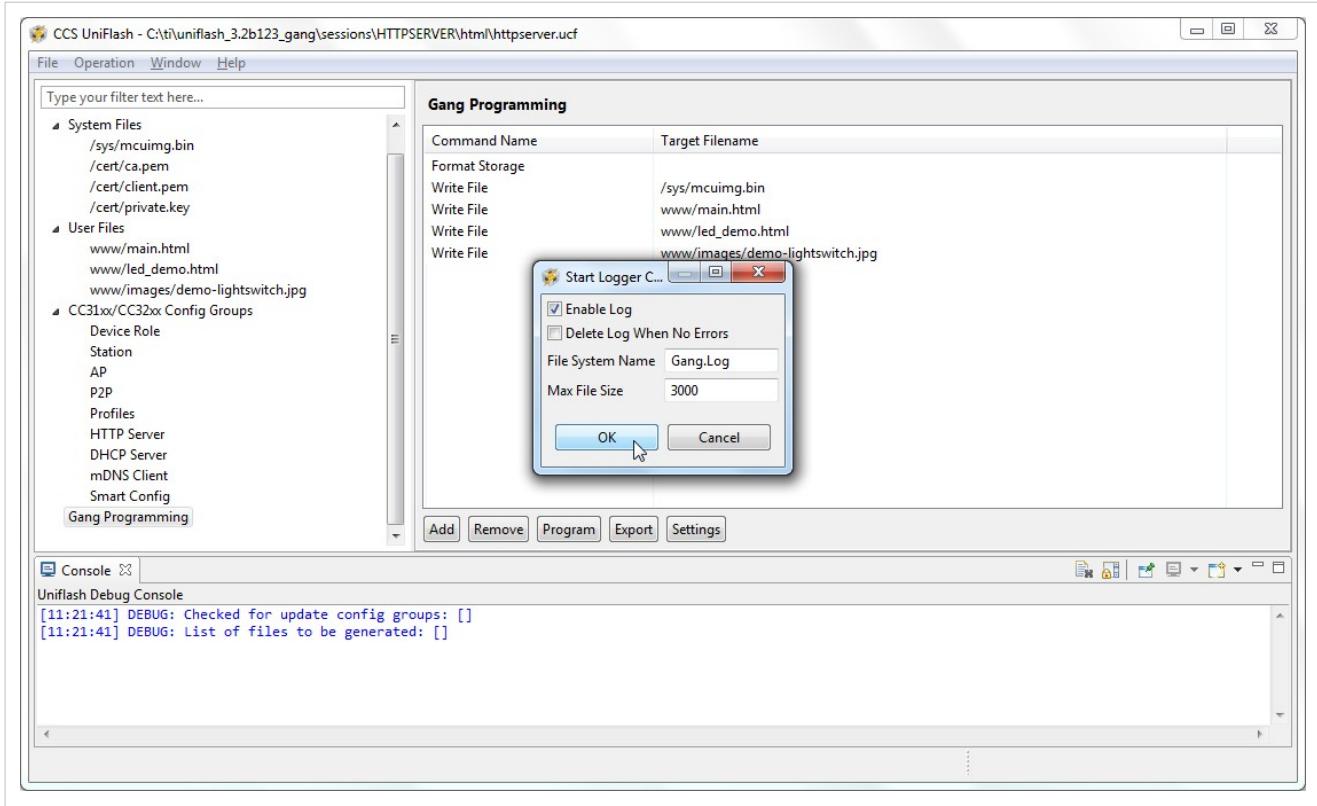
4. Click on Settings button and configure the Storage Capacity Bytes from the drop down window



