RG520N&RG52xF Series
QuecOpen DFOTA Application Note

5G Module Series

Version: 1.0.0

Date: 2023-05-15

Status: Preliminary

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
-	2022-11-04	Kun GE/ Vern LIN	Creation of the document
1.0.0	2023-05-15	Franklin ZHANG	Preliminary

Contents

About the Document **3**
Contents **4**
Table Index **6**
Figure Index **7**

1 Introduction **8**
1.1. Applicable Modules 9

2 Upgrade Package Generation **10**
2.1. Preparing DFOTA Environment 11
2.1.1. Decompressing DFOTA Tool 11
2.1.2. Installing Dependent Libraries 12
2.1.3. Getting to Know the Firmware Package 12
2.2. Preparing targetfiles.zip 13
2.2.1. Preparing Source targetfiles.zip 13
2.2.2. Preparing Target targetfiles.zip 15
2.2.2.1. Decompressing the Source targetfiles.zip 15
2.2.2.2. Replacing boot Images 16
2.2.2.3. Replacing abl Image 16
2.2.2.4. Replacing rootfs 16
2.2.2.5. Repackaging the Modified targetfiles.zip 18
2.3. Copying targetfiles.zip to the Right Directory 18
2.4. Generating Upgrade Package 19
2.5. Modifying File Permission 19
2.6. Preserving User Configurations 20

3 Restrictions on DFOTA **22**

4 DFOTA Upgrade **24**
4.1. Sending update.zip to Module 25
4.1.1. Sending via ADB 25
4.1.2. Sending via HTTPS 27
4.2. Triggering DFOTA Upgrade with AT Command 28
4.2.1. Calling Upgrade AT Command Manually 28
4.2.2. Calling Upgrade AT Command by Program 29
4.3. Viewing Upgrade Progress and Result 29

5 Troubleshooting **31**

6 FAQ **32**
6.1. How to identify the source targetfiles.zip? 32
6.2. How to output logs after an upgrade fails? 32
6.3. How to view logs in recovery mode? 32
6.4. How to ensure the source targetfiles.zip is generated based on the current firmware? 33

6.5. How does the module handle exceptional power off during DFOTA?..... 33

7 Appendix References 36

Table Index

Table 1: Applicable Modules 9
Table 2: Files and Directories 13
Table 3: Summary of <err> 31
Table 4: Related Documents 36
Table 5: Terms and Abbreviations 36

Figure Index

Figure 1: Process of Generating an Upgrade Package 11
Figure 2: Directory of the Firmware Package 13
Figure 3: Directory Storing the Source targetfiles.zip 14
Figure 4: Configuring Permissions 20
Figure 5: DFOTA Upgrade Flowchart 24
Figure 6: Serial Ports 25
Figure 7: Enabling AT Port on QCOM 25
Figure 8: ADB Device Connected 26
Figure 9: Client/Server Connection Structure 27
Figure 10: Example of Sending Upgrade Package via HTTPS 28
Figure 11: Result Displayed on WebUI (Example) 30
Figure 12: Enter recovery Mode 33

1 Introduction

Quectel RG520N series, RG520F series and RG525F-NA modules support QuecOpen® solution. QuecOpen® is an open-source embedded development platform based on Linux system, which is intended to simplify the design and development of IoT applications. For more information on QuecOpen®, see ***document [1]***.

This document introduces how to implement DFOTA on Quectel RG520N series, RG520F series and RG525F-NA QuecOpen® modules, illustrating the steps of upgrade package generation, the workflow of DFOTA upgrade, information that facilitates troubleshooting and common problems along with their handling.

You can use the *SDX6X-delta-gentools* tool provided in the SDK to build upgrade packages for DFOTA upgrade. The tool uses the *targetfiles.zip* files produced by the build system as inputs. DFOTA is implemented in the recovery mode of the module. A typical DFOTA upgrade procedure is as follows:

1. Determine the source *targetfiles.zip* and target *targetfiles.zip*;
2. Use the source *targetfiles.zip* and target *targetfiles.zip* to generate upgrade package *update.zip*;
3. Download the upgrade package to the module, which caches it to the NAND flash data caching area;
4. Execute related AT commands to trigger DFOTA upgrade;
5. The module enters recovery mode to perform DFOTA upgrade;
6. Reboot the module to normal mode after the upgrade is completed.

NOTE

For detailed information of *SDX6X-delta-gentools*, please contact Quectel Technical Support.

1.1. Applicable Modules

Table 1: Applicable Modules

Module Series	Module
RG520N	RG520N series
RG52xF	RG520F series
	RG525F-NA

NOTE

RG520N series, RG520F series and RG525F-NA modules only support DFOTA upgrade under the scheme of 4 + 4 MCP storage and single partition.

2 Upgrade Package Generation

An upgrade package is a file package that stores upgrade data, which resembles the data used for system patching. Such upgrade data records the modifications made to the kernel, rootfs, modem and other minor partitions, namely the differences between the old and new images of the partitions before and after modification. The upgrade data is downloaded into the NAND flash of the module through network or ADB tool for implementing DFOTA on the module.

It is recommended to generate upgrade packages on Ubuntu 14.04 with the *SDX6X-delta-gentools* tool. Specifically, copy the *targetfiles.zip* of the source version and the *targetfiles.zip* of the target version to the *v1* and *v2* directories of *SDX6X-delta-gentools* respectively, and then execute *update_gen.sh* script to generate the required upgrade package *update.zip* (See ***Chapter 2.4*** for details).

In *SDX6X-delta-gentools*:

- *v1* stores the *targetfiles.zip* corresponding to the firmware image version (the source version) that is running in the module;
- *v2* stores the *targetfiles.zip* corresponding to the modified firmware version (the target version) into which the currently running firmware is to be upgraded.

The t*argetfiles.zip* contains all the image files of partitions that support DFOTA upgrade. The image files of kernel, rootfs and modem are *sdxlemur-boot.img*, *sdxlemur-sysfs.ubi* and *NON-HLOS.ubi* respectively. The directory structure of *targetfiles.zip* decompressed is as follows:

```
targetfiles.zip
 ├──── BOOT
 ├──── BOOTABLE_IMAGES   #boot.img for kernel
 ├──── DATA
 ├──── FULL-IMAGES
 ├──── IMAGES
 ├──── META
 ├──── MULTIFOTA    #others partition image
 ├──── OTA
 ├──── RADIO
 ├──── RECOVERY
 └──── SYSTEM            #system rootfs folder
    └────firmware        #modem image folder
```

The process of generating a DFOTA upgrade package is as follows:

