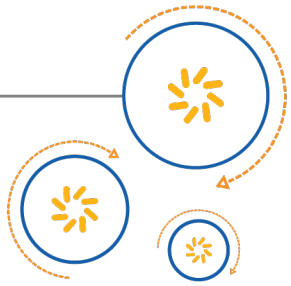
Qualcomm Technologies International, Ltd.



Audio Sink Application DFU

User Guide

80-CT448-1 Rev. AC

October 25, 2017

Revision history

Revision	Date	Description
1	MAY 2015	Initial release. Alternative document number CS-00331713-UG.
2	MAY 2016	Updated to conform to QTI standards; No technical content was changed in this document revision
AC	AUG 2017	Added to the Content Management System. Upated Document Reference Number to use Agile number. No technical content was changed in this document revision.

Contents

Tables

Figures

1 Device Firmware Upgrades - overview

Device Firmware Upgrade is a mechanism to update a device over USB. It can be used to update the following software components:

- Firmware stack
- VM application
- Other files in the VM file system (voice prompts, additional language packs)
- PS Keys

The DFU protocol is standardized by the USB Implementers Forum as a Device Class Specification. The QTIL implementation is compliant with the standard and includes security checks:

- Devices are programmed with RSA public keys when manufactured.
- DFU files are signed with RSA private keys to make sure that only DFU files for the specific product can be used by the end user.

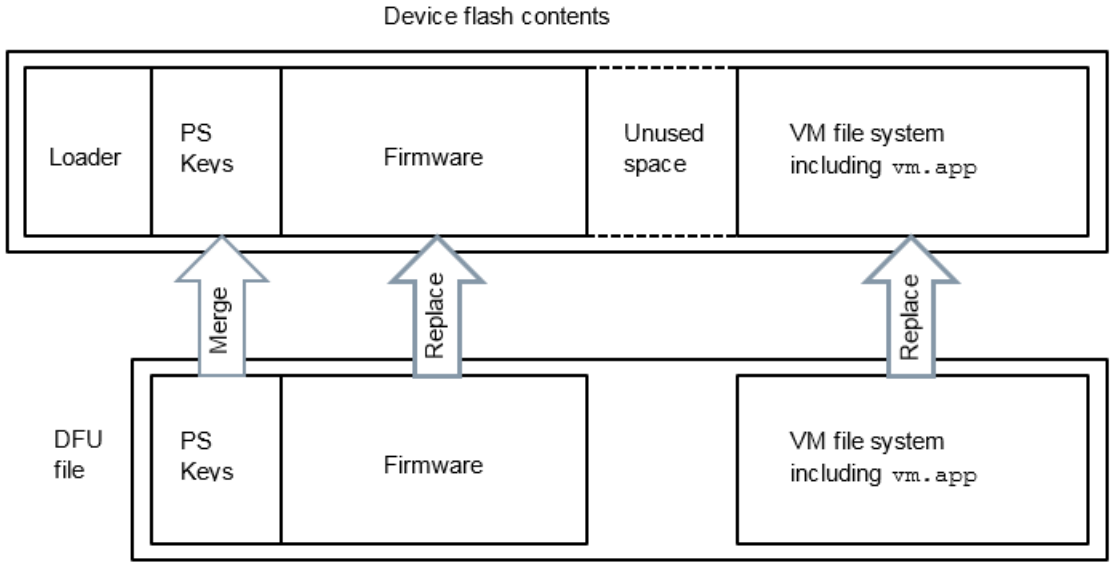


Figure 1-1 Contents of DFU file system

1.1 DFU implementation details

The options and limitations available when using the DFU mechanism are:

- Loader cannot be updated:
  - This makes the DFU process fail safe, the loader is always present and intact so the DFU process can be repeated after an unsuccessful attempt.
- Firmware can be updated (but does not have to be):
  - The firmware stack can be missing from the DFU file, in this case the firmware is not updated.
  - Always include the firmware stack if the default boot mode has no host interface (otherwise the product has just the stack, with no VM, no host interface and will not work)
- Any PS Keys present in the DFU file are modified on the device.
- The VM file system can be updated (but does not have to be):
  - The whole VM file system is replaced by the new one if VM file system is present in the DFU file.
  - It is not possible to update individual VM files, files must be contiguous with no gaps between them.

