UniFlash CC3120, CC3220 SimpleLink™ Wi-Fi® and Internet-on-a chip™ Solution ImageCreator and Programming Tool

The CC3120 and CC3220 devices are part of the SimpleLink™ microcontroller (MCU) platform, which consists of Wi-Fi®, *Bluetooth*® low energy, Sub-1 GHz and host MCUs, which all share a common, easy-to-use development environment with a single core software development kit (SDK) and rich tool set. A one-time integration of the SimpleLink platform enables you to add any combination of the portfolio's devices into your design, allowing 100 percent code reuse when your design requirements change. For more information, visit www.ti.com/simplelink.

This user's guide describes the UniFlash CC3120 and CC3220 SimpleLink Wi-Fi and Internet-on-a chip™ Solution ImageCreator and Programming Tool from Texas Instruments™.

Contents

1 Introduction 3
2 Terms and Concepts 4
3 Installation 4
4 Image Creator Application 5
5 Use 6
5.1 Creating a New Project 6
5.2 Opening a Recent Project 6
5.3 Managing Projects 7
5.4 Device Status and Settings 8
5.5 Setting Device Configuration 10
5.6 Adding the Service Pack 10
5.7 Adding the Trusted Root-Certificate Catalog 10
5.8 User Files 11
5.9 Device File Browser 15
5.10 Creating an Image From a Project 16
5.11 Creating an Image 16
5.12 Creating an Encrypted Image 16
5.13 Creating an OTA 17
5.14 Saving an Image 17
5.15 Programming.bin and Programming.hex 18
5.16 Programming an Image From an Opened Project 18
5.17 Programming an Image Using a .sli File 18
5.18 Secured Image With Key 19
6 Command Line 19
6.1 Project Commands 20
6.2 Image Commands 25
6.3 Tools Commands 26
6.4 Device Commands 26
6.5 GUI Configure Commands 27
7 Tools 28
7.1 Sign File 28
7.2 Activate Image 28

List of Figures

1 Programming Using the Image Creator 3
2 Opening ImageCreator Through UniFlash (1 of 2) 5
3 Opening ImageCreator Through UniFlash (2 of 2) 5
4 New Project 6
5 Open Recent Project 6
6 Project Management 7
7 Device Status: Disconnected 8
8 Device Status: Connected 9
9 Tool Tips 10
10 MCU Image 10
11 File Properties Dialog 12
12 Rename Filename 13
13 Delete File or Folder (1 of 2) 13
14 Delete File or Folder (2 of 2) 14
15 User File Action Monitor 14
16 Delete File 15
17 Get File 15
18 Example 16-Bit Key 16
19 Set Key Filename 16
20 OTA Private Key File Name 17
21 Save Image 17
22 Program Image 18
23 Program Image 19
24 Open Tools 28
25 Sign File 28
26 Activate Image 28

List of Tables

1 Terms and Concepts 4
2 Flags Options 12
3 Other File Properties 13

Trademarks

SimpleLink, Internet-on-a chip, Texas Instruments are trademarks of Texas Instruments.
Bluetooth is a registered trademark of Bluetooth SIG.
Wi-Fi is a registered trademark of Wi-Fi Alliance.
All other trademarks are the property of their respective owners.

1 Introduction

ImageCreator is a part of the UniFlash application used to create a programming image; the ImageCreator can also write the programming image into the SimpleLink™ CC3x20 devices. The programming image is a file containing the SimpleLink device configurations and files required for the operation of the device. For the SimpleLink CC3220 wireless microcontroller (MCU), the programming image also includes the host application file.

A new SimpleLink device should first be programmed by a programming image. The image, created by the ImageCreator, can be programmed onto the device as part of the production procedure, or in development stage. The image can be programmed as follows:

- Using the ImageCreator tool through a UART interface
- Using an external off-the-shelf tool through a serial-flash SPI

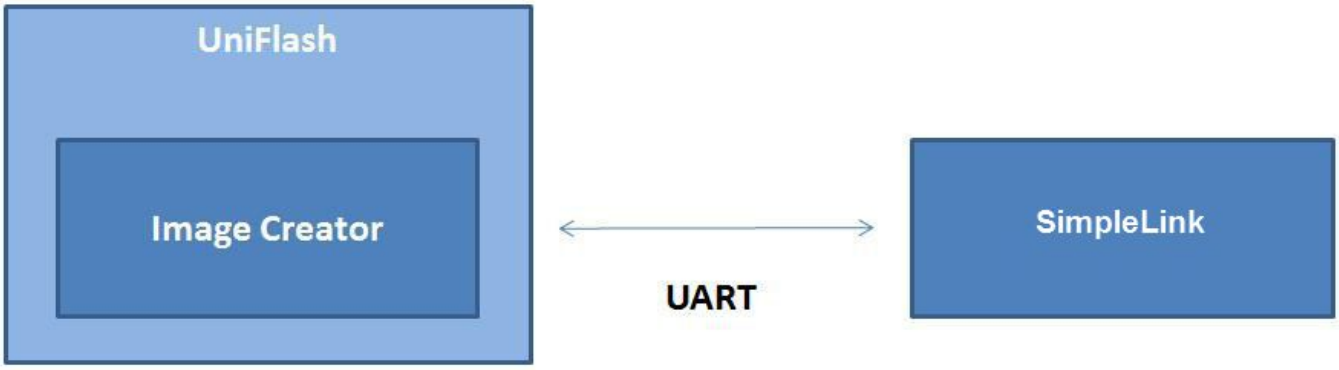


Figure 1. Programming Using the Image Creator

The main features of the ImageCreator are:

- Supports both the SimpleLink CC3220 and the SimpleLink CC3120 devices
- Can create an image in either production mode or development mode. Production images restrict some of the features intended for development, such as the JTAG interface or access to individual files using this tool.
- Creates encrypted programming images
- As part of the programming image creation:
  - Applies the service pack and the certificate store
  - Defines the device configurations, such as Wi-Fi® mode, IP settings, provisioning, and more
  - Adds files and applies attributes per file, such as security settings and fail safe
- Connects to the device and retrieves its properties, such as device type, flash size, and MAC address
- When in development, the image supports on-line access to the device file system
- Programs the device, and can program using an image created by another instance of image creator
- Executes some operations using a command line interface
- Ability to manage projects: import an existing project, or export a project to another machine

This document describes the installation, operation, and usage of the SimpleLink ImageCreator tool as part of the UniFlash.

2 Terms and Concepts

Table 1 lists some of the terms and concepts used in this document.

Table 1. Terms and Concepts

Term or Concept	Description
Image	Image is a packed file which contains the service pack, the system configuration files, the user files, and the host program (in case of the SimpleLink Wi-Fi CC3220 wireless MCU). The process of creating the programming image is an off-line process.
Project	Project is a workspace for creating an image file.
Connection	Users can connect to the device and get its attributes, such as its MAC address, security type, and so forth.
Key	The 16-byte key is used for image encryption.
Programming image file types	The image file is created in several encoding types: • Programming.bin and programming.hex, standard binary and intel hex files, are used for programming by an external serial flash programming tool. • Programming.ucf, (TI proprietary encoding) is used for programming by the host. • Programming.sli, (TI proprietary encoding) is used for programming by the image creator. • Notes – Encrypted images are named programming.encrypt.bin/hex/ucf/sli – The output files are under the image creator installation directory in *\projects\${project_name}\sl_image\Output*

3 Installation

ImageCreator is a part of the UniFlash application. Download and run the latest installer of the UniFlash application from http://www.ti.com/tool/UNIFLASH.

4 Image Creator Application

Run the UniFlash application. A list of all supported devices appears; choose *CC3120 / CC3220* from the device list, as shown in Figure 2.

