Firefox OS uses Android kernel and HALs, with a Gecko-based user interface on top. This article is a guide to port a device supported by CyanogenMod to Firefox OS.

Prerequisites

- An Android device supported in CyanogenMod 12.1
- A computer with Ubuntu 14.04 LTS installed

Porting is possible on OSX and for devices *not* currently supported by CyanogenMod, but any specifics for these situations will not be covered by this guide.

Set up your Ubuntu 14.04 LTS build system

Since both Firefox OS and CyanogenMod use an AOSP base, it is (almost) enough to follow the guide for AOSP provided by Google.

For Firefox OS we only need to install a couple of extra tools:

```
$ sudo apt-get install autoconf2.13 lzop
```

We also need to correctly communicate with the device over USB. We recommend to install UDEV rules according to the CyanogenMod wiki.

Set up your OS X 10 build system

WIP

1. Install Ubuntu 14.04 LTS as a virtual machine.
2. Follow steps above.

~~Install OS X 11 on a case sensitive file system~~
~~Install Xcode 7.1.1 from application store~~
~~Install Xcode 5.1.1 from developer site~~
~~Install Homebrew via http://brew.sh/~~
~~brew install: ccache, cmake, gnu-sed, gnu-tar, gpg, yasm,~~
~~https://raw.github.com/Homebrew/homebrew-versions/master/autoconf213.rb~~

WIP

Download the source code

TODO: Merge needed manifest: Bugzilla: 1211870
Manifests mentioned below can be found on github.com/cm-b2g

We have several useful tools for building Firefox OS, all contained in a single repository. Download these tools via git to create your working directory.

```
$ git clone https://github.com/mozilla-b2g/B2G.git && cd B2G
```

Now we need to download the source code.

```
$ ./config.sh cm-porting
```

The `config.sh` initializes the `repo` tool using the `base-l-cm.xml` manifest found in the `b2g-manifest` repository. This XML file is a list of repositories needed to build Firefox OS.

Downloading all of these repositories, many of which are several gigabytes, will take a while so we recommend doing this overnight on a slow connection, or just before lunch on a faster connection.

This step also creates a `.config` file which you will edit later.

Local manifest for your device

You will need to create a `local_manifest.xml` with all of the repositories for your device. The quickest way to do this is to use `breakfast`, an automated tool written by CyanogenMod to create a local manifest and download the additional repositories directly from CyanogenMod's GitHub account.

```
$ . build/envsetup.sh && breakfast 123
```

Replace `123` *with your device's codename.*

If your device is not officially supported by CyanogenMod, but there is already an unofficial port, you can create the `local_manifest.xml` manually in the `.repo/local_manifests` folder.

```
<?xml version="1.0" encoding="UTF-8"?>
<manifest>
<remote name="xyz" fetch="git://github.com/xyz/" revision="cm-12.1" />
<project name="device_oem_123" path="device/oem/123" remote="xyz" />
<project name="device_oem_1xx" path="device/oem/1xx" remote="xyz" />
<project name="kernel_oem_1xx" path="kernel/oem/1xx" remote="xyz" />
<project name="vendor_oem" path="vendor/oem" remote="xyz" />
</manifest>
```

Remember to run `repo sync` when you have created your manifest!

Prepare your device

If your device has already been modified to run CyanogenMod then great! you can jump to the next section.

Unlock or root your device

Android devices (rightly) come with several security measures to prevent arbitrary code from running on your device. This is to our benefit as we don't want bad actors stealing our banking passwords!

However we need to disable these security measures to install Firefox OS. Some device OEMs provide an official method to unlock your device (Thank you Sony!) but unfortunately for other devices we will need to use a security exploit.

If you are looking to purchase a device specifically for porting then we strongly suggest you choose one that has an official unlock method.

TODO: List of unlock and rooting methods

If you wish to port a device you already own, then how you exploit your device will vary depending on the Android version and the device OEM The best place for this information is the CyanogenMod wiki.

Install Recovery

TODO: Where to get prebuilt recovery

We also need to install a custom recovery so we can quickly install Firefox OS and update it. The CyanogenMod wiki will provide details on how to do this for your device.

Recovery is needed to install OTA updates, so don't skip this step!

Extract blobs from CyanogenMod 12.1

There are certain proprietary blobs that Mozilla cannot distribute, so we need to extract them from a running device. Fortunately all CyanogenMod device repositories include an `extract-files.sh` script to do this.

However we cannot extract these blobs from a device's stock Android because CyanogenMod maintainers also use blobs from other sources. These blobs will have better compatibility with newer versions of Android than the blobs originally on the device. This also means we must install CyanogenMod to extract the blobs.

If your device is officially supported then download and install the latest stable snapshot for your device. If your device is unofficially supported then you can download the latest build directly from the device maintainer.

TODO: Where to find unofficial ports

Installing is as simple as booting the device into recovery and choosing ADB Sideload from the update menu then sideloading CyanogenMod to the device:

```
& adb sideload cm-12.1-20151111-SNAPSHOT-YOG7D-123.zip
```

When finished reboot into the device, enable ADB debugging in developer settings and extract the blobs from the device:

```
$ cd device/xyz/123 && ./extract-files.sh
```

Modify your repos to support Firefox OS

yep.