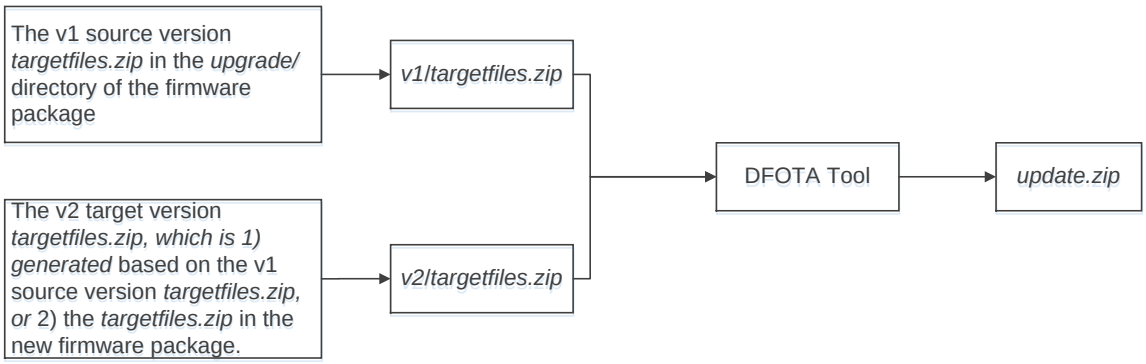

Figure 1: Process of Generating an Upgrade Package

2.1. Preparing DFOTA Environment

- Host: Ubuntu 14.04 64-bit system
- DFOTA tool: *SDX6X-delta-gentools.tar.gz*
- Firmware package: the firmware package with *targetfilse.zip*

2.1.1. Decompressing DFOTA Tool

Run the command **tar -xzvf SDX6X-delta-gentools.tar.gz .** to decompress the DFOTA tool. After the decompression, the following directory is presented:

```
├─── bsdiff
├─── common.py
├─── edify_generator.py
├─── fs_config
├─── imgdiff
├─── libbz2.so.0
├─── libbz2.so.0.0.0
├─── multi_fota.py
├─── ota_from_target_files
├─── ql_global.py
├─── quectel_chmod_filelist.txt
├─── quectel_fullota_filelist.txt
├─── quectel_multifota_filelist.txt
├─── quectel_noota_filelist.txt
├─── quectel_project_filelist.txt
├─── ubi_reader
```

```
├──── update_gen.sh   # the script to generate update.zip
├──── v1              # the folder to store the source targetfiles.zip
└──── v2              # the folder to store the target targetfiles.zip
```

NOTE

There are several configurable txt files in *SDX6X-delta-gentools*. Among them:

1. The *quectel_fullota_filelist.txt* specifies the files to be substituted as a whole for the upgrade instead of to be patched (Currently, such files can only be specified using their full paths, and the operation of specifying an entire folder of files using the folder name is not supported);
2. The *quectel_noota_filelist.txt* specifies the files that do not need upgrade;
3. The *quectel_chmod_filelist.txt* configures the permissions of files or directories after an upgrade;
4. The *quectel_multifota_filelist.txt* configures the upgraded image of the minor partitions. It cannot be changed arbitrarily. If there is a need for modification, please contact Quectel Technical Support for technical evaluation;
5. The *quectel_project_filelist.txt* configures the upgraded project. Currently it is not used and not recommended to be modified.

2.1.2. Installing Dependent Libraries

- Install the compression library:

```
sudo apt-get install python-lzo
```

- Install the bsdiff tool:

```
cd SDX6X-delta-gentools
chmod 777 -R *  # Modify the permission of SDX6X-delta-gentools directory
sudo cp bsdiff /usr/bin
```

- Copy the file *libbz2.so.0.**, on which bsdiff depends, to the library directory:

```
sudo cp -P libbz2.so.0* /usr/lib
```

2.1.3. Getting to Know the Firmware Package

Taking RG520N series module as an example, a firmware package without the suffix "factory" contains a data file used to make the upgrade package. The directory of the firmware package is as follows:

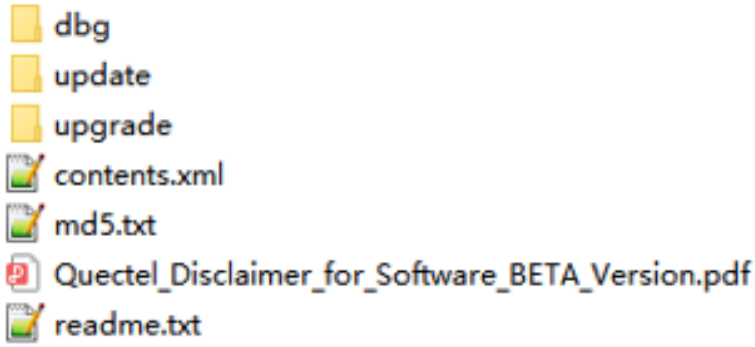

Figure 2: Directory of the Firmware Package

Table 2: Files and Directories

Files/Directories	Description
dbg	Used to debug images such as vmlinux or modem related firmware images. Ignore it when debugging is not needed.
update	Contains *firehose* and images of each partition; used for downloading via QFlash or QFIL.
upgrade	Contains *targetfiles.zip* used to generate the upgrade package.
contents.xml	Firmware package resource description list.
md5.txt	Stores MD5 checksum of each image file for file integrity verification.

2.2. Preparing targetfiles.zip

2.2.1. Preparing Source targetfiles.zip

Before making an upgrade package, it is necessary to ensure that the images *sdxlemur-boot.img*, *sdxlemur-sysfs.ubi* and *NON-HLOS.ubi* in the currently running firmware version are consistent with their counterparts in the source *targetfiles.zip*.

Every firmware version released by Quectel contains a *targetfiles.zip* (the path is *upgrade**targetfiles.zip*). If you perform DFOTA upgrade on this version, use the *targetfiles.zip* in the *update* directory in the firmware package of this version as the source version. Before using this *targetfiles.zip*, it is necessary to check whether its generation time is consistent with that of the running firmware image:

1. Check the generation time of the *targetfiles.zip* in the *upgrade* directory.

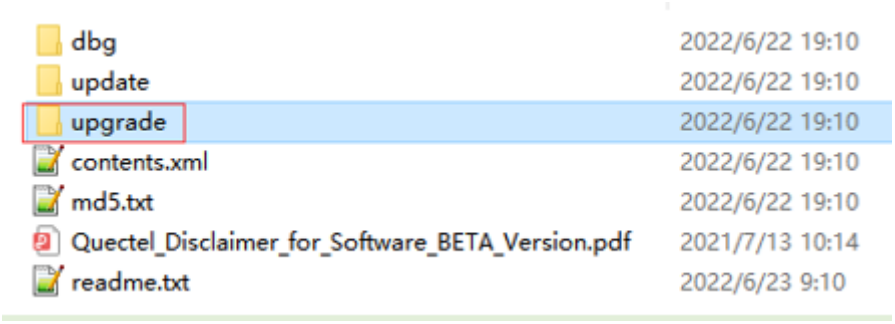

Figure 3: Directory Storing the Source targetfiles.zip

2. Check the version time of the module's Linux AP subsystem. Execute **cat /etc/quectel-project-version** to get the following information, in which the Package Time is the generation time of the firmware package:

```
Project Name: RG520NNADA_VB
Project Rev : RG520NNADAR01A01M4G_OCPU_BETA_20220622B_01.001
Branch  Name: SDX6X
Custom  Name: STD
Package Time: 2022-06-22,15:50
```

NOTE

If the kernel, rootfs and modem images in the *targetfiles.zip* used in *v1* directory are inconsistent with those in the current firmware of the module, the DFOTA upgrade will fail.

