Developing
Introduction
App Development
Platform Development
Hardware Development
Collaborating
Open Source
Common Tasks
Reference
Introduction
Architecture
Software Packaging
Security
Internationalisation and Localisation
Qt
Core Areas & APIs
Toolchain
Android Compatibility
Tools
Introduction
Application SDK
Platform SDK
Hardware SDK
Other development tools
Hardware
Introduction
HW Adaptation Kit
Device Information
Development Services
Introduction
Open Build Service
Git Server
Image Creator
Webhooks
Bugzilla
Deployment Services
SSU
Store

Sailfish OS Cheat Sheet

Contents

Development Commands

The default IP of the device for USB connections is set in Settings > Developer tools > USB IP address, which is by default set to `192.168.2.15`. On this page we are referring to this with **device** which one can use alias for with following methods

A) (RECOMMENDED) add following to your `~/.ssh/config` file, which also makes the connection easier to handle if you are using multiple developer devices behind same IP address

```
# Dev devices which constantly change the ID
Host device
    User nemo
    HostName 192.168.2.15
    StrictHostKeyChecking no
    UserKnownHostsFile=/dev/null
    IdentitiesOnly yes
```

B) example to your `/etc/hosts` file, by adding following line there

```
192.168.2.15    device
```

Connect to the device over usb

```
sudo ifconfig usb0 up device
```

Log into the device. Define password in "Settings -> System settings -> Developer mode -> Set password".

```
ssh nemo@device
```

Change user to root

```
su       # if on SDK, or
devel-su # if on device
```

Remove changed IP from known_hosts

```
ssh-keygen -R device
```

Chroot to development environment

```
/srv/mer/sdks/sdk/mer-sdk-chroot
```

Update development environment

```
sb2 -t <target> -m sdk-install -R zypper ref
sb2 -t <target> -m sdk-install -R zypper update
```

Build project

```
mb2 -t <target> build # finds the spec under rpm
mb2 -t <target> -s rpm/<package>.spec build # specify spec yourself
```

Copy packages to the device

```
scp /RPMS/<package>.rpm nemo@device:
```

Listen to system logs

```
devel-su journalctl -fa # Sailfish
devel-su chroot /opt/alien/ /system/bin/logcat -v time # Android apps
```

See also logcat usage help article

Search log for keyword 'account' ignoring the case

```
devel-su journalctl | grep -i account
```

Open file (apk, media file, vcard, call number, etc.) with appropriate app.

```
xdg-open file # e.g. xdg-open image.jpg
```

List shared library dependencies

```
ldd /usr/lib/qt5/qml/modulepath/libmodule.so
```

List exported symbols

```
zypper in binutils && nm -D /usr/lib/library.so.0
```

Set DConf value

```
dconf write /desktop/meego/background/portrait/picture_filename \'/pathto/wa
llpaper.jpg\'
```

Print DConf value

```
dconf read /desktop/meego/background/portrait/picture_filename
```

List incoming hardware input events

```
evdev_trace -t
```

Find folders that take more than 100MB of space

```
du --all --one-file-system / | awk '{if($1 > 102400) print int($1/1024) "MB"
" " $2 }' # root partition
du --all --one-file-system /home | awk '{if($1 > 102400) print int($1/1024)
"MB" " " $2 }' # home partition
```

List RPM packages that take the most space in the system

```
rpm -qa --queryformat '%{size} %{name}\n' | sort -rn | more
```

Execute QML document.

```
pkcon install qt5-qtdeclarative-qmlscene # install qmlscene
ln -s /usr/lib/qt5/bin/qmlscene /usr/bin/qmlscene # add symbolic link to pat
h
qmlscene app.qml # run
```

Cleaning up the leftover packages from the system, first refresh the database, then check leftover and lastly remove wanted packages:

```
zypper ref
zypper packages --orphaned
zypper remove --clean-deps PACKAGENAME
```

Diagnostics

Saving logs is always good

```
devel-su journalctl -a > ~/saved.journal
```

Add -f to contiously listen to the log output:

```
devel-su journalctl -fa
```

The systemd journal is persistent over reboots in devel branch and in release branches if you want to have this behaviour you can edit /etc/systemd/journald.conf and set

```
Storage=persistent
```

By default, the systemd journal throttles output from particularly noisy processes, which can be frustrating when trying to debug an application. Preventing journald from throttling logging from a verbose process - edit /etc/systemd/journald.conf and set

```
RateLimitBurst=9999
RateLimitInterval=5s
```

