
FotaToolkit User Guide_V2.2

EigenCOMM Wireless Microcontroller

Document Description

This document mainly introduces the features of FotaToolkit and also describes how to use it for the purpose of firmware upgrade.

Contents

1.	General Description.....	4
1.1	Conventions.....	4
1.2	Main Procedure.....	4
1.3	Requirements.....	5
2.	How To Set JSON Format File.....	6
2.1	Introduction of Comments.....	6
2.2	Information of the Product.....	6
2.3	Configuration for FOTA Core Version	6
2.4	Configuration for FOTA Basic Capability.....	6
2.5	Configuration for HLS feature(TBD).....	7
2.6	Configuration for Downloading via AT Command.....	8
2.7	Configuration for System Images.....	8
2.7.1	Interpretation Of Image Attributes	9
2.7.2	Supported State Of Image Attributes.....	10
2.8	Configuration for Extensional OTA Storage Zone	10
2.9	Configuration for Storage Devices	11
2.10	Configuration for Customized Features.....	11
2.10.1	To Set Boot Security	11
2.10.2	To Set Update Security.....	13
2.10.3	To Resize the Loading Zone.....	14
2.11	Configuration for Miscellaneous Options.....	14
3.	How To Use The GUI Tool.....	16
3.1	To Set the Config File.....	16
3.2	To Generate An Upgrade File	17
3.2.1	Upgrade File Of Delta Patches.....	17
3.2.2	Upgrade File Of A Full Image.....	18
3.3	To Merge Upgrade Files	19
3.4	To Sign An Upgrade File	20
3.4.1	To Sign An Upgrade File Individually	20
3.4.2	To Sign During Generating/Merging Upgrade File(s).....	20
3.5	To Download An Upgrade File.....	21
3.6	To Deliver Data To An Extensional Storage Device.....	22
3.7	To Read Data Out Of A Storage Zone	23
4.	How To Use The Command Lines	25
4.1	To Create An Upgrade File	25
4.1.1	Upgrade File Of Delta Patches.....	25
4.1.2	Upgrade File Of Full-Image(s).....	26
4.2	To Merge Upgrade Files	26
4.3	To Sign An Upgrade File	27
4.4	To Download An Upgrade File.....	27

4.5	To Outline An Upgrade File	28
4.6	To Convert An Upgrade File To AT Commands	28
5.	FAQ	29
5.1	Question 1	29
6.	Terms	30
7.	Version.....	31

EiGENCOMM CONFIDENTIAL

1. General Description

FOTA (Firmware Over-The-Air) is a kind of software remote upgrade technology, which can provide firmware update services for the terminal devices with networking capabilities. When FOTA is performed, firmware software package will be downloaded from cloud server via networking and be updated for system repair or optimization.

Our products of NB-IOT or CAT.1 series support the differential (diff for short) upgrade of firmware; and users can download a diff file(s) to the terminal device and take it control from a local PC/MCU or cloud server to complete system upgrade.

1.1 Conventions

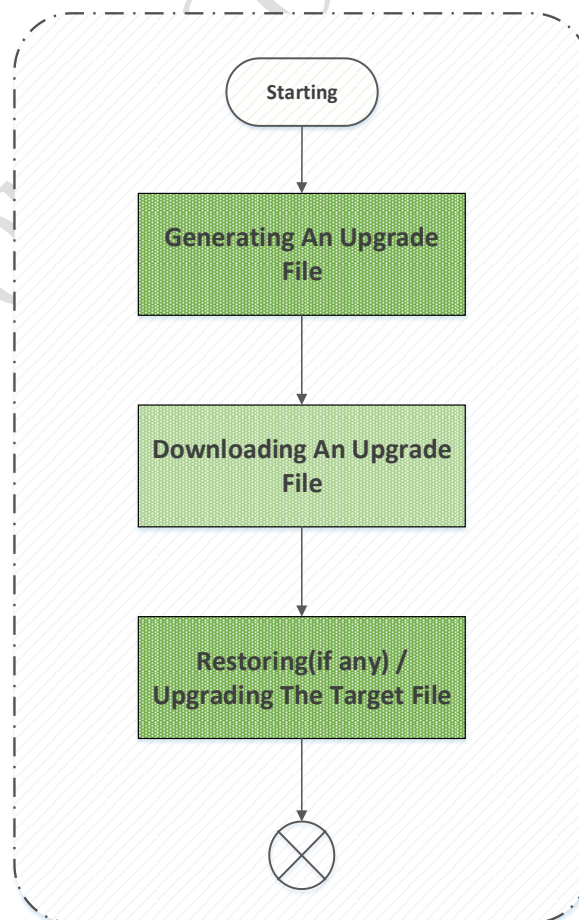
*.par and *.fpar are the default suffix names for the upgrade files of FotaToolkit. Besides, *.par represents the upgrade file of diff(delta) patches while *.fpar represents that of full images.

1.2 Main Procedure

The FOTA service can be subdivided into three steps in terms of the process:

- 1) Generating an upgrade file: to generate a FOTA upgrade file of delta patches or full images.
- 2) Downloading an upgrade file: to download a FOTA upgrade file to the device in a certain way.
- 3) Restoring/Updating an upgrade file: to restore target image(s) (if any) and to update it(them) from a FOTA upgrade file.

Figure 1-1. FOTA main procedure diagram



Before performing the upgrade of device, firstly, users need to generate an upgrade file of delta patches derived from the target and baseline images, using a kind of tool based on a certain differential algorithm, or of full images, then the upgrade file will be downloaded to the device in a certain way. Finally, once the upgrade process is triggered, the target image will be restored (if any) from the diff file and baseline image through the same or matched algorithm, and the baseline image will be updated and replaced!

The tool of FotaToolkit developed by EiGENCOMM mainly focuses on the first two steps of FOTA service mentioned above.

The following sections will introduce the features of this tool and how to use it.

1.3 Requirements

Table 1-1. Runtime Environment

Object	OS system/Platform
FotaToolkit(GUI)	Windows7/7+

2. How To Set JSON Format File

The configuration in JSON format file is essential, which is applied in all features of 'FotaToolkit'. And it can be customized to meet the needs of their products by customers

Taking '[config/ec618.json](#)' for example, all settings in the JSON format file will be introduced in this section.

2.1 Introduction of Comments

The purpose of these comments is to help customers to understand the meaning of some highlighted options and how to set them.

Figure 2-1. Comments in JSON File

```
..{"TheseAreComments!":{
..{"Comment": "'capacity' in 'FotaCapability' is the largest size of all images in one fota processing (0-4MB;1-8MB)!"},
..{"Comment": "'diffMode' in 'FotaCapability' is the flag for handling compressed section(s) in binary file (0-no.zip;1-has.zip)!"},
..{"Comment": "'attr' in 'SysImageMem' must be set with one of these choices('BINPKG/AP/CP/APP/APP2/APP3/SYS'), which are forbidden to be redefined!"},
..{"Comment": "'attr' in 'ExtenOtaMem' must be set with one of these choices('EF/SD'), which are forbidden to be redefined!"},
..{"Comment": "'name' in 'StorageDevice' must be set with one of these choices('AP/CP/EF/SD'), which are forbidden to be redefined!"},
..{"Comment": "'base' in 'StorageDevice' is the global unique starting address, and it can not be modified!"},
..{"Comment": "However, other fields can be redefined to whatever you want, including the name of this config file!"}
..},
```

2.2 Information of the Product

This section will introduce some basic information about the product, and it can be customized to whatever the customers want.

Figure 2-2. Product Info in JSON File

```
"CompanyName": "EiGENCOMM",
"ProductName": "EC618",
```

2.3 Configuration for FOTA Core Version

This section will introduce the version for FOTA core, and different features are included in different cores. The default value for version is matched with the ability of bootloader in the first official release. And it is recommended to double check the ability of bootloader when our customers want to change the version number.

Furthermore, the following features are included in FOTA core V2.x only, which are **not compatible** with core V1.x!

- 1) Support to handle the compressed sections in BIN file;
- 2) Support the maximum capacity (up to 16MB) in one FOTA upgrading procedure;
- 3) Support to set the size of differential block;
- 4) Support the compression for package header length to reduce bits;
- 5) Support the feature of HLS;

Figure 2-3. Core version in JSON File

```
"FotaCoreVer": "2.5",
```

2.4 Configuration for FOTA Basic Capability

This section will introduce some basic capabilities of FOTA.