1.2 DFU upgrade tools

There are two different QTIL tools that can be used to carry out DFU using the USB interface, see Table 1-1.

Table 1-1 DFU upgrade tools

Tool	Notes
DFUWizard	This is referred to as 'Traditional DFU' and requires the end user to install the QTIL DFU driver before the DFU process can be initiated from DFU Wizard.
HidDfu	This uses the USB HID interface to carry out DFU and is referred to as 'Driverless DFU' because the end user does not have to install any drivers as part of the upgrade process.

NOTE Both of these tools are included in the Qualcomm® BlueSuite™ technology download available on createpoint.

1.2.1 DFUWizard application

The **DFUWizard** application has the following features:

- Example code showing how the **DFUEngine** DLL can be used to create a custom GUI.
- Generic tool used for DFU of QTIL devices.

The **DFUEngine** DLL provides higher level functions on top of the QTIL example driver to simplify the creation of customer DFU applications.

The example Kernel mode DFU USB driver can be replaced by a custom driver if required (source code is available). The `.inf` file can be customized to make it look like a driver for the customer specific device.

1.2.2 HidDfu application

The **HidDfuCmd** application is a command line tool used to upgrade devices using USB HID interface.

The **HidDfu** DLL provides higher level functions to simplify the creation of customer applications. Example code for a simple command line application is provided in the `HidDfu.chm` help file.

1.3 PS Keys

There are several types of PS Keys:

- Unprotected Firmware (FW) and Virtual Machine (VM) keys:
  - These keys do not have to be signed but signing them with either FW or VM key does not cause any issues.
- Protected FW keys:
  - These keys are verified by the FW public key stored in the loader.
- Protected VM keys:
  - These keys are verified by the VM public key stored in a protected FW PS Key.
- Table 1-2 describes pseudo keys, that is keys with special meaning.

Table 1-2 Pseudo PS Keys

Key	Details
`&F000`	Erases all implementation store keys Erases only unprotected keys if present in an unsigned PS file Erases only PSKEY_USR0 to PSKEY_USR49 and PSKEY_DSP0 to PSKEY_DSP49 if present in the VM key signed PS file. Erases only FW keys if present in FW key signed PS file.
`&F001`	64 word signature of the PS file
`&F002`	IC Identification number: `0000 0000 0000 0200` for BlueCore5-Multimedia `0000 0000 0000 1000` for CSR8670 **NOTE** An IC Identification number is required for the DFU to work.

1.3.1 PS Key file details

Table 1-3 PS Key file details

PS Key line contents	Details
`&ADDR = <value>`	Sets the key to the value specified
`&ADDR =`	Sets the key to zero length
`&ADDR -`	Deletes the key from the implementation store
`&ADDR + <value>`	Sets the key providing it does not already exist in the implementation store
`&ADDR ~ <value>`	Sets the key providing it already exists in the implementation store

1.4 DFU event sequence

The sequence of events during a DFU is:

1. The Device is operating normally in Stand-alone mode.
2. USB is plugged in, the device may enumerate as a USB device if configured to do so.
3. A custom USB command or key press sequence switches the Headset into DFU mode.

   NOTE This causes the Device to reboot into mode 0. Host interface must be enabled in this mode.

4. The **DFUWizard** instructs the Device to switch into Loader mode, the Device reboots and the loader is then active.

   NOTE This step is not required when carrying out an upgrade using **HidDfu**.

5. The DFU file is downloaded into the Headset and processed by the loader.
6. The Headset reboots back into mode 0 (not required when using **HidDfu**).
7. The Headset reboots back into Stand-alone mode.
8. The Device detects the USB connection and enumerates as a USB device.

2 Generating DFU files

Example code for generating DFU files is available for download, see *Practical Example of using DFU* on createpoint.