Which will cause systemd to throttle journal output from any process which emits more than 9999 lines of output in any five second interval NOTE: after doing changes for /etc/systemd/journald.conf you should reboot the device.
Various processes can be made more verbose by setting certain environment variables:

```
 QT_LOGGING_RULES="*.debug=true"   # any application or service using Qt Cate
gorized Logging
 MSYNCD_LOGGING_LEVEL=8   # any Buteo sync plugin
 SSO_LOGGING_LEVEL=3 SSOUI_LOGGING_LEVEL=3  # Accounts&SSO services
 CONTACTSD_DEBUG=1  # contactsd instant messaging roster synchronisation daem
on
 QTCONTACTS_SQLITE_TWCSA_TRACE=1 QTCONTACTS_SQLITE_TRACE=1  # qtcontacts-sqli
te backend debug output
 KCALDEBUG=1  # calendar application and plugins, mkcal and KCal debug output
 LIPSTICK_COMPOSITOR_DEBUG=1  # homescreen debug output
```

Various processes can be made more verbose by editing certain configuration files and rebooting:

```
 /home/<user>/.config/QtProject/Messageserver.conf  # email, QtMessagingFrame
work configuration file
 /home/<user>/.config/eas-sailfish.conf  # Exchange ActiveSync plugin configu
ration file
```

For example, to make the Exchange ActiveSync plugin fully verbose, first ensure that journald won't throttle logging output (see the notes on editing /etc/systemd/journald.conf above) and then ensure that the /home/<user>/.config/eas-sailfish.conf file contains the following:

```
 [logging]
 Sailfish.eas.debug=dwc
 Sailfish.eas.warning=dwc
 Sailfish.eas.error=dwc
 Sailfish.easwbxml=dwc
 Sailfish.easnetwork=dwc
 Sailfish.easverbose=dwc
```

You can also force sailfish-eas to log to a specific file by setting e.g.
`Sailfish.logfile=/home/nemo/sailfisheas.log` in that conf file.
To make the QMF (generic email) component fully verbose, and output logs to the /home/<user>/qmf.log file, modify /home/<user>/.config/QtProject/Messageserver.conf to contain the following:

```
 [FileLog]
 Enabled=1
 Path=/home/<user>/qmf.log

 [LogCategories]
 IMAP=1
 Messaging=1
 POP=1
 SMTP=1

 [StdStreamLog]
 Enabled=0

 [Syslog]
 Enabled=0
```

Some processes can be made more verbose by installing specific "tracing" packages which configure the service to be more verbose when installed (via `devel-su pkcon install <pkgname>`). Some examples include:

```
 connman-tracing
 bluez5-tracing
 connectionagent-qt5-tracing
```

Unfortunately, due to the heterogeneous nature of the packages in Sailfish OS, there is no unified way to enable verbose debug logging, and so the contributor will in some cases have to read the source code to figure out what they need to do to make debug logging work. It may even require setting some #define in source code, and rebuilding the package.

Home Screen and Compositor diagnostics

Lipstick debugs can be enabled by adding LIPSTICK_COMPOSITOR_DEBUG=1 to /var/lib/environment/compositor/*.conf.

Now restart lipstick and you have a small box at the bottom of the screen for debugging the top most window. "Dump" button outputs data of the top most window to the journal. "Expose" button shows current windows.

In case screen is locked or/and touch is not responding but you have an access to the device. Top most window can be dumped like:

```
 dbus-send --type=method_call --print-reply --dest=org.nemomobile.compositor.debug /debug org.nemomobile.compositor.debug.dump
```

Similarly use LIPSTICK_COMPOSITOR_TOUCH_DEBUG=1 to debug touch issues with edge swipes.

Calling diagnostics

Enabling voicecall manager debug logs requires modifying source codes, rebuilding and installing the project packages:

```
 voicecall/lib/src/common.h -> #define WANT_TRACE
```

Enabling Phone app logs requires modifying source code line:

```
 /usr/share/voicecall-ui-jolla/VoiceCallManager.qml -> enableDebugLog: true
```

Power Diagnostics

Go to your device command line. Download Power issues reporting script on your device

```
 curl -o power-diagnostics.sh https://git.merproject.org/mer-core/statefs-providers/blob/master/tools/power-diagnostics.sh && chmod a+x power-diagnostics.sh
```

and run it

```
 devel-su ./power-diagnostics.sh > /home/<user>/power-state-report.txt
```

Attach resulting /home/<user>/power-state-report.txt file and journal to your bug report.

Backup Diagnostics

On the target device go to the command line. Download there Backup status reporting script and make it executable.

```
 curl -o the-vault-storage-report.sh https://git.merproject.org/mer-core/the-vault/blob/master/tools/the-vault-storage-report.sh && chmod a+x the-vault-storage-report.sh
```

Execute the command in privileged mode.

```
 devel-su -p ./the-vault-storage-report.sh > /home/<user>/backup-status-report.txt
```

Attach resulting /home/<user>/backup-status-report.txt file to your bug report.