- 1) **"capacity"**: means the total size of all images in one single FOTA upgrade process (0-4MB/12MB, 1-16MB/48MB), which should be MATCHED with the capability of bootloader;

- 2) **“diffMode”**: means the mode of generating the differential file(0-treat the files as raw binaries, 1-consider the compressed sections in BIN files);
- 3) **“patchMode”**: means the supported patching method in one single FOTA upgrade process(0-only one method between diff and full-image is supported, 1-both of the methods are supported);
- 4) **“hasPkgHlc”**: means the flag to enable/disable the compression of package header size(0-disable, 1-enable);
- 5) **“metaPseg”**: means the meta size to increase/decrease the differential block, the value of which should be multiple of 4 and the max value is 32, and the real size is the value of ‘metaPseg’ * 1KiB;
- 6) **“bkupMemSize”**: means the differential block size in FOTA upgrading, which is set to 32KB that is NOT permitted to be changed in core V1.x and to the size of multiple 32KB in core V2.x;
- 7) **“deltaMemSize”**: means the maximum size of delta file to be stored in FOTA download zone and it can be resized by customers;
- 8) **“avlbRamSize”**: means the maximum available RAM size that can be applied in FOTA upgrading procedure, the value of which is usually set to $2 * \text{“bkupMemSize”}$;
- 9) **“zipAlgorithm”**: is used to set the compressed algorithm used in FOTA procedure;
 - **“zipMeth”**: means the compressed algorithm used in differential file(0-mixed algorithm, 1-only bzip algorithm);
 - **“bzipMode”**: means the buffer size mode during bzip running time(0-large size, 1-small size);

Figure 2-4. FOTA Basic Capability Info in JSON File

```

{
  "FotaCapability": {
    "capacity": 0,
    "diffMode": 1,
    "patchMode": 1,
    "hasPkgHlc": 1,
    "metaPseg": 32,
    "bkupMemSize": 98304,
    "deltaMemSize": 360448,
    "avlbRamSize": 196608,
    "zipAlgorithm": {
      "zipMeth": 0,
      "bzipMode": 0
    }
  }
}

```

2.5 Configuration for HLS feature(TBD)

This section will introduce the some settings for HLS (short for HuLu Suite) feature.

- 1) **“ver”**: means the internal version of HLS suite;
- 2) **“enable”**: means the flag to enable/disable HLS(0-disable, 1-enable);
- 3) **“adjust”**: means the internal flag for HLS;
- 4) **“wkspSize”**: means the maximum workspace(RAM) size for HLS;

Figure 2-5. Settings for AT Command in JSON File

```
[
  {
    "HuluSuite": {
      "ver": 1,
      "enable": 0,
      "adjust": 1,
      "wkspSize": 450560
    }
  },

```

2.6 Configuration for Downloading via AT Command

This section will introduce the meaning of some fields in download procedure via AT command and how to set it for customers' needs.

- 1) **"mode"**: means the data format in downloading procedure via AT command (0-Hex String, 1-Raw Binary);
- 2) **"ate"**: means AT echo is turned off or on(0-off, 1-on)
- 3) **"ver"**: means the version for AT command set(1-NFWUPD, 2-ECOTA);
- 4) **"name"**: means the customized AT command name, whose format should BE COMPLIED WITH that of 'NFWUPD' or 'ECOTA';
- 5) **"pmss"**: means the maximum segment size of the packet, which is LIMITED by that in UE side;
- 6) **"dmwt"**: means the maximum waiting time of the downloading procedure when it is set to raw binary mode;

Figure 2-6. Settings for AT Command in JSON File

```
{
  "FotaAtCmd": {
    "mode": 1,
    "ate": 0,
    "ver": 1,
    "name": "",
    "pmss": 1024,
    "dmwt": 300
  }
},
```

2.7 Configuration for System Images

This section will introduce the settings about system images and the meanings for the fields in the structure.

- 1) **"attr"**: means the type of system images and is set in string format, such as BINPKG/AP/CP/APP/SYS/...;
- 2) **"addr"**: means the starting address for a certain system image;
- 3) **"size"**: means the maximum size of a certain system image, that is the zone size for the image in the layout;
- 4) **"pmeth"**: means the method to upgrade the target image, whose value could be 0 or 1 (0-differential patches, 1-full image);

Now, we will introduce some key points to which we should pay attention. For example, when a file is BINPKG type, it means that all system related images, such as AP/CP/APP(if any)/SYS and etc., have been packed into an archive from whose control header the information of "attr", "addr" and "size" can be straightly got, wherefores, it means those fields of 'SysImageMem' in JSON file will not be used; however, when an image pair of specific kind will be made to a delta file or data in a specific memory zone will be read out, the fields of 'SysImageMem' are essential and customers should set them to proper values based on the

features of product.

Figure 2-7. System Image Memory Zones in JSON File

```

{
  "SysImageMem": [
    {
      "attr": "BINPKG",
      "addr": 0,
      "size": 0
    },
    {
      "attr": "AP",
      "addr": 8536064,
      "size": 2621440
    },
    {
      "attr": "CP",
      "addr": 142606336,
      "size": 524288
    },
    {
      "attr": "APP",
      "addr": 0,
      "size": 0
    },
    {
      "attr": "SYSH",
      "addr": 8396800,
      "size": 4096
    },
    {
      "attr": "BL2",
      "addr": 8470528,
      "size": 131072,
      "pmeth": 1
    },
    {
      "attr": "BL2H",
      "addr": 8400896,
      "size": 4096,
      "pmeth": 1
    }
  ]
}

```

Full Image

2.7.1 Interpretation Of Image Attributes

Table 2-1. Interpretation Of Image Attributes

	<u>INTERPRETATION</u>	<u>NOTE</u>
<i>BINPKG</i>	An archive file containing all the dependent images, including Bootloader, AP,CP,APP(if any) and the secure signature header BIN file(s)	Bootloader might be split into two parts on some chip series, if so, bootloader will be replaced by BL1 and BL2
<i>AP</i>	AP image of the system	

<i>CP</i>	CP image of the system	
<i>APP</i>	A particular customized image type reserved for customers' need	
<i>APP2</i>	A particular customized image type2 reserved for customers' need	
<i>SYSH</i>	Short for SYstem Security Header, a special secure signature header for authenticating/validating the system images, including AP, CP, and APP/APP2 (if any)	<i>For Secure Boot</i>
<i>BL</i>	Bootloader of the system, FOTA will be included as well if it is not split on some chip series	
<i>BL2</i>	The second part of split Bootloader, mainly used for FOTA	
<i>BL2H</i>	a special secure signature header for authenticating/validating BL2 image	<i>For Secure Boot</i>

2.7.2 Supported State Of Image Attributes

Table 2-2. Supported State Of Image Attributes

	<u>EC6x6 (s)</u>	<u>EC618</u>	<u>EC7xx</u>	<u>...</u>
<i>BINPKG</i>		<i>Y</i>	<i>Y</i>	
<i>AP</i>		<i>Y</i>	<i>Y</i>	
<i>CP</i>		<i>Y</i>	<i>Y</i>	
<i>APP</i>	<i>Y</i>	<i>Y</i>	<i>Y</i>	
<i>APP2</i>		<i>Y</i>	<i>Y</i>	
<i>SYSH</i>		<i>Y</i>	<i>Y</i>	
<i>BL2</i>				
<i>BL2H</i>				
<i>...</i>				

2.8 Configuration for Extensional OTA Storage Zone

This section will introduce the settings about extensional OTA storage zone and the meanings for the fields in the structure. And for now, it is supported to download data to external FLASH device only, which could be divided to several memory zones for storing different types of data by customers.

The meanings of "attr", "addr" and "size" are the same as those in 'SysImageMem', and for now, the only value for "attr" field is EF, which means External Flash device.

Figure 2-8. Extensional OTA Memory Zones in JSON File

```

{
  "ExtenOtaMem": [
    {
      "attr": "EF",
      "addr": 2147483648,
      "size": 1048576
    }
  ],

```

2.9 Configuration for Storage Devices

This section will introduce the basic settings for storage device, which is used to get the global starting address of the device for creating the delta file and downloading the data via AT command.

- 1) **"name"**: means the storage device type, which could be "AP", "CP" or "EF", meaning where the image is stored;
- 2) **"base"**: means the starting address of the device;
- 3) **"size"**: means the total size of the device;

Figure 2-9. Settings for Storage Devices in JSON File

```

"StorageDevice": [
  {
    "name": "AP",
    "base": 8388608,
    "size": 4194304
  },
  {
    "name": "CP",
    "base": 142606336,
    "size": 1048576
  },
  {
    "name": "EF",
    "base": 2147483648,
    "size": 8388608
  }
],

```

2.10 Configuration for Customized Features

This section will introduce some settings for customized features which aims to support some special scenarios and all of which are **optional**.

