



Fibocom 广和通

完美无线体验

MC9x9

RIL 适配指南_Android

V1.1

版权声明

版权所有©2022 深圳市广和通无线股份有限公司。保留一切权利。

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

注意

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

商标申明

 为深圳市广和通无线股份有限公司的注册商标，由所有人拥有。

联系方式

公司网址：<https://www.fibocom.com/>

总部地址：深圳市南山区西丽街道西丽社区打石一路深圳国际创新谷六栋 A 座 10-14 层

总机：+86 755-26733555

目录

修订记录.....	3
1 引言	4
2 Android RIL 驱动概述	5
2.1 RIL 驱动支持的功能.....	5
2.2 支持的 Android 版本	5
3 RIL 集成配置.....	6
3.1 RIL 库源码结构.....	6
3.2 配置 Linux 内核串口驱动.....	7
3.3 添加模块 PID/VID.....	7
3.4 修改 RILD 权限	9
3.5 配置 Android 启动脚本	10
3.6 加载 RIL 库文件.....	11
3.6.1 手动加载 RIL 库文件	11
3.6.2 自动打包 RIL 库文件	12
3.7 MC919 模块设备加载检测.....	13
4 ECM 拨号方式.....	15
4.1 获取 local ip,primary dns,second dns 地址	15
5 ppp 拨号方式.....	16
5.1 修改 ip-down ip-up 文件	16
6 APN 设置和拨号状态查询	18
6.1 手动添加 APN.....	18
6.2 配置 apns-conf.xml 文件	18
7 RIL 调试方法.....	20
8 LOG 中 TAG 标签说明	21
9 当前维护的属性.....	22
9.1 属性设置方法	22
9.1.1 临时设置	22
9.1.2 永久设置	22
9.2 ril.fibocom.dialmode.....	24
9.2.1 简介	24
9.2.2 取值范围	25

9.2.3 示例	25
9.3 ril.fibocom.NetifName	25
9.3.1 作用	25
9.3.2 取值范围	26
9.3.3 示例	26
9.4 ril.fibocom.cid	26
9.4.1 作用	26
9.4.2 取值范围	26
9.4.3 示例	27
9.5 ril.fibocom.cgdcont0	27
9.5.1 作用	27
9.5.2 取值范围	27
9.5.3 示例	27
10 常见问题	28
10.1 无法正常的进行数据业务	28
10.2 电话打不通	28
10.3 无法发送短信	28
10.4 RIL 没有正常工作	28
10.5 设备不支持 PPP 拨号	29
11 附录 A 参考文件	33

修订记录

V1.1 (2022-04-25) 初始版本

1 引言

本文主要介绍如何将 RIL（无线接口层）驱动程序集成到客户的目标设备以及如何修改启动 RIL 服务的配置文件。

2 Android RIL 驱动概述

2.1 RIL 驱动支持的功能

表 1. RIL 支持的功能

功能	是否支持
SMS	YES
VOICE CALL	YES
DATA SERVICE	YES
SIM TOOL KIT	NO

2.2 支持的 Android 版本

目前，Fibocom RIL driver 支持以下的 Android 版本，见下表：

表 2. RIL 支持 android 系统版本

功能	是否支持
Android 2.x	NO
Android 3.x	NO
Android 4.x	YES
Android 5.x	YES
Android 6.x	YES
Android 7.x	YES
Android 8.x	YES
Android 9.x	YES
Android 10.x	YES
Android 11.x	YES

3 RIL 集成配置

本章节介绍 RIL 库集成到客户 Android 系统的操作步骤，包括 RIL 库文件结构，配置 Linux 内核驱动，添加内核支持的 USB 设备 PID/VID，加载 RIL 库文件，以及修改 Android 系统 RILD 执行权限和启动脚本。

3.1 RIL 库源码结构

Fibocom 会提供编译生成的 RIL 库文件 libreference-ril.so 和 RIL 库源码供客户集成使用。RIL 库源码包 21 个主要文件，如图 1 所示。



```
Android.mk
atchannel.c
atchannel.h
at_tok.c
at_tok.h
getdevinfo.c
getdevinfo.h
misc.c
misc.h
network.c
network.h
other_function.c
other_function.h
reference-ril.c
ril_common.h
sim.c
sim.h
sms.c
sms.h
voice.c
voice.h
```

图 1. RIL 库源码文件结构

3.2 配置 Linux 内核串口驱动

串口驱动需要配置 Linux 内核，配置方法如下：

1. cd kernel 进入到内核根目录

```
root@ubuntu:~/testSrc/android5.x_new# cd kernel/
root@ubuntu:~/testSrc/android5.x_new/kernel#
```

2. make menuconfig

```
root@ubuntu:~/testSrc/android5.x_new/kernel# make menuconfig
```