If the currently running firmware version is not the one released by Quectel, for example, the code of the kernel, abl, rootfs or modem partition has been modified, and the *targetfiles.zip* of the modified version is absent, you can get the source *targetfiles.zip* via either of the following two approaches (the first approach is simpler):

1) Update the modified firmware version to the version released by Quectel with QFlash, and then use the *targetfiles.zip* in the updated version as the source version for this upgrade. For the usage of QFlash, see ***document [3]***.

2) Generate a *targetfiles.zip* corresponding to the modified firmware version as the source version referring to ***Chapter 2.2.2***. (Then the target *targetfiles.zip* is generated based on the modifications made to the kernel, abl, rootfs and modem this time.)

2.2.2. Preparing Target targetfiles.zip

You can get the target *targetfiles.zip* via either of the following two approaches:

1) If a new firmware package released by Quectel is the target firmware version, then the *targetfiles.zip* contained in the new firmware package can be used as the target *targetfiles.zip*;
2) Generate the target *targetfiles.zip* through modifying the *v1* source *targetfiles.zip*. (This method is elaborated in the rest part of this section.)

Taking RG520N series module as an example, after modifying the kernel, abl or adding process(es) or library file(s) to *rootfs*, recompile the QuecOpen SDK, and the Linux AP subsystem will generate in the SDK output directory the *rootfs*, *sdxlemur-boot.img* and *abl.elf* needed for preparing the target *targetfiles.zip*:

```
~/sdx6x/rg520nnada/ql-ol-extsdk/target$
rootfs    →    rootfs system (the generated image is sdxlemur-sysfs.ubi, corresponding to the SYSTEM directory in the targetfiles.zip)
sdxlemur-boot.img  →  Linux kernel boot image（stored in BOOTABLE_IMAGES directory）
abl.elf  →  Application bootloader (stored in MULTIFOTA directory)
```

Use the updated kernel image *sdxlemur-boot.img*, *abl.elf* and *rootfs* file to replace the corresponding files in the unzipped source *targetfiles* folder to generate the target *targetfiles.zip*.

The following are the steps of generating target *targetfiles.zip*. Please back up and archive each target *targetfiles.zip* for subsequent DFOTA upgrades.

2.2.2.1. Decompressing the Source targetfiles.zip

Copy *targetfiles.zip* of the source version to the subdirectory of *SDX6X-delta-gentools/v2* in the Ubuntu 14.04 system, and decompress it to the *targetfiles* directory by executing the following commands:

```
cd ~/TOOLS/DeltaFota_tools/SDX6X-delta-gentools/v2/
mkdir -p targetfiles
unzip -d ./targetfiles ./targetfiles.zip  # Unzip the source version package
```

The structure of the decompressed *targetfiles* directory is as follows:

```
v2/targetfiles
 ├──── BOOT
 ├──── BOOTABLE_IMAGES
 ├──── DATA
 ├──── FULL-IMAGES
 ├──── IMAGES
 ├──── META
```

```
├─── MULTIFOTA
├─── OTA
├─── RADIO
├─── RECOVERY
└─── SYSTEM
```

2.2.2.2. Replacing boot Images

Use *sdxlemur-boot.img* in the compilation output directory to replace the *boot.img* in the source t*argetfiles/BOOTABLE_IMAGES* directory. The *boot.img* corresponds to the kernel boot image in normal mode. The commands are as follows.

```
cd targetfiles/..
cp ~/sdx6x/rg520nnada/ql-ol-extsdk/target/sdxlemur-boot.img .
cp -rf sdxlemur-boot.img  ./targetfiles/BOOTABLE_IMAGES/boot.img
```

2.2.2.3.Replacing abl Image

Use *abl.elf* in the compilation output directory to replace the *abl.elf* in the source *targetfiles/MULTIFOTA* directory. The commands are as follows.

```
cd targetfiles/..
cp ~/sdx6x/rg520nnada/ql-ol-extsdk/target/abl.elf .
cp -rf abl.elf ./targetfiles/MULTIFOTA/abl.elf
```

2.2.2.4. Replacing rootfs

Delete all files in the *targetfiles/SYSTEM* directory except the *firmware* and *dev* subdirectories. Then copy the *rootfs* generated after recompilation (excluding *firmware* and *dev* subdirectories) to the *targetfiles/SYSTEM* directory. The *rootfs* directory is usually located in the *ql-ol-extsdk/ql-ol-rootfs* directory in the QuecOpen SDK installed on the host Linux PC. For example, the *rootfs* directory structure of RG520N series module is as follows:

```
~/sdx6x/rg520nnada/ql-ol-extsdk/ql-ol-rootfs
├─── bin
├─── boot
├─── build.prop
├─── cache
├─── data
├─── dev
├─── etc
```

```
├─── firmware
├─── home
├─── lib
├─── media
├─── mnt
├─── oemapp
├─── oemdata
├─── overlay
├─── persist
├─── proc
├─── run
├─── sbin
├─── sdcard -> /mnt/sdcard
├─── sys
├─── system
├─── systemrw
├─── target
├─── tmp
├─── usr
├─── usrdata
├─── var
└─── WEBSERVER
```

The commands to replace *rootfs* are as follows:

```
#Switch to the SYSTEM directory, namely to rootfs
cd ~/TOOLS/DeltaFota_tools/SDX6X-delta-gentools/v2/targetfiles/SYSTEM
#Back up the source modem firmware (SYSTEM/firmware directory)
mv firmware dev ../
rm -rf *  #Delete the source rootfs (namely the source SYSTEM/firmware directory)
#Copy the rootfs generated after recompilation to the targetfiles/SYSTEM directory
cp -prafR  ~/sdx6x/rg520nnada/ql-ol-extsdk/ql-ol-rootfs/*  ./
#Delete the firmware image
rm -rf firmware dev
#Restore the modem firmware backed up
mv ../firmware ../dev ./
```

NOTE

1. Since the empty directories may be lost during DFOTA upgrade, please avoid adding empty directories to the *rootfs* of the target version.
2. The directory structure of *~/TOOLS/DeltaFota_tools/SDX6X-delta-gentools/v2/targetfiles* is consistent with that of *targetfiles.zip*.
3. If the modem image is updated, please use it to replace the same-name image in the

SYSTEM/firmware directory.

2.2.2.5. Repackaging the Modified targetfiles.zip

Execute the command below in the *targetfiles* directory to generate a new *targetfiles.zip* in the same-level directory of *targetfiles*:

```
zip -qry0 ../targetfiles.zip ./*
```

NOTE

The parameter "**y**" in the zip command cannot be omitted. If there is an error message, resolve the error before repacking.