2.10.1 To Set Boot Security

This following settings are applied to the system whose feature of Secure Boot is enabled, and mainly to verify whether the "secure boot header" and the "dependent image(s)" are matched and the signature is legal when the *.par/*.fpar file is creating. The "dependent image(s)" represent for system images and customized image (if any). Generally speaking, the system image mainly includes AP and CP images, and the customized image is called APP image. The signature on the customized image APP (if any) generally has the following two methods:

- a) The customized image APP is signed together with the system image, sharing a signed binary file: SYSH. At this time,

the system image consists of AP, CP, and APP;

- b) The customized image APP is signed independently and has its own signed binary file: APP2. At this time, there are two sets of signed image combo: the customized image APP and its signed file: APP2, and the system image (AP & CP) and its signed file: SYSH.
- c) The customized image APP is signed independently, however, its signed binary file will be combined into the front of the image and treated as a whole. At this time, there are two sets of signed image combo: the customized image and its signed file: APP, and the system image (AP & CP) and its signed file: SYSH.

The meanings of configuration fields in JSON file are introduced as follows:

- 1) **"enable"**: indicates whether to enable the signature verification or not; [*CAUTION: When this field is set to 1("enabled") in JSON file, the information of signature verification will be generated into the *.par file only when the image combination, including the image(s) and it(their) signed binary file, is packed into "BINPKG" files; if only the signed "BIN" file is used to create the *.par file, this field should be set to 0("disabled"), otherwise it will turn to be a failure to create the *.par file due to lack of the dependent images!*]
- 2) **"shaMode"**: represents which SHA algorithm is used for image integrity verification;
- 3) **"bootCombo"**: indicates a combination of images, including system image and its signed binary file, that will be signed. Multiple sets of image combinations could be supported here.
 - **"enable"**: indicates whether to verify the signature of this image combination. If this field is not present, it means "enabled" by default.
 - **"imgs"**: indicates image files that would be signed. And multiple images could be supported here;
 - **"imgh"**: represents the signed binary file for the above images.
 - **"imghFmt"**: represents the format of the signed binary file for customized image APP. If this field is not present, it means the format is the same as that of system signed binary file: SYSH.(1-the format(type 1) of SYSH, 2-the specific format(type 2) for customized image APP).
 - **"keyUri"**: represents the public key file for verifying the signature of the images. If this field is not present or not set, it means the images and their header will be matched only without further verification of the signature.

Figure 2-10. Settings for Boot Security in JSON File

```

{
  "BootSecurity": {
    "enable": 1,
    "shaMode": 256,
    "imghUpgMode": 0,
    "bootCombo": [
      {
        "enable": 1,
        "imgs": "AP, CP",
        "imgh": "SYSH",
        "keyUri": ""
      },
      {
        "enable": 1,
        "imgs": "BL2",
        "imgh": "BL2H",
        "keyUri": ""
      },
      {
        "enable": 0,
        "imgs": "APP",
        "imgh": "APP2",
        "keyUri": ""
      },
      {
        "enable": 0,
        "imgs": "APP",
        "imgh": "APP",
        "keyUri": ""
      }
    ]
  }
}

```

2.10.2 To Set Update Security

These settings are used by customers to adjust some update options when an OTA mission is executed.

- 1) **"antiRollback"**: means that the version rollback will be prevented or not ("enable", "disable");
- 2) **"tokenFromImg"**: indicates the type of image where the secure token or version is located; for now, the token is set in AP image by default, and it may be changed in the future;

Figure 2-11. Settings for Update Security in JSON File

```

{
  "UpdateSecurity": {
    "antiRollback": "disable",
    "tokenFromImg": "AP"
  },

```

2.10.3 To Resize the Loading Zone

The settings aim to support to generate the differential file out of a file pair with a different size, and customers can customize and enable to resize whichever loading zone they want.

- 3) **“enable”**: means the flag to enable/disable the resizing a certain loading zone (0-disable, 1-enable);
- 4) **“attr”**: the type of system images and is set in string format, such as BINPKG/AP/CP/APP/SYS/...;
- 5) **“size”**: means the larger size of a certain system image zone between two released versions (the original size and adjusted one);

Figure 2-12. Settings for Resizing the Loading Zone in JSON File

```

{
  "ResizeMem": [
    {
      "enable": 0,
      "attr": "AP",
      "size": 1261568
    },
    {
      "enable": 0,
      "attr": "CP",
      "size": 360448
    },
    {
      "enable": 0,
      "attr": "APP",
      "size": 0
    }
  ]
},

```

2.11 Configuration for Miscellaneous Options

This section will introduce a few miscellaneous settings for the tool, and for now, the only field is ‘pollRx’, which means the polling time (ms) for serial port receiving.

Figure 2-13. Miscellaneous Settings in JSON File

```

{
  "MiscSetting": {
    "pollRx": 30
  }
}

```

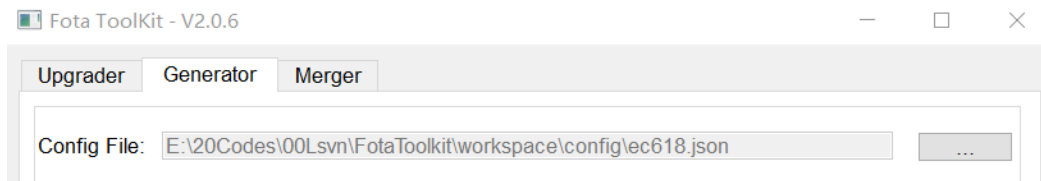
EiGENCOMM CONFIDENTIAL

3. How To Use The GUI Tool

In this section, three important features will be introduced:

- 1) How to use it to generate a diff file;
- 2) How to merge diff files into one;
- 3) How to convert a diff file into AT command lines and upgrade via AT command.

Figure 3-1. FotaToolkit Main GUI



3.1 To Set the Config File

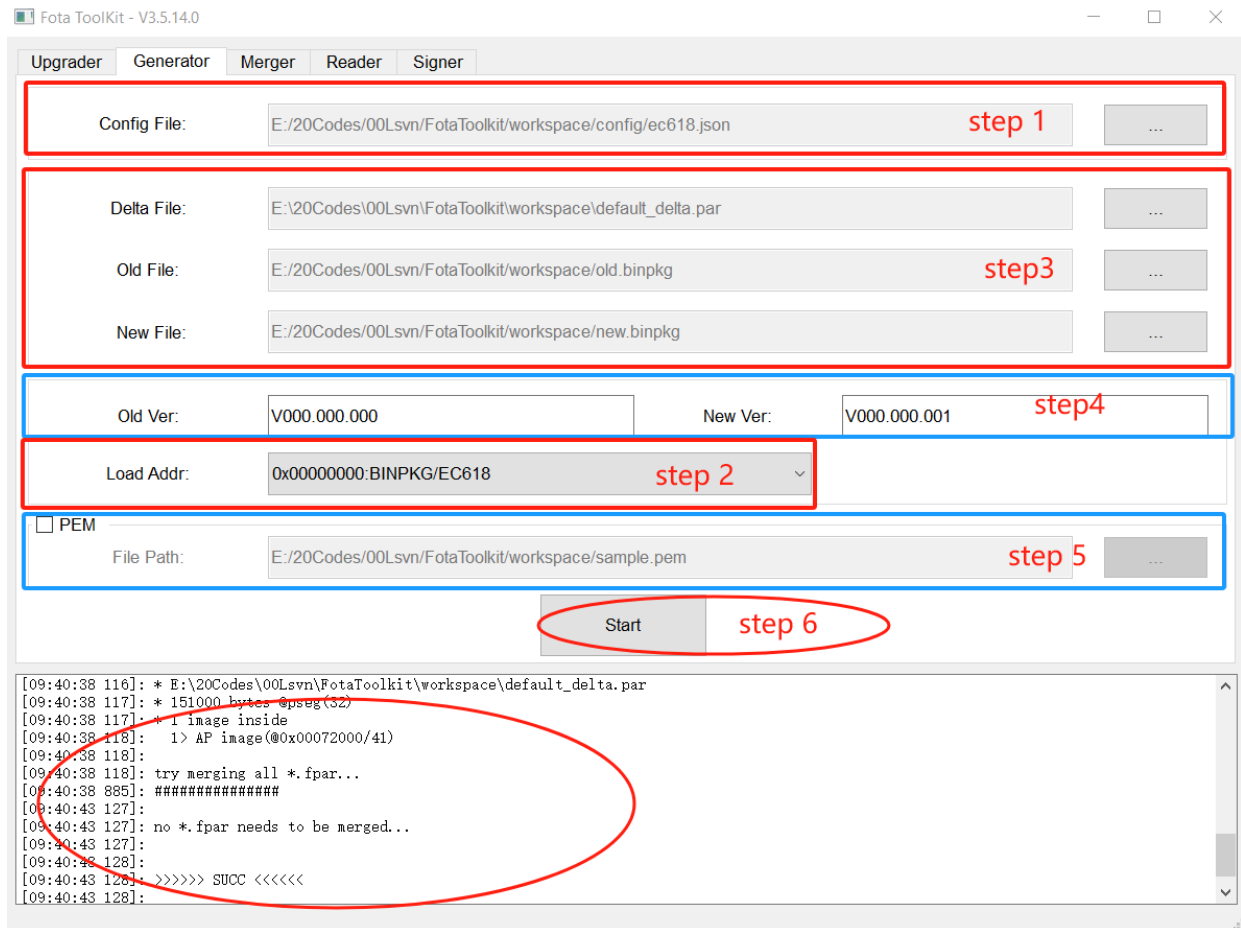
The configuration file in JSON format for this tool is essential, which will be used to generate, merge or download a diff/delta file. What we concern most about the file is the starting address and size of the FOTA and IMAGE areas, which varies per to different products; so do the flags for the features.