3. device drivers -> usb support -> usb serial converter support

选中如下组件：

USB driver for GSM and CDMA modems

```

x x      *** USB port drivers ***
x x      <*> USB Serial Converter support --->
x x      *** USB Miscellaneous drivers ***

x x      Power management options --->
x x      [*] Networking support --->
x x      [ ] Device Drivers --->
x x      File systems --->

x x      <*> Sound card support --->
x x      HID support --->
x x      [ ] USB support --->
x x      <*> MMC/SD/SDIO card support --->

x x      < > USB Xircom / Entegra Single Port Serial Driver
x x      <*> USB driver for GSM and CDMA modems
x x      < > USB ZyXEL omni.net LCD Plus Driver
```

选中后保存退出。

关于 Linux 内核编译配置方法，以客户 android 系统的配置规则为准，文档描述的方法仅供参考。

3.3 添加模块 PID/VID

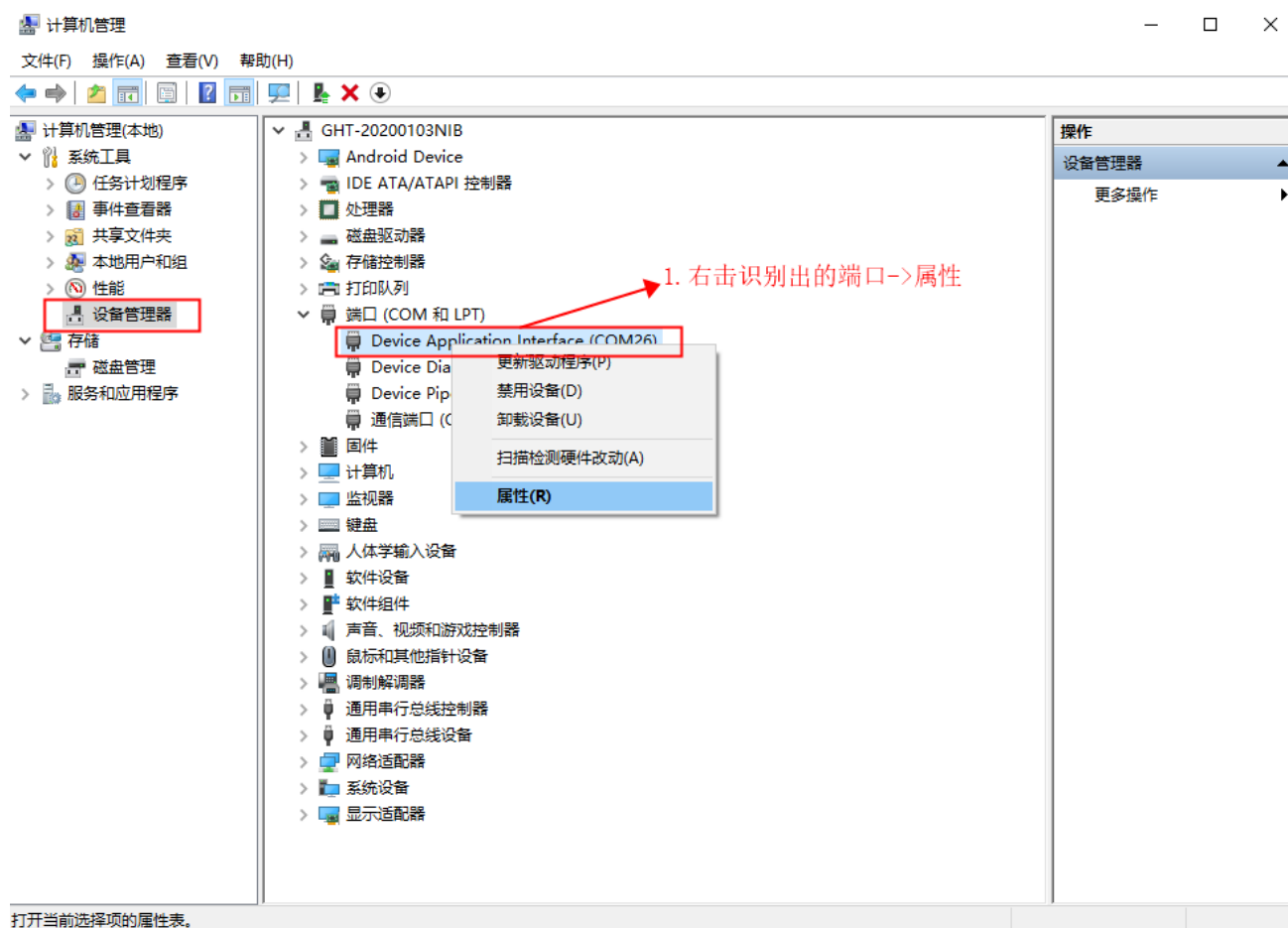
打开内核源码文件 option.c(路径一般为 drivers/usb/serial/option.c)。在源码中找到 option_ids 数组，在数组中添加模块的 PID 和 VID,以 MC919 添加 VID(0x2CB7)和 PID(0x0C01)为例。

```
static const struct usb_device_id option_ids[] = {
.....
{ USB_DEVICE(0x1508, 0x1001) },
.....
}
```

```
}
```

