Sailfish OS Hardware Adaptation Development Kit Documentation

Release 1.0.2-EA2

Jolla Ltd.

July 21, 2014

This is a guide to help you understand how you can port Sailfish OS to devices running the CyanogenMod flavour of Android.

Warning: Modifying or replacing your device's software may void your device's warranty, lead to data loss, hair loss, financial loss, privacy loss, security breaches, or other damage, and therefore must be done entirely at your own risk. No one affiliated with this project is responsible for your actions but yourself. Good luck.

CHAPTER

ONE

OVERVIEW

1.1 Goal

By following this guide you can set up a Mer-core based Linux system that will run on an Android device, on top of the existing Android Hardware Adaptation kernel and drivers.

This consists of:

- **Mer core**: The Linux userspace core
- **Android Hardware Adaptation** (HA/HAL), consisting of:
  - Device-specific **Android Kernel**
  - **Binary device drivers** taken from an Android ROM (e.g. CyanogenMod)
  - The **libhybris interface** built against the binary drivers
  - **Middleware packages** depending on hardware-specific plugins
  - A Qt/Wayland **QPA plugin** utilizing the Android `hwcomposer`
- **Sailfish OS** components

1.2 Development

1.2.1 Requirements

The development environment uses the Mer Platform SDK, with:

- one or more device specific **targets** (a rootfs with device-specific headers and libraries)
- a HA build SDK (a minimal Ubuntu chroot required to build the Android sources)

During the HA development you'll typically have one window/terminal using the HA build SDK where you build and work on Android code and another session using the Mer SDK where you build RPMs for the hardware adaptation.

Setting up the Mer Platform SDK, as well as the device-specific targets and the Ubuntu HA build chroot is described in *Setting up the SDKs*.

Commands and output from the Mer SDK session are indicated using `MER_SDK $` at the top of the code block, like this:

```
MER_SDK $

echo "run this command in the Mer SDK terminal"
```

How to enter `MER_SDK $` is explained in *Setup the Mer SDK*.

Commands and output from the HA build session are indicated using `HABUILD_SDK $` at the top of the code block, like this:

```
HABUILD_SDK $

echo "run this command in the Ubuntu HA build SDK terminal"
```

How to enter `HABUILD_SDK $` is explained in *Entering Ubuntu Chroot*.

1.2.2 The build area root directory

In this guide, we refer to the base of the SDK storage/build area with the environment variable `$MER_ROOT`. You need several gigabytes of space in this area, we suggest the following paths:

- `export MER_ROOT=/srv/mer/` for a system-wide installation
- `export MER_ROOT=$HOME/mer/` for a user-specific installation

1.2.3 Build components

There are a number of components to build; the lower level and Android related components are built in the HA build SDK; the rest are built in the Mer SDK.

- In the **HA build SDK**
  - a kernel
  - a hacking friendly initrd which supports various boot options
  - `hybris-boot.img` and `hybris-recovery.img` (for booting and debugging)
  - a minimal Android `/system/` tree
  - modified Android parts for compatibility with libhybris and Sailfish OS (e.g. Bionic libc, `logcat`, `init`, ...)
- In the **Mer SDK**
  - RPM packages containing all the built binaries and extracted configs
  - Hardware-specific middleware and plugins (e.g. Qt QPA plugins, PulseAudio)

For distribution, RPM packages are uploaded to a HA-specific repository. With this repository, full system images using the `mic` utility. The `mic` utility is usually also run inside the Mer SDK.

1.3 Deployment

The `hybris-boot.img` (containing both the kernel and our custom initrd) is flashed to the device, while the Sailfish OS rootfs is placed in a subdirectory of the `/data/` partition alongside an existing, unmodified Android system.

The Sailfish OS rootfs is then used as a switchroot target with /data bind-mounted inside it for shared access to any user data.

CHAPTER

TWO

PREREQUISITES

2.1 Mobile Device

- An ARMv7 Android device officially supported by CyanogenMod 10.1.x
  - See http://wiki.cyanogenmod.org/w/Devices for a list of compatible devices
  - See *List of Supported Devices* for a list of devices already supported by HADK
- Means to do backup and restore of the device contents (e.g. SD card or USB cable to host computer), as well as flash recovery images to the device

2.2 Build Machine

- A 64-bit X86 machine with a 64-bit Linux kernel
- Mer Platform SDK
- Sailfish OS Target
- At least 16 GiB of free disk space (10 GiB source download + more for building) for a complete Android build; a minimal download and HADK build (only hardware adaptation-related components) requires slightly less space
- At least 4 GiB of RAM (the more the better)

2.3 Want to Port to a New Device?

If you cannot find your gadget among the *List of Supported Devices*, then you should first read through the entire guide to get a feeling for the order in which things are typically done. Then scrupulously follow *Porting Sailfish OS to a New Device*, clicking on all referenced sections (or even whole chapters!) as you go, and backtracking to where you left off when each section/chapter is finished.

So we kindly ask our pioneer porters of new devices to be patient and ensure they use sophisticated PDF readers, making full use of the back/forward ability ;)

CHAPTER

THREE

PREPARING YOUR DEVICE

Verify that you can backup and restore your device and that you understand device recovery options. This is not only useful when flashing images you build with this guide, but also in case you want to reset your device to its factory state with stock Android (note that not all Android vendors provide factory images for download, so you might need to create a full backup of your running Android system and store it in a safe place before starting to erase and reflash the device with your custom builds).

3.1 Backup and Verify Your Device

As mentioned above, it might be helpful to backup the stock image before flashing the CM release for the first time, as getting the stock image might be hard for some vendors (e.g. some stock images are only available as self-extracting .exe package for Windows) or impossible (some vendors do not provide stock images for download).

Use an Android/CyanogenMod Recovery to:

1. Backup to SD card: system, data, boot and recovery partitions
2. Test restoring the backup (important)

Warning: While backing up to internal device storage is possible for some devices, if during porting you end up overwriting that partition, your backups will be gone. In that case (and in case of devices without SD card slots), it's better to also copy the backup data to your development machine (e.g. via `adb pull` in recovery). Recent versions of `adb` support full-device backups to a host computer using the `adb backup` feature.

See the ClockworkMod Instructions for additional help.

3.2 Flash and Test CyanogenMod

The official CyanogenMod flashing instructions can be found on this CyanogenMod wiki page.

You may also want to verify that the CM build for your device is fully functional, to avoid wasting time with hardware adaptations that have known issues. Also, your device might have some hardware defects - testing in Android verifies that all components are working correctly, so you have a functionality baseline to compare your build results with.

You should at least check the following features:

- **OpenGL ES 2.0**: Use e.g. Gears4Android to test (the hz you will get there will be max refresh rate).

- **WLAN connectivity**: Connect to an AP, ad-hoc or set up a mobile access point with your device.
- **Audio**: Headset detection, earpiece speaker, loudspeakers, etc.
- **Bluetooth**: Connect to bluetooth headsets, verify discoverability, send files.
- **NFC**: Check if NFC tags can be detected, read and/or written by the device.
- **SD/MicroSD**: Use a file manager app to see if inserted SD cards can be detected.
- **USB**: MTP, mass storage (if available) and `adb` access.
- **Telephony**: 2G/3G/LTE calls + data connectivity.
- **GPS**: Using GPSTest, check GLONASS too; typical time to fix; AGPS.
- **Sensors**: Using AndroSensor: Accelerometer, Proximity Sensor, Ambient Light Sensor, Gyroscope, Magnetometer (Compass).
- **LEDs**: If your device has notification LEDs or keypad backlights.
- **Camera** (front and back): Also test functionality of zoom, flash, etc..
- **Buttons**: Volume up, volume down, power, camera shutter, etc..
- **Video out**: HDMI / MHL connectivity if you have the necessary adapters. TV out.
- **Screen backlight**: Suspend and backlight control, minimum and maximum brightness.
- **Battery meter**: Charge level, battery health, charging via USB (wall charger and host PC).
- **Vibration motor**: Intensity, patterns.
- **HW composer version**: check `dumpsys surfaceflinger` through ADB (see SF Layer Debugging).

We recommend that you write down the results of these tests, so you can always remember them.

CHAPTER

FOUR

SETTING UP THE SDKS

4.1 Setting up required environment variables

Throughout this guide we will be referencing the location of your SDK, targets and source code. As is customary with Android hardware adaptations, the device vendor (`$VENDOR`) and device codename (`$DEVICE`) are also used, both in scripts and configuration files. For a list of vendor and device names, refer to *List of Supported Devices*.

Now run the following commands on your host operating system substituting the obtained information where indicated with `[]` (MER_ROOT value from *The build area root directory*):

```
HOST $

cat <<'EOF' > $HOME/.hadk.env
export MER_ROOT="[insert value of your choosing]"
export ANDROID_ROOT="$MER_ROOT/android/droid"
export VENDOR="[insert vendor name here]"
export DEVICE="[insert device codename here]"
EOF

cat <<'EOF' >> $HOME/.mersdkubu.profile
function hadk() { source $HOME/.hadk.env${1:+.$1}; echo "Env setup for $DEVICE"; }
export PS1="HABUILD_SDK [\${DEVICE}] $PS1"
hadk
EOF

cat <<'EOF' >> $HOME/.mersdk.profile
function hadk() { source $HOME/.hadk.env${1:+.$1}; echo "Env setup for $DEVICE"; }
hadk
EOF
```

This ensures that the environment is setup correctly when you use the `ubu-chroot` command to enter the Android SDK.

It also creates a function `hadk` that you can use to set or reset the environment variables. As you can see it also supports ~/.hadk.env.<name> to allow you to work on multiple devices in different sessions.

4.2 Setup the Mer SDK

The Mer SDK setup is described on the Mer wiki.

Ensure you are able to open a shell in the Mer SDK before moving on.

