EC200U Series QuecOpen FOTA Application Note

LTE Standard Module Series

Version: 1.0

Date: 2022-03-21

Status: Released

At Quectel, our aim is to provide timely and comprehensive services to our customers. If you require any assistance, please contact our headquarters:

Quectel Wireless Solutions Co., Ltd.
Building 5, Shanghai Business Park Phase III (Area B), No.1016 Tianlin Road, Minhang District, Shanghai 200233, China
Tel: +86 21 5108 6236
Email: info@quectel.com

Or our local offices. For more information, please visit:
http://www.quectel.com/support/sales.htm.

For technical support, or to report documentation errors, please visit:
http://www.quectel.com/support/technical.htm.
Or email us at: support@quectel.com.

Legal Notices

We offer information as a service to you. The provided information is based on your requirements and we make every effort to ensure its quality. You agree that you are responsible for using independent analysis and evaluation in designing intended products, and we provide reference designs for illustrative purposes only. Before using any hardware, software or service guided by this document, please read this notice carefully. Even though we employ commercially reasonable efforts to provide the best possible experience, you hereby acknowledge and agree that this document and related services hereunder are provided to you on an “as available” basis. We may revise or restate this document from time to time at our sole discretion without any prior notice to you.

Use and Disclosure Restrictions

License Agreements

Documents and information provided by us shall be kept confidential, unless specific permission is granted. They shall not be accessed or used for any purpose except as expressly provided herein.

Copyright

Our and third-party products hereunder may contain copyrighted material. Such copyrighted material shall not be copied, reproduced, distributed, merged, published, translated, or modified without prior written consent. We and the third party have exclusive rights over copyrighted material. No license shall be granted or conveyed under any patents, copyrights, trademarks, or service mark rights. To avoid ambiguities, purchasing in any form cannot be deemed as granting a license other than the normal non-exclusive, royalty-free license to use the material. We reserve the right to take legal action for noncompliance with abovementioned requirements, unauthorized use, or other illegal or malicious use of the material.

Trademarks

Except as otherwise set forth herein, nothing in this document shall be construed as conferring any rights to use any trademark, trade name or name, abbreviation, or counterfeit product thereof owned by Quectel or any third party in advertising, publicity, or other aspects.

Third-Party Rights

This document may refer to hardware, software and/or documentation owned by one or more third parties ("third-party materials"). Use of such third-party materials shall be governed by all restrictions and obligations applicable thereto.

We make no warranty or representation, either express or implied, regarding the third-party materials, including but not limited to any implied or statutory, warranties of merchantability or fitness for a particular purpose, quiet enjoyment, system integration, information accuracy, and non-infringement of any third-party intellectual property rights with regard to the licensed technology or use thereof. Nothing herein constitutes a representation or warranty by us to either develop, enhance, modify, distribute, market, sell, offer for sale, or otherwise maintain production of any our products or any other hardware, software, device, tool, information, or product. We moreover disclaim any and all warranties arising from the course of dealing or usage of trade.

Privacy Policy

To implement module functionality, certain device data are uploaded to Quectel's or third-party's servers, including carriers, chipset suppliers or customer-designated servers. Quectel, strictly abiding by the relevant laws and regulations, shall retain, use, disclose or otherwise process relevant data for the purpose of performing the service only or as permitted by applicable laws. Before data interaction with third parties, please be informed of their privacy and data security policy.

Disclaimer

a) We acknowledge no liability for any injury or damage arising from the reliance upon the information.
b) We shall bear no liability resulting from any inaccuracies or omissions, or from the use of the information contained herein.
c) While we have made every effort to ensure that the functions and features under development are free from errors, it is possible that they could contain errors, inaccuracies, and omissions. Unless otherwise provided by valid agreement, we make no warranties of any kind, either implied or express, and exclude all liability for any loss or damage suffered in connection with the use of features and functions under development, to the maximum extent permitted by law, regardless of whether such loss or damage may have been foreseeable.
d) We are not responsible for the accessibility, safety, accuracy, availability, legality, or completeness of information, advertising, commercial offers, products, services, and materials on third-party websites and third-party resources.

About the Document

Revision History

Version	Date	Author	Description
-	2022-01-11	Fei XUE/ Jensen FANG	Creation of the document
1.0	2022-03-21	Fei XUE/ Jensen FANG	First official release

Contents

Table Index

Figure Index

1 Introduction

Quectel EC200U series module supports QuecOpen® solution. QuecOpen® is an embedded development platform based on RTOS, which is intended to simplify the design and development of IoT applications. For more information on QuecOpen®, see ***document [1]***.

This document introduces FOTA upgrade, including FOTA upgrade procedures, upgrade package making, FOTA API and example of Quectel EC200U series module in QuecOpen® solution.

2 FOTA Upgrade Procedures

EC200U series module in QuecOpen solution supports DFOTA and FullFOTA. You can upgrade kernel only, or user application only, or upgrade both kernel and user application. Bootloader cannot be upgraded. The specific upgrade procedures are as follows:

Step 1: Make an upgrade package and upload it to the server. See ***Chapter 3*** for details about package making.

Step 2: The module downloads the upgrade package through the server link address and stores the upgrade package to the file system. Both HTTP and FTP are supported to download the upgrade package.

Step 3: Verify the upgrade package to check the validity of the current upgrade package file.

Step 4: If the upgrade package is verified to be valid, you need to reboot the module, after which the system will automatically perform FOTA upgrade.

Step 5: After the system upgrade is completed, you can query the upgrade result or delete the upgrade package files to free up the file system space.

NOTE

1. After the upgrade package is downloaded, it is generally stored in the file system of the module, so the size of the upgrade package cannot exceed the available space of the file system and a certain space must be reserved. The upgrade package can also be stored in an external SPI Flash or SD card attached to the module.
2. When the upgrade package is stored in internal Flash or external Flash, the storage path supports subdirectories. When the upgrade package is stored in an SD card, the storage path only supports root directory.
3. During FOTA upgrade, the remaining space of Modem partition must be no less than 64 KB, and the remaining space of UFS partition must be no less than 16 KB.