2.3. Copying targetfiles.zip to the Right Directory

Enter the *SDX6X-delta-gentools* directory, and store the target *targetfiles.zip* into the *SDX6X-delta-gentools/v2* directory. The directory structure is as follows:

```
SDX6X-delta-gentools
  …
├──── v1
   └────targetfiles.zip   # Source version
└──── v2
   └────targetfiles.zip    # Target version, which contains updated boot.img, abl.elf, NON-HLOS.ubi
                             and rootfs
```

NOTE

The target *targetfiles.zip* currently stored in the *SDX6X-delta-gentools/v2* directory can be used as the source *targetfiles.zip* in the subsequent version iteration.

2.4. Generating Upgrade Package

Execute **update_gen.sh** and the upgrade package will be generated. It is recommended to redirect the standard output to the Log, so that the console only prints error information upon the occurrence of any error in the script; and the rest log messages are directed to the log file lest redundant information should affect the debugging.

Before executing this command, execute **chmod -R 777 .** under *SDX6X-delta-gentools* directory to ensure the permission is right.

```
./update_gen.sh a > log
```

The common parameters of **update_gen.sh** and their meanings are as follows:

a Make upgrade package for modem, Linux system and Linux kernel and upgrade modem, Linux system and Linux kernel
l Make upgrade package for Linux file system and upgrade Linux file system only
m Make upgrade package for modem and upgrade modem only
o Make upgrade package for Linux kernel and upgrade Linux kernel only

The *update.zip* generated in the *SDX6X-delta-gentools* directory is the upgrade package.

NOTE

1. Parameters ***M***, ***l*** and ***o*** can be used independently or in combination. It is recommended to use **./update_gen.sh a > log** to perform DFOTA upgrade on the entire system to avoid software exceptions caused by incompatible image versions.
2. The time it takes to finish a DFOTA upgrade and the file size of *update.zip* depend on the differences between the source and target versions. For a cross-baseline firmware upgrade, the generated *update.zip* is larger, and the time the upgrade takes is longer. Generally, a *update.zip* file does not exceed 70 MB.

2.5. Modifying File Permission

If the file permissions before and after a DFOTA upgrade are inconsistent, for example, the file permission before the upgrade is 755 while after the upgrade is 644, the user App would fail. A reference solution to this problem is as follows.

The *quectel_chmod_filelist.txt* file under the *SDX6X-delta-gentools* directory is used to modify the reading, writing and execution permissions of files and directories, as well as the user and user group to which a file or directory belongs. Specifically, write into *quectel_chmod_filelist.txt* the file/folder along with

the required permission, user and user group of it according to the format shown in the figure below. In the figure, command lines starting with "#" indicates that they have no effect; command lines starting with "/" are considered valid configurations, and command lines starting with "/" and ending with "/" are parsed as files or folders.

```
this line is comment
#default
#example
#    file or folder       file_permission     userid   groupid     folder_permission
#    default              755                 0        0           755
#    /home/test_folder/   755                 1        0           644
#    /usr/bin/test_bin    644                 2        2
#    /usr/bin/test_bin1   742                 3        1
#    /usr/bin/test_bin2
```

Figure 4: Configuring Permissions

The specific meanings of the parameters are shown in the fourth line in the figure above. When all parameters except file names or paths are omitted, the default configurations are used. The default configurations and their meanings are as follows:

- *file_permission*: File permission. Default: 755.
- *userid*: User ID. Default: 0.
- *groupid*: Group ID. Default: 0.
- *folder_permission*: Directory permission. Default: 755.

2.6. Preserving User Configurations

Because rootfs is read-only, the files in the */etc* and */data* directories configured by users are read-only, they cannot be rewritten. Through the **mount bind** command of the overlayfs mechanism, */etc* is pointed to */usrdata/etc* in the VFS, and */data* is pointed to the */usrdata/data* directory, so the program or user configuration data is saved in */usrdata*. The */usrdata* partition is not upgraded via DFOTA, but the factory configurations are synchronized to the */usrdata/etc* and */usrdata/data* directories after the module is rebooted and the file system is mounted. After the DFOTA upgrade, the data in the */usrdata/etc* and */usrdata/data* directories is covered by the corresponding directories of */etc* and */data* during the first rebooting and mounting, so the user configuration cannot be retained.

SDX6X-delta-gentools provides a mechanism to preserve specified user configurations when making upgrade packages. The user can specify a list of user profiles to be preserved through *quectel_noota_filelist.txt*. Take the *test.cfg* file to be preserved in the *webui* directory under */etc* as an example. To keep the *test.cfg* file, the user only needs to add *SYSTEM/etc/webui/test.cfg* to *quectel_noota_filelist.txt*.

```
SYSTEM/kt_dms/ec21/kt_svc
SYSTEM/kt_dms/ec25/kt_dmc
SYSTEM/kt_dms/ec25/kt_svc
SYSTEM/kt_dms/lib/libkt_dev_detail-1.0.so
SYSTEM/kt_dms/lib/libkt_fw_flags-1.0.so
SYSTEM/kt_dms/lib/libkt_dmc_cm-1.0.so
SYSTEM/kt_dms/lib/libkt_fw_up-1.0.so
SYSTEM/firmware/image/seg_info.txt
SYSTEM/etc/webui/test.cfg
```

NOTE

1. Currently the mechanism only supports adding a single file rather than a specified directory.
2. Add the file to *quectel_noota_filelist.txt* according to the specified format, and do NOT remove the existing files randomly.

3 Restrictions on DFOTA

1. [1)] It is not recommended to add or modify the rootfs files in the module through ADB or the console. It is recommended to update rootfs files by creating a rootfs image. If you manually import the files into rootfs of the module, please update the corresponding file in *targetfiles.zip* synchronously, and make a new targetfiles.zip.

   Cause: Adding or modifying the rootfs files in the module will cause the image running on the module to be inconsistent with the source *targetfiles.zip*, and the data integrity and consistency check of the delta firmware file will fail, resulting in DFOTA upgrade failure. Please modify the corresponding file of *targetfiles.zip* synchronously to maintain data consistency.

2. If the files that have been added to *quectel_noota_filelist.txt* are modified or updated, they need to be removed from the list of *quectel_noota_filelist.txt* and added to the list of *quectel_fullota_filelist.txt*.

   Cause: *quectel_noota_filelist.txt* refers to the file list that is kept during upgraded, and *quectel_fullota_filelist.txt* is the file list that is forced to be covered with the files of new version.

3. Files that exist in the image running on the module but do not exist in the target package will be deleted during DFOTA upgrade.

4. [2)] The newly added profile in */usrdata/etc* should not be too large, otherwise it will use up the remaining space of the system rootfs (preferably maintain 20 MB of free space), and the rootfs cannot be mounted. It is recommended that the system rootfs keep at least 20 MB of free space. If you need to add large files, please expand the size of rootfs partition in advance and pre-adjust the running version before DFOTA upgrade has been pre-adjusted for the partition. For partition adjustment, refer to ***document [4]***.

5. To upgrade files or replace files into the module forcibly, specify them by configuring *quectel_fullota_filelist.txt*.

