
Provisioning the Raspberry Pi Compute Module 3 and 4

Raspberry Pi Ltd

2023-12-13: githash: ee6004c-clean

Colophon

© 2020-2023 Raspberry Pi Ltd (formerly Raspberry Pi (Trading) Ltd.)

This documentation is licensed under a Creative Commons [Attribution-NoDerivatives 4.0 International](#) (CC BY-ND) licence.

build-date: 2023-12-13

build-version: githash: ee6004c-clean

Legal disclaimer notice

TECHNICAL AND RELIABILITY DATA FOR RASPBERRY PI PRODUCTS (INCLUDING DATASHEETS) AS MODIFIED FROM TIME TO TIME ("RESOURCES") ARE PROVIDED BY RASPBERRY PI LTD ("RPL") "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. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW IN NO EVENT SHALL RPL 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 THE RESOURCES, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

RPL reserves the right to make any enhancements, improvements, corrections or any other modifications to the RESOURCES or any products described in them at any time and without further notice.

The RESOURCES are intended for skilled users with suitable levels of design knowledge. Users are solely responsible for their selection and use of the RESOURCES and any application of the products described in them. User agrees to indemnify and hold RPL harmless against all liabilities, costs, damages or other losses arising out of their use of the RESOURCES.

RPL grants users permission to use the RESOURCES solely in conjunction with the Raspberry Pi products. All other use of the RESOURCES is prohibited. No licence is granted to any other RPL or other third party intellectual property right.

HIGH RISK ACTIVITIES. Raspberry Pi products are not designed, manufactured or intended for use in hazardous environments requiring fail safe performance, such as in the operation of nuclear facilities, aircraft navigation or communication systems, air traffic control, weapons systems or safety-critical applications (including life support systems and other medical devices), in which the failure of the products could lead directly to death, personal injury or severe physical or environmental damage ("High Risk Activities"). RPL specifically disclaims any express or implied warranty of fitness for High Risk Activities and accepts no liability for use or inclusions of Raspberry Pi products in High Risk Activities.

Raspberry Pi products are provided subject to RPL's [Standard Terms](#). RPL's provision of the RESOURCES does not expand or otherwise modify RPL's [Standard Terms](#) including but not limited to the disclaimers and warranties expressed in them.

Document version history

Release	Date	Description
1.0	1 Jul 2021	Initial release
1.1	27 Apr 2022	Copy edit, public release
1.2	3 May 2022	Update to add the Raspberry Pi Compute Module 3 and Raspberry Pi Compute Module 4S
1.3	1 Jul 2022	Update to add verification information
1.4	19 Jun 2023	Add appendix for production line scenarios

Scope of document

This document applies to the following Raspberry Pi products:

Pi 0			Pi 1		Pi 2		Pi 3	Pi 4	Pi 400	CM1	CM3	CM4	Pico
0	W	H	A	B	A	B	B	All	All	All	All	All	All
											*	*	

Introduction

The CM Provisioner is a web application designed to make programming a large number of Raspberry Pi Compute Module (CM) devices much easier and quicker. It is simple to install and simple to use.

It provides an interface to a database of kernel images that can be uploaded, and allows the use of scripts to customise various parts of the installation during the flashing process.