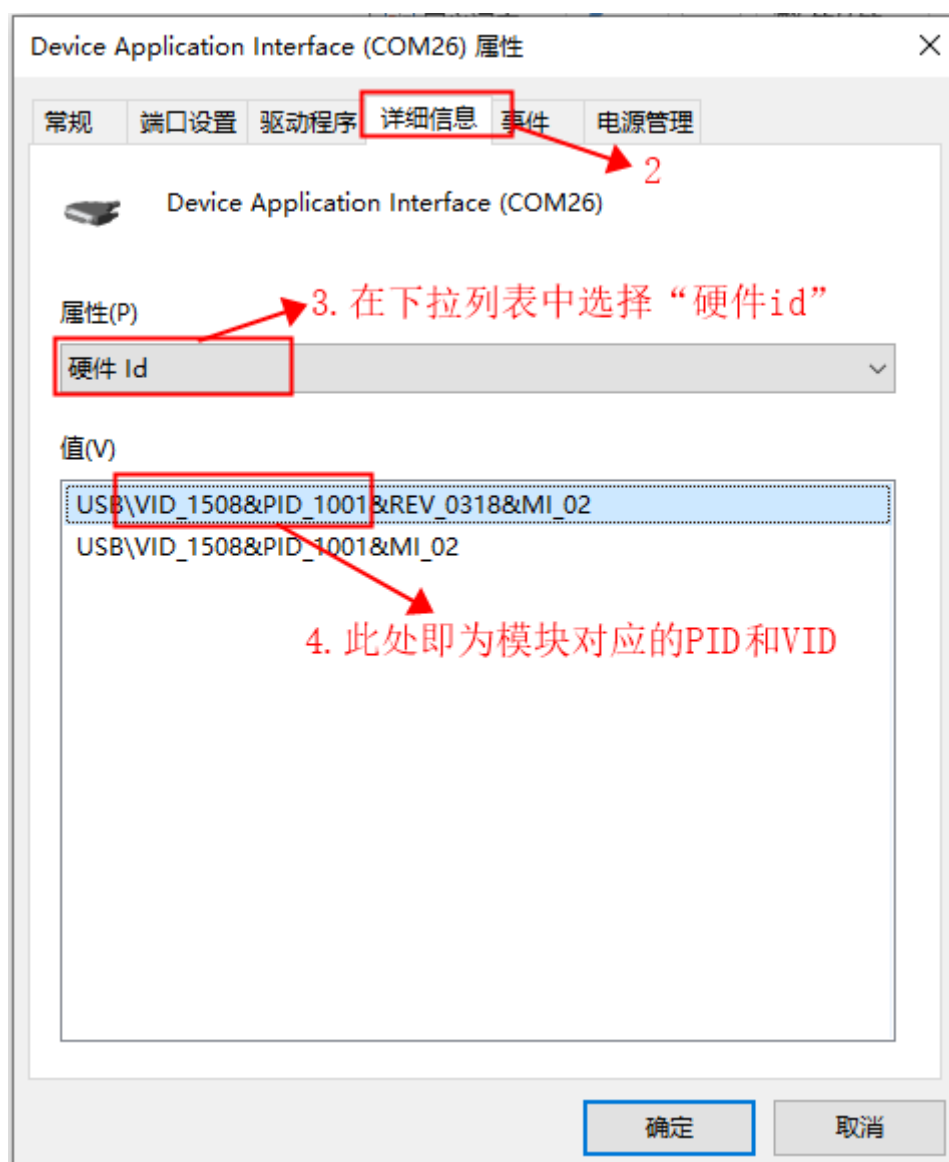
MC919 系列模块 PID/VID 可通过 android 系统命令“lsusb”获取，也可以咨询 Fibocom 技术人员。或者使用如下方法：

