

Realtek NAS SDK

Image-Builder



Image-Builder

- Build Realtek SoC Image file
- Prepare Files for Rescue System
- Customization
 - Specify Firmware Layout in Storage
 - Edit `feed.conf` to describe storage layout
 - Support Customer's Root Filesystem
 - Edit `feed.conf` to describe content of root partition
 - Easy to Set
 - Kernel Boot Argument
 - MAC Address
 - IP information for Bootcode
 - Edit file, `bootargs.conf.{spi,emmc,nand}`



install.img

- Realtek SoC Image file which contains
 - Bluecore.audio
 - Kernel Image
 - DTB for Normal Boot up
 - DTB for Rescue Boot up
 - Root Filesystem for Rescue System
 - Root Filesystem for Normal Boot up (if eMMc or NAND)
 - Layout description file, config.txt
 - Storage Writer, installer



Directories

- build_image.sh
 - Build Script
- arm_bin/
 - arm binary files for running in rescue system
- x86_bin/
 - X86 binary files for creating install.img



Directories

- rescue-rootfs/
 - Rescue Root File System.
 - Use initramfs.sh to compress/decompress rescue root filesystem image.
 - Size limit is 1 MB (1048576 bytes)
 - If exceeding the limit, change CONFIG_ROOTFS_RESCUE_SIZE in Bootcode's U-Boot64/include/configs/rtd161x_qa_{board}.h as in device tree
 - For example, define CONFIG_ROOTFS_RESCUE_SIZE as 0x200000 in rtd161x_qa_spi_64.h when device tree has initrd-start=<0x02200000> and initrd-end=<0x02400000> where initrd start and end are defined in kernel source's include/soc/realtek/memory.h as ROOTFS_RESCUE_START and ROOTFS_RESCUE_END respectively
- feed/
 - Gather Firmware Files
 - Edit feed.conf to descript storage layout
 - Three example files, feeds.conf.emmc, feeds.conf.spi, feeds.conf.nand



feed.conf Content

- storage
 - Define storage type
- storage_size
 - Define size of storage in unit of bytes
- storage_align
 - Define the alignment of storage block in unit of bytes
- storage_eraseblock_size (NAND only)
 - Define the size of erase block in unit of bytes
- storage_start_address
 - The start address of available are for storing firmware. DO NOT CHANGE!
- lzma
 - Use LZMA to compress bluecore.audio and kernel. Recommend for SPI

feed.conf Content

- bootargs
 - Insert U-Boot variables
 - Configuration File
 - bootargs.conf.emmc, bootargs.conf.spi, bootargs.conf.nand
- Content of bootargs.conf.{emmc,spi,nand}
 - ethaddr=00:10:20:30:40:50
 - gatewayip=192.168.100.254
 - ipaddr=192.168.100.1
 - netmask=255.255.255.0
 - SPI Kernel Arguments
 - kernelargs=mtdparts=RtkSFC:1024k(U-Boot)ro,64k(FWtbl)ro,128k(Factory),10944k(FW)ro,4096k(Free),128k(oops) init=/etc/init root=/dev/sda1 rootfstype=ext4 rootwait loglevel=8
 - eMMC Kernel Arguments
 - kernelargs=init=/etc/init root=/dev/mmcblk0p1 rootfstype=squashfs rootwait loglevel=8
 - NAND Kernel Arguments
 - kernelargs=init=/etc/init overlay=/dev/ubi1_0 overlayfs=ubifs rootwait loglevel=8
 - For NAND, build_image.sh script will generate necessary arguments (i.e., mtdparts=, ubi.mtd=, root=, and rootfstype=) according to feeds.conf, and append them to kernelargs



feed.conf Content/bootargs

- Content of bootargs.conf.{emmc,spi,nand}
 - ethaddr
 - Device MAC address
 - gatewayip → Bootcode ifconfig
 - ipaddr → Bootcode ifconfig
 - netmask → Bootcode ifconfig
 - Kernelargs
 - Part of kernel boot arguments
 - Kernel boot arguments is combined with kernelargs and bootargs in DTB files
 - Bootargs = bootargs in dtb + kernelargs
 - Kernelargs could be edited in bootcode and easy to change
 - Init=/et/init or /lib/systemd
 - root=/mmcblk0px
 - rootfstype=squashfs or ext4
 - loglevel



feed.conf Content

- secure_image=[y/n] (default: n)
 - Build secure image, only support EMMC now
- aes_key=[key file] (default: aes_128bit_key_dead.bin)
- rsa_private_key=[key file] (default: rsa_key_2048.fw.pem)
 - Key files are put in the feed/ directory