6. After the DFOTA upgrade command is executed, the problem of error reporting is divided into two stages:
   1) Check whether the upgrade package is correct before DFOTA upgrade. If an error is reported, the upgrade will not continue.
      a) Use **unzip -tq** to test the upgrade package to ensure that it is a normal compressed package. If there is no "No errors" keyword in the return value, an error will be reported.
      b) Read *quectel-project-version* in the upgrade package. If it is different from the module */etc/quectel-project-version*, an error will be reported.

The error code reported above is 505, and the system will not be switched to the recovery mode at this time.

If the upgrade package is too large and exceeds the available cache space of usrdata, and the sum of the maximum file size after the package is decompressed and the size of *targetfiles.zip* is no less than the free space of usrdata, error code 550 will be reported.

2) After the module enters the recovery mode with the correct upgrade package, if the upgrade reports an error, a *last_log* (a dump logfile) file will be generated and is written to the*/usrdata/cache/recovery/last_log* path. If no error occurs, the upgrade information is reported to the */cache/recovery/log* path by default. If the module does not switch to recovery mode, *last_log* will not be generated when you perform a DFOTA upgrade.

NOTE

1. [1)] Data consistency check is required for DFOTA upgrade file. If a file is mistakenly added to the rootfs of the module and *targetfiles.zip* of the running firmware is not updated synchronously, the consistency check will fail, and the upgrade fails.
2. [2)] It is the limitation of ubifs (the space used for DFOTA upgrade exceeds the pre-saved space).
3. Remedial method for upgrade failure: use QFlash to force the upgrade through USB interface, or use PCIe network port to upgrade.

4 DFOTA Upgrade

The workflow of DFOTA upgrade is presented here:

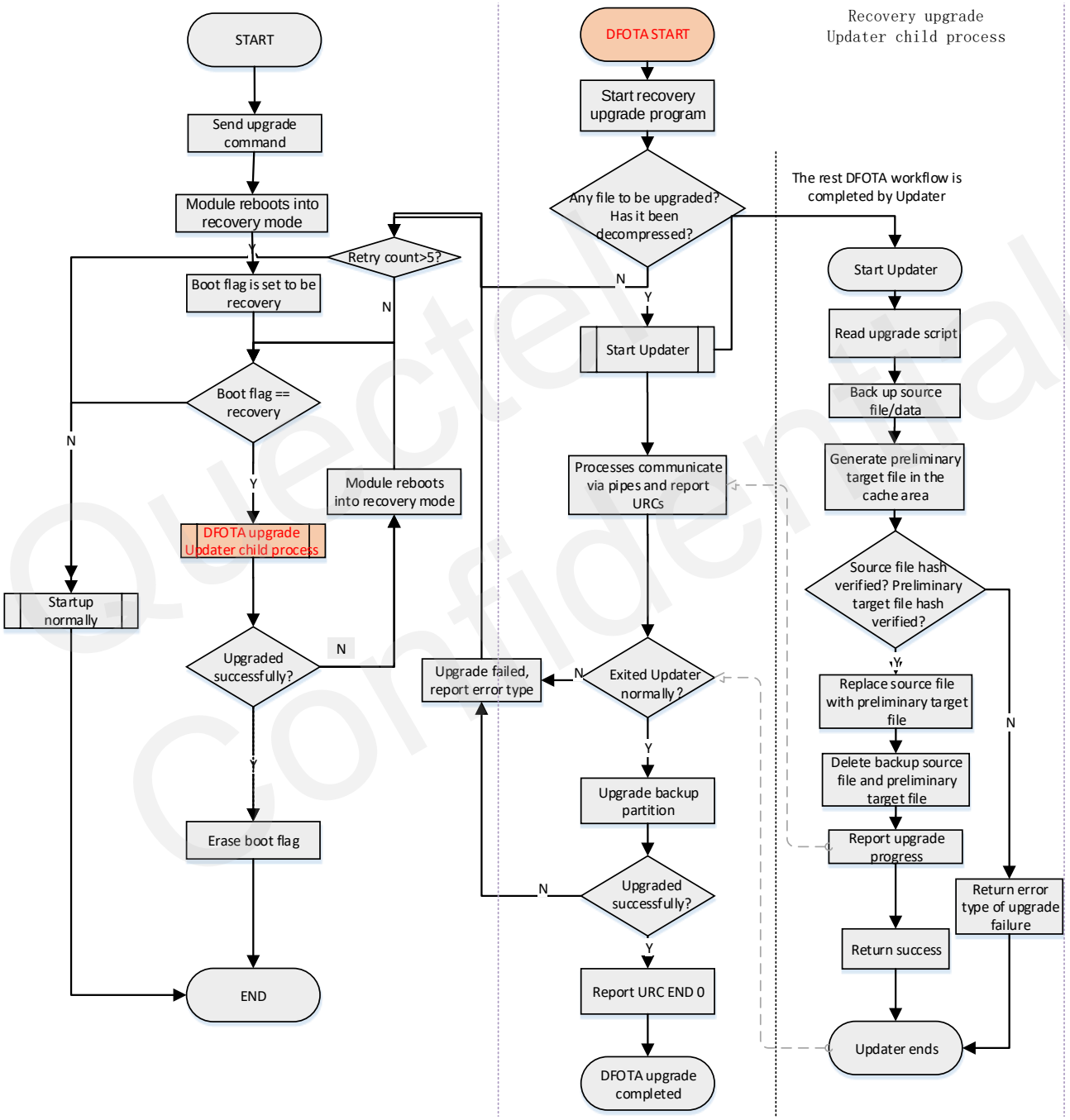

Figure 5: DFOTA Upgrade Flowchart

4.1. Sending update.zip to Module

Make ensure that the upgrade package *update.zip* is generated with the right method and of the correct version before sending it to the module for DFOTA upgrade. The *update.zip* is cached into the NAND flash (user data cache area) of the module before the upgrade. This chapter introduces two ways to send *update.zip* to the module: via ADB and via HTTPS.

NOTE

As the system saves *update.zip* into */usrdata/cache/fota/ipth_package.bin* after downloading it, please reserve enough space in the *usrdata* partition, i.e., keep the available space in *usrdata* larger than the size of *update.zip*. Otherwise, *update.zip* cannot be downloaded successfully. Given that *usrdata* is also used a cache area during the upgrade, the available space in *usrdata* should be larger than 60 MB for upgrades within a baseline and larger than 110 MB for cross-baseline upgrades.

