FoundriesFactory® next Documentation

Table of contents:

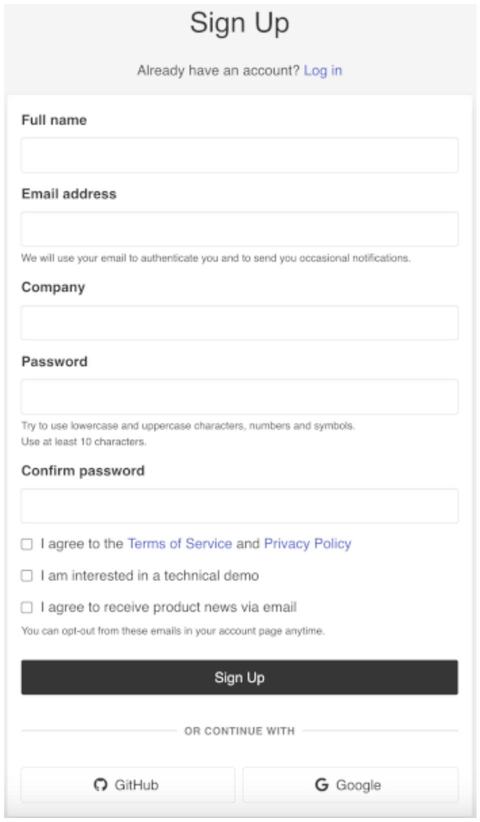
Version: Next

Getting Started

Yocto Version: kirkstone

Signing Up

To begin using FoundriesFactory®, start with creating an account with us.


Sign Up

Already have an account? Log in

Full name

Email address

We will use your email to authenticate you and to send you occasional notifications.

Company

Password

Try to use lowercase and uppercase characters, numbers and symbols.
Use at least 10 characters.

Confirm password

☐ I agree to the Terms of Service and Privacy Policy

☐ I am interested in a technical demo

☐ I agree to receive product news via email

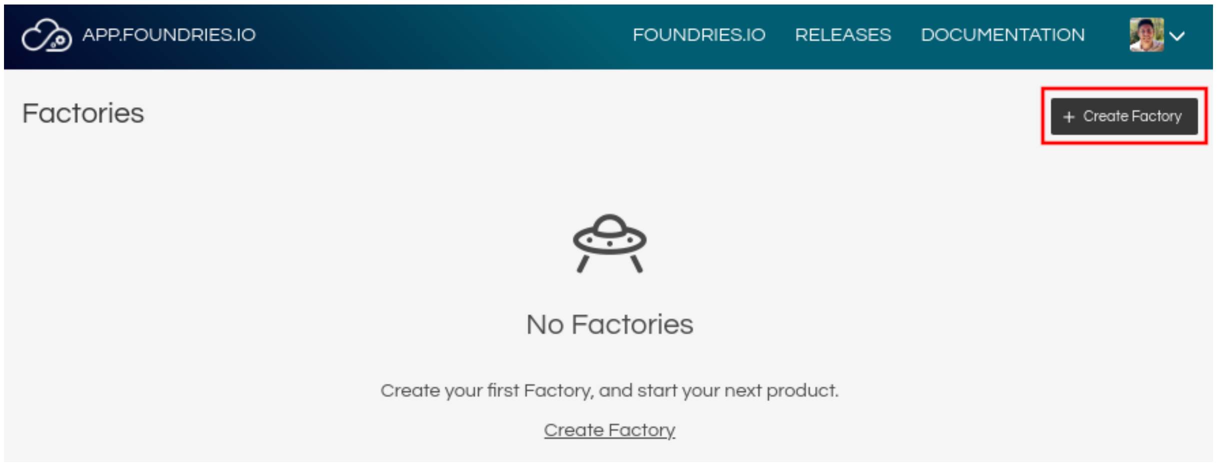
You can opt-out from these emails in your account page anytime.

This is the beginning of your journey.

Creating Your Factory

FoundriesFactory is the start of your embedded OS, tailored specifically for your product. When you create a Factory, we immediately bootstrap the CI build process. This generates an unmodified Linux microPlatform OS Image, which is from this point onward, **owned by you**.

When your account is created, it is not associated with any factories. Create one by clicking `Create Factory`.