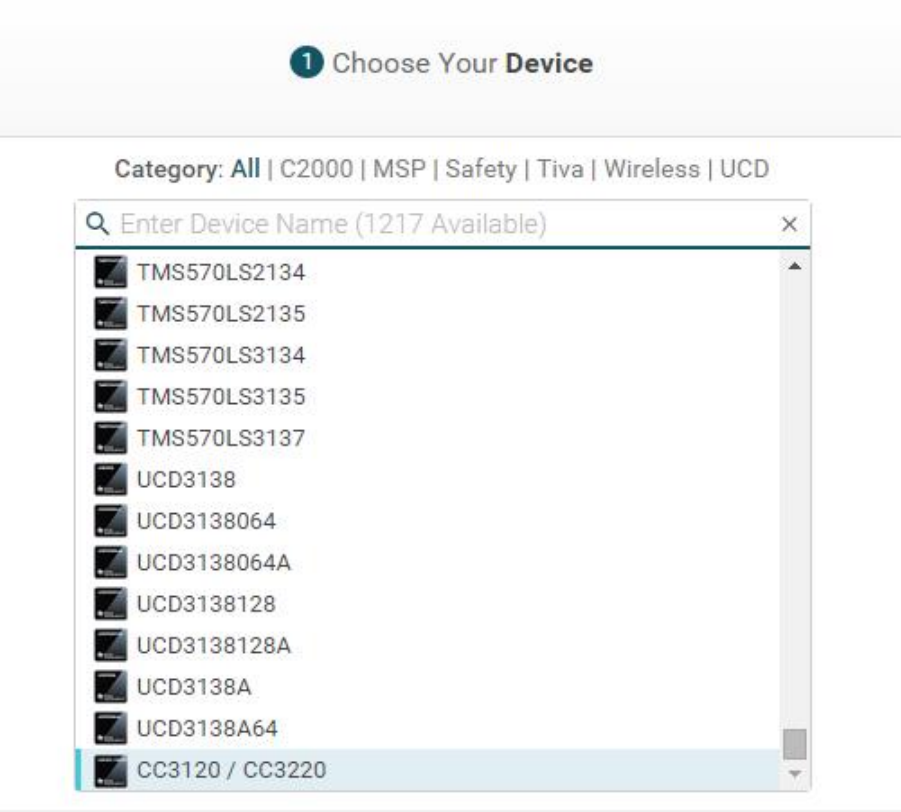


Figure 2. Opening ImageCreator Through UniFlash (1 of 2)

Then press on the Start Image Creator button, as shown in Figure 3.

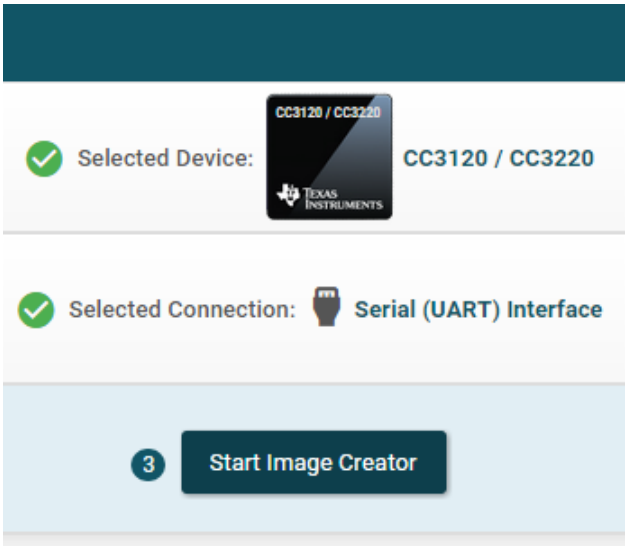


Figure 3. Opening ImageCreator Through UniFlash (2 of 2)

5 Use

5.1 Creating a New Project

Click the New Project button on the Welcome page, as shown in Figure 4.

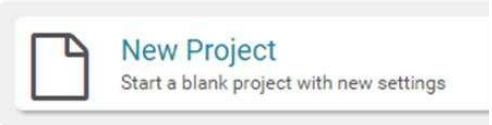


Figure 4. New Project

The Create Project window will appear. Users should fill out the relevant fields, and click the Create Project button. A new project with default parameters is then created.

5.2 Opening a Recent Project

Open an existing project by clicking on the project name in the recent projects list, directly from the main "Welcome…" page. The main page can be navigated by clicking on the  (see Figure 5).

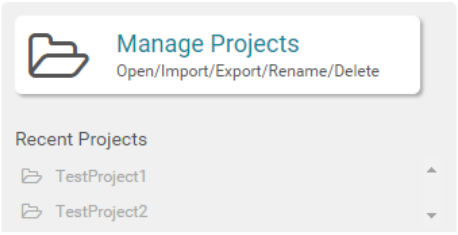


Figure 5. Open Recent Project

5.3 Managing Projects

Open the list of all projects by pressing on the Manage Projects button shown in Figure 5. From here, the screen appears as in Figure 6.

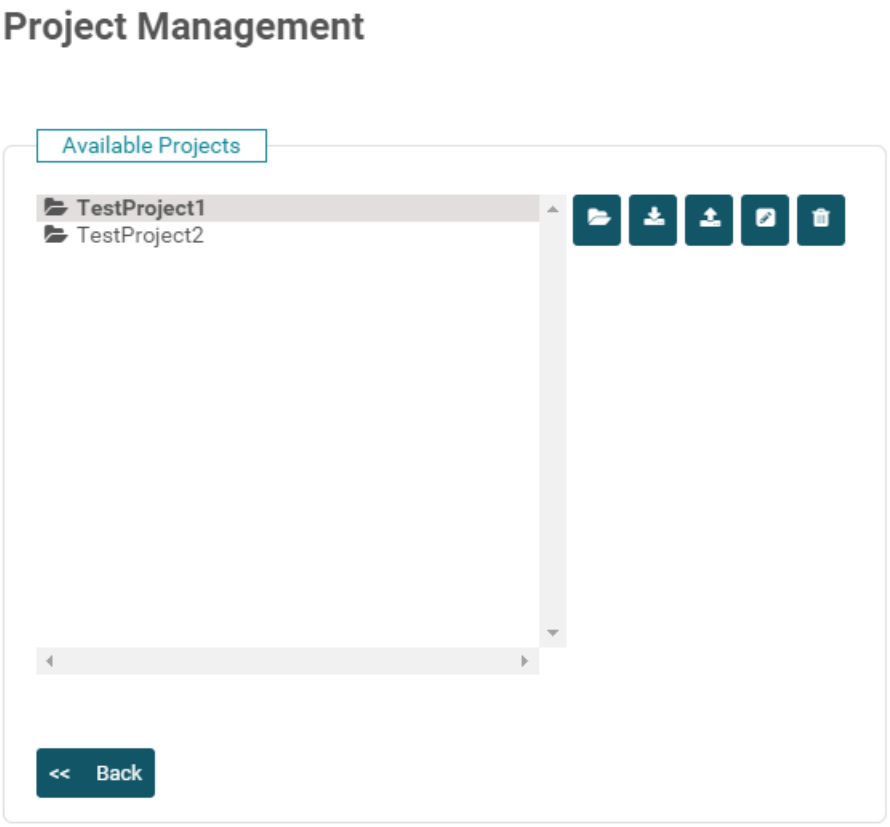


Figure 6. Project Management

The available operations are:

- Open project – Press  to open the project.
- Import project from zip file – Press  to zip the file.
- Export project – Press  to export the project.
- Rename project – Press  to rename the project.
- Delete project – Press  to delete the project.

5.4 Device Status and Settings

A programming image can be prepared and created while offline, for example, while the device is not physically connected to the Image Creator app computer (see Figure 7).

