Android AppSupport

Information on the Android AppSupport in Sailfish OS.

General information can be found in the official documentation at
https://docs.sailfishos.org/Support/Help_Articles/Android_App_Support/

Android Layers in Sailfish OS

This page is mostly based on DrYak's explanations in the forum.

Sailfish OS can be seen in multiple layers of software:

- at the bottom, the manufacturer of the chipset (e.g. Qualcomm) usually only produces drivers that are designed to run in Android (including some of the unusual libraries and interfaces that Android runs unlike other Linux-based OS)
- on top of the hardware, SailfishOS is a relatively standard GNU/Linux distribution (audio handled with PulseAudio, graphics with Waydroid, etc. has a standard C library)
- Jolla uses a special compatibility layer (libhybris) that makes the Android-only driver usable in the classic Linux distro such as SailfishOS.
- on top of the OS, you can run native Linux apps, mostly written using Qt and QML graphics.

To run native applications, no Android is required.

To run Android apps

- Jolla provides a compatibility layer called Android App Support. It's an LXC container (mostly like Docker containers) that you can optionally download and install from Jolla's application store.
- Once the container is started (either by autostart or manually), AOSP runs inside it (only Open-Source Android, not the proprietary parts including Google Play).
  - this makes some Android apps possible to run. You'll most probably rely on F-Droid to download (open source) Android apps.
- optionally you can install microG, it's an opensouce re-implementation of â€œGoogle Play Serviceâ€ the API that are normally provided by the proprietary Google Bit.
  - this makes a lot more apps possible to run. You can use the open-source Aurora (installable from F-Droid) to download and install apps from Google's own Store.
  - some apps might still refuse to run due to some specific things that the original proprietary Google Play Service provides but which are missing in microG (e.g., some Banking apps might complain about a rooted device because they want a full Google SafetyNet remote attestation).
- optionally, you can also install it instead of the original Google Play Service.
  - this kind of defies your de-googling attempts.
  - but this makes the stubborn e-banking app workable
  - also, you don't need to constantly run the container. You can keep the container not running, and only run the container for when you need *that one* Android app you can't find a replacement for

A big difference between a stock Android OS, and Android App Support is that you get more options to control what is running in the phone (even more if you opt to use microG instead of Google's own shit).
e.g.:

- you can manually start Android App Support yourself (and only run native SailfishOS apps the rest of the time).
- you can configure Android App Support to not start applications in the background on startup.
- you can configure microG to not automatically start an application when that gets a message/alert from the cloud servers.

Also:

- the AOSP running inside the container cannot see and use the hardware of the phone. For technical reasons, it needs special supports that expose stuff to Android.
- e.g., you can plug in a USB keyboard or pair a Bluetooth keyboard, and Android will see an available â€œinput deviceâ€.
- e.g.: you can pair Bluetooth speakers, and Android will see an available â€œaudio outâ€
- However, there is currently no support for Bluetooth itself, so you cannot use Android apps that need to directly talk to a Bluetooth device (you cannot use the speaker manufacturer's app to flash a firmware update to the Bluetooth speaker).

On smartphones not officially supported by Jolla, there is the open-source Waydroid solution, which is pretty similar to Android App Support.

- The main differences are in what adapters each has (e.g.: Waydroid currently cannot open multiple windows on SailfishOS)

So TL;DR: You get a whole range of options between:

- only the hardware drivers are written by Qualcomm
- more or less support for apps with some containers and optional service replacement (and you can prevent apps from running in the background).
- full(-er -ish) compatibility by replicating a full Android, but at least you get to shut it down when not in use.

Android app start triggers

This page is mostly based on DrYak's explanations on the forum.

Autostart when Android AppSupport starts

This is controlled with the Sailfish settings app.

Click *Settings* app â† ’ switch to *Apps* tab â† ’ select the app â† ’ use the " *Allow application background service to start on bootup* " checkbox (Needs to be off to prevent autostart).

This will prevent the app from autostarting whenever Android App Support is started.