Your journey begins empty handed

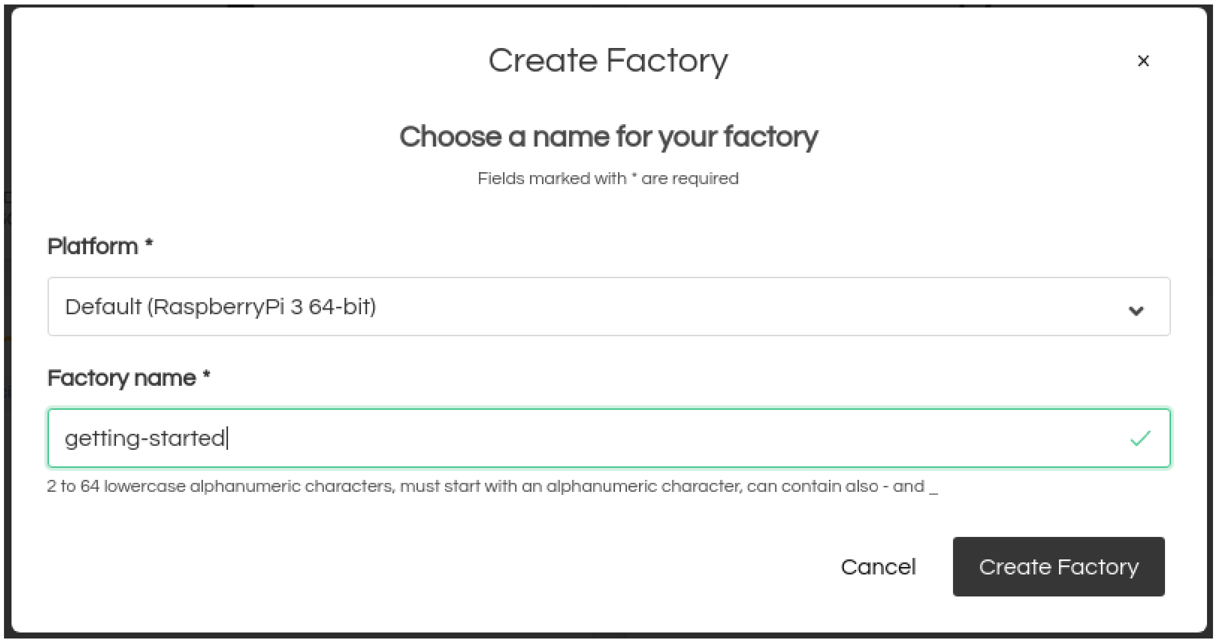
Selecting Your Platform

Choose a hardware platform from the dropdown menu in the **Create Factory** wizard and continue. Click `Create Factory` once your details are entered.

DANGER

Once a Factory is created, the chosen platform/machine and Factory name cannot be changed. Create a new Factory or contact support if a mistake is made.

The Linux MicroPlatform supports a wide range of platforms out of the box. This includes QEMU images for ARM and RISC-V architectures.


Create Factory
×
Choose a name for your factory
Fields marked with * are required
Cancel
Create Factory

Platform *

Default (RaspberryPi 3 64-bit)

Factory name *

getting-started

2 to 64 lowercase alphanumeric characters, must start with an alphanumeric character, can contain also - and _

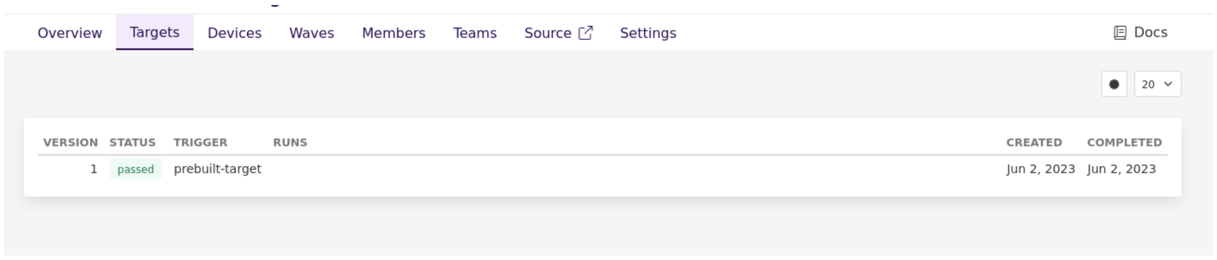
Create Factory

TIP

Your chosen platform determines the value for the `machines:` key for your builds.

