

Linux Embarqué

Mender 2.0 101: Bien démarrer avec les 'update modules'

Pierre-Jean Texier & Joris Offouga

Dimanche 17 Novembre

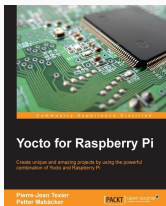
Toulouse, CDL 2019

/Pierre-Jean

- CTO & Ingénieur Linux Embarqué **Koncepto**



- 29 ans
- FOSS enthusiast
- Contributions : U-Boot, Kernel Linux, Yocto/OE, Buildroot ...
- Co-auteur de **"Yocto for Raspberry Pi"** et auteur dans GNU/Linux magazine et Open silicium (RIP)



/Joris

- Apprenti Ingénieur Linux Embarqué **Evbox Bordeaux**



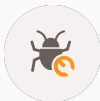
- 22 ans
- Open Source enthusiast
- Contributions : U-Boot, Yocto/OE, Buildroot

1. Mise à jour des systèmes embarqués
2. stm32mp1
3. Mender
4. Demo
5. Conclusion

Mise à jour des systèmes embarqués

Pourquoi un système de mise à jour ?

- Bugs Logiciels



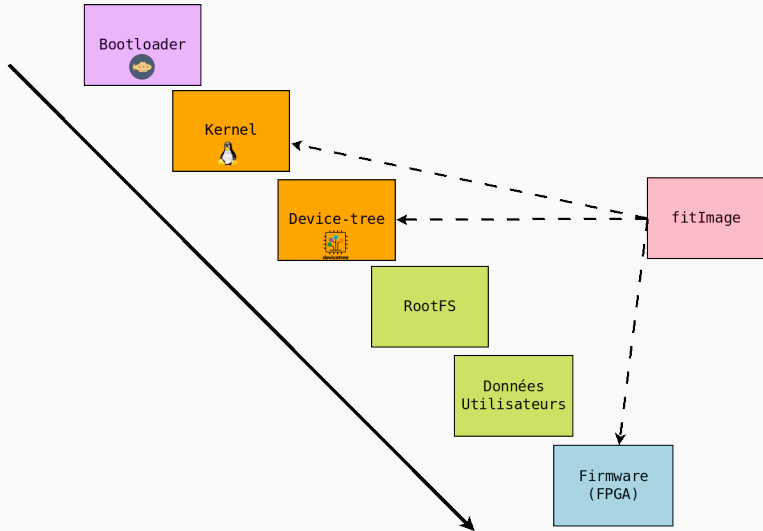
- Ajouts de fonctionnalités



- Correctifs liés à la sécurité (CVE)



Ok, mais quels composants ?



stm32mp1



Basé sur une Architecture "Hétérogène" ARM Cortex A7/ ARM Cortex M4.

- Single/Dual Arm Cortex A7 @ 650 MHz
- Single Arm cortex m4 @ 200 MHz
- 3D GPU OpenGL @ 533 MHz
- Multiples périphériques I²C, UART, USART, QSPI, CRYPTO, DSI/TFT, HDMI..
- Secure Boot, TrustZone

- **STM32MP151**
 - Single Arm Cortex A7
 - Single Arm Cortex M4
 - Pas de GPU
 - Multiples périphériques sauf CAN et DSI
- **STM32MP153**
 - Single Arm Cortex A7
 - Single Arm Cortex M4
 - Pas de GPU
 - Multiples périphériques sauf DSI
- **STM32MP157**
 - Dual Arm Cortex A7
 - Single coeur Arm Cortex M4
 - 3D GPU
 - Multiples périphériques

STM32MP157

System

5x LDOs
Crystal & internal oscillators
MDMA + 2x DMA
Reset and Clock
Watchdogs (2x I & W)
96-bit unique ID
Up to 176 GPIOs

Security

TrustZone
DES, TDES, AES-256
SHA-256, MD5, HMAC
3x Tamper Pins with 1 active
T*, V and 32KHz detection
Secure ROM and RAMs
Secure Peripherals
Secure RTC
Analog true RNG

Dual Cortex-A7 @ 650MHz

Core 1 @ 650MHz L1 32KB I / 32KB D	Core 2 @ 650MHz L1 32KB I / 32KB D
NEON SMD	NEON SMD
256KB L2 cache	

Cortex-M4 @ 200MHz

FPU	MPU
-----	-----

DDR3/DDR3L 32-bit @ 533MHz

LPDDR2/LPDDR3 32-bit @ 533MHz

System RAM 256KB	MCU System RAM 384KB
Retention RAM 64KB	Backup RAM 4KB
Boot ROM 128KB	OTP Fuse 3Kb

Control

2x 16-bit motor control PWM synchronized AC timer	
10x 16-bit timer	5x 16-bit LP timer
2x 32-bit timer	

3D GPU OpenGL ES2.0 @ 533MHz

26Mtri/sec, 133Mpix/sec

Connectivity

24-bit Parallel RGB Display	MIPI DSI 2 lanes @ 1Gbps
Camera Interface	HDMI CEC
1Gbps Ethernet	2x FDCAN / TTCAN
2x USB2.0 Host HS	USB2.0 OTG FS/HS
MDIO	DFSDM 8 channels / 6 filters
6x PC	4x UART, 4x USART
6x SPI / 3x PS	4x SAI
SPDIF Tx / Rx 4 inputs	Dual Quad SPI
3x SDIO3.0 / SD3 / eMMC 4.51	16-bit SLC NAND, 8-bit-ECC