The tools used for generating DFU files are included in the BlueSuite download in the **DFU Tools** folder. These tools usually have the following command structure:

```
Dfutoolname <output file name> <input file name> <input key>
```

2.1 Generate DFU keys

Table 2-1 Generate DFU keys

Description	Command	Output files
Generate DFU keys	`dfukeygenerate -o keys`	`keys.private.key` `keys.public.key`

2.2 Sign loader and firmware

Table 2-2 Signing loader and firmware

Description	Commands	Output files
Insert key for verifying authenticity of stack into loader	`dfukeyinsert -v -o loader_signed -l loader_unsigned.xdv -ks keys.public.key`	`loader_signed.xdv` `loader_signed.xpv`
Sign the stack with a key corresponding to the one inserted in the loader	`dfusign -v -o stack_signed -s stack_unsigned.xpv -ks keys.private.key`	`stack_signed.xdv` `stack_signed.xpv`

2.3 Build and pack the VM application

Table 2-3 Building and packing VM application

Description	Commands	Output files
Compile the application	`xipbuild.exe -f -n <configuration_name> <project_name>.xip`	*<app_name.app>*
Create image directory	`createpath output`	-

Table 2-3 Building and packing VM application (cont.)

Description	Commands	Output files
Rename and copy app file into image directory	`copyfile <app_name>.app output\vm.app`	`vm.app`
Pack the image directories into file systems	`packfile output output.fs`	`output.fs`

2.4 Sign PS Keys and application

Table 2-4 Signing PS Keys and application

Description	Commands	Output files
Insert key for verifying authenticity of VM file system into PS Keys	`dfukeyinsert -v -o image_signed -ps image.psr -ka keys.public.key`	`image_signed.psr`
Sign the VM file system with a key corresponding to the one inserted in the PS Key file	`dfusign -v -o image_signed -h output.fs -ka keys.private.key`	`image_signed.fs`

2.5 Build binary image

Table 2-5 Building binary image

Description	Commands	Output files
Image generated includes: 1) Loader with key inserted 2) Signed firmware stack 3) Signed VM file system	`vmbuilder -size <flash size> merge.xpv stack_signed.xpv image_signed.fs`	`merge.xdv` `merge.xpv`

`image_signed.psr` generated in Sign PS Keys and application is not built into the binary image generated. Therefore to make a production image:

- Program the `merge.xpv` file onto the target device using **BlueFlash**
- Merge `image_signed.psr` using **PSTool**
- Dump the contents of the device into production image file using **BlueFlash**

2.6 Generating DFU file

Table 2-6 Generating DFU file

Description	Commands	Output Files
Sign PS Keys to be included in the DFU image	`dfusign -v -o dfu_vm_signed -pa example_DFU_vm.psr -ka keys.private.key`	`dfu_vm_signed.app.psr`
	`dfusign -v -o dfu_fw_signed -ps example_DFU_fw.psr -ks keys.private.key`	`dfu_fw_signed.stack.psr`
Generate the DFU file	`dfubuild -v pedantic -f image.dfu -uv <USB VID> -up <USB PID> -s stack_signed.xpv -d stack_signed.xdv -h image_signed.fs -p3 . dfu_fw_signed.stack.psr dfu_vm_signed.app.psr`	`image.dfu`

Document references

Document	Reference
Practical Example of using DFU	CS-00231230-WI

Terms and definitions

Term	Definition
BlueCore	Group term for the range of QTIL Bluetooth wireless technology ICs
Bluetooth	Set of technologies providing audio and data transfer over short-range radio connections
BlueSuite	QTIL Production Tools
DFU	Device Firmware Upgrade
DFUWizard	QTIL Tool for carrying out DFU
FW	Firmware
HID	Human Interface Device
IC	Integrated Circuit
HidDfu	QTIL Tool for Driverless DFU
PS Keys	Persistent Store Keys
QTIL	Qualcomm Technologies International, Ltd.
RSA	Rivest Shamir Adleman (public/private key encryption)
USB	Universal Serial Bus
VM	Virtual Machine