1. 将模块和 PC 相连，在“设备管理器->端口”里找到模块识别出的端口，右键点击后选择“属性”。



若模块连到电脑上后没有识别出端口，可以检查一下是否已正常安装相应驱动。

2. 点击“属性”后，依次选择“详细信息->硬件 id”，即可看到模块对应的 PID 和 VID。



3.4 修改 RILD 权限

RILD 程序执行过程中需要 root 权限，可在 rild.c (<\$android dir> /hardware/ril/rild/rild.c) 中的 main 函数中，去掉 switchuser 函数的调用来达到此目的，如下图 2 所示：

```
224     arg_device[sizeof(arg_device)-1] = 0;
225     arg_overrides[1] = "-d";
226     arg_overrides[2] = arg_device;
227     done = 1;
228
229     } while (0);
230
231     if (done) {
232         argv = arg_overrides;
233         argc = 3;
234         i = 1;
235         hasLibArgs = 1;
236         rilLibPath = REFERENCE_RIL_PATH;
237
238         |    RLOGD("overriding with %s %s", arg_overrides[1], arg_overrides[2]);
239     }
240 }
241 OpenLib:
242 #endif
243     //switchUser();
244
245     dlHandle = dlopen(rilLibPath, RTLD_NOW);
246
247     if (dlHandle == NULL) {
248         RLOGE("dlopen failed: %s", dlerror());
249         exit(-1);
250     }
251 }
```

图 2. RILD 中 main 函数修改



有的安卓版本中没有 switchUser 函数，如 Android10，那么可以忽略该节的内容。

3.5 配置 Android 启动脚本

配置 init.rc 文件添加 ril-daemon 进程(针对 Android7 及以下版本，Android8 及以上版本可跳过该步骤)：

```
service ril-daemon /system/bin/rild -l /system/lib64/libreference-ril.so -- -w 0
class main
socket rild stream 660 root radio
socket rild-debug stream 660 radio system
user root
group radio cache inet misc audio sdcard_rw log
```

- -l 指定 RIL 库加载路径
- -w 指定拨号方式，0：ppp 拨号，1：ECM 拨号

也可在 Android OS 属性文件/system/build.prop 中增加属性项 ril.fibocom.dialmode 来指定拨号方式。具体请参考章节 [9.2](#)。



如果 init.rc 文件中没有配置-w 参数，且没有用 ril.fibocom.dialmode 属性指定拨号方式，则 RIL 默认是 ppp 拨号方式，

3.6 加载 RIL 库文件

3.6.1 手动加载 RIL 库文件

对 Android8(不含)以前、64 位的安卓设备，需要将 libreference-ril.so 拷贝到/system/lib64 目录，adb 终端执行操作如下：

```
adb root
adb remount
adb push <local-dir>/libreference-ril.so /system/lib64
```

如果是 32 位系统，需要拷贝到/system/lib

```
adb push <local-dir>/libreference-ril.so /system/lib
```

对 Android8 及以上的版本，由于没有修改 init.rc 文件，故加载方法略有不同，一般是直接替换 Android 设备中原有的 RIL 库文件，具体方法如下：

1. 首先寻找设备中 RIL 库的路径。使用如下命令找到 ril.libpath 属性：

```
adb shell
getprop | grep ril
```

```
Administrator@GHT-20200103NIB D:\wu_Desktop\Tools\cmdex
> adb shell
rk3568_firefly_aioj:/ $ getprop|grep ril
[gsm.version.ril-impl]: []
[init.svc.ril-daemon]: [running]
[init.svc.debug_pid.ril-daemon]: [372]
[ril.currentapntype]: [default]
[ril.datachannel]: [/dev/ttyUSB6]
[ril.fibocom.NetifName]: [eth2]
[ril.fibocom.cgdcont0]: [1]
[ril.fibocom.cid]: [2]
[ril.fibocom.dialmode]: [1]
[ril.fibocom.version]: [L61x&MC61x_RIL_V11X.06.V1.0.3]
[ril.function.dataonly]: [1]
[rild.libargs]: [-d]
[rild.libpath]: [/vendor/lib64/libreference-ril.so]
[ro.boot.noril]: [false]
[ro.boottime.ril-daemon]: [6145985979]
[ro.ril.ecclist]: [112,911]
rk3568_firefly_aioj:/ $
adb.exe*[32]:48400
```

2. 将 RIL 库 (reference-ril.so 文件) push 到得到 rild.libpath 属性标识的路径下 (上图中为 /vendor/lib64/libreference-ril.so)。使用如下命令：

```
adb root
adb remount
adb push <local-dir>/libreference-ril.so /vendor/lib64/libreference-ril.so
```

```
Administrator@GHT-20200103NIB D:\wu_Desktop\Tools\cmdex
> adb root

Administrator@GHT-20200103NIB D:\wu_Desktop\Tools\cmdex
> adb remount
remount succeeded

Administrator@GHT-20200103NIB D:\wu_Desktop\Tools\cmdex
> adb push C:\Users\Administrator\Desktop\libreference-ril.so /vendor/lib64/libreference-ril.so
C:\Users\Administrator\Desktop\libreference-ril.so: 1 file pushed. 13.7 MB/s (208960 bytes in 0.015s)
```

3.6.2 自动打包 RIL 库文件

将 Fibocom 提供的 RIL 库源码压缩包，解压到<android-dir>/hardware/ril/reference-ril/目录下编译，编译系统固件的时候会自动编译 ril 库，并且会打包到 system.img 镜像文件中。

