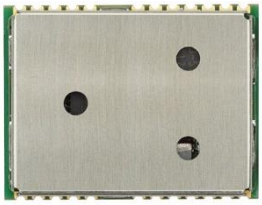
Android RIL

Production delivery – Source code

Application Note



Abstract

This document describes how to build and use the RIL library for u-blox cellular modules on the Android operating system.

www.u-blox.com

UBX-13002041 - R21

Document Information

Title	**Android RIL**	
Subtitle	Production delivery – Source code	
Document type	Application Note	
Document number	UBX-13002041	
Revision and date	R21	24-Oct-2018
Disclosure Restriction		

This document applies to the following products:

Product name
TOBY-L2 series
TOBY-L4 series
TOBY-R2 series
LARA-R2 series
SARA-R4 series
LISA-U2 series
SARA-U2 series
LISA-C2 series
LEON-G1 series
SARA-G3 series

Contents

1 Introduction

☞ An index finger points out key information pertaining to integration and performance.

⚠ A warning symbol indicates actions that could negatively impact performance or damage the device.

☞ This revision of the application note applies to the Android RIL v11.00 and onwards.

The software was developed for the following Android versions (AOSP) and platforms:

- Android 2.3 (Gingerbread): BeagleBoard-XM
- Android 4.0 (Ice Cream Sandwich): BeagleBoard-XM
- Android 4.1, 4.2, 4.3 (Jelly Bean): PandaBoard
- Android 4.4 (KitKat): PandaBoard
- Android 5.0 (Lollipop): Nexus 5
- Android 6.0 (Marshmallow): Nexus 5
- Android 7.0 (Nougat): Nexus 5X
- Android 8.0 (Oreo): Nexus 5X

See the RIL release note for the list of Android software deliveries and interfaces supported by u-blox cellular modules.

In the following chapters, <name> indicates a parameter that can:

- be customized
- be set corresponding to the system configuration
- provide software version

The following chapters describe the production packages delivered by u-blox.

An overview of the system setup is provided as well as the procedure to perform a log.

The following symbols are used to highlight important information within this document:

☞ The RIL source code provided by u-blox is for reference purposes only.

☞ u-blox RIL does not control or manage the device (e.g. u-blox module or any other peripheral) GPIOs, such as to power off/on or reset the module. Power management is under the responsibility of the Android OS and the kernel subsystem.

⚠ u-blox assumes no responsibility for inappropriate use of RIL by customers.

2 Production delivery

2.1 Software release

The delivery consists of the RIL library source code.

⚠ Android RIL does not offer a standard interface for powering off the module, so it is necessary to modify the Android Java framework application that manages system power-off and sends the AT+CPWROFF command.

2.2 Android delivery contents

☞ The Android delivery contents structure for all Android versions is described below. If the `RIL_sc_<version>.zip` does not include a component mentioned in the delivery contents, it means that the package has been updated for that release and the component is no longer required.

2.2.1 Android 2.3 delivery contents

The RIL production delivery is provided in the `RIL_sc_<version>.zip` compressed file with the content structured as follows:

`RIL_sc_<version>.zip`	
`ril_sc_<version>`	Source overlay for Android platform
`external/ppp/`	Data connection files
`hardware/ril/rapid_ril/`	RIL core directory
`system/core/init/property_service.c`	Set GPRS service property
`system/core/rootdir/Android.mk`	Compilation script
`system/core/rootdir/ueventd.rc`	Device permission
`system/core/rootdir/etc/init.gprs-pppd`	Script for starting PPP service
`system/core/rootdir/etc/ppp/`	PPP script directory
`system/core/rootdir/etc/rril-repo.sh`	RIL script
`system/core/rootdir/etc/rril/repository.txt`	RIL parameters

2.2.2 Android 4.0, 4.1, 4.2, 4.3, 4.4 deliveries contents

The RIL production delivery is provided in the `RIL_sc_<version>.zip` compressed file with the content structured as follows:

`RIL_sc_<version>.zip`	
`ril_sc_<version>`	Source overlay for Android platform
`build/target/product`	Device configuration folder files
`external/ppp/`	Data connection files
`hardware/ril/ublox_ril/`	RIL core directory
`system/core/rootdir/Android.mk`	Compilation script
`system/core/init/property_service.c`	Set RIL services property
`system/core/liblog/logd_write.c`	Log configuration files
`system/core/rootdir/init.ublox.rc`	u-blox Init.rc file
`build/target/product/rootdir/core_ublox.mk`	Make file include all scripts
`build/target/product/rootdir/etc/init.gprs-pppd`	Script to start PPP service
`build/target/product/rootdir/etc/uril-repo.sh`	RIL script
`build/target/product/rootdir/etc/uril/repository.txt`	RIL parameters
`build/target/product/rootdir/etc/init_data`	Script to start data service in Router/Bridge mode
`build/target/product/rootdir/etc/stop_data`	Script to stop data service in Router/Bridge mode
`build/target/product/rootdir/etc/ppp`	Scripts for data connection

2.2.3 Android 5.0, 6.0, 7.0, 8.0 delivery contents

The RIL production delivery is provided in the `RIL_sc_<version>.zip` compressed file with the content structured as follows:

`RIL_sc_<version>.zip`	
`ril_sc_<version>`	Source overlay for Android platform
`external/ppp/`	Data connection files
`hardware/ril/ublox_ril/`	RIL core directory
`system/core/liblog/logd_write.c`	Log configuration files
`system/core/rootdir/init.ublox.rc`	u-blox Init.rc file
`build/target/product`	Device configuration folder files
`build/target/product/rootdir/core_ublox.mk`	Make file include all scripts
`build/target/product/rootdir/etc/ppp`	Scripts for data connection
`build/target/product/rootdir/etc/uril/repository.txt`	RIL parameters
`build/target/product/rootdir/etc/init.gprs-pppd`	Script to start PPP service
`build/target/product/rootdir/etc/init_data`	Script to start data service in Router/Bridge mode
`build/target/product/rootdir/etc/stop_data`	Script to stop data service in Router/Bridge mode
`build/target/product/rootdir/etc/init_data_android_6`	Scripts in Router/Bridge mode(Android 6,7,8)
`build/target/product/rootdir/etc/stop_data_android_6`	Scripts in Router/Bridge mode(Android 6,7,8)

3 Build source code

☞ This section applies to the Linux operating system.

- Create the build directory using the following commands, where `<android_root>` is the directory name:

```
mkdir <android_root>
cd <android_root>
```

- Download an Android distribution for the platform. Follow the tutorial, manual, etc. related to the platform's distribution.

⚠ Android distribution must be one of the deliveries listed in section 1. Full functionality of software in other Android OS versions is not guaranteed.

- Create a directory and decompress the RIL file

⚠ The password provided from u-blox to decrypt the compressed RIL file is needed.

```
mkdir <ril_dir>
cd <ril_dir>
cp <path_of_ril>/RIL_sc_<version>.zip .
unzip -P <ril_password> RIL_sc_<version>.zip
```

- Copy the RIL source code into the Android distribution
- For Android 4.0 / 4.1 / 4.2 / 4.3 / 4.4 / 5.0 / 6.0 / 7.0 / 8.0 delivery:

```
cp -pvRf ril_sc_<version>/Android<version>/external/ppp/*
<android_root>/external/ppp
cp -pvRf ril_sc_<version>/Android<version>/system/*  <android_root>/system
cp -pvRf ril_sc_<version>/Common/build/target/product/
<android_root>/build/target/product
cp -pvRf ril_sc_<version>/Common/system/core/rootdir/
<android_root>/system/core/rootdir
cp -pvRf ril_sc_<version>/Common/hardware/ril/ublox_ril <android_root>/hardware/ril/
```

☞ The delivery overwrites or adds some Android system files.

- Modify the kernel configuration as in appendix A and configure the script files module connection to the Android platform as described in:
  - Appendix B: Android 4.1/4.2/4.3/4.4/5.0/6.0/7.0/8.0 module configuration
  - Appendix C: for Android 4.0 (Ice Cream Sandwich)
  - Appendix D: for Android 2.3 (Gingerbread)

☞ For module specific modifications, see appendix K.

☞ For information on MUX interfacing, see the Multiplexer Implementation Application Note [5].

- Build the Android system for the device or for the emulator

☞ Use the tutorial, manual, etc. related to the platform's distribution.

- Insert a microSD card (minimum 2 GB) into the PC. Determine the SD card device name. Prompt `dmesg` command on a system shell. An example of the result of this command is below

```
[85560.292608] sd 6:0:0:2: [sdd] Assuming drive cache: write through
[85560.292613] sd 6:0:0:2: [sdd] Attached SCSI removable dis
```

The device name is written in the brackets (e.g. sdd)

- The next step partitions and populates the microSD card with the root file system, bootloader, etc.
- Use the command provided in the platform distribution to create an SD card for the platform

⚠ This step can erase the hard disk drive if the SD card device name entered after the script name is incorrect.