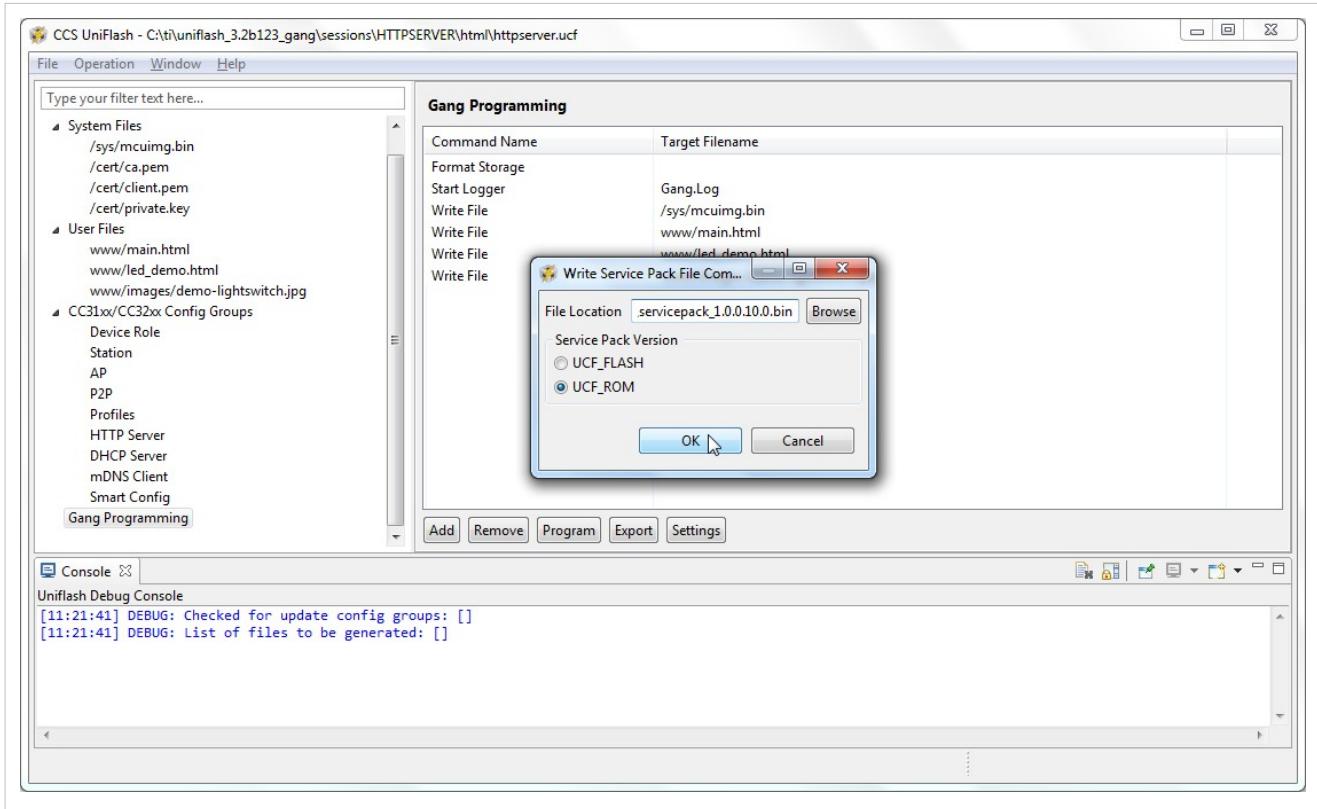
5. Add a format command by clicking the Add>Add Format Storage Command. This step is mandatory as it creates the FAT table