3 FOTA Upgrade Package

3.1. Upgrade Package Making

The current running version of the module is called "old version" and the target version is called "new version" in this document. The firmware package used when making the upgrade package is called "base package".

dtools, the FOTA package making tool, is provided in the directory of *tools/win32* in QuecOpen SDK, as shown in the following figure:

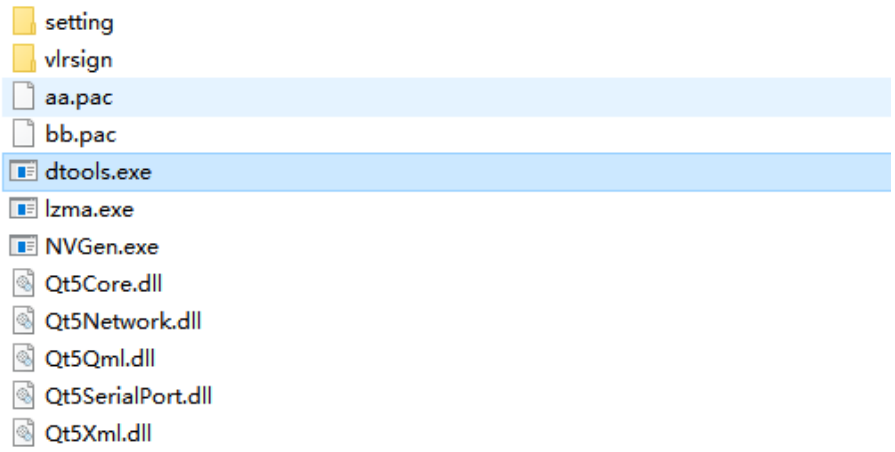


Figure 1: Directory of dtools

Main files of dtools are as below:

- Firmware package of the old version (Firmware package name can be customized), such as *aa.pac*;
- Firmware package of the new version (Firmware package name can be customized), such as *bb.pac*;
- FOTA package generated by dtools (Firmware package name can be customized), such as *output.pack*;
- *dtools.exe*: The program used to make a FOTA package.
- *setting*: A folder used for storing the configurations of making a FOTA package.

You can execute **dtools.exe fotacreate2 [-? /-h /-help]** in CLI or PowerShell (Only Windows 10 supports PowerShell) to query the usage of dtools, as shown in the following figure:

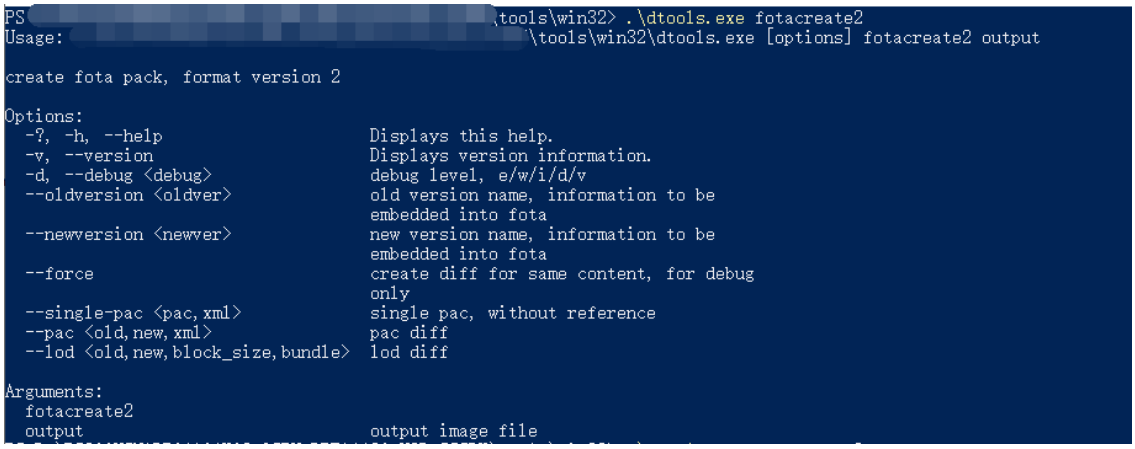


Figure 2: dtools Help Information

3.1.1. DFOTA Package Making

Before making a DFOTA package, you need to prepare firmware package of the old version and firmware package of the new version. This chapter takes the firmware package of the old version *aa.pac* and the firmware package of the new version *bb.pac* as examples, and the specific steps are as follows:

Step 1: Copy *aa.pac* and *bb.pac* to the directory of *tools/win32* in SDK.
Step 2: Open *dtools.exe*.
Step 3: Use **fotacreate2** of dtools as the tool to make upgrade package. Enter the directory of *tools/win32*, open CLI or PowerShell (Only Windows 10 supports PowerShell) and execute **dtools.exe fotacreate2 --pac aa.pac,bb.pac,setting\fota8910.xml output.pack -d v** to make a DFOTA package.

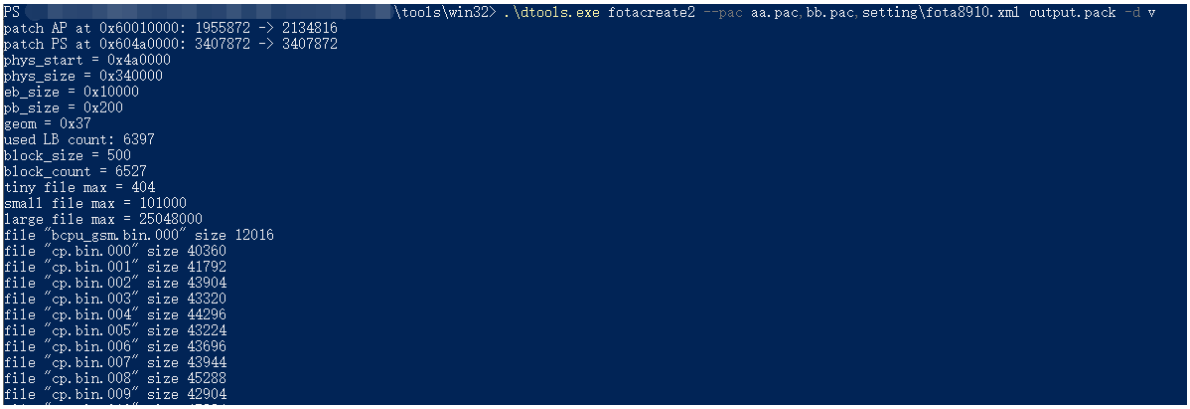


Figure 3: Logs of Making a DFOTA Package

NOTE

1. The base package used to make the DFOTA package must be consistent with the firmware package of the old version.
2. If you do not input **-d v** in CLI or PowerShell (Only Windows 10 supports PowerShell) when you make an upgrade package, no logs will be outputted in terminal windows.

