# AN13872

## Enabling SWUpdate on i.MX 6ULL, i.MX 8M Mini, and i.MX 93

Rev. 5 — 18 March 2024 Application note

**Document information**

| Information | Content |
|---|---|
| Keywords | AN13872, i.MX 6, i.MX 7, i.MX 8, i.MX 9, OTA, Linux |
| Abstract | SWUpdate is a Linux Update agent to provide an efficient and safe way to update an embedded Linux system in the field. SWUpdate supports local and OTA updates, as well as multiple update strategies, and it is designed with security function. |

# 1 Introduction

SWUpdate is a Linux Update agent to provide an efficient and safe way to update an embedded Linux system in the field. SWUpdate supports local and OTA updates, as well as multiple update strategies, and it is designed with a security function.

Generally, SWUpdate is a framework that provides various configurable features:

- Update of all components of a device (`rootfs`, kernel, bootloader, microcontroller FW).
- Installed on embedded media (eMMC, SD, Raw NAND, UBIFS, NOR, and SPI-NOR flashes).
- Streaming mode without temporary copies of artifacts.
- Multiple interfaces (local and OTA) for getting software.
  - Local storage (USB, and so on)
  - Integrated web server
  - Integrated REST client connector to Hawkbit for fleet updates
  - Remote server download
- Software delivered as images, gzipped tarball, and so on.
- Allow custom handlers to install FPGA firmware and microcontroller firmware through custom protocols.
- Delta updates based on `librsync`.
- Fail-safe and atomic update.
- Lua interpreter to extend the updated rules as required.
- Cryptographic sign and verification of updates.
  - Support for OpenSSL
  - Support for mbedTLS
  - Support for wolfSSL
- Encryption of artifacts through a symmetric AES key.

The i.MX 6ULL is a high-performance and ultra-efficient processor family with featuring NXP advanced implementation of the single Arm Cortex-A7 core, which operates at speeds of up to 792 MHz.

The i.MX 8M Mini is a family of products focused on delivering an excellent video and audio experience. It combines media-specific features with high-performance processing optimized for low-power consumption.

The i.MX 93 applications processors deliver efficient Machine Learning (ML) acceleration and advanced security with integration.

EdgeLock secure enclave to support energy-efficient edge computing.

This application note gives a general view of SWUpdate. It helps users to set up SWUpdate on the i.MX 6ULL, i.MX 8M Mini, and i.MX 93 chips, which can also be spread to other i.MX platforms.

This application note helps the audience to:

- Get familiar with SWUpdate.
- Configure SWUpdate for some general cases.
- Build SWUpdate inside images from Yocto.
- SWUpdate update image generation.
- Get familiar with the SWUpdate `sw-description` file.
- Enable the Mongoose update server.

SWUpdate provides various features, and the example in this document only demonstrates some common features.

# 2 Acronyms and abbreviations

**Table 1. Acronyms and abbreviations**

| Acronym | Meaning |
|---|---|
| AN | Application Note |
| DTB | Device Tree Blob |
| DTS | Device Tree Source |
| FOSS | Free/Open-Source Software |
| GUI | Graphical User Interface |
| OTA | Over-The-Air |
| PIO | Programming Input/Output Model |
| SoC | System on Chip |
| SWUpdate | Software Update |

# 3 SWUpdate introduction

As Embedded Systems become more complex, their software reflects the augmented complexity. It is vital that the software on an embedded system can be updated in a reliable way, as new features and fixes are added.

SWUpdate is a Linux Update agent to provide an efficient and safe way to update an embedded Linux system in the field. SWUpdate supports local and OTA updates, as well as multiple update strategies, and it is designed with a security function.

## 3.1 Compared with other update systems

Besides SWUpdate, there are also other update systems on the Linux operating system.

**Table 2. Update systems on the Linux operating system**

| Mechanism | Type | Disk layout | Rootfs | Updates from | Updates what | Code stability | OE/Yocto integration | Resource requirements | | Failure resilience | Complexity | Downtime | Security |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | on server | on client | | | | |
| **swupd** | File-based | Flexible | Read/write | HTTP(S) server, local media | Depending on setup | Relatively stable, under active development | meta-swupd | Moderate, suitable for frequent updates | Minimal download, sufficient free space in rootfs required | Favors fast updates over failure resilience | Some plannings required | Minimal, reboot optional | Compatible with Linux IMA, Smack, SELinux. Signed update data, HTTPS transfer protection. |
| **sbabic's swupdate** | Block-based / file-based | Flexible | Depending on setup (read-only supported) | Local and remote (plain HTTP(S) or custom server) | Depending on setup | Code relatively stable, 6 months release cycle | meta-swupdate | Archiving full image per build | Download and write full (compressed) image, zero-copy | Integrated rollback (bootloader support required) | Easy to use (but requires customization!?) | Reboot required | Signed and encrypted images, HTTPS |
| **Mender** | Block-based / file-based | flexible (minimum four partitions), U-Boot, or GRUB as bootloader | Supporting read/write and read-only | Remote using Mender management server (managed mode) or local using CLI (standalone mode) | Complete rootfs, including kernel (built-in). Customizable with Update Modules. | Relatively stable, fully supported and tested upgrade path | meta-mender | Compressed rootfs per build | Download and write compressed rootfs | Integrated rollback | Easy when using meta-mender | Reboot required | HTTPS enforced, signed images |
| **OSTree** | File-based | Flexible, but supporting only a limited set of bootloaders | Read/write, operating system trees bind mounted read-only, / etc and /var writable | Local and remote repositories | Kernel and file system | Relatively stable, significant user base, under active development | Meta-ostree (WIP), meta-updater (public) | Generating commits based on new builds, storing them in a repository | Updating local repository, hard links for sharing unchanged content between deployments | Roll back to a different deployed operating system tree | Some work required | Reboot required | GPG-signed commits |
| **RAUC** | Block-based/ file-based (tar) | Flexible (block-device/ MTD) | Depending on setup (read-only supported) | Depending on setup | Depending on setup (any storage device) | Relatively stable, under active development | meta-rauc | Archiveing full (compressed) image per build | Download and write full (compressed) image | Integrated rollback (requires bootloader support) | Some customizations required | Reboot required | X509-signed update bundles |

## 3.2 Benefits of SWUpdate

The benefits of choosing SWUpdate are as follows:

- SWUpdate is already widely used in many devices in the field. It can provide a reliable way to update your products.


**Figure 1. Devices using SWUpdate**

- Full open-source.
  SWUpdate is a full open-source project, which means that 100 % of the code is released under the open-source license. SWUpdate is integrated and it works with other Free and Open-Source Software (FOSS) projects. For commercial support, see Services - SWUpdate.
- Plenty of features.

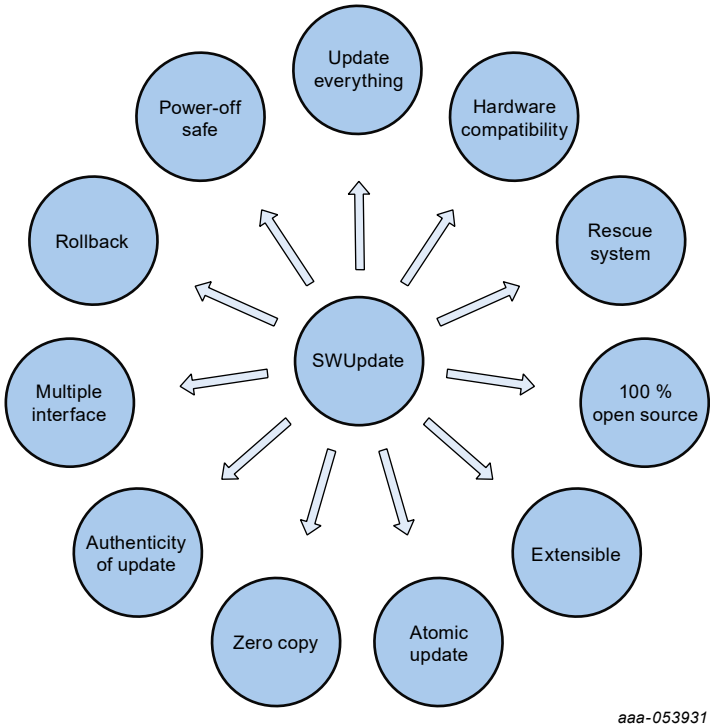



**Figure 2. Features**

# 4 Update strategy

This section describes the update strategy that is used in a general updating procedure.

For more practices, see SWUpdate Best Practice.

## 4.1 Update procedure

To update the OTA, perform the following steps:

1. Create a base image and an update image.
2. Boot the kernel with the base image and enter OTA mode.
3. Download the update images to the target partition.
4. Reboot the system and synchronize the update status.
5. Perform recovery if the update fails.

Figure 3 shows the procedure provided in the demo, and the web server used is Mongoose.

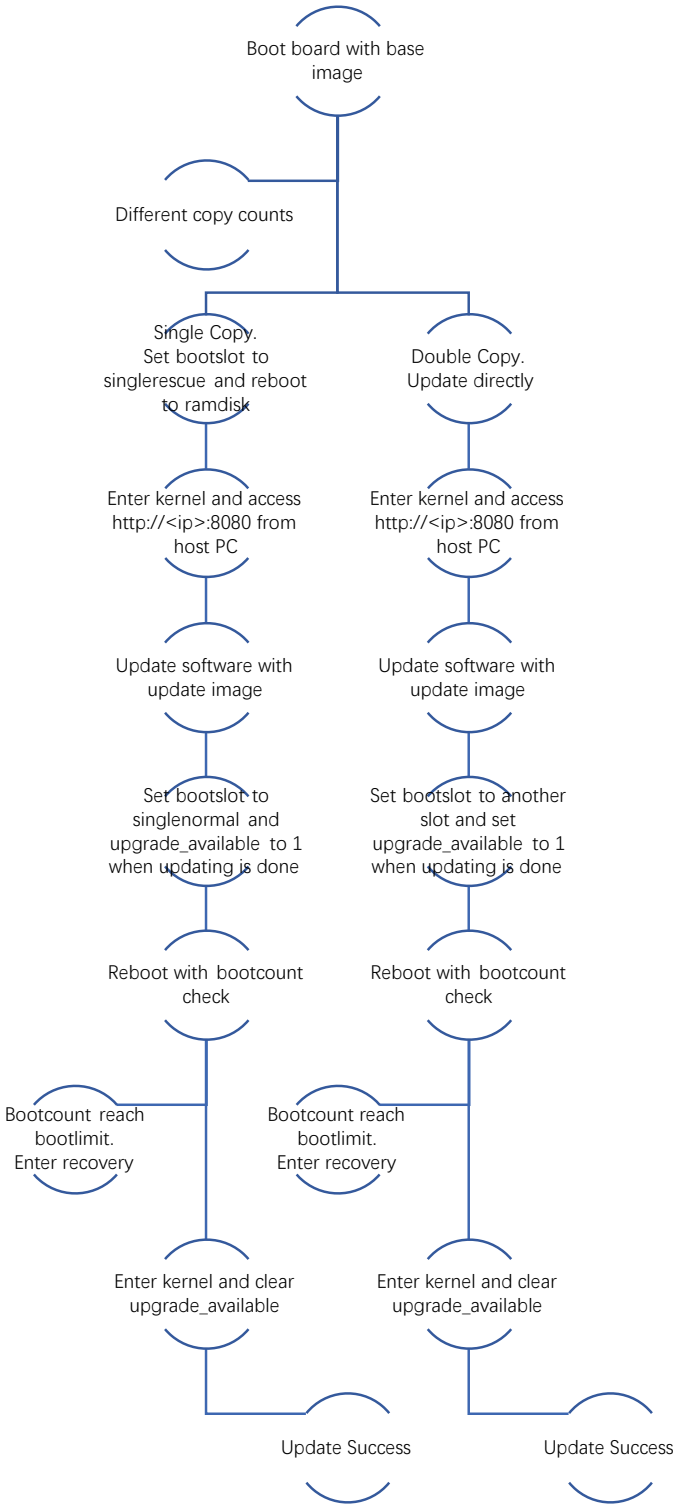



**Figure 3. Update procedure**

## 4.2 Single copy and double copy

A system that can be updated must meet the following requirements:

- It has SWUpdate inside, so that the OTA mode or interface can be launched.
- The mechanism to write downloaded images, that is, a single copy or double copy of the system.

### 4.2.1 Single copy – running as a standalone image

The software upgrade application consists of a kernel (maybe reduced dropping not required drivers) and a small root file system, with the application and its libraries. The whole size is smaller than a single copy of the system software (2.5 MB - 8 MB). The system can be put in **upgrade** mode, simply signaling to the bootloader that the upgrading software must be started (either by using the bootloader environment or GPIO).

The bootloader starts **SWUpdate**, booting the SWUpdate kernel and the `initrd` image as the root file system. Because it runs in RAM, it is possible to upgrade the whole storage. Different from that in the double-copy strategy, the system must reboot to enter update mode.

This concept consumes less space in storage than having two copies, but it does not guarantee a fall-back without updating again the software. However, it guarantees that the system enters automatically in upgrade mode when the productivity software is not found or corrupts, and when the upgrade process is interrupted for certain reasons.

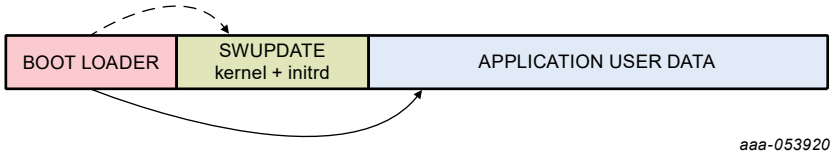



**Figure 4. Single copy**

Summary:

- Standalone image consists of `kernel`/`dt` + `initrd`.
- Smaller than the entire system.
- Bootloader in charge of loading standalone images.
- The system must reboot to enter the update process.

### 4.2.2 Double copy with fall-back

If there is enough space in the storage to save two copies of the whole software, it is possible to guarantee that there is always a working copy even if the software update is interrupted or a power off occurs.

Each copy must contain the kernel, the root file system, and each further component that can be updated. It requires a mechanism to identify which version is running.

SWUpdate must be inserted in the application software. The application software triggers the SWUpdate when an update is required. The SWUpdate aims to update the standby copy, leaving the running copy of the software untouched.

A synergy with the bootloader is often necessary, because the bootloader must decide to start which copy. In addition, it must be able to switch between the two copies. After a reboot, the bootloader decides to run which copy.

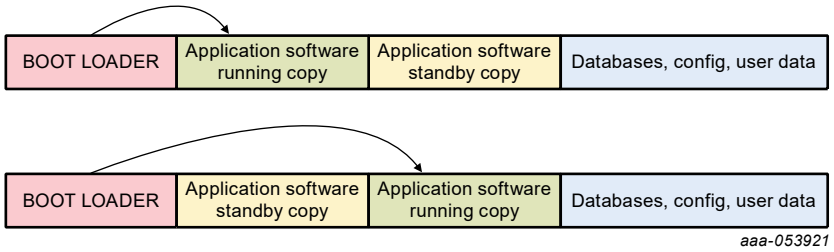



**Figure 5. Double copy with fall-back**

Summary:

- Double copy requires twice the space than a single copy.
- Double copy guarantees that there is always a working copy.
- In double copy, the bootloader is in charge of booting proper images.

## 4.3 Bootloader upgrade

As the bootloader is used to check the update result, switch the boot system and perform the recovery. Generally, there is no second copy of the bootloader. Usually, it is not recommended to upgrade the bootloader. The device might get bricked when a power down occurs during the update. Some SOCs allow multiple copies of the bootloader. However, there is no general solution for this issue because it is hardware-dependent. i.MX series chips have a secondary boot for two copied bootloader, but a bootloader upgrade is still not recommended.

***Note:***

*It is not guaranteed that all i.MX chips support a secondary boot image.*

*For how to create a boot image for secondary boot, see Section 13.2.*

## 4.4 Power failure recovery

When a power down occurs, the system must be guaranteed to work again, which means to restart SWUpdate or restore an old copy of the software.

Generally, the behavior can be split according to the chosen scenario:

- Single copy: SWUpdate is interrupted, and the update transaction does not end successfully. The bootloader can restart SWUpdate, and can update the software again.
- Double copy: SWUpdate does not switch between the standby and current copy. The same version of the software, which is not touched by the update, is restarted.

For safe purposes, SWUpdate and the bootloader must exchange some information. The bootloader must detect if an update was interrupted due to power down, and restarts SWUpdate until an update is successful. U-Boot has a power-safe environment which SWUpdate can read and change to communicate with them. SWUpdate sets a variable as a flag when it starts to update the system and resets the same variable after completion. The bootloader can read this flag to check if an update was running before power off.

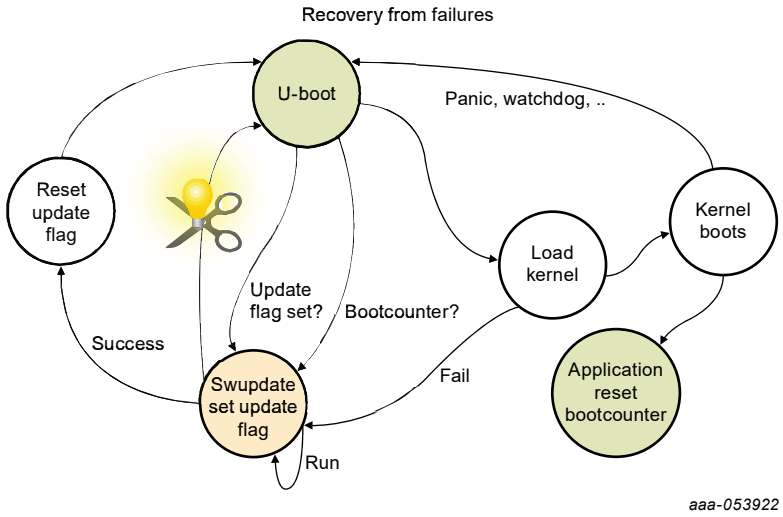



**Figure 6. Recovery from failures**

## 4.5 Upgrading SWUpdate

SWUpdate must be used in the whole development process, replacing the customized process to update the software during the development. Before going into production, test the SWUpdate for a project.