- Insert the SD card into the board's SD slot
- Connect the board with the u-blox cellular module using one of the supported interfaces
- Power on both devices

4 Debug the RIL

4.1 Host PC configuration

4.1.1 Windows configuration

- On a Windows host, download and extract the Android SDK [4]
- Update the Android SDK using the command:

```
<home>\android-sdk-windows\tools\android.bat
```

- Download the USB driver for Android [3] and extract it into

```
<home>\android-sdk-windows\usb-drivers\
```

- Modify the section [Google.NTx86] of the `android_winusb.inf` in the `<home>\android-sdk-windows\usb-drivers\usb_driver_r03-windows` directory with the following lines:

```
;
;TI EVM
%SingleAdbInterface%        = USB_Install, USB\VID_18D1&PID_9018
%CompositeAdbInterface%     = USB_Install, USB\VID_18D1&PID_9018&MI_01
```

- Run a shell and type the following command:

```
echo 0x18D1 > "%USERPROFILE%\.android\adb_usb.ini"
```

- Install the downloaded USB drivers for the Android OS
- Connect the device to the PC
- The device is now ready to start communications

4.1.2 Linux configuration

- On a Linux host, download and extract the Android SDK [4]
- On Linux, type the following commands in a shell:

```
sudo su <passwd>
mkdir ~/.android
vi ~/.android/adb_usb.ini
echo "0x0451" > ~/.android/adb_usb.ini
```

- Connect the device to the PC
- The device is now ready to start communications

4.2 RIL log

Set up the Android Debugger (ADB) as mentioned above. By means of the ADB, it is possible to choose which RIL radio- and telephony-related log messages to receive by using the following command:

```
adb logcat  -v time -b radio
```

The above command generates a log output like the following example:

```
12-21 11:42:22.654 D/RILD  (33): RILD: rilLibPath='/system/lib/librapid-ril-core.so'
12-21 11:42:22.654 D/RILD  (33): Connecting to 'qemud' socket
12-21 11:42:22.693 I/RIL-repo(51): /data/rril/repository.txt exists
```

```
12-21 11:42:23.883 I/RILR  (33): Log level [2]
12-21 11:42:23.883 I/RILR  (33): CSystemManager::CSystemManager() - Enter
12-21 11:42:23.883 I/RILR  (33): CSystemManager::CSystemManager() - Exit
12-21 11:42:23.883 I/RILR  (33): CSystemManager::InitializeSystem() - Enter
12-21 11:42:24.002 I/RILR  (33): CSystemManager::InitializeSystem() - Retrieved Last CLIP
Value: 0x0
12-21 11:42:24.114 I/RILR  (33): CSystemManager::InitializeSystem() - Retrieved Last CLIR
Value: 0x0
12-21 11:42:24.243 I/RILR  (33): CSystemManager::InitializeSystem() - Retrieved Last COLP
Value: 0x0
12-21 11:42:24.342 I/RILR  (33): CSystemManager::InitializeSystem() - Retrieved Last COLR
Value: 0x0
12-21 11:42:24.683 I/RILR  (33): RIL_Init,init not finish:0
12-21 11:42:25.533 I/RILR    (   33): CChannelBase::OpenPort() - Opening COM
Port=[/dev/socket/qemud]  g_bIsSocket=[1]
12-21 11:42:25.533 I/RILR   (33): CPort::OpenSocket('/dev/socket/qemud', 'gsm')
12-21 11:42:25.533 I/RILR   (33): CFile::Open() - Enter
12-21 11:42:25.533 I/RILR   (33): CFile::Open() : pszFileName=[/dev/socket/qemud]
12-21 11:42:25.533 I/RILR   (33): CFile::Open() : fIsSocket=[1]
12-21 11:42:25.533 I/RILR   (33): CFile::Open() - Exit  m_fInitialized=[1]
12-21 11:42:25.533 I/RILR   (33): CPort::OpenSocket() - Port is open!!
12-21 11:42:25.533 I/RILR   (33): CPort['/dev/socket/qemud'] << 'gsm'
12-21 11:42:25.684 I/RILR   (33): RIL_Init,init not finish:0
```

It is possible to provide the general Android log messages using the following command:

```
adb logcat  -v time
```

The above command generates a log output like the following example:

```
12-21 11:42:22.453 I/DEBUG   (   31): debuggerd: Dec  2 2011 16:20:35
12-21 11:42:22.503 D/qemud   (   39): entering main loop
12-21 11:42:22.513 I/Netd    (   30): Netd 1.0 starting
12-21 11:42:22.564 I/Vold    (   29): Vold 2.1 (the revenge) firing up
12-21 11:42:22.564 D/Vold    (   29): USB mass storage support is not enabled in the kernel
12-21 11:42:22.564 D/Vold    (   29): usb_configuration switch is not enabled in the kernel
12-21 11:42:22.564 D/Vold    (   29): Volume sdcard state changing -1 (Initializing) -> 0 (No-
Media)
12-21 11:42:22.854 D/qemud   (   39): fdhandler_accept_event: accepting on fd 9
12-21 11:42:22.854 D/qemud   (   39): created client 0xe078 listening on fd 10
12-21 11:42:22.854 D/qemud   (   39): client_fd_receive: attempting registration for service
'boot-properties'
```

4.3 Logger configuration

The logging level can be set in the **Group Logging** section of the `repository.txt` file by changing the value of the **LogLevel** field.

File path: `build/target/product/rootdir/etc/uril/repository.txt`

Allowed values: Table 1 lists the various values that are currently supported.

[LogLevel]	Logging Levels
1	Verbose
2	Info
3	Warning
4	Critical

Table 1: Supported logging levels

Example: following lines are present in the `Repository.txt` file to define the Log Level.

```
// Log levels
Group Logging
LogLevel        2   // 4: Critical, 3: Warning, 2: Info, 1: Verbose
```

4.4 PPPD debug log

To enable the PPPD log, **debug** should be uncommented in the corresponding files. `build/target/product/rootdir/etc/ppp/gprsX`

(where X is the PPPD interface used by RIL)

```
# uncomment the following line to debug pppd
debug
ttyACM1
crtscts
nodetach
usepeerdns
defaultroute
ipcp-max-failure 30
connect "chat -t 15 -f /system/etc/ppp/chat-isp1"
```

Appendix

A Kernel configuration

Configure the kernel to support the u-blox cellular module and the applicable interface connection.

A.1 USB kernel configuration

Enable the kernel configuration as follows:

```
Device Drivers
   USB Support
      <*>USB Modem (CDC ACM) support
```

A.2 UART kernel configuration

This device is enabled in the default kernel configuration.

A.3 SPI kernel configuration

Enable the kernel configuration as follows:

```
Device Drivers
   SPI Support
      <*>Debug support for SPI driver
      <*>GPIO-based bitbanging SPI Master
      <*>McSPI driver for OMAP24xx/OMAP34xx
```

⚠ Implement the dedicated SPI protocol (see the *SPI interface Application note* [1]) on the SPI kernel driver to correctly work with the cellular modules supporting the SPI interface.

A.4 RNDIS / ECM kernel configuration

Enable the kernel configuration as follows:

```
Device Drivers
   Network device support
USB Network Adapters
   <*>Multi-purpose USB Networking Framework
   <*>CDC Ethernet support
   <*>Host for RNDIS and ActiveSync devices
```

⚠ This configuration is required only for the TOBY-L2 configured with 1 CDC-ACM port and 1 RNDIS (factory-programmed configuration) or 1 ECM (AT+UUSBCONF=2).

A.5 PPP kernel configuration

Enable the kernel configuration as follows:

```
Device Drivers
   Network device support
      <*>PPP (point-to-point protocol) support
      <*>PPP support for async serial ports
      <*>PPP support for sync tty ports
      <*>PPP Deflate compression
```

☞ Check if the “chat” package is included in the build. If not, add it using the following line in one of the .mk files included in the build.

```
PRODUCT_PACKAGES += chat
```

B Android 4.1/4.2/4.3/4.4/5.0/6.0/7.0/8.0 module configuration

☞ TOBY-L2 series only: See appendix G.1 for the automatic profile configuration.

B.1 core.mk configuration

Following modification are required to include core_ublox.mk in android source.