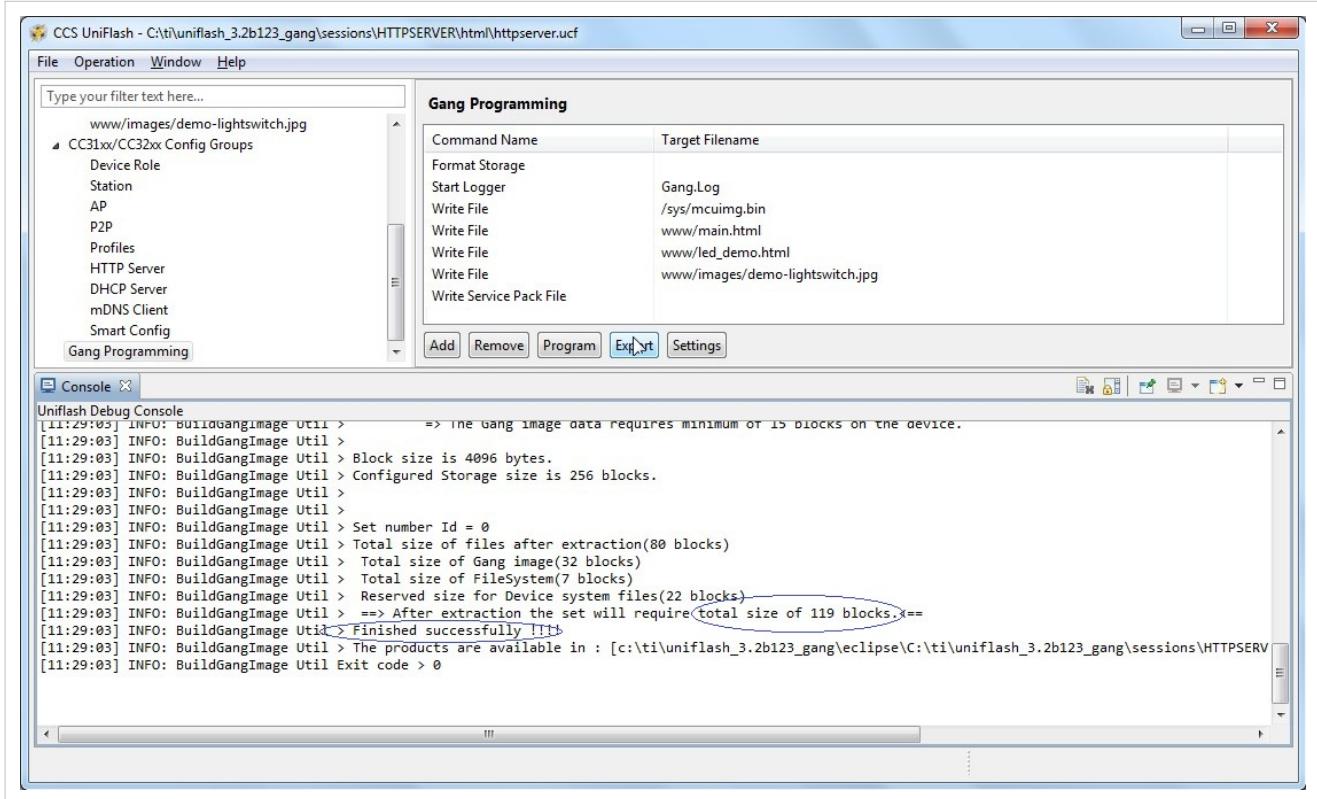
6. It is optional to add logging during image extraction. It can be done by clicking the Add>Add Start Logger Command. It is recommended mainly in development phase and not during production line



7. 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



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



9. 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 the figure

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

Serial Flash usage and design

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.

File system guidelines

The guidelines are device generic and not related specifically to 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:
 - 1 Block (4KB) for file with no fail safe support
 - 2 Blocks (8KB) for file with fail safe support (twice the original size)
- The file system aims to store all or some of the followings: ServicePack file, system files, configuration files, certificate files, web pages files, user files
- 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 $4000+440 > 4096$

Memory usage

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

Following table describes all 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/pmcfg.ini	Power Management	true	2
/sys/ipcfg.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/smconfigkeys.cfg	SmartConfig Keys	true	2
TOTAL			28

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.