For more details, you can refer to Section 2 “How To set JSON Format File”!

3.2 To Generate An Upgrade File

3.2.1 Upgrade File Of Delta Patches

Figure 3-2. GUI for Generating An Upgrade File of Delta Patches

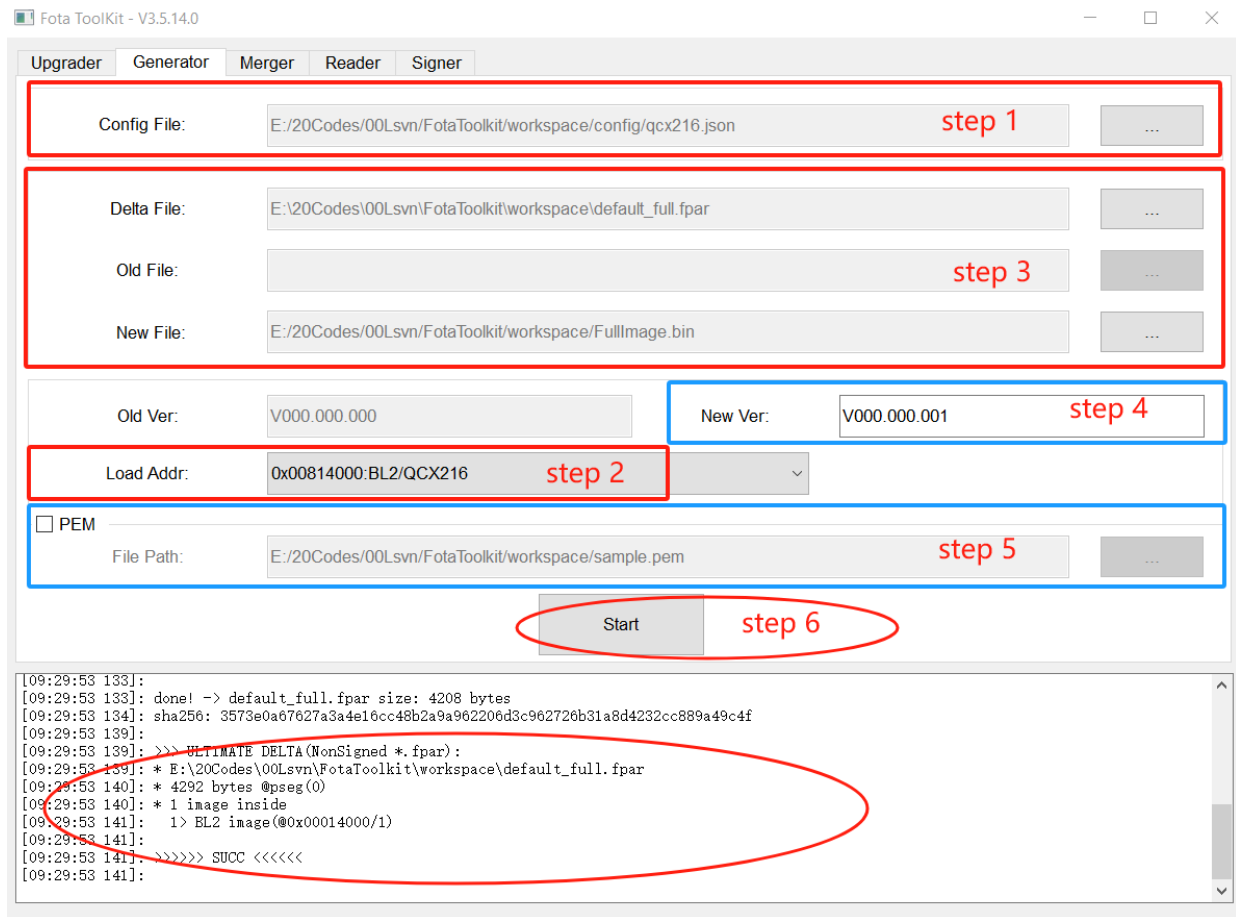


Open the FotaToolkitGUI.exe, and select the 'Generator' tab to generate a diff file in following five steps.

- 1) Select the configuration file of a specific product project, such as config/ec618.json;
- 2) Select the right FLASH configuration information according to the attribute of the baseline or target images, which have been set in the JSON file;
- 3) Set the name and path for the diff file (whose default path is the directory where FotaToolkitGUI.exe is located and default name is 'default_delta.par'), and then select the files of baseline image and target image, from which the diff file derives;
- 4) **This step is optional.** Fill the version of the baseline and target images, whose format is Vmmm.nnn.ppp (m, n, and p are figure 0-9, representing major, minor and the patch version of the image respectively). It should be noted that the version number of the target & baseline image cannot be the same; additionally, **the version number is only for tracking and recording the information of the baseline and target image and it will not be checked and verified;**
- 5) **This step is optional.** Enable the "PEM" option and select the dependent PEM file if a signed upgrade file is expected.
- 6) Press "START" button to create a diff file straightly when the above steps are completed. And the display window will output all the LOG information of the running process. Once it is done, a pop-up will show the execution result: "success" or "failure", and also more detailed information of the ultimate file will also be revealed in the display window.