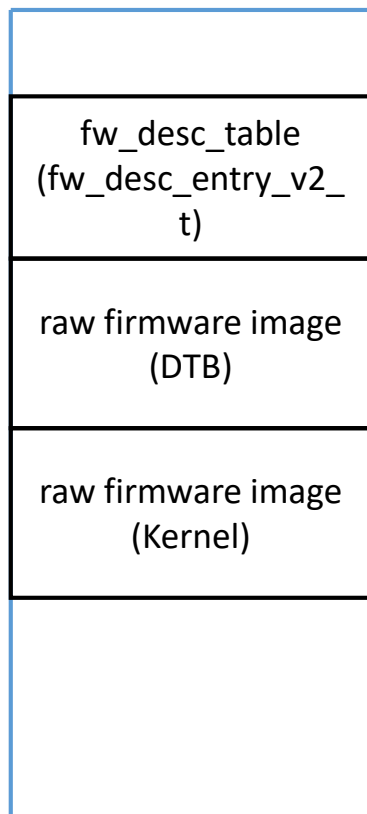
feed.conf Content/secure image

- All firmwares are AES-128-bit encrypted and appended with RSA-signed signature during build stage
- All firmwares are decrypted and check signature validation during booting stage
- AES 128-bit key:
 - should be burned into OTP in advance
 - Encrypt firmware using AES-128-ECB
- RSA private key:
 - 2048-bit, and the public key is burned in storage media with uboot and loaded to memory by FSBL before loading firmwares

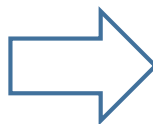
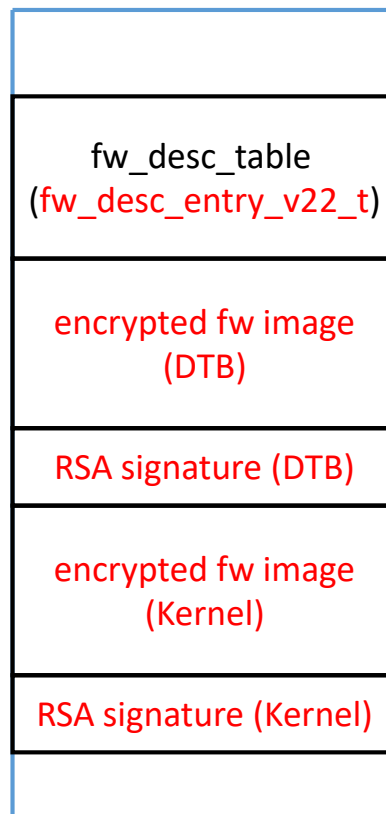


feed.conf Content/secure image

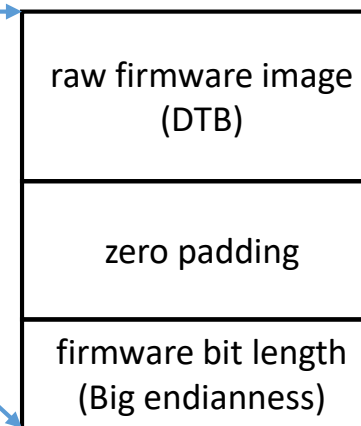
Non-Secure firmware



Secure firmware



AES
Decryption





feed.conf Content

- update_1stfw, update_2ndfw
 - Image-Builder Supports Dual Firmware Updating.
 - Set to 'y' to update the set of firmwares.
 - Set to 'n' to ignore the set of firmwares while burning the storage.
- seqnum_1stfw, seqnum_2ndfw
 - Firmware version
 - Bootcode compares seqnum to decide the latest firmware set. Bigger number is the latest firmware.



feed.conf Content

- kerneldtb_file
 - Filename of DTB for normal booting
- kerneldtb_zone
 - The size of a storage area for storing the kerneldtb_file
- kerneldtb_1stfw_addr, kerneldtb_2ndfw_addr
 - The start add of first/second kerneldtb_zone

ALL values MUST aligned by storage_align, in unit of bytes.