File Path: build/target/product/core.mk

Modifications: Add below mentioned line at end of core.mk.

```
#***u-blox Modifications***#
$(call inherit-product, $(SRC_TARGET_DIR)/product/core_ublox.mk)
#***u-blox Modifications***#
```

B.2 ueventd.rc configuration

This file is used to create or remove the device node (/dev/xxx) by receiving uevent messages from the kernel. To create the node for the u-blox modules, add the following configuration in ueventd.[platform].rc or ueventd.rc. Where [platform] labels depend on the Board Support Package (BSP), following these permissions:

Modifications:

For USB/ttyACM Mode:

```
/dev/ttyACM*          0660  radio  radio   #[if using CDC-ACM Interface]
```

For UART/MUX Mode:

```
/dev/<serial device>  0660  radio radio   #[if using VCP Interface e.g. ttyUSB, ttyS or ttymxc]
/dev/pts*             0660  radio radio   #[if using pts Interface with MUX]
```

B.3 device.mk

File Path: /device/<platform>/device.mk

Modifications:

- Set port path and ril lib path using following lines.

```
rild.libpath=/system/lib64/librapid-ril-core.so <For 64-bit architecture>
rild.libpath=/system/lib/librapid-ril-core.so   <For 32-bit architecture>
rild.libargs=-a/dev/ttyACM0
```

☞ For TOBY-L4 series only use interface name ttyACM2.

☞ For SARA-R4 series only use interface name ttyUSB1.

- Update following property to set default network type to global (LTE, GSM, WCDMA).

```
ro.telephony.default_network=9
```

B.4 init.rc configuration

Following modifications are required to include init.ublox.rc in Android source.

File Path: system/core/rootdir/init.rc

Modifications: Add below mentioned line at top of `init.rc`.

```
#***u-blox Modifications***#
import /init.ublox.rc
#***u-blox Modifications***#
```

B.5 `init.ublox.rc` configuration

The `init.ublox.rc` file contains all the necessary information used for the RIL integration. All the services, properties, permissions used for u-blox RIL are mentioned in this file.

This file is self-explanatory and contains all the steps used for the RIL initialization. Follow these steps to configure this file as per requirements.

☞ In the RIL delivery package the **init.ublox.rc** file at the following path: "ril_sc_<version>/Common/system/core/rootdir".

☞ [For Android 5 and above] After the integration follow the steps mentioned in init.ublox.rc (the default setting is for (usb)ttyACM).

B.6 SEPolicy configuration

Add the following lines in their respective files to allow the ril and ppp daemons access to the required data, system and device files. All of the policy files are located in the folder shown below depending on the Android version being used.

- ANDROID_SOURCE/external/sepolicy **(For Android 5 and 6)**
- ANDROID_SOURCE/system/sepolicy **(For Android 7 and onwards)**

B.6.1 `file_contexts`

Add the required files to their respective contexts as shown below so that any service/process can access them according to the permissions granted to it (if required).

Binaries	Contexts
/system/bin/init.gprs-pppd	u:object_r:rild_exec:s0
/system/bin/stop_pppd	u:object_r:rild_exec:s0
/system/bin/init_data	u:object_r:rild_exec:s0
/system/bin/stop_data	u:object_r:rild_exec:s0
/system/bin/ip-up	u:object_r:rild_exec:s0
/system/bin/ip-down	u:object_r:rild_exec:s0

Table 1: Context settings for binaries

Data Files	Contexts
/data/uril(/.*)?	u:object_r:radio_data_file:s0

Table 2: Context settings for data files

Devices	Contexts
#USB Interface (If using CDC-ACM Interface)	
/dev/ttyACM[0-5]*	u:object_r:radio_device:s0
#UART Interface (For VCP Interface (ttyS, ttyUSB, ttymxc))	
/dev/<serial device>	u:object_r:radio_device:s0
#pts Channel (For GSMMUX Interface)	
/dev/pts*	u:object_r:radio_device:s0

Table 3: Context settings for device files

B.6.2 property_contexts

Add the required properties in their respective contexts as shown below so that any service/process can access them according to the permissions granted to it (if required).

Network Properties	Contexts
net.data	u:object_r:net_radio_prop:s0
net.interfaces.defaultroute	u:object_r:net_radio_prop:s0
net.uril.repository	u:object_r:net_radio_prop:s0
net.usb	u:object_r:net_radio_prop:s0

Table 4: Context settings for network properties

B.6.3 rild.te

```
# allows rild (via init.gprs-pppd) to execute the ppp daemon
allow rild ppp_exec:file rx_file_perms;
# allows rild (via ppp daemon) access to the /dev/ppp device
allow rild ppp_device:chr_file rw_file_perms;
# allows the ril daemon to execute scripts like init.gprs-pppd etc.
allow rild rild_exec:file rx_file_perms;
# allows ril daemon to set properties in the default context
allow rild ctl_default_prop:property_service set;
# required to eliminate the kernel support error by ppp
allow rild self:capability { dac_override };

# required if MUX is being used
allow rild devpts:chr_file { rw_file_perms };

# For Android 7.0 and later
allow rild toolbox_exec:file rx_file_perms;

# For Android 8.0 and later
set_prop(rild, net_radio_prop);

# For Android 7 only: allow permission rild to open socket via netcfg
allow rild self:packet_socket create_socket_perms;
allow rild self:netlink_kobject_uevent_socket create_socket_perms;

# For Android 8 only: allow permission rild to open socket via netcfg
allow rild self:packet_socket create_socket_perms_no_ioctl;
allow rild self:netlink_kobject_uevent_socket create_socket_perms_no_ioctl;

# For Android 8 only: If using MUX service
allowxperm rild devpts:chr_file ioctl { ppp_ioctls TCSETSF TIOCMBIS TIOCMBIC
TIOCEXCL TIOCSETD TIOCNXCL };
```

Add the following lines to the `rild.te` file:

C Android 4.0 module configuration

C.1 USB

To enable the USB module connection, configure the `AndroidProducts.mk` and `init.rc` files as follows.

C.1.1 `AndroidProducts.mk`

☞ Not required for RIL release 09.00 and above.

Add the following lines to the `AndroidProduct.mk` file for the specific device contained in the device configuration folder (e.g. `/device/ti/beagleboard`):

```
# u-blox RIL communication interface
RIL_COM_INTERFACE := usb
```

C.1.2 `Init.rc` configuration

Add the following lines to the `init.rc` file:

```
# Change permissions for modem
    chmod 0660 /dev/ttyACM0
    chown radio radio /dev/ttyACM0
    chmod 0660 /dev/ttyACM1
    chown radio radio /dev/ttyACM1
    chmod 0660 /dev/ttyACM2
    chown radio radio /dev/ttyACM2

# Set permissions for u-blox RIL Repository
    chown radio radio /system/etc/rril
    chmod 0770 /system/etc/rril
    chown radio radio /system/etc/rril/repository.txt
    chmod 0660 /system/etc/rril/repository.txt

# Set u-blox RIL repository state
    setprop net.rril.repository notready
# Prepare u-blox RIL repository
service rril-repo /system/bin/rril-repo.sh
    user root
    group radio
    oneshot
# Load u-blox RIL
service ril-daemon /system/bin/rild -l /system/lib/librapid-ril-core.so -- -a
/dev/ttyACM0 -n /dev/ttyACM1
    class main
    socket rild stream 660 root radio
    socket rild-debug stream 660 radio system
    user root
    group radio cache inet misc audio

service pppd_data0 /system/bin/init.gprs-pppd
    user root radio
    group radio cache inet misc
    disabled
    oneshot

service pppd_data1 /system/bin/init.gprs-pppd
    user root radio
    group radio cache inet misc
    disabled
    oneshot
```

```
service pppd_term /system/bin/stop_pppd 15
    class main
    disabled
    oneshot

service pppd_kill /system/bin/stop_pppd 9
    class main
    disabled
    oneshot
```

C.2 UART

☞ To enable the UART connection, configure the `AndroidProducts.mk` and `init.rc` files as follows.

C.2.1 AndroidProducts.mk

☞ Not required for RIL release 09.00 and above.