Image specific

The image includes not only the file content but also some metadata that is required during image extraction and programming. 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

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
 - Please 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
 - Please open an E2E ticket in all other cases

Troubleshoot/Debugging

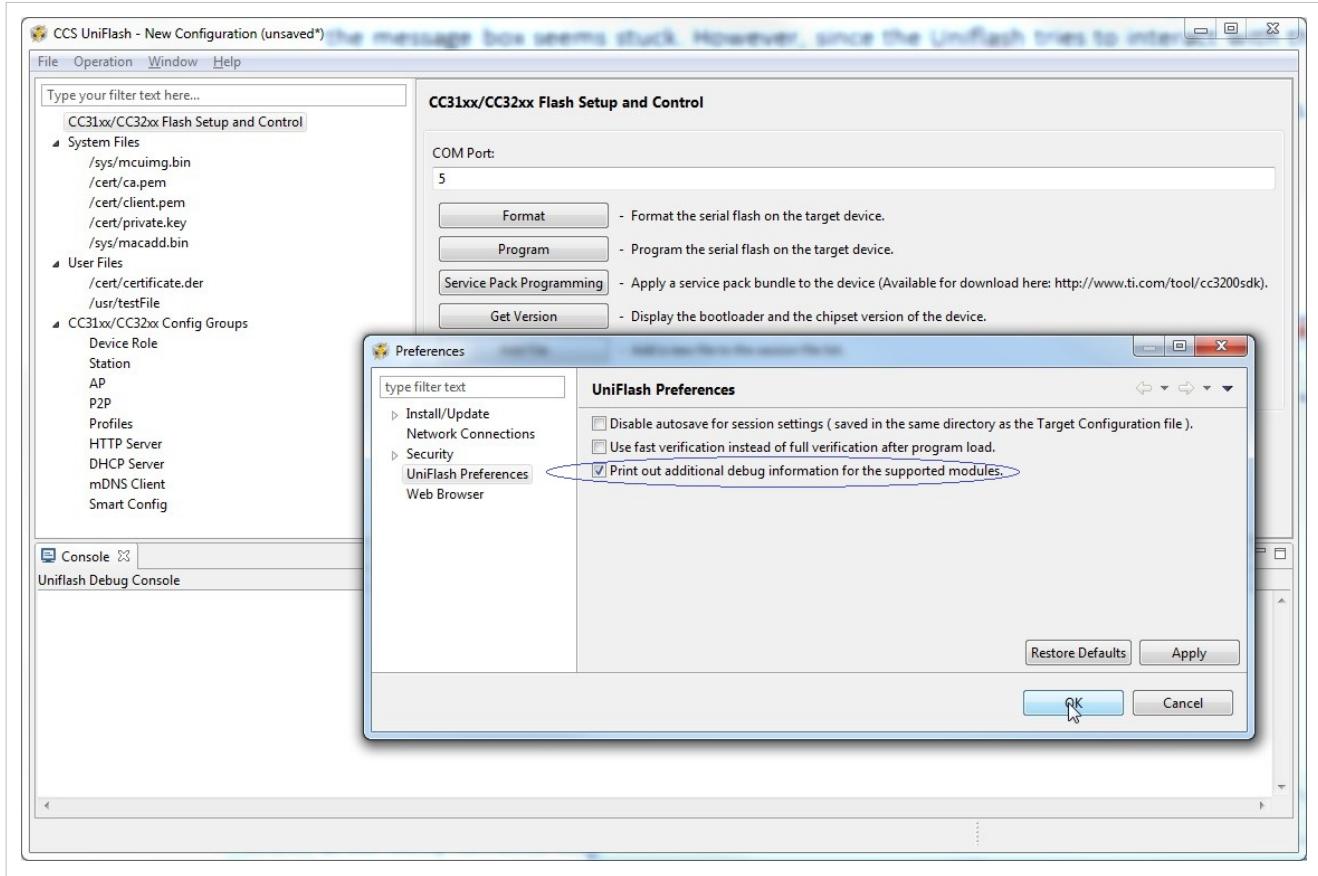
Debug messages are printed on the '**Console**' screen. It is possible to control the verbose level of these printouts from the 'Window -> Preferences'. Under '**Uniflash Preferences**', the '**Print out additional debug information for the supported modules**' should be checked for full debug messages.