4.3 Preparing the Mer SDK

You'll need some tools which are not installed into the Mer SDK by default:

- **android-tools** contains tools and utilities needed for working with the Android SDK
- **createrepo** is needed to build repositories locally if you want to create or update local RPM repositories
- **zip** is needed to pack the final updater package into an .zip file

The latest SDK tarballs should include these but if not you can install those tools with the following command:

```
MER_SDK $

sudo zypper in android-tools createrepo zip
```

4.4 Setting up an Android Build Environment

4.4.1 Downloading and Unpacking Ubuntu Chroot

In order to maintain build stability, we use an *Ubuntu GNU/Linux* `chroot` environment from within the Mer SDK to build our Android source tree. The following commands download and unpack the rootfs to the appropriate location:

```
MER_SDK $

hadk

TARBALL=ubuntu-trusty-android-rootfs.tar.bz2
curl -O http://img.merproject.org/images/mer-hybris/ubu/$TARBALL
UBUNTU_CHROOT=/parentroot/$MER_ROOT/sdks/ubuntu
sudo mkdir -p $UBUNTU_CHROOT
sudo tar --numeric-owner -xvjf $TARBALL -C $UBUNTU_CHROOT
```

4.4.2 Entering Ubuntu Chroot

```
MER_SDK $

hadk

ubu-chroot -r /parentroot/$MER_ROOT/sdks/ubuntu

#FIXME: Hostname resolution might fail. This error can be ignored.
Can be fixed manually by adding the hostname to /etc/hosts
```


CHAPTER

FIVE

BUILDING THE ANDROID HAL

5.1 Checking out CyanogenMod Source

Our build process is based around the CyanogenMod projects source tree, but when required we've modified some projects, in order to apply patches required to make libhybris function correctly, and to minimise the built-in actions and services in the `init.*.rc` files.

Ensure you have setup your name and e-mail address in your Git configuration:

```
MER_SDK $

git config --global user.name "Your Name"
git config --global user.email "you@example.com"
```

You also need to install the `repo` command from the AOSP source code repositories, see Installing repo.

After you've installed the `repo` command, the following set of commands download the required projects for building the modified parts of Android used in libhybris-based Mer device hardware adaptations.

```
HABUILD_SDK $

hadk

sudo mkdir -p $ANDROID_ROOT
sudo chown -R $USER $ANDROID_ROOT
cd $ANDROID_ROOT
repo init -u git://github.com/mer-hybris/android.git -b hybris-10.1
repo sync
```

The expected disk usage for the source tree after `repo sync` is **9.4 GB** (as of 2014-02-18). Depending on your connection, this might take some time. In the mean time, make yourself familiar with the rest of this guide.

5.2 Building Relevant Bits of CyanogenMod

In the Android build tree, run the following in a `bash` shell (if you are using e.g. `zsh`, you need to run these commands in a `bash` shell, as the Android build scripts are assuming you are running `bash`):

```
HABUILD_SDK $
```

```
hadk

source build/envsetup.sh
export USE_CCACHE=1

breakfast $DEVICE

rm .repo/local_manifests/roomservice.xml
```

The last command removes the CyanogenMod "roomservice" repository list, which contains any additional device-specific repositories you need. In our case, the `hybris-10.1` manifest file already contains device-specific repositories, and the repositories added by roomservice would conflict with those.

```
HABUILD_SDK $

make -j4 hybris-hal
```

The relevant output bits will be in `out/target/product/$DEVICE/`, in particular:

- `hybris-boot.img`: Kernel and initrd
- `hybris-recovery.img`: Recovery boot image
- `system/` and `root/`: HAL system libraries and binaries

The expected disk usage for the source and binaries after `make hybris-hal` is **16 GB** (as of 2014-02-18).

5.2.1 For Supported Devices

See *List of Supported Devices* for a list of devices supported by HADK. Supported devices are automatically downloaded as part of the HADK android build environment.

5.3 Common Pitfalls

- If `repo sync` fails with a message like *fatal: duplicate path device/samsung/smdk4412-common in /home/nemo/android/.repo/manifest.xml*, remove the local manifest with `rm .repo/local_manifests/roomservice.xml`
- If you notice `git clone` commands starting to write out *"Forbidden ..."* on github repos, you might have hit API rate limit. To solve this, put your github credentials into `~/.netrc`. More info can be found following this link: Perm.auth. with Git repositories
- In some cases (with parallel builds), the build can fail, in this case, use `make -j1 hybris-hal` to retry with a non-parallel build and see the error message without output from parallel jobs. The build usually ends with the following output:

```
HABUILD_SDK $

...
Install: .../out/target/product/$DEVICE/hybris-recovery.img
...
Install: .../out/target/product/$DEVICE/hybris-boot.img
```



```
...
Made boot image: .../out/target/product/$DEVICE/boot.img
```


CHAPTER

SIX

SETTING UP SCRATCHBOX2 TARGET

It is necessary to setup a Scratchbox2 target to use for packaging your hardware adaptation packages in the next section. Download and create your Scratchbox2 target with the following commands:

```
MERSDK $

hadk

cd $HOME

SFFE_SB2_TARGET=/parentroot/$MER_ROOT/targets/$VENDOR-$DEVICE-armv7hl
TARBALL_URL=http://releases.sailfishos.org/sdk/latest/targets/targets.json
TARBALL=$(curl $TARBALL_URL | grep 'armv7hl.tar.bz2' | cut -d\" -f4)
curl -O $TARBALL

sudo mkdir -p $SFFE_SB2_TARGET
sudo tar --numeric-owner -pxjf $(basename $TARBALL) -C $SFFE_SB2_TARGET

sudo chown -R $USER $SFFE_SB2_TARGET

cd $SFFE_SB2_TARGET
grep :$(id -u): /etc/passwd >> etc/passwd
grep :$(id -g): /etc/group >> etc/group

sb2-init -d -L "--sysroot=/" -C "--sysroot=/" \
         -c /usr/bin/qemu-arm-dynamic -m sdk-build \
         -n -N -t / $VENDOR-$DEVICE-armv7hl \
         /opt/cross/bin/armv7hl-meego-linux-gnueabi-gcc

sb2 -t $VENDOR-$DEVICE-armv7hl -m sdk-install -R rpm --rebuilddb

sb2 -t $VENDOR-$DEVICE-armv7hl -m sdk-install -R zypper ar \
  -G http://repo.merproject.org/releases/mer-tools/rolling/builds/armv7hl/packages/ \
  mer-tools-rolling

sb2 -t $VENDOR-$DEVICE-armv7hl -m sdk-install -R zypper ref --force
```

The “collect2: cannot find ‘ld’” error/warning after executing sb2-init can be ignored.

To verify the correct installation of the Scratchbox2 target, cross-compile a simple “Hello, World!” C application with `sb2`:

```
MERSDK $

cd $HOME
cat > main.c << EOF
```

```
#include <stdlib.h>
#include <stdio.h>

int main(void) {
  printf("Hello, world!\n");
  return EXIT_SUCCESS;
}
EOF

sb2 -t $VENDOR-$DEVICE-armv7hl gcc main.c -o test
```

If the compilation was successful you can test the executable by running the following command (this will run the executable using `qemu` as emulation layer, which is part of the `sb2` setup):

```
sb2 -t $VENDOR-$DEVICE-armv7hl ./test
```

The above command should output "Hello, world!" on the console, this proves that the target can compile binaries and execute them for your architecture.

CHAPTER

SEVEN

PACKAGING DROID HAL

In this chapter, we will package the build results of *Building the Android HAL* as RPM packages and create a local RPM repository. From there, the RPM packages can be added to a local target and used to build libhybris and the QPA plugin. They can also be used to build the rootfs.

7.1 Packaging `droid-hal-device`

This step requires:

- A populated `$ANDROID_ROOT` from *Building the Android HAL*
- A Mer Platform SDK installation (chroot) for RPM building

Inside your `$ANDROID_ROOT`, there is a copy of `droid-hal-device` in the `rpm/` directory (since it appears in the manifest).

The master git repo for the packaging is here: https://github.com/mer-hybris/droid-hal-device

This `rpm/` dir contains the necessary `.spec` files to make a set of RPM packages that form the core Droid hardware adaptation part and configuration file setup of the hardware adaptation. It also builds a development package that contains libraries and headers, which are used when building middleware components (see *Middleware*).

7.1.1 Building the droid-hal-device packages

The next step has to be carried out in a Mer SDK chroot:

```
MER_SDK $

cd $ANDROID_ROOT

# THE COMMAND BELOW WILL FAIL. It's normal, carry on with the next one.
# Explanation: force installing of build-requirements by specifying the
# .inc file directly, but build-dependencies will be pulled in via
# zypper, so that the next step has all macro definitions loaded
mb2 -t $VENDOR-$DEVICE-armv7hl -s rpm/droid-hal-device.inc build

mb2 -t $VENDOR-$DEVICE-armv7hl -s rpm/droid-hal-$DEVICE.spec build
```

This should leave you with several RPM packages in `$ANDROID_ROOT/RPMS/`.

If the second `mb2` fails by writing out inconsistencies in kernel `CONFIG_` flags, refer to the kernel verifier section: *Kernel config*.

7.1.2 Create a local RPM repository

Now we create a local repository that can be used to create images using `mic` or to install the development headers into our `sb2` target for building middleware components:

```
MER_SDK $

mkdir -p $ANDROID_ROOT/droid-local-repo/$DEVICE

rm -f $ANDROID_ROOT/droid-local-repo/$DEVICE/droid-hal-*rpm
mv RPMS/*${DEVICE}* $ANDROID_ROOT/droid-local-repo/$DEVICE

createrepo  $ANDROID_ROOT/droid-local-repo/$DEVICE
```

7.1.3 Add local RPM repo to Target

This will allow build dependencies to be met from locally built packages:

```
MER_SDK $

sb2 -t $VENDOR-$DEVICE-armv7hl -R -m sdk-install \
  ssu ar local-$DEVICE-hal file://$ANDROID_ROOT/droid-local-repo/$DEVICE
```

(safe to ignore warnings about connman or DBus):

Check it's there:

```
MER_SDK $

sb2 -t $VENDOR-$DEVICE-armv7hl -R -msdk-install ssu lr
```

7.1.4 The device specific configuration

Now build the droid-hal-configs file. This is split into its own package to make supporting multiple devices easier.

```
MER_SDK $

hadk

cd $ANDROID_ROOT
mb2 -t $VENDOR-$DEVICE-armv7hl \
  -s hybris/droid-hal-configs/rpm/droid-hal-configs.spec \
  build
```

7.2 The `/etc/hw-release` file

Sailfish OS Hardware Adaptations use the file `/etc/hw-release` to store variables related to the device adaptation. This file is read by different middleware components to determine which adaptation repositories to enable and which device-specific tweaks to apply.