3.2.2 Upgrade File Of A Full Image

Figure 3-3. GUI for Generating An Upgrade File of Full Images

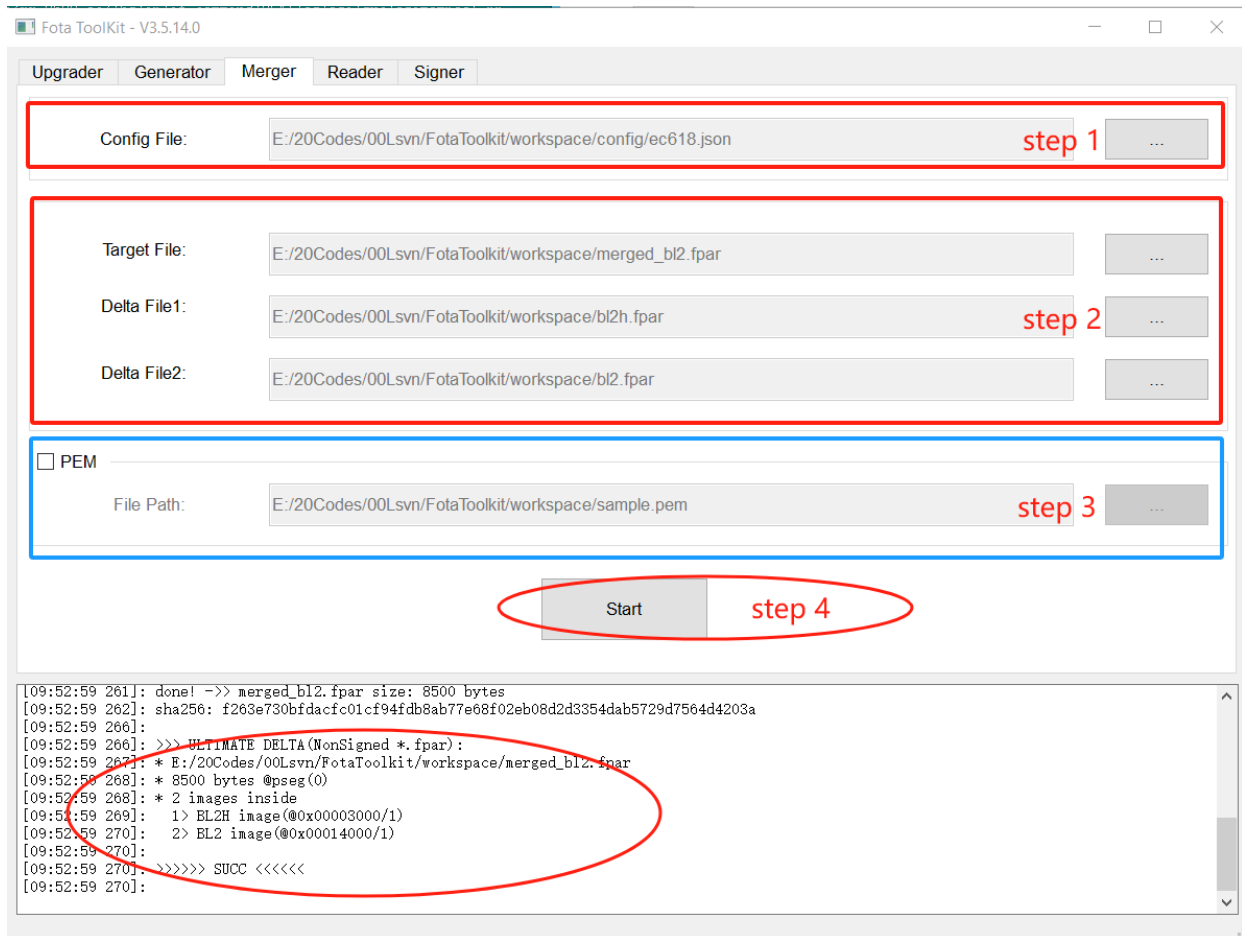


Open the FotaToolkitGUI.exe, and select the 'Generator' tab to generate a diff file in following five steps.

- 1) Select the configuration file of a specific product project, such as config/qcx216.json;
- 2) Select the right FLASH configuration information according to the attribute of the target image (**DO MAKE SURE THE ATTRIBUTE OF FILE AND THAT OF CONFIG ARE MATCHED!**), which have been set in the JSON file;
- 3) Set the name and path for the outputted full-image upgrade file (whose default path is the directory where FotaToolkitGUI.exe is located and default name is 'default_full.fpar'), and then select the target image file, from which the upgrade file comes;
- 4) **This step is optional.** Fill the version of the target image, whose format is Vmmm.nnn.ppp (m, n, and p are figure 0-9, representing major, minor and the patch version of the image respectively). It should be noted that **the version number is only for tracking and recording the information of the target image and it will not be checked and verified**;
- 5) **This step is optional.** Enable the "PEM" option and select the dependent PEM file if a signed upgrade file is expected.
- 6) Press "START" button to create a full-image upgrade file straightly when the above steps are completed. And the display window will output all the LOG information of the running process. Once it is done, a pop-up will show the execution result: "success" or "failure", and also more detailed information of the ultimate file will also be revealed in the display window.

3.3 To Merge Upgrade Files

Figure 3-4. GUI for Merging Upgrade Files



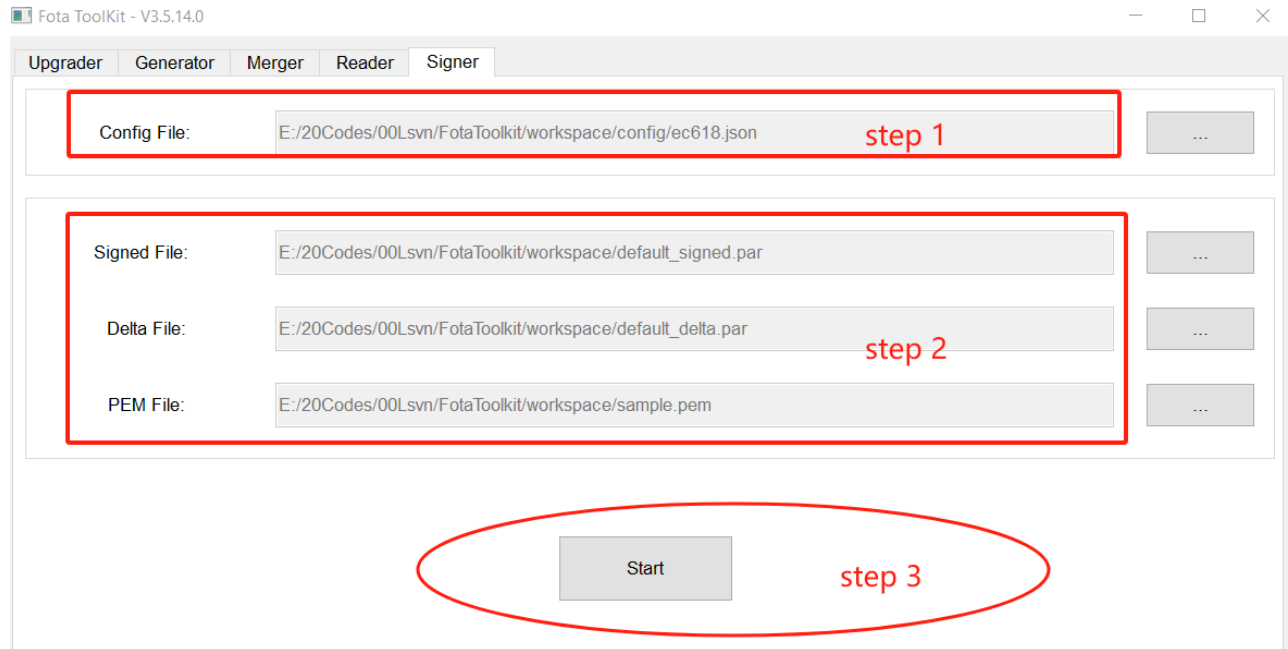
Open the FotaToolkitGUI.exe, and select the 'Merger' tab to merge upgrade files in following three steps.

- 1) Select the configuration file of a specific product project, such as config/ec618.json;
- 2) Set the name and path for the target merged file (whose default path is the directory where FotaToolkitGUI.exe is located and default name is 'default_mdeltapar'), and then select the diff file 1 and file 2 as the input files, such as 'bl2h.fpar' and 'bl2.fpar';
- 3) **This step is optional.** Enable the "PEM" option and select the dependent PEM file if a signed upgrade file is expected.
- 4) Press "START" button to merge the diff files straightly when the above steps are completed. Once it is done, a pop-up will show the execution result: "success" or "failure", and also more detailed information of the ultimate file will also be revealed in the display window.

3.4 To Sign An Upgrade File

3.4.1 To Sign An Upgrade File Individually

Figure 3-5. GUI for Signing An Upgrade File



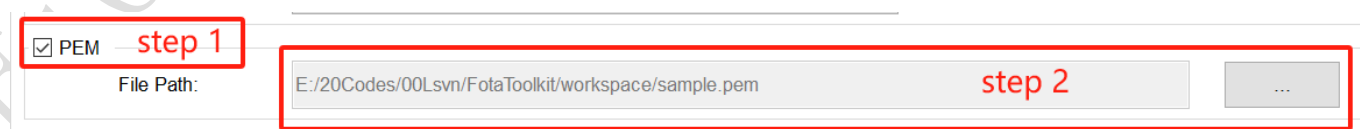
Open the FotaToolkitGUI.exe, and select the 'Signer' tab to sign the upgrade file in following three steps.

- 1) Select the configuration file of a specific product project, such as config/ec618.json;
- 2) Set the name and path for the outputted signed file, and then select the upgrade file to be signed and its dependent PEM file;
- 3) Press "START" button and the running log will be show in the display window.

