CycloneBOOT

# ApplicationImageBuilder

# User Manual

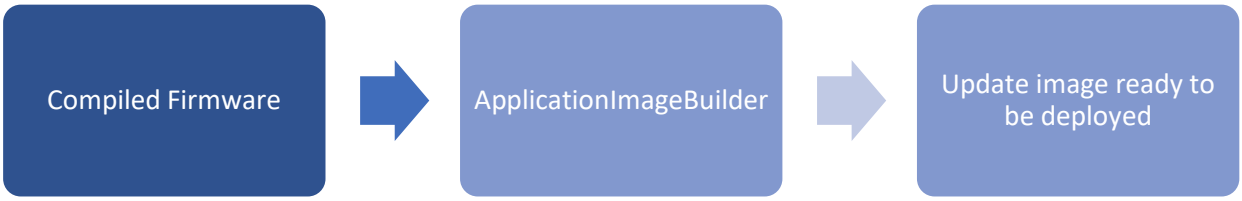
# 1 Table of Contents

# 2 Purpose of the application

Before updating the firmware on your device using CycloneBOOT, it must be “packaged” to a compatible image format. This means in addition to the firmware binary; an additional header and a footer field are added to provide important metadata about the firmware file and information needed to ensure its authenticity and integrity.

Using the ApplicationImageBuilder therefore ensures that your firmware update images are perfectly compatible with CycloneBOOT.

In addition, the ApplicationImageBuilder binary is compiled with the same cryptographic library (CycloneCRYPTO) as used within CycloneBOOT. This also ensures a compatibility between any crypto algorithms that you may choose on both host and device sides.



# 3 How to use the application

When you first execute app_image_builder executable file (contained within the utils/ directory on the downloaded CycloneBOOT package), the following help message is displayed:

```
Usage: app_image_builder [OPTION]...
Generates a firmware update image compatible with CycloneBOOT.
CLI Tool Version: 2.1.0.20221222T1226

  -i, -I, --input=<my_firmware.bin>                    [REQUIRED] Path to
firmware binary.
  -o, -O, --output=<my_firmware_update.img>            [REQUIRED] Path to save
firmware update image.
  -x, -X, --firmware-index=<numeric value>             [OPTIONAL] Custom
firmware index. Default value: 0
  -p, -P, --add-padding                                [OPTIONAL] Generate an
image compatible with the static bootloader in Single Bank Mode.
  --firmware-version=X.X.X                             [OPTIONAL] Version of
firmware update. Obligatory for fallback support.
  --enc-algo=<AES-CBC>                                 [OPTIONAL] Encryption
algorithm used. Supported algorithms: aes-cbc.
  --enc-key=<my_encryption_key>                        [OPTIONAL] Encryption
Key. Optional unless encryption is required.
  --auth-algo=<HMAC-|MD5|SHA256|SHA512|>               [OPTIONAL]
Authentication algorithm used. Supported algorithms: hmac-md5, hmac-sha256,
hmac-sha512.
  --auth-key=<my_auth_key>                             [OPTIONAL]
Authentication Key. Optional unless authentication is required.
  --sign-algo=<ECDSA-SHA256|RSA-SHA256>                [OPTIONAL] Signature
algorithm used. Supported algorithms: ecdsa-sha256, rsa-sha256.
  --sign-key=<my_sign_key.pem>                         [OPTIONAL] Signature
Key. Optional unless signature is required.
  --integrity-algo=<CRC32|MD5|SHA1|SHA256|SHA512>      [OPTIONAL] Integrity
algorithm used. CRC32 by default.
  --verbose                                            [OPTIONAL] Verbose mode.
  -v, -V, --version                                    Show CLI version.
  -h, --help                                           Show the help message.


Example: Create a clear-text firmware image with SHA256 integrity algorithm:
./app_image_builder.exe -i <firmware_binary.bin> -o
<firmware_binary_update.img> --integrity-algo=sha256
```

It also provides an example of usage, generating a clear text image with SHA256 as an integrity check algorithm.

# 4 Generating firmware application images

Following table summarizes the different command-line options and flags used to encrypt (if required) and add a method of verification to the generated application firmware image.

| I. Input and Output Files | II. Firmware Options | III. Cipher Options | IV. Verification Options ("check data") |
|---|---|---|---|
| **Input firmware file**<br><br>**Output firmware application image file** | **Firmware Version**<br><br>**Firmware Index Number**<br><br>**Add VTOR padding/offet in Single Bank mode** | **Cipher algorithm used.**<br>**AES-CBC**<br>*Note: Currently only AES-CBC cipher algorithm is supported by the application. More algorithms will be added in future versions.* | **Integrity**<br>*OR*<br>**Authentication**<br>*OR*<br>**Signature** |
| Required Argument | Optional Argument | Optional Argument | Choose a **single option** |

Step I. Choose input and output files

In this step, user must choose the firmware binary (-input) and path to the generated output application image file.

i. –input OR -i: path to firmware binary (i.e. .bin file). Required
ii. –output OR -o: path to output application image (i.e. .img file) Required

Step II. Firmware options

i. –firmware-version: firmware version. This information is verified when anti-rollback option is enabled. Optional
ii. –add-padding: Adds a padding of 1024 bytes between header and the firmware image. Please refer to section 8.2.1 of the CycloneBOOT User Guide for more information. Not used unless generating a bootable image in "Single Bank" mode.
iii. –firmware-index: Adds a number for the generated application image, which is used to calculate the most recent update image. Please refer to section 8.1 of the CycloneBOOT User Guide for more information. Optional.