File is autogenerated during the build of `droid-hal-device` (see `droid-hal-device.inc`). If you wish to provide more customisations, please read the remainder of this section.

The format of this file is a line-based `KEY=value` format. The `KEY` is a non-empty string consisting of only upper case characters (`A-Z`) and the underscore (_), it must not begin with an underscore (or in other words, it must match the regular expression `[A-Z][A-Z_]*`). Lines starting with `#` are considered comments and are ignored. Lines must not have any leading or trailing whitespace (any such whitespace is stripped when the file is parsed), and the = character must also not be surrounded by any whitespace. Values can contain any valid UTF-8 character (but no newline character).

An example file could look like this:

```
# This is a comment
MER_HA_DEVICE=mako
MER_HA_VENDOR=lge
```

As far as Droid-based hardware adaptations are concerned, the following keys are mandatory and specified:

- `MER_HA_DEVICE`: Must be set to the device name, e.g. `mako`
- `MER_HA_VENDOR`: Must be set to the device vendor, e.g. `lge`

All other keys are not yet specified, and should not be used; parsers should ignore all lines that don't start with a known key.

CHAPTER

EIGHT

CREATING THE SAILFISH OS ROOT FILESYSTEM

8.1 Additional Packages for Hardware Adaptation

Some additional packages are used to allow access to device features. These middleware packages are usually built against droid-headers / libhybris, and therefore need to be built separately for each target device. To build, clone the repository from `mer-hybris` on Github. See *Middleware* for a list of all middleware components (not all middleware components are used for all device adaptations).

Via the flexible system of patterns, you will be able to select only working/needed functions for your device.

8.2 Creating and Configuring the Kickstart File

The kickstart file is generated using `ssuks`, which is part of the SSU utility.

Ensure you have done the steps to *Create a local RPM repository*.

```
MER_SDK $

cd $ANDROID_ROOT
mkdir -p tmp

HA_REPO="repo --name=adaptation0-$DEVICE-@RELEASE@"
sed -e \
 "s|^$HA_REPO.*$|$HA_REPO --baseurl=file://$ANDROID_ROOT/droid-local-repo/$DEVICE|" \
 $ANDROID_ROOT/installroot/usr/share/kickstarts/Jolla-@RELEASE@-$DEVICE-@ARCH@.ks \
 > tmp/Jolla-@RELEASE@-$DEVICE-@ARCH@.ks
```

If you only want to rebuild some of the packages locally (and are confident that there are no changes that require custom rebuilds) then you can use the public build if there is one; we'll use `sed` to find (//) the HA_REPO and then 'a'ppend a new line with the OBS repo url:

```
MOBS_URI="http://repo.merproject.org/obs"
HA_REPO="repo --name=adaptation0-$DEVICE-@RELEASE@"
HA_REPO1="repo --name=adaptation1-$DEVICE-@RELEASE@ \
--baseurl=$MOBS_URI/sailfishos:/devel:/hw:/$DEVICE/sailfish_latest_@ARCH@/"
sed -i -e "/^$HA_REPO.*$/a$HA_REPO1" tmp/Jolla-@RELEASE@-$DEVICE-@ARCH@.ks
```

Feel free to replace `sailfishos:/devel:/hw:` with path to any suitable HA repo within Mer OBS.

8.3 Patterns

The selection of packages for each hardware adaptation has to be put into a pattern file, so that creating the image as well as any system updates in the future can pull in and upgrade all packages related to the hardware adaptation.

Ensure you have done the steps to *Create a local RPM repository*.

Add/update metadata about patterns using this script (NB: it will fail with a non-critical error):

```
MER_SDK $

hadk

cd $ANDROID_ROOT
rpm/helpers/process_patterns.sh
```

As mentioned above, safely ignore the following error:

```
Exception AttributeError: "'NoneType' object has no attribute
  'px_proxy_factory_free'"...
```

To modify a pattern, edit its respective template under `rpm/patterns/{common,hybris,templates}` and then run `rpm/helpers/add_new_device.sh`. Take care and always use `git status/stash` commands.

8.4 Building the Image with MIC

Ensure you re-generated *Patterns* (needs to be run after every `createrepo`)

Building a rootfs using RPM repositories and a kickstart file:

```
MER_SDK $

# always aim for the latest:
RELEASE=1.0.8.19
# WARNING: EXTRA_NAME currently does not support '.' dots in it!
EXTRA_NAME=-my1
sudo mic create fs --arch armv7hl \
    --tokenmap=ARCH:armv7hl,RELEASE:$RELEASE,EXTRA_NAME:$EXTRA_NAME \
    --record-pkgs=name,url \
    --outdir=sfa-mako-ea-$RELEASE$EXTRA_NAME \
    --pack-to=sfa-mako-ea-$RELEASE$EXTRA_NAME.tar.bz2 \
    $ANDROID_ROOT/tmp/Jolla-@RELEASE@-$DEVICE-@ARCH@.ks
```

Once obtained the `.zip` file, proceed installation as per instructions to Early Adopters Release Notes.

Currently HADK does not support creating images with Jolla Store functionality.

If creation fails due to absence of a package required by pattern, note down the package name and proceed to *Dealing with a Missing Package*.

A more obscure error might look like this:

```
Warning: repo problem: pattern:jolla-configuration-$DEVICE-(version).noarch
  requires jolla-hw-adaptation-$DEVICE,
```



```
but this requirement cannot be provided, uninstallable providers:
pattern:jolla-hw-adaptation-$DEVICE-(version).noarch[$DEVICE]
```

This means a package dependency cannot be satisfied down the hierarchy of patterns. A quick in-place solution:

- Substitute the line `@Jolla Configuration $DEVICE` with `@jolla-hw-adaptation-$DEVICE` in your .ks
- Rebuild .ks
- Repeat the steps above substituting respective pattern to walk down the patterns hierarchy – you'll eventually discover the offending package
- If that package is provided by e.g. droid-hal-device (like `droid-hal-mako-pulseaudio-settings`), it means that some of its dependencies are not present:
- Edit .ks file by having `%packages` section consisting only of single `droid-hal-mako-pulseaudio-settings` (note there is no `@` at the beginning of the line, since it's a package, not a pattern) – another `mic` run error will show that the offending package is actually `pulseaudio-modules-droid`

Now you're ready to proceed to the *Dealing with a Missing Package* section.

8.4.1 Dealing with a Missing Package

If that package is critical (e.g. `libhybris`, `qt5-qpa-hwcomposer-plugin` etc.), build and add it to the local repo as explained in *Build HA Middleware Packages*. Afterwards perform:

- *Patterns*
- *Building the Image with MIC*

Otherwise if a package is not critical, and you accept to have less functionality (or even unbootable) image, you can temporarily comment it out from patterns in `rpm/patterns/$DEVICE` and orderly perform:

- *Building the droid-hal-device packages*
- *Create a local RPM repository*
- *Creating and Configuring the Kickstart File*
- *Patterns*
- *Building the Image with MIC*

Alternatively (or if you can't find it among patterns) add `-NAME_OF_PACKAGE` line to your .ks `%packages` section (remember that regenerating .ks will overwrite this modification).

CHAPTER

NINE

GETTING IN

9.1 Boot and Flashing Process

This varies from device to device. There are a few different boot loaders and flashing mechanisms used for Android devices:

- **fastboot**: Used by most Nexus devices
- **odin**: Used by most Samsung devices

For flashing fastboot-based devices, use `fastboot` (available in the Mer SDK), for odin-based devices, use Heimdall.

9.2 Operating Blind on an Existing Device

Long story short, you will have to assume that you cannot:

- See any framebuffer console
- See any error messages of any kind during bootup
- Get any information relayed from your startup process
- Set any kind of modified kernel command lines

Hence, we have to learn how to operate blind on a device. The good news is that when you have a working kernel, you can combine it with a init ramdisk and that Android's USB gadget is built in to most kernel configurations. It is possible then for the ramdisk to set up working USB networking on most devices and then open up a telnet daemon.

The **hybris-boot** repository contains such an initrd with convenient USB networking, DHCP and telnet server, plus the ability to boot into a Sailfish OS system. The init system in the hybris-boot initrd will attempt to write information via the USB device serial number and model. So `dmesg` on the host could produce:

```
[1094634.238136] usb 2-2: Manufacturer: Mer Boat Loader
[1094634.238143] usb 2-2: SerialNumber: Mer Debug setting up (DONE_SWITCH=no)
```

However `dmesg` doesn't report all changes in the USB subsystem and the init script will attempt to update the iSerial field with information so also do:

```
$ lsusb -v | grep iSerial
 iSerial     3 Mer Debug telnet on port 23 on rndis0 192.168.2.15 - also running udhcpd
```

9.3 Splitting and Re-Assembling Boot Images

A **boot.img** file is basically a combination of a Linux kernel and an initramfs as `cpio` archive. The Mer SDK offer the `mkbootimg` to build a boot image from a kernel and cpio archive. To split a boot image, use `split_bootimg` in the SDK.

In the CyanogenMod-based Sailfish OS port, a boot image with Sailfish OS- specific scripts will be built automatically. These boot images are then available as **hybris-boot.img** (for booting into Sailfish OS) and **hybris-recovery.img** (for debugging via telnet and test-booting).