3.1.2. FullFOTA Package Making

Only the firmware package of the new version is required when you make a FullFOTA package. This chapter takes *bb.pac* as an example, and the steps are as follows:

Step 1: Copy *bb.pac* to the directory of *tools/win32* in SDK.
Step 2: Open *dtools.exe*.
Step 3: Use **fotacreate2** of dtools as the tool to make upgrade package. Enter the directory of *tools/win32*, open CLI or PowerShell (Only Windows 10 supports PowerShell) and execute **dtools fotacreate2 --single-pac bb.pac,setting\fota8910.xml out.pac -d v** to make a FullFOTA package.

```
PS                                    \tools\win32> .\dtools fotacreate2 --single-pac bb.pac,setting\fota8910.xml out.pac -d v
patch AP at 0x60010000: 2134816
patch PS at 0x604a0000: 3407872
phys_start = 0x4a0000
phys_size = 0x340000
eb_size = 0x10000
pb_size = 0x200
geom = 0x37
used LB count: 6397
block_size = 500
block_count = 6527
tiny file max = 404
small file max = 101000
large file max = 25048000
file "bcpu_gsm.bin.000" size 12016
file "cp.bin.000" size 40360
file "cp.bin.001" size 41792
file "cp.bin.002" size 43904
file "cp.bin.003" size 43320
file "cp.bin.004" size 44296
file "cp.bin.005" size 43224
file "cp.bin.006" size 43696
file "cp.bin.007" size 43944
file "cp.bin.008" size 45288
```

Figure 4: Logs of Making a FullFOTA Package

NOTE

If you do not input **-d v** in CLI or PowerShell (Only Windows 10 supports PowerShell) when you make an upgrade package, no logs will be outputted in terminal windows.

3.2. Upgrade Package Information View

After making a FOTA package, you can view the FOTA package information through dtools. You can execute **dtools.exe fotacreate2 [-? /-h /-help]** in CLI or PowerShell (Only Windows 10 supports PowerShell) to query the usage of dtools, as shown in the following figure:

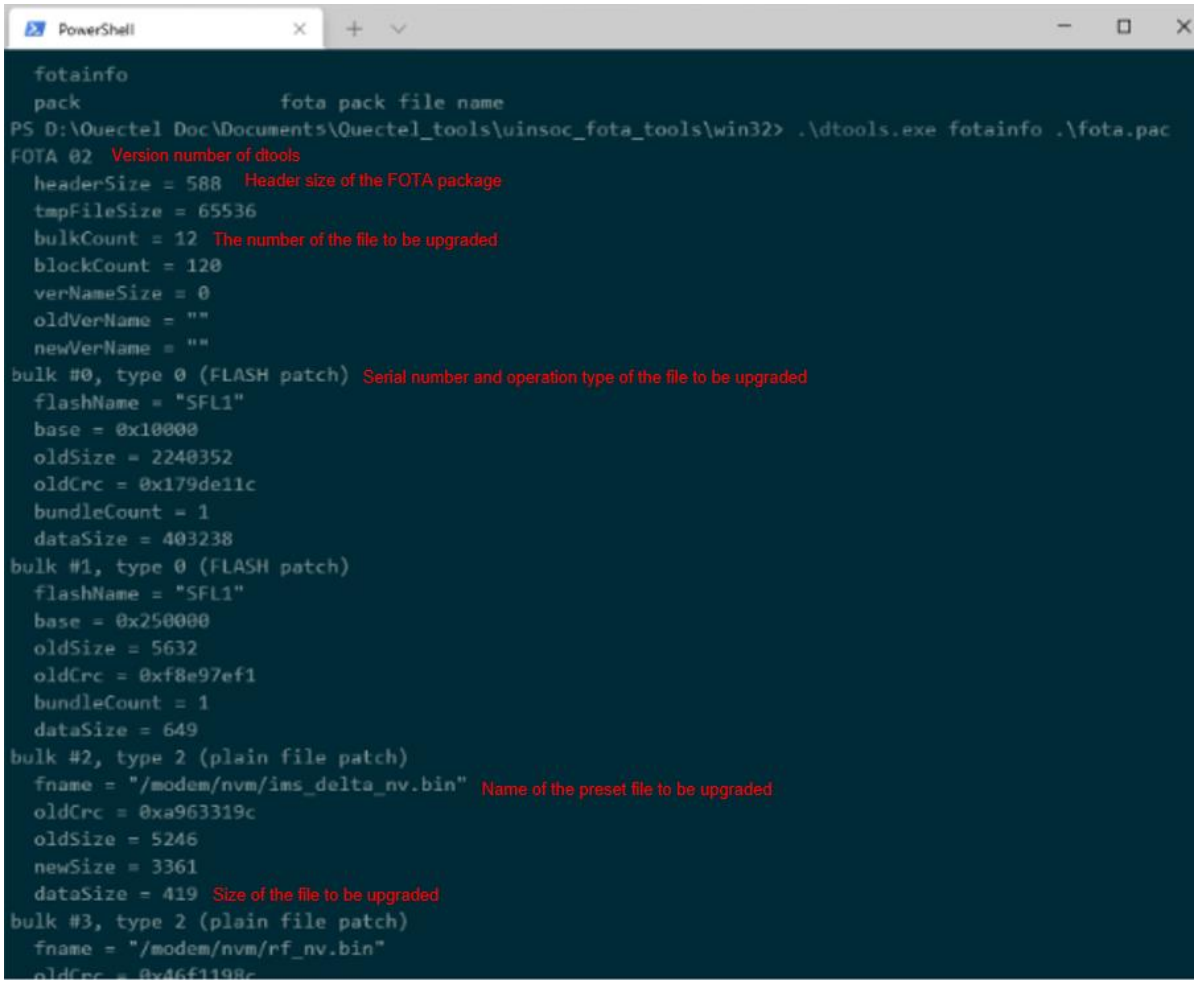
```
PS D:\Quectel_Doc\Documents\Quectel_tools\uinsoc_fota_tools\win32> .\dtools.exe fotainfo
Usage: D:\Quectel_Doc\Documents\Quectel_tools\uinsoc_fota_tools\win32\dtools.exe [options] fotainfo pack

show fota pack information

Options:
  -?, -h, --help       Displays this help.
  -v, --version        Displays version information.
  -d, --debug <debug>  debug level, e/w/i/d/v

Arguments:
  fotainfo
  pack                 fota pack file name
PS D:\Quectel_Doc\Documents\Quectel_tools\uinsoc_fota_tools\win32>
```