Watching Your Build

Once you have created your Factory, the initial artifacts from the Foundries.io™ Linux® microPlatform (LmP) will be generated. This is the base to build your product. You can monitor the progress in the `Targets` tab of your Factory. Additionally, you will receive an email once the Factory initial setup is complete.



VERSION	STATUS	TRIGGER	RUNS	CREATED	COMPLETED
1	passed	prebuilt-target		Jun 2, 2023	Jun 2, 2023

FoundriesFactory Targets

Targets are a reference to a platform image and Docker applications. When developers push code, FoundriesFactory produces a new target. Registered devices then update and install Targets.

NOTE

If you would like to learn more, we wrote a blog about what Targets are and why we made them the way they are.

The `Targets` tab of the Factory will become more useful as you begin to build your application and produce new Targets for the Factory to build.

Tags: signup start first steps

Version: Next

Flashing Your Device

NOTE

The initial FoundriesFactory® set up and build is finished very quick. Follow its status with steps listed in Watching Your Build.

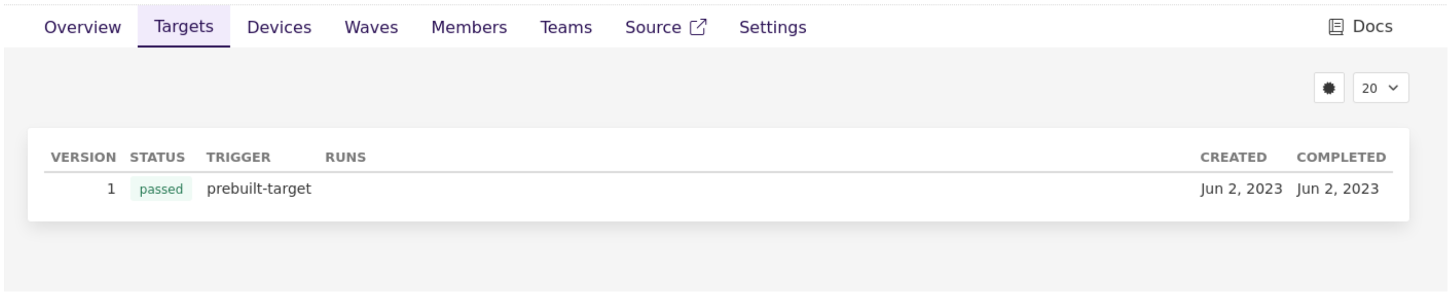
Prerequisites and Pre-Work

- A supported board which is either:
  - Capable of booting from eMMC, **supported by default if available**
  - **Or** capable of booting from a suitable microSD Card
- Wired or WiFi network with internet access.
  - Ethernet cable (if choosing Wired)
  - Console access to your hardware via UART serial (if choosing WiFi)