If SWUpdate must be updated, the update cannot be safe if there is only one copy of SWUpdate in the storage. Safe updates can be guaranteed only if SWUpdate is duplicated.

If SWUpdate is a part of the upgraded image, to avoid this issue, use either of the followings:

- Get two copies of SWUpdate.
- Take the risk but have a rescue procedure using the bootloader.

# 5 Building SWUpdate

SWUpdate is supported and integrated in the modem embedded Linux build system, such as Yocto, Debian, and buildroot.

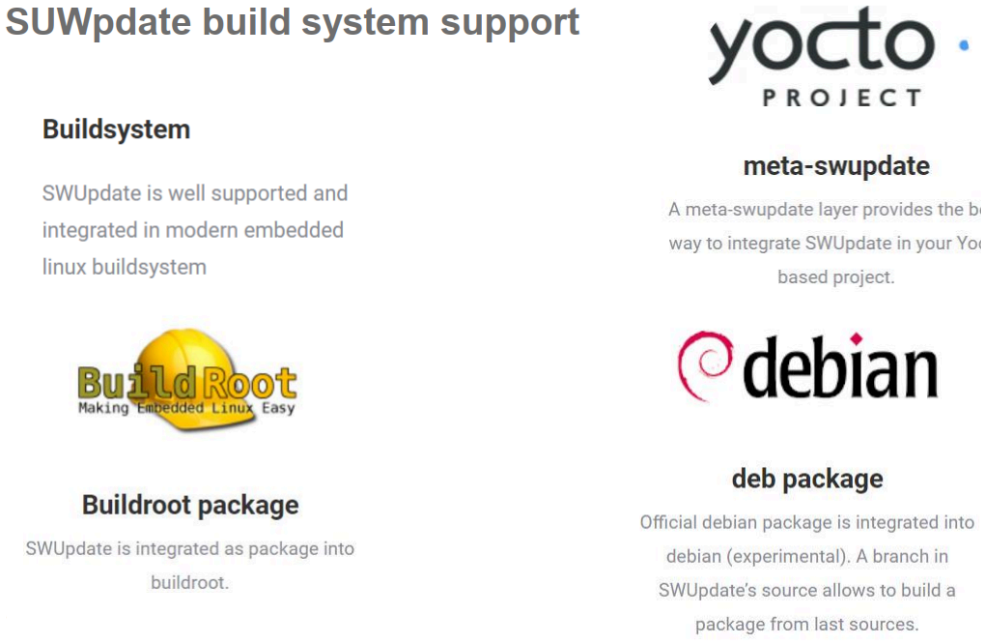



**Figure 7. SWUpdate build system support**

As i.MX chips use Yocto to build Linux images, this section demonstrates using Yocto. This section describes the repositories for SWUpdate and how to use these repositories to build SWUpdate into `rootfs`.

## 5.1 Repositories provided by SWUpdate

SWUpdate is an open-source project and all these repositories can be found on Stefano Babic.

**Table 3. Repositories provided by SWUpdate**

| Repository name | Comment |
|---|---|
| `swupdate` | Source code of SWUpdate. |
| `meta-swupdate` | Yocto layer for deploy tool. It provides Yocto recipes to generate a SWUpdate root file system. For different Yocto version, use different branch. |
| `libubootenv` | Generic library and tools to access and modify U-Boot environment from User Space. |
| `meta-swupdate-boards` | Examples on how to use SWUpdate.<br>These examples may not be useful, as new boards are rarely supported in this way. |
| `SWUpdateGUI` | A simple GUI for SWUpdate in rescue mode.<br>It was not used before and it is not demonstrated in this document. |

## 5.2 Repositories provided by NXP

To support i.MX chips, some repositories are provided to simplify the usage of SWUpdate.

**Table 4. Repositories provided by NXP**

| Repository name | Comment |
|---|---|
| `meta-swupdate-imx` | Yocto layer of SWUpdate for i.MX chips. Similar to meta-swupdate, in this repository, we also provide various branches to support different Yocto versions. For example, langdale, kirkstone_5.15.32_2.0.0, kirkstone_5.15.71_2.2.0 and mickledore_6.1.36_2.1.0, and so on. |

**Table 4. Repositories provided by NXP*...continued***

| Repository name | Comment |
|---|---|
| `swupdate-scripts` | Script tools to create base and update images. |

## 5.3 Building SWUpdate inside rootfs from Yocto

To build a customized SWUpdate from Yocto, perform the following steps, which take Kirkstone as an example:

1. Clone Yocto.
   The command (using 5.15.32 as an example) is as below:

```
mkdir fsl-release-bsp-5.15.32
cd fsl-release-bsp-5.15.32
repo init -u https://source.codeaurora.org/external/imx/imx-manifest -b imx-
linux-kirkstone -m imx-5.15.32-2.0.0.xml
repo sync
```

2. Clone `meta-swupdate`.
   ***Note:*** *The branch must match the Yocto version.*

```
cd fsl-release-bsp-5.15.32/source
git clone https://github.com/sbabic/meta-swupdate.git -b kirkstone
```

3. Clone `meta-swupdate-imx`.
   ***Note:*** *The branch must match the Yocto and kernel version.*

```
cd fsl-release-bsp-5.15.32/source
git clone https://github.com/nxp-imx-support/meta-swupdate-imx.git -b
 kirkstone_5.15.32_2.0.0
```

4. To modify Yocto source, add layers for `meta-swupdate` and `meta-swupdate-imx`.
   The patch is as below:

```
diff --git a/sources/meta-imx/tools/imx-setup-release.sh b/sources/meta-imx/
tools/imx-setup-release.sh
--- a/sources/meta-imx/tools/imx-setup-release.sh
+++ b/sources/meta-imx/tools/imx-setup-release.sh
@@ -184,6 +186,8 @@ echo "BBLAYERS += \"\${BSPDIR}/sources/meta-openembedded/
meta-networking\"" >> $
echo "BBLAYERS += \"\${BSPDIR}/sources/meta-qt6\"" >> $BUILD_DIR/conf/
bblayers.conf
+echo "BBLAYERS += \"\${BSPDIR}/sources/meta-swupdate\"" >> $BUILD_DIR/conf/
bblayers.conf
+echo "BBLAYERS += \"\${BSPDIR}/sources/meta-swupdate-imx\"" >> $BUILD_DIR/
conf/bblayers.conf
```

5. To modify Yocto source, add packages to the image installation list.
   The patch is as below:

```
diff --git a/sources/meta-imx/tools/imx-setup-release.sh b/sources/meta-imx/
tools/imx-setup-release.sh
--- a/sources/meta-imx/tools/imx-setup-release.sh
+++ b/sources/meta-imx/tools/imx-setup-release.sh
@@ -158,6 +158,8 @@ echo >> conf/local.conf
echo "# Switch to Debian packaging and include package-management in the
 image" >> conf/local.conf
echo "PACKAGE_CLASSES = \"package_deb\"" >> conf/local.conf
echo "EXTRA_IMAGE_FEATURES += \"package-management\"" >> conf/local.conf
```

```
+echo "IMAGE_INSTALL:append = \" lua swupdate swupdate-www swupdate-progress
 swupdate-client swupdate-tools-ipc u-boot-imx u-boot-fw-utils systemd-swusys
 json-c\"" >> conf/local.conf
+echo "IMAGE_FSTYPES = \" ext4 ext4.gz wic.bmap wic.gz\"" >> conf/local.conf
```

6. (Optional) Add `mmcblks` to `udev` mount ignore list.
   The patch is as below:

```
--- a/sources/poky/meta/recipes-core/udev/udev-extraconf/mount.ignorelist
+++ b/sources/poky/meta/recipes-core/udev/udev-extraconf/mount.ignorelist
@@ -3,3 +3,4 @@
/dev/mtdblock
/dev/md
/dev/dm-*
+/dev/mmcblk?p[1-12]
```

   This patch prevents partitions from being automatically mounted. When applying the SWUpdate dual-copy solution, developers probably do not want users to easily discover or access the partitions where the second copy is stored. For example, when using slot A `rootfs`, users may not want to see slot B partitions in */run/media*.
7. Build SWUpdate inside images from Yocto.
   The command is as below:

```
cd fsl-release-bsp-5.15.32/
DISTRO=fsl-imx-wayland MACHINE source imx-setup-release.sh -b build-xwayland-
swupdate
bitbake swupdate-image
bitbake core-image-base
```

   ***Note:***
   *`bitbake core-image-base` builds an SWUpdate inside the core image. `bitbake swupdate-image` builds an SWUpdate inside the `ramfs` image. The `ramfs` image can be used as a rescue system in single copy and double copy. Besides `core-image-base`, other build targets can also build SWUpdate inside images.*
   *For i.MX 8M Mini, the MACHINE is `imx8mm-lpddr4-evk`.*
   *For i.MX 93, the MACHINE is `imx93evk` and choose BSP release of version 6.1.22 or higher here.*

***Note:***

*The Yocto build adds SWUpdate mandatory configurations:*

- */etc/fw_env.config*: *Configuration for `fw_printenv/fw_setenv`.*
- */etc/hwrevision*: *Hardware revision, SWUpdate to check the hardware compatibility.*
- */etc/swupdate.cfg*: *The SWUpdate configuration file. In this demo, it tells the location of the public key.*
- */etc/swu_public.pem*: *Public key for sign image checking.*

## 5.4 Customization of SWUpdate

### 5.4.1 Mongoose or Suricatta

SWUpdate provides two daemon modes for web download: Mongoose and Suricatta. Each of them provides a different method to do remote updating.

Table 5 lists the differences between Mongoose and Suricatta.

**Table 5. Differences between Mongoose and Suricatta**

| | Daemon mode | | Comments |
|---|---|---|---|
| | Mongoose | Suricatta | |
| Local Server or not? | Yes | No | Mongoose is an integrated web server to allow remote updating. It provides a web server, web interface, and web application on the board. Suricatta regularly polls a remote server to update, download, and install them. Therefore, on the board, Mongoose is a host server and Suricatta is a client. |
| Complexity | Simple | Complex | As Mongoose runs a server on the board, it cannot provide lots of features (due to performance or storage limitations of embedded devices).<br>While Suricatta is a client that polls a remote server, complex features can be added. |
| Parallel Run | Yes | Yes | Both Suricatta and Mongoose can run in parallel at the same time. |
| User | Demo | Commerce | For demo purposes, we use Mongoose most of the time, but commercial users may choose Suricatta or both. |
| Related configurations | `CONFIG_MONGOOSE`<br>`CONFIG_MONGOOSEIPV6`<br>`CONFIG_MONGOOSESSL` | `CONFIG_SURICATTA`<br>`CONFIG_SURICATTA_HAWKBIT` | By default, in *meta-swupdate-imx/recipes-support/swupdate/swupdate/defconfig*, both MONGOOSE and SURICATTA are enabled. |
| Webpage | Mongoose | Suricatta | — |

***Note:***

*Besides the modes for web access, SWUpdate also supports updating images from U-Disk, SD card, and so on.*

### 5.4.2 fw_setenv and fw_printenv

`fw_printenv` and `fw_setenv` are the applications that can modify the U-Boot environment variables from the Linux side. In the demo, Yocto is built from `u-boot-fw-utils`, and it can also be built from U-Boot. This is the way to communicate between U-Boot and Linux.

If u-boot is not used, develop a similar program to read and write environment variables.

### 5.4.3 Security

#### 5.4.3.1 Security mechanisms

It is important that a device must be safely updated, and it can verify if the delivered image comes from a known source and it is not corrupted when introducing any malware.

To achieve this goal, SWUpdate provides the following mechanisms:

- Image signing.
  SWUpdate provides signing for compound images and subimages.
- Hash verification.
  SWUpdate combines the signed `sw-description` with the verification of hashes for each single image. This means that the installer only accepts the `sw-description` generated by a verified source. `sw-`

`description` contains hashes for each subimage to verify that each delivered subimage belongs to the release.

***Note:*** *More security mechanisms, such as, Manufacturing Protection mechanism, can be found in the Secure Over-the-Air Prototype for Linux Using CAAM and Mender (document AN12900)*

#### 5.4.3.2 Security configuration macros

In SWUpdate, the security configuration macros listed in Table 6 are used in the demo.

**Table 6. Security configuration macros**

| Configuration macro | Description |
|---|---|
| `CONFIG_HASH_VERIFY` | The hash of each image in the update image is generated and filled in `sw-description`. |
| `CONFIG_SIGNED_IMAGES` | All images in the update image are signed. |
| `CONFIG_ENCRYPTED_IMAGES` | All images in the update image are encrypted. |

The related configuration file is `<yocto_dir>/sources/meta-swupdate-imx/recipes-support/swupdate/swupdate/defconfig`.

To enable security features or remove them to disable security features, add these configuration macros in the *defconfig* file.

To enable security features, the example code is as below:

```
+CONFIG_HASH_VERIFY=y
+CONFIG_SIGNED_IMAGES=y
+CONFIG_ENCRYPTED_IMAGES=y
```

***Note:***

- *This can also be done with SWUpdate menuconfig.*
- *The security feature depends on `openssl/wolfssl/mbedtls` (optional) for cryptographic operations. If crypto hardware acceleration is enabled in `openssl/wolfssl/mbedtls`, security in SWUpdate can have a better security performance.*

SWUpdate supports various different security configuration files. See meta-swupdate: building with Yocto.

#### 5.4.3.3 Generating a key

Depending on different encryption algorithm choices, the commands to encrypt are different. The following is a common usage using RSA.

```
Private key: openssl genrsa -aes256 -out priv.pem
Public key: openssl rsa -in priv.pem -out swu_public.pem -outform PEM -pubout
```

In Yocto, *swu_public.pem* must be put into *source/meta-swupdate-imx/recipes-support/swupdate/swupdate/swu_public.pem*. Then it will be built into *rootfs*. The path on the target board is */etc/swu_public.pem*.

For other key generation and sign methods, see Update images from verified source.

### 5.4.4 Lua scripts

In SWUpdate, the Lua interpreter is linked to SWUpdate and runs in the context of the SWUpdate process without forking a child process. It is used to extend the runtime behavior.

For example, Lua scripts can be used as an embedded script in the *sw-description* file to detect boot pins, check update partitions, and so on.

Related configuration files are as below:

```
CONFIG_LUA=y
CONFIG_LUAPKG="lua"
CONFIG_LUASCRIPTHANDLER=y
CONFIG_HANDLER_IN_LUA=y
```

Generally, the Lua interpreter in SWUpdate uses pure Lua syntax. However, when it is used as an embedded script in `sw-description`, as usage of double quotes can interfere with the parser, each double quote must be escaped.

This means that a simple Lua code as `print ("Test")` must be changed to `print (\"Test\")`. Otherwise, the parser regards that it has the closure of the script, and this causes an error.

For more details about Lua and embedded scripts, see SWUpdate: syntax and tags with the default parser.

# 6 Creating a base image and an update image

In SWUpdate, users must create two types of images, base image and update image.

The base image is known as `.sdcard` or `.wic` image, and the update image is known as `.swu` image.

## 6.1 `swupdate-script` repository

To facilitate the procedure of creating images, some scripts are created in `swupdate-script`.

The `swupdate-scripts` repository holds the scripts that are used to generate the base and update images.

Directories and files are listed as follows:

```
├── base_image_assembling
│   ├── assemble_base_image.sh
│   ├── common
│   │   └── swu_dualslot_7.5G.pt
│   ├── readme.txt
│   ├── slota -> slota
│   ├── slotb -> slotb
│   └── workspace
│       ├── 00-swu_7.5G.pt -> ../common/swu_dualslot_7.5G.pt
│       ├── 01-imx-boot -> ../slota/imx-boot-imx8mmevk-sd.bin-flash_evk
│       ├── 02-Image -> ../slota/Image
│       ├── 03-imx8mm-evk.dtb -> ../slota/imx8mm-evk.dtb
│       ├── 04-swupdate-image -> ../slota/swupdate-image-imx8mmevk.cpio.gz.u-
boot
│       ├── 05-boot_pt -> ../common/slota_boot_pt_120M.mirror
│       ├── 06-imx-image-multimedia -> ../slota/imx-image-multimedia-
imx8mmevk.ext4
│       ├── 12-Image -> ../slotb/Image
│       ├── 13-imx8mm-evk.dtb -> ../slotb/imx8mm-evk.dtb
│       ├── 15-boot_pt -> ../common/slotb_boot_pt_120M.mirror
│       ├── 16-imx-image-multimedia -> ../slotb/imx-image-multimedia-
imx8mmevk.ext4
│       └── padding_file_create.sh
├── boards
│   ├── cfg_boards.cfg
│   ├── cfg_imx6ull_base.cfg
```

```
│   ├── cfg_imx6ull_update_image.cfg
│   ├── cfg_imx6ullwch_base.cfg
│   ├── cfg_imx6ullwch_update_image.cfg
│   ├── cfg_imx8mm_base.cfg
│   ├── cfg_imx8mm_update_image.cfg
│   ├── cfg_imx93_base.cfg
│   ├── cfg_imx93_update_image.cfg
│   ├── sw-description-imx6ull-sd-dualcopy-image.template
│   ├── sw-description-imx6ull-sd-singlecopy-image.template
│   ├── sw-description-imx6ullwch-emmc-dualcopy-image.template
│   ├── sw-description-imx6ullwch-emmc-dualcopy-image.template.cmdline
│   ├── sw-description-imx8mm-emmc-dualcopy-image.template
│   ├── sw-description-imx8mm-sd-dualcopy-image.template
│   └── sw-description-imx93-emmc-dualcopy-image.template
├── legacy
├── LICENSE.txt
├── README.md
├── update_image_build
│   ├── emmc_bootpart.sh
│   ├── env_set_bootslot.sh
│   ├── priv.pem
│   ├── readme.txt
│   ├── slot_update -> ./slot_update
│   ├── swu_generate_part_img.sh
│   └── swu_update_image_build.sh
└── utils
    └── utils.sh
```

Descriptions:

- `base_image_assembling`:
  This directory holds the script to generate the base image to be updated. In this demo, the base image is built based on 5.4.70_2.3.0.
- `assemble_base_image.sh`:
  Script to assemble images to a base image. See `-h` for help:

```
nxa08304@lsv11258:~/data/SWUpdate/swupdate_scripts/base_image_assembling$ ./assemble_base_image.sh -h
./assemble_base_image.sh - generate update image
-o specify output image name. Default is swu_<SLOT>_rescue_<soc>_<storage>_<date>.sdcard
-d enable double slot copy. Default is single slot copy.
-e enable emmc. Default is sd.
-b soc name. Currently, imx8mm and imx6ull are supported.
-m Only regenerate or overwrite MBR. This option can be used to generate MBR individually.
   Suppose that we don't need to generate MBR every time. Normally we only need to generate it once.
-h print this help.
```

- `common`:
  Currently, this directory holds some MBR files.
- `swu_dualslot_7.5G.pt`:
  A predefined MBR file for a 7.5 GB dual-slot image. As the MBR or GPT is not changed frequently, a predefined one is created. The `-m` option in `assemble_base_image` can help to create an MBR or GPT.
- `slota`:
  This link file must be linked to the images in the first slot. For example,

```
nxa08304@lsv11258:~/data/SWUpdate/swupdate_scripts/base_image_assembling$ ln -s  /home/nxa08304/data/SWUpdate/fsl-release-bsp-imx8mm-5.4.70/build-xwayland-swupdate-imx6ullevk/tmp/deploy/images/imx6ull14x14evk ./slota
nxa08304@lsv11258:~/data/SWUpdate/swupdate_scripts/base_image_assembling$ ls -l ./slota
lrwxrwxrwx 1 nxa08304 nxp 127 Aug 14 17:06 ./slota -> /home/nxa08304/data/SWUpdate/fsl-release-bsp-imx8mm-5.4.70/build-xwayland-swupdate-imx6ullevk/tmp/deploy/images/imx6ull14x14evk
```

- `slotb`:
  This link file must be linked to the images in the second slot. For example,

```
nxa08304@lsv11258:~/data/SWUpdate/swupdate_scripts/base_image_assembling$ rm ./slotb
nxa08304@lsv11258:~/data/SWUpdate/swupdate_scripts/base_image_assembling$ ln -s  /home/nxa08304/data/SWUpdate/fsl-release-bsp-imx8mm-5.4.70/build-xwayland-swupdate-imx6ullevk/tmp/deploy/images/imx6ull14x14evk ./slotb
nxa08304@lsv11258:~/data/SWUpdate/swupdate_scripts/base_image_assembling$ ls -l ./slotb
lrwxrwxrwx 1 nxa08304 nxp 127 Aug 14 17:07 ./slotb -> /home/nxa08304/data/SWUpdate/fsl-release-bsp-imx8mm-5.4.70/build-xwayland-swupdate-imx6ullevk/tmp/deploy/images/imx6ull14x14evk
```

- `workspace`:
  Contains the files that can be used to create the base image manually. For details, see the *readme.txt* in the folder and SWUpdate OTA i.MX8MM EVK/i.MX8QXP MEK.
- `boards`:
  Holds configuration files for each board.
- `cfg_imx6ull_base.cfg` and `cfg_imx6ull_update_image.cfg`:
  Configurations to build an i.MX 6ULL base image. For details, see the comments in this file.
- `cfg_imx8mm_base.cfg` and `cfg_imx8mm_update_image.cfg`:
  Configurations to build an i.MX 8M Mini base image. For details, see the comments in this file.
- `cfg_imx93_base.cfg` and `cfg_imx93_update_image.cfg`:
  Configurations to build an i.MX 93 base image. For details, see the comments in this file.
- `sw-description-*.template`:
  The template file is used to help users to generate the `sw-description` file automatically. For details about the `sw-description` file, see SWUpdate: syntax and tags with the default parser.
- `update_image_build`:
  This directory holds scripts to generate update images.
- `emmc_bootpart.sh`:
  This script is packed with the update image as post-installed scripts. It enables MMC boot partition.
- `priv.pem`:
  Private key. The password of this key is **test**. For details about `priv.pem`. See Security.
  ***Note:*** *The public key is in `rootfs /etc/ swu_public.pem`. It is built into rootfs during Yocto building.*
- `Readme.txt`:
  This readme file provides guidance to generate the update image manually.
- `slot_update`:
  Link file. The user should link this file to the build directory of the update image. For example,

```
nxa08304@lsv11258:~/data/SWUpdate/swupdate_scripts/update_image_build$ rm ./slot_update
nxa08304@lsv11258:~/data/SWUpdate/swupdate_scripts/update_image_build$ ln -s /home/nxa08304/data/SWUpdate/fsl-release-bsp-imx8mm-5.10.9/build-xwayland-swupdate-imx6ullevk/tmp/deploy/images/imx6ull14x14evk ./
slot_update
nxa08304@lsv11258:~/data/SWUpdate/swupdate_scripts/update_image_build$ ls -l ./slot_update
lrwxrwxrwx 1 nxa08304 nxp 127 Aug 14 17:14 ./slot_update -> /home/nxa08304/data/SWUpdate/fsl-release-bsp-imx8mm-5.10.9/build-xwayland-swupdate-imx6ullevk/tmp/deploy/images/imx6ull14x14evk
nxa08304@lsv11258:~/data/SWUpdate/swupdate_scripts/update_image_build$
```

- `swu_generate_part_img.sh`:
  Script to generate a partition mirror image. See details with `-h`.
- `swu_update_image_build.sh`:
  Script to generate an update package that updates images to the card. See `-h` for help:

```
./swu_update_image_build.sh - generate update image
-o specify output image name. Current default is <SOC_NAME>_<CONTAINER_VER>_<slot>_<BSP_VER>_<COPY_MODE>_<STORAGE_DEVICE>_<date>.
-d enable double slot copy. default is single slot copy.
-e enable emmc. default is sd.
-s Specify public key file for sign image generation.
-b soc name. Currently, imx8mm and imx6ull are supported.
-g Compress image with gzip. Note that compressed package need to be decompressed in RAM. Make sure ram is enough to hold the image.
-h print this help.
```

## 6.2 Creating a base image

The base image is a whole image of the storage device that is used as the initial image (*.sdcard* or *.wic* image), which can be downloaded to the board using UUU. It is in a single-copy or second-copy layout.

### 6.2.1 Base image layout

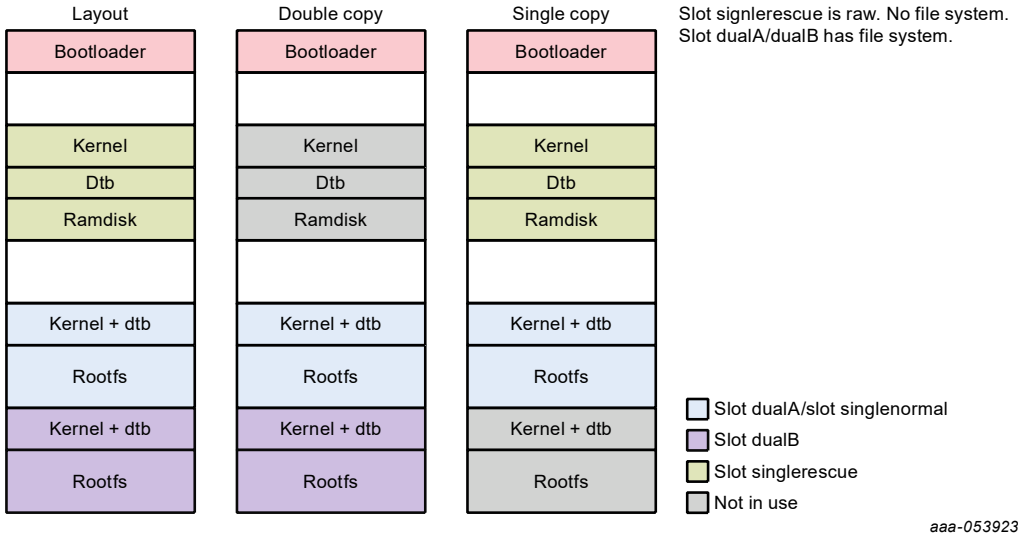



**Figure 8. Base image layout**

Double-copy layout does not have *ramdisk* and single-copy layout does not have the second copy of *kernel*, *dtb*, and *rootfs*.

### 6.2.2 Creating a base image with scripts

The scripts in `swupdate-scripts/base_image_assembling` helps users to create a base image automatically, including:

- Get artifacts for the base image automatically according to the configuration file.
- Help create an MBR or GPT for the image.
- Assemble artifacts into a `.sdcard` mirror.

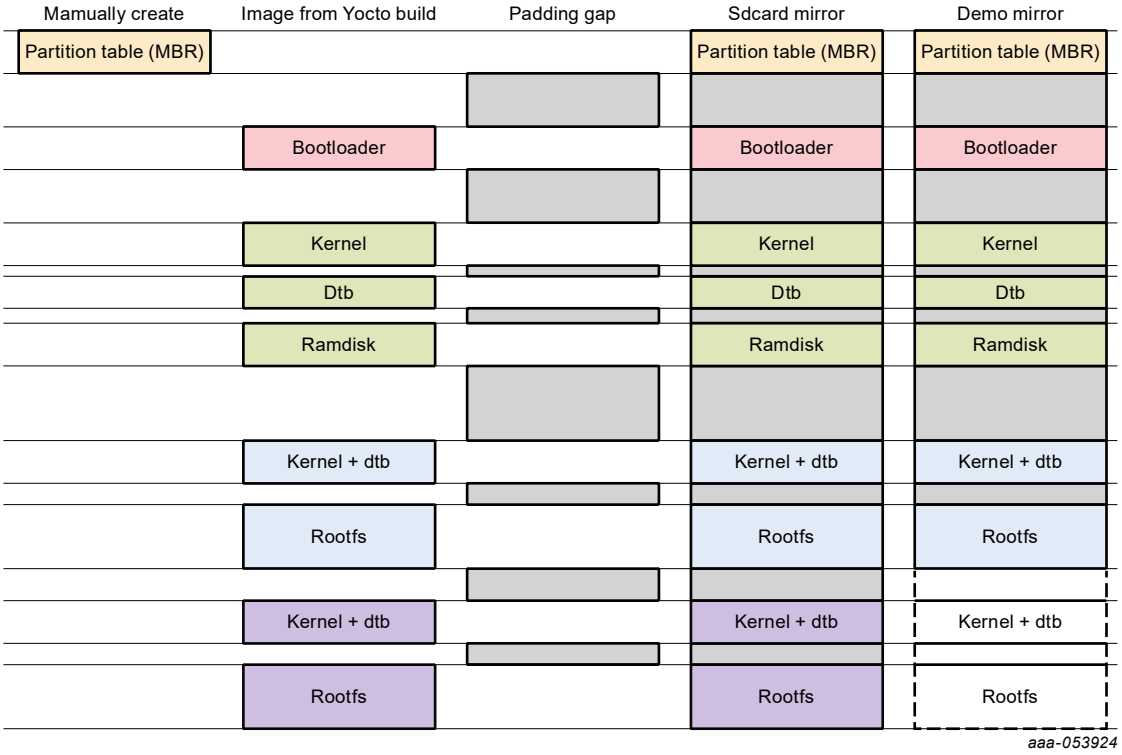



**Figure 9. Creating the base image with scripts**

To create the base image, perform the following steps:

1. Enter *swupdate-scripts*/*base_image_assembling*.
2. Link *slota* to Yocto built images.

```
nxa08304@lsv11258:~/data/SWUpdate/swupdate_scripts/base_image_assembling$ ln -s  /home/nxa08304/data/SWUpdate/fsl-release-bsp-imx8mm-5.4.70/build-xwayland-swupdate-imx6ullevk/tmp/deploy/images/imx6ull14x14evk ./slota
nxa08304@lsv11258:~/data/SWUpdate/swupdate_scripts/base_image_assembling$ ls -l ./slota
lrwxrwxrwx 1 nxa08304 nxp 127 Aug 14 17:06 ./slota -> /home/nxa08304/data/SWUpdate/fsl-release-bsp-imx8mm-5.4.70/build-xwayland-swupdate-imx6ullevk/tmp/deploy/images/imx6ull14x14evk
```

3. Link *slotb* to Yocto built images.

```
nxa08304@lsv11258:~/data/SWUpdate/swupdate_scripts/base_image_assembling$ rm ./slotb
nxa08304@lsv11258:~/data/SWUpdate/swupdate_scripts/base_image_assembling$ ln -s  /home/nxa08304/data/SWUpdate/fsl-release-bsp-imx8mm-5.4.70/build-xwayland-swupdate-imx6ullevk/tmp/deploy/images/imx6ull14x14evk ./slotb
nxa08304@lsv11258:~/data/SWUpdate/swupdate_scripts/base_image_assembling$ ls -l ./slotb
lrwxrwxrwx 1 nxa08304 nxp 127 Aug 14 17:07 ./slotb -> /home/nxa08304/data/SWUpdate/fsl-release-bsp-imx8mm-5.4.70/build-xwayland-swupdate-imx6ullevk/tmp/deploy/images/imx6ull14x14evk
```

4. Edit *boards/cfg_imx6ull_base.cfg* to specify images and offsets.
5. Run *assemble_base_image.sh* to generate the base image.
   - For i.MX 6ULL, the command is as below:

     ```
     ./assemble_base_image.sh -b imx6ull
     ```

   - For i.MX 8M Mini, the command is as below:

     ```
     ./assemble_base_image.sh -b imx8mm
     ```

   - For i.MX 93, the command is as below:

     ```
     ./assemble_base_image.sh -b imx93
     ```

```
nxa08304@lsv11258:~/data/SWUpdate/swupdate_scripts/base_image_assembling$ ./assemble_base_image.sh -b imx6ull
No output image name specified! Use default name!
Output image name is: swu_singlecopy_rescue_imx6ull_sd_20220814.sdcard
>>>> Check MBR file...DONE
>>>> Check slota boot partition mirror...DONE
>>>> Check slotb boot partition mirror...DONE
>>>> Check slota link...DONE
>>>> Check slotb link...DONE
>>>> Making header...
pad_base: 0
pad_filename: /home/nxa08304/data/SWUpdate/swupdate_scripts/base_image_assembling/common/swu_dualslot_7.5G.pt
pad_size: 1K
output_pad_file: ./tmp.bin
512 need to add to pad to 1024
pad_base: 1K
pad_filename: /home/nxa08304/data/SWUpdate/swupdate_scripts/base_image_assembling/slota/u-boot-imx6ull14x14evk.imx
pad_size: 8M
output_pad_file: ./tmp.bin
```

```
pt_size_num: 3145728000
Calculated partition size: 3145728000
e2fsck 1.45.5 (07-Jan-2020)
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts
Pass 5: Checking group summary information
/home/nxa08304/data/SWUpdate/swupdate_scripts/base_image_assembling/slota/imx-image-multimedia-imx6ull14x14evk.ext4: 44281/759552 files (0.1% non-contiguous), 295038/768000 blocks
resize2fs 1.45.5 (07-Jan-2020)
The filesystem is already 768000 (4k) blocks long.  Nothing to do!

DONE
================================================================================
Create base image swu_singlecopy_rescue_imx6ull_sd_20220814.sdcard successfully
================================================================================
```

### 6.2.3 Creating a base image with other methods

You can also use a real SD card or `kpartx` to create the `.sdcard` base mirror.

In the script, `cat` is used to merge images.

## 6.3 Creating an update image

The update image is the image that is used to update the firmware. It is a collection of partition images (for `rootfs`), files (`zImage`, `dtb`, and so on), scripts, and `sw-description` file.

These artifacts are encrypted (or not), compressed (or not), and merged into one `.swu` file.

***Note:***

*From the SWUpdate document, the update image could be generated with Yocto. It needs some changes or new recipes in Yocto. For details, see meta-swupdate: building with Yocto.*

*This document does not cover this operation, but introduces an additional script to do the same thing.*

### 6.3.1 Update image format

The `.swu` image delivered by a manufacture may contain multiple images and files. In addition, it must contain an `sw-description` file with meta information about the image.


**Figure 10. Update image format**

The `sw-description` file is important for the update image, which is described in the next section.

### 6.3.2 Simple script to create a signed image

A simple script to create a signed image is as follows:

```
#!/bin/bash

MODE="RSA-PKCS-1.5"
PRODUCT_NAME="myproduct"
CONTAINER_VER="1.0"
IMAGES="rootfs kernel"
FILES="sw-description sw-description.sig $IMAGES"

#if you use RSA
if [ x"$MODE" == "xRSA-PKCS-1.5" ]; then
    openssl dgst -sha256 -sign priv.pem sw-description > sw-description.sig
elif if [ x"$MODE" == "xRSA-PSS" ]; then
    openssl dgst -sha256 -sign priv.pem -sigopt rsa_padding_mode:pss \
        -sigopt rsa_pss_saltlen:-2 sw-description > sw-description.sig
else
    openssl cms -sign -in  sw-description -out sw-description.sig -signer
 mycert.cert.pem \
        -inkey mycert.key.pem -outform DER -nosmimecap -binary
fi
for i in $FILES;do
        echo $i;done | cpio -ov -H crc >  ${PRODUCT_NAME}_${CONTAINER_VER}.swu
```

This script does sign on each artifact and merges all artifacts into a CPIO package.

The script is also described in Building a single image.

### 6.3.3 Creating an update image with scripts in the `swupdate_script` repository

The scripts in `swupdate-scripts/update_image_build` aim to help generate an update image automatically. It includes the following:

- Get update artifacts for update image according to the configuration file automatically.

- Compress all artifacts.
- Generate an `sw-description` file based on the template.
- Sign the image.
- Assemble artifacts into a CPIO package.

To help generate an update image, there are two scripts:

- `swu_generate_part_img.sh`
  This script helps generate a partition mirror image, for example, to update the whole boot partition with a new zImage and DTB files. To generate a 120M-byte boot partition mirror with `vfat` file system, perform the following steps:
  1. Copy a kernel image and DTBs to a directory, such as `boot_pt_120M`.
  2. Run the command `./swu_generate_part_img.sh -o ./boot_pt_120M.mirror -d ./boot_pt_120M -s 120M -f vfat`.

  ***Note:*** *Because the image generated by Yocto includes the rootfs, dtb, and kernel image, we can directly use these files and define both files and images in the `sw-description`. Use the files method to update the `dtb` and `kernel` image, and use the images method to update the rootfs. For details, see the sw-description-imx93-sd-dualcopy-image.template in `swupdate-scripts` repo.*
- `swu_update_image_build.sh`
  This script reads configurations from `cfg_<soc>_update_image.cfg` and generates the update image.