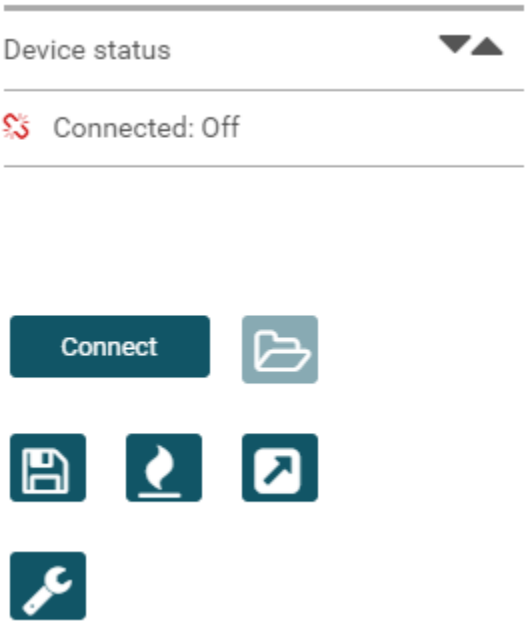


Figure 7. Device Status: Disconnected

If a device is physically connected by UART, the user can click the Connect button to automatically detect the device and perform an initial connection. The connect method retrieves the settings from the device and displays them, as shown in Figure 8.

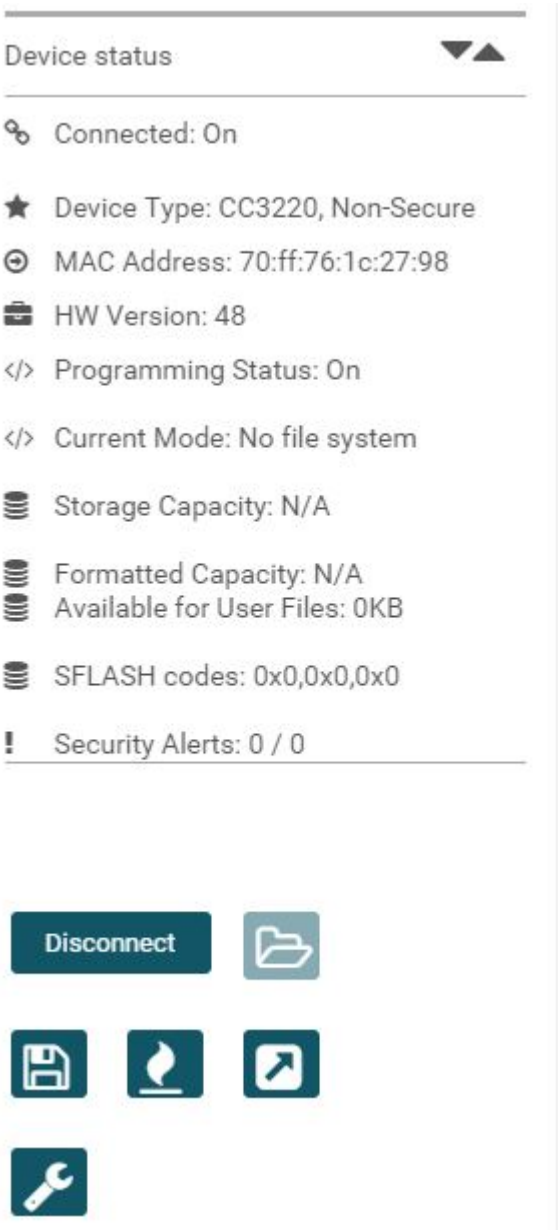


Figure 8. Device Status: Connected

The user can choose production or development mode:

- Production mode – The created image is programmable on any device.
  - For the device security, production mode exposes limited operations:
    - An online operation on the file system using the SimpleLink Image Creator is disabled.
    - Using JTAG (on the CC3220 device) is disabled.
- Development mode – Requires the target device MAC address to program it. The target device MAC address is set by the Image Creator setting window. This mode allows:
  - Browsing and modification of the device file system (see Section 5.9).
  - Using JTAG (CC3220) is enabled.
  - The programming image file can be used to program only the device with the same MAC address as the one set into the image.

5.5 Setting Device Configuration

The left side of the screen contains links to the configuration pages, organized in a tree structure. The tree structure enables quick navigation to any configuration page in a single click.

Fields within the pages contain tool tips, with explanations that appear when the mouse is moved over the question mark icon, as shown in Figure 9.

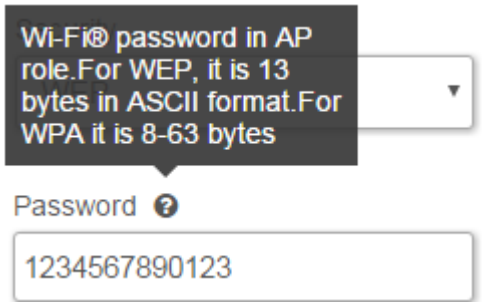


Figure 9. Tool Tips

5.6 Adding the Service Pack

The service pack is used to upgrade the device software. The service pack file is provided by TI. TI recommends adding the service pack to the programming image; this action, however, is not mandatory. If it is not programmed, the device will use its factory code.

When adding the service pack, the user selects the file location; however, the ImageCreator does not keep a link to the original file. To change the service pack, the new service pack file should be selected again.

5.7 Adding the Trusted Root-Certificate Catalog

The trusted root-certificate is a file provided by TI. The store contains a list of known and trusted root CAs and a list of revoked certificates. The list of the CAs supported by TI can be found in the *SimpleLink™ CC3120, CC3220 Wi-Fi® Internet-on-a chip™ Solution Built-In Security Features Application Report*.

The ImageCreator installation has a default trusted root-certificate catalog used by the ImageCreator. The default trusted root-certificate can be overridden by selecting a different file and its signature file. The ImageCreator has no link to the selected trusted root-certificate original file. To change the trusted root-certificate content, select a new file.

5.7.1 Adding the Host Application File (CC3220)

The SimpleLink ImageCreator allows adding the host application file for CC3220 devices. On the User Files, open the action drop menu and select the Select MCU Image, as shown in Figure 10.

Figure 10. MCU Image

Press the Browse button and select the MCU Image file from the local drive. The File properties dialog appears. See Section 5.8.3.

Configuring the host application file properties should be followed by clicking on the Save button. The host application file is created, with the following name on the device:

- For CC3220 R/RS: /sys/mcuimg.bin
- For CC3220SF: /sys/mcuflashimg.bin

For secure devices, the host file must be created with the flags secure-signed. To enable future updates of the file (by OTA), the user opens it with the public-write flag. In addition, the maximum size of the file should consider the future growth of the file, as the maximum size of a file cannot be changed after the file creation.

5.7.1.1 Host Application for the CC3220SF Device

The CC3220SF application requires adding 20 bytes of SHA to the beginning of the host file. The SHA is the result of a hash algorithm, which is calculated on the host file content. The file signature is calculated on the host file content, including the SHA. The creation steps are as follows:

1. Add SHA to the beginning of the file (host_final).
2. Calculate the host signature (signature of the host_final file).

The ImageCreator offers two methods of adding the host application:

- The application file, including the SHA, is created by the user and the file signature. The input is the host_final and the signature of the host_final.
- The host_file, without the SHA, is set by the user:
  1. The ImageCreator calculates the SHA and the host_final.
  2. The file signature of the host_final file is calculated by the ImageCreator, using the private key in DER format as an input.

NOTE: Increase the File Maxsize setting of the host application by 20 bytes, to include the file SHA.

5.8 User Files

A user file can be added to the image because the ImageCreator supports files operations, including adding or removing a file, creating a directory, and viewing file properties.

File operations are available while moving the cursor over the file/directory icon. After a file is chosen, the file is saved as part of the project.

The ImageCreator files are not linked to the original selected files. To change a file, content in the file should be deleted and added again.

5.8.1 Secure Signed User Files

For secure signed files, the Image Creator must receive a signed certificate and the file signature. For more information regarding how to retrieve a signed certificate and how to create a file signature, refer to the secure file system chapter in the user manual (search for the sl_FsClose () function).

When the certificate is chained to another certificate, the name of the chained certificate should be in the certificate “issued to” field. All the certificates in the chain are added to the project before adding the file signed by them.

Image Creator adds secure signed files using the following methods:

- Sets a file signature.
- Receives the private key; using the key, the Image Creator creates the file signature.