Downloading the LmP System Image

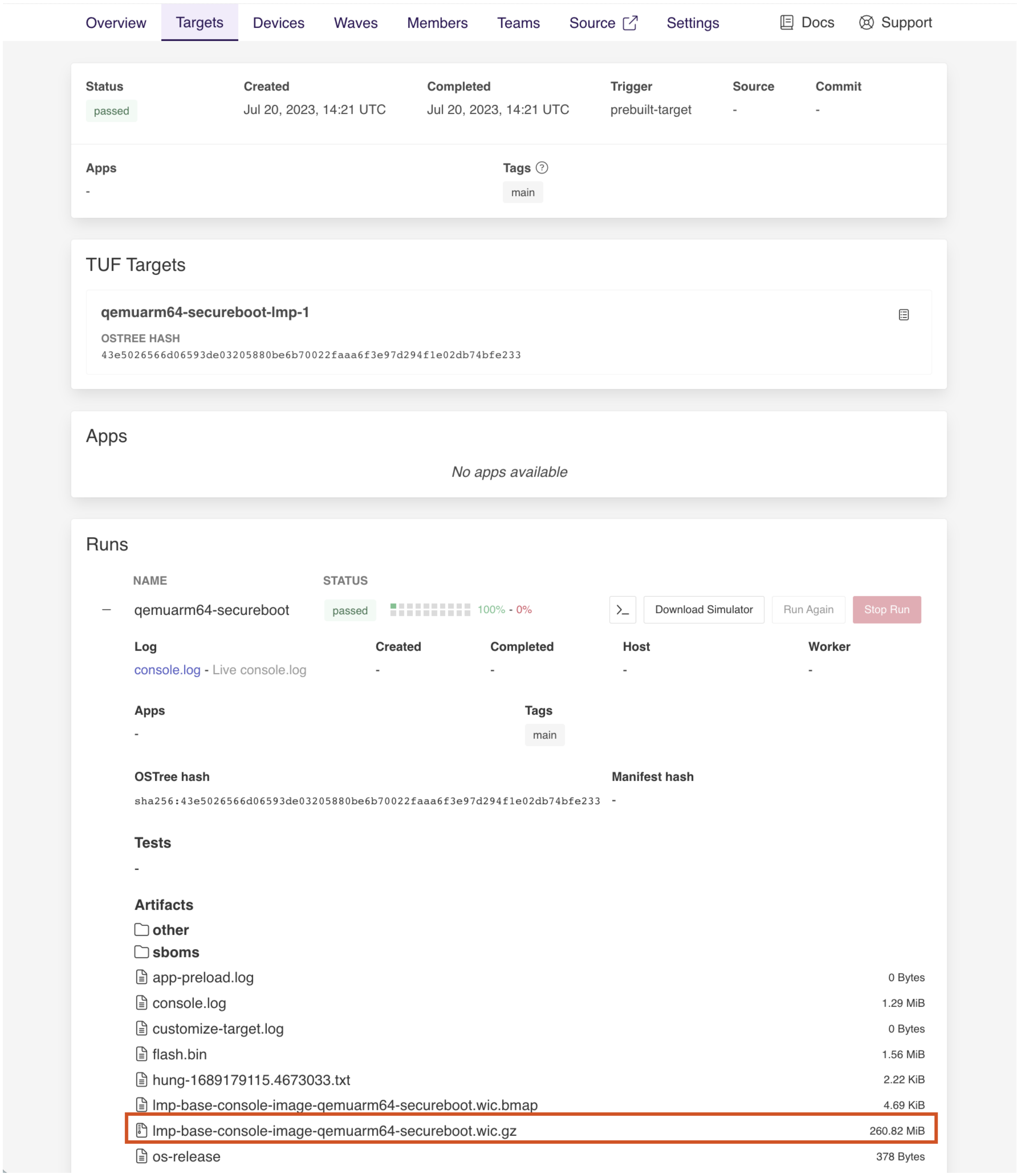
After a successful build, FoundriesFactory produces build artifacts which can be downloaded from the `Targets` tab of your Factory.

1. Navigate to the `Targets` section of your Factory.
2. Click the Target with the `prebuilt-target` `Trigger`.



VERSION	STATUS	TRIGGER	RUNS	CREATED	COMPLETED
1	passed	prebuilt-target		Jun 2, 2023	Jun 2, 2023

3. Expand the `Targets` tab clicking on it. This shows a link to the Factory image artifact. Download the Factory image for your machine, e.g: `lmp-base-console-image-<machine_name>.wic.gz`:



Status	Created	Completed	Trigger	Source	Commit
passed	Jul 20, 2023, 14:21 UTC	Jul 20, 2023, 14:21 UTC	prebuilt-target	-	-

Apps	Tags
-	main

TUF Targets

qemuarm64-secureboot-lmp-1

OSTREE HASH

```
43e5026566d06593de03205880be6b70022faaa6f3e97d294f1e02db74bfe233
```

Apps

No apps available

Runs

NAME	STATUS
qemuarm64-secureboot	passed 100% - 0%

Log	Created	Completed	Host	Worker
console.log - Live console.log	-	-	-	-

Apps	Tags
-	main

OSTree hash	Manifest hash
sha256:43e5026566d06593de03205880be6b70022faaa6f3e97d294f1e02db74bfe233	-

Tests

-

Artifacts

other

sboms

app-preload.log	0 Bytes
console.log	1.29 MiB
customize-target.log	0 Bytes
flash.bin	1.56 MiB
hung-1689179115.4673033.txt	2.22 KiB
lmp-base-console-image-qemuarm64-secureboot.wic.bmap	4.69 KiB
lmp-base-console-image-qemuarm64-secureboot.wic.gz	260.82 MiB
os-release	378 Bytes

NOTE

Most platforms require more than the `lmp-factory-image-<machine_name>.wic.gz` artifact for flashing. The required artifacts are board specific and listed in respective pages under. Targets publish all needed files for each platform under `Runs`.