To create an update image, perform the following steps:

1. Prepare update artifacts.
   For partition mirror images, use `swu_generate_part_img.sh`.
2. Link `slot_update` to the prepared images. For example,

```
nxa08304@lsv11258:~/data/SWUpdate/swupdate_scripts/update_image_build$ rm ./slot_update
nxa08304@lsv11258:~/data/SWUpdate/swupdate_scripts/update_image_build$ ln -s /home/nxa08304/data/SWUpdate/fsl-release-bsp-imx8mm-5.10.9/build-xwayland-swupdate-imx6ullevk/tmp/deploy/images/imx6ull14x14evk ./
slot_update
nxa08304@lsv11258:~/data/SWUpdate/swupdate_scripts/update_image_build$ ls -l ./slot_update
lrwxrwxrwx 1 nxa08304 nxp 127 Aug 14 17:14 ./slot_update -> /home/nxa08304/data/SWUpdate/fsl-release-bsp-imx8mm-5.10.9/build-xwayland-swupdate-imx6ullevk/tmp/deploy/images/imx6ull14x14evk
nxa08304@lsv11258:~/data/SWUpdate/swupdate_scripts/update_image_build$
```

3. Change `UPDATE_IMAGES` in `swupdate-scripts/boards/cfg_imx6ull_update_image.cfg`. To update all slot B partitions, change `UPDATE_IMAGES`.

```
UPDATE_IMAGES="
slotb_boot_pt_120M.mirror
core-image-base-imx6ull14x14evk.ext4
u-boot-imx6ull14x14evk.imx
"
```

4. The `UPDATE_SCRIPTS` contains postscripts to be executed by SWUpdate when the firmware download is finished.
5. Change `swupdate_scripts/boards/sw-description-imx6ull-emmc-dualcopy-image.template`. The `sw-description-imx6ull-emmc-dualcopy-image.template` is a template file that is used to generate the `sw-description` file. Check `swupdate_scripts/boards/sw-description-imx6ull-emmc-dualcopy-image.template` for details.
   ***Note:*** *The mechanism of the template file is that the script helps replace `<filename_ sha256>` with the SHA256 string. For `sw-description`, see [SWUpdate: syntax and tags with the default parser] for a complete guide. This example only shows several typical cases.*
6. Run `swu_update_image_build.sh` with arguments. The arguments of `swu_update_image_build.sh` are as below:

```
./swu_update_wch_image_build.sh - generate update image
-o specify output image name. Current default is <SOC_NAME>_<CONTAINER_VER>_<slot>_<BSP_VER>_<COPY_MODE>_<STORAGE_DEVICE>_<date>.
-d enable double slot copy. default is single slot copy.
-e enable emmc. default is sd.
-s Specify public key file for sign image generation.
-b soc name. Currently, imx6ullwch is supported.
-g Compress image with gzip. Note that if installed-directly = true; is not specified in sw-description, compressed package need to be decompressed in RAM. Make sure ram is enough to hold the image.
-h print this help.
```

   Run `swu_update_image_build.sh` to generate an image.
   The command is as below:

```
cd
```

```
./swu_update_image_build.sh -e -s ./priv.pem -b imx6ull -g
```

```
nxa08304@lsv11258:~/data/SWUpdate/swupdate_scripts/update_image_build$ ./swu_update_wch_image_build.sh -e -s ./priv.pem -b imx6ullwch -g
>>>> Check slot_update link...DONE
>>>> Testing update images...Truncating /home/nxa08304/data/SWUpdate/swupdate_scripts/update_image_build/ddr_cfg_samsung_iot_hub.imx.mbpB to 1M
Truncating /home/nxa08304/data/SWUpdate/swupdate_scripts/update_image_build/lk.imxB to 10M
Truncating /home/nxa08304/data/SWUpdate/swupdate_scripts/update_image_build/zImageB to 20M
Truncating /home/nxa08304/data/SWUpdate/swupdate_scripts/update_image_build/imx6ull-14x14-evk-emmc.dtb.mbpB to 1M
Truncating /home/nxa08304/data/SWUpdate/swupdate_scripts/update_image_build/rootfs.ext4B to 2G
DONE
>>>> Check post scripts...DONE
>>>> Compress update images...
Compressing env_set_bootslot.sh...OK
Compressing emmc_bootpart.sh...OK
Compressing rootfs.ext4B...OK
Compressing imx6ull-14x14-evk-emmc.dtb.mbpB...OK
Compressing zImageB...OK
Compressing lk.imxB...OK
Compressing ddr_cfg_samsung_iot_hub.imx.mbpB...OK
DONE
>>>> Check sw-decription file...software desc file: /home/nxa08304/data/SWUpdate/swupdate_scripts/update_image_build/sw-description
template file: /home/nxa08304/data/SWUpdate/swupdate_scripts/update_image_build/../boards/sw-description-imx6ullwch-emmc-dualcopy-image.template
image_list: env_set_bootslot.sh emmc_bootpart.sh rootfs.ext4B imx6ull-14x14-evk-emmc.dtb.mbpB zImageB lk.imxB ddr_cfg_samsung_iot_hub.imx.mbpB
Generating software description file...
92e64f971c9bc5442d8842f4f8aea842ef9a00fb70447ef4629d5f17710d25a1 /home/nxa08304/data/SWUpdate/swupdate_scripts/update_image_build/env_set_bootslot.sh.gz
2f87ab90678827b2267b395f6f814a8bc844b68b626d1031666aaf58433c478d /home/nxa08304/data/SWUpdate/swupdate_scripts/update_image_build/emmc_bootpart.sh.gz
80dc1bcdeac62b79b6de3f5a0dadaea8f420c7cbb0505d4f8b23e84f619bb4f2 /home/nxa08304/data/SWUpdate/swupdate_scripts/update_image_build/rootfs.ext4B.gz
87b5578c42da335743a72df8fb83682626edc73f704a932c8f3591b104e07273 /home/nxa08304/data/SWUpdate/swupdate_scripts/update_image_build/imx6ull-14x14-evk-emmc.dtb.mbpB.gz
11d65c06a19291ec1b7d08936606e24cc010e2496487c301cbcce15df2a7dedd /home/nxa08304/data/SWUpdate/swupdate_scripts/update_image_build/zImageB.gz
2b6fdc544b9d5cb705110a78b4f184f4f9952aff0b0efaf8d951de891d3f3e0e /home/nxa08304/data/SWUpdate/swupdate_scripts/update_image_build/lk.imxB.gz
d86ada1d3f46fda5a187a9fb831308b9158eb5f8b706f54d3c7f168c84d2dcaf /home/nxa08304/data/SWUpdate/swupdate_scripts/update_image_build/ddr_cfg_samsung_iot_hub.imx.mbpB.gz
DONE
DONE
>>>> Check if need a sign image...YES
>>>> Generating signature...Enter pass phrase for priv.pem:
```

Enter **test** when it asks to enter the pass phrase.
The full log is as below:

```
nxa08304@lsv11258:~/data/SWUpdate/swupdate_scripts/update_image_build$ ./swu_update_wch_image_build.sh -e -s ./priv.pem -b imx6ullwch -g
>>>> Check slot_update link...DONE
>>>> Testing update images...Truncating /home/nxa08304/data/SWUpdate/swupdate_scripts/update_image_build/ddr_cfg_samsung_iot_hub.imx.mbpB to 1M
Truncating /home/nxa08304/data/SWUpdate/swupdate_scripts/update_image_build/lk.imxB to 10M
Truncating /home/nxa08304/data/SWUpdate/swupdate_scripts/update_image_build/zImageB to 20M
Truncating /home/nxa08304/data/SWUpdate/swupdate_scripts/update_image_build/imx6ull-14x14-evk-emmc.dtb.mbpB to 1M
Truncating /home/nxa08304/data/SWUpdate/swupdate_scripts/update_image_build/rootfs.ext4B to 2G
DONE
>>>> Check post scripts...DONE
>>>> Compress update images...
Compressing env_set_bootslot.sh...OK
Compressing emmc_bootpart.sh...OK
Compressing rootfs.ext4B...OK
Compressing imx6ull-14x14-evk-emmc.dtb.mbpB...OK
Compressing zImageB...OK
Compressing lk.imxB...OK
Compressing ddr_cfg_samsung_iot_hub.imx.mbpB...OK
DONE
>>>> Check sw-decription file...software desc file: /home/nxa08304/data/SWUpdate/swupdate_scripts/update_image_build/sw-description
template file: /home/nxa08304/data/SWUpdate/swupdate_scripts/update_image_build/../boards/sw-description-imx6ullwch-emmc-dualcopy-image.template
image_list: env_set_bootslot.sh emmc_bootpart.sh rootfs.ext4B imx6ull-14x14-evk-emmc.dtb.mbpB zImageB lk.imxB ddr_cfg_samsung_iot_hub.imx.mbpB
Generating software description file...
92e64f971c9bc5442d8842f4f8aea842ef9a00fb70447ef4629d5f17710d25a1 /home/nxa08304/data/SWUpdate/swupdate_scripts/update_image_build/env_set_bootslot.sh.gz
2f87ab90678827b2267b395f6f814a8bc844b68b626d1031666aaf58433c478d /home/nxa08304/data/SWUpdate/swupdate_scripts/update_image_build/emmc_bootpart.sh.gz
80dc1bcdeac62b79b6de3f5a0dadaea8f420c7cbb0505d4f8b23e84f619bb4f2 /home/nxa08304/data/SWUpdate/swupdate_scripts/update_image_build/rootfs.ext4B.gz
87b5578c42da335743a72df8fb83682626edc73f704a932c8f3591b104e07273 /home/nxa08304/data/SWUpdate/swupdate_scripts/update_image_build/imx6ull-14x14-evk-emmc.dtb.mbpB.gz
11d65c06a19291ec1b7d08936606e24cc010e2496487c301cbcce15df2a7dedd /home/nxa08304/data/SWUpdate/swupdate_scripts/update_image_build/zImageB.gz
2b6fdc544b9d5cb705110a78b4f184f4f9952aff0b0efaf8d951de891d3f3e0e /home/nxa08304/data/SWUpdate/swupdate_scripts/update_image_build/lk.imxB.gz
d86ada1d3f46fda5a187a9fb831308b9158eb5f8b706f54d3c7f168c84d2dcaf /home/nxa08304/data/SWUpdate/swupdate_scripts/update_image_build/ddr_cfg_samsung_iot_hub.imx.mbpB.gz
DONE
DONE
>>>> Check if need a sign image...YES
>>>> Generating signature...Enter pass phrase for priv.pem:
DONE
>>>> Creating CPIO package...
sw-description sw-description.sig env_set_bootslot.sh.gz emmc_bootpart.sh.gz rootfs.ext4B.gz imx6ull-14x14-evk-emmc.dtb.mbpB.gz zImageB.gz lk.imxB.gz ddr_cfg_samsung_iot_hub.imx.mbpB.gz
sw-description
sw-description.sig
env_set_bootslot.sh.gz
emmc_bootpart.sh.gz
rootfs.ext4B.gz
imx6ull-14x14-evk-emmc.dtb.mbpB.gz
zImageB.gz
lk.imxB.gz
ddr_cfg_samsung_iot_hub.imx.mbpB.gz
65065 blocks
DONE
========================================================================
Create update image imx6ullwch_1.0_LF_v5.15.32_2.0.0_singlecopy_emmc_image_20220909_sign.swu successfully
========================================================================
```

# 7 `sw-description` file

## 7.1 Introduction

In SWUpdate, `sw-description` is the central file that describes a new software release and how a release must be installed. It uses the `libconfig` syntax or JSON. It can use Lua with a custom syntax. This file describes the `.swu` image content and allows to plan various update scenarios by setting appropriate flags in each section.

The following example shows what `sw-description` is:

```
software =
{
    version = "1.0";
```

```
    description = "Firmware update example";
    hardware-compatibility: [ "1.0", "1.2", "1.3"];
    imx6ull14x14evk: {
        images: (
        {
            filename = "example-rootfs-image.ext4";
            sha256 =
 "1664b68b017ddb43d76fac60329406510118eccc2b656533b018bdc24277f7c3";
            compressed = "zlib";
            device = "/dev/mmcblk1p8";
            installed-directly = true;
            hook = "preinst"
        }
        );
        scripts: (
        {
            filename = "env_set_bootslot.sh";
            sha256 =
 "7fcd0b30da5f623d5aacd2507884d47b08d7a23f81f7cf896fa1ae006d1e3284";
            type = "postinstall";
        }
        );
    }
}
```

## 7.2 `sw-description` template in scripts

In `swupdate-scripts`, there is a `sw-description` template to generate `sw-description` automatically.

To modify the template, add or remove tags for their usage.

***Note:*** *The mechanism of the template file is that the script helps replace* `<filename_sha256>` *with the SHA256 string.*

For a complete guide about `sw-description`, see SWUpdate: syntax and tags with the default parser.

The SWUpdate document describes another method to generate the hash string automatically with Yocto.

For details, see meta-swupdate: building with Yoct.

## 7.3 Useful tags

This section describes some useful common tags that are used widely in `sw-description`.

### 7.3.1 Hardware compatibility

Hardware compatibility can avoid the risk of installing software on a wrong platform. `sw-description` must contain a list of compatible hardware revisions:

```
hardware-compatibility: [ "1.0", "1.2", "1.3"];
```

Hardware revision is saved in a file (`/etc/hwrevision` by default) in the following format:

```
board_name board_revision
```

When creating an update image, use this tag to check the image compatibility.

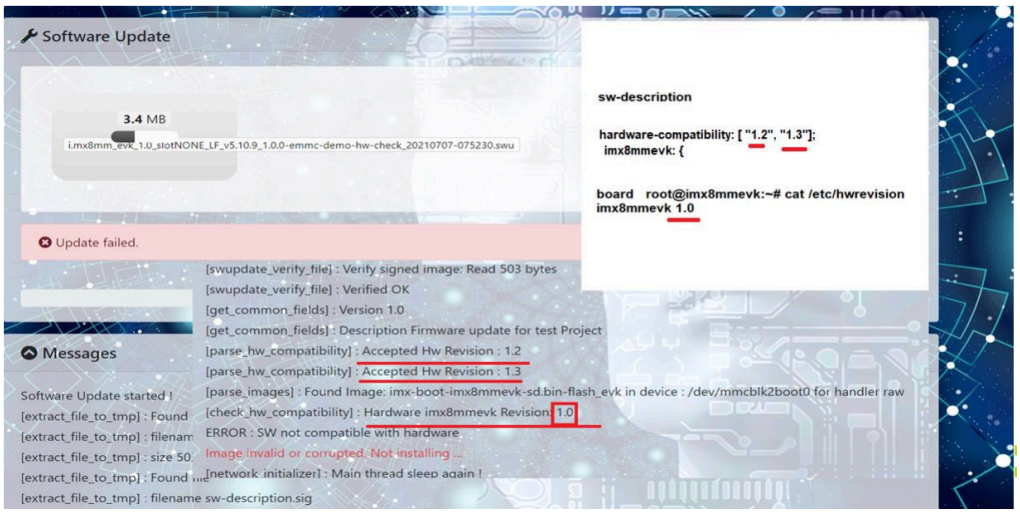



**Figure 11. Tag used to check image compatibility**

For the detailed settings, see SWUpdate: syntax and tags with the default parser.

### 7.3.2 Installed-directly

When `installed-directly` is used, SWUpdate is enabled for zero-copy (or streaming mode), which means that the incoming SWU is analyzed on the fly, and it is installed by the associated handler without any temporary copy.

If it is not set, SWUpdate creates a temporary copy in `$TMPDIR` before passing it to the handlers. `$TMPDIR` generally points to a RAMDISK and storing files there reduces the amount of memory available for the application. Disable the flag if the artifact is a single point of failure. A typical example could be the bootloader (not duplicated on the devices), and if the SWU is corrupted or the connection gets broken, the board is left in a bricked state. Download the whole artifact before installing.

Therefore, use `installed-directly` when possible.

### 7.3.3 SHA256

The SWU image is a CPIO archive with CRC (new ASCII format), but the check-in CPIO is weak. Do not trust it but enable a SHA256 hash for each artifact.

Each image inside `sw-description` must have the attribute `sha256`, with the SHA256 sum of the image. If an image does not have the SHA256 attribute, the whole compound image results to be as not verified and SWUpdate stops with an error before starting to install.

### 7.3.4 Compress images

The tag `compressed` is used for compressed images. This string is used to indicate that the `filename` is compressed and must be decompressed before being installed. The value denotes the compression type. Currently supported values are `zlib` and `zstd`.

When creating an update image, one of the concerns while using the whole `rootfs` image update approach is the size of the single update image. This tag provides handling of `gzip` compressed images in SWUpdate.

Use `compressed` for images.

***Note:***

*The tag `compressed` can only be used for images. For file compression, see Section 7.3.5. Do not use it for scripts.*

### 7.3.5 Compress files

`type = "archive"` is used for compressed files.

To compress a file, enable `CONFIG_ARCHIVE` in `defconfig`. It supports all compressed formats of `libarchive`.

### 7.3.6 Hook

`hook` is the name of the function (Lua) to be called when the entry is parsed.

### 7.3.7 Type

`type` is the string identifier for the handler, as it is set by the handler when it registers itself. For example: `ubivol`, `raw`, and `rawfile`.

### 7.3.8 Code example

The code example shows the usage of useful tags for an image in `sw-description`.

The code example is as below:

```
      {
            filename = "zImageB.gz";
            sha256 =
"3df83ed0f7a94a64429b2caf9b1d4310fa267c012ff505f17a6a395e31a51c1f";
            compressed = "zlib";
            device = "/dev/mmcblk1p6";
            installed-directly = true;
            hook = "preinst"
        },
```

## 7.4 Preventing installing an image to active partitions

One of the potential issues in `sw-description` is installing images to active partitions.

For example, the system uses SlotA rootfs, but in `sw-description`, the rootfs installation is pointed to SlotA partitions.

There are two solutions to solve this issue.

### 7.4.1 Using symbol links

Create symbol links in `rootfs` for partitions to be updated.