NOTE: In both methods, a signed certificate containing the public key must be supplied.

To enable future updates of the file (by OTA), add the file with the public write flag. Another option is to use the vendor token flag and define the file master token.

5.8.2 Adding a File

To add a file, users should click the icon, or drag-and-drop the desired file from the appropriate folder. After a file is added, the File properties dialog appears. See Figure 11.

5.8.3 Editing a File

To edit the properties of a file, select the file and press . The File properties dialog appears, as shown in Figure 11.

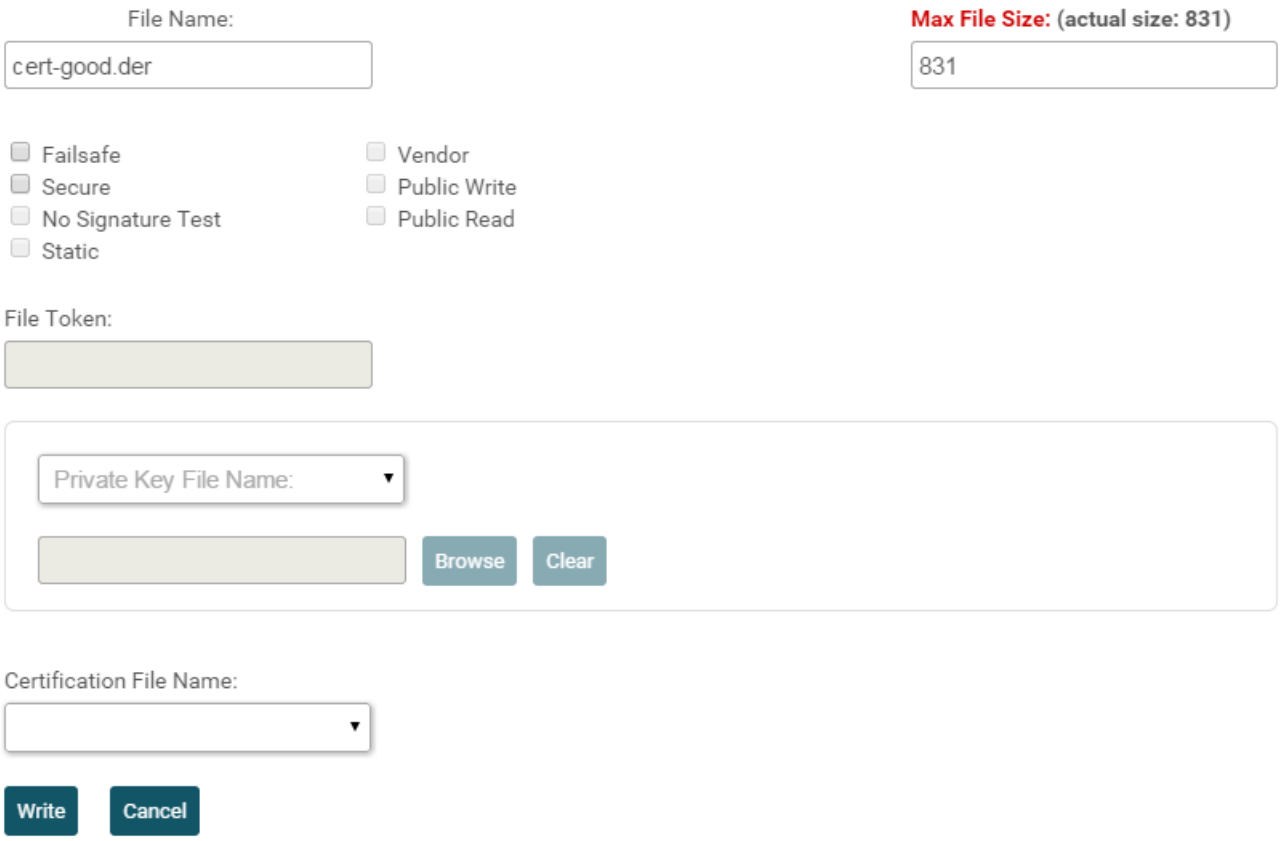


Figure 11. File Properties Dialog

Table 2 lists the flag options.

Table 2. Flags Options

Flag Option	Description
FailSafe	Editing the file is fail-safe. For more details, refer to the secure file system chapter in the user's manual, search for: SL_FS_CREATE_FAILSAFE
Secure	File is encrypted on the serial flash. For more details, refer to the secure file system chapter in the user's manual, search for: SL_FS_CREATE_SECURE
No Signature Test	Relevant only for secure files. By default, secure files require a signature. For more details, refer to the secure file system chapter in the user's manual, search for: SL_FS_CREATE_NOSIGNATURE
Static	Relevant only for secure files. Tokens are not replaced each time a file is open for write. For more details, refer to the secure file system chapter in the user's manual, search for: SL_FS_CREATE_STATIC_TOKEN

Table 2. Flags Options (continued)

Flag Option	Description
Vendor	Relevant only for secure files. The master token is set by the vendor. For more details, refer to the secure file system chapter in the user's manual, search for: SL_FS_CREATE_VENDOR_TOKEN
Public Write	Relevant only for secure files. The file can be written without a token. For more details, refer to the secure file system chapter in the user's manual, search for: SL_FS_CREATE_PUBLIC_WRITE
Public Read	Relevant only for secure files. The file can be read without a token. For more details, refer to the secure file system chapter in the user's manual, search for: SL_FS_CREATE_PUBLIC_READ

Table 3. Other File Properties

File Property	Description
File token	Relevant only when using the vendor flag. Token for secured file.
Signature filename	Relevant only when using the secure-signed flag, the signature file should be picked by browsing it on the local machine. The filename should then appear in the test box.
Certification filename	Relevant only when using the secure-signed flag, the list of previously added certificates should appear. Users should pick the relevant file from the list.
Maximum size	The size of the storage to allocate for the file. By default, the ImageCreator sets the maximum size as the actual size of the file. To enable future updates of the file (by OTA) set the maximum size to the maximum future growth of the file, (maximum size of a file cannot be changed for the existing file). The maximum size of a file will be rounded up by the device to correlate the serial flash block size (4096 bytes); for more information, see the secure file system chapter in the user manual.

To rename a file, use the File Name field on the on the File Properties dialog, as shown in Figure 12.

Figure 12. Rename Filename

5.8.4 Adding a Folder

To add a new folder, locate the desired position on the root folder and click the icon.

5.8.5 Deleting a File or Folder

Check the files or folders to be deleted. In the Action drop-down menu, choose Remove Selected (see Figure 13), then click Apply (see Figure 14).

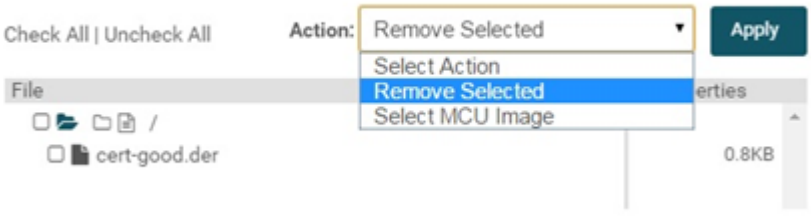


Figure 13. Delete File or Folder (1 of 2)

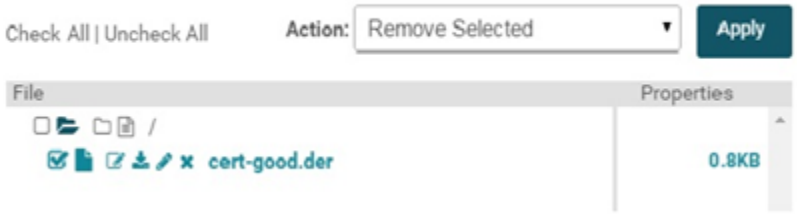


Figure 14. Delete File or Folder (2 of 2)

The user can also delete a single file or folder by directly clicking its X (delete) button.