Flashing the Image

The flashing procedure is board specific and we cover separate steps in `ref-boards`{.interpreted-text role="ref"}. Please refer to this section for specifics on flashing your system image using the vendor provided tools. See `ref-qemu`{.interpreted-text role="ref"} for booting Qemu images.

> **NOTE**
>
> LmP enforces eMMC boot whenever possible as this is the path to enable all security features it provides. So for platforms with available eMMC, such as the NXP® i.MX EVKs, booting from eMMC rather than SD is highly recommended and enabled by default.

Booting and Connecting to the Network

After flashing and booting the board with the respective steps for your hardware, follow these steps to connect to the network.

> **NOTE**
>
> By default, the `username` and `password` to log in your device after boot are `fio`/`fio`. We recommend changing them once you are in development.

Ethernet (Recommended) Wifi

Ethernet works out of the box if a DHCP server is available on the local network. Connect an Ethernet cable to the board. Your board will connect to the network via Ethernet soon after booting.

Logging in via SSH

To login via SSH, run:

```
ssh fio@<machine-name>.local
```

Where `fio` is the username and `<machine-name>` is the hostname of your device. The default password is `fio`.

By default, your device hostname is set to a unique string that specify the platform chosen during Factory creation (`machine`). Check `ref-linux-supported`{.interpreted-text role="ref"} for a list of supported platform and their `machine` values.

::: tip

Here are some examples of default hostnames:

| `raspberrypi4-64.local` | `intel-corei7-64.local` | `imx8mm-lpddr4-evk.local` :::

::: note

For this to work, your PC needs to support zeroconf. The hostname must be unclaimed.

If this does not work, see `Troubleshooting <gs-troubleshooting>`{.interpreted-text role="ref"} below for advice. :::

Troubleshooting

If the above methods to SSH into your board do not work, there are additional things to try.

1. Get the IP address of your device:

    - Temporarily enable and connect to the UART serial (detailed steps for some platforms can be found in `ref-board` and determine available IP addresses with:

    - Ethernet:

      ```
      ip addr show eth0 scope global
      ```

    - WiFi:

      ```
      ip addr show wlan0 scope global
      ```

   - **Or** list the connected devices and their local IP addresses on your network router's administrative interface.

2. Connect to the device by IP address:

   ```
   ssh fio@<ip-address>
   ```

Tags: getting started flashing prebuilt-target

Version: Next

arm

Booting in QEMU

INFO

If you are using a prebuilt Target. your artifacts begin with `lmp-base-console-image` instead.

For arm:

```
└── | arm
├── lmp-factory-image-qemuarm.wic.gz
├── other
│   └── lmp-factory-image-qemuarm.wic.qcow2 # optional
└──u-boot-qemuarm.bin
```

QEMU CLI

```
 qemu-system-arm -machine virt,highmem=off -cpu cortex-a7 -m 1024M \ -bios u-boot-qemuarm.bin \ -serial mon:vc -serial null \ -drive id=disk0,file=lmp-factory-image-qemuarm.wic,if=none,format=raw -device virtio-blk-device,drive=disk0 \ -object rng-random,filename=/dev/urandom,id=rng0 -device virtio-rng-pci,rng=rng0 \ -device virtio-net-device,netdev=usernet \ -netdev user,id=usernet,hostfwd=tcp::22222-:22 \ -no-acpi -d unimp -nographic
```

TIP

You can register your device by following the steps from the Getting Started Guide.

Version: Next

x86_64

Booting in QEMU

INFO

If you are using a prebuilt Target. your artifacts begin with `lmp-base-console-image` instead.

For x86_64:

```
└── | x86_64
├── lmp-factory-image-intel-corei7-64.wic.gz
├── other
│   └── lmp-factory-image-intel-corei7-64.wic.qcow2 # optional
└──ovmf.secboot.qcow2
```

QEMU CLI

```
qemu-system-x86_64 -m 1024 -cpu kvm64 -enable-kvm -serial mon:stdio -serial null \
-drive file=lmp-factory-image-intel-corei7-64.wic.qcow2,format=qcow2,if=none,id=hd
\ -device virtio-scsi-pci,id=scsi -device scsi-hd,drive=hd -device virtio-rng-pci
\ -drive if=pflash,format=qcow2,file=ovmf.secboot.qcow2 \ -net
user,hostfwd=tcp::22223-:22 -net nic -nographic
```

TIP

You can register your device by following the steps from the Getting Started Guide.