Insert the following lines into the AndroidProduct.mk file contained in the device configuration folder (e.g. /device/ti/beagleboard).

```
# u-blox RIL communication interface
RIL_COM_INTERFACE := uart
```

C.2.2 Init.rc configuration

Add the following lines to the `init.rc` file:

```
# Change permissions for the modem
    chmod 0660 /dev/<serial_port>
    chown radio radio /dev/<serial_port>

# Set permissions for the u-blox RIL repository
    chown radio radio /system/etc/rril
    chmod 0770 /system/etc/rril
    chown radio radio /system/etc/rril/repository.txt
    chmod 0660 /system/etc/rril/repository.txt

# Set the u-blox RIL repository state
    setprop net.rril.repository notready

# Multiplexing device
service gsmmuxd /system/bin/logwrapper /system/bin/gsm0710muxd -s /dev/<serial_port>
-n3 -v7 -mbasic
    class main
    user radio
    group radio cache inet misc
    oneshot

# Use this service for stopping gsmmuxd
service mux_stop /system/bin/stop_muxd 15
    class main
    disabled
    oneshot

# Prepare the u-blox RIL repository
service rril-repo /system/bin/rril-repo.sh
    class main
    user root
    group radio
```

```
    oneshot

# Load u-blox RIL
service ril-daemon /system/bin/rild -l /system/lib/librapid-ril-core.so -- -a
/dev/pts/0 -n /dev/pts/1
    class main
    socket rild stream 660 root radio
    socket rild-debug stream 660 radio system
    user root
    group radio cache inet misc audio
service pppd_data0 /system/bin/init.gprs-pppd
    user root radio
    group radio cache inet misc
    disabled
    oneshot
service pppd_data1 /system/bin/init.gprs-pppd
    user root radio
    group radio cache inet misc
    disabled
    oneshot

# use these services for stopping pppd. Terminate pppd with SIGTERM
service pppd_term /system/bin/stop_pppd 15
    disabled
    oneshot
# forcefully kill pppd with SIGKILL
service pppd_kill /system/bin/stop_pppd 9
    disabled
    oneshot
```

C.2.3 `Ueventd.rc` configuration

Add the following permissions to the `ueventd.rc` file:

```
# Permissions for virtual devices created by MUX
/dev/pts*                 0660   radio      radio
# Permissions for the USB serial device
/dev/ttyUSB*              0660   radio      radio
```

C.3 SPI

To enable the SPI connection, configure the `AndroidProducts.mk` and `init.rc` files as follows.

C.3.1 `AndroidProducts.mk`

☞ Not required for RIL release 09.00 and above.

Insert the following lines into the `AndroidProduct.mk` file contained in the device configuration folder (e.g. `/device/ti/beagleboard`).

```
# u-blox RIL communication interface
RIL_COM_INTERFACE := spi
```

C.3.2 `Init.rc` configuration

Add the following lines to the `init.rc` file:

```
# Change permissions for the modem
    chmod 0660 /dev/<spi_port>
    chown radio radio /dev/<spi_port>

# Set permissions for the u-blox RIL repository
```

```
    chown radio radio /system/etc/rril
    chmod 0770 /system/etc/rril

    chown radio radio /system/etc/rril/repository.txt
    chmod 0660 /system/etc/rril/repository.txt

# Set the u-blox RIL repository state
    setprop net.rril.repository notready

# Multiplexing device
service gsmmuxd /system/bin/logwrapper /system/bin/gsm0710muxd -s/dev/<spi_port> -n3
-v7 -mbasic
    user radio
    group radio cache inet misc
    oneshot

# Use this service for stopping gsmmuxd
service mux_stop /system/bin/stop_muxd 15
    class main
    disabled
    oneshot

# Prepare the u-blox RIL repository
service rril-repo /system/bin/rril-repo.sh
    class main
    user root
    group radio
    oneshot

# Load the u-blox RIL
service ril-daemon /system/bin/rild -l /system/lib/librapid-ril-core.so -- -a
/dev/pts/0 -n /dev/pts/1
    class main
    socket rild stream 660 root radio
    socket rild-debug stream 660 radio system
    user root
    group radio cache inet misc audio

service pppd_data0 /system/bin/init.gprs-pppd
    user root radio
    group radio cache inet misc
    disabled
    oneshot

service pppd_data1 /system/bin/init.gprs-pppd
    user root radio
    group radio cache inet misc
    disabled
    oneshot

# use these services for stopping pppd

# terminate pppd with SIGTERM
service pppd_term /system/bin/stop_pppd 15
    class main
    disabled
    oneshot
# forcefully kill pppd with SIGKILL
service pppd_kill /system/bin/stop_pppd 9
    class main
    disabled
    oneshot
```

D Android 2.3 module configuration

Configure the kernel to support the u-blox cellular module and the applicable interface connection.

D.1 USB

To enable the USB module connection, configure the `BoardConfig.mk`, `init.rc` and `ueventd.[platform].rc` files as follows.

D.1.1 `BoardConfig.mk` configuration

Add the following lines to the `BoardConfig.mk` file to build the RIL library for the USB connection:

```
# u-blox RIL communication interface
RIL_COM_INTERFACE := usb
```

D.1.2 `Init.rc` configuration

Add the following lines to the `init.rc` file:

```
# Change permissions for modem
    chmod 0660 /dev/ttyACM0
    chown radio radio /dev/ttyACM0
    chmod 0660 /dev/ttyACM1
    chown radio radio /dev/ttyACM1
    chmod 0660 /dev/ttyACM2
    chown radio radio /dev/ttyACM2

# Set permissions for the u-blox RIL Repository
    chown radio radio /system/etc/rril
    chmod 0770 /system/etc/rril

    chown radio radio /system/etc/rril/repository.txt
    chmod 0660 /system/etc/rril/repository.txt

# Set the u-blox RIL repository state
    setprop net.rril.repository notready

# Prepare the u-blox RIL repository
service rril-repo /system/bin/rril-repo.sh
  user root
  group radio
  oneshot

# Load the u-blox RIL
service ril-daemon /system/bin/rild -l /system/lib/librapid-ril-core.so -- -a
/dev/ttyACM0 -n /dev/ttyACM1
    socket rild stream 660 root radio
    socket rild-debug stream 660 radio system
    user root
    group radio cache inet misc audio

service pppd_gprs /system/bin/init.gprs-pppd
    user root radio
    group radio cache inet misc
    disabled
    oneshot

service pppd_term /system/bin/stop_pppd 15
    disabled
    oneshot
```

```
service pppd_kill /system/bin/stop_pppd 9
    disabled
    oneshot
```

D.1.3 Ueventd.[platform].rc configuration

Add the following permissions to the ueventd.[platform].rc, where "platform" label depends on the BSP:

```
/dev/ttyACM0              0660   radio      radio
/dev/ttyACM1              0660   radio      radio
/dev/ttyACM2              0660   radio      radio
```

D.2 UART

☞ To enable the UART module connection, configure the BoardConfig.mk, init.rc and ueventd.rc files as follows.

D.2.1 BoardConfig.mk configuration

☞ Not required for RIL release 09.00 and above.

To build the RIL library for a UART connection, add the following lines to the BoardConfig.mk file:

```
# u-blox RIL communication interface
RIL_COM_INTERFACE := uart
```

D.2.2 Init.rc configuration

Add the following lines to the init.rc file:

```
# Change permissions for the modem
    chmod 0660 /dev/<serial_port>
    chown radio radio /dev/<serial_port>

# Set permissions for the RIL repository
    chown radio radio /system/etc/rril
    chmod 0770 /system/etc/rril

    chown radio radio /system/etc/rril/repository.txt
    chmod 0660 /system/etc/rril/repository.txt

# Set the RIL repository state
    setprop net.rril.repository notready

# Multiplexing device
service gsmmuxd /system/bin/logwrapper /system/bin/gsm0710muxd -s /dev/<serial_port>
-n3 -v7 -mbasic
    user radio
    group radio cache inet misc
    oneshot

# Use this service for stopping gsmmuxd
service mux_stop /system/bin/stop_muxd 15
    disabled
    oneshot

# Prepare the u-blox RIL repository
service rril-repo /system/bin/rril-repo.sh
    user root
    group radio
    oneshot
```

```
# Load the u-blox RIL
service ril-daemon /system/bin/rild -l /system/lib/librapid-ril-core.so -- -a
/dev/pts/0 -n /dev/pts/1
    socket rild stream 660 root radio
    socket rild-debug stream 660 radio system
    user root
    group radio cache inet misc audio

service pppd_gprs /system/bin/init.gprs-pppd
    user root radio
    group radio cache inet misc
    disabled
    oneshot

# Use these services for stopping pppd
#
# terminate pppd with SIGTERM
service pppd_term /system/bin/stop_pppd 15
    disabled
    oneshot

# forcefully kill pppd with SIGKILL
service pppd_kill /system/bin/stop_pppd 9
    disabled
    oneshot
```

D.2.3 `Ueventd.rc` configuration

Add the following permissions to the `ueventd.rc` file:

```
# Permissions for virtual devices created by MUX
/dev/pts*                 0660   radio      radio
# Permissions for the USB serial device
/dev/ttyUSB*              0660   radio      radio
```

☞ `ttyUSB` shall be replaced with `ttyS` when using the UART serial port of the platform.

D.3 SPI

☞ To enable the SPI module connection, configure the `BoardConfig.mk` and `init.rc` files as follows.