3.4.2 To Sign During Generating/Merging Upgrade File(s)

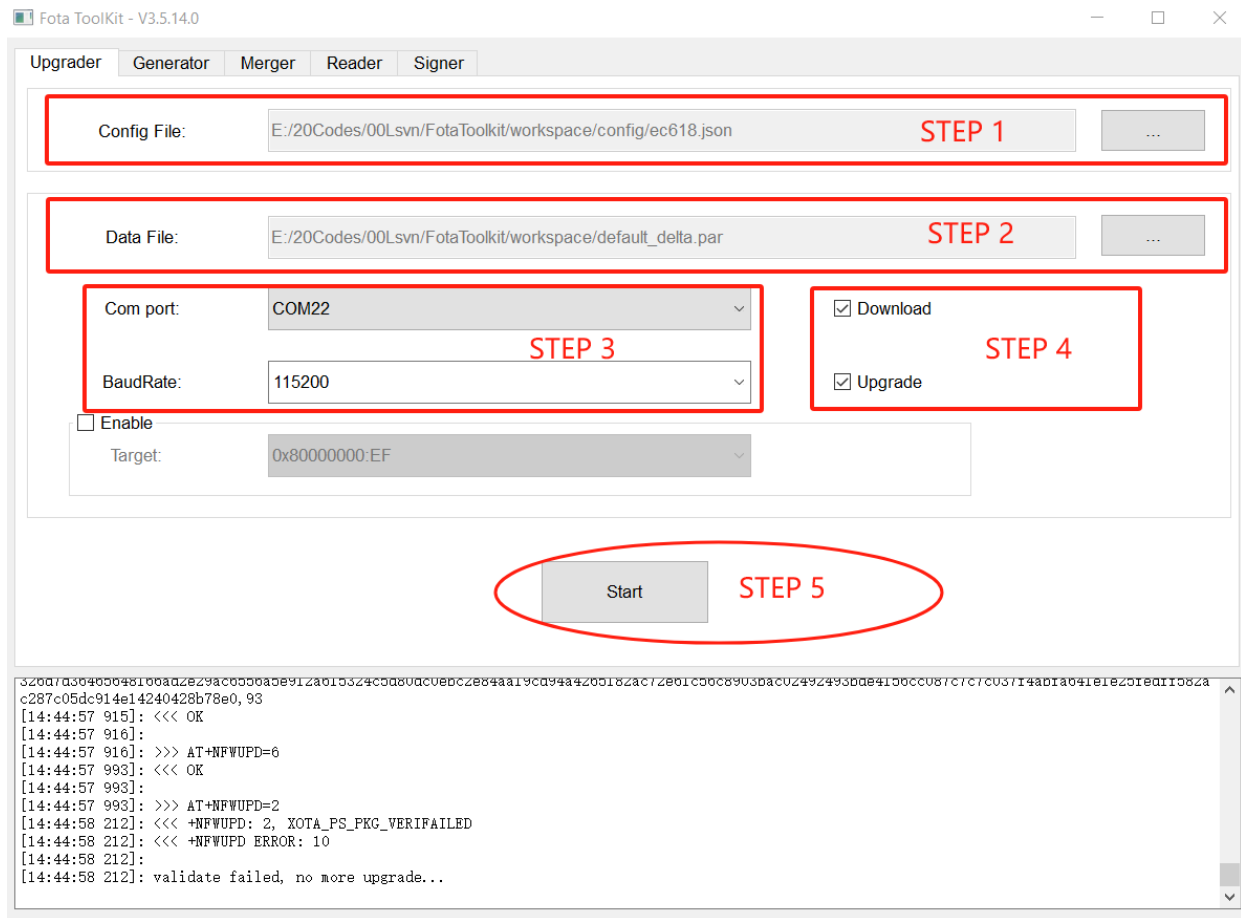
In most cases, signing an upgrade file has been already completed during generating or merging it. Thus, if it is expected, please enable the PEM option and set the right dependent PEM file.

Figure 3-6. Option for Signing When Generating/Merging An Upgrade File



3.5 To Download An Upgrade File

Figure 3-7. GUI for Downloading An Upgrade File via AT Command



Open the FotaToolkitGUI.exe, and select the 'Upgrader' tab to upgrade the device in following five steps.

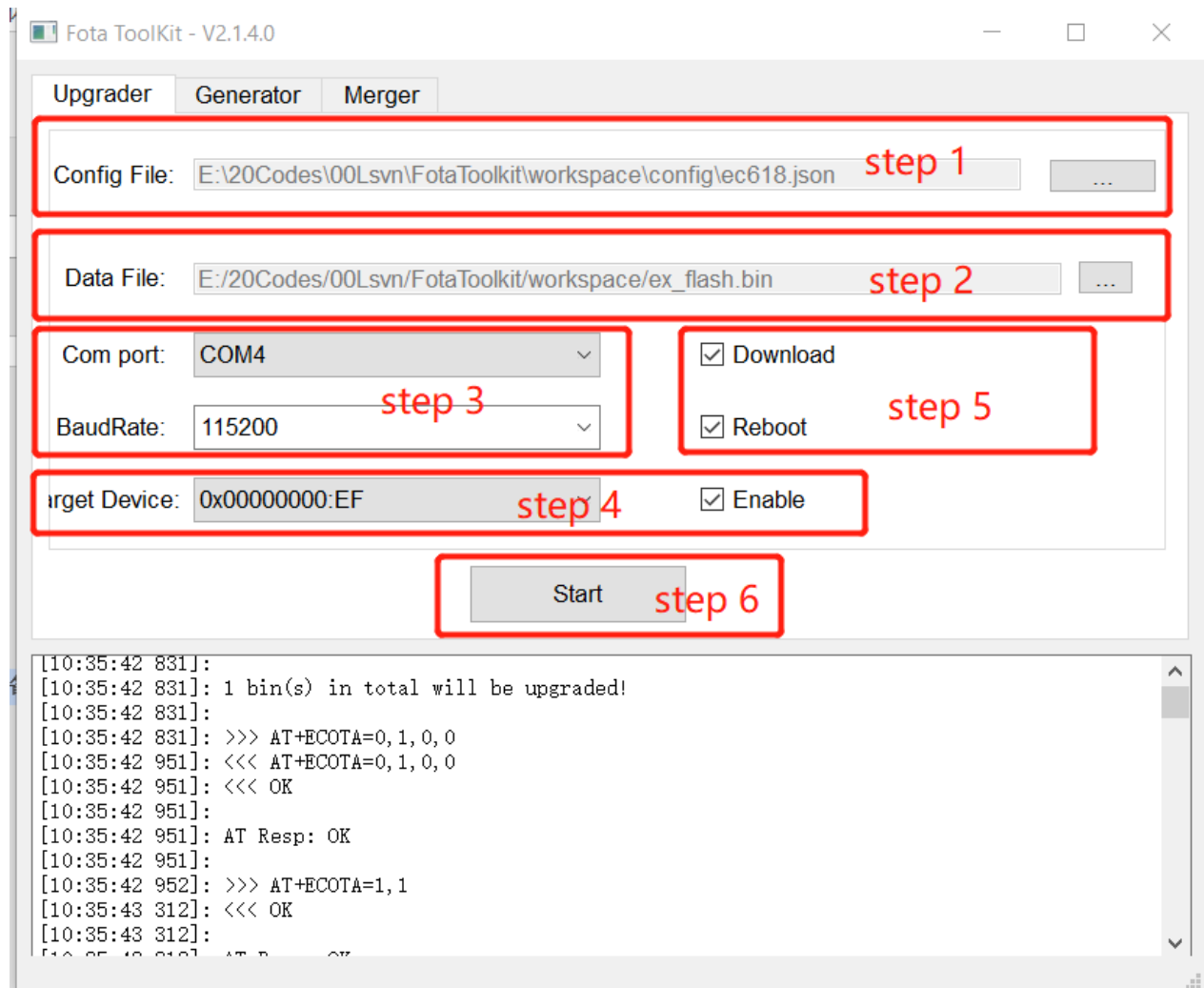
- 1) Select the configuration file of a specific product project, such as config/ec618.json;
- 2) Select the diff file to be upgraded, such as 'delta_st.par' in Figure2-5;
- 3) Select the serial COM port for the AT channel of the connected device, and then set the matched baud rate (9600 and 115200 by default). And the user can input any baud rate they want in the Dropdown List if there is no proper option;
- 4) Before pressing "START" button, the user can choose whether to "Download" the differential file or "Upgrade" the device for their needs. if "Download" option is not set, the window will output the AT command lines converted from the diff file; if "Upgrade" option is not set, the diff file will be downloaded to the device only and the upgrade process, which can be performed whenever it is needed by users later, will not be triggered.
- 5) Press "START" button and the running log will be show in the display window.

※※NOTE※※

AT command is only a kind of way for downloading a diff file to the device from a PC or MCU, and also there is another networking way for downloading from a cloud server, which usually occurs through some application protocols, such as FTP, HTTP(s) and etc.

3.6 To Deliver Data To An Extensional Storage Device

Figure 3-8. GUI for Delivering Data To An Extensional Storage Device via AT command



Open the FotaToolkitGUI.exe, and select the 'Upgrader' tab to upgrade the device in following six steps.