RIL 库源码可单独编译，操作如下：

```
cd <$android_dir>
source build/envsetup.sh
lunch <编译的项目>
mmm hardware/ril/reference-ril/
```

64 位系统生成 RIL 库 libreference-ril.so 文件目录

① Android7 及以下的安卓版本

```
<$android-dir>/out/target/product/.../system/lib64/libreference-ril.so
```

② Android8 及以上的版本

```
<$android-dir>/out/target/product/.../vendor/lib64/libreference-ril.so
```

32 位系统生成 RIL 库 libreference-ril.so 文件目录

① Android7 及以下的安卓版本

```
<$android-dir>/out/target/product/.../system/lib/libreference-ril.so
```

② Android8 及以上的版本

```
<$android-dir>/out/target/product/.../vendor/lib/libreference-ril.so
```

编译生成的 RIL 库 libreference-ril.so，可参照章节 [3.6.1](#) 手动加载到 android 系统中。

3.7 MC919 模块设备加载检测

将上述所有修改烧录入安卓设备后，通过 USB 连接 MC9x9 模块和安卓，并重启设备，预期这时可以识别出模块端口来。

先用 adb shell 命令进入安卓设备，然后用如下命令确认是否识别出了端口：

```
root@android:/ # ls /dev/ttyUSB* ①若设备正常挂载，将会有如下内容返回：
```

```
ls /dev/ttyUSB*
```

```
/dev/ttyUSB0
```

```
/dev/ttyUSB1
```

```
/dev/ttyUSB2
```

```
/dev/ttyUSB3
```

表 3. USB 端口功能说明

设备名	作用	备注
ttyUSB0	DIAG	获取 modem log 端口
ttyUSB1	MODEM	ppp 拨号端口

ttyUSB2	AT	RIL 程序发送 AT 命令请求和接收命令响应的端口。
ttyUSB3	NA	未用到



注意：

适配 RIL 需要用 AT+GTUSBMODE? 查询 USB 模式是否为 70，如果不是，建议先用 AT+GTUSBMODE=70 设置为 70。

4 ECM 拨号方式

ECM 拨号需要将 ril.fibocom.dialmode 属性设置为 1，具体请参考章节 [9.2](#)；同时要保证模块的 USBMODE 是 70。

4.1 获取 local ip,primary dns,second dns 地址

NDIS 拨号成功后,local-ip,primary-dns,second-dns 等信息会被写入到 Android OS 属性项 net.eth1.local-ip, net.eth1.dns1, net.eth1.dns2 中，执行如下操作可获取：

1. 获取 local ip，执行 Android OS 命令：

```
getprop net.eth1.local-ip
```

2. 获取 primary dns， second dns：执行 Android OS 命令：

```
getprop net.eth1.dns1
```

```
getprop net.eth1.dns2
```

如果 ECM 拨号使用的是其它网卡(如 usb0、eth2 等)，将上述属性中的“eth1”改成其它网卡名称即可。例如 net.usb0.local-ip、 net.usb0.dns1 等。

5 ppp 拨号方式

5.1 修改 ip-down ip-up 文件

修改 out/target/product/...../system/etc/ppp/目录下的 ip-up,ip-down 文件。

ip-up 文件修改如下:

```
#!/system/bin/sh
/system/bin/setprop "net.interfaces.defaultroute" "ppp0"
/system/bin/setprop "net.ppp0.dns1" "$DNS1"
/system/bin/setprop "net.ppp0.dns2" "$DNS2"
/system/bin/setprop "net.ppp0.local-ip" "$IPLOCAL"
/system/bin/setprop "net.ppp0.remote-ip" "$IPREMOTE"
exit 0
```

ip-down 文件修改如下:

```
#!/system/bin/sh
case $1 in
ppp1)
echo 0 > /proc/sys/net/ipv4/ip_forward;
esac
rm /etc/ppp/ppp*.pid
# Use interface name if linkname is not available
NAME=${LINKNAME:-"$1"}
/system/bin/setprop "net.$NAME.local-ip" ""
/system/bin/setprop "net.$NAME.remote-ip" ""
/system/bin/setprop "net.interfaces.defaultroute" ""
/system/bin/setprop "net.ppp0.dns1" ""
/system/bin/setprop "net.ppp0.dns2" ""
/system/bin/setprop "net.ppp0.local-ip" ""
/system/bin/setprop "net.ppp0.remote-ip" ""
```

若安卓属性项不是 net.ppp0.xxx，则需要修改为对应 ppp 拨号端口，如 net.modem.xxx。

ppp 拨号成功后，IP 地址，primary dns，second dns 被写入到属性 net.ppp0.local-ip, net.ppp0.dns1, net.ppp0.dns2 中，执行如下操作可获取：