Browser Diagnostics / Debugging

Console API messages

- EMBED_CONSOLE=1 environment variable be used to log messages to terminal / journal
- Simply execute browser with EMBED_CONSOLE=1

```
 EMBED_CONSOLE=1 sailfish-browser
```

- Errors in JavaScript etc are logged

NSPR module logging

- Environment variable NSPR_LOG_MODULES can be used to enable various module logs
- A comma separated list with verbosity
- Verbosity levels are: 0 = Disabled, 1 = Error, 2 = Warnings, 3 = Info, 4 = Debug, 5 = Verbose

- EmbedLite and EmbedLiteTrace are log components of embedlite
- Grep is your friend when trying to find the correct module if you're not aware of those (LazyLogModule | NSPR_LOG_MODULES)
- For instance "Layers" can be used to analyze problems related to composited layers
- Example

```
 NSPR_LOG_MODULES="AudioStream:5,MediaFormatReader:5,GStreamerFormatHelper:5,
MediaSource:5" sailfish-browser youtube.com
 Start video playback
```

- Note: some modules require debug build out of Gecko (xulrunner)
- Note: NSPR_LOG_MODULES="All:5" will gather all logs but the end result will be quickly huge

Debugging User-Agent problems

- Current user-agent is "Mozilla/5.0 (Linux; U; Sailfish 3.0; Mobile; rv:45.0) Gecko/45.0 Firefox/45.0 SailfishBrowser/1.0"
- User-agent string can be overriden site specifically
- Debug user agents with CUSTOM_UA environment variable
- Input for the over the air updates is located in browser's git repository
- There can be full user agents and replacements done with regular expression to fix user-agent strings
- Example fix for slashdot user-agent

Debugging omni.ja of gecko

```
 mkdir /home/<user>/omni-mytest
 cd /home/<user>/omni-mytest
 cp /usr/lib/xulrunner-qt5-45.9.1/omni.ja .
 unzip omni.ja
 rm omni.ja

 // Edit jsm or js file or copy over from your close

 zip -qr9XD omni.ja *
 devel-su mv omni.ja /usr/lib/xulrunner-qt5-45.9.1/

 If you need to revert omni changes just have a backup of your omni.ja or rei
nstall xulrunner package
```

Restart System Services

Restart user session

```
 systemctl restart user@100000
```

Restart networking. Warning! Disconnects your SSH connection.

```
 systemctl restart connman.service
```

Restart home screen

```
 systemctl --user restart lipstick
```

Restart keyboard

```
 systemctl --user restart maliit-server
```

Restart Phone application

```
 systemctl --user restart voicecall-ui-prestart
```

Restart Phone middleware

```
 systemctl restart ofono
 systemctl-user restart voicecall-manager
```

Package Handling

Root rights required

```
devel-su
```

Show SW version

```
version
```

Update software

```
version --dup
```

Pkcon commands

```
pkcon refresh   # Update repositories

pkcon search name [PACKAGE_NAME]
pkcon install [PACKAGE_NAME]
pkcon get-details [PACKAGE_NAME]
pkcon remove [PACKAGE_NAME]
pkcon update [PACKAGE_NAME]

pkcon install-local [FILE_NAME]

pkcon repo-list
pkcon repo-enable [REPO_ID]
pkcon repo-disable [REPO_ID]

pkcon       # Lists the full command syntax and options.
```

Zypper commands for SDK (pkcon is preferred on device)

```
zypper lr # list repositories
zypper ref # update repositories
zypper update # update packages
zypper se packagename # search packages
zypper in packagename # install packages
zypper info packagename # check package information
zypper info -t pattern patternname # check pattern information
zypper verify # check dependencies
```

RPM commands

```
rpm -e <package> # remove package
rpm -ql <package-name> # list files in package
rpm -qlP <file> # list files in package
rpm -qf <file> # find out what package file belongs to
rpm -qpR <rpm-file> # find out package dependencies
rpm -qR <package-name> # find out package dependencies
rpm -q --whatrequires <package> # find out reverse dependencies
rpm -qa | xargs rpm -qR | grep -b5 <package> # query all packages, check whe
ther they depend on package
rpm -U --oldpackage --replacepkgs  --replacefiles <package> # reinstall rpm
package
rpm -qa | awk '{print $0" "$0}' | xargs printf "echo PACKAGE: %s && rpm -q -
-scripts %s\n" | sh # List RPM scripts by package
```

Clear corrupted rpm database (as root):

```
rm -rf /var/lib/rpm/__db* ; rpm --rebuilddb
```

Clearing, Importing and Exporting User Data

Phone

Install commhistory-tool if not already installed.

```
pkcon install  libcommhistory-qt5-tools
```

Clear call logs, run as user

```
commhistory-tool deleteall -calls
```

Add call logs data, run as user

```
commhistory-tool import-json calllogs.json
```