By default, this mode is enabled.

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

- Pressing '**cancel**' during program operation. In this case, the progress bar on the message box is freezed 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 (i.e. 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.



Limitations

1. 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.

Note: it is possible to remove it from the GUI as well as from the device. Please refer to **Removing existing file from the template** paragraph.

2. Storage Format:

it is the user responsibility to know the capacity of the serial flash and format it accordingly

3. UART connection:

only 921600bps is supported

4. Linux related:

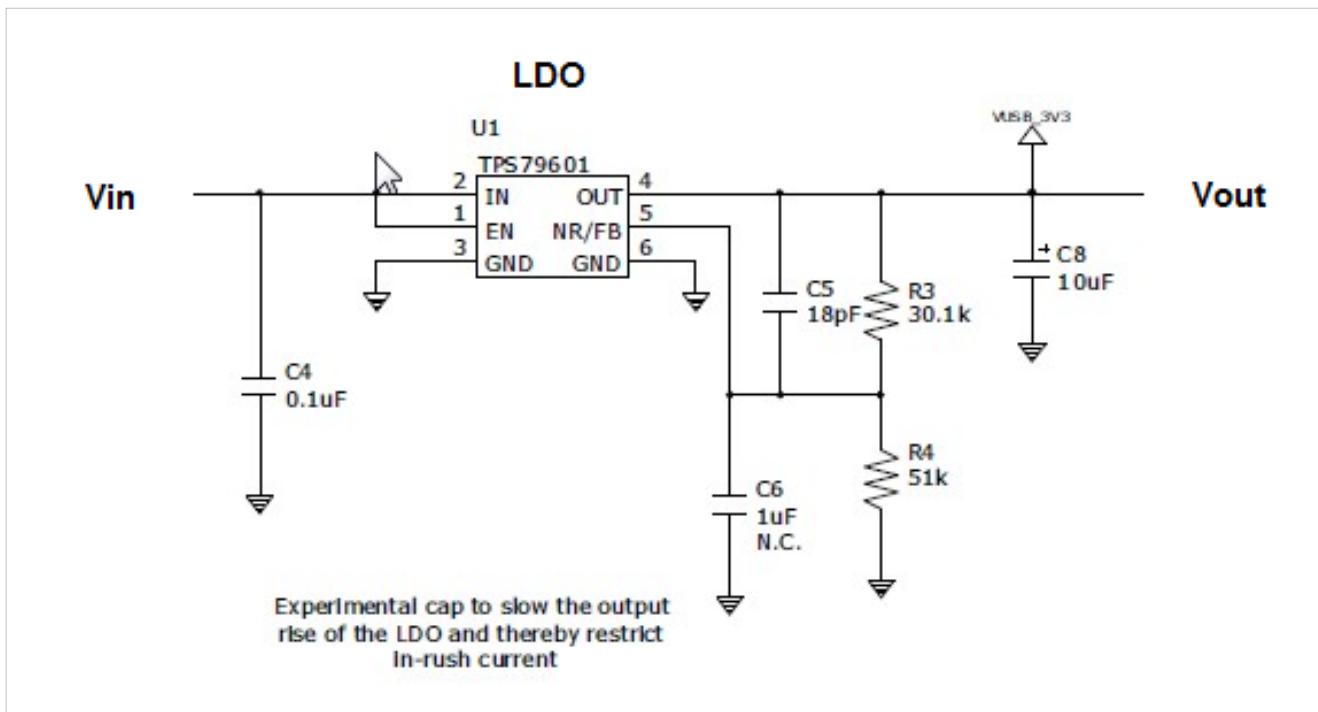
- No automatic reset of the connected target is supported (unlike Windows version)
- Using 'Verify' option (for files or image) results in much slower content reading than the Windows version
- It has been observed occasionally that communication with the target device is halted. It mainly happens during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

Annex A: 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). Please 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 5V coming from the USB to 3.3V ($V_{ref} \cdot (1+R_2/R_1) = 1.2246V \cdot (1+51K/30.1K) = 3.3V$). In this mode, all IO lines are tight to 3.3V and the board works in a pre-regulated 3.3V scheme. In this case, the UART lines used for the tool need to work in 3.3V as well.