Analog

2x 16-bit ADC 3MSPS 22 channels	
2x 12-bit DAC	
Temperature sensor	

stm32mp1: Discovery Kit 1



- STM32MP157A Dual Cortex A7 + Single Cortex M4
- 512 MB DDR3L RAM
- Port Gigabit Ethernet
- Raspberry Compatible Header
- Arduino Compatible Header
- 1 USB type C OTG
- 4 USB 2.0 Host ports
- Micro SD Slot
- Debugger ST-Link V2
- Buttons, Leds, Audio, HDMI

stm32mp1: Support logiciel

- Yocto/OE
 - meta-st-stm32mp
 - meta-stm32mp1
- Buildroot
 - stm32mp157c-dk2
- U-Boot
 - stm32mp1
- Kernel Linux
 - stm32mp157c-dk1
 - stm32mp157c-dk2
 - stm32mp157c-ed1
 - stm32mp157c-ev1
- Zephyr OS
 - stm32mp157c-dk2
- Optee OS

Mender



Mender is an end-to-end open source update manager for IoT

- <https://github.com/mendersoftware>
- Apache 2.0
- Solution **Clé en main** (client + serveur)
- Client lourd (6.3 Mo)
- Mise à jour Symétrique seulement
 - 4 Partitions (Boot + RootFS A + RootFS B + Data)
- U-Boot, GRUB
- SD/eMMC, UBI, Raw NAND, NOR
- HTTP(S)

- Archive TAR .mender (gestion signature via l'outil mender-artifact)
- Rollback intégré
- **State Scripts** pour la gestion des actions durant l'opération de mise à jour
- Mise à jour locale (**mode standalone**) ou distante (**Managed**)
- Facile d'intégration (Auto-patch)
- **mender -commit** pour valider la mise à jour en mode **standalone**
- Bien documenté
- Exemples sur <https://hub.mender.io/>
- Build System



Mender: La nouveauté

- Maintenant en 3 parties



Mender Open Source

On-premise management server, free to download and use with no vendor lock-in.

[GET STARTED HERE](#)

[LEARN MORE](#) →



Mender Professional

Connect your devices to our hosted server, and manage your updates through our web UI.

[SIGN UP HERE](#)

[LEARN MORE](#) →



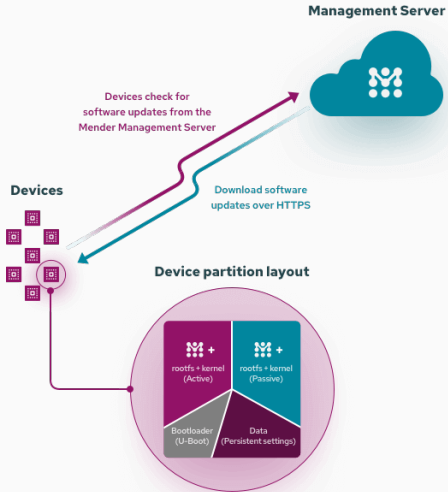
Mender Enterprise

Manage your own production server at scale, with customized updates and advanced features.

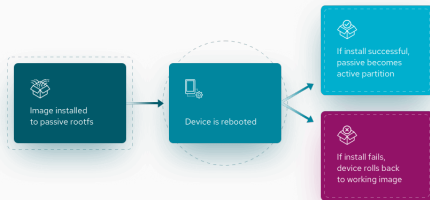
[CONTACT US FOR A QUOTE](#)

[CHECK FEATURES](#) →

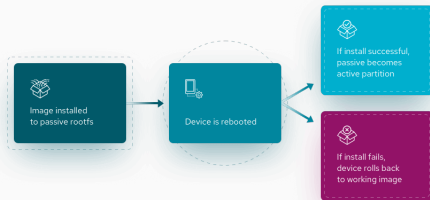
Mender: Architecture



Mender: schéma A/B

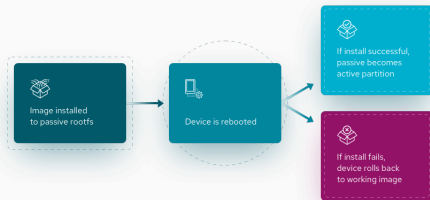


Mender: schéma A/B



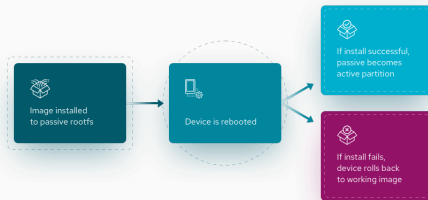
- Avantages:

Mender: schéma A/B



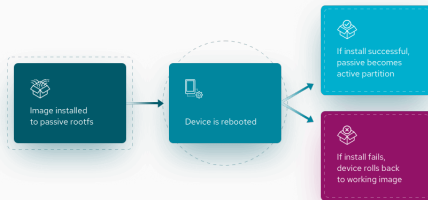
- Avantages:
 - Robuste (atomique)