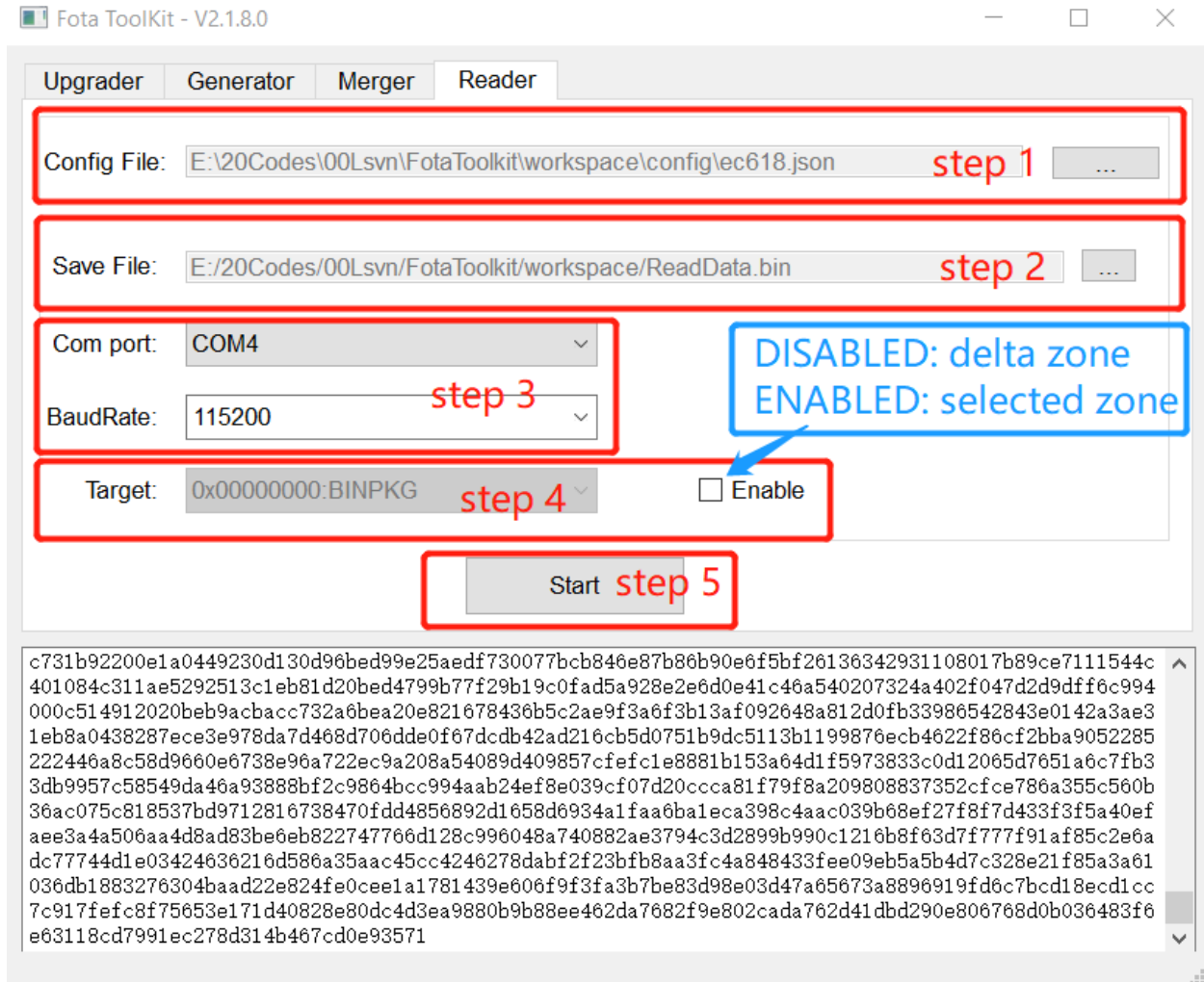
- 1) Select the configuration file, which would be used as the default one once it is selected and the GUI is never closed, of a specific product project, such as config/ec618.json;
- 2) Select the binary file to be delivered, such as 'ex_flash.bin' in Figure2-6;
- 3) Select the serial COM port for the AT channel of the connected device, and then set the matched baud rate (9600 and 115200 by default). And the user can input any baud rate they want in the Dropdown List if there is no proper option;
- 4) "Enable" the feature of delivering to external storage device, then select the target zone whose starting address and total size can be customized in *.json file; however, if the starting address and total size are not changed and still set with the default value, then the file will be delivered to the default external storage zone of the system.

[NOTE] If the "Enable" button is NOT selected; the file will be delivered/downloaded to the FOTA zone of the system.

- 5) Before pressing "START" button, the user can choose whether to "Download" the file or "Upgrade" the device for their needs. if "Download" option is not set, the window will output the AT command lines converted from the file; if "Upgrade" option is not set, the file will be downloaded to the device only.
- 6) Press "START" button and the running log will be show in the display window.

3.7 To Read Data Out Of A Storage Zone

Figure 3-9. GUI for Reading Data Out Of A Storage Zone via AT Command



Open the FotaToolkitGUI.exe, and select the 'Reader' tab to read data out of a specific storage zone in following five steps.

- 1) Select the configuration file, which would be used as the default one once it is selected and the GUI is never closed, of a specific product project, such as config/ec618.json;
- 2) Set the binary file for saving, such as 'ReadData.bin' in Figure2-7;
- 3) Select the serial COM port for the AT channel of the connected device, and then set the matched baud rate (9600 and 115200 by default). And the user can input any baud rate they want in the Dropdown List if there is no proper option;
- 4) If the data in FOTA download zone is to be read out, do not "Enable" the target storage zone; however, if the data in other storage zone (such as AP/CP/EF zone), do "Enable" the target zone and select the zone which you want.

[NOTE] If the "Enable" button is NOT selected; data in the FOTA download zone of the system will be read out.

- 5) Press "START" button and the running log will be show in the display window.

EiGENCOMM CONFIDENTIAL

4. How To Use The Command Lines

The detail usage of command lines can be displayed by executing “FotaToolkit.exe -h” on Windows OR “./FotaToolkit -h” on Linux, and the example of each option is listed as well.

Figure 4-1. Help Information for “FotaToolkit.exe -h” on Windows

```
E:\20Codes\00Lsvn\FotaToolkit\workspace>FotaToolkit.exe -h
[09:56:19 351]: (E) Welcome to EiGENCOMM D-FOTA Toolkit, Today is 2025/03/03!
[09:56:19 356]:
[09:56:19 356]:
[09:56:19 356]:
[09:56:19 356]:
[09:56:19 356]:
[09:56:19 356]:
[09:56:19 372]: (C) Copyright 2020, All Rights Reserved.
[09:56:19 374]: (V) Version(1.2), Built @Mar 1 2025 15:14:52
[09:56:19 374]:
[09:56:19 538]: usage:
[09:56:19 540]: FotaToolkit.exe -a configFile dataBin isDelta [outFile]
[09:56:19 547]: FotaToolkit.exe -d configFile memAttr deltaBin baseBin targetBin [[[baseVer] [targetVer]] [pemFile]]
[09:56:19 554]: FotaToolkit.exe -f configFile memAttr deltaBin targetBin [[targetVer] [pemFile]]
[09:56:19 559]: FotaToolkit.exe -m configFile mergedDelta deltaBin0 deltaBin1 [pemFile]
[09:56:19 563]: FotaToolkit.exe -s configFile signedBin deltaBin pemFile
[09:56:19 567]: FotaToolkit.exe -l configFile deltaBin
[09:56:19 569]: FotaToolkit.exe -u configFile isUpgrade deltaBin serialPort baudRate
[09:56:19 577]: FotaToolkit.exe -e configFile reboot dataBin serialPort baudRate memAttr [memAddr]
[09:56:19 581]: FotaToolkit.exe -r configFile serialPort baudRate memAttr memAddr outFile
[09:56:19 585]: FotaToolkit.exe -h
```

Figure 4-2. Help Information for “./FotaToolkit -h” on Linux

```
xuawang@cfang-OptiPlex-3046:~/FotaToolkit_V3.6.0.0$ ./FotaToolkit -h
[10:07:29 933]: (E) Welcome to EiGENCOMM D-FOTA Toolkit, Today is 2025/03/03!
[10:07:29 933]:
[10:07:29 933]:
[10:07:29 933]:
[10:07:29 933]:
[10:07:29 933]:
[10:07:29 933]:
[10:07:29 933]: (C) Copyright 2020, All Rights Reserved.
[10:07:29 933]: (V) Version(2.0), Built @Mar 1 2025 15:17:56
[10:07:29 933]:
[10:07:29 941]: usage:
[10:07:29 941]: ./FotaToolkit -a configFile dataBin isDelta [outFile]
[10:07:29 941]: ./FotaToolkit -d configFile memAttr deltaBin baseBin targetBin [[[baseVer] [targetVer]] [pemFile]]
[10:07:29 941]: ./FotaToolkit -f configFile memAttr deltaBin targetBin [[targetVer] [pemFile]]
[10:07:29 941]: ./FotaToolkit -m configFile mergedDelta deltaBin0 deltaBin1 [pemFile]
[10:07:29 941]: ./FotaToolkit -s configFile signedBin deltaBin pemFile
[10:07:29 941]: ./FotaToolkit -l configFile deltaBin
[10:07:29 941]: ./FotaToolkit -h
```

In this section, we will take Windows platform as an example to introduce how to use these command lines; and it should be kept in mind that some command lines could be available on Windows only, please refer to the result of option ‘-h’!