References

- [1] <http://www.ti.com/tool/uniflash>

Article Sources and Contributors

CC31xx & CC32xx UniFlash *Source:* <http://processors.wiki.ti.com/index.php?oldid=204792> *Contributors:* A0221015, A0387625, A0389326, A0406448, Aambi, Alons, Beatrice, Dmyers, Made4engineering

Image Sources, Licenses and Contributors

File:Cc31xx_cc32xx_return_home.png *Source:* http://processors.wiki.ti.com/index.php?title=File:Cc31xx_cc32xx_return_home.png *License:* unknown *Contributors:* A0221015
Image:upgrade_123 to 321.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:Upgrade_123_to_321.jpg *License:* unknown *Contributors:* A0387625
Image:upgrade_65 to 120.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:Upgrade_65_to_120.jpg *License:* unknown *Contributors:* A0387625
Image:sw update problem.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:Sw_update_problem.jpg *License:* unknown *Contributors:* A0387625
Image:sw update preferences.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:Sw_update_preferences.jpg *License:* unknown *Contributors:* A0387625
Image:Uniflash block diagram.png *Source:* http://processors.wiki.ti.com/index.php?title=File:Uniflash_block_diagram.png *License:* unknown *Contributors:* A0387625
Image:session conversion.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:Session_conversion.jpg *License:* unknown *Contributors:* A0387625
Image:Saving a session.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:Saving_a_session.jpg *License:* unknown *Contributors:* A0387625
Image:Loading a session.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:Loading_a_session.jpg *License:* unknown *Contributors:* A0387625
Image:Loading a session from bar.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:Loading_a_session_from_bar.jpg *License:* unknown *Contributors:* A0387625
Image:Target Configuration screen.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:Target_Configuration_screen.jpg *License:* unknown *Contributors:* A0387625
Image:Main screen.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:Main_screen.jpg *License:* unknown *Contributors:* A0387625
Image:Format 3.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:Format_3.jpg *License:* unknown *Contributors:* A0387625
Image:Get version.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:Get_version.jpg *License:* unknown *Contributors:* A0387625
Image:Non secure flashing.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:Non_secure_flashing.jpg *License:* unknown *Contributors:* A0387625
Image:servicepack flashing.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:Servicepack_flashing.jpg *License:* unknown *Contributors:* A0387625
Image:Add file.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:Add_file.jpg *License:* unknown *Contributors:* A0272569, A0387625
Image:Erase file.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:Erase_file.jpg *License:* unknown *Contributors:* A0387625
Image:deleting file.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:Deleting_file.jpg *License:* unknown *Contributors:* A0387625
Image:Mcu flashing.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:Mcu_flashing.jpg *License:* unknown *Contributors:* A0387625
Image:device role configuration.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:Device_role_configuration.jpg *License:* unknown *Contributors:* A0387625
Image:station flow.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:Station_flow.jpg *License:* unknown *Contributors:* A0387625
Image:station configuration.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:Station_configuration.jpg *License:* unknown *Contributors:* A0387625
Image:ap flow.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:Ap_flow.jpg *License:* unknown *Contributors:* A0387625
Image:ap configuration.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:Ap_configuration.jpg *License:* unknown *Contributors:* A0387625
Image:p2p flow.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:P2p_flow.jpg *License:* unknown *Contributors:* A0387625
Image:p2p configuration.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:P2p_configuration.jpg *License:* unknown *Contributors:* A0387625
Image:http server flow.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:Http_server_flow.jpg *License:* unknown *Contributors:* A0387625
Image:http server configuration.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:Http_server_configuration.jpg *License:* unknown *Contributors:* A0387625
Image:dhcp server flow.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:Dhcp_server_flow.jpg *License:* unknown *Contributors:* A0387625