CHAPTER

TEN

FLASHING THE ROOTFS IMAGE

In order to be able to use Sailfish OS on the device, the parts that we built and assembled in the previous chapters now need to be flashed to the device. After flashing, Sailfish OS should boot on your device on the next reboot.

10.1 Prerequisites

- Android Recovery flashed to your device
- The stock firmware image (for your version and device)
- The vanilla CM release (for your version and device)
- A Sailfish OS rootfs update .zip, created by `mic`

10.2 Flashing back to Stock Android

It is important that you start with a fresh stock image that matches the Android version of the CyanogenMod release you are going to flash (which in turn is dictated by the Sailfish OS image you are going to flash).

While the CM .zip contains all files in `/system/` (e.g. libraries and libhardware modules), the stock image also contains firmware parts and flashables for partitions that are not included in the CM .zip.

For example, if you are running stock 4.4.2 on a Nexus 4 (mako), and you are going to flash CM 10.1.3 and Sailfish OS to it, you have to first flash the stock 4.2.2 (note that this is 4.2, not 4.4) first, so that the firmware bits are matching the CM version.

If you do not flash the right stock version (and therefore firmware), there might be some issues when booting into Sailfish OS:

- Problems accessing `/sdcard/` in recovery (e.g. `adb push` does not work)
- WLAN, sensors, audio and other hardware not working

If you experience such issues, please make sure you first flash the stock system, ROM, followed by a Recovery image and CyanogenMod, and finally the Sailfish OS update. Please also note that you can't just take the latest stock ROM and/or CyanogenMod ROM - both versions have to match the Sailfish OS version you are going to install, as the Sailfish OS parts are built against a specific version of the HA.

10.3 Flashing using Android Recovery

1. Boot into Android Recovery
2. Upload the CM release: `adb push cm-10.1.3-$DEVICE.zip /sdcard/`
3. Upload Sailfish OS: `adb push sailfishos-$DEVICE-devel-1.2.3.4.zip /sdcard/`
4. In the Recovery on the device:

1. Clear data and cache (factory reset)
2. Install the CM release by picking the CM image
3. Install Sailfish OS by picking the SFOS image
4. Reboot the device

CHAPTER

ELEVEN

MANUAL INSTALLATION AND MAINTENANCE

This assumes you are booted into CyanogenMod on your device, can `adb shell` to it to get a root shell and have your boot image and rootfs tarball ready.

Some of these approaches also work in Android Recovery (there's an `adbd` running), but you obviously won't have network connectivity for downloading updates.

11.1 Extracting the rootfs via adb

Replace `i9305-devel.tar.gz` with the name of your rootfs tarball:

```
MER_SDK $

adb push i9305-devel.tar.gz /sdcard/
adb shell
su
mkdir -p /data/.stowaways/sailfishos
tar --numeric-owner -xvzf /sdcard/i9305-devel.tar.gz \
    -C /data/.stowaways/sailfishos
```

11.2 Flashing the boot image via adb

The following example is for `i9305`, for other devices the output partition and filename is obviously different:

```
MER_SDK $

adb push out/target/product/i9305/hybris-boot.img /sdcard/
adb shell
su
dd if=/sdcard/hybris-boot.img of=/dev/block/mmcblk0p8
```

11.3 Interacting with the rootfs via adb from Android

You can interact with the Sailfish OS rootfs and carry out maintenance (editing files, installing packages, etc..) when booted into an Android system. You have to have your rootfs already installed/extracted. You can use Android's WLAN connectivity to connect to the Internet and download updates:

```
MER_SDK $

adb shell
su
mount -o bind /dev /data/.stowaways/sailfishos/dev
mount -o bind /proc /data/.stowaways/sailfishos/proc
mount -o bind /sys /data/.stowaways/sailfishos/sys
chroot /data/.stowaways/sailfishos/ /bin/su -
echo "nameserver 8.8.8.8" >/etc/resolv.conf
...
```


CHAPTER

TWELVE

MODIFICATIONS AND PATCHES

Running Sailfish OS using libhybris and Mer requires a few modifications to a standard Android/CM system. We maintain forks of some repos with those patches aplied.

12.1 For Supported Devices

See *List of Supported Devices* for a list of devices supported by HADK. Supported devices are automatically downloaded as part of the HADK android build environment.

12.1.1 Mer Modifications to CyanogenMod

Our modifications are tracked by our own hybris-specific repo manifest file, currently at version *hybris-10.1* which is based on the *CyanogenMod* 10.1.x releases. The below sections outline our modifications to these sources for developing *libhybris* based adaptations.

12.2 Droid System

In order to work with `libhybris`, some parts of the lower levels of Android need to be modified:

- **bionic/**
  - Pass `errno` from bionic to libhybris (`libdsyscalls.so`)
  - Rename `/dev/log/` to `/dev/alog/`
  - TLS slots need to be re-assigned to not conflict with glibc
  - Support for `HYBRIS_LD_LIBRARY_PATH` in the linker
  - Add `/usr/libexec/droid-hybris/system/lib` to the linker search path
- **external/busybox/: Busybox is used in the normal and recovery boot images. We need** some additional features like `mdev` and `udhcpd`.
- **system/core/**
  - Make `cutils` and `logcat` aware of the new log location (`/dev/alog/`)
  - Add `/usr/libexec/droid-hybris/lib-dev-alog/` to the `LD_LIBRARY_PATH`
  - Force SELINUX off since mer doesn't support it

    - Remove various `init` and `init.rc` settings and operations that are handled by `systemd` / Mer on a Sailfish OS system.
- **frameworks/base/: Only build `servicemanager`, `bootanimation` and `androidfw`** to make the minimal Droid HAL build smaller (no Java content)
- **libcore/**: Don't include `JavaLibrary.mk`, as Java won't be available

All these modifications have already been done in the **mer-hybris** Git collection of forks from the original CyanogenMod sources. If the hybris repo manifest is used, these changes will be included automatically.

In addition to these generic modifications, for some devices and SoCs we also maintain a set of patches on top of CyanogenMod to fix issues with drivers that only happen in Sailfish OS, for example:

- **hardware/samsung/**: SEC hwcomposer: Avoid segfault if `registerProcs` was never called

12.3 Kernel

For the Kernel, some configuration options must be enabled to support `systemd` features, and some configuration options must be disabled, because they conflict or block certain features of Sailfish OS.

- **Required Configuration Options**: See `initramfs/init` for a list of required kernel options
- **Conflicting Configuration Options: `CONFIG_ANDROID_PARANOID_NETWORK`:** This would make all network connections fail if the user is not in the group with ID 3003.

As an alternative to checking the kernel options in the initramfs, the script `$ANDROID_ROOT/hybris/mer-kernel-check` can also be used to verify if all required configuration options have been enabled.

12.3.1 Configuring and Compiling the Kernel

For supported devices, the kernel is built as part of `mka hybris-hal` with the right configuration.

For new devices, you have to make sure to get the right kernel configuration included in the repository. For this, clone the kernel repository for the device into **mer-hybris** and configure the kernel using `hybris/mer-kernel-check`.

CHAPTER

THIRTEEN

MIDDLEWARE

This chapter contains some background information about the middleware parts that are part of the Hardware Adapation. Using this info, it should be possible to customize and build the middleware parts for a given device.

13.1 MCE libhybris Plugin

TODO

13.2 Non-Graphic Feedback Daemon Droid Vibrator Plugin

TODO

13.3 Non-Graphic Feedback Daemon PulseAudio Plugin

TODO

13.4 PulseAudio Droid Modules

TODO

13.5 Qt5 QtFeedback Droid Vibrator Plugin

TODO

13.6 Qt5 Hardware Composer QPA

This Qt Platform Abstraction plugin makes use of the libhardware hwcomposer API to send rendered frames from the Wayland Compositor to the actual framebuffer. While for some older devices, just flipping the fbdev was enough, more recent devices actually require using hwcomposer to request flipping and for vsync integration.

The important environment variables are:

- `EGL_PLATFORM` / `HYBRIS_EGLPLATFORM`: For the Wayland Compositor, this needs to be set to `fbdev` on devices with older hwcomposer versions, and to `hwcomposer` for hwcomposer version 1.1 and newer. For best results, first try `fbdev`, and if it doesn't work, try `hwcomposer` instead. For the Wayland Clients, this always needs to be set to `wayland`.
- `QT_QPA_PLATFORM`: For the Wayland Compositor, this needs to be set to `hwcomposer` to use the plugin. Previously, `eglfs` was used, but the `hwcomposer` module replaces the old plugin on Sailfish OS on Droid. For Wayland Clients, this always needs to be set to `wayland`.