1. 获取 local ip: 执行 Android OS 命令”

```
getprop net.ppp0.local-ip
```

2. 获取 primary dns，second dns：执行 Android OS 命令

```
getprop net.ppp0.dns1
```

```
getprop net.ppp0.dns2
```



注意：ip-up,ip-down 文件需要具有执行权限 555。

6 APN 设置和拨号状态查询

在 APN 为空的情况下，需要设置 APN 才能拨号，有两种方式。

6.1 手动添加 APN

请执行如下操作(仅适用于调试):

- 将 MC919 模块安装在 Android 开发板上
- 在开发板上插入 SIM 卡
- 打开电源，等待开机。
- 设置 ppp 拨号或者 NDIS 拨号用到的 APN：
 - 设置->无线和网线（更多）->移动网络->接入点名称（APN）
 - 按菜单键选择“新建 APN”，输入 Name, APN, MCC, MNC，按菜单键保存。

如果运营商有提供特殊 APN 需要鉴权用户名、密码时，请使用运营商提供的 APN、鉴权用户名、鉴权密码。

- 选用设置的 APN 拨号，可以使用以下命令确认 RIL 是否拨号成功：

1. 查看 usb0 或者 ppp0 网络接口状态，执行 Android OS 命令

```
ifconfig
```

2. 获取 ppp 拨号或者 NDIS 拨号成功后的 local ip，执行 Android OS 命令

```
getprop net.ppp0.local-ip
```

```
getprop net.usb0.local-ip
```

6.2 配置 apns-conf.xml 文件

Android 设备路径：/system/etc/apns-conf.xml

修改 apns-conf.xml 文件时，根据 SIM 卡的实际信息进行添加，添加的主要信息有：apn,mcc,mnc,user,password,type,protocol 等。

以下是列举的某个 APN 的信息添加：

```
<apn carrier="China Mobile"
```

```
apn="cmnet"
mcc="460"
mnc="04"
user=""
server=""
password=""
proxy=""
port=""
mmsproxy=""
mmsport=""
mmsc=""
type="default,net,supl"
preferred="true"
localized_name="APN_NAME_CMNET"
protocol="IPV4V6"
roaming_protocol="IPV4V6"
/>
```

更新 APN 配置列表后，由于 Android 设备启动很多次，telephony.db 已经初始化完成，因此需要先删除数据库文件，然后再次启动，才会加载新的配置文件。

Android 设备上 telephony.db 的路径：
/data/data/com.android.providers.telephony/databases/telephony.db

7 RIL 调试方法

1. 获取 RIL log 的方法通过在 windows 操作系统的 cmd 窗口，输入如下命令：

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

2. 获取 Android OS 运行 log，输入如下命令：

```
adb logcat -v time
```

3. 获取 ppp 拨号 pppd log，输入如下命令：

```
adb logcat -s pppd
```

4. 在一些特殊使用场景，如在许多设备上或很长一段时间内执行测试，可执行以下命令将 log 保存在 Android 系统指定的目录下：

```
adb shell
```

```
logcat -b radio -v time > <$android_dir>/<filename>
```

5. 提取设备中保存的 log 文件，执行如下命令：

```
adb pull <filename> <$local_directory>
```

8 LOG 中 TAG 标签说明

RIL log 中各个进程的 log 都在一个文件中，可以通过 tag 标签来区分 log 来自于那个文件，tag 标签与对应的文件，见下表。

表 4. RIL log 中 TAG 标签说明

TAG	文件
RLOG-RIL	hardware/ril/reference-ril/reference-ril.c
RLOG-AT	hardware/ril/reference-ril/atchannel.c
RILC	hardware/ril/libril/ril.cpp
RILD	hardware/ril/rild/rild.c
RILJ	frameworks/opt/telephony/src/java/com/android/internal/telephony/RIL.java

9 当前维护的属性

RIL 库中有多个自定义的系统属性，客户可以通过修改这些属性的值来自定义一些配置。修改这些属性的方法分两种：临时设置和永久设置。临时设置的属性在设备重启后失效，永久设置的属性在设备重启后依然保留。

9.1 属性设置方法

9.1.1 临时设置

可以使用 `setprop` 命令临时设置某属性，该命令的使用方法如下：

```
setprop <NAME> <VALUE>
```

客户也可以使用 `getprop` 命令来查看某属性的值。该命令的使用方法如下：

```
getprop <NAME>
```

下图是将属性“`ril.fibocom.dialmode`”从 1 临时设置成 2 的示例：

```
rk3568_firefly_aioj:/ $ setprop ril.fibocom.dialmode 1
rk3568_firefly_aioj:/ $ getprop ril.fibocom.dialmode
1
rk3568_firefly_aioj:/ $ getprop ril.fibocom.dialmode
1
rk3568_firefly_aioj:/ $
rk3568_firefly_aioj:/ $
rk3568_firefly_aioj:/ $
rk3568_firefly_aioj:/ $
rk3568_firefly_aioj:/ $
rk3568_firefly_aioj:/ $ setprop ril.fibocom.dialmode 2
rk3568_firefly_aioj:/ $
rk3568_firefly_aioj:/ $
rk3568_firefly_aioj:/ $
rk3568_firefly_aioj:/ $
rk3568_firefly_aioj:/ $
rk3568_firefly_aioj:/ $
rk3568_firefly_aioj:/ $
rk3568_firefly_aioj:/ $
rk3568_firefly_aioj:/ $ getprop ril.fibocom.dialmode
2
rk3568_firefly_aioj:/ $
adb.exe*[32]:48400
```