This chapter takes the upgrade package *fota.pac* as an example, and the steps are as follows:

Step 1: Copy the completed upgrade package *fota.pac* to the directory of *tools/win32* in SDK.

Step 2: Open *dtools.exe*.

Step 3: Use **fotainfo** of dtools to view the FOTA package information. Enter the directory of *tools/win32*, open CLI or PowerShell (Only Windows 10 supports PowerShell) and execute **.\dtools.exe fotainfo .\fota.pac** to start viewing the information. The FOTA package information is shown in the following figure.



3.3. DFOTA Upgrade Configuration

3.3.1. XML File Configuration Description

fota8910.xml file located in the directory of *tools/win32/setting* configures DFOTA upgrade for the AP side or PS domain of the module, different NV items, preset files, and the application layer. If there is no need to modify, the default settings are applied. If you need to modify the settings, you can modify them according to the annotation in *fota8910.xml* file. You can perform DFOTA upgrade through *fota8910.xml* file in four parts, as follows:

1. Pacflash performs DFOTA upgrade to the data downloaded in Flash.

The following describes the configuration attributes:

id	File ID
flash	Specify the internal flash (SFL1) or the external flash (SFL2)
blocksize	Default: 0x10000
bundleblock	Default: 0
method	Specify the DFOTA upgrade method

diff: differentiate (default)
ignore: ignore
remove: remove
replace: replace

For example:

```
<pacflash id="AP" flash="SFL1" blocksize="0x10000" bundleblock="0"method="diff"/>
```

2. Pacsffs performs DFOTA upgrade to the data downloaded in file system

You can perform DFOTA upgrade to the NV items of *nv.bin* file located in the directory of *nvm* mounted to the Modem partition, or you can specify the mounted partition.

The following describes the configuration attributes:

id	File ID
ebsize	Size of the specified *erase_block* in *partinfo_xx.json*
pbsize	Size of the specified *logic_block* in *partinfo_xx.json*
mount	The specified mounted partition in *partinfo_xx.json*
method	Specify the DFOTA upgrade method *diff*: differentiate (default) *ignore*: ignore *remove*: remove *replace*: replace

For example:

```
<pacsffs id="PS" ebsize="0x10000" pbsize="0x200" mount="/modem"method="diff"/><file
name="nvm/audio_calib.bin" method="ignore"/>
```

3. Pacnv performs DFOTA upgrade to Running NV file.

Determine whether to delete Running NV file and restore Fix NV file.

The following describes the configuration attributes:

nvitem id	ID of NV item. You can find the correspondence between ID and NV file in *nvm_8910.c*.
running nv	Path of name of Running NV.
cleanrunning onchange	Operations of clearing Running NV file. Parameters are as follows: *always*: In any case, this Running NV is always cleared during FOTA upgrade. *never*: Under no circumstances will this Running NV file be cleared during FOTA upgrade *any*: If the Fix NV file corresponding to some Running NV file is modified, the Running NV file will be deleted.

For example:

```
<nvitem id="0x1ba" runningnv="/modemnvm/bt_sprd.bin">
                <cleanrunning onchange="0x1ba"/>
    </nvitem>
```

As mentioned above, when the Fix NV corresponding to "0x1ba" is modified (the two versions of "/modem/nvm/bt_sprd.bin" before and after the upgrade are different), "/modemnvm/bt_sprd.bin" will be deleted.

NOTE

Fix NV is the original NV file, which is not allowed to be modified while the module is running. The Running NV file is a backup of the Fix NV file and can be modified. dtools makes a FOTA package by comparing Fix NV files. Fix NV files are stored in the directory of *nvm* in the Modem partition, and Running NV files are stored in the directory of */modemnvm*. You can get the ID of NV items in *nvm_8910.c* file.

4. Paccpio performs DFOTA upgrade to the preset file.

The following describes the configuration attributes:

id File ID (The ID of the preset file of the module is PREPACK)

method Specify the DFOTA upgrade method

diff: differentiate (default)

ignore: ignore

replace: replace

For example:

```
<paccpio id="PREPACK" method="diff"> </paccpio>
```

3.3.2. Intermediate Upgrade Package Making

During FOTA upgrade, if the upgrade package to be made is too large, FOTA verification may fail, which lead to FOTA upgrade failure or brick the module. In order to avoid the shortage of remaining space in UFS and Modem partitions, the module supports step-by-step FOTA upgrade in the following two methods:

Method 1: Add an intermediate version to make an intermediate upgrade package

Select an intermediate version (*cc.pac*, for example) between the old version package (*aa.pac*, for example) and the new version package (*bb.pac*, for example). First, make the DFOTA package between *aa.pac* and *cc.pac*, and upgrade the module firmware to the intermediate version. Then make the DFOTA package between *cc.pac* and *bb.pac*, upgrade to the target version, and then complete the FOTA upgrade.