5.8.6 Overwriting a File

The user can overwrite the file by clicking on the pencil button ( ).

5.8.7 User File Action Monitor

The user can drag a file and drop it on the user file area, as shown in Figure 15. During the action, the user file action monitor shows the action itself.

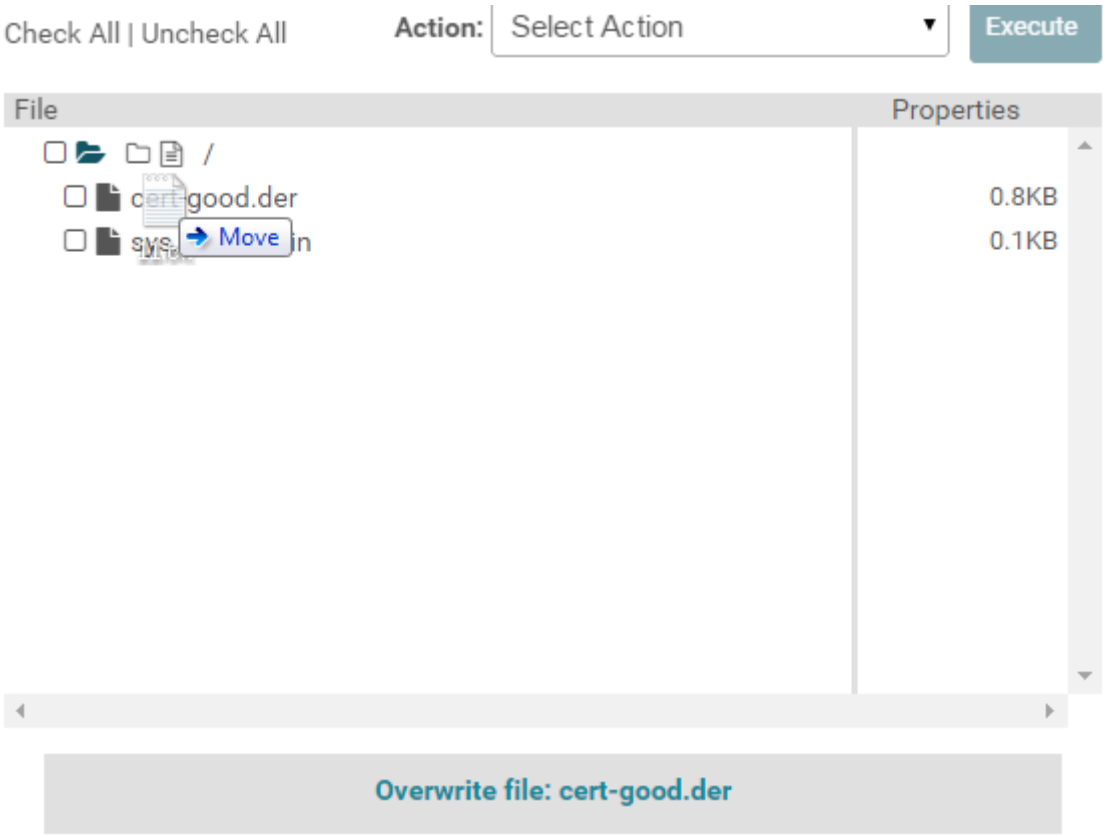


Figure 15. User File Action Monitor

5.9 Device File Browser

A device programmed with a development image allows for browsing its files, user files, and system files, and can perform several operations listed in the following subsections.

Most system files can not be viewed in the file list, and have no read/write access.

The online browser edits the file on the device serial flash; the changes do not affect the programming image file, which is prepared with the offline browser.

To view the online file browser, press the following icon: 

5.9.1 Adding a File

To add a file, users should click the  icon.

The File properties dialog appears. See Section 5.8.

5.9.2 Editing a File

Move the cursor over a filename, and click the  icon.

The File properties dialog appears. See Figure 11.

5.9.3 Adding a Folder

To add new folder, locate its position on the root folder and click the  icon.

5.9.4 Deleting a File

To delete a file, point to the file to be deleted and click the Delete button, as shown in Figure 16. If the file is secure, a prompt for the file token will appear.

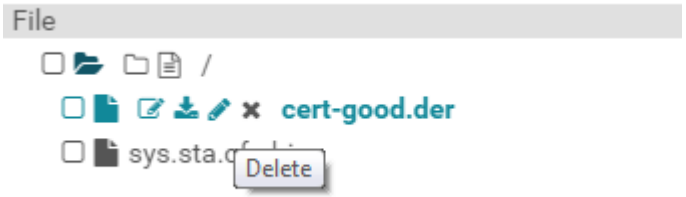


Figure 16. Delete File

5.9.5 Retrieving a File

To retrieve (upload) a file from the device, move the cursor over the file and click the Get File button, as shown in Figure 17. If the file is secure, a prompt for the file token will appear.

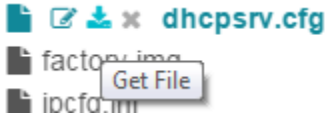


Figure 17. Get File

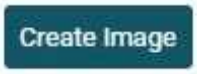
5.10 Creating an Image From a Project

When the configuration phase is over, creating an image is done by clicking on the Generate Image button

.

5.11 Creating an Image

On this phase, all types of programming image files are created, and can be found under the ImageCreator installation directory.

To create an image, click the Create Image button: 

.

5.12 Creating an Encrypted Image

The ImageCreator lets the user to create encrypted images (using AES-CTR encryption). An encrypted image can only be used with its key.

To create an encrypted image, create a binary file that contains a 16-byte key, such as the one shown in Figure 18.

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E
00000000:	11	22	33	44	55	66	77	88	99	00	11	22	33	44	55

Figure 18. Example 16-Bit Key

Then, in the General → Setting window, set the key filename as shown in Figure 19.

Figure 19. Set Key Filename

5.13 Creating an OTA

To create an OTA image, click the Create OTA button 

.

The OTA security dialog appears as in Figure 20.

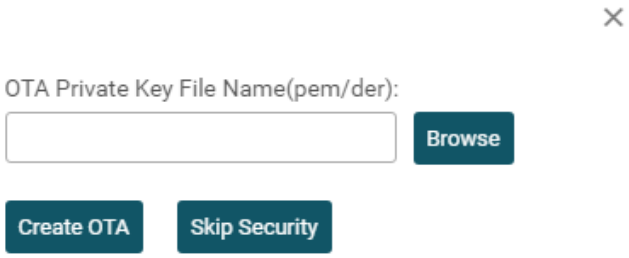


Figure 20. OTA Private Key File Name

5.13.1 Creating an OTA With a Security Sign

1. Click on the Browse button and load the OTA private file name (pem or der).
2. Click the Create OTA button.

5.13.2 Creating an OTA Without a Security Sign

Click Skip Security button.

5.14 Saving an Image

The Save Image buttons become clickable upon a successful image creation.

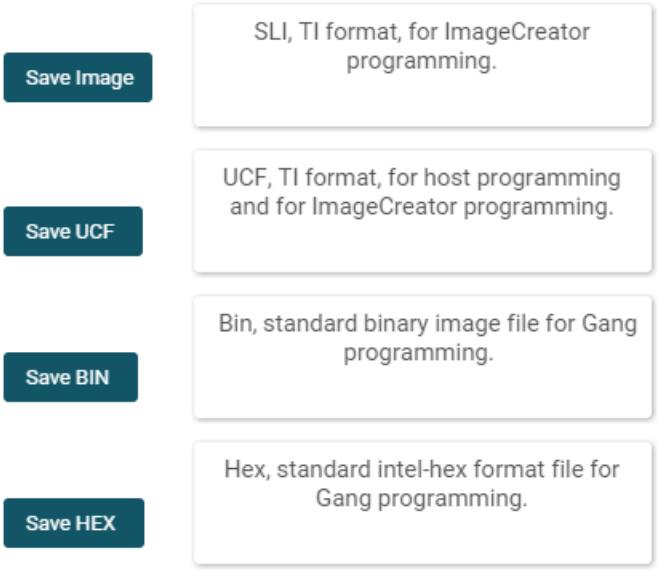


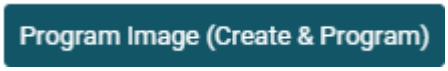
Figure 21. Save Image

5.15 Programming.bin and Programming.hex

Standard binary and intel hex files are used for programming by an external Sflash programming tool.