Device additions for building

A CyanogenMod repository does not need all configuration files as needed by a device supported on AOSP. Specifically there may be missing `AndroidProducts.mk` and `vendorsetup.sh`. These are simple files that the AOSP build system uses to find valid build targets.

Device additions for Firefox OS

Part of a device's configuration is found in XML overlay files used by Android apps. Firefox OS does not use these so we need to reimplement some of these options, such as on screen home button, emulated storage etc.

Below is a list of common additions and deletions.

```
# for Gecko to use the correct resolution assets
# Valid options are: 1.5 | 2 | 2.25
GAIA_DEV_PIXELS_PER_PX := 2.25

# for Gecko to use the correct boot animation
# Valid options are: hvga | fwvga | qHD | 720p | 1080p
BOOTANIMATION_ASSET_SIZE := 1080p

# for Gecko to support separate internal storage partition
# This is for legacy devices only
GECKO_BOARD_SEPARATE_STORAGE_PARTITON := true

# for Gonk to support Camera blobs
PRODUCT_PACKAGES += \
    libandroid

# for Gecko to support usb mass storage
# You may need to add mass_storage to init.oem.usb.rc
PRODUCT_DEFAULT_PROPERTY_OVERRIDES += \
    persist.sys.usb.config=mtp
    persist.sys.usb.config=mass_storage

# for Gecko to support NFC
PRODUCT_PROPERTY_OVERRIDES += \
    ro.moz.nfc.enabled=true
```

```
PRODUCT_PACKAGES += \
    nfcd

# for Gecko to support virtual home button
PRODUCT_PROPERTY_OVERRIDES += \
    ro.moz.has_home_button=0

# Extra mk import in BoardConfig.mk (many useful defaults)
include vendor/cm/BoardConfig.mk

# Extra mk import in BoardConfig.mk (Needed for Recovery)
include vendor/cm/sepolicy/sepolicy.mk

# Extra mk import in device.mk
$(call inherit-product-if-exists, vendor/cm/config/common_full.mk)

# Changes in init.device.rc
on property:init.svc.bootanim=running
on property:init.svc.bootanim=stopped
on property:service.bootanim.exit=1
on property:sys.boot_completed=1
```

Kernel additions for Firefox OS

We need to enable some extra security features in the kernel for Firefox OS. For many older kernels this will need some extra patches backported to the device.

https://bugzilla.mozilla.org/show_bug.cgi?id=790923

Find which kernel defconfig your devices uses:

```
$ grep -r TARGET_KERNEL_CONFIG device/oem/123
```

Add the following to the defconfig, found in `arch/arm/configs`:

```
CONFIG_SECCOMP=y
CONFIG_SECCOMP_FILTER=y
CONFIG_SECCOMP_DEBUG=n
CONFIG_FANOTIFY=y
```

Depending on the CyanogenMod maintainer there may be a minimized defconfig (good!) or a full defconfig (bad!). if you have a minimized defconfig it is enough to add these to the bottom of the manifest. If you have a full defconfig then you will need to search and replace the values.

Build Firefox OS!

Remember the `.config` file created earlier? Now we need to replace `cm-porting` with your device codename.

```
$ grep -r PRODUCT_NAME device/oem/123
```

Note: *don't* use the value in `cm.mk`, use the one in `device.mk`.

- `cm_123` ✗
- `full_123` ✓

You can do the replacement manually, or simply with `sed`:

```
$ sed -i "s/cm-porting/full_123/g" .config
```

Now it's time to kick off the build!

```
$ ./build.sh
```

This will take an hour or two depending on your PC, so now might be a good time to pop to the shop and buy some popcorn.

Install Firefox OS via fastboot

If your device supports fastboot then you can simply flash the partition images directly:

```
$ cd out/target/product/123/
$ fastboot flash boot boot.img
$ fastboot flash recovery recovery.img
$ fastboot flash system system.img
$ fastboot flash userdata userdata.img
```

Install Firefox OS via recovery

If you device does not support fastboot then you need to build an update.zip which can be installed on the device via sideload.

```
$ export B2G_FOTA_FULLIMG_PARTS="/boot:boot.img /system:system.img" && ./build.sh gecko-update-fota-fullimg
```

Enter recovery according to your device specific method then:

```
$ adb sideload out/target/product/123/fota/fullimg/update.zip
```

Success!

Troubleshooting

Is something not working? It's time to roll up your sleeves!

First try to determine if the feature is working in CyanogenMod. It may simply be a missing configuration for Firefox OS.

If the feature is *not* working on CyanogenMod then it means you need to implement that feature to your port. It would be nice if you push your fix back upstream too!

Camera

Camera blobs will probably need libandroid.so, This was solved for shinano-kk port (codeaurora.org) but needed adapting for LG camera. If your camera blob complains about missing functions, check what was removed.

Bluetooth

https://bugzilla.mozilla.org/show_bug.cgi?id=1212943

gist.github.com/silvio

Gecko

~~CyanogenMod has extra metadata to decide offloading: gist.github.com/AdFad666~~

NFC

For now we recommend using the sony-aosp-l branch here: platform_external_libnfc-nci
Use this patch if you are missing a kernel header: gist.github.com/AdFad666

Sensors

Sensors not loading correctly? AOSP and CM(CAF) need different sec_config files due to differences in how they handle the defined permissions.

WiFi

The device may not see certain access points if they are on an unofficial channel. This seems to be a CyanogenMod or even AOSP problem as it affects different hardware and different OEMs.