Method 2: Modify the upgrade method of the preset file to make an intermediate upgrade package

1. When making the upgrade package, ignore the preset file to reduce the size of the upgrade package. In the XML file, add the method of *ignore* to perform DFOTA upgrade to the preset file, see ***Chapter 3.3.1*** for details. The example is as follows:

```
    </pacnv>
    <paccpio id="PREPACK" method="diff">
        <file name="user/firm" method="ignore"/>
        <file name="user/boot" method="ignore"/>
    </paccpio>
</pacdiff>
```

After modifying the XML file, make the intermediate upgrade package, download the package and perform FOTA upgrade, and upgrade the module firmware to the intermediate version.

NOTE

Before modifying the XML file, you can view the information of the firmware package of the old version through **fotainfo** to determine the preset file that need to be upgraded.

2. After the firmware version upgrades to the intermediate version, you can make a second upgrade package based on the intermediate version and the target version. First, modify the XML file, apply the method of *replace* to perform DFOTA upgrade to the preset file which has been ignored before, then make an upgrade package, and then perform FOTA upgrade after downloading the upgrade package, so as to upgrade the module firmware to the target version.

```
</pacnv>
<paccpio id="PREPACK" method="diff">
    <file name="user/firm" method="replace"/>
    <file name="user/boot" method="replace"/>
</paccpio>
```

NOTE

1. If the remaining space of UFS partition is insufficient, either of the above methods can be used for step-by-step upgrade.
2. You cannot upgrade the files in modem partition separately. Therefore, if the Modem partition space is insufficient, it is recommended to use **Method 1**.
3. The size of the remaining space in the UFS partition can be obtained by calling *int64 ql_fs_free_size(const char *path)*. The size of the remaining space in the Modem partition can be obtained from FOTA-related logs, as described in ***Chapter 5.4***.

4 FOTA API

4.1. Header File

ql_api_fota.h, the header file of FOTA API, is located in the *components\ql-kernel\in* directory. Unless otherwise specified, all header files mentioned in this document are all located in this directory.

4.2. API Overview

Table 1: API Overview

Function	Description
ql_fota_get_pack_name()	Gets upgrade package path information.
ql_fota_image_verify()	Verifies the upgrade package information stored in file system of the module and sets the FOTA upgrade ready flag.
ql_fota_image_verify_without_setflag()	Verifies the upgrade package information stored in file system of the module.
ql_fota_file_reset()	Deletes FOTA package.
ql_fota_get_result()	Gets FOTA upgrade result.
ql_power_reset()	Reboots the module.

4.3. API Description

4.3.1. ql_fota_get_pack_name

This function gets upgrade package path information.

- **Prototype**

```
ql_errcode_fota_e ql_fota_get_pack_name(char *p_pac_file_name,int* length)
```

- **Parameter**

p_pac_file_name:
[In] Storage address of upgrade package path information.

length:
[In] Length of the storage address of upgrade package path information.

- **Return Value**

Result codes. See ***Chapter 4.3.1.1*** for details.

4.3.1.1. ql_errcode_fota_e

FOTA result codes is comprised of relevant component ID and standard error codes, where the component ID is the high 16 bits and the standard error code is the low 16 bits. The component ID of FOTA is 0x8C00; QL_FOTA_ERRCODE_BASE = 0x8C00<<16. The enumeration of FOTA result codes:

```
typedef enum
{
    QL_FOTA_SUCCESS                 = 0,
    QL_FOTA_EXECUTE_ERR             = 502|QL_FOTA_ERRCODE_BASE,
    QL_FOTA_IMAGE_VERIFY_READY_ERR  = 510|QL_FOTA_ERRCODE_BASE
    QL_FOTA_IMAGE_VERIFY_ERR        = 547|QL_FOTA_ERRCODE_BASE

    QL_FOTA_FLAG_SET_ERR            = 548|QL_FOTA_ERRCODE_BASE
    QL_FOTA_BUSY_ERR                = 549|QL_FOTA_ERRCODE_BASE
    QL_FOTA_POINT_NULL_ERR          = 550|QL_FOTA_ERRCODE_BASE
    QL_FOTA_PARAM_INVALID           = 551|QL_FOTA_ERRCODE_BASE
    QL_FOTA_PACKPATH_INVALID        = 552|QL_FOTA_ERRCODE_BASE
    QL_FOTA_ADAPTION_RESET_ERR      = 553|QL_FOTA_ERRCODE_BASE
    QL_FOTA_ADAPTION_SAVE_ERR       = 554|QL_FOTA_ERRCODE_BASE
    QL_FOTA_ADAPTION_DELETE_ERR     = 555|QL_FOTA_ERRCODE_BASE
    QL_FOTA_ADAPTION_LOAD_ERR       = 556|QL_FOTA_ERRCODE_BASE
```

```
    QL_FOTA_ADAPTION_LENGTH_ERR        = 557|QL_FOTA_ERRCODE_BASE
} ql_errcode_fota_e;
```

- **Member**

Member	Description
QL_FOTA_SUCCESS	Successful execution.
QL_FOTA_EXECUTE_ERR	Execution exception.
QL_FOTA_IMAGE_VERIFY_READY_ERR	Failed to verify FOTA package and set the FOTA upgrade ready flag.
QL_FOTA_IMAGE_VERIFY_ERR	Failed to verify FOTA package.
QL_FOTA_FLAG_SET_ERR	Failed to set the FOTA upgrade ready flag.
QL_FOTA_BUSY_ERR	FOTA busy.
QL_FOTA_POINT_NULL_ERR	NULL pointer.
QL_FOTA_PARAM_INVALID	Invalid parameter.
QL_FOTA_PACKPATH_INVALID	Invalid upgrade package storage path entered.
QL_FOTA_ADAPTION_RESET_ERR	Failed to reset upgrade package path information to default.
QL_FOTA_ADAPTION_SAVE_ERR	Failed to save the configuration file for upgrade package path information.
QL_FOTA_ADAPTION_DELETE_ERR	Failed to delete the configuration file for upgrade package path information.
QL_FOTA_ADAPTION_LOAD_ERR	Failed to load the configuration file for upgrade package path information.
QL_FOTA_ADAPTION_LENGTH_ERR	Incorrect cache size for upgrade package storage path.