- Programming.ucf (TI proprietary encoding) is used for programming by the host.
- Programming.sli (TI proprietary encoding) is used for programming by the image creator.

5.16 Programming an Image From an Opened Project

To program an image, click the Program Image button: 

As part of the image programming, the ImageCreator first connects to the device (there is no need to connect to the device by pressing the Connect button before the image programming).

The program image also creates the image, so there is no need to create the image before the programming.

5.17 Programming an Image Using a .sli File

Once a .sli file is created, it can be used by any instance of the image creator to program the device. To use an existing .sli file, click the Program Image button on the Welcome page, as shown in Figure 22.

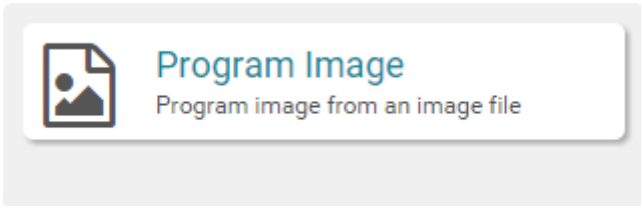


Figure 22. Program Image

The Open image dialog appears. Set the .sli file to be programed, and click on the Program Image button (see Section 5.16).

A sign file tool is in the tools section. After the user chooses a file to sign and a private key (see file system user manual for more information about supported key formats), the user can get the signed file as either binary or base 64 (see Section 7.2).

5.18 Secured Image With Key

For programming a secured .sli file image, set the key file which was used for the image encrypting, then click the Program Image button (see Figure 23).

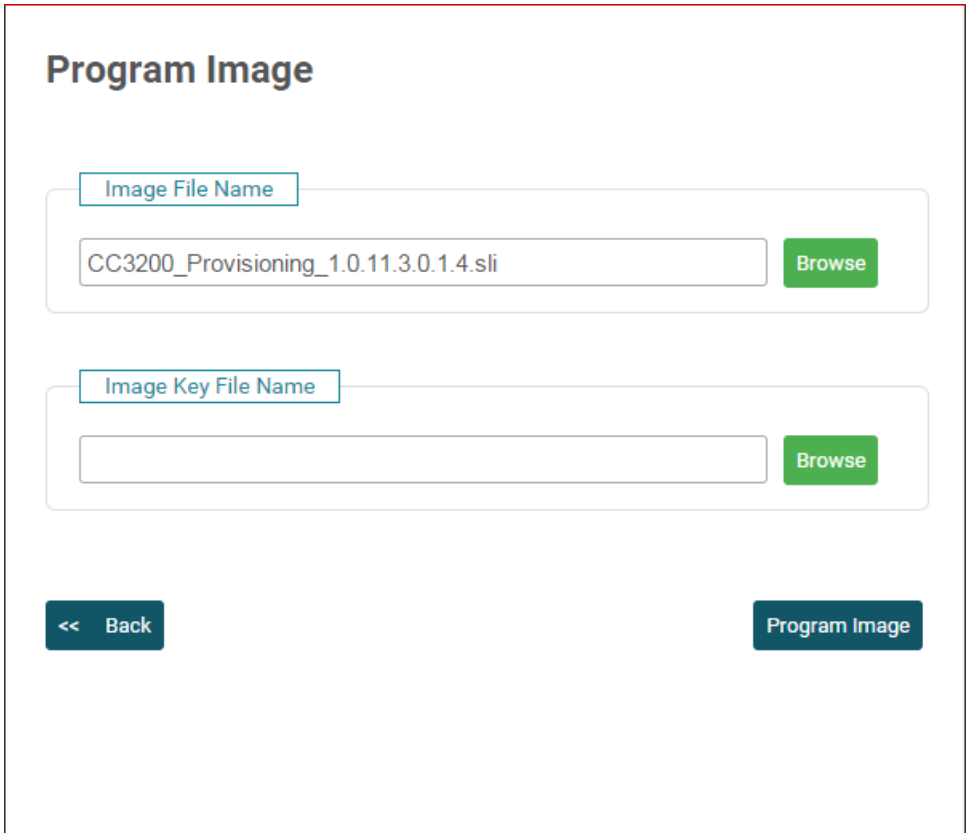


Figure 23. Program Image

6 Command Line

Navigate to the UniFlash install directory:

- For Windows:

```
cd  c:\ti\uniflash_4.0
```

- For Linux:

```
cd /home/YOUR_USER/ti/uniflash_4.0
```

- For Mac OS X:

```
cd /Users/YOUR_USER/ti/uniflash_4.0
```

Use the dslite shell script to send commands in cc3210/cc3220 mode:

- For Windows:

```
dslite.bat --mode cc3120 COMMAND
dslite.bat --mode cc3220 COMMAND
```

- For Linux and Mac OS X:

```
./dslite.sh --mode cc3120 COMMAND
./dslite.sh --mode cc3220 COMMAND
```

6.1 Project Commands

6.1.1 Adding a File or Setting an MCU Image

Basic command:

```
project add_file --name PROJECT_NAME --file MCU_FILENAME.bin --mcu
       -OR-
project add_file --name PROJECT_NAME --file MCU_FILENAME.bin --fs_path /path/filename.ext
```

Required arguments:

--name PROJECT_NAME	Name of the project to use
--file FILENAME	File to add
--fs_path /path/filename.ext -OR- --mcu	File path or name in the SimpleLink file system The file is an MCU image

Optional arguments:

--sign SIGNATURE_FILENAME -OR- --priv PRIVATE_KEY_FILENAME	Signature file used to sign the MCU image file Private key to be used to generate a signature for the file
--flags flag1,flag2,...	File flags, available values: failsafe, secure, nosignaturetest, static, vendor, publicread, publicwrite, nofailsafe, nopublicwrite. The last two are negative flags. The default is that failsafe and publicwrite are true. To disable that, user can set negative flags.
--token TOKEN_NUMBER	File token a 32-bit unsigned integer, used in conjunction with the vendor flag
--max_size MAX_SIZE_IN_BYTES	Maximum size in bytes to be allocated for the file in the SimpleLink file system
--cert CERT_NAME	Certificate file to use (from the device file system); does not change if omitted. Use --cert "" to erase.
--overwrite	Force overwrite in case the file already exists
--project_path PROJECT_PATH	Path to the projects folder
--cfg_json PATH_CFG_JSON_FILE	Full path to the cfg.json file

Notes:

- In case of "--mcu", the security properties and maximum file size are selected automatically in accordance to the project type, but can also be overridden with the "--flags" and "--max_size" options.
- The command prints an error and exits if the file already exists in the project; use "--overwrite" to force an overwrite.