Mender: schéma A/B



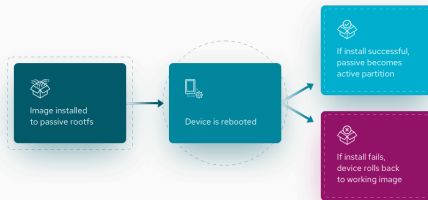
- Avantages:
 - Robuste (atomique)
 - Retour en arrière possible si erreur

Mender: schéma A/B



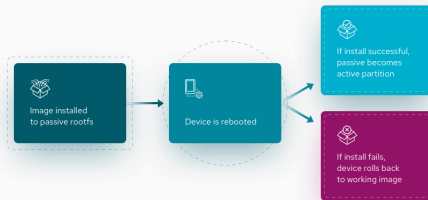
- Avantages:
 - Robuste (atomique)
 - Retour en arrière possible si erreur
 - Downtime faible

Mender: schéma A/B



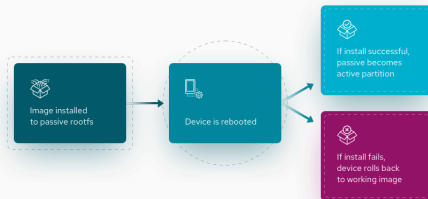
- Avantages:
 - Robuste (atomique)
 - Retour en arrière possible si erreur
 - Downtime faible
- Inconvénients:

Mender: schéma A/B



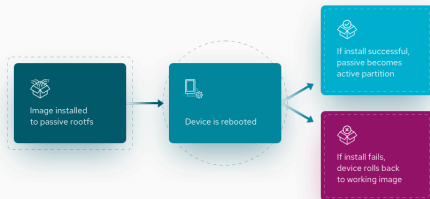
- Avantages:
 - Robuste (atomique)
 - Retour en arrière possible si erreur
 - Downtime faible
- Inconvénients:
 - L'espace mémoire requis

Mender: schéma A/B



- Avantages:
 - Robuste (atomique)
 - Retour en arrière possible si erreur
 - Downtime faible
- Inconvénients:
 - L'espace mémoire requis
 - La bande passante

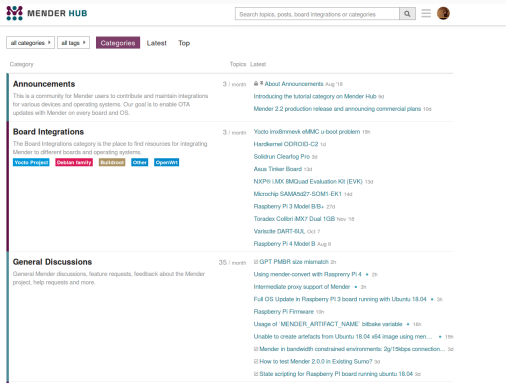
Mender: schéma A/B



- Avantages:
 - Robuste (atomique)
 - Retour en arrière possible si erreur
 - Downtime faible
- Inconvénients:
 - L'espace mémoire requis
 - La bande passante
 - Demande des efforts d'intégrations avec le bootloader (GRUB, U-Boot, ...)

Mender: Mender Hub

- Référentiel piloté par la communauté
- +40 cartes supportées (Buildroot, Yocto/OE, Debian, ...)
- Pour commencer rapidement avec Mender
- Forum actif !



The screenshot displays the Mender Hub website interface. At the top, there is a search bar with the placeholder text "Search topics, posts, board integrations or categories". Below the search bar, there are navigation tabs: "all categories", "all tags", "Categories", "Latest", and "Top". The main content area is divided into three sections: "Announcements", "Board Integrations", and "General Discussions". Each section has a description and a list of topics. The "Announcements" section includes a topic about introducing the tutorial category. The "Board Integrations" section lists various board integrations such as Yocto imx6mev eMMC u-boot problem, Hardkernel ODROID-C2, and SolidRun Clearfog Pro. The "General Discussions" section lists topics like GPT PMBR size mismatch, Using mender-convert with Raspberry Pi 4, and Intermediate proxy support of Mender.

MENDER HUB

Search topics, posts, board integrations or categories

all categories all tags Categories Latest Top

Category Topics Latest

Announcements 3 / month

This is a community for Mender users to contribute and maintain integrations for various devices and operating systems. Our goal is to enable OTA updates with Mender on every board and OS.

9 About Announcements Aug 16

Introducing the tutorial category on Mender Hub 16

Mender 2.2 production release and announcing commercial plans 106

Board Integrations 3 / month

The Board Integrations category is the place to find resources for integrating Mender to different boards and operating systems.

Yocto Project Debian family Buildroot Other OpenWrt

Yocto imx6mev eMMC u-boot problem 136

Hardkernel ODROID-C2 16

SolidRun Clearfog Pro 36

Asus Tinker Board 136

NXP i.MX8MQ Evaluation Kit (EVK) 136

Microchip SAM5A27-SOM-EK1 146

Raspberry Pi 3 Model B/B+ 276

Toradex Colibri iMX7 Dual 1GB Rev 10

VeriSilicon DART-6UL Oct 7

Raspberry Pi 4 Model B Aug 6

General Discussions 35 / month

General Mender discussions, feature requests, feedback about the Mender project, help requests and more.