D.3.1 `BoardConfig.mk` configuration

To build the RIL library for the SPI connection, add the following lines to the `Boardconfig.mk` file:

```
# u-blox RIL communication interface
RIL_COM_INTERFACE := spi
```

D.3.2 `Init.rc` configuration

Add the following lines to the `init.rc` file:

```
# Change permissions for modem
    chmod 0660 /dev/<spi_port>
    chown radio radio /dev/<spi_port>

# Set permissions for u-blox RIL repository
    chown radio radio /system/etc/rril
    chmod 0770 /system/etc/rril

    chown radio radio /system/etc/rril/repository.txt
    chmod 0660 /system/etc/rril/repository.txt
```

```
# Set u-blox RIL repository state
    setprop net.rril.repository notready

# Multiplexing device
service gsmmuxd /system/bin/logwrapper /system/bin/gsm0710muxd -s/dev/<spi_port> -n3
-v7 -mbasic
    user radio
    group radio cache inet misc
    oneshot

# Use this service for stopping gsmmuxd
service mux_stop /system/bin/stop_muxd 15
    class main
    disabled
    oneshot

# Prepare u-blox RIL repository
service rril-repo /system/bin/rril-repo.sh
 user root
 group radio
 oneshot

# Load u-blox RIL
service ril-daemon /system/bin/rild -l /system/lib/librapid-ril-core.so -- -a
/dev/pts/0 -n /dev/pts/1
    socket rild stream 660 root radio
    socket rild-debug stream 660 radio system
    user root
    group radio cache inet misc audio
service pppd_gprs /system/bin/init.gprs-pppd
    user root radio
    group radio cache inet misc
    disabled
    oneshot

# use these services for stopping pppd
#
# terminate pppd with SIGTERM
service pppd_term /system/bin/stop_pppd 15
    disabled
    oneshot

# forcefully kill pppd with SIGKILL
service pppd_kill /system/bin/stop_pppd 9
    disabled
    oneshot
```

E AT pass through commands

The Android operating system does not allow sending commands to the module without using the RIL software layer, so two requests provided by RIL's API must be used:

- `RIL_REQUEST_OEM_HOOK_RAW`: it passes raw byte arrays back and forth
- `RIL_REQUEST_OEM_HOOK_STRINGS`: it passes strings back and forth

The following is sample code that could be used to send commands directly to the module:

⚠ Make a backup of the `<android_root>/packages/apps/Settings` folder before making any changes to it.

- Add the following lines in the bottom of `packages/apps/Settings/AndroidManifest.xml` before the `</application>` field.

```
<activity android:name="RilOemHookTest" android:label="@string/testing_RIL_OEMHook"
    android:process="com.android.phone">
    <intent-filter>
      <action android:name="android.intent.action.MAIN" />
      <category android:name="android.intent.category.DEVELOPMENT_PREFERENCE" />
    </intent-filter>
</activity>
```

- Add the following lines in the bottom of `packages/apps/Settings/res/values/strings.xml` file before the `</resources>` field.

```
<!-- For RIL OEM HOOK testing -->
<string name="testing_RIL_OEMHook">RIL OEM Hook Test</string>
<string name="ril_oem_choose">Choose a RIL OEM Hook API to test:</string>
<string name="foat_fls">Start FOAT (fls file=C800)</string>
<string name="foat_dffs">Start FOAT (dffs file=C801)</string>
<string name=" radio_usb ">Set Prop (modInf=USB)</string>
<string name=" radio_uart "> Set Prop (modInf=UART)</string>
<string name="radio_api1">API 1 (datalen=0)</string>
<string name="radio_api2">API 2 (datalen=1)</string>
<string name="radio_api3">API 3 (datalen=6)</string>
<string name="radio_api4">API 4 Type command :</string>
<string name="radio_run">RUN</string>
<string name="ril_oem_response">Module response:</string>
```

At the bottom of the `packages/apps/Settings/res/xml/testing_settings.xml` file before the `</PreferenceScreen>` field, add the following lines:

```
<PreferenceScreen
        android:title="@string/testing_RIL_OEMHook" >
    <intent
          android:action="android.intent.action.MAIN"
          android:targetPackage="com.android.settings"
          android:targetClass="com.android.settings.RilOemHookTest" />
</PreferenceScreen>
```

- Run the following command to merge the OEM Hook test application code into the Android source.
  `cp -pvRf ril_sc_<version>/packages/apps/Settings/* <android_root>/packages/apps/Settings/`
- Add the following Java code to the relevant file depending on the Android version being used:

Android version	File name and path	Java code
For Android 2.x - 4.1	<android_root>/frameworks/base/telephony/java/com/android/internal/telephony/PhoneProxy.java	public CommandsInterface getCommandsInterface() { return mCommandsInterface; }
For Android 4.2 - 6.x	<android_root>/frameworks/opt/telephony/src/java/com/android/internal/telephony/PhoneProxy.java	
For Android 7.x - 8.x	<android_root>/frameworks/opt/telephony/src/java/com/android/internal/telephony/Phone.java	public CommandsInterface getCommandsInterface() { return mCi; }

- It is possible to start this application from an Android shell using the following command or it can also be accessible using the Android secret code *#*#4636#*#*:

```
am start -a android.intent.action.MAIN -n
com.android.settings/com.android.settings.TestingSettings
```

A screen such as Figure 1 is displayed.

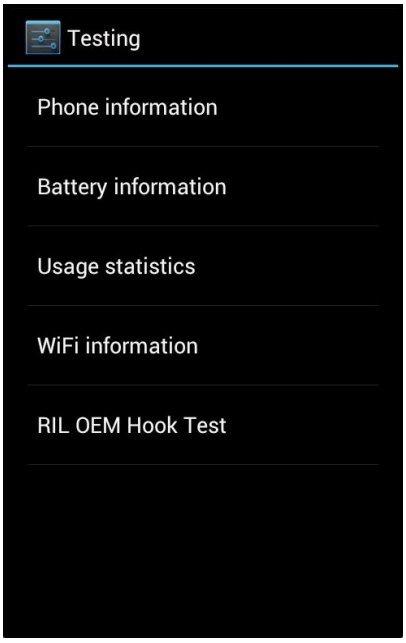


Figure 1: Testing menu settings

Now click on the “RIL OEM Hook Test” menu. A screen such as Figure 2 is displayed.

Figure 2: RIL OEM Hook application

The “Start FOAT” options are to be used for the module firmware update if the provided RIL and module are supported. See appendix H for more information on this. Apart from that, the first three APIs send bytes to the RIL layer to execute a set of commands.

The fourth API sends the command string to the module. Type the command into the white field and click the “RUN” button to execute the command. The module response is displayed in the bottom field.

⚠ RIL uses the numeric error value (AT+CMEE=1) as the error result codes format. A change in the error report setting causes unexpected errors in the RIL behavior.

F Default EPS bearer in LTE (initial PDP context)

F.1 Default EPS bearer in LTE

An EPS bearer is established when the UE connects to a PDN, and remains established throughout the lifetime of the PDN connection to provide the UE with always-on IP connectivity to that PDN. This bearer refers to the default EPS bearer. The UE can have additional default bearers as well. Each default bearer comes with an IP address and it has nominal QoS applied by the LTE network.

A default EPS bearer is allocated for an APN. To create multiple "Default EPS Bearers" in an LTE network, more than one APN should be working in the LTE network.

F.2 `apns-conf.xml` configurations (manually/during build process)

The `apns-conf.xml` file should have at least one corresponding MCC/MNC entry/record with **type** field "ia", where "ia" stands for "Initial Attach". Multiple values can be defined in the **type** field e.g. default,supl,mms,ia

Example: during the source build process, the user can enter the corresponding network settings in the `apns-conf.xml` file. The sample values are given below.

```
<apn carrier="Cosmote Wireless Internet"
  mcc="202"
  mnc="01"
  apn=""
  type="ia"
/>
<apn carrier="Cosmote Wireless Internet"
  mcc="202"
  mnc="01"
  apn="internet"
  type="default,supl,mms"
/>
```

F.3 `apns-conf.xml` configurations (Android UI)

The user can also set the APN according to the service provider manually through the Android UI.

- Go to Android's main settings.
- Under Wireless & Networks, go to "mobile networks" (Figure 3). Click "more settings" to get this option. (Figure 4).
- Tap on Access Point Names (that is what APN stands for).
- Press menu, and then click "New APN."
- Enter in all of the required information in each field by clicking into it and then clicking "OK" to save (see Figure 5).
- After having finished entering all the settings, click menu again and then Save.