feed.conf Content

- rescuedtb_file
 - Filename of DTB for rescue system booting
- rescuedtb_zone
 - The size of a storage area for storing the rescuedtb_file
- rescuedtb_1stfw_addr, rescuedtb_2ndfw_addr
 - The start add of first/second rescuedtb_zone

ALL values MUST aligned by storage_align, in unit of bytes.



feed.conf Content

- rescuefs_file
 - Filename of initramfs image for rescue system booting
 - The example file is rescue-rootfs/rescue_rootfs.cpio.gz
- rescuefs_zone
 - The size of a storage area for storing the rescuedtb_file
- rescuefs_1stfw_addr, rescuefs_2ndfw_addr
 - The start add of first/second rescuefs_zone

ALL values MUST aligned by storage_align, in unit of bytes.



feed.conf Content

- bluecore_file
 - Filename of bluecore.audio
 - The example file is Packages/fw/bluecore.audio/bluecore.audio.zip
 - Unzip it first
- bluecore_zone
 - The size of a storage area for storing the bluecore_file
- bluecore_1stfw_addr , bluecore_2ndfw_addr
 - The start add of first/second bluecore_zone

ALL values MUST aligned by storage_align, in unit of bytes.



feed.conf Content

- kernel_file
 - Filename of kernel image
 - The file is kernel/arch/arm64/boot/Image
- kernel_zone
 - The size of a storage area for storing the kernel_file
- kernel_1stfw_addr , kernel_2ndfw_addr
 - The start add of first/second kernel_zone

ALL values MUST aligned by storage_align, in unit of bytes.



feed.conf Content

- bootlogo_file
 - Filename of boot logo image
 - For videoplayback configuration only
- bootlogo_zone
 - The size of a storage area for storing the kernel_file
- bootlogo_1stfw_addr , bootlogo_2ndfw_addr
 - The start address of bootlogo_zone
 - By default, there is only one bootlogo file will be placed in the storage. Therefore, addr. of bootlogo in two fw. entries are the same.
- For more information about customizing bootlogo, please read the readme file for boot logo.(readme.bootlogo.pdf)

ALL values MUST aligned by storage_align, in unit of bytes.



feed.conf Content

- `bootpart_dir`
 - Directory name of root filesystem
 - For eMMC and NAND only
 - Customer can build rootfs by themselves
- `bootpart_type`
 - For eMMC, support squashfs only; for NAND, support squashfs and ubifs
- `bootpart_zone`
 - The size of a storage area for storing the boot partition
- `bootpart_addr`
 - The start address of `bootpart_zone`
- `bootpart_name` and `bootpart_ini` (NAND only)
 - MTD partition name and configuration ini-file for ubinize, respectively

ALL values MUST aligned by `storage_align`, in unit of bytes.



feed.conf Content (EMMC and NAND only)

- normalpart_count
 - Count of normal partitions which decides how many normal partitions will be created (MAX=3)
- normalpartX_type (X=1 or 2 or 3)
 - normal partition could be treated as swap, overlay or general purpose partition according to this value (swap, overlay, or ext4 for eMMC; ubifs for NAND)
 - overlay: specify this overlay partition in kernelargs
- normalpartX_file
 - if normalpartX_type is ext4, this will be the filename of ext4 image which will be write on storage
- normalpartX_zone
 - Partition size of normalpartX
- normalpartX_addr
 - Starting address of normalpartX
- normalpartX_name, normalpartX_dir, normalpart1_ini (NAND only)
 - MTD partition name, the directory for UBIFS filesystem, and configuration ini-file for ubinize, respectively

ALL values MUST aligned by storage_align, in unit of bytes.



feed.conf Content (spi only)

- initramfs_file
 - File name of initramfs cpio compressed image
- initramfs_zone
 - The size of storage area for storing initramfs_file
- initramfs_addr
 - The starting address of initramfs_zone

ALL values MUST aligned by storage_align, in unit of bytes.