Image:dhcp server configuration.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:Dhcp_server_configuration.jpg *License:* unknown *Contributors:* A0387625
Image:mdns flow.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:Mdns_flow.jpg *License:* unknown *Contributors:* A0387625
Image:mdns main screen.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:Mdns_main_screen.jpg *License:* unknown *Contributors:* A0387625
Image:mdns configuration.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:Mdns_configuration.jpg *License:* unknown *Contributors:* A0387625
Image:profiles flow.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:Profiles_flow.jpg *License:* unknown *Contributors:* A0387625
Image:profiles main screen.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:Profiles_main_screen.jpg *License:* unknown *Contributors:* A0387625
Image:open profile screen.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:Open_profile_screen.jpg *License:* unknown *Contributors:* A0387625
Image:wep profile screen.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:Wep_profile_screen.jpg *License:* unknown *Contributors:* A0387625
Image:wpa profile screen.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:Wpa_profile_screen.jpg *License:* unknown *Contributors:* A0387625
Image:p2p profile screen.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:P2p_profile_screen.jpg *License:* unknown *Contributors:* A0387625
Image:tls profile screen.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:Tls_profile_screen.jpg *License:* unknown *Contributors:* A0387625
Image:ttlsMschapv2 profile screen.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:TtlsMschapv2_profile_screen.jpg *License:* unknown *Contributors:* A0387625
Image:peapPsk profile screen.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:PeapPsk_profile_screen.jpg *License:* unknown *Contributors:* A0387625
Image:fast profile screen.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:Fast_profile_screen.jpg *License:* unknown *Contributors:* A0387625
Image:smartconfig flow.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:Smartconfig_flow.jpg *License:* unknown *Contributors:* A0387625
Image:smartconfig configuration.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:Smartconfig_configuration.jpg *License:* unknown *Contributors:* A0387625
Image:Image Add command.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:Image_Add_command.jpg *License:* unknown *Contributors:* A0387625
Image:Image Multiple Remove.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:Image_Multiple_Remove.jpg *License:* unknown *Contributors:* A0387625
Image:Image move commands.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:Image_move_commands.jpg *License:* unknown *Contributors:* A0387625
Image:Image Export button.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:Image_Export_button.jpg *License:* unknown *Contributors:* A0387625
Image:Image General Settings button.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:Image_General_Settings_button.jpg *License:* unknown *Contributors:* A0387625
Image:Image Programming button.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:Image_Programming_button.jpg *License:* unknown *Contributors:* A0387625
Image:Image Programming GUI flow.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:Image_Programming_GUI_flow.jpg *License:* unknown *Contributors:* A0387625
Image:Image Import Session.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:Image_Import_Session.jpg *License:* unknown *Contributors:* A0387625
Image:Image General Settings.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:Image_General_Settings.jpg *License:* unknown *Contributors:* A0387625
Image:Image Add Format.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:Image_Add_Format.jpg *License:* unknown *Contributors:* A0387625
Image:Image Add Logger.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:Image_Add_Logger.jpg *License:* unknown *Contributors:* A0387625
Image:Image Add ServicePack.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:Image_Add_ServicePack.jpg *License:* unknown *Contributors:* A0387625
Image:Image Export.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:Image_Export.jpg *License:* unknown *Contributors:* A0387625
Image:Debugging.jpg *Source:* <http://processors.wiki.ti.com/index.php?title=File:Debugging.jpg> *License:* unknown *Contributors:* A0387625
Image:UART connection.png *Source:* http://processors.wiki.ti.com/index.php?title=File:UART_connection.png *License:* unknown *Contributors:* A0387625