4.3.2. ql_fota_image_verify

This function verifies the upgrade package information stored in file system of the module and sets the FOTA upgrade ready flag. If the upgrade package is valid, the function is executed successfully and the FOTA upgrade flag is set to *QL_FOTA_READY*, and then *ql_power_reset()* is called to reboot the module and the firmware upgrade starts. If the upgrade package is invalid, an error code is returned.

- **Prototype**

```
ql_errcode_fota_e ql_fota_image_verify(char* PackFileName)
```

- **Parameter**

PackFileName:
[In] Storage path of upgrade package in file system of the module. Default: UFS:fota.pac.

- **Return Value**

Result codes. See ***Chapter 4.3.1.1*** for details.

4.3.3. ql_fota_image_verify_without_setflag

This function verifies the upgrade package information stored in file system of the module. If the upgrade package is valid, the function is executed successfully. If the upgrade package is invalid, an error code is returned.

- **Prototype**

```
ql_errcode_fota_e ql_fota_image_verify_without_setflag(char* PackFileName)
```

- **Parameter**

PackFileName:
[In] Storage path of upgrade package in file system of the module. Default: UFS:fota.pac.

- **Return Value**

Result codes. See ***Chapter 4.3.1.1*** for details.

NOTE

This function only verifies the upgrade package and does not set the FOTA upgrade ready flag. If you need to reboot the module for FOTA upgrade immediately after successful verification, it is recommended to call *ql_fota_image_verify()* directly.

4.3.4. ql_fota_file_reset

This function deletes FOTA package.

- **Prototype**

```
ql_errcode_fota_e ql_fota_file_reset(bool del_image)
```

- **Parameter**

del_image:
[In] If this parameter is set to *TRUE*, the upgrade package is deleted. If the parameter is set to *FALSE*, the upgrade package is not deleted.

- **Return Value**

Result codes. See ***Chapter 4.3.1.1*** for details.

4.3.5. ql_fota_get_result

This function gets FOTA upgrade result.

- **Prototype**

```
ql_errcode_fota_e ql_fota_get_result(ql_fota_result_e *p_fota_result)
```

- **Parameter**

p_fota_result:
[In] The pointer of FOTA upgrade result. See ***Chapter 4.3.5.1*** for details.

- **Return Value**

Result codes. See ***Chapter 4.3.1.1*** for details.

4.3.5.1. ql_fota_result_e

The enumeration of FOTA upgrade results:

```
typedef enum
{
    QL_FOTA_FINISHED = 0,
    QL_FOTA_NOT_EXIST,
    QL_FOTA_READY,
    QL_FOTA_STATUS_INVALID,
    QL_FOTA_PACK_CHECK_ERR,
} ql_fota_result_e
```

- **Member**

Member	Description
QL_FOTA_FINISHED	Upgrade finished.
QL_FOTA_NOT_EXIST	The latest FOTA package is not detected.
QL_FOTA_READY	The latest FOTA package is detected, waiting for system upgrade.
QL_FOTA_STATUS_INVALID	Invalid status.
QL_FOTA_PACK_CHECK_ERR	FOTA package verification failure.

4.3.6. ql_power_reset

This function reboots the module. After the FOTA package passes the verification, it is necessary for you to actively call this function to reboot the module. After reboot, the module automatically enters the upgrade mode and starts to upgrade the firmware. See ***document [2]*** for details about *ql_power_reset()*.

- **Prototype**

```
ql_errcode_power ql_power_reset(ql_ResetMode reset_mode)
```

- **Parameter**

reset_mode:
[In] Reboot mode. See ***Chapter 4.3.6.1*** for details.

- **Return Value**

See ***document [2]*** for details.

4.3.6.1. ql_ResetMode

The enumeration of reboot modes:

```
typedef enum
{
    RESET_QUICK,
    RESET_NORMAL
}ql_ResetMode;
```

- **Member**

Member	Description
RESET_QUICK	Quick reboot.
RESET_NORMAL	Normal reboot.

5 Example

5.1. Example Codes

QuecOpen SDK provides example code files, *components\ql-application\http_fota\inc\fota_http_demo.h* and *components\ql-application\http_fota\fota_http_demo.c*, respectively showing the download and upgrade process of FOTA package.

To use the example code, replace the upgrade package path in the code with link address of the remote server where the upgrade package generated by dtools is located.

5.2. API Calling Procedures

Step 1: Download FOTA package to file system of the module and save it.
Step 2: Call *ql_fota_image_verify()* to verify the upgrade package and set FOTA upgrade ready flag after successful verification. If the parameter is NULL when you call *ql_fota_image_verify()*, the default path UFS:fota.pac is used by system.
Step 3: Call *ql_power_reset()* to reboot the module. After reboot, the module automatically perform FOTA upgrade in Bootloader.
Step 4: After the FOTA upgrade is complete, call *ql_fota_get_result()* to get the FOTA upgrade result.
Step 5: After the FOTA upgrade is complete, call *ql_fota_file_reset()* to delete the FOTA package.

5.3. Verification

The following test procedures are for DFOTA upgrade. You can use the firmware package of the old version *aa.pac* and the firmware package of the new version *bb.pac* to make DFOTA package *output.pack* and store it in HTTP server. The firmware package name can be customized.

Step 1: Download the firmware package of the old version *aa.pac* into the module. After downloading, you can get the current version information through *ql_dev_get_firmware_version()*. The FOTA test routine downloads the DFOTA package *output.pack* from the HTTP server, regardless of whether the firmware package of the old version downloaded is *aa.pac* or not. If the firmware package of the old version downloaded is not *aa.pac*, an error will be returned when the FOTA package is verified, leading to FOTA upgrade failure.

Step 2: Download the DFOTA package *output.pack*. If the DFOTA package is downloaded normally and passes verification, the module will automatically reboot and perform FOTA upgrade.