**Note**: *In single bank mode, the bootloader requires a bootable image containing the initial application firmware binary on which it can jump if it is a "valid" image. The validation process (integrity check) uses additional information contained within the image. As the image contains this additional information before the firmware binary, there must be some padding after the header data to obtain a correct bootable offset address. The option "--add-padding" inserts the correct amount of padding to the bootable image which holds the initial firmware binary. This image is then written to the flash at the section immediately after the bootloader. This step is only*

*required when an **image is generated for the first time** in single bank mode. All subsequent images can be generated without the "--add-padding" option.*

Step III. Cipher options

i. –enc-key: encryption key (i.e., string of characters) Optional unless you want to encrypt the firmware file.
ii. –enc-algo: encryption algorithm (i.e., string of characters)

**Note**: *current version of ApplicationImageBuilder uses **AES-CBS** cipher algorithm for encryption. Therefore, please supply an encryption key of appropriate length (between 16-32 bytes.)*

Step IV. Verification options ("check data")

CycloneBOOT expects an application image to have **ONE** of the following three verification methods: authentication, signature, or an integrity algorithm.

If neither verification option is specified, a CRC32 checksum is performed by default to ensure the integrity of the firmware data.

i. –auth-algo: authentication algorithm. Optional.
*Supported algorithms*: **"hmac-md5**"," **hmac-sha256","hmac-sha512"**
**ii.** –auth-key: authentication key. (i.e., string of characters). Required if authentication algorithm is chosen
**iii.** –sign-key: signature algorithm. Optional.
*Supported algorithms*: **"ecdsa-sha256"," rsa-sha256"**
**iv.** –sign-key: signature key (i.e., path to a .pem file) Required if signature algorithm is chosen
**v.** –integrity-algo: integrity algorithm. Optional.
*Supported algorithms*: **"md5"," sha1","sha256","sha512"**

Once each 4 steps above are chosen, ApplicationImageBuilder generates a binary image composed of 64-byte header (which contains the length of the binary file, among other information), followed by the binary file itself (encrypted or non-encrypted) terminated by the computed integrity tag or an authentication tag or a signature (check data).

**Note**: *when the image contains an encrypted binary, the initialization vector (IV) used in the encryption process will also be copied in between the header and the binary. This is to allow correct decryption of the encrypted firmware on the device.*

Please refer to "CycloneBOOT User Guide" for a more detailed explanation about the different information contained within a firmware application image.

Information from the header of the application image file is used by CycloneBOOT to ensure:

- that the firmware file doesn't exceed the flash size
- to perform fallback operations
- to verify the version of the firmware file.

The application image verification options ("check data") are used to:

- verify either the integrity
- or the authentication
- or the signature of the application

before proceeding to write the firmware in to the MCU flash and finalizing the update by rebooting the device.

# 5 Reference

| *Flag* | *Description* |
|---|---|
| ***-h / --help*** | Print usage. |
| ***-v / --version*** | Print AppImageBuilder version. |
| ***--enc-key*** | Encryption Key (for encryption algorithm AES-CBC). Could be 16 bytes, 24 bytes or 32 bytes long. |
| ***--auth-algo*** | Authentication algorithm.<br>Accepted options:<br>*"hmac-md5", "hmac-sha256", "hmac-sha512"* |
| ***--auth-key*** | Authentication key. Recommended size 64 bytes (equal to SHA256 block size). |
| ***--integrity-algo*** | Integrity algorithm.<br>Accepted options:<br>*"md5"," sha1", "sha-256", "sha-512"* |
| ***--sign-algo*** | Signature Algorithm.<br>Accepted options:<br>*"ecdsa-sha256", "ecdsa-sha512","ecdsa-md5", "rsa-sha256", "rsa-sha512", "rsa-md5"* |
| ***--sign-key*** | Signature Key. |
| ***--firmware-version*** | Firmware Version. |
| ***--firmware-number*** | Image Number. |
| ***--add-padding*** | Add padding to create correct VTOR offset in Single Bank mode. Required in Single Bank Mode to generate bootable image. |
| ***-i OR --input*** | Input binary file path. |
| ***-o OR --output*** | Output binary image path. |

# 6 Version History

| Revision | Date | Modifications |
|---|---|---|
| 2.0.0 | 2022/01/25 | Initial release |
| 2.1.0 | 2022/12/23 | Incorporated changes related to the Open-Source release of the ApplicationImageBuilder utility. |