When starting up an application (e.g. the Wayland Compositor, `lipstick`), the systemd journal (`journalctl -fa` as user root) will show some details about the detected screen metrics, which will come from the framebuffer device:

```
HwComposerScreenInfo:251 - EGLFS: Screen Info
HwComposerScreenInfo:252 -  - Physical size: QSizeF(57, 100)
HwComposerScreenInfo:253 -  - Screen size: QSize(540, 960)
HwComposerScreenInfo:254 -  - Screen depth: 32
```

Also, it will print information about the hwcomposer module and the device. In this specific case, the hwcomposer version is 0.3:

```
== hwcomposer module ==
* Address: 0x40132000
* Module API Version: 2
* HAL API Version: 0
* Identifier: hwcomposer
* Name: Qualcomm Hardware Composer Module
* Author: CodeAurora Forum
== hwcomposer module ==
== hwcomposer device ==
* Version: 3 (interpreted as 30001)
* Module: 0x40132000
== hwcomposer device ==
```

The source tree contains different implementations of hwcomposer backends, each one for a different hwcomposer API version (see `hwcomposer/hwcomposer_backend.cpp`). Based on that detection, one of the existing implementations is used. Right now, the following implementations exist:

- *hwcomposer_backend_v0*: Version 0.x (e.g. 0.3) of the hwcomposer API. It can handle swapping of an EGL surface to the display, doesn't use any additional hardware layers at the moment and can support switching the screen off. The VSync period is queried from the hwcomposer device, but it will fall back to 60 Hz if the information cannot be determined via the libhardware APIs. (`HYBRIS_EGLPLATFORM=fbdev`)
- *hwcomposer_backend_v10*: Version 1.0 of the hwcomposer API. It supports one display device, handles VSync explicitly and uses a single hardware layer that will be drawn via EGL (and not composed via hwcomposer). Swapping is done by waiting for VSync and uses libsync-based synchronization of posting buffers. Switching the screen off is also supported, and sleeping the screen disables VSync events. Also, the same VSync period algorithm is used (try to query from libhardware, fall back to 60 Hz if detection fails). (`HYBRIS_EGLPLATFORM=fbdev`)
- *hwcomposer_backend_v11*: Version 1.1, 1.2 and 1.3 of the hwcomposer API. Version 1.3 only supports physical displays, whereas 1.1 and 1.2 support also virtual displays. This requires libsync and hwcomposer-egl from libhybris. Most of the hwcomposer 1.0 API properties apply, with the exception that frame posting and synchronization happens with the help of libhybris' hwcomposer EGL platform. (`HYBRIS_EGLPLATFORM=hwcomposer`)

Instead of running the Wayland Compositor (lipstick) on top of the hwcomposer QPA plugin, one can also run all other Qt 5-based applications, but the application can only open a single window (multiple windows are not supported, and will cause an application abort). For multiple windows, Wayland is used. This means that for testing, it is possible to run a simple, single-window Qt 5 application on the framebuffer (without any Wayland Compositor in between) by setting the environment variables `HYBRIS_EGLPLATFORM` and `QT_QPA_PLATFORM` according to the above.

13.7 SensorFW Qt 5 / libhybris Plugin

TODO

13.8 Build HA Middleware Packages

13.8.1 Target setup

Setup to use droid headers

If not done already, as a one-off (per device-target) we need to add the local repo to our target, as indicated in *Add local RPM repo to Target*.

Now set the SDK target to use an up-to-date repo:

```
MER_SDK $

sb2 -t $VENDOR-$DEVICE-armv7hl -R -msdk-install ssu domain sales
sb2 -t $VENDOR-$DEVICE-armv7hl -R -msdk-install ssu dr sdk
```

And install the droid-hal-device headers:

```
MER_SDK $

sb2 -t $VENDOR-$DEVICE-armv7hl -R -msdk-install zypper ref
sb2 -t $VENDOR-$DEVICE-armv7hl -R -msdk-install \
    zypper install droid-hal-$DEVICE-devel
```

If you rebuild the droid-side then you'll need to repeat the two commands above.

13.8.2 Build Area Setup

Setup an area to build packages

```
MER_SDK $

mkdir -p $MER_ROOT/devel/mer-hybris
cd $MER_ROOT/devel/mer-hybris
```

13.8.3 Packages

libhybris

Check out the libhybris source code from Git:


```
MER_SDK $

PKG=libhybris
cd $MER_ROOT/devel/mer-hybris
git clone https://github.com/mer-hybris/libhybris.git
cd libhybris
```

Some packages will use submodules:

```
MER_SDK $

git submodule update
cd libhybris
```

Now use `mb2` to build the package. This essentially runs a slightly modified `rpmbuild` using the Scratchbox2 target. It also pulls in build requirements into the target. Note that this makes the target 'dirty' and you may miss build dependencies. This should be caught during clean builds.

```
MER_SDK $

mb2 -s ../rpm/libhybris.spec -t  $VENDOR-$DEVICE-armv7hl build
```

Now add the packages you just built to the local repo and refresh the repo cache:

```
MER_SDK $

mkdir -p $ANDROID_ROOT/droid-local-repo/$DEVICE/$PKG/
rm -f $ANDROID_ROOT/droid-local-repo/$DEVICE/$PKG/*.rpm
mv RPMS/*.rpm $ANDROID_ROOT/droid-local-repo/$DEVICE/$PKG
createrepo  $ANDROID_ROOT/droid-local-repo/$DEVICE
sb2 -t  $VENDOR-$DEVICE-armv7hl -R -msdk-install zypper ref
```

Note that all repositories that are in `tar_git` format (for use with OBS) will have their RPM packages built locally might not always have the right release and version set.

At this point, and for the libhybris package only, you can remove the mesa-llvmpipe packages from the target:

```
MER_SDK $

sb2 -t  $VENDOR-$DEVICE-armv7hl -R -msdk-build zypper rm mesa-llvmpipe
```

Failure to do this will cause problems pulling in build requirements for other packages.

qt5-qpa-hwcomposer-plugin

```
MER_SDK $

PKG=qt5-qpa-hwcomposer-plugin
cd $MER_ROOT/devel/mer-hybris
git clone https://github.com/mer-hybris/$PKG.git
cd $PKG
mb2 -s rpm/$PKG.spec -t  $VENDOR-$DEVICE-armv7hl build
mkdir -p $ANDROID_ROOT/droid-local-repo/$DEVICE/$PKG/
rm -f $ANDROID_ROOT/droid-local-repo/$DEVICE/$PKG/*.rpm
mv RPMS/*.rpm $ANDROID_ROOT/droid-local-repo/$DEVICE/$PKG
```



```
createrepo  $ANDROID_ROOT/droid-local-repo/$DEVICE
sb2 -t  $VENDOR-$DEVICE-armv7hl -R -msdk-install zypper ref
```

sensorfw

```
MER_SDK $

PKG=sensorfw
SPEC=sensorfw-qt5-hybris
OTHER_RANDOM_NAME=hybris-libsensorfw-qt5

cd $MER_ROOT/devel/mer-hybris
git clone https://github.com/mer-hybris/$PKG.git
cd $PKG
mb2 -s rpm/$SPEC.spec -t  $VENDOR-$DEVICE-armv7hl build
mkdir -p $ANDROID_ROOT/droid-local-repo/$DEVICE/$PKG/
rm -f $ANDROID_ROOT/droid-local-repo/$DEVICE/$PKG/*.rpm
mv RPMS/*.rpm $ANDROID_ROOT/droid-local-repo/$DEVICE/$PKG
createrepo  $ANDROID_ROOT/droid-local-repo/$DEVICE
sb2 -t  $VENDOR-$DEVICE-armv7hl -R -msdk-install zypper ref
```

ngfd-plugin-droid-vibrator

```
MER_SDK $

PKG=ngfd-plugin-droid-vibrator
SPEC=$PKG

cd $MER_ROOT/devel/mer-hybris
git clone https://github.com/mer-hybris/$PKG.git
cd $PKG
mb2 -s rpm/$SPEC.spec -t  $VENDOR-$DEVICE-armv7hl build
mkdir -p $ANDROID_ROOT/droid-local-repo/$DEVICE/$PKG/
rm -f $ANDROID_ROOT/droid-local-repo/$DEVICE/$PKG/*.rpm
mv RPMS/*.rpm $ANDROID_ROOT/droid-local-repo/$DEVICE/$PKG
createrepo  $ANDROID_ROOT/droid-local-repo/$DEVICE
sb2 -t  $VENDOR-$DEVICE-armv7hl -R -msdk-install zypper ref
```

qt5-feedback-haptics-droid-vibrator

```
MER_SDK $

PKG=qt5-feedback-haptics-droid-vibrator
SPEC=$PKG

cd $MER_ROOT/devel/mer-hybris
git clone https://github.com/mer-hybris/$PKG.git
cd $PKG
mb2 -s rpm/$SPEC.spec -t  $VENDOR-$DEVICE-armv7hl build
mkdir -p $ANDROID_ROOT/droid-local-repo/$DEVICE/$PKG/
rm -f $ANDROID_ROOT/droid-local-repo/$DEVICE/$PKG/*.rpm
mv RPMS/*.rpm $ANDROID_ROOT/droid-local-repo/$DEVICE/$PKG
```



```
createrepo  $ANDROID_ROOT/droid-local-repo/$DEVICE
sb2 -t  $VENDOR-$DEVICE-armv7hl -R -msdk-install zypper ref
```

pulseaudio-modules-droid

```
MER_SDK $

PKG=pulseaudio-modules-droid
SPEC=$PKG

cd $MER_ROOT/devel/mer-hybris
git clone https://github.com/mer-hybris/$PKG.git
cd $PKG
mb2 -s rpm/$SPEC.spec -t  $VENDOR-$DEVICE-armv7hl build
mkdir -p $ANDROID_ROOT/droid-local-repo/$DEVICE/$PKG/
rm -f $ANDROID_ROOT/droid-local-repo/$DEVICE/$PKG/*.rpm
mv RPMS/*.rpm $ANDROID_ROOT/droid-local-repo/$DEVICE/$PKG
createrepo  $ANDROID_ROOT/droid-local-repo/$DEVICE
sb2 -t  $VENDOR-$DEVICE-armv7hl -R -msdk-install zypper ref
```


CHAPTER

FOURTEEN

PORTING SAILFISH OS TO A NEW DEVICE

14.1 Find Device Info

See *Prerequisites* for sourcing information about your device.

Go through *Preparing Your Device*.

Ensure your device boots CM. You should not need to build the whole of CM but it can be a useful (and slow!) diagnostic if you hit problems in later stages. In this case use a separate folder and follow instructions on the CM device wiki to be sure your environment can reproduce a working CM kernel and rootfs images.

This example has snippets based on the Encore. The CyanogenMod base ROM has been downloaded using:

```
MER_SDK $

curl -L -O \
  http://download.cyanogenmod.org/get/jenkins/51847/cm-10.1.3.2-encore.zip
```

Installation of the ROM is described in the Encore install guide.

14.2 Prepare Environment

Go through *Setting up the SDKs*.