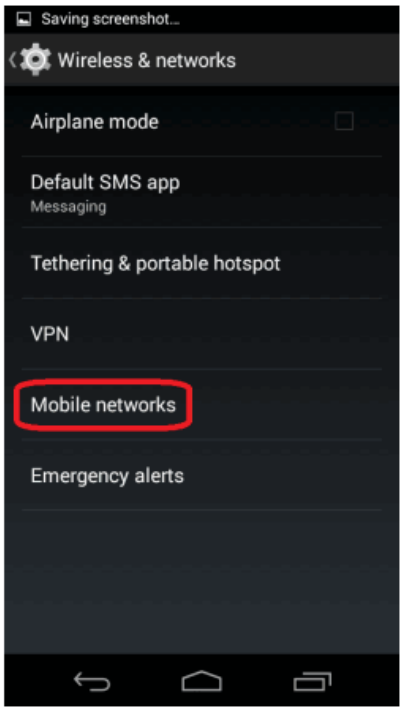


Figure 3

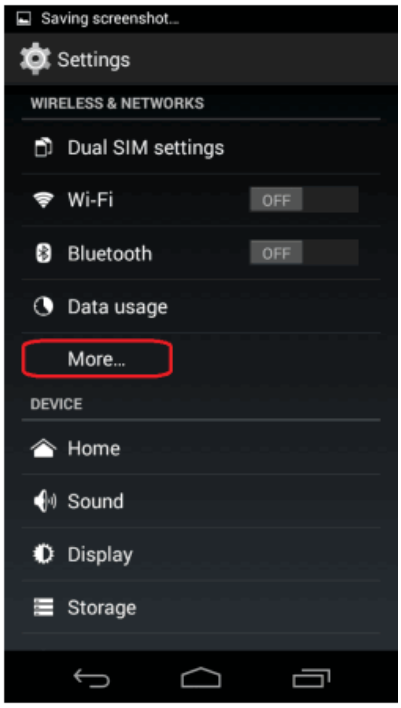


Figure 4

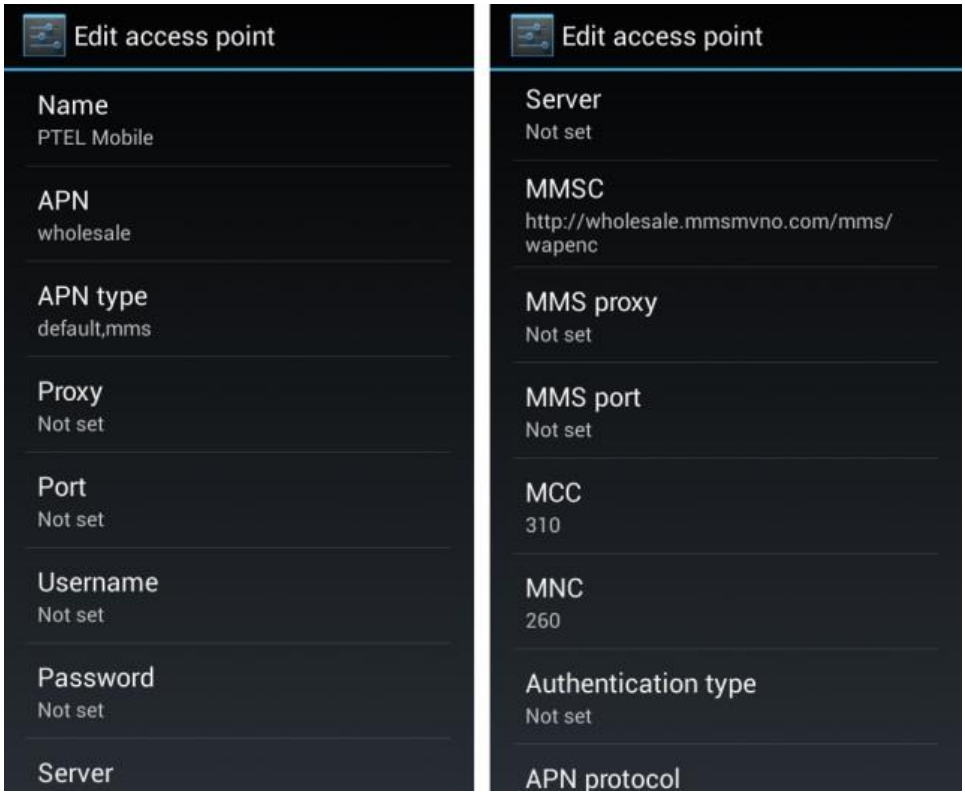


Figure 5

F.4 Single default EPS bearer allowed – RIL handling

Some LTE networks allow only one default EPS bearer establishment: a single working APN is available for the user.

In the scenario where only a single working APN is available, the RIL tries to establish data on CID=1 (CID=1 is requested by OS) when `RIL_REQUEST_SETUP_DATA_CALL` is triggered by the OS. If a single default bearer case is encountered, RIL failed in its maximum tries while establishing data on CID=1.

Once the RIL has failed to establish the data on the requested CID by the OS, the RIL tries to establish data on CID=4, which is already established during the time of the LTE attach process, on the next try of `RIL_REQUEST_SETUP_DATA_CALL` by the OS.

G Repository file configuration

`Repository.txt` is a part of the u-blox RIL source, and this file is used to set various user defined values for various fields. These fields are split into sections. Operational timeouts, TTY constants, RIL settings, last values, USB/UBM and log level etc. are some examples of the fields/values defined in this file.

G.1 TOBY-L2 USB profile configuration

The USB and UBM configurations can be set in the **Group USBUBM** section of the `repository.txt` file by setting the value of the **USBCONF** and **UBMCONF** fields respectively. The various values that are currently supported are shown in Table 2 and Table 3. If these settings are not configured, then the module's default settings will be used.

[USBCONF]	Configurations
0	CDC-ACM
2	CDC-ECM
3	RNDIS

Table 2: Supported USB configurations

[UBMCONF]	Configurations
1	Router
2	Bridge

Table 3: Supported u-blox modem configurations

G.1.1 USB profiles configuration +UUSBCONF

This command configures the active USB profile. Each USB profile includes a certain number of USB endpoints, depending on the overall USB functions of the USB profile. The USB profile switch is not performed at run-time. The settings are saved in the NVM at module power-off; the new configuration will be effective at the subsequent module reboot. The command syntax is:

AT+UUSBCONF=[<id>[,<network>[,<audio>]]]

Configuration	Command setting	Remarks
6 CDC-ACM ports	AT+UUSBCONF=0 and AT+CFUN=16	The data connection uses the PPP process
1 CDC-ACM port and 1 RNDIS port	AT+UUSBCONF=3 and AT+CFUN=16	This is the factory-programmed configuration. The data connection uses RNDIS port (network usb<X>)
1 CDC-ACM port and 1 ECM port	AT+UUSBCONF=2 and AT+CFUN=16	The data connection uses ECM port (network eth<X>).

Table 4: TOBY-L2 series allowed profile configurations

G.1.2 Boot mode configuration +UBMCONF

This command configures the system networking mode. The chosen configuration is stored in the NVM and it will be applied at the subsequent module reboot. The command syntax is:

AT+UBMCONF=[<networking_mode>]

Configuration	Command setting	Remarks
Router mode	AT+UBMCONF=1 and AT+CFUN=16	Default and factory-programmed value: router mode
Bridge mode	AT+UBMCONF=2 and AT+CFUN=16	bridge mode

Table 5: TOBY-L2 series possible boot mode configurations

G.2 CDMA network operator name

The network operator names for CDMA are set in the "*Operator Settings*" group of the `repository.txt` file by setting the "OperatorNames" field to its appropriate name and the MCC and MNC values.

Example: the following lines are present in the `Repository.txt` file to define the network operator name.

```
// Operator Settings (only for CDMA)
Group OperatorNames
SuportedOperator        Sprint
MCCMNC                  310120
```

G.3 Timeouts

The timeout values in the **Request timeouts** and **other timeouts** groups present in the repository file have already been tuned by u-blox for optimum functionality, but they can be modified according to need.

G.4 Emergency numbers

From the developer's point of view, emergency numbers can be set in Android by assigning either the `ro.ril.ecclist` or `ril.ecclist` property, where the first one can be set at build-time only and is read-only, while the latter is dynamic and can be set/modified at runtime. When dialing emergency numbers, by default Android first checks the `ril.ecclist` property for the specific number, but if this property is not defined, then it checks the number in the `ro.ril.eccclist` property. The numbers currently present in the list are shown in Table 6. For more information on these numbers, kindly consult this Wikipedia link.

Numbers	Listing
111,113,115,117,122,125,127	Common emergency numbers used in various countries
112,911,000,08,110,999,118,119	ETSI Standard

Table 6: Supported emergency numbers

H Module firmware update

☞ See the Android RIL Update Manager Source Code Application Note [7] for the generic update manager (applicable for all Android versions).

H.1 Firmware over AT (FOAT) – For Android 2.3 only

This method involves updating the u-blox module firmware using AT commands that are sent by the RIL.