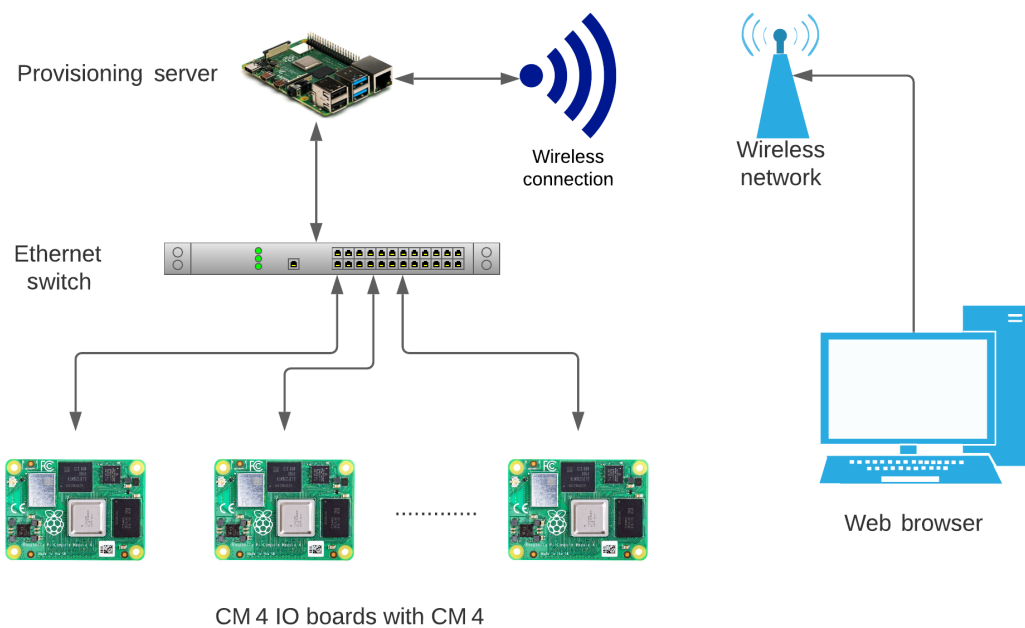
Label printing and firmware updating are also supported.

This white paper assumes that the Provisioner server, software version 1.5 or newer, is running on a Raspberry Pi.

How it all works

Raspberry Pi Compute Module 4

The Provisioner system needs to be installed on its own wired network; the Raspberry Pi running the server is plugged into a switch, along with as many Raspberry Pi Compute Module 4 devices as the switch can support. Any CM4 plugged into this network will be detected by the provisioning system and automatically flashed with the user's required firmware. The reason for having a dedicated wired network becomes clear when you consider that *any* CM4 plugged into this network will be provisioned, so keeping the network separate from any live network is essential to prevent the unintentional reprogramming of devices.



By using a Raspberry Pi as the server, it is possible to use wired networking for the Provisioner but still have access to external networks using wireless connectivity. This allows easy downloading of images to the server, ready for the provisioning process, and allows the Raspberry Pi to serve up the Provisioner web interface. Multiple images can be downloaded; the Provisioner keeps a database of images and makes it easy to select the appropriate image for setting up different devices.

When a Raspberry Pi Compute Module 4 is attached to the network and is powered up it will try to boot, and once other options have been tried, network booting is attempted. At this point the Provisioner Dynamic Host Configuration Protocol (DHCP) system responds to the booting CM4 and provides it with a [minimal bootable image](#) that is downloaded to the CM4 and then run as root. This image can program the embedded Multi-Media Card (eMMC) and run any required scripts, as instructed by the Provisioner.

More details

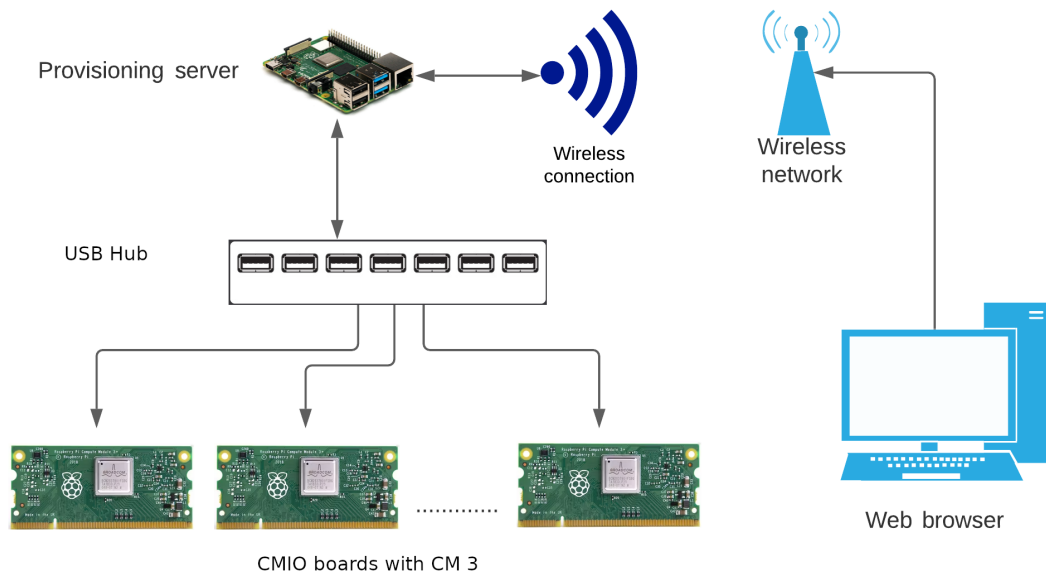
Raspberry Pi Compute Module 4 ships with a boot configuration that will try to boot from eMMC first; if that fails because the eMMC is empty, it will perform a preboot execution environment (PXE) network boot.