4.1.1. Sending via ADB

First, configure the serial port for ADB transmission. Connect the module to the PC through USB, open the device manager to confirm the COM port number of AT Port (such as COM13 in the figure below), and enable the COM port corresponding to the AT Port with QCOM:

Figure 6: Serial Ports

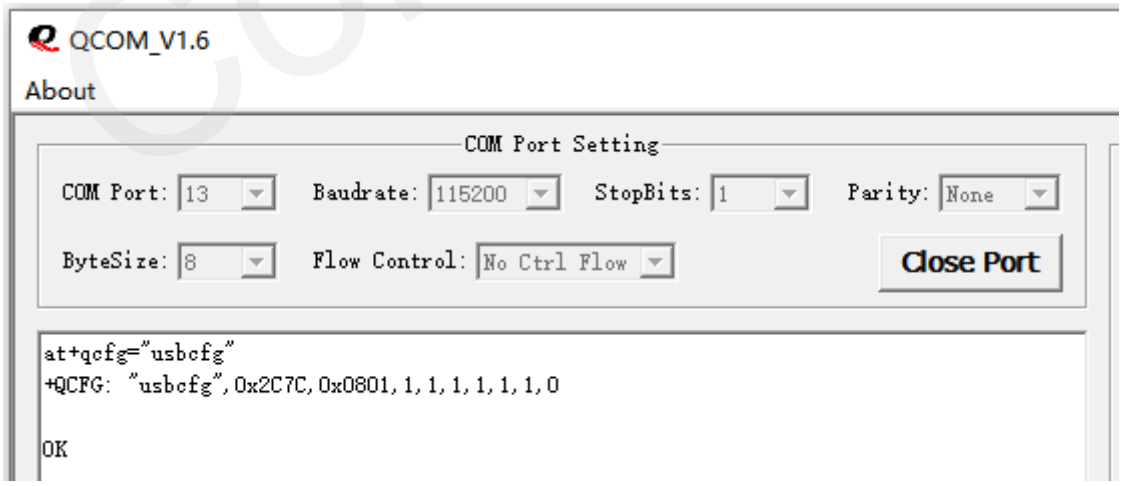

Figure 7: Enabling AT Port on QCOM

Then, follow these steps to enable the ADB function on the module:

Step 1: Execute **AT+QCFG="usbcfg"** to query the VID and PID information of the USB;

Step 2: Execute **AT+QCFG="usbcfg",<VID>,<PID>,1,1,1,1,1,1,0** (among which the sixth parameter is set to 1 to enable ADB function). **<VID>** and **<PID>** are the VID and PID get in ***Step 1***. For example, **AT+QCFG="usbcfg",0x2c7c,0x0801,1,1,1,1,1,1,0**.

Step 3: Execute **AT+CFUN=1,1** to restart the module, so that the ADB function can take effect.

Step 4: Open the device manager and confirm that the ADB device is connected. The interface is as follows:

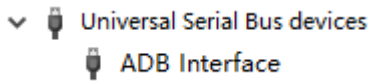

Figure 8: ADB Device Connected

Step 5: Open cmd to enable ADB service in the PC.

```
adb devices
```

If the ADB service does not run normally, execute the following commands to restart it.

```
adb kill-server
adb start-server
adb devices
```

Step 6: Execute the following command to set the root file system to be readable and writable.

```
adb shell mount -o remount, rw /
```

Step 7: Send *update.zip* to the module with the adb commands.

```
#Send update.zip with adb (from PC source directory to the module directory)
adb push D:/test/update.zip  /usrdata/cache/ufs
#Save update.zip
adb shell sync
```

NOTE

The above stated steps are applicable only to QuecOpen solution. When the module is used in its standard version, the following operations require to be performed between ***Step 1*** and ***Step 2***: Execute **AT+QADBKEY=?** to query the ADB key. If there is no key returned, the command returns **ERROR**; otherwise, it returns an 8-digit module ID. Send the module ID to Quectel Technical Support and you can

obtain the corresponding ADB key. Then, execute **AT+QADBKEY=<key>**, where **<key>** is the generated ADB key.

4.1.2. Sending via HTTPS

To send *update.zip* via HTTPS, the module should connect with the PC in the form of Client/Server connection, where data is transmitted via sockets. The steps are as follows:

Step 1: The module serves as the network server to provide network access through WebUI;
Step 2: The PC works as the client which connects to the network server through PCIe network port;
Step 3: The client's IP address is allocated by the module, and the client accesses the network provided by the module on https://192.168.225.1;
Step 4: The module realizes uploading and downloading CGI scripts, allowing the client to upload *update.zip* to it.

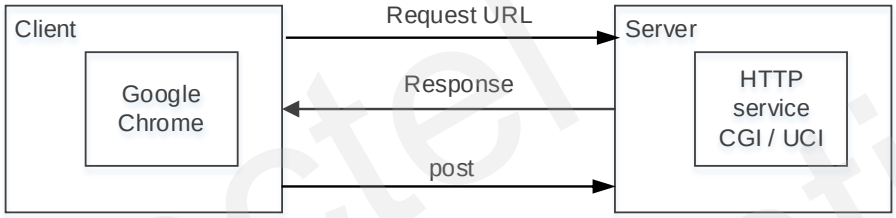

Figure 9: Client/Server Connection Structure

NOTE

The applications on WebUI are designed by yourselves according to your own specific needs. Quectel does not provide such customized designing services.

Taking RG520N series modules as an example, the process of transmitting upgrade package through HTTPS is as follows:

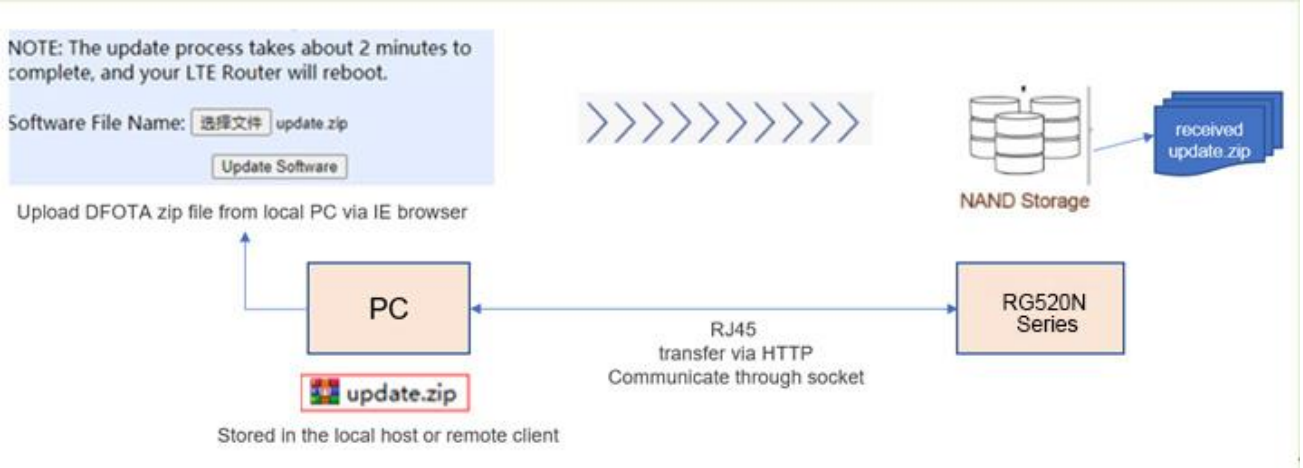

Figure 10: Example of Sending Upgrade Package via HTTPS

4.2. Triggering DFOTA Upgrade with AT Command

4.2.1. Calling Upgrade AT Command Manually

Execute **AT+QFOTADL=<URL>** and the automatic upgrade is triggered, where **<URL>** is the address of the delta firmware package; it can be an FTP/HTTP/HTTPS address or an internal path of the module (the */usrdata/* directory). Please use either port smd7 or smd11 to perform the DFOTA upgrade. The concrete steps of calling AT command in the module for DFOTA upgrade are as follows:

Step 1: Execute the following commands in the console to confirm whether the serial port chosen for the upgrade is occupied. If there is no response, the port is available; otherwise, it is occupied. If you decide to use an occupied port, execute **kill ${PID}** to forcibly exit the service that is occupying it.

```
user /dev/smd7   #Obtain the PID of the process that is occupying smd7
kill ${PID}      # forcibly exit the service to release smd7
```

Step 2: Execute the following commands to start DFOTA upgrade.

```
cat /dev/smd7 &   #Enable command echoing
echo -e " at\r\n " > /dev/smd7
echo -e "AT+QFOTADL=\"URL\"\r\n" > /dev/smd7
```

NOTE

1. The "URL" in the above command line needs to be replaced with the actual address of the upgrade package. For example, if the actual storage address is */usrdata/cache/ufs/update.zip*, you need to

execute the following command:
echo -e "AT+QFOTADL=\"/usrdata/cache/ufs/update.zip"\r\n" > /dev/smd7
2. If performing upgrade through USB AT Port with QCOM, you need to execute the following:
AT+QFOTADL="/usrdata/cache/ufs/update.zip"

4.2.2. Calling Upgrade AT Command by Program

To call upgrade AT command by a program, see ***document [2]*** for details.

4.3. Viewing Upgrade Progress and Result

During a DFOTA upgrade, there are two kinds of response:

1. DFOTA upgrade progress and results reported in the form of URC:

```
+QIND: "FOTA","START"
+QIND: "FOTA","UPDATING",<percent>
+QIND: "FOTA","UPDATING",<percent>
...
+QIND: "FOTA","END",<err>
```

You can check the response details via the QCOM tool or the console.

2. DFOTA upgrade progress and results written into the following file by the recovery process:

```
cat /usrdata/cache/fota/ipth_config_dfs.txt
```

The returned message types are as follows:

- *IP_PREVIOUS_UPDATE_SUCCESSFUL*: Upgrade successfully
- *IP_PREVIOUS_UPDATE_CHECK_FILE_FAILED*: File error occurs
- *IP_PREVIOUS_UPDATE_IN_PROGRESS*: Fail to write in

If the upgrade is implemented through web, the upgrade result can be displayed through the WebUI interface.

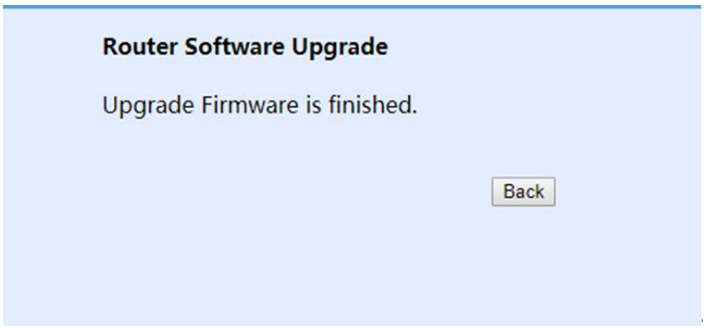

Figure 11: Result Displayed on WebUI (Example)

5 Troubleshooting

In general, the time it takes to complete a DFOTA upgrade is proportional to the size of the upgrade package. If the upgrade package fails to be generated, please check if there is any exception in the relevant logs during the upgrade.

During the upgrade process, not only normal URCs such as the upgrading progress is reported, but also the URC of a corresponding error code is reported if an exception occurs and the upgrade fails, as shown below:

```
+QIND: "FOTA","START"
+QIND: "FOTA","UPDATING",<percent>
+QIND: "FOTA","UPDATING",<percent>
...
+QIND: "FOTA","END",<err>
```

Here are the meanings of some common seen **<err>** values.

Table 3: Summary of <err>

<err>	Description
0	DFOTA upgrade succeeded
501	When the upgrade starts, the delta firmware package cannot be decompressed normally, or the space required for decompression is insufficient
502	The upgrade process is exited abnormally
503	MD5 checksum error of the upgrade package
504	Incorrect format of the upgrade package, which will thus be deleted.
510	File verification failed during upgrade
511	Insufficient space during upgrade
OK	Command executed successfully (only applicable to upgrade triggered by AT command)
ERROR	Command format error (only applicable to upgrade triggered by AT command)

6 FAQ

6.1. How to identify the source targetfiles.zip?

After finishing a DFOTA upgrade using the upgrade package generated based on the source *targetfiles.zip* in the *v1* directory and the target *targetfiles.zip* in the *v2* directory, which version of *targetfiles.zip* should be reserved as the source version for the next DFOTA upgrade?

The current target *targetfiles.zip* in the *v2* directory should be the source *targetfiles.zip* for the next DFOTA upgrade, and the next target *targetfiles.zip* is generated based on the modifications of the current target *targetfiles.zip.* If you still use the current source *targetfiles.zip* as the source *targetfiles.zip* for the next DFOTA upgrade, the upgrade will fail due to inconsistency of firmware image versions.

It is recommended to archive every target *targetfiles.zip* based on its dates or versions for the convenience of subsequent DFOTA upgrades.

6.2. How to output logs after an upgrade fails?

Real-time logs are usually stored in the */cache/recovery/log* directory. If the upgrade fails, the logs are directed to the */usrdata/cache/recovery/last_log* directory. If you need to output the logs for further analyses by Quectel Technical Supports, get the module enter the recovery mode, redirect logs to the */data* directory, restart the module to normal mode, and then export the logs with the **adb pull** command.

6.3. How to view logs in recovery mode?

After restarting the module, connect its Debug port to the PC. During SBL startup, press **Ctrl** + **C** on the PC, and then input "recovery" to enter the recovery mode, where you can view directly the log information stored in */cache/recovery/log*.

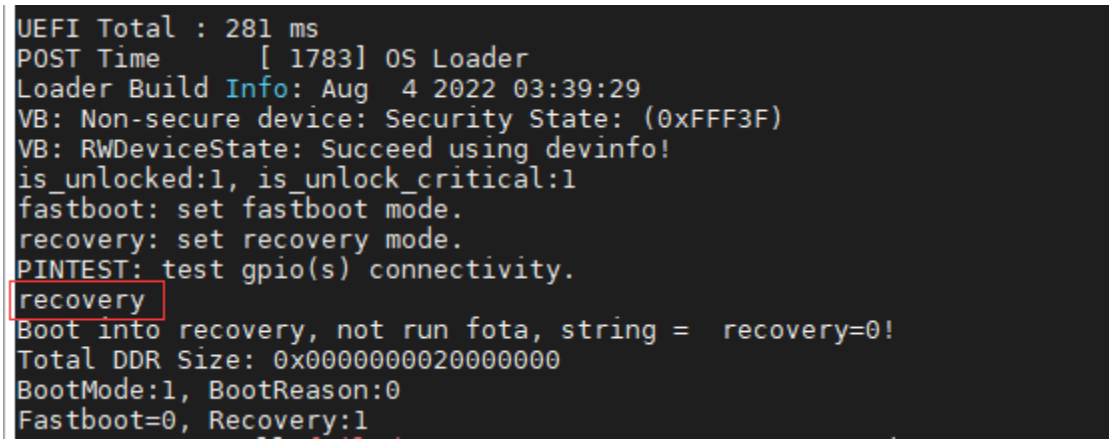

Figure 12: Enter recovery Mode

6.4. How to ensure the source targetfiles.zip is generated based on the current firmware?

The *update.zip* generated by the DFOTA tool contains the file *quectel-project-version*. Execute **cd ~/p4/TOOLS/DeltaFota_tools/update && tree -L 1** and you will see the directory where this file is in:

```
.
├── max_file
├── META-INF
├── MULTIFOTA
├── patch
├── quectel-project-version
├── recovery
└── system
```