⚠ Currently this firmware update feature is only supported for SARA-U2, SARA-G340 / SARA-G350 "02S" and "02X" product versions on Android 2.3 using *.fls and *.dffs files.

H.1.1 Prerequisites

The default location for the firmware file is set to "/data/updates/ublox/" with the default update filename set as "update.[File extension]" e.g. `update.fls`, or `update.dffs`. The file permissions for the folder and the firmware file should be *radio:radio* (user:group).

H.1.2 Update Initiation

The update process is initialized by sending the `RIL_REQUEST_OEM_HOOK_RAW` API request to the RIL.

H.1.3 URC

During the update process, the RIL continuously sends the intermediate update progress as a percentage figure in regular intervals of 0.5 MiB (by default) using the `RIL_UNSOL_OEM_HOOK_RAW` API.

H.1.4 Status codes

If an error occurs during the update process, the RIL sends the particular status/error code to the user application in the form of a URC and then retries to update the module.

☞ The APIs and status codes mentioned above are explained in the Android RIL FOAT Protocol Specs (UBX-16007616).

☞ See appendix E of this document for testing the update feature using the OEMHOOK app.

I Multi module support in Android RIL

⚠ This section only applies to RIL version 09.01 and above.

I.1 Multi module

This method allows the u-blox RIL to switch between two modules connected on different interfaces. Only the USB and UART interfaces can be used to set up this feature.

When the module is configured in CDC-ACM or RNDIS mode, then it will be symbolized it as a USB interface. If the module is connected using a serial port or virtual COM port on a serial interface, then it is symbolized it as a UART interface.

The main purpose of this feature is to use two modules on both interfaces but activate one at a time; RIL has a functionality to switch between the two modules on a particular trigger.

I.1.1 Prerequisites

Both modules should be connected on the respective interfaces according to need. By default, it is in "ttyACM[usb]" mode. The default mode can be configured through the szInterfaceDefault variable in the function named `mainloop` in the `rildmain.cpp` file.

I.1.2 Mode switching

u-blox also provides a sample test application source for the user to trigger between both modules by setting a specific property. See appendix E to integrate the testing application in the Android source code. This application uses OEM_HOOK_API to communicate with RIL and perform specific operations.

The switching process is initialized by selecting the required interface property in the RIL OEM HOOK Test APK, e.g. Set Property (modinf=ttyACM) for the USB interface and Set Property (modinf=pts) for the UART interface.

By setting the following property switching, operations can be performed.

persist.radio.uril.modInf

Possible values of this property are

- For CDC-ACM interface: "ttyACM"
- For UART interface with GSMMUX: "pts"

☞ Comment out the defined value of this property (`persist.radio.uril.modInf`) from the `init.rc` file to persist its desired set value.

Follow these steps to use this feature manually:

1. Set the property (`persist.radio.uril.modInf`) for the desired interface.
2. Power off the module by means of the AT command.
3. Cut off the power IO to stop the module power.
4. Now enable the power of the second module.
5. The RIL will automatically connect to the set interface if it is available at.

I.1.3 Switching status

After the property has been successfully set, the power of the currently active module should be cut off, which can be done by issuing the AT+CPWROFF command from the RIL OEM HOOK test APK. Then turn the power back on to the new active module.

J Verizon network

⚠ This section only applies to RIL version 09.03 and onwards.

J.1 init.rc

The following service needs to be added to the service section of `init.ublox.rc` file as follows:

```
# APN update service for Verizon Network
service static_apn /system/bin/static_apn
user root
group radio cache inet misc
class main
disabled
oneshot
```

J.2 SE Policy

The SE Policy needs to be update for the Verizon network accordingly:

- ANDROID_SOURCE/external/sepolicy (For **Android 5 and 6**)
- ANDROID_SOURCE/system/sepolicy (For **Android 7 and onwards**)

J.2.1 File_contexts

Add the required files to their respective contexts as shown below so that any service/process can access them according to the permission granted to it (if required).

Binaries	Contexts
/system/bin/static_apn	u:object_r:rild_exec:s0

J.3 Core.mk

Update the following file to copy the target directory on the device:

- ANDROID_SOURCE/build/target/product/core_ublox.mk

Add the following line to `PRODUCT_COPY_FILES`

```
$(LOCAL_PATH)/rootdir/etc/static_apn:system/bin/static_apn
```

K Module specific configurations

This section applies to modules requiring some customized modifications in the Android source or kernel to properly communicate with the u-blox RIL driver.

K.1 SARA-R4

K.1.1 Enable USB serial drivers in the kernel

The SARA-R4 module enumerates as a USB serial device (/dev/ttyUSB) instead of a ttyACM when connected to an Android device. USB serial driver support needs to be enabled in the kernel for a proper enumeration of the SARA-R4 module.

After a successful enumeration of SARA-R4, three ttyUSB* ports namely ttyUSB0, ttyUSB1 and ttyUSB2 will be visible in /dev.

The ttyUSB1 port is used for communication purposes with the u-blox RIL.

L Audio codec

⚠ LISA-C2 product series does not support external codec management by AT commands.

L.1 Configuration

This section describes a set of u-blox proprietary AT commands to be used for the audio features configuration. Customers can configure this section for custom audio features according to their platform needs.

The section below in the RIL code is divided into two main sections:

- Settings to manage external codec or other external audio IC
- Setting the audio path and configure I^2S interfaces

For further details, see the u-blox AT Commands Manual [6].

L.2 Example

Below, a RIL code snapshot is provided to mark the location where the customer can update/add/manage their own custom audio codec.

(ublox_ril\core\nd\Systemmanager.cpp under ::InitializeModemFeatures())

```
//Initialized digital audio features (I2C interface)
pCmd = new CCommand(RIL_CHANNEL_ATCMD, NULL,ND_REQ_ID_NONE,
"AT+UMCLK=2,1;+UI2CO=1,0,0,0x10,0;+UI2CW=\"00000000108F2024000010330025000000 8A\",18;+UI2C
W=\"049E\",2;+UI2CC\r", &CTE::ParseSupportedFeatures);

//Initialized digital audio features (I2S interface)
pCmd = new CCommand(RIL_CHANNEL_ATCMD, NULL,ND_REQ_ID_NONE,
"AT+USPM=1,1,0,0,2;+UI2S=1,1,0,3,0\r", &CTE::ParseSupportedFeatures);
```

M i.MX 6 platform specific notes

Consider the following points during the integration on the i.MX 6 platform.

See section B for details about changes required in the following file.

M.1 `init.rc` configuration

The modifications described below are required to include `init.ublox.rc` in the Android source.

File Path: android_source/device/fsl/sabresd_6dq/init.rc

- Copy `init.ublox.rc` file in the path mentioned in this section.
- Add below mentioned line at top of `init.rc and` remove or comment any already present **ril-daemon** service and **pppd_gprs** from init.rc file at the path.

```
#***u-blox Modifications***#
import /init.ublox.rc
#***u-blox Modifications***#
```

M.2 `ueventd.freescale.rc`

The file path for `ueventd.freescale.rc` is:

```
android_source/device/fsl/imx6/etc/ueventd.freescale.rc
```

M.3 imx6.mk

Open the `imx6.mk` file for the path `device/fsl/imx6/imx6.mk` and apply the following changes.

```
#Remove following lines from imx6.mk file
   - chat
   - ip-up-vpn
   - ip-up-ppp0
   - ip-down-ppp0
   - device/fsl/imx6/etc/ppp/init.gprs-pppd:system/etc/ppp/init.gprs-pppd \
```

M.4 apn-config.xml

Place the `apns-conf.xml` file at /device/fsl/imx6/ for any possible data connectivity issues.

N Android RIL integration FAQ

Q: RIL cannot open the COM port to connect ttyACMX or ttyUSBX.

A: Make the following verification checks if the COM port is not opened:

- In the Android platform (while the u-blox module is connected), use adbshell and verify that the `ttyACMX` or `ttyUSBX` ports are present in the `/dev` path. If these are not present, see appendix A.
- Make sure that the necessary permissions (`ttyACMx`, `ttyUSBx`) are added in the `uevent.platform.rc` file.
- Verify the SEPolicies for the required port, see appendix B.6.

Q: All the scripts are not working or RIL is not working after integration in Android.

A: It is recommended that the RIL package is downloaded and opened on the Linux machine. If the zip file is opened on a Windows machine, it may result in script corruption because of illegal characters (^M).

Q: How to enable different log levels in RIL?

A: In the `repository.txt` file, the "LogLevel" field can be used to set the log level. The default value is set to "2=info logs". The following values are available:

- 4: critical
- 3: warning
- 2: info
- 1: verbose

Q: How to shift from PPP mode to RNDIS mode or vice versa?

A: u-blox RIL supports both PPP and RNDIS modes, and this can be selected from the `repository.txt` file.