So, with Raspberry Pi Compute Module 4 units that have not yet been provisioned, and that have an empty eMMC, a network boot will be performed by default.

During a network boot on a provisioning network, a lightweight utility operating system image (actually a Linux kernel and a `scriptexec` `initramfs`) will be served by the provisioning server to the Raspberry Pi Compute Module 4 over the network, and this image handles the provisioning.

CM3 and CM4S

Compute Module devices based on the SODIMM connector cannot network boot, so programming is achieved over USB. Each device will need to be connected to the Provisioner. If you need to connect more than four devices (the number of USB ports on Raspberry Pi), you can use a USB hub. Use good quality USB A to micro USB cables, connecting from the Raspberry Pi or hub to the USB slave port of each CMIO board. All the CMIO boards will also need a power supply, and the J4 USB slave boot enable jumper should be set to enable (en).



! IMPORTANT

When using the Provisioner to program SODIMM-based Compute Modules over USB, do NOT connect the Ethernet port of the server Raspberry Pi. Its wireless connection is used to access the management web interface.

Installation

The following instructions were correct at the time of issue. The very latest installation instructions can be found on the Provisioner [GitHub page](#).

Installing the Provisioner web application on a Raspberry Pi

⚠ WARNING

Make sure `eth0` connects to an Ethernet switch to which only the Raspberry Pi Compute Module 4 units that are to be provisioned are connected. Do *not* connect `eth0` to your office network or public network, or it may 'provision' other Raspberry Pi devices in your network as well. Use the Raspberry Pi wireless connection to connect to your local network.

The Lite version of Raspberry Pi OS is recommended as the base OS on which to install the Provisioner. For simplicity, use Raspberry Pi Imager, and activate the advanced settings menu (Ctrl-Shift-X) to set up the password, hostname, and wireless settings.

Once the OS is installed on the Raspberry Pi, you will need to set up the Ethernet system:

1. Configure `eth0` to have a static Internet Protocol (IP) address of 172.20.0.1 inside a /16 subnet (netmask 255.255.0.0) by editing the DHCP configuration:

- `sudo nano /etc/dhcpd.conf`
- Add the following to the bottom of the file:

```
interface eth0
static ip_address=172.20.0.1/16
```

- Reboot to allow the changes to take effect.

2. Ensure the OS installation is up to date:

```
sudo apt update
sudo apt full-upgrade
```

3. The Provisioner is supplied as a ready-made `.deb` file on the Provisioner [GitHub page](#). Download the latest version from that page or use `wget`, and install it using the following command:

```
sudo apt install ./cmprovision4*_all.deb
```