GPPT PMBR size mismatch 26

Using mender-convert with Raspberry Pi 4 26

Intermediate proxy support of Mender 26

Full OS Update in Raspberry Pi 3 board running with Ubuntu 18.04 26

Raspberry Pi Firmware 136

Usage of 'MENDER_ARTIFACT_NAME' bitbake variable 16

Unable to create artefacts from Ubuntu 18.04 x64 image using men... 176

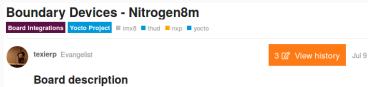
Mender in bandwidth constrained environments: 2g/15kba connection... 36

How to test Mender 2.0.0 in Existing Sumo? 36

State scripting for Raspberry Pi board running ubuntu 18.04 36

Mender: Mender Hub, exemples ...

- **Boundary Devices Nitrogen8m**



Mender: Mender Hub, exemples ...

- Boundary Devices Nitrogen8m
- Technexion PICO-PI-IMX7

Boundary Devices - Nitrogen8m

[Board Integrations](#) [Yocto Project](#) [imx8](#) [thud](#) [nsp](#) [yocto](#)



texierp Evangelist

3 View history

Jul 9

Board description

Technexion PICO-PI-IMX7

[Board Integrations](#) [Yocto Project](#) [imx7](#) [pico-imx7](#) [technexion](#) [thud](#) [sumo](#) [nsp](#) [yocto](#)



jorisoffouge Pioneer

7 View history

Sep 27

The official [Mender documentation](#) explains how Mender works. This is a board-specific complement to the official documentation.

Mender: Mender Hub, exemples ...

- Boundary Devices Nitrogen8m
- Technexion PICO-PI-IMX7
- Microchip sama5d27-som1-ek1

Boundary Devices - Nitrogen8m

Board Integrations **Yocto Project**     








textierp Evangelist

3  View history Jul 9

Board description


Microchip SAMA5d27-SOM1-EK1

Board Integrations **Yocto Project**      



textierp Evangelist

10  View history 30d

The official [Mender documentation](#)  explains how Mender works. This is a board-specific complement to the official documentation.

Technexion PICO-PI-IMX7

Board Integrations **Yocto Project**       



jorisoffouga Pioneer

7  View history Sep 27

The official [Mender documentation](#)  explains how Mender works. This is a board-specific complement to the official documentation.

Mender: Mender Hub, exemples ...

- Boundary Devices Nitrogen8m
- Technexion PICO-PI-IMX7
- Microchip sama5d27-som1-ek1
- Raspberry Pi 4 Model B

Boundary Devices - Nitrogen8m

Board Integrations Yocto Project imx8 thud rasp yocto



textierp Evangelist

3 View history Jul 9

Board description

Microchip SAMA5d27-SOM1-EK1

Board Integrations Yocto Project sama5 atmel microchip thud sumo yocto



textierp Evangelist

10 View history 30d

The official Mender documentation explains how Mender works. This is a board-specific complement to the official documentation.

Technexion PICO-PI-IMX7

Board Integrations Yocto Project imx7 pico-imx7 technexion thud sumo rasp yocto



jorisoffouga Pioneer

7 View history Sep 27

The official Mender documentation explains how Mender works. This is a board-specific complement to the official documentation.

Raspberry Pi 4 Model B

Board Integrations Yocto Project raspberry raspberry-pi-4 yocto



mirzak Leader

2 View history Oct 3

Board description

State scripts

Mender : State Scripts

- La machine d'état de Mender compte 9 états

Mender : State Scripts

- La machine d'état de Mender compte 9 états
- Elle se charge d'invoquer les "state scripts" entre chaque transition ...

Mender : State Scripts

- La machine d'état de Mender compte 9 états
- Elle se charge d'invoquer les "state scripts" entre chaque transition ...
- ... et d'analyser le code retour de chacun, 3 codes retours possibles:

Mender : State Scripts

- La machine d'état de Mender compte **9** états
- Elle se charge d'invoquer les "state scripts" entre chaque transition ...
- ... et d'analyser le code retour de chacun, **3** codes retours possibles:
 - **0**: **OK**, la mise à jour se poursuit
 - **1**: **NOK**, la mise à jour s'arrête, et Mender "Roll-Back"
 - **21**: **Retry later**, la machine d'état rentre en attente

State scripts

<STATE_NAME>_<ACTION>_<ORDERING_NUMBER>_<OPTIONAL_DESCRIPTION>

State scripts

<STATE_NAME>_<ACTION>_<ORDERING_NUMBER>_<OPTIONAL_DESCRIPTION>

Exemple: ArtifactCommit_Enter_00-wait-sync

```
#!/bin/sh
# Wait until the network is connected and the time is synced
# before committing an update

RETRY_LATER=21
OK=0

ping 8.8.8.8 -c1 > /dev/null 2>&1
if [ $? != 0 ]; then
    exit ${RETRY_LATER}
fi

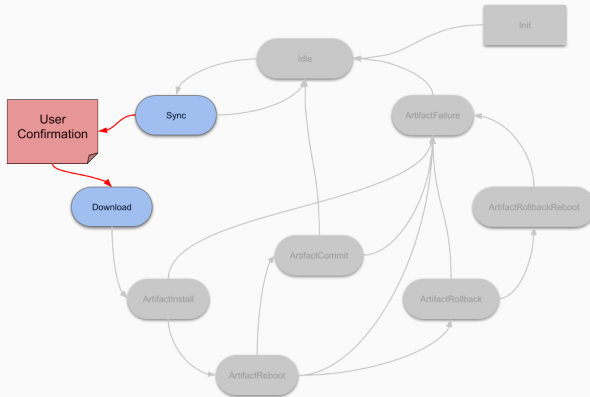
TIME_SYNC_STATUS=$(timedatectl | grep NTP | awk '{print $NF}')
if [ "${TIME_SYNC_STATUS}" != "active" ]
then
    exit ${RETRY_LATER}
fi

exit ${OK}
```