Make sure all the commands use the correct `$DEVICE` and `$VENDOR` by updating your `~/.hadk.env` (in this example, `DEVICE=encore` and `VENDOR=bn`) or creating a new one `~/.hadk.env.encore` and using `hadk encore`.

14.3 Build Android

Go through section *Building Relevant Bits of CyanogenMod* up to (not including) the *repo sync* part, to ensure everything is setup and environment is initialised, then come back here and proceed with sections below.

14.3.1 Device repos

You'll need a new local manifest. The example given below is for encore. Modify it appropriately.

The entries will vary per-device but you'll need the device and kernel repos as a minimum. Depending on any build issues that arise you may need additional `hardware/` and/or `external/` repositories (the example ones probably won't be needed for your device). You'll need to fork the kernel repository in order to update the default config:

- `device/$VENDOR/$DEVICE`
- `kernel/$VENDOR/$DEVICE`

```
HABUILD_SDK $

mkdir .repo/local_manifests/
cat <<EOF > .repo/local_manifests/encore.xml
<?xml version="1.0" encoding="UTF-8"?>
<manifest>
  <project path="device/bn/encore" name="CyanogenMod/android_device_bn_encore"
    revision="cm-10.1" />
  <project path="kernel/bn/encore" name="lbt/android_kernel_bn_encore"
    revision="cm-10.1-staging" />
  <project path="hardware/ti/wlan" name="CyanogenMod/android_hardware_ti_wlan"
    revision="cm-10.1" />
  <project path="external/wpa_supplicant_8"
    name="CyanogenMod/android_external_wpa_supplicant_8"
    revision="cm-10.1" />
</manifest>
EOF
```

Once you have a local manifest you can sync the Git repositories:

```
HABUILD_SDK $

repo sync
breakfast $DEVICE
```

If you get errors about duplicate repositories or github starts saying "Forbidden ...", see the *Common Pitfalls* section (from *Building the Android HAL* chapter).

14.3.2 Configure mountpoint information

Until `systemd` is updated we need to patch `hybris/hybris-boot/fixup-mountpoints` for the device. The idea here is to ensure the udev-less initrd mounts the correct `/boot` and `/data` partition. If you're lucky the device will simply use `/dev/block/<somedev>` and you can use the i9305 approach. If not then look in the recovery `fstab` for the right mapping.

To double check, you can boot to CM and `adb shell` to examine `/dev/block*` and `/dev/mmc*` (udev-full) contents. Also boot into ClockworkMod Recovery, to check those (udev-less) paths there too.

The build log will also have provided feedback like:

```
HABUILD_SDK $

hybris/hybris-boot/Android.mk:48: ********************* /boot should
  live on /dev/block/platform/msm_sdcc.1/by-name/boot
hybris/hybris-boot/Android.mk:49: ********************* /data should
  live on /dev/block/platform/msm_sdcc.1/by-name/userdata
```


Note that a subsequent `repo sync` will reset this unless you update your `.repo/local_manifests/encore.xml` to point to a fork of the hybris-boot repo.

14.3.3 Additional packages

Additional tools can be downloaded inside the Android Ubuntu chroot. For example, devices based on the U-Boot bootloader require the `mkimage` utility, which can be installed with the following command:

```
HABUILD_SDK $

sudo apt-get install uboot-mkimage
```

14.3.4 Do a build

You'll probably need to iterate this a few times to spot missing repositories, tools, configuration files and others:

```
HABUILD_SDK $

mka hybris-hal
```

For example, an error about `hardware/ti/wlan/mac80211/compat_wl12xx` leads us to check `.repo/manifests/cm-10.1.3.xml` and find a likely looking project; you can see in the example above it was added to `.repo/local_manifests/encore.xml`.

If you're building for encore, try removing it from the local manifest and removing the `hardware/ti` directory to see the errors. Repeat this for other local projects you may find. Also note that you may have to run `mka hybris-hal` multiple times; please report a bug if that happens as something will be wrong with dependencies.

If you hit any other issues then please report them too.

14.3.5 Kernel config

Once the kernel has built you can check the kernel config. You can use the Mer kernel config checker:

```
HABUILD_SDK $

tmp/mer_verify_kernel_config ./out/target/product/$DEVICE/obj/KERNEL_OBJ/.config
```

Apply listed modifications to the defconfig file that CM is using. Which one? It's different for every device:

- Check CM kernel's commit history of the `arch/arm/configs` folder, look for defconfig
- Double-check which defconfig is taken when you're building kernel
- Check it's name under $ANDROID_ROOT/device/$VENDOR/*/BoardConfigCommon.mk

After you'll have applied the needed changes, re-run `mka hybris-boot` and re-verify. Lather, rinse, repeat :) Run also `mka hybris-recovery` in the end when no more errors.

14.3.6 Success

You've finished this section when your build finishes with:

```
HABUILD_SDK $

Install: $ANDROID_ROOT/out/target/product/$DEVICE/hybris-recovery.img
Install: $ANDROID_ROOT/out/target/product/$DEVICE/hybris-boot.img
```

14.4 Mer-side package building

As you may expect this section is done in the Mer SDK. Again, ensure the environment is correct:

```
MER_SDK $

hadk
```

14.4.1 Device specific target

Setup a device-specific target. This step is generally only needed when working with the HA layer because the target will contain device-specific information that is not usually needed in a target.

Setup a device target by going through the whole *Setting up Scratchbox2 Target* chapter.

Create a simple `.spec` file that sets the correct variables and then includes `droid-hal-device.inc`, which contains the RPM building logic:

```
MER_SDK $

cd $ANDROID_ROOT
cat <<EOF > rpm/droid-hal-$DEVICE.spec
# device is the cyanogenmod codename for the device
# eg mako = Nexus 4
%define device $DEVICE
# vendor is used in device/%vendor/%device/
%define vendor $VENDOR

%include rpm/droid-hal-device.inc
EOF
```

And generate device folder structure and patterns:

```
MER_SDK $

cd $ANDROID_ROOT

rpm/helpers/add_new_device.sh
```

14.4.2 Device specific config

You'll need as a minimum:


```
MER_SDK $

COMPOSITOR_CFGS=rpm/device-$VENDOR-$DEVICE-configs/var/lib/environment/compositor
mkdir -p $COMPOSITOR_CFGS
cat <<EOF >$COMPOSITOR_CFGS/droid-hal-device.conf
# Config for $VENDOR/$DEVICE
HYBRIS_EGLPLATFORM=fbdev
QT_QPA_PLATFORM=hwcomposer
LIPSTICK_OPTIONS=-plugin evdevtouch:/dev/input/event0 \
  -plugin evdevkeyboard:keymap=/usr/share/qt5/keymaps/droid.qmap
EOF
```

14.4.3 Build the HAL

Go through the whole *Packaging Droid HAL* chapter.

14.4.4 HAL specific packages

See section *Build HA Middleware Packages*.

CHAPTER

FIFTEEN

LIST OF SUPPORTED DEVICES

Devices currently supported by HADK (with $DEVICE/$VENDOR in brackets)

- **Nexus 4** (mako/lge)
- **Nexus 7 2012 GSM** (tilapia/asus)
- **Nexus 7 2012 WIFI** (grouper/asus)
- **Samsung Galaxy SIII LTE** (i9305/samsung)

For an up-to-date list of supported features for each device, see Adaptations/libhybris in the Mer Wiki.

15.1 Devices: $DEVICE and $VENDOR

To get this information find your device here: CyanogenMod Devices, note down the "*Manufacturer*" and "*Codename*" values, which are displayed in the side bar on the right. The Codename is the DEVICE and the Manufacturer is the VENDOR.

15.2 For New Devices

Please refer to the *Want to Port to a New Device?* section.

CHAPTER

SIXTEEN

LIST OF REPOSITORIES

droid-hal-device Contains RPM packaging and conversion scripts for converting the results of the Android HAL build process to RPM packages and systemd configuration files.

droid-system-packager Helper package for `droid-hal-device`, installed on the device.

hybris-boot Script run during Android HAL build that will combine the kernel and a custom initrd to `hybris-boot.img` and `hybris-recovery.img`. Those are used to boot a device into Sailfish OS and for development purposes.

hybris-installer Combines the `hybris-boot` output and the root filesystem into a .zip file that can be flashed via Android Recovery.

libhybris Library to allow access to Bionic-based libraries from a glibc-based host system (e.g. hwcomposer, EGL, GLESv2, ..).

qt5-qpa-hwcomposer-plugin Qt 5 Platform Abstraction Plugin that allows fullscreen rendering to the Droid-based hardware abstraction. It utilizes libhybris and the Android hwcomposer module.

mer-kernel-check A script that checks if the kernel configuration is suitable for Sailfish OS. Some features must be enabled, as they are needed on Sailfish OS (e.g. to support `systemd`), other features must be disabled, as they conflict with Sailfish OS (e.g. `CONFIG_ANDROID_PARANOID_NETWORK`) at the moment.

CHAPTER

SEVENTEEN

PACKAGE NAMING POLICY

For consistency, certain hardware adaptation / middleware plugin packages have to be named after a certain pattern.

As in the other chapters of this guide, `$DEVICE` should be replaced with the device codename (e.g. `mako` for Nexus 4), and the asterisk (*) is used as wildcard / placeholder.