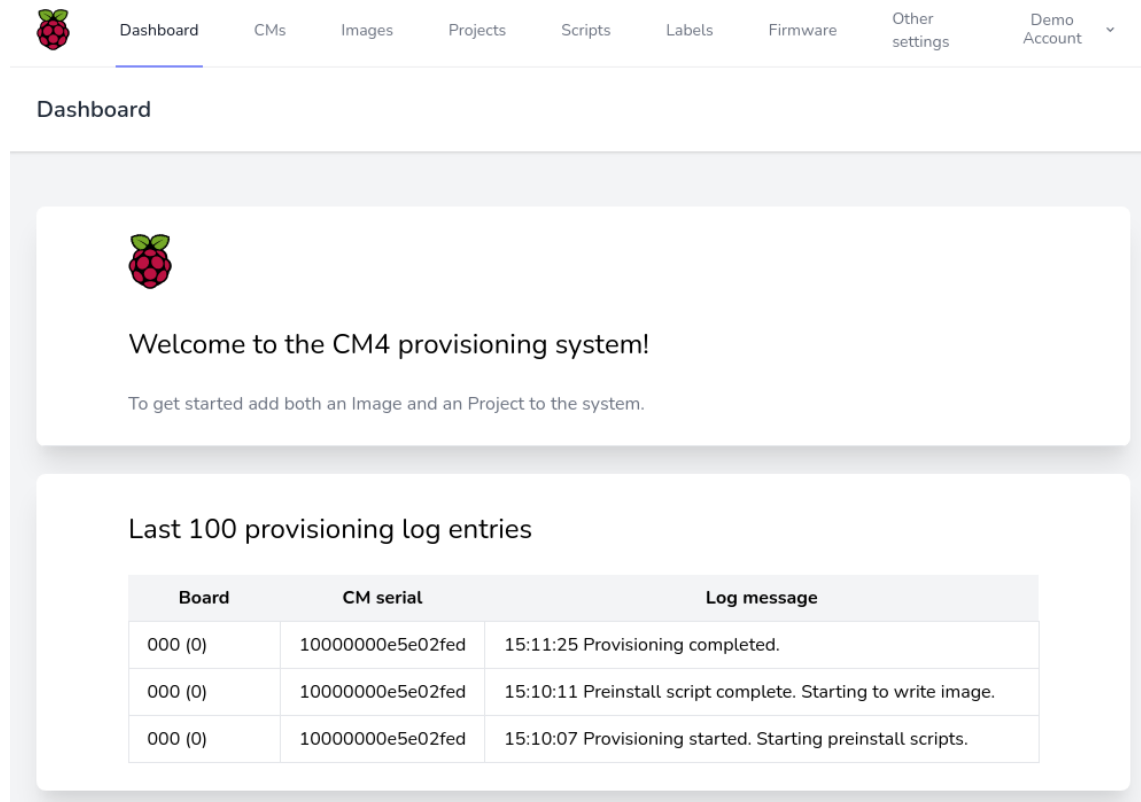
4. Set the web application username and password:

```
sudo /var/lib/cmprovision/artisan auth:create-user
```

You can now access the web interface of the Provisioner from a web browser, using the wireless IP address of the Raspberry Pi and the username and password that you entered in the previous section. Type or paste the IP address into the address bar of your browser and press Enter, and then enter the username and password when prompted.

Usage

When you first connect to the Provisioner web application with your web browser you will see the Dashboard screen, which will look something like this:



Dashboard

Dashboard

CMs Images Projects Scripts Labels Firmware Other settings Demo Account

Welcome to the CM4 provisioning system!

To get started add both an Image and an Project to the system.