Step 3: After the upgrade is successful, the FOTA test routine judges whether the FOTA upgrade is completed, and the FOTA test routine automatically deletes FOTA related files if the upgrade is completed.

The following describes the log analysis of the FOTA upgrade example:

1. Red box 1: *http_fota_demo* is started. If you need to use SD card, you need to mount the SD card. See ***document [3]*** for details about mounting.
2. Red box 2: The FOTA package is not detected.
3. Red box 3: The current firmware version information of the module.
4. Red box 4: Length of the FOTA package to be downloaded.
5. Red box 5: FOTA package is downloading.
6. Red box 6: The download is complete and the length of the downloaded FOTA package is shown.
7. Red box 7: Storage path of the current FOTA package.
8. Red box 8: Remaining space size of UFS and Modem partitions after the upgrade package is downloaded.
9. Red box 9: Verification information logs of upgrade package, which can be compared with the firmware package information obtained by **fotainfo**.
10. Red box 10: FOTA package verification failed because the base package used to make the DFOTA package is not consistent with the firmware package of old version.
11. Red box 11: Failed to verify FOTA package and set FOTA upgrade ready flag.

Tick	Level	Description
63770	QOPN/I	[ql_FOTA_http][ql_fota_http_app_init, 784] http fota demo support!
16013	QOPN/I	[ql_FOTA_http][fota_http_app_thread, 728] ql sdmmc mount : 0
16018	QOPN/I	[ql_fota_api][ql_fota_get_result, 418] read PackFileName cfg failed!
16021	QOPN/I	[ql_fota_api][ql_fota_get_result, 426] fup_status 0
16021	QOPN/I	[ql_FOTA_http][fota_http_result_process, 715] fota file not exist
16022	QOPN/I	[ql_FOTA_http][fota_http_app_thread, 740] current version: EC200UCNABR02A01M08
16069	QOPN/I	[ql_FOTA_http][fota_http_init, 428] init file name:[UFS:fota.pack]
16079	QOPN/I	[ql_FOTA_http][fota_http_init, 429] init file size:[0]
16079	QOPN/I	[ql_FOTA_http][fota_http_init, 430] init file stage:[0]
16080	QOPN/I	[ql_FOTA_http][fota_http_app_thread, 752] start [1] times download fota packge
16080	QOPN/I	[ql_FOTA_http][fota_http_info_cfg, 101] init file name:[UFS:fota.pack]
16080	QOPN/I	[ql_FOTA_http][fota_http_info_cfg, 102] init file stage:[0]
16080	QOPN/I	[ql_FOTA_http][fota_http_info_cfg, 103] init file download:[0]
16080	QOPN/I	[ql_FOTA_http][fota_http_info_cfg, 104] init file file_size:[0]
16085	QOPN/I	[ql_FOTA_http][fota_http_info_cfg, 105] init file real file_size:[0]
16085	QOPN/I	[ql_FOTA_http][fota_http_info_cfg, 106] init file is_show:[1]
16085	QOPN/I	[ql_FOTA_http][fota_http_info_cfg, 107] init file last_percent:[0]
16085	QOPN/I	[ql_FOTA_http][fota_http_info_cfg, 108] init file space:[1]
16090	QOPN/I	[ql_FOTA_http][fota_http_get_fd, 529] over write file [UFS:fota.pack]
35893	QOPN/I	[ql_FOTA_http][fota_http_event_cb, 160] response code:200 chunk_encode 0
36122	QOPN/I	[ql_FOTA_http][fota_dload_file_clran, 90] clran write file [UFS:fota.pack] open fd 1027
36123	QOPN/I	[ql_FOTA_http][fota_http_event_cb, 183] content_length:[406846] totalsize=[406846]
36439	QOPN/I	[ql_FOTA_http][fota_http_write_file, 250] write [2535] size
36555	QOPN/I	[ql_FOTA_http][fota_http_write_file, 252] write ret=[2535]
40699	QOPN/I	[ql_FOTA_http][fota_http_write_file, 250] write [5120] size
47165	QOPN/I	[ql_FOTA_http][fota_http_write_file, 252] write ret=[5120]
47168	QOPN/I	[ql_FOTA_http][fota_http_write_file, 266] dload progress:===[1%]===total size[406846] file_size[7655] dload size[7655]
47580	QOPN/I	[ql_FOTA_http][fota_http_write_file, 250] write [5120] size