How to Use Image-Builder

- Prepare Firmwares
 - Kernel
 - arch/arm64/boot/Image
 - Rescue System DTB
 - arch/arm64/boot/dts/realtek/rtd16xx/rtd-1619-nas-qa-rescue.dtb
 - Normal Operation DTB
 - arch/arm64/boot/dts/realtek/rtd16xx/rtd-1619-nas-mjolnir-2GB.dtb
 - Bluecore.audio
 - Packages/fw/bluecore.audio/bluecore.audio.zip, unzip it
 - Rescue System Root Filesystem
 - Image-Builder/rescue-rootfs/rescue_rootfs.cpio.gz
 - Root Filesystem
 - OpenWRT-LEDE/build_dir/target-aarch64_cortex-a55_glibc/root-realtek
- Copy Firmwares to Image-Builder/feed



How to Use Image-Builder

- Edit feed/feed.conf
 - Reference files, feed.conf.{spi,emmc,nand}
- In Image-Builder
 - Run command ./build-image.sh feed
 - X86/storage_layout/ layout-checker helps to check the position of each firmware zone.

```
william@ubuntu:~/tank/nvs/daily/RTD16xx_SDKRelease/OpenWRT-LEDE/target/linux/realtek/image/rtk-imagefile
File Edit View Search Terminal Help
26.14% of uncompressed inode table size (86216 bytes)
Directory table size 24156 bytes (23.68 Kbytes)
44.13% of uncompressed directory table size (54763 bytes)
Number of duplicate files found 272
Number of inodes 2453
Number of files 1856
Number of fragments 133
Number of symbolic links 457
Number of device nodes 0
Number of fifo nodes 0
Number of socket nodes 0
Number of directories 140
Number of ids (unique uids + gids) 1
Number of uids 1
william (1000)
Number of gids 1
william (1000)
#fw twpart name : actual_size(dec) [(hex)] / zone_size(dec) [(hex)]
-----
FREE SPACE : 8376356864 bytes (0x1f3451000) | 0x000200000000
| bootpart squashfs : 37138432 bytes (0x0236b0000) / 167772168 bytes (0x00a000000) | 0x00000c8a7000
| bootpart.img.padding : 167772168 bytes (0x00a000000) | 0x000002ba7000
2ndfw linuxKernel : 15224832 bytes (0x00e850000) / 15269888 bytes (0x00e900000) | 0x000001d1f000
| Image.padding : 15269888 bytes (0x00e900000) | 0x000001be5000
2ndfw audioKernel : 1286144 bytes (0x0013a0000) / 1286144 bytes (0x0013a0000) | 0x000001b2b000
| bluecore.audio.padding : 1286144 bytes (0x0013a0000) | 0x000001b1f000
2ndfw rescueRootFS : 761856 bytes (0x000ba0000) / 761856 bytes (0x000ba0000) | 0x000001ac0000
| rescue_rootfs.cp10.gz.padding : 761856 bytes (0x000ba0000) | 0x000000c3d000
2ndfw rescueDT : 49152 bytes (0x0000c0000) / 49152 bytes (0x0000c0000) | 0x000000b03000
| rescue.dtb.padding : 49152 bytes (0x0000c0000) | 0x000000a49000
2ndfw kernelDT : 49152 bytes (0x0000c0000) / 49152 bytes (0x0000c0000) | 0x000000a30000
| normal.dtb.padding : 49152 bytes (0x0000c0000) | 0x000000a31000
FREE SPACE : 286720 bytes (0x000400000) | 0x000000a31000
1stfw linuxKernel : 15224832 bytes (0x00e850000) / 15269888 bytes (0x00e900000) | 0x000000a31000
| Image.padding : 15269888 bytes (0x00e900000) | 0x000000a31000
1stfw audioKernel : 1286144 bytes (0x0013a0000) / 1286144 bytes (0x0013a0000) | 0x000000a31000
| bluecore.audio.padding : 1286144 bytes (0x0013a0000) | 0x000000a31000
1stfw rescueRootFS : 761856 bytes (0x000ba0000) / 761856 bytes (0x000ba0000) | 0x000000a31000
| rescue_rootfs.cp10.gz.padding : 761856 bytes (0x000ba0000) | 0x000000a31000
1stfw rescueDT : 49152 bytes (0x0000c0000) / 49152 bytes (0x0000c0000) | 0x000000a31000
| rescue.dtb.padding : 49152 bytes (0x0000c0000) | 0x000000a31000
1stfw kernelDT : 49152 bytes (0x0000c0000) / 49152 bytes (0x0000c0000) | 0x000000a31000
| normal.dtb.padding : 49152 bytes (0x0000c0000) | 0x000000a31000
~/tank/nvs/daily/RTD16xx_SDKRelease/OpenWRT-LEDE/target/linux/realtek/image/rtk-imagefile/workspaces/nas ~/tank/nvs/daily/RTD16xx_SDKRelease/OpenWRT-LEDE/target/linux/realtek/image/rtk-imagefile
william@ubuntu:~/tank/nvs/daily/RTD16xx_SDKRelease/OpenWRT-LEDE/target/linux/realtek/image/rtk-imagefiles
```