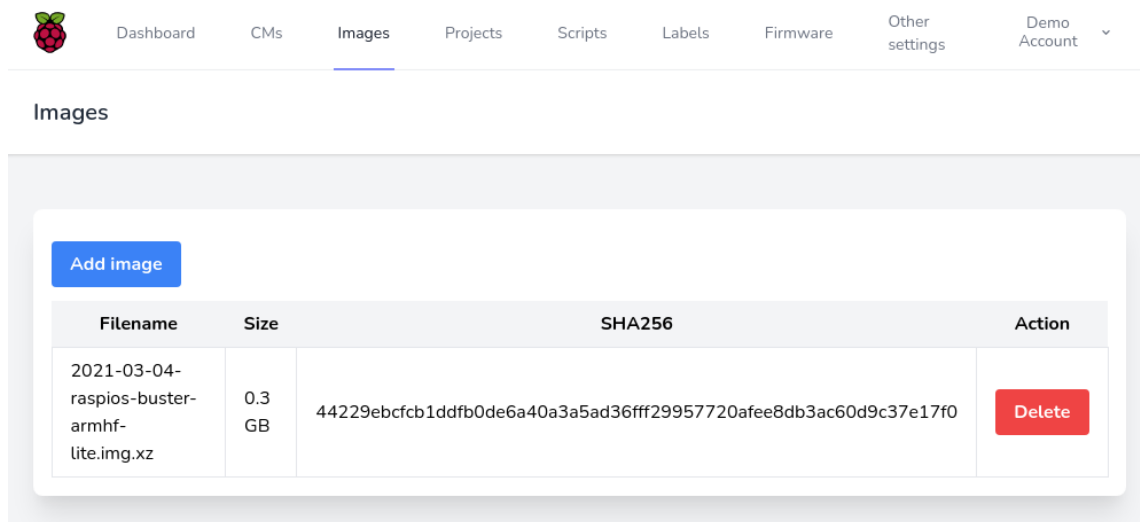
Last 100 provisioning log entries

Board	CM serial	Log message
000 (0)	10000000e5e02fed	15:11:25 Provisioning completed.
000 (0)	10000000e5e02fed	15:10:11 Preinstall script complete. Starting to write image.
000 (0)	10000000e5e02fed	15:10:07 Provisioning started. Starting preinstall scripts.

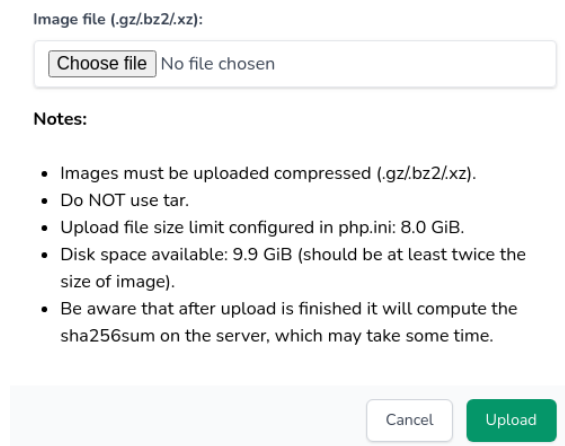
This landing page simply gives some information on the latest action performed by the Provisioner (in the example above, a single Raspberry Pi Compute Module 4 has been provisioned).

Uploading images

The first operation you need to carry out when setting up is to load your image to the server, from where you can use it to provision your Compute Module 4 boards. Click the 'Images' menu item at the top of the web page and you should see a screen similar to the one shown below, displaying a list of currently uploaded images (which will initially be empty).



Select the Add Image button to upload an image; you will see this screen:



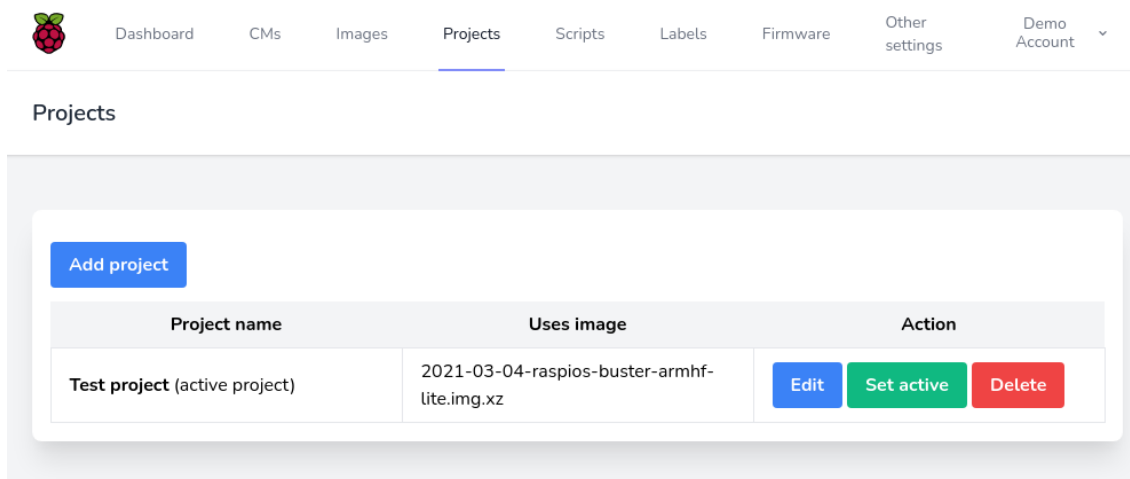
The image needs to be accessible on the device on which the web browser is running, and in one of the image formats specified. Select the image from your machine using the standard file dialogue, and click 'Upload'. This will copy the image from your machine to the Provisioner server running on the Raspberry Pi; this can take some time.