-> [mender/examples/state-scripts/wait-for-network](#)

State scripts: Root File System

- Partie intégrante du Root file system sous `/etc/mender/scripts`



- Example "Retry Later"

Mise à jour logicielle

La version 9.1.0.249(C432E1R2P1) est prête à être installée. Toucher ULTÉRIEUREMENT implique qu'elle sera automatiquement installée entre 02:00 – 04:00 si votre appareil n'est pas utilisé.

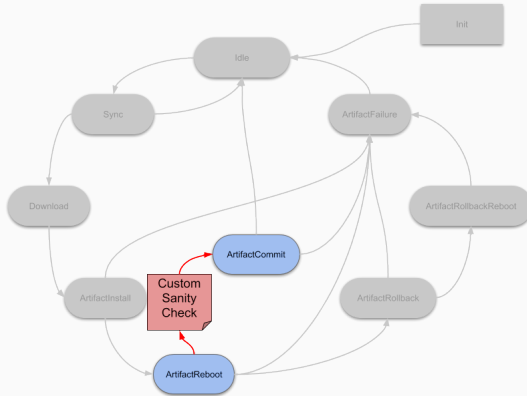
INSTALLER

DÉTAILS

ULTÉRIEUREMENT

State scripts: Artifact

- Dans l'artefact *.mender



Intégration

Mender & stm32mp1: intégration 1/2

■ Intégration avec U-Boot:

```
MENDER_FEATURES_ENABLE_append = " mender-uboot mender-uboot-stm32mp1 \  
    mender-image-gpt"  
MENDER_FEATURES_DISABLE_append = " mender-grub mender-image-uefi"  
  
# Environment  
MENDER_UBOOT_ENV_STORAGE_DEVICE_OFFSET_1 = "0x280000"  
# Redundant environment  
MENDER_UBOOT_ENV_STORAGE_DEVICE_OFFSET_2 = "0x300000"  
  
# Remove the Boot partition  
MENDER_BOOT_PART_SIZE_MB = "0"  
  
MENDER_DATA_PART = "${MENDER_STORAGE_DEVICE_BASE}6"  
MENDER_ROOTFS_PART_A = "${MENDER_STORAGE_DEVICE_BASE}4"  
MENDER_ROOTFS_PART_B = "${MENDER_STORAGE_DEVICE_BASE}5"  
MENDER_UBOOT_FSBL1 = "u-boot-spl.stm32"  
MENDER_UBOOT_FSBL2 = "u-boot-spl.stm32"  
MENDER_UBOOT_SSBL = "u-boot.img"  
MENDER_UBOOT_FSBL1_NAME = "fsbl1"  
MENDER_UBOOT_FSBL2_NAME = "fsbl2"  
MENDER_UBOOT_SSBL_NAME = "uboot"
```

Mender & stm32mp1: intégration 2/2

- meta-cdl/recipes-bsp/u-boot/u-boot-mender-st.inc

```
FILESEXTRAPATHS_prepend := "${THISDIR}/u-boot-mainline:"  
  
SRC_URI_remove = " \  
    file://0005-fw_env_main.c-Fix-incorrect-size-for-malloc-ed-strin.patch \  
    file://0006-env-Kconfig-Add-descriptions-so-environment-options-.patch \  
    "  
  
SRC_URI_append = " file://0001-ARM-STM32MP1-Add-support-to-mender.patch"  
  
MENDER_UBOOT_AUTO_CONFIGURE = "0"  
BOOTENV_SIZE = "0x2000"
```

- Ajout des variables `CONFIG_BOOTCOUNT_ENV` et `CONFIG_BOOTCOUNT_LIMIT`

```
+CONFIG_BOOTCOUNT_LIMIT=y  
+CONFIG_BOOTCOUNT_ENV=y
```


Mender & stm32mp1: Paramètres de l'image

- Utilisation de systemd :

```
# Use systemd as init system
VIRTUAL-RUNTIME_init_manager = "systemd"
DISTRO_FEATURES_BACKFILL_CONSIDERED = "sysvinit"
VIRTUAL-RUNTIME_initscripts = ""
DISTRO_FEATURES_append = " systemd wifi"
```

- Configuration Artefact et Serveur :

```
MENDER_ARTIFACT_NAME = "CDL-release-1"
MENDER_SERVER_URL = "https://hosted.mender.io"
MENDER_TENANT_TOKEN = "..."
```

Mender & stm32mp1: génération

```
$: bitbake core-image-base
```

- core-image-base-stm32mp157a-dk1.gptimg
- core-image-base-stm32mp157a-dk1.mender
- Copie de l'image :

```
$: sudo bmaptool copy core-image-base-stm32mp157a-dk1.gptimg /dev/sdX
```



Mender & stm32mp1: Hosted Mender

autorisation du device:

[illegible]



- Peu flexible (schéma A/B seulement)
- Du retard par rapport à **RAUC**, **SWUpdate**
- Binary Delta Update pour la partie commerciale uniquement ...