For example, to update the firmware for slot B, create symbol links for all slot B partitions.

```
/dev/mmcblk1ddr_update => /dev/mmcblk1p2
/dev/mmcblk1boot_update => /dev/mmcblk1p4
/dev/mmcblk1kernel_update=> /dev/mmcblk1p6
/dev/mmcblk1dtb_update=> /dev/mmcblk1p8
/dev/mmcblk1rootfs_update=> /dev/mmcblk1p10
```

Then in `sw-description`, change the device to these symbol links.

For example,

```
{
    filename=”xxx”
    ……
    device = "/dev/mmcblk1ddr_update";
    ……
},
```

These partition symbol links are created when the system boots.

### 7.4.2 Using Lua function to check boot slots

Enable a Lua interpreter in SWUpdate and write Lua scripts in `sw-description` to handle the case.

Example code:

```
function preinst()
    local cmd = `fw_printenv bootslot | cut -f 2 -d'='`
    local boot_slot = os.capture(cmd, 1)
    if (string.sub(boot_slot, 1, 5) == \"dualA\") then
        swupdate.trace(\"Slot is A. Updated allowed: go on !\")
        return true, boot_slot
    else
        swupdate.trace(\"Slot is B. Updated forbidden: STOP !\")
        return false, boot_slot
    end
end
```

In the example code, use `fw_printenv` to get an active slot `bootslot` and check if the update is allowed.

***Note:***

*More Lua script examples can be found in meta-swupdate-boards.*

### 7.4.3 Using partition labels (GPT only)

When the partition format is GPT, the partition labels can be found in */dev/disk/by-partlabel/*.

For example:

```
root@imx93evk:~# ls -l /dev/disk/by-label/
lrwxrwxrwx 1 root root 15 Feb 19 06:38 boot -> ../../mmcblk0p1
lrwxrwxrwx 1 root root 15 Feb 19 06:38 root -> ../../mmcblk0p2
```

So we can use these partition labels without knowing the exact partitions.

For example:

```
{
  filename=”xxx”
  ……
  device = "/dev/disk/by-label/rootfsB";
  ……
}
```

# 8 Debug

## 8.1 Debugging SWUpdate

To debug, use the following methods:

1. Check the **Messages** window on the webpage. When updating images from the web, the log of progress, and errors are shown in the **Messages** window.

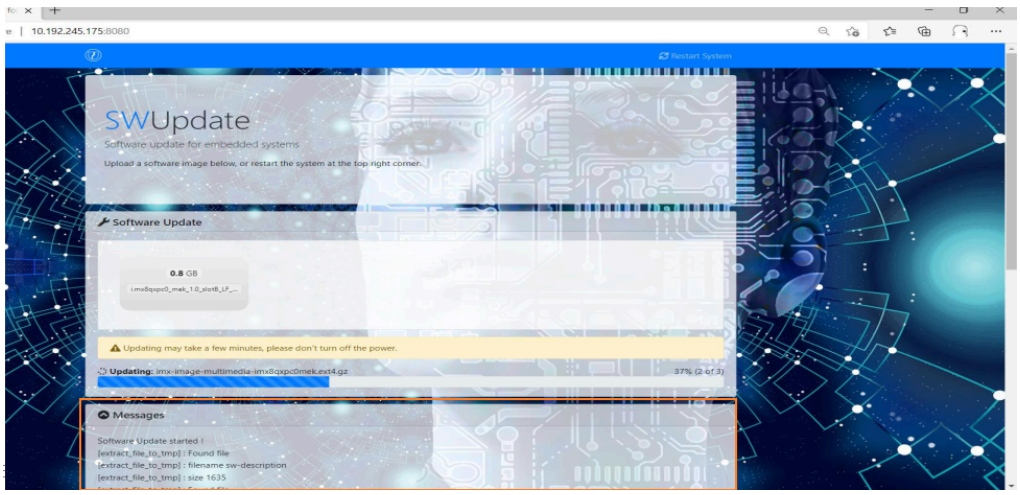



**Figure 12. Message window**

2. Try updating the `swu` image from the board and check the logs. Some time, there are not enough logs from the webpage. Run the SWUpdate application on the board manually to check the logs:
   a. Transfer the `swu` image to a board with the SCP command or other methods.
   b. Use the command `swupdate -v -i <swu_image>` to do updates manually.
   c. Check the logs.

```
root@imx6ull14x14evk:~# swupdate -v -i
 imx6ull_1.0_LF_v5.10.9_1.0.0_singlecopy_sd_simple_20220812_sign-
onlydtb.swu
Swupdate v2021.04.0
Licensed under GPLv2. See source distribution for detailed copyright
 notices.
[INFO ] : SWUPDATE running :  [main] : Running on imx6ull14x14evk Revision
 1.0
[INFO ] : SWUPDATE running :  [print_registered_handlers] : Registered
 handlers:
[INFO ] : SWUPDATE running :  [print_registered_handlers] :     dummy
[INFO ] : SWUPDATE running :  [print_registered_handlers] :     uboot
[INFO ] : SWUPDATE running :  [print_registered_handlers] :     bootloader
[INFO ] : SWUPDATE running :  [print_registered_handlers] :     raw
[INFO ] : SWUPDATE running :  [print_registered_handlers] :     rawfile
[INFO ] : SWUPDATE running :  [print_registered_handlers] :     rawcopy
… …
```

## 8.2 Debugging `sw-description`

The `sw-description` file must be written carefully. In debugging `sw-description`, we must:

1. Check the syntax.
2. Check the scope of tags.

3. Try to do the update from the command line on board and check logs. See Section 8.1.

## 8.3 Debugging Lua scripts

Generally, debugging Lua scripts is like debugging SWUpdate where the users check the logs.

However, when using embedded scripts in `sw-description`, each double quote must be escaped.

## 8.4 Possible issues

### 8.4.1 Not enough free space to exact …

If such an error:

```
[extract_files] :  filename imx-image-multimedia-imx6ull14x14evk.ext4
[extract_files] :  size 3145728000 required
ERROR : Not enough free space to extract imx-image-multimedia-imx6ull14x14evk.ext4 (needed 3145728000, got 130326528)
Image invalid or corrupted. Not installing ...
```

Check if the tag `installed-directly = true` in `sw-description` is used.

Without the tag `installed-directly = true`, the update image is downloaded to the temp directly. In the `ramfs`, it consumes DDR memory. If the update image is larger than the DDR size, this error occurs.

### 8.4.2 EXT4-fs error

During the update process, if such an error occurs in the kernel command window:

```
[ 1954.298810] EXT4-fs error (device mmcblk1p2): htree_dirblock_to_tree:1022: inode #213: block 23626: comm crond: bad entry in directory: inode out of bounds - offset=0, inode=1010713, rec_len=256, name_len=25, size=4096
[ 1954.322857] EXT4-fs error (device mmcblk1p2): htree_dirblock_to_tree:1022: inode #44255: block 266657: comm crond: bad entry in directory: rec_len is smaller than minimal - offset=0, inode=0, rec_len=0, name_len=0, size=4096
```

Maybe the user is trying to update the image to the same partition.

For example, the SWUpdate is launched from `/dev/mmcblk0p3` and the updater is programming the image to `/dev/mmcblk0p3`.

### 8.4.3 Unrecognized file system type after reboot

The error occurs when the update is done and the system reboots. U-Boot cannot load files from the partition and displays:

```
** Unrecognized filesystem type **
** Unrecognized filesystem type **
```

The reason is probably that the `images` tag is incorrectly used in `sw-description`. For file update that must be copied to a partition, use `files`. For example,

```
images: (
        {
                filename = "u-boot-imx6ull14x14evk.imx";
                sha256 = "<u-boot-imx6ull14x14evk.imx_sha256>";
                device = "/dev/mmcblk1";
                offset = "1K";
```

```
            }
        );
Here, as u-boot-imx6ull14x14evk.imx will be copied to /dev/mmcblk1, “files”
 should be used.
files: (
            {
                filename = "u-boot-imx6ull14x14evk.imx";
                sha256 = "<u-boot-imx6ull14x14evk.imx_sha256>";
                device = "/dev/mmcblk1";
                offset = "1K";
            }
        );
```

### 8.4.4 SWUpdate task not launched or timeout

#### 8.4.4.1 Phenomenon of the error

When an SWUpdate inside the system is running, you can see the SWUpdate tasks.

```
root@imx6ull14x14evk:~# ps | grep swupdate
  202 root      6988 S    /usr/bin/swupdate-sysrestart -w
  203 root     52268 S    /usr/bin/swupdate -v -w -r /www -p 8080
  204 root      1640 S    /usr/bin/swupdate-progress -r -w
  217 root     52268 S    /usr/bin/swupdate -v -w -r /www -p 8080
  251 root      2348 S    grep swupdate
```

With default `meta-swupdate`, some or all SWUpdate daemon tasks may be missing. In addition, some errors and daemon tasks may not run properly.

```
root@imx6ull14x14evk:/usr/local/bin# systemctl status swupdate-sysrestart
× swupdate-sysrestart.service - swupdate-sysrestart daemon
     Loaded: loaded (/lib/systemd/system/swupdate-sysrestart.service; enabled; vendor preset: enabled)
     Active: failed (Result: exit-code) since Fri 2022-03-11 20:26:50 UTC; 27min ago
       Docs: https://github.com/sbabic/swupdate
             https://sbabic.github.io/swupdate
    Process: 271 ExecStart=/usr/bin/swupdate-sysrestart -w (code=exited, status=203/EXEC)
   Main PID: 271 (code=exited, status=203/EXEC)

Mar 11 20:26:48 imx6ull14x14evk systemd[1]: Started swupdate-sysrestart daemon.
Mar 11 20:26:48 imx6ull14x14evk systemd[271]: swupdate-sysrestart.service: Failed to locate executable /usr/bin/swupdate-sysrestart: No such file or directo>
Mar 11 20:26:48 imx6ull14x14evk systemd[271]: swupdate-sysrestart.service: Failed at step EXEC spawning /usr/bin/swupdate-sysrestart: No such file or direct>
Mar 11 20:26:50 imx6ull14x14evk systemd[1]: swupdate-sysrestart.service: Main process exited, code=exited, status=203/EXEC
Mar 11 20:26:50 imx6ull14x14evk systemd[1]: swupdate-sysrestart.service: Failed with result 'exit-code'.
lines 1-13/13 (END)
```

Restarting the SWUpdate service may not solve the issue.

```
root@imx6ull14x14evk:~# systemctl restart swupdate
Job for swupdate.service failed because a timeout was exceeded.
See "systemctl status swupdate.service" and "journalctl -xeu swupdate.service" for details.
root@imx6ull14x14evk:~#
```

The detailed log of this issue is as follows:

```
× swupdate.service - SWUpdate daemon
     Loaded: loaded (/lib/systemd/system/swupdate.service; enabled; vendor
 preset: enabled)
     Active: failed (Result: timeout) since Fri 2021-11-19 17:21:05 UTC; 9min
 ago
TriggeredBy: ● swupdate.socket
       Docs: https://github.com/sbabic/swupdate
             https://sbabic.github.io/swupdate
    Process: 357 ExecStart=/usr/lib/swupdate/swupdate.sh (code=exited, status=0/
SUCCESS)
   Main PID: 357 (code=exited, status=0/SUCCESS)
```

```
Nov 19 17:19:35 blueye swupdate.sh[357]: [TRACE] : SWUPDATE running :
 [network_initializer] : Main loop daemon
Nov 19 17:19:35 blueye swupdate.sh[357]: [TRACE] : SWUPDATE running :
 [listener_create] : creating socket at /tmp/swupdateprog
Nov 19 17:19:35 blueye swupdate.sh[357]: [TRACE] : SWUPDATE running :
 [listener_create] : creating socket at /tmp/sockinstctrl
Nov 19 17:19:35 blueye swupdate.sh[357]: [TRACE] : SWUPDATE running :
 [start_swupdate_subprocess] : Started webserver with pid 386 and fd 10
Nov 19 17:19:35 blueye swupdate.sh[357]: [INFO ] : SWUPDATE running :
 [start_mongoose] : Mongoose web server version 6.18 with pid 386 started on
 port(s) 8080 with web root [/www]
Nov 19 17:21:05 blueye systemd[1]: swupdate.service: start operation timed out.
 Terminating.
Nov 19 17:21:05 blueye systemd[1]: swupdate.service: Killing process 386
 (swupdate) with signal SIGKILL.
Nov 19 17:21:05 blueye systemd[1]: swupdate.service: Failed with result
 'timeout'.
Nov 19 17:21:05 blueye systemd[1]: swupdate.service: Unit process 386 (swupdate)
 remains running after unit stopped.
Nov 19 17:21:05 blueye systemd[1]: Failed to start SWUpdate daemon.
```

#### 8.4.4.2 Possible reason

After SWUpdate 2022.05, as SWUpdate supports the `notify` type, the service type of the SWUpdate system service is set to `notify`.

With this change, activate the `CONFIG_SYSTEMD`, but this was not done in `config` or in `defconfig` in `meta-swupdate`.

There are two ways to solve this issue:

- Enable the `CONFIG_SYSTEMD` flag and add `systemd` to the recipe `DEPENDS`.
- Change the service type of SWUpdate from `notify` to `exec`.

The example is as below:

```
diff --git a/recipes-support/swupdate/swupdate/swupdate.service b/recipes-
support/swupdate/swupdate/swupdate.service
index 7f8e966..c0253aa 100644
--- a/recipes-support/swupdate/swupdate/swupdate.service
+++ b/recipes-support/swupdate/swupdate/swupdate.service
@@ -4,7 +4,7 @@ Documentation=https://github.com/sbabic/swupdate
 Documentation=https://sbabic.github.io/swupdate

 [Service]
-Type=notify
+Type=exec
 ExecStart=@LIBDIR@/swupdate/swupdate.sh
 KillMode=mixed
```

### 8.4.5 Preinstall script does not work

The `preinstall` script does not work with the `direct-install` option.

A typical example of `direct-install` is as follows:

```
scripts: (
            {
```

```
                filename = "validate_boot_slot.sh";
                sha256 = "<validate_boot_slot.sh_sha256>";
                type = "preinstall";
            },
            {
                filename = "env_set_bootslot.sh";
                sha256 = "<env_set_bootslot.sh_sha256>";
                type = "postinstall";
            }
        );
```

When the `preinstall` script is used with `direct-install`, the `preinstall` script is not executed. The script of `postinstall` is not affected.

For details, see Preinstall script won't run before INSTALLING RAW IMAGE.

### 8.4.6 No space left on device

The error log of **No space left on device** is as follows:

```
Mar 11 14:03:28 imx6ull14x14evk swupdate.sh[286]: [TRACE] : SWUPDATE running :  [check_hw_compatibility] : Hardware imx6ull14x14evk Revision: 1.0
Mar 11 14:03:28 imx6ull14x14evk swupdate.sh[286]: [TRACE] : SWUPDATE running :  [check_hw_compatibility] : Hardware compatibility verified
Mar 11 14:03:28 imx6ull14x14evk swupdate.sh[286]: [TRACE] : SWUPDATE running :  [extract_files] : Found file
Mar 11 14:03:28 imx6ull14x14evk swupdate.sh[286]: [TRACE] : SWUPDATE running :  [extract_files] :       filename env_set_bootslot.sh.gz
Mar 11 14:03:28 imx6ull14x14evk swupdate.sh[286]: [TRACE] : SWUPDATE running :  [extract_files] :       size 188 required
Mar 11 14:03:28 imx6ull14x14evk swupdate.sh[286]: [TRACE] : SWUPDATE running :  [extract_files] : Found file
Mar 11 14:03:28 imx6ull14x14evk swupdate.sh[286]: [TRACE] : SWUPDATE running :  [extract_files] :       filename rootfs.ext4B.gz
Mar 11 14:03:28 imx6ull14x14evk swupdate.sh[286]: [TRACE] : SWUPDATE running :  [extract_files] :       size 56388786 required
Mar 11 14:03:28 imx6ull14x14evk swupdate.sh[286]: [TRACE] : SWUPDATE running :  [extract_files] : Installing STREAM rootfs.ext4B.gz, 56388786 bytes
Mar 11 14:03:28 imx6ull14x14evk swupdate.sh[286]: [TRACE] : SWUPDATE running :  [install_single_image] : Found installer for stream rootfs.ext4B.gz raw
Mar 11 14:07:04 imx6ull14x14evk swupdate.sh[286]: [ERROR] : SWUPDATE failed [0] ERROR : cannot write 3310 bytes: No space left on device
Mar 11 14:07:04 imx6ull14x14evk swupdate.sh[286]: [TRACE] : SWUPDATE running :  [install_single_image] : Installer for raw not successful !
Mar 11 14:07:04 imx6ull14x14evk swupdate.sh[286]: [ERROR] : SWUPDATE failed [0] ERROR : Error streaming rootfs.ext4B.gz
Mar 11 14:07:04 imx6ull14x14evk swupdate.sh[286]: [ERROR] : SWUPDATE failed [1] Image invalid or corrupted. Not installing ...
Mar 11 14:07:04 imx6ull14x14evk swupdate.sh[286]: [TRACE] : SWUPDATE running :  [network_initializer] : Main thread sleep again !
Mar 11 14:07:04 imx6ull14x14evk swupdate.sh[286]: [INFO ] : No SWUPDATE running :  Waiting for requests...
```

The cause of this error is that the partition image size is not equal to the partition size. For example, when the `rootfs.ext4` is intended to overwrite `/dev/mmcblk0p2`, the size of `rootfs.ext4` must be equal to the size of the partition `/dev/mmcblk0p2`.