Examples:

Set MCU image:

```
project --name MY_PROJECT --file MCU_FILENAME.bin --mcu
```

Add a file:

```
project --name MY_PROJECT --file MY_TEXT_FILE.txt --fs_path /mydir/myfilename.txt --
flags failsafe,publicwrite
```

Set secure MCU:

- Add certificate file first (certificates always reside in the SimpleLink file system root directory):

```
project --name MY_PROJECT --file MY_CA_CERT.der --fs_path CA_CERT
```

- Set MCU image and sign with private key with project from non-default folder:

```
project --name MY_PROJECT --project_path FULL_PROJECTS_PATH --file MCU_FILENAME.bin --mcu --
priv MY_PRIVATE_KEY_FILENAME.key --
cert CA_CERT
```

6.1.2 Setting the Service Pack

Basic command:

```
project set_sp --name PROJECT_NAME --file SP_FILENAME.bin
```

Required arguments:

--name PROJECT_NAME	Name of the project to use
--file FILENAME	Service pack .bin file

Optional arguments:

--project_path PROJECT_PATH	Path to the projects folder
--cfg_json CFG_JSON_PATH	Full path to the cfg.json file

6.1.3 Programming an Image (From a Project)

Basic command:

```
project program --name PROJECT_NAME
```

Required arguments:

--name PROJECT_NAME	Name of the project to use

Optional arguments:

--port COM<port_number>	COM port to use
--reconfig RECONFIG_FILENAME.json	Apply reconfiguration file on the fly (without saving to project)
-dev	Confirm programming in case the project is in development mode
--project_path PROJECT_PATH	Path to the projects folder
--cfg_json CFG_JSON_PATH	Full path to the cfg.json file
--script_path SCRIPT_PATH	Full path to the power_on/off scripts folder

Notes:

- If the project is in development mode, programming is not allowed without passing the "--dev" argument.

Example:

```
project program --name PROJECT_NAME --project_path PROJECT_PATH --script_path SCRIPT_PATH --
port COM11
```

6.1.4 Setting the Trusted Root-Certificate Catalog

Basic command:

```
project set_certstore --name PROJECT_NAME --file CERT_STORE.lst --sign certstore.lst.signed
```

Required arguments:

--name PROJECT_NAME	Name of the project to use
--file CERT_STORE.lst	Trusted Root-Certificate Catalog file
--sign CERT_STORE.lst.signed	Trusted Root-Certificate Catalog signature file

Optional arguments:

--project_path PROJECT_PATH	Path to the projects folder
--cfg_json CFG_JSON_PATH	Full path to the cfg.json file

Notes:

- Specifing both "--file" and "--sign" as empty switches back to using the default files provided. For example: project set_certstore --name PROJECT_NAME --file "" --sign ""

6.1.5 Exporting a Project

Basic command:

```
project export --name PROJECT_NAME --file EXPORTED_PROJECT.zip
```

Required arguments:

--name PROJECT_NAME	Name of the project to use
--file EXPORTED_PROJECT.zip -OR- --path PATH	Exported project archive file name Write archive to path with time stamped filename

Optional arguments:

--project_path PROJECT_PATH	Path to the projects folder
--cfg_json CFG_JSON_PATH	Full path to the cfg.json file

Notes:

- Using "--path" generates a time stamped filename for the archive, and creates the file in the given path.

6.1.6 Importing a Project

Basic command:

```
project import --file EXPORTED_PROJECT.zip
```

Required arguments:

--file EXPORTED_PROJECT.zip	An exported project archive file

Optional arguments:

--overwrite	Force overwriting existing project with the same name
--project_path PROJECT_PATH	Path to the projects folder
--cfg_json CFG_JSON_PATH	Full path to the cfg.json file

Notes:

- The command refuses to import if a project with the same name exists; use "--overwrite" to force overwriting.

6.1.7 Cloning a Project

Basic command:

```
project clone --name PROJECT_NAME --new NEW_PROJECT_NAME
```

Required arguments:

--name PROJECT_NAME	Name of the project to clone
--new NEW_PROJECT_NAME	New project name for the cloned project

Optional arguments:

--overwrite	Force overwriting existing project with the same name
--with_key	Copy encryption key to the cloned project
--project_path PROJECT_PATH	Path to the projects folder
--cfg_json CFG_JSON_PATH	Full path to the cfg.json file

Notes:

- The command refuses to import if a project with the same name exists; use "--overwrite" to force overwriting.
- If the project has an encryption key, the command does not copy it to the cloned project unless "--with_key" is used.

6.1.8 New Project

Basic command:

```
project new --name NEW_PROJECT_NAME
```

Required arguments:

--name NEW_ PROJECT_NAME	Name of the project to create

Optional arguments:

--device	Device type (“CC3220SF”, “CC3220”, “CC3120”)
--mode	Mode development/production
--description	Project description
--project_path PROJECT_PATH	Path to the projects folder
--cfg_json CFG_JSON_PATH	Full path to the cfg.json file

Example:

```
project new --name PROJ_NAME --device DEVICE_TYPE --mode MODE --description DESC
```

Notes:

- Default values are CC3220 Production without description

6.1.9 Creating an Image (From a Project)

Basic command:

```
project create_image --name PROJECT_NAME --file IMAGE_FILENAME.sli
```

Required arguments:

--name PROJECT_NAME	Name of the project to use
--file IMAGE_FILENAME.sli	Image file name to write

Optional arguments:

--reconfig RECONFIG_FILENAME.json	Apply reconfiguration file on the fly (without saving to project)
--project_path PROJECT_PATH	Path to the projects folder
--cfg_json CFG_JSON_PATH	Full path to the cfg.json file

6.1.10 Reconfiguring a Project

Basic command:

```
project reconfigure --name PROJECT_NAME --file RECONFIG_FILENAME.json
```

Required arguments:

--name PROJECT_NAME	Name of the project to use
--file RECONFIG_FILENAME.json	Apply reconfiguration file

Optional arguments:

--project_path PROJECT_PATH	Path to the projects folder
--cfg_json CFG_JSON_PATH	Full path to the cfg.json file

Notes:

- The reconfiguration is applied and saved into the project.
- reconfig.json supported arguments:
  - “macAddress”
  - “devMac “
  - “apSsid”
  - “apPassword”
  - “deviceName”
  - “staNetwork”
    - “ip”
    - “mask”
    - “gateway”
    - “dns”
    - “dhcp”
  - “apNetwork”
    - “ip”
    - “mask”
    - “gateway”
    - “dns”
    - “startIp”
    - “lastIp”
  - each file should start with simpelink name. Example:
    - {"simplelink":{"devMac": "AA:BB:CC:00:11:22"}}

6.1.11 Listing Available Projects

Basic command:

```
project list
```

Optional arguments:

--project_path PROJECT_PATH	Path to the projects folder
--cfg_json CFG_JSON_PATH	Full path to the cfg.json file

Notes:

- Prints out all available projects
- Unless specified in a cfg.json file, the project directory is situated at:
  - Win 7/10 – c:\Users\the_user\.SLImageCreator\projects
  - Linux – /home/the_user/.SLImageCreator/projects
  - Mac OS X – /Users/the_user/.SLImageCreator/projects

6.1.12 Creating an OTA Archive From the Project

Basic command:

```
project create_ota --name PROJECT_NAME --file TAR_FILE
```

-OR-

```
project create_ota --name PROJECT_NAME --path TAR_PATH
```

Required arguments:

--name PROJECT_NAME	Name of the project to use
--file -OR- --path	File path for destination tar file Destination path – <PROJECT_NAME>.tar file will be created

Optional arguments:

--project_path PROJECT_PATH	Path to the projects folder
--cfg_json CFG_JSON_PATH	Full path to the cfg.json file
--priv PRIVATE_KEY_FILENAME	Full path to the OTA private file name(pem/der)

6.2 Image Commands

6.2.1 Programming an Image

Basic command:

```
image program --file IMAGE_FILENAME.sli
```

Required arguments:

--file IMAGE_FILENAME.sli	Image file name

Optional arguments:

--key KEY_FILENAME	Key file name
--port COM<port_number>	COM port to use
--script_path SCRIPT_PATH	Path to the power_on/off scripts folder
--cfg_json CFG_JSON_PATH	Full path to the cfg.json file

6.3 Tools Commands

6.3.1 Signing a File

Basic command:

```
tools sign --file FILENAME --priv PRIVATE_KEY_FILENAME --out_file OUT_FILENAME
```

Required arguments:

--file FILENAME	File to sign
--priv PRIVATE_KEY_FILENAME	Private key to use for signing
--out_file SIGNATURE_FILENAME	Signature file name

Optional arguments:

--fmt	Signature format: BINARY or BASE64 Defaults to BASE64

6.3.2 Activating Image

Basic command:

```
tools activate --key KEY_FILENAME
```

Required arguments:

--key KEY_FILENAME	Key file name

Optional arguments:

--port COM<port_number>	COM port to use
--script_path SCRIPT_PATH	Path to the power_on/off scripts folder
--cfg_json CFG_JSON_PATH	Full path to the cfg.json file

6.4 Device Commands

6.4.1 Get Device Information

Basic command:

```
device info
```

Optional arguments:

--json	Print to stderr in JSON format
--port COM<port_number>	COM port to use
--script_path SCRIPT_PATH	Path to the power_on/off scripts folder
--cfg_json CFG_JSON_PATH	Full path to the cfg.json file

Notes: Using "--json" prints a JSON object with the information into stderr, allowing a script to easily capture and deserialize the information.