- ... Les "update modules"

Update Modules: quesako ?!

- Une extension au client Mender depuis la version 2.0

Update Modules: quesako ?!

- Une extension au client Mender depuis la version 2.0
- Articulé autour du même flow d'exécution que les **state scripts**

Update Modules: quesako ?!

- Une extension au client Mender depuis la version 2.0
- Articulé autour du même flow d'exécution que les **state scripts**
- Un "simple" exécutable (bash, C/C++, go, rust, ...)

Update Modules: quesako ?!

- Une extension au client Mender depuis la version 2.0
- Articulé autour du même flow d'exécution que les **state scripts**
- Un "simple" exécutable (bash, C/C++, go, rust, ...)
- Appelé à chaque état (**\$1** = état / **\$2** = payload de l'artefact)

Update Modules: quesako ?!

- Une extension au client Mender depuis la version **2.0**
- Articulé autour du même flow d'exécution que les **state scripts**
- Un "simple" exécutable (bash, C/C++, go, rust, ...)
- Appelé à chaque état (**\$1** = état / **\$2** = payload de l'artefact)
- Permet un nouveau type de mise à jour

Update Modules: quesako ?!

- Une extension au client Mender depuis la version 2.0
- Articulé autour du même flow d'exécution que les **state scripts**
- Un "simple" exécutable (bash, C/C++, go, rust, ...)
- Appelé à chaque état (**\$1** = état / **\$2** = payload de l'artefact)
- Permet un nouveau type de mise à jour
- Définit comment le client Mender installe une mise à jour sur un périphérique.

Update Modules: quesako ?!

- Une extension au client Mender depuis la version 2.0
- Articulé autour du même flow d'exécution que les **state scripts**
- Un "simple" exécutable (bash, C/C++, go, rust, ...)
- Appelé à chaque état (**\$1** = état / **\$2** = payload de l'artefact)
- Permet un nouveau type de mise à jour
- Définit comment le client Mender installe une mise à jour sur un périphérique.
- De nombreux exemples fournis par Mender

Update Modules: quesako ?!

- Une extension au client Mender depuis la version 2.0
- Articulé autour du même flow d'exécution que les **state scripts**
- Un "simple" exécutable (bash, C/C++, go, rust, ...)
- Appelé à chaque état (**\$1** = état / **\$2** = payload de l'artefact)
- Permet un nouveau type de mise à jour
- Définit comment le client Mender installe une mise à jour sur un périphérique.
- De nombreux exemples fournis par Mender
- exemple:
 - Comment installer un conteneur (**docker**, **systemd-nspawn**, ...)

Update Modules: quesako ?!

- Une extension au client Mender depuis la version 2.0
- Articulé autour du même flow d'exécution que les **state scripts**
- Un "simple" exécutable (bash, C/C++, go, rust, ...)
- Appelé à chaque état (**\$1** = état / **\$2** = payload de l'artefact)
- Permet un nouveau type de mise à jour
- Définit comment le client Mender installe une mise à jour sur un périphérique.
- De nombreux exemples fournis par Mender
- exemple:
 - Comment installer un conteneur (**docker**, **systemd-nspawn**, ...)
 - Comment installer un 'simple' fichier (.deb, .ipk, **OP-TEE**, **FPGA**, ...)

Update Modules: quesako ?!

- Une extension au client Mender depuis la version 2.0
- Articulé autour du même flow d'exécution que les **state scripts**
- Un "simple" exécutable (bash, C/C++, go, rust, ...)
- Appelé à chaque état (**\$1** = état / **\$2** = payload de l'artefact)
- Permet un nouveau type de mise à jour
- Définit comment le client Mender installe une mise à jour sur un périphérique.
- De nombreux exemples fournis par Mender
- exemple:
 - Comment installer un conteneur (**docker**, **systemd-nspawn**, ...)
 - Comment installer un 'simple' fichier (.deb, .ipk, **OP-TEE**, **FPGA**, ...)
 - Comment installer une arborescence donnée ...

Update Modules: quesako ?!

- Une extension au client Mender depuis la version 2.0
- Articulé autour du même flow d'exécution que les **state scripts**
- Un "simple" exécutable (bash, C/C++, go, rust, ...)
- Appelé à chaque état (**\$1** = état / **\$2** = payload de l'artefact)
- Permet un nouveau type de mise à jour
- Définit comment le client Mender installe une mise à jour sur un périphérique.
- De nombreux exemples fournis par Mender
- exemple:
 - Comment installer un conteneur (**docker**, **systemd-nspawn**, ...)
 - Comment installer un 'simple' fichier (.deb, .ipk, **OP-TEE**, **FPGA**, ...)
 - Comment installer une arborescence donnée ...
 - `/etc/foo.conf - /usr/bin/foo -`
`/lib/systemd/system/foo.service`

Update Modules: quesako ?!