There are two possible reasons:

- There is an error in `sw-description` that an incorrect image mirror is used for a partition. Therefore, the size of the image mirror is not equal to the target partition size.
- Size mismatches when creating partitions. For example, the user uses kB to do partition on the eMMC.

```
FBK: ucmd mmc=`cat /tmp/mmcdev`; sgdisk -a 8 -n 1:128:+1024K -t 1:8300 -c
 1:"ddr.cfgA" /dev/mmcblk${mmc}
FBK: ucmd mmc=`cat /tmp/mmcdev`; sgdisk -a 8 -n 2:0:+1024K -t 2:8300 -c
 2:"ddr.cfgB" /dev/mmcblk${mmc}
FBK: ucmd mmc=`cat /tmp/mmcdev`; sgdisk -a 8 -n 3:0:+10240K -t 3:8300 -c
 3:"boot.binA" /dev/mmcblk${mmc}
FBK: ucmd mmc=`cat /tmp/mmcdev`; sgdisk -a 8 -n 4:0:+10240K -t 4:8300 -c
 4:"boot.binB" /dev/mmcblk${mmc}
FBK: ucmd mmc=`cat /tmp/mmcdev`; sgdisk -a 8 -n 5:0:+20480K -t 5:8300 -c
 5:"kernel.ImageA" /dev/mmcblk${mmc}
FBK: ucmd mmc=`cat /tmp/mmcdev`; sgdisk -a 8 -n 6:0:+20480K -t 6:8300 -c
 6:"kernel.ImageB" /dev/mmcblk${mmc}
FBK: ucmd mmc=`cat /tmp/mmcdev`; sgdisk -a 8 -n 7:0:+1024K -t 7:8300 -c
 7:"kernel_dtb.binA" /dev/mmcblk${mmc}
FBK: ucmd mmc=`cat /tmp/mmcdev`; sgdisk -a 8 -n 8:0:+1024K -t 8:8300 -c
 8:"kernel_dtb.binB" /dev/mmcblk${mmc}
```

```
FBK: ucmd mmc=`cat /tmp/mmcdev`; sgdisk -a 8 -n 9:0:+2048000K -t 9:8300 -c
 9:"rootfs.imgA" /dev/mmcblk${mmc}
FBK: ucmd mmc=`cat /tmp/mmcdev`; sgdisk -a 8 -n 10:0:+2048000K -t 10:8300 -c
 10:"rootfs.imgB" /dev/mmcblk${mmc}
```

However, in the `cfg` file in `swupdate-scripts`, MB and GB are used to create the update image.

```
UPDATE_IMAGES="
ddr_cfg_samsung_iot_hub.imx.mbpB:1M
lk.imxB:10M
zImageB:20M
imx6ull-14x14-evk-emmc.dtb.mbpB:1M
rootfs.ext4B:2G
"
```

Therefore, in this way, some bytes missing may cause the issue.

### 8.4.7 Partition size not enough

Error message:

```
e2fsck 1.45.5 (07-Jan-2020)
```

The filesystem size (according to the superblock) is 887599 blocks.

The physical size of the device is 768000 blocks.

Either the superblock or the partition table is likely to be corrupt!

```
Abort<y>?
```

This error indicates that you must enlarge size of partition when creating base image. If swupdate scripts are used, the partition table in *cfg_<soc>_base.cfg* must be changed.

### 8.4.8 ext4 rootfs has unsupported feature(s): FEATURE_C12

Error message:

```
e2fsck 1.46.5 (30-Dec-2021)
swupdate-scripts/base_image_assembling/slota/core-image-base-imx93-11x11-
lpddr4x-
                        evk.ext4 has unsupported feature(s): FEATURE_C12
e2fsck: Get a newer version of e2fsck!

swupdate-scripts/base_image_assembling/slota/core-image-base-imx93-11x11-
lpddr4x-
                           evk.ext4: ********** WARNING: Filesystem still has
 errors **********

resize2fs 1.46.5 (30-Dec-2021)
resize2fs: Filesystem has unsupported feature(s)
```

Solution:

Build and install `e2fsprogs` from source.

```
wget https://mirrors.edge.kernel.org/pub/linux/kernel/people/tytso/e2fsprogs/
v1.47.0/e2fsprogs-1.47.0.tar.xz
tar -xf e2fsprogs-1.47.0.tar.xz
```

```
cd e2fsprogs-1.47.0/
./configure
make -j16
sudo make install
```

### 8.4.9 mtools missing

Error message:

```
/home/nxf65025/imx-yocto-bsp/swupdate-scripts/base_image_assembling/../utils/
utils.sh: line 58: mdir: command not found
/home/nxf65025/imx-yocto-bsp/swupdate-scripts/base_image_assembling/../utils/
utils.sh: line 66: mcopy: command not found
/home/nxf65025/imx-yocto-bsp/swupdate-scripts/base_image_assembling/../utils/
utils.sh: line 66: mcopy: command not found
/home/nxf65025/imx-yocto-bsp/swupdate-scripts/base_image_assembling/../utils/
utils.sh: line 68: mdir: command not found
```

Solution:

Install `mtools`.

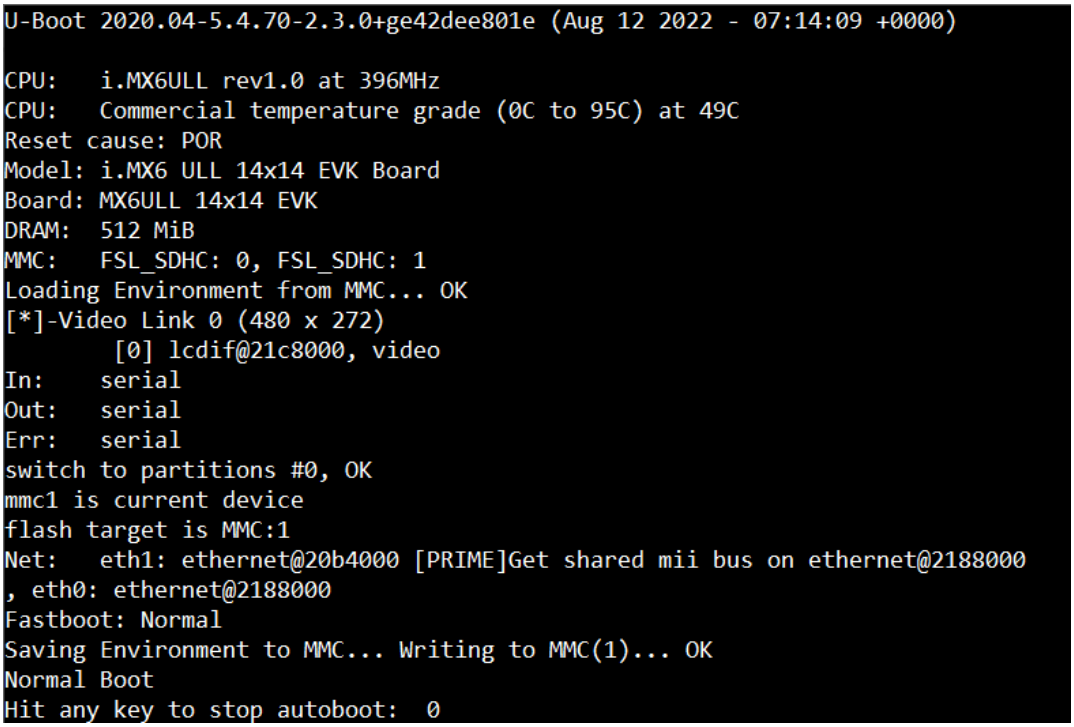
```
sudo apt-get install mtools
```

# 9 Test procedure

This section describes the test procedure for upgrading from 5.4.70 to 5.10.9.

## 9.1 Booting the board with the base image

1. Program the base image to the SD/eMMC card and boot.



2. Enter the kernel and check the kernel version.

```
root@imx6ull14x14evk:~# uname -a
Linux imx6ull14x14evk 5.4.70-2.3.0+g4f2631b022d8 #1 SMP PREEMPT Wed Aug 10 09:04
```

3. Check the IP address.

```
root@imx6ull14x14evk:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:04:9f:04:9b:2a
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

eth1      Link encap:Ethernet  HWaddr 00:04:9f:04:9b:29
          inet addr:10.193.102.33  Bcast:10.193.102.255  Mask:255.255.255.0
          inet6 addr: fe80::204:9fff:fe04:9b29/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:22531 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1629 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2526683 (2.4 MiB)  TX bytes:1528531 (1.4 MiB)
```

4. Launch the browser and enter http://10.193.102.33:8080/.

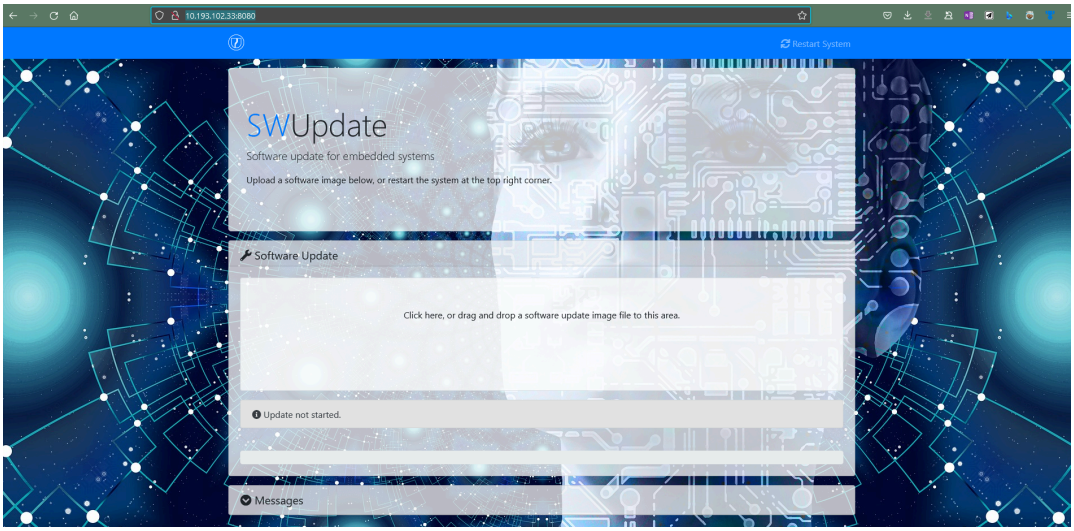



**Figure 13. SWUpdate webpage**

## 9.2 Updating the update image

1. Taking a single-copy as an example, update the whole partition under SWUpdate `ramfs`. Set `env` to `u-boot` to enter `ramfs`.

```
fw_setenv bootslot singlerescue -f /etc/u-boot-imx-initial-env
fw_setenv upgrade_available 1
```

***Note:*** *`-f /etc/u-boot-imx-initial-env` is only used when `env` is not saved in U-Boot.*

```
root@imx6ull14x14evk:~# fw_setenv bootslot singlerescue -f /etc/u-boot-imx-initi
al-env
Cannot read environment, using default
root@imx6ull14x14evk:~# fw_setenv upgrade_available 1
root@imx6ull14x14evk:~# reboot
root@imx6ull14x14evk:~#          Stopping Session c1 of user root.
```

2. In U-Boot, the logs of `ramfs` are as follows.

```
Hit any key to stop autoboot:  0
swuboot ramdisk

MMC read: dev # 1, block # 16384, count 61440 ... 61440 blocks read: OK

MMC read: dev # 1, block # 77824, count 512 ... 512 blocks read: OK

MMC read: dev # 1, block # 86016, count 86016 ... 86016 blocks read: OK
Kernel image @ 0x80800000 [ 0x000000 - 0x899a18 ]
## Loading init Ramdisk from Legacy Image at 86800000 ...
```

3. Check the IP address in `ramfs`.

```
root@imx6ull14x14evk:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:04:9f:04:9b:2a
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

eth1      Link encap:Ethernet  HWaddr 00:04:9f:04:9b:29
          inet addr:10.193.102.33  Bcast:10.193.102.255  Mask:255.255.255.0
          inet6 addr: fe80::204:9fff:fe04:9b29/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:22531 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1629 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2526683 (2.4 MiB)  TX bytes:1528531 (1.4 MiB)
```

4. Launch IE and enter http://10.193.102.33:8080/.

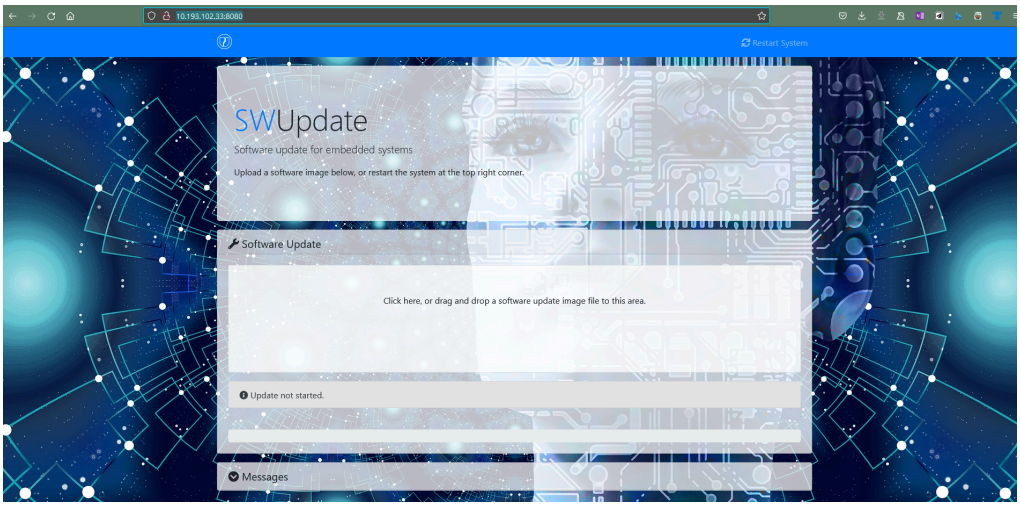



**Figure 14. SWUpdate webpage**

5. In the webpage, click and select the update image.

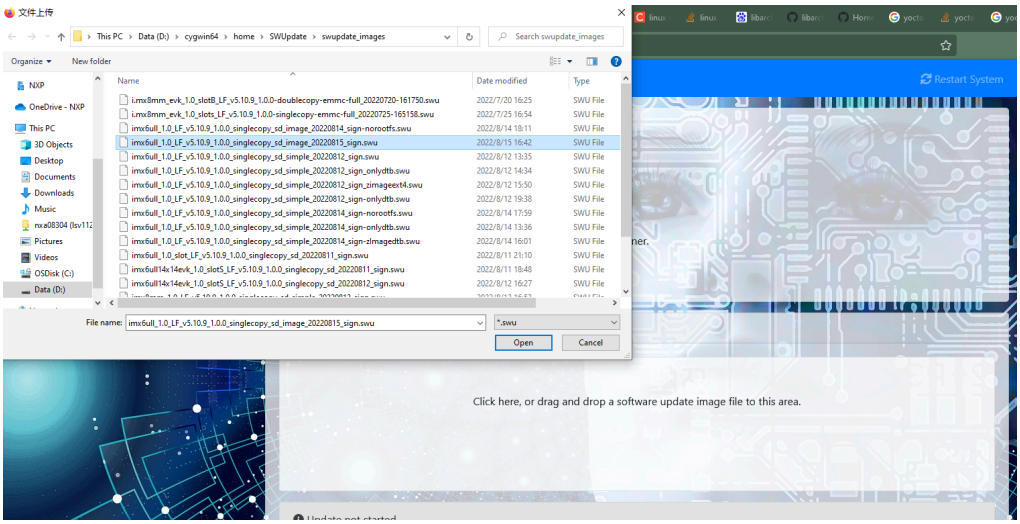



**Figure 15. Select the upate image**

6. Wait for the update to finish. The board reboots automatically.

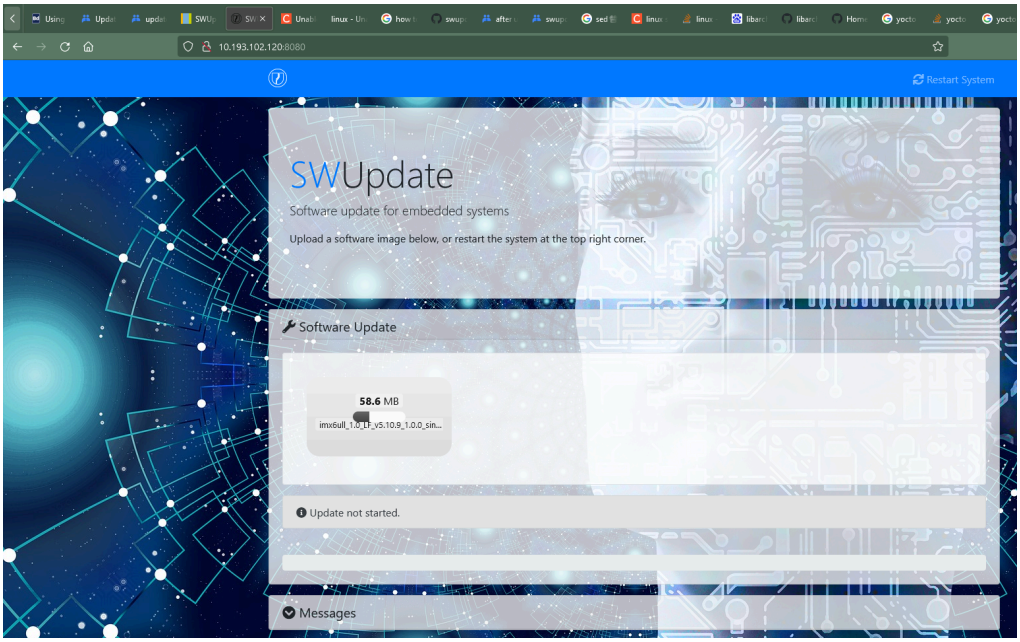
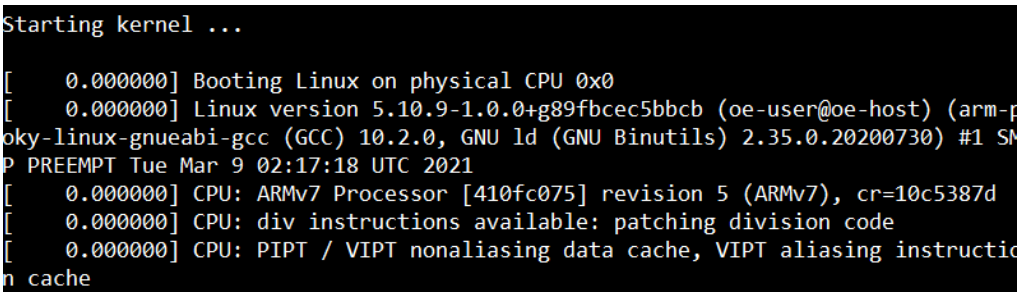



**Figure 16. Wait for update to finish**

7. When rebooting, check U-Boot and the kernel version.

```
U-Boot 2020.04-5.10.9-1.0.0+gad7b74b415 (Mar 05 2021 - 07:05:56 +0000)

CPU:   i.MX6ULL rev1.0 at 396MHz
CPU:   Commercial temperature grade (0C to 95C) at 45C
Reset cause: POR
Model: i.MX6 ULL 14x14 EVK Board
Board: MX6ULL 14x14 EVK
DRAM:  512 MiB
MMC:   FSL_SDHC: 0, FSL_SDHC: 1
Loading Environment from MMC... OK
[*]-Video Link 0 (480 x 272)
```




### 9.3 Checking `sysinfo.cgi` for update status

Take Mongoose daemon mode, single-copy as an example.