How to Use Image-Builder--Output

- Image-Builder/install.img
 - Rescue system use install.img to upgrade firmwares
- Image-Builder/workspace/rescue
 - Rescue system files
 - {spi,emmc,nand}.ulmage
 - rescue.{spi,emmc,nand}.dtb
 - rescue.root.{spi,emmc,nand}.cpio.gz_pad.img



How to use WebUI to build config

- Open Image-Builder/StorageLayoutUI/index.html
- Choose the storage type
 - EMMC
 - Spi
- And corresponding settings
 - [Bootargs](#)
 - [Storage size](#)
 - [Storage align](#)
- Supports Dual Firmware Updating
 - Storage 2nd firmware address
 - [Update firmware](#)
 - [Seqnum firmware](#)

Firmware Layout Customization

version:

1.0.0

project:

nas

debug_level:

- ☒ FAIL_LEVEL
- ☒ INFO_LEVEL
- ☒ LOG_LEVEL
- ☒ DEBUG_LEVEL
- ☒ WARNING_LEVEL
- ☒ UI_LEVEL

storage:

emmc

bootargs:

bootargs.conf.emmc

storage size:

8

GB

align:

4096

Byte

storage 2nd firmware start address:

0x 2871000

lzma: ☒ y ☐ n

update_1stfw: ☒ y ☐ n

update_2ndfw: ☒ y ☐ n

seqnum_1stfw:

0

seqnum_2ndfw:

0

burn_bootpart:

Y

burn_factory: ☒ y ☐ n

How to use WebUI to build config

- Choose six files
 - [Device Tree](#)
 - [Rescue Device Tree](#)
 - [Rescue RootFS](#)
 - [Bluecore](#)
 - [Kernel](#)
 - [Boot Partition](#) (only emmc)
- Zone MUST aligned by storage_align
- Customization [Normal Part](#)
 - Emmc: 0, 1, 2, 3
 - Spi: none

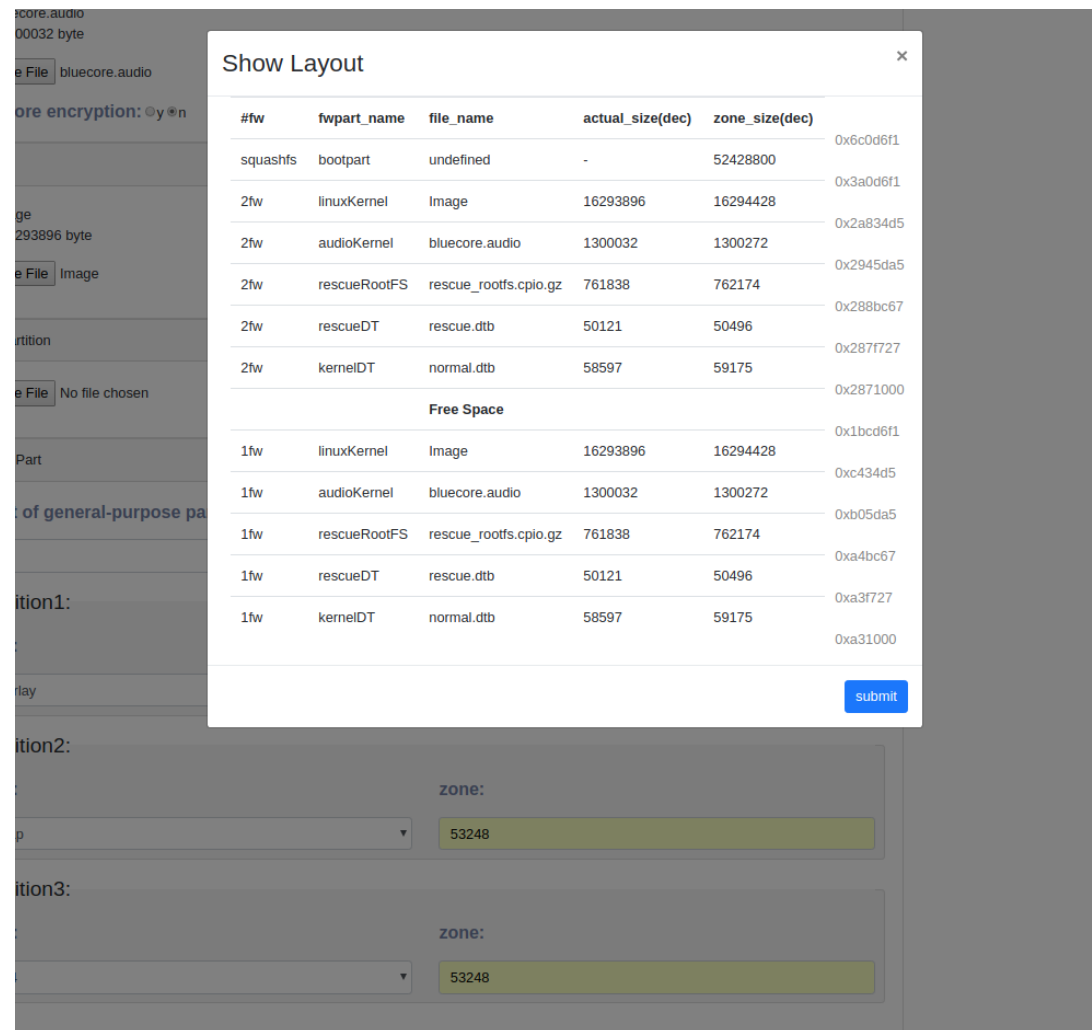
The screenshot displays a web-based configuration interface for building a system image. It features several sections, each with a 'Choose File' button, a 'No file chosen' status, and a 'zone' input field. The sections are: Device Tree, Rescue Device Tree, Rescue RootFS, Bluecore, Kernel, Boot Partition, and Normal Part. The 'Bluecore' section includes a 'Bluecore encryption' toggle set to 'off'. The 'Boot Partition' section has a 'zone' field with the value '52428800'. The 'Normal Part' section includes a 'Count of general-purpose partitions' dropdown menu set to '0'. A 'Show' button is located at the bottom left of the form.