1. 某属性的值原为1

2. 使用setprop命令将该值改为2

3. 该属性的值被成功改为2

9.1.2 永久设置

在 `/system/build.prop` 文件里增加属性和它们的值，就可以永久地保存这些属性，在设备重启后依然保留。具体步骤如下：

1. 使用 `adb pull` 命令将 `/system/build.prop` 文件从设备中 pull 出来：

adb root	❶ 需要进入 root 模式
adb remount	❷ 将 system 分区重新挂载成可读写模式
adb pull /system/build.prop <LOCAL_PATH>	❸ 将 build.prop 文件 pull 出来

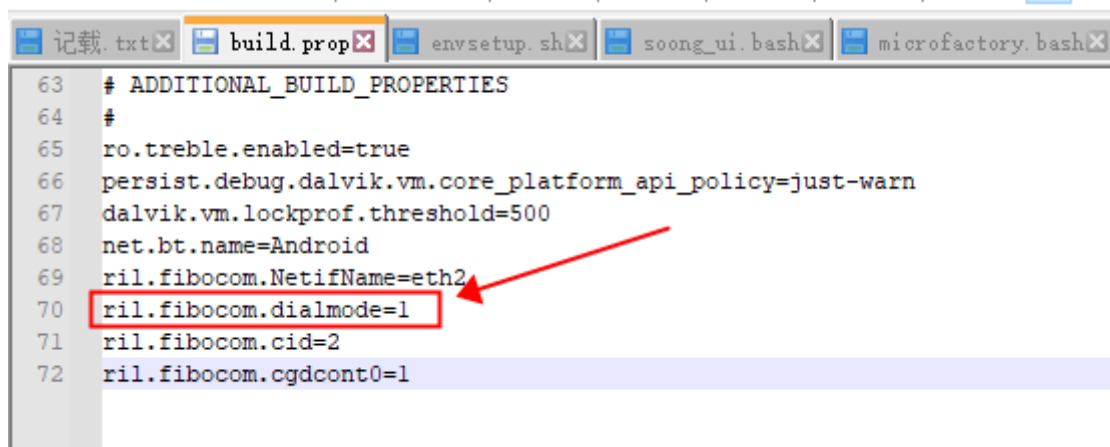
```
Administrator@GHT-20200103NIB D:\wu_Desktop\Tools\cmdr
> adb root

Administrator@GHT-20200103NIB D:\wu_Desktop\Tools\cmdr
> adb remount
remount succeeded

Administrator@GHT-20200103NIB D:\wu_Desktop\Tools\cmdr
> adb pull /system/build.prop d:/log/build.prop
/system/build.prop: 1 file pulled. 0.6 MB/s (2557 bytes in 0.004s)

Administrator@GHT-20200103NIB D:\wu_Desktop\Tools\cmdr
>
```

2. 在 build.prop 文件末尾加入我们的属性(以“ril.fibocom.dialmode”为例)，然后保存修改：



```
63 # ADDITIONAL_BUILD_PROPERTIES
64 #
65 ro.treble.enabled=true
66 persist.debug.dalvik.vm.core_platform_api_policy=just-warn
67 dalvik.vm.lockprof.threshold=500
68 net.bt.name=Android
69 ril.fibocom.NetifName=eth2
70 ril.fibocom.dialmode=1
71 ril.fibocom.cid=2
72 ril.fibocom.cgdcont0=1
```

3. 将 build.prop 文件 push 进设备里的原位置，并重启设备：

adb push d:/log/build.prop /system/build.prop	❶ 将 build.prop 文件重新 push 进设备里
adb reboot	❷ 重启设备

```
Administrator@GHT-20200103NIB D:\wu\Desktop\Tools\cmdr
> adb push d:/log/build.prop /system/build.prop
d:/log/build.prop: 1 file pushed. 0.1 MB/s (2557 bytes in 0.017s)

Administrator@GHT-20200103NIB D:\wu\Desktop\Tools\cmdr
> adb reboot
```