Once your image is uploaded, you will see it on the Images page.

Adding a project

Next you need to create a project. You can specify any number of projects, and each can have a different image, set of scripts, or label. The active project is the one that is currently used for provisioning.

Click on the 'Projects' menu item to bring up the Projects page. The example in the image below already has one project, called 'Test project', set up.



Now click on 'Add project' to set up a new project.

Project name:

Image to write:

Verify that image was written correctly

Destination storage device:

EEPROM firmware update to apply:

EEPROM settings:

When to print label:

Extra scripts to apply:

- Add dtoverlay=dwc2 to config.txt
- Format eMMC as pSLC (one time setttable only)
- Resize ext4 partition

Other options:
 Set as active project

Give the project an appropriate name, then select the image you want this project to use from the drop-down list. You can also set several other parameters at this stage, but often just specifying the image will suffice.

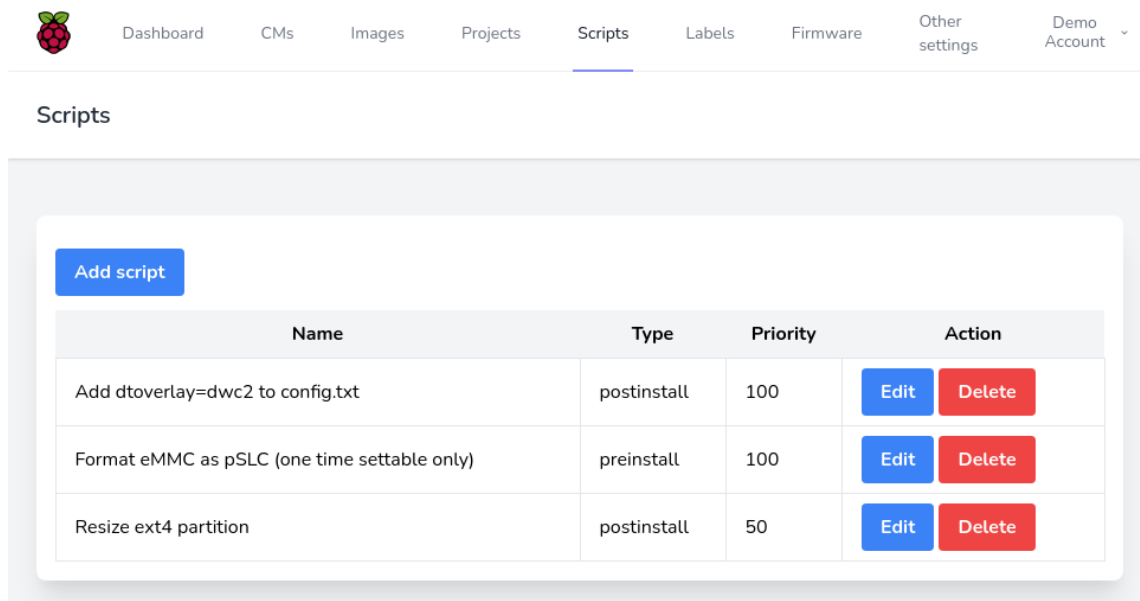
If you are using v1.5 or newer of the Provisioner, then you have the option of verifying that the flashing has been completed correctly. Selecting this will read back the data from the CM device after flashing, and confirm that it matches the original image. This will add extra time to the provisioning of each device; the amount of time added will depend on the size of the image.

If you select the firmware to install (this is optional), you can also customise this firmware with some specific configuration entries that will be merged into the bootloader binary. The available options can be found in the [documentation on the Raspberry Pi website](#).

Click 'Save' when you have finished defining your new project; you will return to the Projects page, and your new project will be listed. Note that only one project can be active at any one time, and you can select it from this list.

Scripts

A really useful feature of the Provisioner is the ability to run scripts on the image, before or after installation. Three scripts are installed by default in the Provisioner and can be selected when creating a new project. They are listed on the Scripts page.