Notifications

Apps can automatically start to show an alert pop-up (if you set some alarm clock thingy inside your app, e.g., a game reminding you that your crops are now ready to harvest or whatever). You need to access the Android settings to change that (either you can use the â€œ*Open Android Settings*â€ button in the Appâ€™s settings, or thereâ€™s a patch called â€œ*Launcher icon for android settings*â€ in the Patchmanager Webcatalog)

Either:
click *Settings* app â† ’ switch to *Apps* tab â† ’ select the app â† ’ click the â€œ*Open Android Settings* â€ button
then from the Appâ€™s Android settings â† ’ click *Notifications*
in the notifications setting, tweak to your wishes.

Or:
click *Android Settings* app â† ’ select *App & notifications* â† ’ select *Notifications* â† ’ scroll to the section titled *Recently sent*
tweak to your wishes

This will prevent applications from autostarting a showing toaster alerts about in-game events, google maps showing alerts about traffic in your area, etc.

Google Cloud Messaging (push notifications)

Whenever an app receives a push notification from the network, the app can start to fetch that network message and act on it. (e.g.: somebody writes you a chat message, so Googleâ€™s cloud pings your phone to tell you that thereâ€™s a chat message waiting for you, WhatsApp auto-starts, receives the chat message, and displays it) (or: your bank wants you to click on their 2FA app to confirm a transaction, so Cloud pings your phone, the 2FA app starts, and you get the â€œPlease confirmâ€ pop-up). The purpose is to avoid every single app on your phone wasting battery by constantly probing their server for updates, instead everything is centralized through the push notification system.

This can **only** be controlled if you use microG, as far as I know the proprietary Google Play Service donâ€™t let you configure that.

click *microG settings* app â†’ select *Cloud Messaging* â†’ select the app.
You can tweak 2 settings:

- *Start app on push message*: You can then select if the app is allowed to autostart to process alerts from the internet when they arrive, or if the alerts will wait until you manually start the app.
- *Allow registration*: If the app is even allowed to receive alerts from the internet.

If this function isnâ€™t turned on (you didnâ€™t even install microG), chat apps might not immediately get chat messages but only periodically when they directly contact their server, sometimes only when you bring the app into focus.(Usually I allow WhatsApp and the banking app, and kill everything else).

Share suggestions

Whenever you hit a *Share* button in an Android app, in addition to showing you a list of apps that can share that media type (e.g.: all chat apps can share JPEG photos) like when you click Share on SailfishOS, on Android some apps can auto start and generate a list of suggested share (e.g.: WhatsApp will suggest your top most frequently used contacts).

AliExpressâ€™ shop absolutely loves this function and constantly autostarts whenever you hit an Android share button.

There's no known way to disable this except either using the SailfishOS share option (it doesnâ€™t trigger autostart) or manually killing the autostarted application (e.g., in Crest).

microG installation (Google Play simulation)

Please refer to Installing microG on Sailfish OS in the forum.

Migrating Android app data

Transfer Signal using Signal-native Backup and Restore

The best source to install Signal is always direct from the website

https://signal.org/android/apk/#target

Export on the old phone

Open the Signal app and go to the settings. Navigate to *Chats* and then *Backups* (at the bottom). If they are disabled, turn them on.

signal-export-5-location.png

signal-export-6-choose.png

signal-export-7-passphrase.png

Note down the passphrase, you'll need it for the restoration!

Data transfer

Import on the new phone

Starting Signal the first time, it asks you if you want to set it up newly or restore from a backup.

On the next screen you will be asked if to restore from another device or from a file. Chose the latter:

signal-restore-1-choose.png

On the next screen click the button "Choose backup":

signal-export-6-choose.png

This opens the native Android file chooser. With the button top left you can switch between internal storage and memory card.

signal-restore-3-open.png

Choose your backup file.

The next screen will show you a few meta information on the file (age, size):

signal-restore-4-confirm.png

Tapping on "Restore backup" will ask you for the passphrase:

signal-restore-5-passphrase.png

Now, the restoration is in progress:

signal-restore-6-progress.png

Guide to be continued.....

Transfer WhatsAppusing native backup and restore

See https://faq.whatsapp.com/618575946635920/?locale=en_US&cms_platform=android

The backups are stored in `android_storage/Whatsapp`

Transfer Threema/ThreemaLibre data with native backups

Either Cloud backup or local backup

Cloud Backup: Chats included?