Example:

```
device info --json
```

- Capture stderr
- If the return code is 0, then deserialize the text captured from stderr as a JSON object.

6.4.2 Restore to Factory Image

Basic command:

```
device restore
```

Optional arguments:

--defaults_only	Restore defaults only
--port COM<port_number>	COM port to use
--script_path SCRIPT_PATH	Path to the power_on/off scripts folder
--cfg_json CFG_JSON_PATH	Full path to the cfg.json file

6.5 GUI Configure Commands

6.5.1 Configure GUI

Basic command:

```
gui_cfg
```

Optional arguments:

--project_path PROJECT_PATH	Path to the projects folder
--port COM<port_number>	COM port to use
--script_path SCRIPT_PATH	Path to the power_on/off scripts folder
--cfg_json CFG_JSON_PATH	Full path to the cfg.json file

Notes:

- This command allows setting the com port for GUI.
- Cfg.json supports next parameters:

projectDir	Path to the projects folder
tempDir	Path to the log folder
scriptDir	Path to the power_on/off scripts folder

7 Tools

Click on the Tools button on the welcome page, as shown in Figure 24.

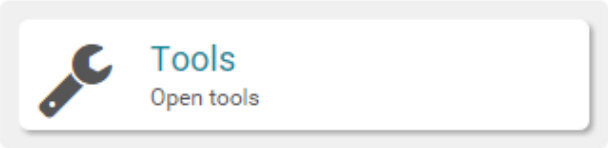


Figure 24. Open Tools

Or click on the Tools button  inside the project.

7.1 Sign File

Using the tool shown in Figure 25, the user can sign a file with a private key and get, as output, a signed file as binary or base-64.

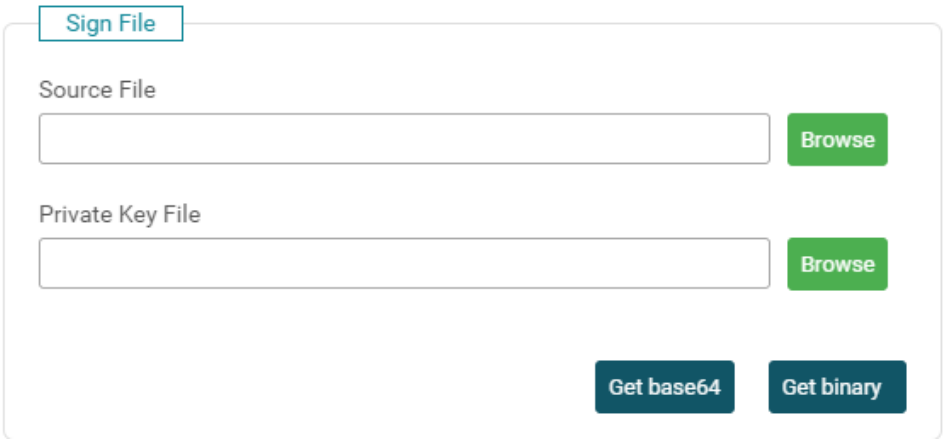


Figure 25. Sign File

7.2 Activate Image

Using the tool shown in Figure 26, the user can activate a programmed encrypted image.

Figure 26. Activate Image

Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

Changes from March 11, 2017 to November 14, 2017 **Page**

- Updated formatting 1
- Updated Create OTA section. 17
- Added Create OTA With a Security Sign section. 17
- Added Create OTA Without a Security Sign section. 17
- Updated Optional Arguments in Create OTA Archive From the Project section. 25

IMPORTANT NOTICE FOR TI DESIGN INFORMATION AND RESOURCES

Texas Instruments Incorporated ('TI") technical, application or other design advice, services or information, including, but not limited to, reference designs and materials relating to evaluation modules, (collectively, "TI Resources") are intended to assist designers who are developing applications that incorporate TI products; by downloading, accessing or using any particular TI Resource in any way, you (individually or, if you are acting on behalf of a company, your company) agree to use it solely for this purpose and subject to the terms of this Notice.

TI's provision of TI Resources does not expand or otherwise alter TI's applicable published warranties or warranty disclaimers for TI products, and no additional obligations or liabilities arise from TI providing such TI Resources. TI reserves the right to make corrections, enhancements, improvements and other changes to its TI Resources.

You understand and agree that you remain responsible for using your independent analysis, evaluation and judgment in designing your applications and that you have full and exclusive responsibility to assure the safety of your applications and compliance of your applications (and of all TI products used in or for your applications) with all applicable regulations, laws and other applicable requirements. You represent that, with respect to your applications, you have all the necessary expertise to create and implement safeguards that (1) anticipate dangerous consequences of failures, (2) monitor failures and their consequences, and (3) lessen the likelihood of failures that might cause harm and take appropriate actions. You agree that prior to using or distributing any applications that include TI products, you will thoroughly test such applications and the functionality of such TI products as used in such applications. TI has not conducted any testing other than that specifically described in the published documentation for a particular TI Resource.

You are authorized to use, copy and modify any individual TI Resource only in connection with the development of applications that include the TI product(s) identified in such TI Resource. NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE TO ANY OTHER TI INTELLECTUAL PROPERTY RIGHT, AND NO LICENSE TO ANY TECHNOLOGY OR INTELLECTUAL PROPERTY RIGHT OF TI OR ANY THIRD PARTY IS GRANTED HEREIN, including but not limited to any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information regarding or referencing third-party products or services does not constitute a license to use such products or services, or a warranty or endorsement thereof. Use of TI Resources may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

TI RESOURCES ARE PROVIDED "AS IS" AND WITH ALL FAULTS. TI DISCLAIMS ALL OTHER WARRANTIES OR REPRESENTATIONS, EXPRESS OR IMPLIED, REGARDING TI RESOURCES OR USE THEREOF, INCLUDING BUT NOT LIMITED TO ACCURACY OR COMPLETENESS, TITLE, ANY EPIDEMIC FAILURE WARRANTY AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

TI SHALL NOT BE LIABLE FOR AND SHALL NOT DEFEND OR INDEMNIFY YOU AGAINST ANY CLAIM, INCLUDING BUT NOT LIMITED TO ANY INFRINGEMENT CLAIM THAT RELATES TO OR IS BASED ON ANY COMBINATION OF PRODUCTS EVEN IF DESCRIBED IN TI RESOURCES OR OTHERWISE. IN NO EVENT SHALL TI BE LIABLE FOR ANY ACTUAL, DIRECT, SPECIAL, COLLATERAL, INDIRECT, PUNITIVE, INCIDENTAL, CONSEQUENTIAL OR EXEMPLARY DAMAGES IN CONNECTION WITH OR ARISING OUT OF TI RESOURCES OR USE THEREOF, AND REGARDLESS OF WHETHER TI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

You agree to fully indemnify TI and its representatives against any damages, costs, losses, and/or liabilities arising out of your non-compliance with the terms and provisions of this Notice.

This Notice applies to TI Resources. Additional terms apply to the use and purchase of certain types of materials, TI products and services. These include; without limitation, TI's standard terms for semiconductor products http://www.ti.com/sc/docs/stdterms.htm), evaluation modules, and samples (http://www.ti.com/sc/docs/sampterms.htm).

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265