Restart Phone application to see changes in effect.

```
pkill voicecall-ui
```

Messages

Remove all message conversations, run as user

```
commhistory-tool deleteall -groups
```

Import message data, run as user

```
commhistory-tool import-json messages.json
```

Restart Messages application to see changes in effect.

```
pkill jolla-messages
```

People

Install vcardconverter if not already installed.

```
pkcon install nemo-qml-plugin-contacts-qt5-tools
```

Import contacts from vCard

```
devel-su -p vcardconverter contacts.vcf
```

Export local contacts to vCard

```
devel-su -p vcardconverter --export contacts.vcf
```

Calendar

Install icalconverter if not already installed

```
pkcon install nemo-qml-plugin-calendar-qt5-tools
```

Import events from iCal

```
devel-su -p icalconverter import calendar.ics
```

Import events using Calendar import page

```
 dbus-send --print-reply --type=method_call --dest=com.jolla.calendar.ui /com
/jolla/calendar/ui com.jolla.calendar.ui.importFile string:/home/<user>/calen
dar.ics
```

Export local calendar events to iCal

```
devel-su -p icalconverter export calendar.ics
```

Browser

Set the home page.

```
dconf write /apps/sailfish-browser/settings/home_page "'http://jolla.com'"
```

Media

Transfer content to the device

```
scp *.jpg nemo@device:Pictures
scp *.mp4 nemo@device:Videos
scp *.pdf nemo@device:Documents
scp *.ogg nemo@device:Music
```

Ambiences

Set image as the ambience.

```
dbus-send --session --print-reply --dest=com.jolla.ambienced /com/jolla/ambie
nced com.jolla.ambienced.setAmbience string:"file://home/<user>/Pictures/imag
e.jpg"
```

Homescreen

Reset order of apps in Homescreen launcher.

```
rm /home/<user>/.config/lipstick/applications.menu
```

Weather

Remove weather locations

```
rm /home/<user>/.local/share/sailfish-weather/weather.json
```

Blocking Device Suspend

Install mcetool

```
zypper in mce-tools
```

Disable late suspend

```
mcetool -searly
```

Disable early suspend

```
mcetool -sdisabled
```

Restore normal suspend policy

```
mcetool -senabled
```

Screen Brightness

Install mcetool

```
zypper in mce-tools
```

Set brightness setting to maximum value

```
mcetool -b5
```

Disable screen dimming when home screen or applications are open

```
mcetool -Don
```

Disable screen dimming when the lock screen is open

```
mcetool -tdisabled
```

Go back to normal behavior

```
mcetool -Doff -tenabled
```

For problem with unusually dark display, try disabling als-based display brightness filtering

```
mcetool -gdisabled
```

Reset all mce values to their defaults

```
systemctl stop mce.service
rm /var/lib/mce/builtin-gconf.values
systemctl start mce.service
```

Show Dialogs

Alarm Dialog

Show timer alarm in 3 seconds (ticker=3).

```
timedclient-qt5 -b'TITLE=button0' -e'APPLICATION=nemoalarms;TITLE=Timer;type
=countdown;timeOfDay=1;triggerTime=1395217218;ticker=3'
```

Show clock alarm in 3 seconds (ticker=3).

```
timedclient-qt5 -b'TITLE=button0' -e'APPLICATION=nemoalarms;TITLE=Clock;type
=event;timeOfDay=772;ticker=3'
```

Connection Dialog

```
dbus-send --print-reply --type=method_call --dest=com.jolla.lipstick.Connect
ionSelector /  com.jolla.lipstick.ConnectionSelectorIf.openConnection string:
```

USB Dialog

Connect cable. Make sure "Settings -> USB -> Default USB mode" is set to "Always ask".

Unresponsive App Dialog

Make app unresponsive by stopping it's execution.

```
kill -SIGSTOP `pgrep appname` # e.g. jolla-messages
```

Continue execution by calling

```
kill -SIGCONT `pgrep appname` # e.g. jolla-messages
```

Side Loading Dialog

```
xdg-open package.rpm
```

Call Request Dialog

```
xdg-open "tel://0123456789"
```

Supplementary Service Dialog

Type USSD code with Phone dialer, for example "*#31#" shows the status of caller line restriction.

Audio Warning Dialog

Change headset audio warning timeout by adding following lines to /etc/pulse/mainvolume-listening-time-notifier.conf.

```
"timeout = 1
sink-list = sink.primary
mode-list = lineout"
systemctl --user restart pulseaudio.service
```

Now play a song over 1 minute with normal headset in Media Player to see a warning dialog.
Reset too-loud volume warning.

```
/usr/bin/dconf write /desktop/<user>/audiowarning true
```

Now play a song over headset and turn the volume to maximum to see a warning dialog.

The Other Half Installation Dialog

Sign in to Jolla store. Attach new TOH back cover.