Before the upgrade:

http://10.192.245.175:8080/sysinfo.cgi

```
Linux imx6ull14x14evk 5.4.70-2.3.0+g4f2631b022d8 #1 SMP PREEMPT Sun Jun 27
 18:06:47 UTC 2021 aarch64 aarch64 aarch64 GNU/Linux
console=ttymxc1,115200 root=/dev/mmcblk2p2 rootwait rw cur_slot=singlenormal U-
Boot_ver=2020.04-5.4.70-2.3.0+ge42dee801e(Jun 27 2021-18:23:27)
```

During the upgrade:

http://10.192.245.175:8080/sysinfo.cgi

```
Linux imx6ull14x14evk 5.4.70-2.3.0+g4f2631b022d8 #1 SMP PREEMPT Sun Jun 27
 18:06:47 UTC 2021 aarch64 aarch64 aarch64 GNU/Linux
console=ttymxc1,115200 root=/dev/mmcblk2p2 rootwait rw cur_slot=singlerescue U-
Boot_ver=2020.04-5.4.70-2.3.0+ge42dee801e(Jun 27 2021-18:23:27)
```

After the upgrade:

http://10.192.245.175:8080/sysinfo.cgi

```
Linux imx6ull14x14evk 5.10.9-1.0.0+g32513c25d8c7 #1 SMP PREEMPT Tue Mar 9
 02:17:18 UTC 2021 aarch64 aarch64 aarch64 GNU/Linux
console=ttymxc1,115200 root=/dev/mmcblk2p2 rootwait rw cur_
 cur_slot=singlenormal U-Boot_ver=2020.04-5.10.9-1.0.0+gad7b74b415(Mar 05
 2021-07:05:56)
```

## 10 Using SWUpdate on other i.MX platforms

SWUpdate is supported on i.MX 6ULL, i.MX 8M Mini, i.MX 8QuadXPlus, and i.MX 93 platforms.

For other platforms, users could add the support by themselves.

Besides configuration changes inside SWUpdate, two changes must be made in `meta-swupdate-imx` to support SWUpdate for a new i.MX platform.

To support a new platform, most changes can be made in `recipes-bsp/u-boot/u-boot-imx_%.bbappend` in `meta-swupdate-imx`.

## 10.1 Creating `fw_env.config` in `u-boot-imx_%.bbappend`

Take the i.MX 8M Mini as an example in `recipes-bsp/u-boot/u-boot-imx_%.bbappend`.

```
do_install:append:mx8mm-nxp-bsp () {

    echo "/dev/mmcblk1 0x400000 0x2000" > ${D}/${sysconfdir}/fw_env.config
    echo "/dev/mmcblk1 0x402000 0x2000" >> ${D}/${sysconfdir}/fw_env.config
    echo "${MACHINE} ${SWU_HW_REV}" > ${D}/${sysconfdir}/hwrevision

}
```

For a new platform, add a similar `do_install:append:mx{XXX}-nxp-bsp` function to create `/etc/fw_env.config` and `/etc/hwrevision` in `rootfs`.

The `fw_env.config` file is used by `fw_setenv` to know the location of U-Boot environment data and size. This offset `0x400000` and `0x402000` are values of `CONFIG_ENV_OFFSET` and `CONFIG_ENV_OFFSET_REDUND`, which can be found in `<u-boot>/configs/<soc>_defconfig`.

The `hwrevision` file contains hardware information, which can be checked after upgrade to know the new system information.

***Note:***

*For different Yocto versions, the syntax of the function is different.*

*For example, for Gatesgarth (5.4.70 kernel), the function is defined as `do_install_append_mx8mm()`. But for Kirkstone (5.15.32 kernel) and Langdale (5.15.71 kernel), the function is defined as `do_install:append:mx8mm-nxp-bsp()`.*

## 10.2 Creating U-Boot patches to enable `REDUNDANT_ENV` and SWU bootargs

Environment data of U-Boot is important in SWUpdate.

SWUpdate has bindings to various bootloaders to store persistent state information across reboots.

In SWUpdate, the bootloader is used to check the update result, and switch the boot system according to the update result. It means that some state information must be passed across the bootloader and the update agent. Usually, this is done through persistent variables that are available to both SWUpdate and the bootloader.

As for Linux operating system, U-Boot is used to start the kernel. Therefore, environment variables are used in U-Boot to coordinate these states.

Therefore, the U-Boot patches are intended to:

- Enable a redundant `env` feature as the `env` data is important for SWUpdate. `CONFIG_SYS_REDUNDAND_ENVIRONMENT` grants U-Boot to have two copies of environment data, which can back up each other.
- Enable U-Boot bootcount feature. `u-boot CONFIG_BOOTCOUNT_BOOTLIMIT` is used to trigger the U-Boot to execute the `altbootcmd`. In this demo, it is `bootlimit=3`. When `upgrade_available = 1`, U-Boot increases the bootcount.

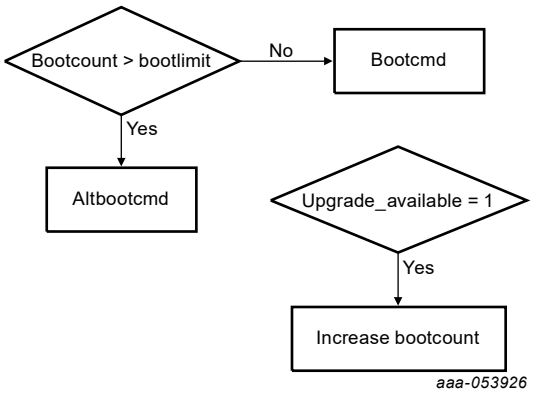



**Figure 17. Enable U-Boot bootcount feature**

- Add macros to switch the boot system according to the update result.
  See the U-Boot patches under `meta-swupdate-imx/recipes-bsp/u-boot/files/` and add similar patches for another platforms.
  Table 7 lists the descriptions of macros in the U-Boot environment.

**Table 7. Macros in U-Boot environment**

| Macros | Description |
|---|---|
| `bootslot` | Current `bootslot` to start. The value can be:<br>– `dualA`: Start Slot A in double copy.<br>– `dualB`: Start Slot B in double copy.<br>– `singlenormal`: Start SWUpdate inside ramdisk in a single copy.<br>– `singlerescue`: Start rescue system in single copy. |
| `adjustbootsource` | Adjust boot source according to `bootslot`. |
| `altbootusb` | Optional and currently not used. Alterative boot from USB. |
| `altbootsingle` | Optional and currently not used. Alterative boot that runs `altbootusb` when `bootslot` is `singlerescue` and runs `swuboot` when `bootslot` is `singlenormal`. |
| `adjustbootsourceB` | Start the system from slot B in double copy. |
| `adjustbootsourceA` | Start the system from slot A in double copy. |
| `altbootRollbackB` | Roll back to start the system from slot B. |
| `altbootRollbackA` | Roll back to start the system from slot A. |
| `altbootdual` | Switch the boot slot to another slot for rollback. |
| `swuboot` | Start SWUpdate inside ramdisk. |
| `altbootcmd` | Alternative boot command. Provided by U-Boot `bootcount` feature. For detauils, see *u-boot/ doc/README.bootcount*. |
| `bootcount` | Current boot counts. Provided by U-Boot `bootcount` feature to indicate the current reboot counts. When `bootcount` reaches `bootlimit`, U-Boot switches to `singlerescue` in single copy and another copy of slot in double copy. For details, see *u-boot/ doc/README.bootcount*. |
| `bootlimit` | `const` value to indicate the maximum count of reboot times. Provided by U-Boot `bootcount` feature. For details, see *u-boot/ doc/README.bootcount*. |
| `upgrade_available` | Set after the image update and cleared in the first boot into kernel. Provided by U-Boot `bootcount` feature. For details, see *u-boot/ doc/README.bootcount*. |

## 10.3 Adding a platform to `swupdate-scripts`

This section describes how to add a platform to `swupdate-scripts` so that users can use scripts to create base and update images.

### 10.3.1 Adding a platform to `boards/cfg_boards.cfg`

`swupdate-scripts/boards/cfg_boards.cfg` contains boards that the script supports.

Add your SoC to `SUPPORTED_SOC` in `cfg_boards.cfg`.

### 10.3.2 Adding a configuration file for base image generation

The name of the configuration file is `cfg_<soc>_base.cfg`.

Copy an existing configuration file from a similar platform and modify macros.

For example, copy `cfg_imx8mm_base.cfg` as `cfg_<soc>_base.cfg` and modify macros in it.

Table 8 lists the descriptions of macros.

**Table 8. Macros**

| Macro name | Description |
|---|---|
| `SOC_NAME` | Name of the SoC. This name is used to source this board-specific cfg file. |
| `IMAGE_PT_TBL_FMT` | Partition table format, MBR or GPT.<br>Example:<br>`IMAGE_PT_TBL_FMT="MBR"` |
| `IMAGE_PT_TBL_PATH` | Path of the MBR/GPT file. If `-m` is specified in the command option, this file is regenerated.<br>Example:<br>`IMAGE_PT_TBL_PATH="${WRK_DIR}/common/swu_dualslot_7.5G.pt"` |
| `IMAGE_PT_TBL` | Information of the MBR file.<br>Format:<br>`[FILENAME:<OFFSET_START>:<OFFSET_END>]`<br>Example:<br>`IMAGE_PT_TBL="${IMAGE_PT_TBL_PATH}:0:7500M"` |
| `IMAGE_PT_TBL_LENGTH` | MBR/GPT header length. For MBR, this is always 512.<br>Example:<br>`IMAGE_PT_TBL_LENGTH=512` |
| `IMAGE_PT_TABLE_STRUCT` | MBR partition information.<br>When the MBR/GPT must be regenerated, this structure is used to generate a new MBR/GPT.<br>Format:<br>`[PARTITION_NAME:<OFFSET_START>:<OFFSET_END>:<Filesystem Type>]` |

**Table 8. Macros***...continued*

| Macro name | Description |
|---|---|
| | Numbers and the FS type in this struct are passed to the command directly.<br>`sudo parted <PARTITION_NAME> unit MiB mkpart primary <Filesystem Type> <OFFSET_START> <OFFSET_END>`<br>Example:<br>`IMAGE_PT_TABLE_STRUCT="`<br>`1:SLOTA_BOOT_PT:100:220:fat32`<br>`2:SLOTA_ROOTFS:220:3220:ext4`<br>`3:SLOTB_BOOT_PT:3220:3340:fat32`<br>`4:SLOTB_ROOTFS:3340:6340:ext4`<br>`"` |
| IMAGES_HEADER | Header of an image. Contains MBR/GPT, bootloader, and padding.<br>Format:<br>`[FILEPATH:<OFFSET_START>:<OFFSET_END>]`<br>Example:<br>`IMAGES_HEADER="`<br>`${IMAGE_PT_TBL_PATH}:0:1K`<br>`${WRK_DIR}/slota/u-boot-imx6ull14x14evk.imx:1K:8M`<br>`"` |
| IMAGES_SWUPDATE | SWUpdate image. Contains zImage, dtb, and ramfs for SWUpdate.<br>Format:<br>`[FILEPATH:<OFFSET_START>:<OFFSET_END>]`<br>Example:<br>`IMAGES_SWUPDATE="`<br>`${WRK_DIR}/slota/zImage:8M:38M`<br>`${WRK_DIR}/slota/imx6ull-14x14-evk.dtb:38M:42M`<br>`${WRK_DIR}/slota/swupdate-image-imx6ull14x14evk.cpio.gz.u-boot:42M:100M`<br>`"` |
| SLOTA_BOOT_PT_FILES | Slot A boot partition files list, which is copied into slot B boot partition.<br>Example:<br>`SLOTA_BOOT_PT_FILES="`<br>`${WRK_DIR}/slota/imx6ull-14x14-evk.dtb`<br>`${WRK_DIR}/slota/zImage`<br>`"` |
| SLOTA_BOOT_PT<br>SLOTA_ROOTFS<br>SLOTA_IMAGES | Slot A images. Contains boot partition and rootfs.<br>Format:<br>`[FILEPATH:<OFFSET_START>:<OFFSET_END>]`<br>Example:<br>`SLOTA_BOOT_PT="${WRK_DIR}/common/slota_boot_pt_120M.mirror:100M:220M"`<br>`SLOTA_ROOTFS="${WRK_DIR}/slota/core-image-base-imx6ull14x14evk.ext4:220M:3220M"` |

**Table 8. Macros***...continued*

| Macro name | Description |
|---|---|
| | SLOTA_IMAGES="<br>${SLOTA_BOOT_PT}<br>${SLOTA_ROOTFS}<br>" |
| SLOTB_BOOT_PT_FILES | Slot B boot partition files list, which is copied into slot B boot partition. This macro is only available in dual copy.<br>Example:<br>SLOTB_BOOT_PT_FILES="<br>${WRK_DIR}/slotb/imx6ull-14x14-evk.dtb<br>${WRK_DIR}/slotb/zImage<br>" |
| SLOTB_BOOT_PT<br>SLOTB_ROOTFS<br>SLOTB_IMAGES | Slot B images. Contains boot partition and `rootfs`. This macro is only available in dual copy.<br>Format:<br>[FILEPATH:<OFFSET_START>:<OFFSET_END>]<br>Example:<br>SLOTB_BOOT_PT="${WRK_DIR}/common/<br>slotb_boot_pt_120M.mirror:3220M:3340M"<br>SLOTB_ROOTFS="${WRK_DIR}/slotb/core-image-base-<br>imx6ull14x14evk.ext4:3340M:6340M"<br>SLOTB_IMAGES="<br>${SLOTB_BOOT_PT}<br>${SLOTB_ROOTFS}<br>" |

# 11 Performance

This section describes the performance of SWUpdate.

When the update process is slow, the users can improve the performance to reduce the update time. It depends on the network quality, core speed, DDR frequency, update mechanism, and so on.

This section provides solutions on the update mechanism.

## 11.1 Compressed or decompressed

By default, in SWUpdate, all artifacts are compressed before integrated into the update image. This might introduce effort for the CPU to do decompressing.

## 11.2 Partial update

Updating a whole image is straightforward, but this means to transfer a bigger amount of data if only a few files are updated. It is possible to split the update into several smaller parts to reduce the transfer size. This is called a partial update.

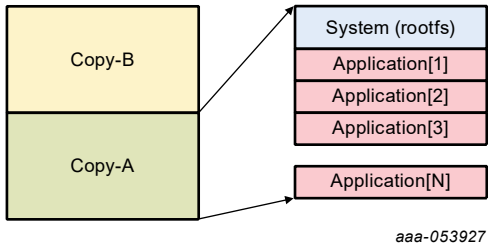



**Figure 18. Partial update**

To do a partial update, it depends on `sw-description` and update image organization.

Code example of `sw-description`:

```
imx6ull14x14evk: {
        files: (
            {
                filename = "zImage";
                path="/zImage"
                sha256 = "<zImage_sha256>";
                device = "/dev/mmcblk1p1";
                filesystem = "vfat";
                installed-directly = true;
                hook = "preinst";
            },
            {
                filename = "imx6ull-14x14-evk.dtb";
                path="/imx6ull-14x14-evk.dtb"
                sha256 = "<imx6ull-14x14-evk.dtb_sha256>";
                device = "/dev/mmcblk1p1";
                filesystem = "vfat";
                installed-directly = true;
                hook = "preinst";
            }
        );
```

***Note:*** *The `files` is used here. In practice, `images` programs the file to the device directly, which overwrites the filesystem and `files` copies the file to the path in the device.*

The partial update must also focus on the compatibility between the system and application, which can be solved by customized Lua scripts in the `sw-description` file. SWUpdate supports versioning for each artifact, and users can add their own rules to verify the compatibility between components.

***Note:*** *This application note only demonstrates upgrading specific partitions using the images method. For the files method, we provide complete `sw-description` files (template files ending with `-file`) under swupdate-scripts/legacy/boards for reference.*

## 11.3 Delta update

Delta update is another way to update the whole partition.

A device is upgraded to a version that is like the running one but adds new features and solves some bugs. When a minor issue is fixed, the new version is similar to the original one. In this case, the user can use Delta update to download the differences from the current software without downloading a full image.

For Delta update, several algorithms are used for `delta encoding` to find the differences between files, generally in binary format.

This requires an integration in `sw-description` and SWUpdate.

For details, see Delta Update with SWUpdate.

***Note:*** *As we have not tested this method, here we only provide a description and a link.*

# 12 Conclusion

This document provides a whole view of SWUpdate and guides users to set up SWUpdate on i.MX 6ULL, i.MX 8M Mini, and i.MX 93. This is a common use case.

SWUpdate provides a powerful and reliable update mechanism. It is used to download and perform updates according to metadata written into the `sw-description` file. The users may also perform other operations such as picking the right software from the collection, getting current and available partitions, preparing bootloader scripts SWUpdate is flexible to provide a growing list of features used to design update system to meet user requirements.

# 13 Appendix

## 13.1 How to run the Hawkbit server

### 13.1.1 Building the Hawkbit server

Follow Hawkbit Getting Started Guide to build the Hawkbit server or run the docker version.

### 13.1.2 Running the Hawkbit server

Assume you are in the Hawkbit directory.

Command:

```
java -jar ./hawkbit-runtime/hawkbit-update-server/target/hawkbit-update-
server-0.3.0-SNAPSHOT.jar \
--hawkbit.dmf.rabbitmq.enabled=false \
--hawkbit.server.ddi.security.authentication.anonymous.enabled=true
```

When the Hawkbit server is running, users can log in and manage the server at *http://<ip>:8080*. Both username and password are `admin`.