- For PPP mode, the setting is "USBCONF=0"".
- For RNDIS (router) mode, the settings are "USBCONF=3" and "UBMCONF=1".

Q: Data in PPP mode is not working after integrating the u-blox RIL.

A: In PPP mode, check that the pppd_data service is running properly. If not, check the permissions and SEPolicy for this script. The pppd_data service starts the PPP daemon and sets many properties, so it should be verified that the necessary SEPolicies for pppd should be set.

Also verify that the "chat" script is added and running properly in pppd.

Q: Data in RNDIS is not working after integrating u-blox RIL.

A: USB0 is used to connect data in RNDIS mode so it should be verified that ttyUSB0 is used for the u-blox module. The "netd_ena" service should also be running properly.

Q: How to increase log buffer size?

A: There is a setting in the "Log buffer sizes" developer option that can be set from 256 k to 16 M. If this setting is not present in the developer options, then an alternative way to change the buffer size is to set following property at the start of the "boot" section in the `init.rc` file.

setprop persist.logd.size 16777216

Q: How do you go back to 4G as there is no option in the preferred network type?

A: This can be done by dialing "`*#*#4636#*#*`" on an Android device and a menu would pop up. Select the Phone information menu option and then it is possible to set the preferred network type from there.

Q: How to get the MDN info displayed in Android?

A: The MDN is saved in a special memory area by the service provider, with the name "ON": Own number phonebook (read/write) and the content can be shown by means of +CNUM. Sometimes it is written/saved/available by the SIM provider and sometimes not. So the +CNUM AT command can read this number (if it is available).

Q: Why `gsm0710muxd` is used?

A: `gsm0710muxd` is used for serial communications with the GSM modem. It uses the AT+CMUX command to enable the multiplexing protocol control to make the various channels available to RIL. These channels are virtually mapped with the /pts channel of `gsm0710muxd` and connect RIL with the /pts channel.

Q: How to check the properties set in RIL?

A: The properties set in RIL can be checked in the adb shell using the "`getprop`" command, and any property can be set manually using the "`setprop`" command.

Q: pppd exits with error code 17; how to resolve this error?

A: This error happens when the ppp server starts up too slowly. When the ppp daemon sends the link configuring packets, they get bounced from the remote modem, thereby giving a serial loopback error. To resolve this, in the gprs script, `build/target/product/rootdir/etc/ppp/peers`, add the following line.

"`lcp-max-configure 50`"

Q: How to set the default Preferred Network Settings in Android?

A: To set the default preferred network in Android, set the **`ro.telephony.default.network`** property in `device.mk` to the desired network settings. For example,

```
ro.telephony.default.network=10 /* LTE, CDMA, EvDo, GSM/WCDMA */
```

Recompile the source after setting the property and verifying the change from the build.prop file in the out directory.

Q: How to enable adb logging using WiFi in RIL?

A: Add the following line to the boot section of the init.rc file, that is:

```
setprop service.adb.tcp.port 5555
```

Q: How to enable the USB serial driver (e.g. FTDI) in the Android kernel?

A: It can be configured by enabling the USB serial driver in the USB section using the “make menuconfig” command. Alternatively, it can also be enabled by adding the following lines into the defconfig file of the respective hardware.

```
CONFIG_USB_SERIAL=y
CONFIG_USB_SERIAL_CONSOLE=y
CONFIG_USB_SERIAL_FTDI_SIO=y
CONFIG_USB_SERIAL_QUALCOMM=y
```

Related documents

[1] u-blox SPI Interface Application Note, Docu No UBX-13001919
[2] Ubuntu download: http://www.ubuntu.com/desktop/get-ubuntu/download
[3] USB driver for Android: https://dl-ssl.google.com/android/repository/usb_driver_r03-windows.zip
[4] Android SDK: http://developer.android.com/sdk/index.html
[5] u-blox Multiplexer Implementation Application Note, Docu No UBX-13001887
[6] u-blox AT Commands Manual, Docu No UBX-13002752
[7] Android RIL Update Manager Source Code Application Note, Docu No UBX-16029275

☞ For regular updates to u-blox documentation and to receive product change notifications, register on our homepage (www.u-blox.com).

Revision history

Revision	Date	Name	Comments
-	11-Jul-2011	fpic	Initial release
1	05-Sep-2011	fpic	Document aligned to RIL Delivery 01.023
2	23-Sep-2011	lpah	Added chapter Appendix 1: Module connection
3	26-Jan-2012	lpah	Extended to include LISA-U2 series
4	26-Mar-2012	fpic	Extended to include Android 4.0 delivery
5	21-Jun-2012	fpic	Extended to include LEON-G100 and LEON-G200 series Added compatibility matrix for Android software deliveries and supported interface by u-blox wireless modules
6	20-Nov-2012	fpic / lpah	Android 4.1 supported (Last revision with docu number 3G.G2-CS-11003)1
A	30-Aug-2012	fpic	Extended to include SARA-G350 series Insert minor changes and support for 2G modules into Android 4.113
R08	08-Apr-2014	fpic	Android 4.2 and 4.3 supported Added note on power off handling
R09	31-Jul-2014	fpic	Android 4.4 supported Extended the document applicability to SARA-U2 and TOBY-L2 series
R10	24-Oct-2014	fpic	Included modification for latest TOBY-L2 delivery
R11	11-Sep-2015	yasi	PPP support for TOBY-L2 Android 4.4 supported with MUX interface for SARA-G310 Android 5.0 supported
R12	05-Feb-2016	msin	Extended to RIL version 08.01
R13	03-May-2016	bqam / bahm	Extended to RIL version 09.00 Android 6.0 supported Firmware update over AT (FOAT) section added for Android 2.3 (SARA-U270-00S)
R14	31-Mar-2017	bkha / fdil	Extended to RIL version 09.02 Extended the document applicability to TOBY-R2 and LARA-R2 series Updated TOBY-L2 series Profile setting section
R15	28-Jul-2017	bahm / yasi	Android 7.0 supported Added section about Verizon network settings
R16	30-Aug-2017	uafz	Added section about module specific modifications
R17	10-Oct-2017	bkha	Updated the init.rc configuration
R18	27-Nov-2017	bkha	Updated multi module and OEM Hook App sections
R19	17-Jan-2018	bkha	Android Oreo Support and init.ublox.rc, Core_ublox.rc file addition.
R20	16-May-2018	bkha	Extended document applicability to TOBY-L4 series.
R21	24-Oct-2018	bkha / fdil	Added section regarding the integration on i.MX 6 platform.

Contact

For complete contact information, visit us at www.u-blox.com.

u-blox Offices

North, Central and South America

u-blox America, Inc.

Phone: +1 703 483 3180
E-mail: info_us@u-blox.com

Regional Office West Coast:

Phone: +1 408 573 3640
E-mail: info_us@u-blox.com

Technical Support:

Phone: +1 703 483 3185
E-mail: support@u-blox.com

Headquarters
Europe, Middle East, Africa

u-blox AG

Phone: +41 44 722 74 44
E-mail: info@u-blox.com
Support: support@u-blox.com

Asia, Australia, Pacific

u-blox Singapore Pte. Ltd.

Phone: +65 6734 3811
E-mail: info_ap@u-blox.com
Support: support_ap@u-blox.com

Regional Office Australia:

Phone: +61 2 8448 2016
E-mail: info_anz@u-blox.com
Support: support_ap@u-blox.com

Regional Office China (Beijing):

Phone: +86 10 68 133 545
E-mail: info_cn@u-blox.com
Support: support_cn@u-blox.com

Regional Office China (Chongqing):

Phone: +86 23 6815 1588
E-mail: info_cn@u-blox.com
Support: support_cn@u-blox.com

Regional Office China (Shanghai):

Phone: +86 21 6090 4832
E-mail: info_cn@u-blox.com
Support: support_cn@u-blox.com

Regional Office China (Shenzhen):

Phone: +86 755 8627 1083
E-mail: info_cn@u-blox.com
Support: support_cn@u-blox.com

Regional Office India:

Phone: +91 80 405 092 00
E-mail: info_in@u-blox.com
Support: support_in@u-blox.com

Regional Office Japan (Osaka):

Phone: +81 6 6941 3660
E-mail: info_jp@u-blox.com
Support: support_jp@u-blox.com

Regional Office Japan (Tokyo):

Phone: +81 3 5775 3850
E-mail: info_jp@u-blox.com
Support: support_jp@u-blox.com

Regional Office Korea:

Phone: +82 2 542 0861
E-mail: info_kr@u-blox.com
Support: support_kr@u-blox.com

Regional Office Taiwan:

Phone: +886 2 2657 1090
E-mail: info_tw@u-blox.com
Support: support_tw@u-blox.com