17.1 List of naming rules

Packages that are arch-specific (e.g. `armv7hl`), device-specific and contain `$DEVICE` in their name:

- The arch-specific HAL RPMs (built from `droid-hal-device`) should be named **droid-hal-$DEVICE** (e.g. `droid-hal-mako`, `droid-hal-mako-devel`, `droid-hal-mako-img-boot`, `droid-hal-mako-kernel`, `droid-hal-mako-kernel-modules`, `droid-hal-mako-kickstart-configuration`, `droid-hal-mako-patterns`, `droid-hal-mako-policy-settings` and `droid-hal-mako-pulseaudio-settings`)
- The package containing kickstart files for `mic` should be named **ssu-kickstarts-$DEVICE** (e.g. `ssu-kickstarts-mako`)

Package that are arch-independent (`noarch`), device-specific and contain `$DEVICE` in their name:

- The arch-independent HAL RPMs (built from `droid-hal-device`) should be named: **droid-hal-$DEVICE-*** (e.g. `droid-hal-mako-img-recovery` and `droid-hal-mako-sailfish-config`)
- The SensorFW libhybris plugin configuration should be named **hybris-libsensorfw-qt5-configs** (`hybris-libsensorfw-qt5-configs`)

Packages that are arch-specific (e.g. `armv7hl`), device-specific, but do not contain `$DEVICE`:

- RPMs built from libhybris should be named **libhybris-*** (e.g. `libhybris-libEGL`)
- Plugins for the non-graphic feedback daemon should be named **ngfd-plugin-*** (e.g. `ngfd-plugin-droid-vibrator`); as well as their Qt plugin **qt5-feedback-haptics-droid-vibrator** (`qt5-feedback-haptics-droid-vibrator`)
- The QPA hwcomposer plugin should be named **qt5-qpa-hwcomposer-plugin** (`qt5-qpa-hwcomposer-plugin`)
- The PulseAudio support modules should be named **pulseaudio-modules-droid** (`pulseaudio-modules-droid`)

- The GStreamer plugins should be named **libgstreamer0.10-*** and/or **gstramer0.10-*** (e.g. `libgstreamer0.10-gralloc`, `libgstreamer0.10-nativebuffer`, `gstreamer0.10-omx`, `gstreamer0.10-droideglsink` and `gstreamer0.10-droidcamsrc`)
- The SensorFW libhybris plugin should be named **hybris-libsensorfw-qt5** (`hybris-libsensorfw-qt5`)

17.2 List of Provides

- **droid-hal-$DEVICE-*** provides **droid-hal-*** (e.g. `droid-hal-$DEVICE-pulseaudio-settings` provides `droid-hal-pulseaudio-settings`)

17.3 TODO

The above "rules" are the current state of our hardware adaptation. Here are some things that should be improved there:

- Some arch-specific packages contain arch-independent config files or binary blobs - make them arch-independent (`noarch`) instead
- Unify the GStreamer plugin naming (either **libgstreamer0.10-*** or **gstreamer0.10-***) to not have two naming schemes there
- The PulseAudio settings package usually is called **pulseaudio-settings-$DEVICE** (we currently have **droid-hal-$DEVICE-pulseaudio-settings**, maybe this can be implemented as a `Provides:`?)
- The Linux kernel modules are in **droid-hal-$DEVICE-kernel-modules** at the moment, in other hardware adaptations we use **kmod-xyz-$DEVICE**
- The recovery partition in the image at the moment is **droid-hal-$DEVICE-img-recovery**, but for other hardware adaptations we use **jolla-recovery-$DEVICE**

CHAPTER

EIGHTEEN

LICENSE

Creative Commons Legal Code

Attribution-NonCommercial-ShareAlike 3.0 Unported

> CREATIVE COMMONS CORPORATION IS NOT A LAW FIRM AND DOES NOT PROVIDE LEGAL SERVICES. DISTRIBUTION OF THIS LICENSE DOES NOT CREATE AN ATTORNEY-CLIENT RELATIONSHIP. CREATIVE COMMONS PROVIDES THIS INFORMATION ON AN "AS-IS" BASIS. CREATIVE COMMONS MAKES NO WARRANTIES REGARDING THE INFORMATION PROVIDED, AND DISCLAIMS LIABILITY FOR DAMAGES RESULTING FROM ITS USE.

License

THE WORK (AS DEFINED BELOW) IS PROVIDED UNDER THE TERMS OF THIS CREATIVE COMMONS PUBLIC LICENSE ("CCPL" OR "LICENSE"). THE WORK IS PROTECTED BY COPYRIGHT AND/OR OTHER APPLICABLE LAW. ANY USE OF THE WORK OTHER THAN AS AUTHORIZED UNDER THIS LICENSE OR COPYRIGHT LAW IS PROHIBITED.

BY EXERCISING ANY RIGHTS TO THE WORK PROVIDED HERE, YOU ACCEPT AND AGREE TO BE BOUND BY THE TERMS OF THIS LICENSE. TO THE EXTENT THIS LICENSE MAY BE CONSIDERED TO BE A CONTRACT, THE LICENSOR GRANTS YOU THE RIGHTS CONTAINED HERE IN CONSIDERATION OF YOUR ACCEPTANCE OF SUCH TERMS AND CONDITIONS.

1. Definitions

1. "Adaptation" means a work based upon the Work, or upon the Work and other pre-existing works, such as a translation, adaptation, derivative work, arrangement of music or other alterations of a literary or artistic work, or phonogram or performance and includes cinematographic adaptations or any other form in which the Work may be recast, transformed, or adapted including in any form recognizably derived from the original, except that a work that constitutes a Collection will not be considered an Adaptation for the purpose of this License. For the avoidance of doubt, where the Work is a musical work, performance or phonogram, the synchronization of the Work in timed-relation with a moving image ("synching") will be considered an Adaptation for the purpose of this License.

2. "Collection" means a collection of literary or artistic works, such as encyclopedias and anthologies, or performances, phonograms or broadcasts, or other works or subject matter other than works listed in Section 1(g) below, which, by reason of the selection and arrangement of their contents, constitute intellectual creations, in which the Work is included in its entirety in unmodified form along with one or more other contributions, each constituting separate and independent works in themselves, which together are assembled into a collective whole. A work that constitutes a Collection will not be considered an Adaptation (as defined above) for the purposes of this

License.

3. "Distribute" means to make available to the public the original and copies of the Work or Adaptation, as appropriate, through sale or other transfer of ownership.

4. "License Elements" means the following high-level license attributes as selected by Licensor and indicated in the title of this License: Attribution, Noncommercial, ShareAlike.

5. "Licensor" means the individual, individuals, entity or entities that offer(s) the Work under the terms of this License.

6. "Original Author" means, in the case of a literary or artistic work, the individual, individuals, entity or entities who created the Work or if no individual or entity can be identified, the publisher; and in addition (i) in the case of a performance the actors, singers, musicians, dancers, and other persons who act, sing, deliver, declaim, play in, interpret or otherwise perform literary or artistic works or expressions of folklore; (ii) in the case of a phonogram the producer being the person or legal entity who first fixes the sounds of a performance or other sounds; and, (iii) in the case of broadcasts, the organization that transmits the broadcast.

7. "Work" means the literary and/or artistic work offered under the terms of this License including without limitation any production in the literary, scientific and artistic domain, whatever may be the mode or form of its expression including digital form, such as a book, pamphlet and other writing; a lecture, address, sermon or other work of the same nature; a dramatic or dramatico-musical work; a choreographic work or entertainment in dumb show; a musical composition with or without words; a cinematographic work to which are assimilated works expressed by a process analogous to cinematography; a work of drawing, painting, architecture, sculpture, engraving or lithography; a photographic work to which are assimilated works expressed by a process analogous to photography; a work of applied art; an illustration, map, plan, sketch or three-dimensional work relative to geography, topography, architecture or science; a performance; a broadcast; a phonogram; a compilation of data to the extent it is protected as a copyrightable work; or a work performed by a variety or circus performer to the extent it is not otherwise considered a literary or artistic work.

8. "You" means an individual or entity exercising rights under this License who has not previously violated the terms of this License with respect to the Work, or who has received express permission from the Licensor to exercise rights under this License despite a previous violation.

9. "Publicly Perform" means to perform public recitations of the Work and to communicate to the public those public recitations, by any means or process, including by wire or wireless means or public digital performances; to make available to the public Works in such a way that members of the public may access these Works from a place and at a place individually chosen by them; to perform the Work to the public by any means or process and the communication to the public of the performances of the Work, including by public digital performance; to broadcast and rebroadcast the Work by any means including signs, sounds or images.

10. "Reproduce" means to make copies of the Work by any means including without limitation by sound or visual recordings and the right of fixation and reproducing fixations of the Work, including storage of a protected performance or phonogram in digital form or other electronic medium.

2. Fair Dealing Rights. Nothing in this License is intended to reduce, limit, or restrict any uses free from copyright or rights arising from limitations or exceptions that are provided for in connection with the copyright protection under copyright law or other applicable laws.

3. License Grant. Subject to the terms and conditions of this License, Licensor hereby grants You a worldwide, royalty-free, non-exclusive, perpetual (for the duration of the applicable copyright) license to exercise the rights in the Work as stated below:

1. to Reproduce the Work, to incorporate the Work into one or more Collections, and to Reproduce the Work as incorporated in the Collections;
2. to create and Reproduce Adaptations provided that any such Adaptation, including any translation in any medium, takes reasonable steps to clearly label, demarcate or otherwise identify that changes were made to the original Work. For example, a translation could be marked "The original work was translated from English to Spanish," or a modification could indicate "The original work has been modified.";
3. to Distribute and Publicly Perform the Work including as incorporated in Collections; and,
4. to Distribute and Publicly Perform Adaptations.

The above rights may be exercised in all media and formats whether now known or hereafter devised. The above rights include the right to make such modifications as are technically necessary to exercise the rights in other media and formats. Subject to Section 8(f), all rights not expressly granted by Licensor are hereby reserved, including but not limited to the rights described in Section 4(e).