Figure 5: Logs of Procedures and Status of Download and Upgrade-A

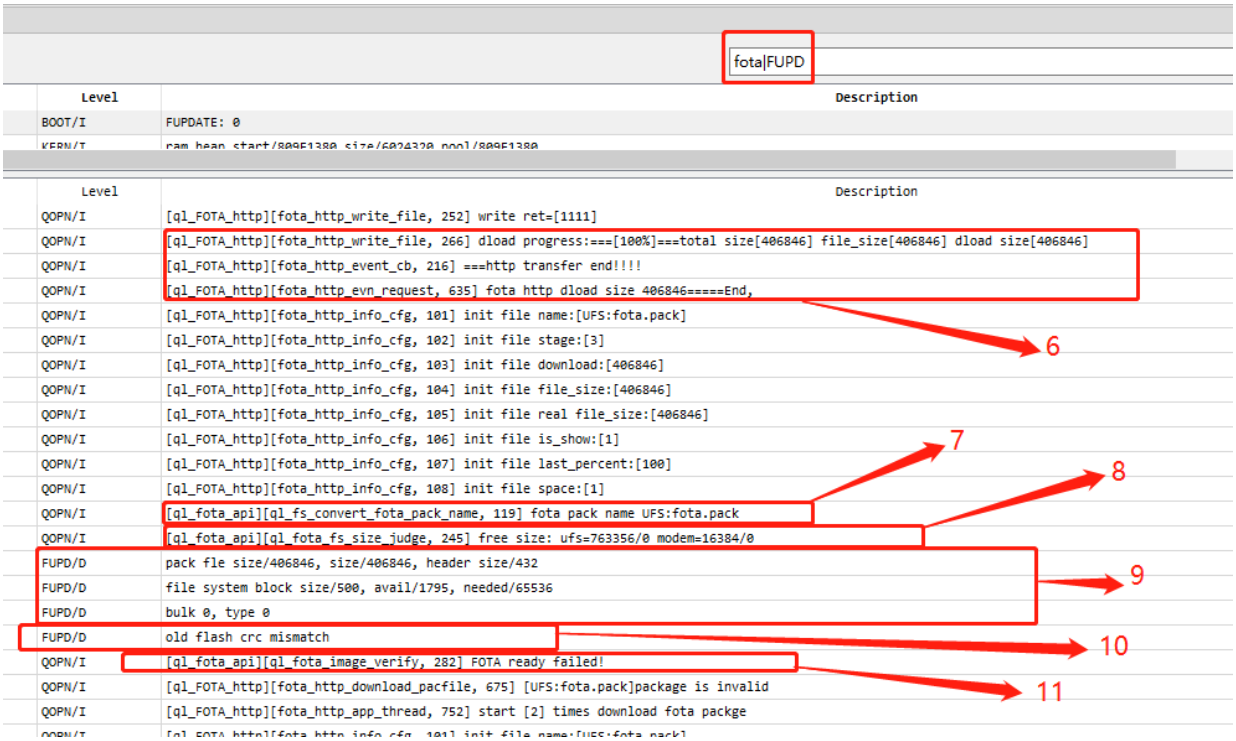


Figure 6: Logs of Procedures and Status of Download and Upgrade-B

5.4. Analysis of FOTA Upgrade Exception Logs

After the FOTA package is verified successfully, you can manually reboot or call API to reboot the module to start FOTA upgrade. If the upgrade package verification fails or FOTA upgrade fails, the module cannot boot, and it will always be in start-up failure cycle. At this point, you can use *coolwatcher* to capture the log and analyze the reasons for the FOTA upgrade failure. Several common failure reasons and solutions are as follows.

1. The remaining space of the system is insufficient.

The exception logs are as follows:

```
FUPD/E : flash new data backup write failed
FUPD/E : plain file new data backup write failed
FUPD/E : plain file write new data failed
FUPD/E : blocked file new data backup write failed
```

Analysis:
The above logs indicate a failure to write to a file, usually because the remaining space of the system is insufficient.

Solution:
Make an intermediate package to perform step-by-step FOTA upgrade. See ***Chapter 3.3.2*** for details.

NOTE

The preset files support FOTA upgrade, and the maximum preset file size supported by the system by default is 64 KB.

2. File information is inconsistent.

The exception log is as follows:

```
FUPD/E : plain file read old data size crc mismatch
```

Analysis:
The above log indicates that information of the file to be upgraded in the current running firmware version of the module is inconsistent with that of the file in DFOTA package. For example, */prepack/example.txt* file is in the base package used to make the DFOTA package, but the content of this file has been modified in the current running firmware version. At this time, during the upgrade verification process, it will be found that the file information is inconsistent, and the DFOTA upgrade cannot be performed.

Solution:
Upgrade files with inconsistent information by ignoring or replacing. Set *method* of this file to either *ignore* or *replace* in *fota8910.xml*. The example is as follows:

```
<pacdiff>
<paccpio id="PREPACK" method="replace">
<file name="prepack/example.txt" method="replace"/>
            </paccpio>
      </pacdiff>
```

3. The remaining space of Modem partition is insufficient.

The exception log is as follows:

```
FUPD/E : blocked file write new data failed
```

Analysis:
The above log indicates that the FOTA upgrade failed because the remaining space of Modem partition is insufficient. The remaining space of Modem partition in the old version and firmware package of the new versions must be no less than 64 KB, otherwise the upgrade may fail.

Solution:
If the remaining space of the Modem partition is not less than 64 KB, the above exception log can still be printed. You can enable savespace feature for the Modem partition upgrade in the *fota8910.xml* file of dtools to reduce the occupation of the Modem partition during the upgrade process. The example is as follows:

```
<pacsffs id="PS" ebsize="0x10000" pbsize="0x200" mount="/modem" method="diff" savespace="y">
```

NOTE

savespace feature is available for the dtools with version of or greater than V1.0-109.

4. The file in AP side or Flash of the firmware package of the old version of the module is inconsistent with that in the base package used to make DFOTA package.

The exception log is as follows:

```
FUPD/E : flash old crc mismatch
```

Analysis:
The above log indicates that the file in AP side (Kernel layer) or Flash of the firmware package of the old version of the module is inconsistent with that in the base package used to make DFOTA package.

Solution:
Replace the base package and re-make the DFOTA package. The base package used to make DFOTA package should be consistent with the firmware package of the old version.

5. The file in CP side of the firmware package of the old version of the module is inconsistent with that in the base package used to make DFOTA package.

The exception logs are as follows:

```
FUPD/E : failed to check blocked file size crc
FUPD/E : failed to check blocked file content size crc
```

Analysis:
The above logs indicate that the file in the Modem partition (CP side) of the firmware package of the old version of the module is inconsistent with the corresponding files in the base package

Solution:
Replace the base package and re-make the DFOTA package. The base package used to make DFOTA package should be consistent with the firmware package of the old version.

6 Appendix References

Table 2: Related Documents

Document Name
[1] Quectel_EC200U_Series_QuecOpen_CSDK_Quick_Start_Guide
[2] Quectel_EC200U_Series_QuecOpen_Booting&Shutdown_Development_Guide
[3] Quectel_EC200U_Series_QuecOpen_SDMMC_API_Reference_Manual

Table 3: Terms and Abbreviations

Abbreviation	Description
AP	Application Processor
API	Application Program Interface
App	Application
FOTA	Firmware Over-the-Air
FTP	File Transfer Protocol
HTTP	HyperText Transfer Protocol
ID	Identifier
IoT	Internet of Things
LTE	Long-Term Evolution
NV	Non-volatile Memory
PS	Packet Switch
RTOS	Real-Time Operating System

SD	Secure Digital Card
SDK	Software Development Kit
SPI	Serial Peripheral Interface