Section	Choose File	No file chosen	zone
Device Tree	Choose File	No file chosen	
Rescue Device Tree	Choose File	No file chosen	
Rescue RootFS	Choose File	No file chosen	
Bluecore	Choose File	No file chosen	
Bluecore encryption: <input type="checkbox"/> y <input checked="" type="checkbox"/> n			
Kernel	Choose File	No file chosen	
Boot Partition	Choose File	No file chosen	52428800
Normal Part	Count of general-purpose partitions 0		

Show

How to use WebUI to build config

- Click “OK”
 - You can review the layout
- click “submit”
 - It can produce feed.conf.{spi, emmc}
 - In /home/Downloads
- Copy feed.conf.{spi, emmc} to
RTD16xx_SDK_Develop/Image-Builder/feed



The screenshot shows a 'Show Layout' dialog box with a table of file layout information. The table has columns: #fw, fwpart_name, file_name, actual_size(dec), zone_size(dec), and a hex address. The data is organized into sections: 'squashfs' (bootpart), '2fw' (linuxKernel, audioKernel, rescueRootFS, rescueDT, kernelDT), and 'Free Space'. Below the table is a 'submit' button. The background shows a file selection interface with a dropdown menu and a 'zone:' label.

#fw	fwpart_name	file_name	actual_size(dec)	zone_size(dec)	
squashfs	bootpart	undefined	-	52428800	0x6c0d6f1
2fw	linuxKernel	Image	16293896	16294428	0x3a0d6f1
2fw	audioKernel	bluecore.audio	1300032	1300272	0x2a834d5
2fw	rescueRootFS	rescue_rootfs.cpio.gz	761838	762174	0x2945da5
2fw	rescueDT	rescue.dtb	50121	50496	0x288bc67
2fw	kernelDT	normal.dtb	58597	59175	0x287f727
Free Space					
1fw	linuxKernel	Image	16293896	16294428	0x1bcd6f1
1fw	audioKernel	bluecore.audio	1300032	1300272	0xc434d5
1fw	rescueRootFS	rescue_rootfs.cpio.gz	761838	762174	0xb05da5
1fw	rescueDT	rescue.dtb	50121	50496	0xa4bc67
1fw	kernelDT	normal.dtb	58597	59175	0xa3f727
0xa31000					

submit