The screenshot shows the 'Scripts' page in the Provisioner interface. At the top, there is a navigation bar with a Raspberry Pi logo and menu items: Dashboard, CMs, Images, Projects, Scripts (highlighted), Labels, Firmware, Other settings, and Demo Account. Below the navigation bar, the page title 'Scripts' is displayed. A blue 'Add script' button is located at the top left of the main content area. Below the button is a table with the following data:

Name	Type	Priority	Action	
Add dtoverlay=dwc2 to config.txt	postinstall	100	Edit	Delete
Format eMMC as pSLC (one time settable only)	preinstall	100	Edit	Delete
Resize ext4 partition	postinstall	50	Edit	Delete

An example use of scripts might be to add custom entries to `config.txt`. The standard script `Add dtoverlay=dwc2 to config.txt` does this, using the following shell code:

```
#!/bin/sh
set -e

mkdir -p /mnt/boot
mount -t vfat $PART1 /mnt/boot
echo "dtoverlay=dwc2,dr_mode=host" >> /mnt/boot/config.txt
umount /mnt/boot
```

Click on 'Add script' to add your own customisations:

Script name:

Script type:

Priority:

Script with lowest number is started first

Run in background

Script:

```
#!/bin/sh
set -e
```

Available environment variables:
\$SERVER \$STORAGE \$PART1 \$PART2

Labels

The Provisioner has the facility to print out labels for the devices being provisioned. The Labels page shows all the predefined labels that can be selected during the project editing process. For example, you may wish to print out DataMatrix or quick response (QR) codes for each board provisioned, and this feature makes this very easy.

Name	Type	Printer FTP hostname	Action
Datamatrix on Brady/CAB printer	ftp	CHANGEME	<input type="button" value="Edit"/> <input type="button" value="Delete"/>

Click on 'Add label' to specify your own:

Label name:

Method to send print job to printer:

Command to queue print job:

File extension for print job:

Label template:

Available variables:
Serial \$mac \$provisionboard

Firmware

The Provisioner provides the ability to specify which version of the bootloader firmware you wish to install on the Raspberry Pi Compute Module 4. On the Firmware page, there is a list of all the possible options, but the most recent one is usually the best.

Dashboard
CMS
Images
Projects
Scripts
Labels
Firmware
Other settings
Demo Account ▾

Firmware

Download new firmware from github

Name	Channel
pieeprom-2021-04-29.bin	stable
pieeprom-2021-03-18.bin	stable
pieeprom-2021-02-16.bin	stable
pieeprom-2021-01-16.bin	stable
pieeprom-2021-01-11.bin	stable
pieeprom-2020-12-11.bin	stable
pieeprom-2020-09-03.bin	stable
pieeprom-2020-07-31.bin	stable

To update the list with the latest versions of the bootloader, click on the 'Download new firmware from GitHub' button.

Possible problems

Out-of-date bootloader firmware

If your Raspberry Pi Compute Module 4 is not detected by the Provisioner system when you plug it in, the bootloader firmware may be out of date. Note that all CM4 devices manufactured since February 2021 have the correct bootloader installed at the factory, so this will only happen with devices that were manufactured before that date.

Already programmed eMMC

If the CM4 already has boot files in the eMMC from a previous provisioning attempt, then it will boot from the eMMC and the network boot required for provisioning will not occur.

If you wish to reprovision a CM4, you will need to:

- Connect a USB cable between the provisioning server and the micro USB port of the Raspberry Pi Compute Module 4 IO Board (labelled 'USB slave').
- Put a jumper on the CM 4 IO Board (J2, 'Fit jumper to disable eMMC boot').

This will cause the Compute Module to perform a USB boot, in which case the provisioning server will transfer the files of the utility OS over USB.

After the utility OS has booted, it will contact the provisioning server over Ethernet to receive further instructions, and download additional files (e.g. the OS image to be written to eMMC) as usual.

Note, therefore, that an Ethernet connection in addition to the USB cable is still necessary.

Spanning Tree Protocol (STP) on managed Ethernet switches

PXE booting will not work correctly if STP is enabled on a managed Ethernet switch. This can be the default on some switches (e.g. Cisco), and if that is the case then STP will need to be disabled for the provisioning process to work correctly.