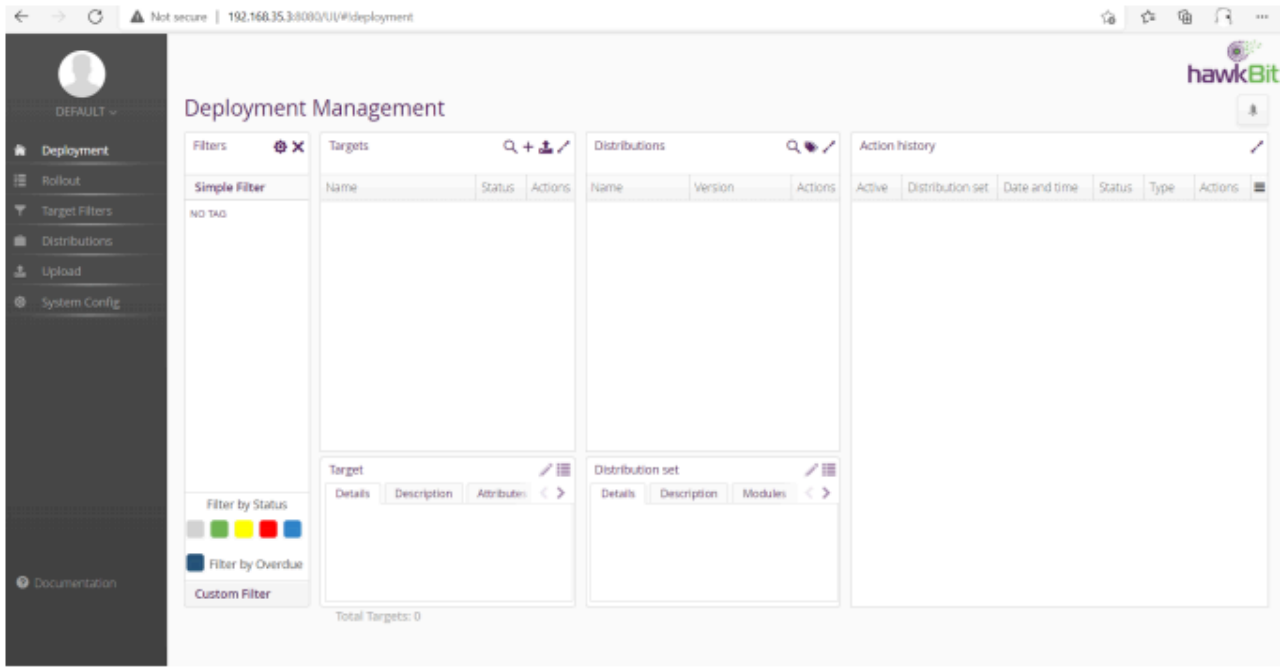



**Figure 19. User login**

### 13.1.3 Suricatta access Hawkbit

Taking i.MX 8M Mini as an example, the command is as follows:

```
swupdate -l 5 -u '-t default -u http://10.192.245.169:8080 -i iMX8MM_EVK ' -p
 "swupdate -u '-t default -u
http://10.192.245.169:8080 -i iMX8MM_EVK -c 2’”
```

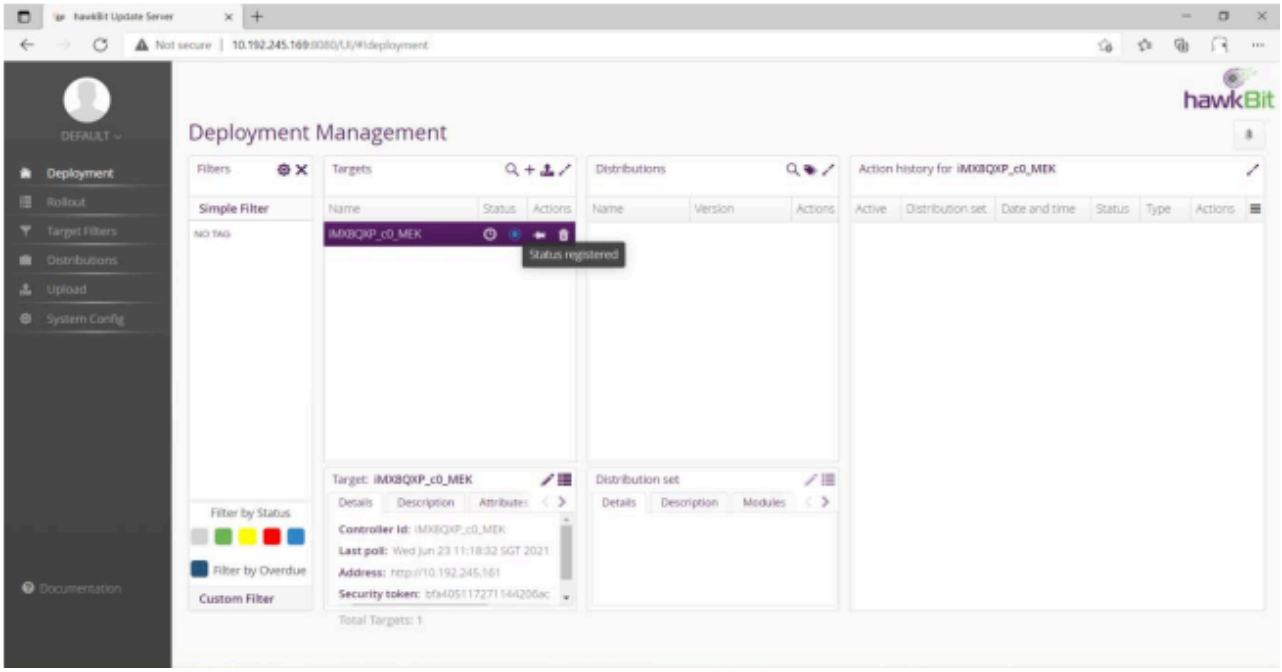



**Figure 20. Suricatta access Hawkbit**

## 13.2 How to create an image for Secondary Boot

### 13.2.1 Introduction of Secondary Boot

According to the *chip Reference Manual*, the Secondary Boot is also known as the Redundant Boot. See Section **Redundant boot support for expansion device**.

When Secondary Boot is used, for the closed security setting (for detail, see chip Security Reference Manual), if there are failures during primary image authentication, the boot ROM turns on the `PERSIST_SECONDARY_BOOT` bit and performs a software reset. After the software reset, the secondary image is used.

Table 9 describes the `PERSIST_SECONDARY_BOOT` bit.

**Table 9. Persistent bits**

| Bit name | Bit location | Description |
|---|---|---|
| `PERSIST_SECONDARY_BOOT` | SRC_GPR10[30] | This bit identifies which image must be used: primary and secondary. Used only for the boot modes that support redundant boot. |

Figure 21 shows the routine of Secondary Boot.

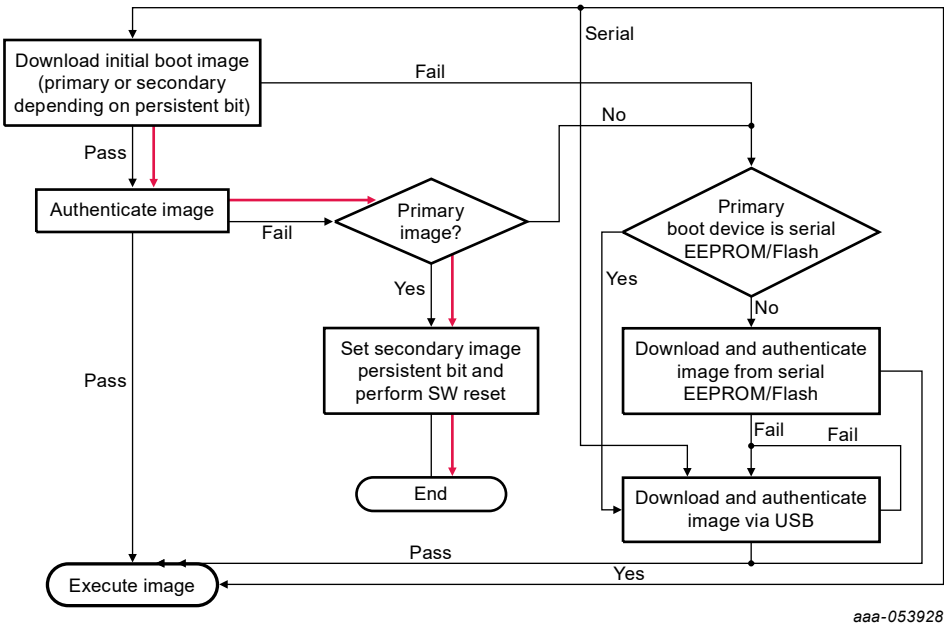



**Figure 21. Routine of Secondary Boot**

### 13.2.2 Creating a boot image with Secondary Boot

As described in the *chip Reference Manual*,

If the PERSIST_SECONDARY_BOOT is 0, the boot ROM uses address 0x0 for the primary image.

If the PERSIST_SECONDARY_BOOT is 1, the boot ROM reads the secondary image table from address 0x200 on the boot media and uses the address specified in the table.

**Table 8-19. Secondary image table format**

| |
|---|
| Reserved (chipNum) |
| Reserved (driveType) |
| tag |
| firstSectorNumber |
| Reserved (sectorCount) |

Where:

- The tag is used as an indication of the valid secondary image table. It must be 0x00112233.
- The firstSectorNumber is the first 512-byte sector number of the secondary image.

**Figure 22. Secondary image table format**

For the secondary image support, the primary image must reserve the space for the secondary image table. The following figure shows the typical structure layout on an expansion device.

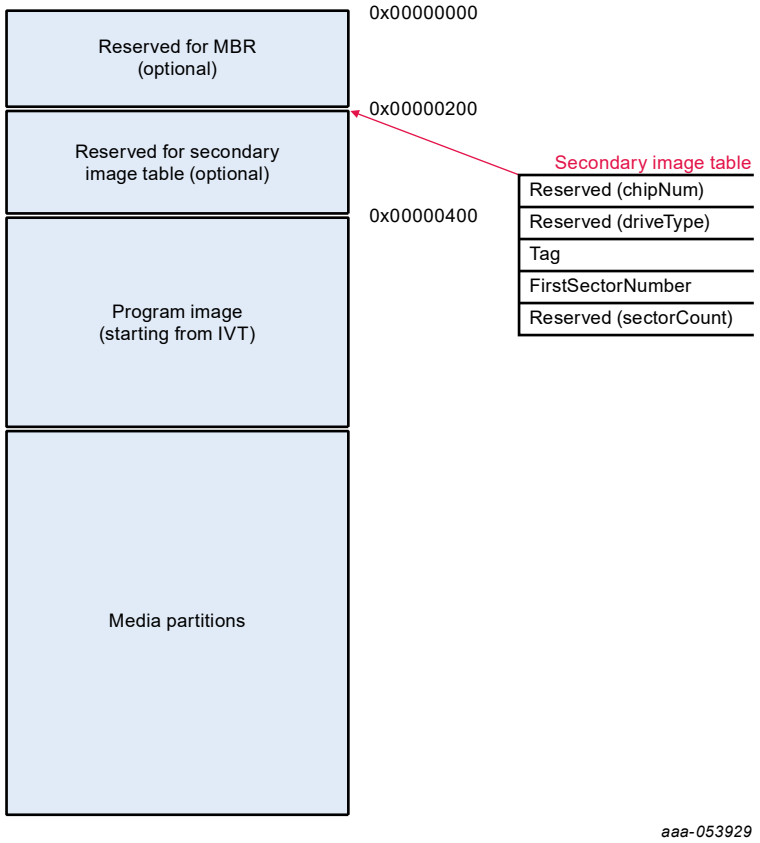



**Figure 23. Typical structure layout**

To create a boot image with Second Boot, perform the following steps:

1. Create an empty header binary. The following table lists the size of the header.

**Table 10. Header size**

| Platform | Header Size |
|---|---|
| i.MX 6, i.MX 7 | 1 kB |
| i.MX 8QuadMax A0, i.MX 8QuadXPlus A0, i.MX 8MQuad | 33 kB |
| i.MX 8QuadXPlus B0, i.MX 8QuadMax B0, i.MX 8DualX, i.MX 8DXL, i.MX 8M Nano, i.MX 8M Mini, i.MX 8M Plus, i.MX 93 | 32 kB |

The command (taking i.MX 8M Mini as an example) is as below:

```
dd if=/dev/zero of=./header.bin bs=1K count=132
```

2. Use the hex editor to modify `header.bin` for tag and `firstSectorNumber`. Change offset `0x208` to tag `0x00112233` and `0x20C` to the first sector number. In addition, pay attention to the `endian`. For example,

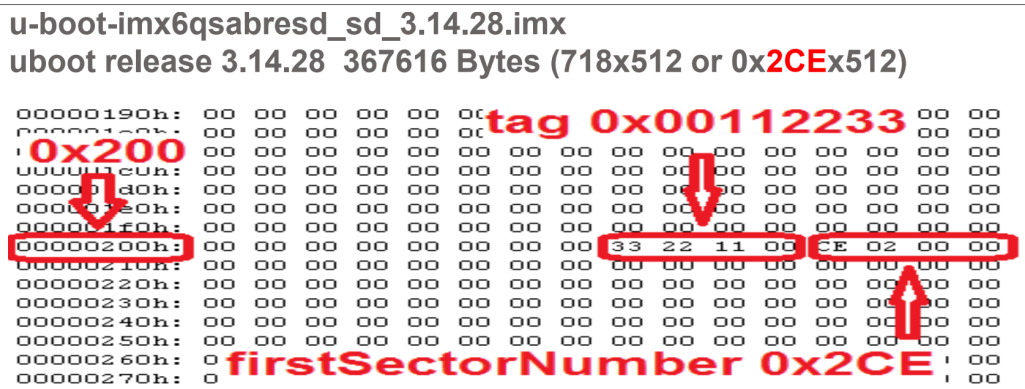



**Figure 24. Modify `header.bin` and `firstSectorNumber`**

3. Use the `cat` command to merge `header.bin`, the primary U-Boot, and secondary U-Boot image. For example,

```
cat header.bin u-boot-primary.imx u-boot-secondary.imx > u-boot-primary-secondary.imx
```

4. Program `u-boot-primary-secondary.imx` to the card.
Command:

```
dd if=./u-boot-primary-secondary.imx of=/dev/sdc bs=512 skip=1 seek=1
```

***Note:***
*As the first 512 Bytes is for MBR, do skip and seek here.*

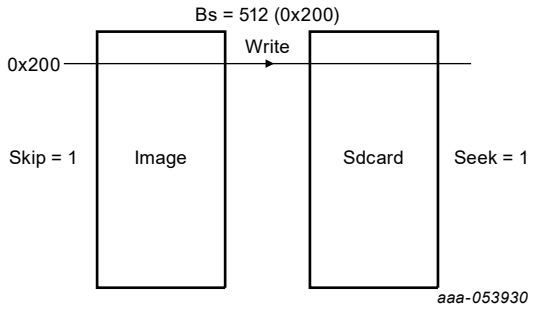



***Figure 25. Skip and seek***

### 13.2.3 Testing Secondary Boot

The `PERSIST_SECONDARY_BOOT` bit decides the boot image.

**Table 11. Persistent bits**

| Bit name | Bit location | Description |
|---|---|---|
| `PERSIST_SECONDARY_BOOT` | SRC_GPR10[30] | This bit identifies which image must be used: primary and secondary. Used only for the boot modes that support redundant boot. |

Therefore, turn on the `PERSIST_SECONDARY_BOOT` bit in U-Boot to test it.

Check the `SRC_GPR10` register address from the *chip Reference Manual* and use the following command to enable Secondary Boot:

```
mw <SRC_GPR10> 0x40000000
```

After the modification, type `reset` in U-Boot to see Secondary Boot.

***Note:***

*Do not do a POR reset here as SRC General-Purpose registers can only keep their values after a warm reset.*

## 14 References

- SWUpdate online documentation:
  - Software Management on embedded systems
  - SWUpdate Best Practice
- Hawkbit
- SWUpdate OTA i.MX8MM EVK/i.MX8QXP MEK

***Note:***

*The Arm attribution must be updated from **ARMnnn** to reflect the correct product name.*

## 15 Note About the Source Code in the Document

Example code shown in this document has the following copyright and BSD-3-Clause license:

Copyright 2024 NXP Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

## 16 Revision history

Table 12 summarizes the revisions to this document.

**Table 12. Revision history**

| Document ID | Release date | Description |
|---|---|---|
| AN13872 v.5 | 18 March 2024 | • Updated Table 3Updated<br>• Updated Table 4<br>• Updated Section 5.3<br>• Updated Section 6.3.3<br>• Added Section 7.4.3 |

**Table 12. Revision history**...*continued*

| Document ID | Release date | Description |
|---|---|---|
| | | • Added Section 8.4.7<br>• Added Section 8.4.8<br>• Added Section 8.4.9 |
| AN13872 v.4 | 22 January 2024 | • Updated images to svg format<br>• Added i.MX 8M Mini and i.MX 93<br>• Updated Section 1<br>• Updated Section 5.3<br>• Updated Section 5.4.2<br>• Updated Section 5.4.3.1<br>• Updated Section 6.1<br>• Updated Section 6.2.2<br>• Updated Section 14 |
| AN13872 v.3 | 25 July 2023 | Updated a GitHub link in Section 5.3. |
| AN13872 v.2 | 19 May 2023 | • Updated SWUpdate scripts usage.<br>• Added new `swu_generate_part_img.sh` to generate the partition mirror image.<br>• Updated some trival descriptions. |
| AN13872 v.1 | 02 May 2023 | • Added editorial updates<br>• Added Legal information page to the end of this document |
| AN13872 v.0 | 02 March 2023 | Initial release |

# Legal information

## Definitions

**Draft** — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

## Disclaimers

**Limited warranty and liability** — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

**Right to make changes** — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

**Suitability for use** — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

**Applications** — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

**Terms and conditions of commercial sale** — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at https://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

**Export control** — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

**Suitability for use in non-automotive qualified products** — Unless this document expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

**Translations** — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

**Security** — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

**NXP B.V.** — NXP B.V. is not an operating company and it does not distribute or sell products.

## Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

**NXP** — wordmark and logo are trademarks of NXP B.V.

# Contents





For more information, please visit: https://www.nxp.com

**Date of release: 18 March 2024**
**Document identifier: AN13872**