4.1 To Create An Upgrade File

4.1.1 Upgrade File Of Delta Patches

[FUNCTION]

To create a *signed or non-signed* delta upgrade file out of the image pair based on a certain of differential algorithm.

[FORMAT]

FotaToolkit.exe -d configFile memAttr deltaBin baseBin targetBin [[[baseVer] [targetVer]][pemFile]]

[USAGE]

```

-d/--diff    Create a delta update file out of a file pair.
             'configFile' is the config file for FotaToolkit [MANDATORY]
             'memAttr' is the attribute of image memory [MANDATORY]
             'deltaBin' is the ultimate delta file derived from old & new files [MANDATORY]
             'baseBin' is the old file that the delta file is based on [MANDATORY]
             'targetBin' is the new file that the image will be upgraded to [MANDATORY]
             'baseVer' is the version number of old binary file [OPTIONAL]
             'targetVer' is the version number of new binary file [OPTIONAL]
             'pemFile' is the pem file for signing the delta file [OPTIONAL]

```

4.1.1.1 Non-signed Delta Upgrade File

[EXAMPLE]

```
FotaToolkit.exe -d config/ec618.json BINPKG delta.par old.binpkg new.binpkg
```

4.1.1.2 Signed Delta Upgrade File

[EXAMPLE]

```
FotaToolkit.exe -d config/ec618.json BINPKG delta.par old.binpkg new.binpkg V000.000.001 V000.000.002 fota_keys.pem
```

4.1.2 Upgrade File Of Full-Image(s)

[FUNCTION]

To create a *signed or non-signed* full-image upgrade file from the target image in a certain format.

[FORMAT]

```
FotaToolkit.exe -f configFile memAttr deltaBin targetBin [[targetVer] [pemFile]]
```

[USAGE]

```

-f/--full    Create a update file including a full target image
             'configFile' is the config file for FotaToolkit [MANDATORY]
             'memAttr' is the attribute of image memory [MANDATORY]
             'deltaBin' is the ultimate update file created from the target file [MANDATORY]
             'targetBin' is the target file that the image will be updated to [MANDATORY]
             'targetVer' is the version number of target binary file [OPTIONAL]
             'pemFile' is the pem file for signing the update file [OPTIONAL]

```

4.1.2.1 Non-signed Full-Image Upgrade File

[EXAMPLE]

```
FotaToolkit.exe -f config/ec618.json BL2 delta.fpar new.bin
```

4.1.2.2 Signed Full-Image Upgrade File

[EXAMPLE]

```
FotaToolkit.exe -f config/ec618.json BL2 delta.fpar new.bin V000.000.002 fota_keys.pem
```

4.2 To Merge Upgrade Files

[FUNCTION]

To merge two *signed or non-signed* upgrade files (containing one or more images in each) into a *signed or non-signed* one

(containing all the images).

[FORMAT]

FotaToolkit.exe -m configFile mergedDelta deltaBin0 deltaBin1 [pemFile]

[USAGE]

```
-m/--merge Merge multiple update files into one.
'configFile' is the config file for FotaToolkit [MANDATORY]
'mergedDelta' is the ultimate merged update file [MANDATORY]
'deltaBin0' is the first update file to be merged [MANDATORY]
'deltaBin1' is the second update file to be merged [MANDATORY]
'pemFile' is the pem file for signing the merged update file [OPTIONAL]
```

4.2.2.1 Merged Upgrade File without Signature

[EXAMPLE]

FotaToolkit.exe -m config/ec618.json mDelta.par delta1.par delta2.par

4.2.2.2 Merged Upgrade File with Signature

[EXAMPLE]

FotaToolkit.exe -m config/ec618.json mDelta.par delta1.par delta2.par [fota_keys.pem](#)

4.3 To Sign An Upgrade File

[FUNCTION]

To (re-)sign a delta/full-image upgrade file into a new one.

[FORMAT]

FotaToolkit.exe -s configFile signedBin deltaBin pemFile

[USAGE]

```
-s/--sign Encipher the update binary file with a pem file
'configFile' is the config file for FotaToolkit [MANDATORY]
'signedBin' is the ultimate signed update file [MANDATORY]
'deltaBin' is the update file to be signed [MANDATORY]
'pemFile' is the pem file for signing the update file [OPTIONAL]
```

[EXAMPLE]

FotaToolkit.exe -s config/ec618.json signed.par delta.par [keys.pem](#)

4.4 To Download An Upgrade File

[FUNCTION]

To download a *signed or non-signed* delta/full-image upgrade file to UE side in AT commands format for upgrading.

[FORMAT]

FotaToolkit.exe -u configFile reboot deltaBin serialPort baudRate

[USAGE]

```

-u/--upgrade Upgrade the delta file over serial port.
'configFile' is the config file for FotaToolkit [MANDATORY]
'reboot' is the flag for rebooting the device or not [MANDATORY]
'deltaBin' is the delta file to be upgraded over COM port [MANDATORY]
'serialPort' is the COM port used to download the delta file [MANDATORY]
'baudRate' is the baud rate of a specified COM port [MANDATORY]

```

[EXAMPLE]

FotaToolkit.exe -u config/ec618.json 1 delta.par COM1 115200

4.5 To Outline An Upgrade File

[FUNCTION]

To outline the details of a delta/full-image upgrade file.

[FORMAT]

FotaToolkit.exe -l configFile deltaBin

[USAGE]

```

-l/--list Outline the delta file in details.
'configFile' is the config file for FotaToolkit [MANDATORY]
'deltaFile' is the update file to be outlined [MANDATORY]

```

[EXAMPLE]

FotaToolkit.exe -l config/ec618.json delta.par

4.6 To Convert An Upgrade File To AT Commands

[FUNCTION]

To convert a delta/full-image upgrade file into a series of AT command lines.

[FORMAT]

FotaToolkit.exe -a configFile dataBin isDelta [outFile]

[USAGE]

```

-a/--atcmd Convert a delta binary file to AT command lines.
'configFile' is the config file for FotaToolkit [MANDATORY]
'dataBin' is the data file to be converted to AT command [MANDATORY]
'isDelta' is the flag of delta file or not [MANDATORY]
'outFile' is the output file of AT command lines [OPTIONAL]

```

[EXAMPLE]

FotaToolkit.exe -a config/ec618.json delta.par 1 atcmd.txt

5. FAQ

5.1 Question 1

Q: What are the meanings of different 'loading address' options in the GUI?

A: As what you see, the option contains 3 pieces of important information: the starting address of image, the attribute of image and the chip family that the image belongs to.

Taking ec618 for example, the meaning of attribute is explained as follows:

- 1) BINPKG is usually used for the binary package combined by multiple images in a certain format, such as the image combination of 'ap_at_command.bin' & 'cp-demo-flash.bin'.
- 2) AP is only used for the image of 'ap_at_command.bin'.
- 3) CP is only used for the image of 'cp-demo-flash.bin'.
- 4) APP is reserved for the customized image, which is not used now!

6. Terms

Table 6-1. Terms and Interpretation

Terms	Interpretation
diff	differential
par	patch archive

7. Version

Version	Date	Author	Comments
1.0	2022-3-22	Xu.Wang	Initiated by XuWang
1.1	2023-2-20	Xu.Wang	New feature for delivering data to an extensional storage device
1.2	2023-4-19	Xu.Wang	New feature for reading data out of a specific memory zone
2.0	2023-11-11	Xu.Wang	New feature for HLS and resizing the loading zone
2.1	2024-05-20	Xu.Wang	Support to generate an upgrade file of full images and to sign it with a PEM file
2.2	2025-03-10	Xu.Wang	Support to executing command lines on Linux