- Une extension au client Mender depuis la version 2.0
- Articulé autour du même flow d'exécution que les **state scripts**
- Un "simple" exécutable (bash, C/C++, go, rust, ...)
- Appelé à chaque état (**\$1** = état / **\$2** = payload de l'artefact)
- Permet un nouveau type de mise à jour
- Définit comment le client Mender installe une mise à jour sur un périphérique.
- De nombreux exemples fournis par Mender
- exemple:
 - Comment installer un conteneur (**docker**, **systemd-nspawn**, ...)
 - Comment installer un 'simple' fichier (.deb, .ipk, **OP-TEE**, **FPGA**, ...)
 - Comment installer une arborescence donnée ...
 - `/etc/foo.conf - /usr/bin/foo -
/lib/systemd/system/foo.service`
 - Comment installer un firmware (série, Bus CAN, MQTT, ...), ...

Update Modules: quesako ?!

- Une extension au client Mender depuis la version 2.0
- Articulé autour du même flow d'exécution que les **state scripts**
- Un "simple" exécutable (bash, C/C++, go, rust, ...)
- Appelé à chaque état (**\$1** = état / **\$2** = payload de l'artefact)
- Permet un nouveau type de mise à jour
- Définit comment le client Mender installe une mise à jour sur un périphérique.
- De nombreux exemples fournis par Mender
- exemple:
 - Comment installer un conteneur (**docker**, **systemd-nspawn**, ...)
 - Comment installer un 'simple' fichier (.deb, .ipk, **OP-TEE**, **FPGA**, ...)
 - Comment installer une arborescence donnée ...
 - `/etc/foo.conf - /usr/bin/foo -
/lib/systemd/system/foo.service`
 - Comment installer un firmware (série, Bus CAN, MQTT, ...), ...

Update Modules: un firmware ?!

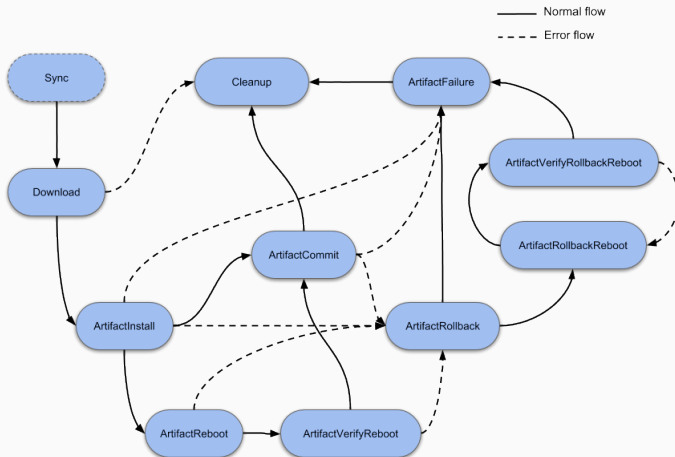
```
stm32flash -b 115200 -R -i 1@0,-2@0,3@0,-4@0 /dev/ttyS2 \  
-w "$FILES"/files/fw.hex
```

```
GPIO sequence start  
  setting gpio 1 to 1... OK  
  delay 100000 us  
  setting gpio 2 to 0... OK  
  delay 100000 us  
  setting gpio 3 to 1... OK  
  delay 100000 us  
  setting gpio 4 to 0... OK  
GPIO sequence end
```

```
Version      : 0x31  
Option 1     : 0x00  
Option 2     : 0x00  
Device ID    : 0x0431 (STM32F411xx)  
- RAM        : Up to 128KiB (12288b reserved by bootloader)  
- Flash      : Up to 512KiB (size first sector: 1x16384)  
- Option RAM : 16b  
- System RAM : 30KiB
```

```
Resetting device...  
Reset done.
```

Update Modules: Le workflow



Update Modules: Dans les détails

Un **état** essentiel: **ArtifactInstall**

- c'est l'état qui définit le comportement du client => **Où et comment ?**
- le premier état à implémenter lors de la création d'un **update module**
- celui qui sera en charge de créer un backup si le **Roll Back** est supporté

Même si optionnel, deux autres états sont tout aussi importants:

Update Modules: Dans les détails

Un **état** essentiel: **ArtifactInstall**

- c'est l'état qui définit le comportement du client => **Où et comment ?**
- le premier état à implémenter lors de la création d'un **update module**
- celui qui sera en charge de créer un backup si le **Roll Back** est supporté

Même si optionnel, deux autres états sont tout aussi importants:

- **ArtifactCommit**: Action à réaliser lors de la validation de la mise à jour.
 - Exemple: logique Bootloader: `fw_setenv upgrade_available 0`

Update Modules: Dans les détails

Un **état** essentiel: **ArtifactInstall**

- c'est l'état qui définit le comportement du client => **Où et comment ?**
- le premier état à implémenter lors de la création d'un **update module**
- celui qui sera en charge de créer un backup si le **Roll Back** est supporté

Même si optionnel, deux autres états sont tout aussi importants:

- **ArtifactCommit**: Action à réaliser lors de la validation de la mise à jour.
 - Exemple: logique Bootloader: `fw_setenv upgrade_available 0`
- **ArtifactRollback**: Action à réaliser pour revenir dans un état fonctionnel
 - Exemple: installation des fichiers de backup

Update Modules: exemple

- exécutable dans : `/usr/share/mender/modules/v3`

Update Modules: exemple

- exécutable dans : `/usr/share/mender/modules/v3`

```
cat << "EOF" > cdl-demo
#!/bin/sh

STATE="$1"
FILES="$2"

case "$STATE" in
    ArtifactInstall)
        cp "$FILES"/files/* /etc
        ;;
esac
exit 0
EOF
```