4. Restrictions. The license granted in Section 3 above is expressly made subject to and limited by the following restrictions:

1. You may Distribute or Publicly Perform the Work only under the terms of this License. You must include a copy of, or the Uniform Resource Identifier (URI) for, this License with every copy of the Work You Distribute or Publicly Perform. You may not offer or impose any terms on the Work that restrict the terms of this License or the ability of the recipient of the Work to exercise the rights granted to that recipient under the terms of the License. You may not sublicense the Work. You must keep intact all notices that refer to this License and to the disclaimer of warranties with every copy of the Work You Distribute or Publicly Perform. When You Distribute or Publicly Perform the Work, You may not impose any effective technological measures on the Work that restrict the ability of a recipient of the Work from You to exercise the rights granted to that recipient under the terms of the License. This Section 4(a) applies to the Work as incorporated in a Collection, but this does not require the Collection apart from the Work itself to be made subject to the terms of this License. If You create a Collection, upon notice from any Licensor You must, to the extent practicable, remove from the Collection any credit as required by Section 4(d), as requested. If You create an Adaptation, upon notice from any Licensor You must, to the extent practicable, remove from the Adaptation any credit as required by Section 4(d), as requested.
2. You may Distribute or Publicly Perform an Adaptation only under: (i) the terms of this License; (ii) a later version of this License with the same License Elements as this License; (iii) a Creative Commons jurisdiction license (either this or a later license version) that contains the same License Elements as this License (e.g., Attribution-NonCommercial-ShareAlike 3.0 US) ("Applicable License"). You must include a copy of, or the URI, for Applicable License with every copy of each Adaptation You Distribute or Publicly Perform. You may not offer or impose any terms on the Adaptation that restrict the terms of the Applicable License or the ability of the recipient of the Adaptation to exercise the rights granted to that recipient under the terms of the Applicable License. You must keep intact all notices that refer to the Applicable License and to the disclaimer of warranties with every copy of the Work as included in the Adaptation You Distribute or Publicly Perform. When You Distribute or Publicly Perform the Adaptation, You may not impose any effective technological measures on the Adaptation that restrict the ability of a recipient of the Adaptation from You to exercise the rights granted to that recipient under the terms of the Applicable License. This Section 4(b) applies to the Adaptation as incorporated in a Collection, but this

does not require the Collection apart from the Adaptation itself to be made subject to the terms of the Applicable License.

3. You may not exercise any of the rights granted to You in Section 3 above in any manner that is primarily intended for or directed toward commercial advantage or private monetary compensation. The exchange of the Work for other copyrighted works by means of digital file-sharing or otherwise shall not be considered to be intended for or directed toward commercial advantage or private monetary compensation, provided there is no payment of any monetary compensation in con-nection with the exchange of copyrighted works.

4. If You Distribute, or Publicly Perform the Work or any Adaptations or Collections, You must, unless a request has been made pursuant to Section 4(a), keep intact all copyright notices for the Work and provide, reasonable to the medium or means You are utilizing: (i) the name of the Original Author (or pseudonym, if applicable) if supplied, and/or if the Original Author and/or Licensor designate another party or parties (e.g., a sponsor institute, publishing entity, journal) for attribution ("Attribution Parties") in Licensor's copyright notice, terms of service or by other reasonable means, the name of such party or parties; (ii) the title of the Work if supplied; (iii) to the extent reasonably practicable, the URI, if any, that Licensor specifies to be associated with the Work, unless such URI does not refer to the copyright notice or licensing information for the Work; and, (iv) consistent with Section 3(b), in the case of an Adaptation, a credit identifying the use of the Work in the Adaptation (e.g., "French translation of the Work by Original Author," or "Screenplay based on original Work by Original Author"). The credit required by this Section 4(d) may be implemented in any reasonable manner; provided, however, that in the case of a Adaptation or Collection, at a minimum such credit will appear, if a credit for all contributing authors of the Adaptation or Collection appears, then as part of these credits and in a manner at least as prominent as the credits for the other contributing authors. For the avoidance of doubt, You may only use the credit required by this Section for the purpose of attribution in the manner set out above and, by exercising Your rights under this License, You may not implicitly or explicitly assert or imply any connection with, sponsorship or endorsement by the Original Author, Licensor and/or Attribution Parties, as appropriate, of You or Your use of the Work, without the separate, express prior written permission of the Original Author, Licensor and/or Attribution Parties.

5. For the avoidance of doubt:

   (a) Non-waivable Compulsory License Schemes. In those jurisdictions in which the right to collect royalties through any statutory or compulsory licensing scheme cannot be waived, the Licensor reserves the exclusive right to collect such royalties for any exercise by You of the rights granted under this License;

   (a) Waivable Compulsory License Schemes. In those jurisdictions in which the right to collect royalties through any statutory or compulsory licensing scheme can be waived, the Licensor reserves the exclusive right to collect such royalties for any exercise by You of the rights granted under this License if Your exercise of such rights is for a purpose or use which is otherwise than noncommercial as permitted under Section 4(c) and otherwise waives the right to collect royalties through any statutory or compulsory licensing scheme; and,

3. Voluntary License Schemes. The Licensor reserves the right to collect royalties, whether individually or, in the event that the Licensor is a member of a collecting society that administers voluntary licensing schemes, via that society, from any exer-

cise by You of the rights granted under this License that is for a purpose or use which is otherwise than noncommercial as permitted under Section 4(c).

6. Except as otherwise agreed in writing by the Licensor or as may be otherwise permitted by applicable law, if You Reproduce, Distribute or Publicly Perform the Work either by itself or as part of any Adaptations or Collections, You must not distort, mutilate, modify or take other derogatory action in relation to the Work which would be prejudicial to the Original Author's honor or reputation. Licensor agrees that in those jurisdictions (e.g. Japan), in which any exercise of the right granted in Section 3(b) of this License (the right to make Adaptations) would be deemed to be a distortion, mutilation, modification or other derogatory action prejudicial to the Original Author's honor and reputation, the Licensor will waive or not assert, as appropriate, this Section, to the fullest extent permitted by the applicable national law, to enable You to reasonably exercise Your right under Section 3(b) of this License (right to make Adaptations) but not otherwise.

5. Representations, Warranties and Disclaimer

UNLESS OTHERWISE MUTUALLY AGREED TO BY THE PARTIES IN WRITING AND TO THE FULLEST EXTENT PERMITTED BY APPLICABLE LAW, LICENSOR OFFERS THE WORK AS-IS AND MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND CONCERNING THE WORK, EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, INCLUDING, WITHOUT LIMITATION, WARRANTIES OF TITLE, MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NONINFRINGEMENT, OR THE ABSENCE OF LATENT OR OTHER DEFECTS, ACCURACY, OR THE PRESENCE OF ABSENCE OF ERRORS, WHETHER OR NOT DISCOVERABLE. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO THIS EXCLUSION MAY NOT APPLY TO YOU.

6. Limitation on Liability. EXCEPT TO THE EXTENT REQUIRED BY APPLICABLE LAW, IN NO EVENT WILL LICENSOR BE LIABLE TO YOU ON ANY LEGAL THEORY FOR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL, PUNITIVE OR EXEMPLARY DAMAGES ARISING OUT OF THIS LICENSE OR THE USE OF THE WORK, EVEN IF LICENSOR HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

7. Termination

1. This License and the rights granted hereunder will terminate automatically upon any breach by You of the terms of this License. Individuals or entities who have received Adaptations or Collections from You under this License, however, will not have their licenses terminated provided such individuals or entities remain in full compliance with those licenses. Sections 1, 2, 5, 6, 7, and 8 will survive any termination of this License.

2. Subject to the above terms and conditions, the license granted here is perpetual (for the duration of the applicable copyright in the Work). Notwithstanding the above, Licensor reserves the right to release the Work under different license terms or to stop distributing the Work at any time; provided, however that any such election will not serve to withdraw this License (or any other license that has been, or is required to be, granted under the terms of this License), and this License will continue in full force and effect unless terminated as stated above.

8. Miscellaneous

1. Each time You Distribute or Publicly Perform the Work or a Collection, the Licensor offers to the recipient a license to the Work on the same terms and conditions as the license granted to You under this License.

2. Each time You Distribute or Publicly Perform an Adaptation, Licensor offers to the recipient a

license to the original Work on the same terms and conditions as the license granted to You under this License.

3. If any provision of this License is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this License, and without further action by the parties to this agreement, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.
4. No term or provision of this License shall be deemed waived and no breach consented to unless such waiver or consent shall be in writing and signed by the party to be charged with such waiver or consent.
5. This License constitutes the entire agreement between the parties with respect to the Work licensed here. There are no understandings, agreements or representations with respect to the Work not specified here. Licensor shall not be bound by any additional provisions that may appear in any communication from You. This License may not be modified without the mutual written agreement of the Licensor and You.
6. The rights granted under, and the subject matter referenced, in this License were drafted utilizing the terminology of the Berne Convention for the Protection of Literary and Artistic Works (as amended on September 28, 1979), the Rome Convention of 1961, the WIPO Copyright Treaty of 1996, the WIPO Performances and Phonograms Treaty of 1996 and the Universal Copyright Convention (as revised on July 24, 1971). These rights and subject matter take effect in the relevant jurisdiction in which the License terms are sought to be enforced according to the corresponding provisions of the implementation of those treaty provisions in the applicable national law. If the standard suite of rights granted under applicable copyright law includes additional rights not granted under this License, such additional rights are deemed to be included in the License; this License is not intended to restrict the license of any rights under applicable law.

Creative Commons Notice

> Creative Commons is not a party to this License, and makes no warranty whatsoever in connection with the Work. Creative Commons will not be liable to You or any party on any legal theory for any damages whatsoever, including without limitation any general, special, incidental or consequential damages arising in connection to this license. Notwithstanding the foregoing two (2) sentences, if Creative Commons has expressly identified itself as the Licensor hereunder, it shall have all rights and obligations of Licensor.
>
> Except for the limited purpose of indicating to the public that the Work is licensed under the CCPL, Creative Commons does not authorize the use by either party of the trademark "Creative Commons" or any related trademark or logo of Creative Commons without the prior written consent of Creative Commons. Any permitted use will be in compliance with Creative Commons' then-current trademark usage guidelines, as may be published on its website or otherwise made available upon request from time to time. For the avoidance of doubt, this trademark restriction does not form part of this License.
>
> Creative Commons may be contacted at http://creativecommons.org/.

CHAPTER

NINETEEN

INDICES AND TABLES

- *genindex*
- *search*