Production scenarios

How to identify devices after programming

Often, a production line uses the Provisioner to program multiple devices at the same time and prints out labels that need to be appropriately assigned to boards; for example, with unique serial number information. This creates a need for some way of identifying which board requires which label.

Raspberry Pi Compute Module 4 IO board GPIO method

If you are using the Raspberry Pi Compute Module 4 IO board to program devices, then you can use the GPIO pins in a binary fashion on the IO board to define a programming device number. A post-install script can then be used to determine which board has just finished programming, and the label can be printed with, for example, the Raspberry Pi Compute Module 4 IO board designation so a production line operator can apply the label correctly.

Use jumpers at GPIO 5, 13, and 21 to the GND pin opposite those pins (pin 29 to pin 30, pin 33 to pin 34, and pin 39 to pin 40). The binary number you set with the jumpers will set the `$provisionboard` macro in the label template. Note that with three GPIO pins and thus three bits, only eight different programming stations are supported.

USB device method

If you are programming devices in situ in the final product, there may be no GPIO pins available for setting a board identifier. In this case, if USB ports are available, then a set of small-capacity USB memory sticks can be used, each one containing a file with a unique identifier. A USB stick is inserted into each device being programmed, and when programming is complete, a post-programming script reads the identifier; this ID is then used on the label so the operator knows which device to apply the label to.

Managed Ethernet switch

A further possibility is to use a managed Ethernet switch, which can identify which device is plugged into which port. One option is to colour-code the cables from the switch to the IO boards, and indicate on the labels which colour they correspond to.

Resetting a CM4 to factory settings

Once a Raspberry Pi CM4 device has been programmed, it will no longer appear to the Provisioner if you connect it wishing to reprogram it. This is because it now has a bootable system on its eMMC, so the network boot required by the Provisioner is never executed (unless the previous provisioning step has specifically enabled network boot as its primary boot mechanism).

Using USB to program the device has already been described in a previous section. There is also a procedure to reset the device to factory settings, should you wish to do this. The following instructions assume the use of a Raspberry Pi device as the host as well as the standard Raspberry Pi Compute Module 4 IO board. The host should be running Raspberry Pi OS with desktop.

Install the USBBOOT application and associated utilities using the instructions in its GitHub repository; this only needs to be done once: <https://github.com/raspberrypi/usbboot#readme>

Mount the Raspberry Pi CM 4 on the IO board. Connect the IO board to the host Raspberry Pi, and install a link on the header marked "Fit jumper to disable eMMC boot".

Now use `rpiboot` to start the CM4 in Mass Storage Mode.

```
cd usbboot
sudo ./rpiboot
```

Power up the IO board. After a few moments, one or more drives will appear on the host Raspberry Pi. You can display these devices using the following:

```
lsblk
```

Which will produce output similar to the following:

```
NAME            MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
sda              8:0    1 14.6G  0 disk
├─sda1           8:1    1  256M  0 part /media/pi/bootfs
└─sda2           8:2    1   3.6G  0 part /media/pi/rootfs
mmcblk0         179:0    0 14.8G  0 disk
├─mmcblk0p1     179:1    0  256M  0 part /boot
└─mmcblk0p2     179:2    0 14.6G  0 part /
```

You now need to reformat the device that is mounted at `/media/pi/bootfs`, which is `sda1`, as follows:

⚠ WARNING

Make sure you have the right device! This could reformat the SD card in the Raspberry Pi if you get the wrong one.

```
umount /media/pi/bootfs
sudo mkfs -t vfat /dev/sda1
```

For most situations, the Raspberry Pi CM4 should now appear to the Provisioner as a blank device. However, you may wish to format the `/media/pi/rootfs` device as well, to completely clear the system of data.

If the bootloader was changed during the original programming, then one final action is needed to reset that to factory settings:

```
cd usbboot/recovery
./update-pieeprom.sh
../rpiboot -d .
```

Reboot the IO board and the device's bootloader will be replaced with the default.

i NOTE

Because of the way eMMC uses wear levelling, reformatting a device will not necessarily clear all the data. If you have private data on the eMMC you should use more advanced methods to wipe the device.



Raspberry Pi

Raspberry Pi is a trademark of the Raspberry Pi Foundation

Raspberry Pi Ltd