Update Modules: exemple

- exécutable dans : `/usr/share/mender/modules/v3`

```
cat << "EOF" > cdl-demo
#!/bin/sh

STATE="$1"
FILES="$2"

case "$STATE" in
    ArtifactInstall)
        cp "$FILES"/files/* /etc
        ;;
esac
exit 0
EOF
```

Génération de l'artefact (sur la partie host):

```
$: mender-artifact write module-image -t "stm32mp157a-dk1" \
  -o cdl-demo.mender \
  -T cdl-demo \
  -n cdl-demo-1.0 -f cdl-demo-file
```

Update Modules: exemple

Vérification de l'artefact (sur la partie host):

```
$: mender-artifact read cdl-demo.mender
Mender artifact:
  Name: cdl-demo-1.0
  Format: mender
  Version: 3
  Signature: no signature
  Compatible devices: '[stm32mp157a-dk1]'
  Provides group:
  Depends on one of artifact(s): []
  Depends on one of group(s): []
  State scripts:

Updates:
  0:
    Type:    cdl-demo
    Provides: Nothing
    Depends: Nothing
    Metadata: Nothing
    Files:
      name:    cdl-demo-file
      size:    21
  ...
```

Update Modules: exemple

Installation de l'artefact sur cible:

```
root@stm32mp157a-dk1:~# mender -install cdl-demo.mender
```

Update Modules: exemple

Installation de l'artefact sur cible:

```
root@stm32mp157a-dk1:~# mender -install cdl-demo.mender
```

Ou

```
root@stm32mp157a-dk1:~# mender -install \  
    http://192.168.1.17:8000/cdl-demo.mender
```

Update Modules: exemple

Installation de l'artefact sur cible:

```
root@stm32mp157a-dk1:~# mender -install cdl-demo.mender
```

Ou

```
root@stm32mp157a-dk1:~# mender -install \  
    http://192.168.1.17:8000/cdl-demo.mender
```

Résultat:

```
INFO[0000] Mender running on partition: /dev/mmcblk0p4  
INFO[0000] Start updating from local image file: [cdl-demo.mender]  
Installing Artifact of size 5120...  
...  
Artifact doesn't support rollback. Committing immediately.  
Committing Artifact...
```

Update Modules: exemple - ajout du Roll back

Rajout du support pour le Roll Back:

```
case "$STATE" in
+   SupportsRollback)
+       echo "Yes"
+       ;;
+
+   ArtifactInstall)
+       # Gestion de la sauvegarde pour le
+       # Rollback à mettre ici.

+       cp "$FILES"/files/* /etc
+       ;;
+
+   ArtifactRollback)
+       # Restauration des fichiers 'backup'
+       ;;
+   esac
+   exit 0
```

SupportsRollback est obligatoire afin de notifier la machine d'état du client Mender que l'état **ArtifactRollback** doit être exécuté.

Update Modules: exemple - ajout du Roll back

Nouveau test sur cible:

```
root@stm32mp157a-dk1:~# mender -install cdl-demo.mender
...
INFO[0000] Mender running on partition: /dev/mmcblk0p4
INFO[0000] Start updating from local image file: [cdl-demo.mender]
...
Use -commit to update, or -rollback to roll back the update.
```


Update Modules: exemple - ajout du Roll back

Nouveau test sur cible:

```
root@stm32mp157a-dk1:~# mender -install cdl-demo.mender
...
INFO[0000] Mender running on partition: /dev/mmcblk0p4
INFO[0000] Start updating from local image file: [cdl-demo.mender]
...
Use -commit to update, or -rollback to roll back the update.
```

pour aller plus loin:

```
+ NeedsArtifactReboot)
+     echo "Yes"
+     ;;
```

Update Modules: exemple - ajout du Roll back

Nouveau test sur cible:

```
root@stm32mp157a-dk1:~# mender -install cdl-demo.mender
...
INFO[0000] Mender running on partition: /dev/mmcblk0p4
INFO[0000] Start updating from local image file: [cdl-demo.mender]
...
Use -commit to update, or -rollback to roll back the update.
```

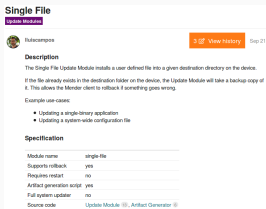
pour aller plus loin:

```
+ NeedsArtifactReboot)
+     echo "Yes"
+ ;;
```

Pour plus d'informations sur les **update modules**: [► Le protocole](#)

Demo

- STM32MP1
- Yocto/OE - mender 2.1 (warrior)
- Utilisation du module `single-file`



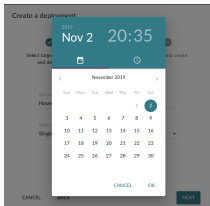
- Intégration de la `meta-mender-demo` ...
- ... pour surcharger le script `boot-script.sh`
- Utilisation des state-scripts pour la gestion du service `boot-script.service`

Conclusion

Conclusion

Ce qu'il faut retenir :

- Mender dispose maintenant d'une flexibilité
- Des exemples à disposition: **mender-update-modules**
- Des nouveautés sur la partie GUI:



- Toujours du retard sur:



Questions?