The *quectel-project-version* contains the version information of the Linux AP system build. The upgrading program will read this version information and compare it with the version information contained in the */etc/quectel-project-version* file to check whether the source *targetfiles.zip* is generated based on the current version firmware. If not, the upgrade will stop and the error code 505 be reported.

6.5. How does the module handle exceptional power off during DFOTA?

It is necessary for the module to be powered normally throughout the DFOTA process. Yet sometimes the power supply can be cut off unexpectedly. To compensate for that, the recovery mode has the following mechanisms.

1. Erase the upgrade flag only when all the data to be updated has been updated. So long as the upgrade flag exists (in misc partition), the module enters recovery mode every time upon reboot.

2. Implement incremental upgrade in order. The updater process scans the upgrade script for the information of files and data to be updated, and upgrade the files or partitions one by one according to their order in the script (for partitions without file systems, such as the kernel image *boot.img,* the upgrade data will be updated to the corresponding partitions).

   /META-INF/com/google/android/updater-script:

```
assert(apply_patch_check("/system/firmware/image/modem_pr/so/895_0_0.mbn",
"a7af5e329ed9768ceff263853996b45865b95db4",
"b28e70d093c2a55a758ce8ce1b2acc4e29a05e7f"));
set_progress(0.827153);
…
apply_patch("/system/firmware/image/modem_pr/so/895_0_0.mbn", "-",
        a7af5e329ed9768ceff263853996b45865b95db4, 1624072,
        b28e70d093c2a55a758ce8ce1b2acc4e29a05e7f,
package_extract_file("patch/system/firmware/image/modem_pr/so/895_0_0.mbn.p"));
set_progress(0.829951);
```

   In the script, the sentences containing **apply_patch_check** are used to verify the data integrity of to-be-updated files and partitions before every upgrade while sentences containing **apply_patch** are used to trigger the upgrade. Data integrity is verified by calculating the hash values of files and partition data, as shown below:

```
assert(apply_patch_check("/system/firmware/image/modem_pr/so/895_0_0.mbn",
"a7af5e329ed9768ceff263853996b45865b95db4",
"b28e70d093c2a55a758ce8ce1b2acc4e29a05e7f"));

a7af5e329ed9768ceff263853996b45865b95db4 → hash value of the file or partition after the upgrade (hash value of the target file)
b28e70d093c2a55a758ce8ce1b2acc4e29a05e7f → hash value of the file or partition before the upgrade
```

3. Back up data before every upgrade. Before upgrading a specified file or partition, the recovery upgrade process read and back up the source data or partition into */cache/saved.file*, which is used to restore data once the upgrade or the data integrity verification fails.

4. Check data integrity.

   1) Perform pre-upgrade checks:
      a. Check whether the hash value of the source file or partition data is consistent with the hash value of the source file or partition data as returned after the **apply_patch_check** sentence in the upgrading script. If it is, the source file or partition has yet to be modified or upgraded;

if not, the source file or partition has been modified and Step b will be performed.

b. Check whether the hash value of the source file or partition data is consistent with the hash value of the target file or partition data as returned after the **apply_patch_check** sentence in the upgrading script. If it is, the source file or partition has been upgraded and the upgrading program will move on to upgrade the next file or partition to be upgraded; if not, the source file or partition has been modified but not fully upgraded and Step c will be performed.

c. Check whether the hash value of the backup source file or partition data is consistent with the hash value of the source file or partition data as returned after the **apply_patch_check** sentence in the upgrading script. If it is, the upgrading program will use the backup source file or partition data as the source file or partition data to perform the upgrade from the beginning; if not, error will be reported and the upgrade stopped.

2) Replace the source file or partition data: merging the upgrade data in *update.zip* and data in the source file or partition (or the backup source file or partition data) to generate a preliminary target file, perform a check on data integrity and data attributes, and replace the source file or partition data with the preliminary target file.

3) Delete the source and backup file or partition data only after the data integrity check and the upgrade.

5. Repeat Step 3 and Step 4 until all the designated files or partitions are upgraded.

6. Retry after the upgrade fails. To address power failures disrupting the upgrade, the upgrading program perform data integrity checks on files or partitions listed in the upgrade script in sequence after the module reboots, and decide whether to continue the upgrade or restore the backup data.

The upgrading program retry at most 5 times after upgrade failures; if the fifth retry fails, the system will remove the upgrade package and erase the upgrade flag, and try to enter normal mode. If the file system is damaged and cannot be started normally, you have to download the firmware into the module with QFlash.

7 Appendix References

Table 4: Related Documents

Document Name	
[1]	Quectel_RG520N&RG52xF&RG530F_Series_QuecOpen_Quick_Start_Guide
[2]	Quectel_RG520N&RG52xF&RG530F_Series_QuecOpen_Virtual_Port_Programming_for_AT_Command_Sending_and_Receiving_Guide
[3]	Quectel_QFlash_User_Guide
[4]	Quectel_RG520N&RG52xF&RG530F_Series_QuecOpen_Partition_Adjustment_Guide

Table 5: Terms and Abbreviations

Abbreviation	Description
ABL	Application Bootloader
ADB	Android Debug Bridge
AOP	Always on Processor
AP	Application Processor
App	Application
CGI	Common Gateway Interface
CPU	Central Processing Unit
DFOTA	Delta Firmware Upgrade Over-The-Air
FOTA	Firmware Upgrade Over-The-Air
FTP	File Transfer Protocol
HTTP	Hyper Text Transfer Protocol
HTTPS	Hyper Text Transfer Protocol over Secure Socket Layer

HW	Hardware
ID	Identifier
IoT	Internet of Things
IP	Internet Protocol
IPA	IP Application Accelerator
MCP	Multiple Chip Package
MD5	Message Digest Algorithm 5
OEM	Original Equipment Manufacturer
PC	Personal Computer
PCIe	Peripheral Component Interconnect Express
PID	Production ID
SBL	Secondary Bootloader
SDK	Software Development Kit
UCI	Unified Configuration Interface
UEFI	Unified Extensible Firmware Interface
URC	Unsolicited Result Code
URL	Uniform Resource Locator
USB	Universal Serial Bus
VFS	Virtual File System
VID	Vendor ID
XBL	eXtensible Bootloader