4. 使用 adb shell 命令进入安卓设备，然后用 getprop|grep ril 命令即可看到我们设置的属性。如下：

adb shell	❶ 进入安卓设备
getprop grep ril	❷ 查找与“ril”相关的属性

```
Administrator@GHT-20200103NIB D:\wu\Desktop\Tools\cmdr
> adb shell
rk3568_firefly_aioj:/ $ getprop|grep ril
[gsm.version.ril-impl]: []
[init.svc.ril-daemon]: [running]
[init.svc_debug_pid.ril-daemon]: [383]
[ril.currentapntype]: [default]
[ril.datachannel]: [/dev/ttyUSB6]
[ril.fibocom.NetifName]: [eth2]
[ril.fibocom.cgdcont0]: [1]
[ril.fibocom.cid]: [2]
[ril.fibocom.dialmode]: [1]
[ril.fibocom.version]: [L61x&MC61x_RIL_V11X.06.V1.0.3_RESET]
[ril.function.dataonly]: [1]
[rild.libargs]: [-d]
[rild.libpath]: [/vendor/lib64/libreference-ril.so]
[ro.boot.noril]: [false]
[ro.boottime.ril-daemon]: [6395227170]
[ro.ril.ecclist]: [112,911]
rk3568_firefly_aioj:/ $
```

我们自己设置的属性

9.2 ril.fibocom.dialmode

9.2.1 简介

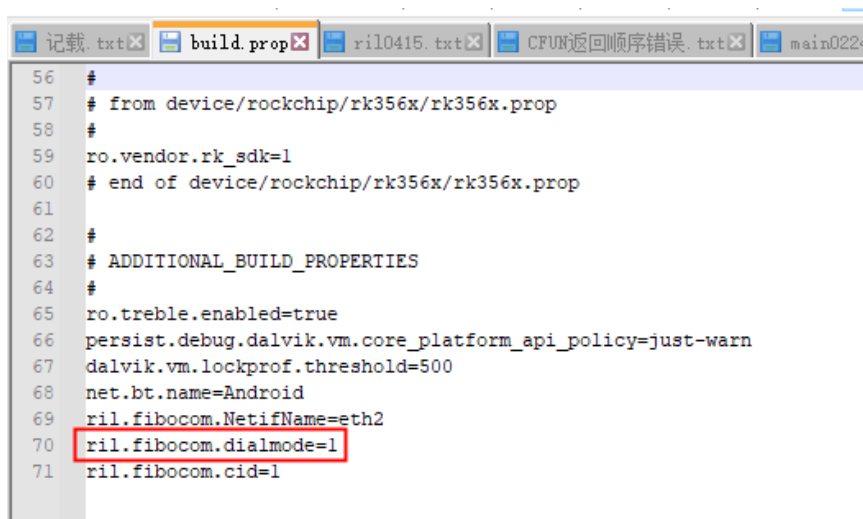
该属性用于指定 RIL 使用的拨号方式是 ECM、PPP 还是 RNDIS 拨号。

9.2.2 取值范围

表 5. dialmode 取值范围

ril.fibocom.dialmode	拨号方式
0(默认)	PPP
1	ECM
2	RNDIS

9.2.3 示例



```
56 #
57 # from device/rockchip/rk356x/rk356x.prop
58 #
59 ro.vendor.rk_sdk=1
60 # end of device/rockchip/rk356x/rk356x.prop
61
62 #
63 # ADDITIONAL_BUILD_PROPERTIES
64 #
65 ro.treble.enabled=true
66 persist.debug.dalvik.vm.core_platform_api_policy=just-warn
67 dalvik.vm.lockprof.threshold=500
68 net.bt.name=Android
69 ril.fibocom.NetifName=eth2
70 ril.fibocom.dialmode=1
71 ril.fibocom.cid=1
```

9.3 ril.fibocom.NetifName

9.3.1 作用

自定义 RIL 拨号使用的网卡(常用于 ECM 拨号)。

RIL 发起 ECM 拨号时, 选择网卡的顺序是: NetifName 指定->RIL 预设(如 eth1)->usb0。只有前两种方法都没有找到网卡时, 才会使用默认网卡“usb0”。

9.3.2 取值范围

常用的取值为 usb0、eth0 和 eth1 等，默认值是 usb0。

9.3.3 示例



```
56 #
57 # from device/rockchip/rk356x/rk356x.prop
58 #
59 ro.vendor.rk_sdk=1
60 # end of device/rockchip/rk356x/rk356x.prop
61
62 #
63 # ADDITIONAL_BUILD_PROPERTIES
64 #
65 ro.treble.enabled=true
66 persist.debug.dalvik.vm.core_platform_api_policy=just-warn
67 dalvik.vm.lockprof.threshold=500
68 net.bt.name=Android
69 ril.fibocom.NetifName=eth2
70 ril.fibocom.dialmode=1
71 ril.fibocom.cid=1
```

9.4 ril.fibocom.cid

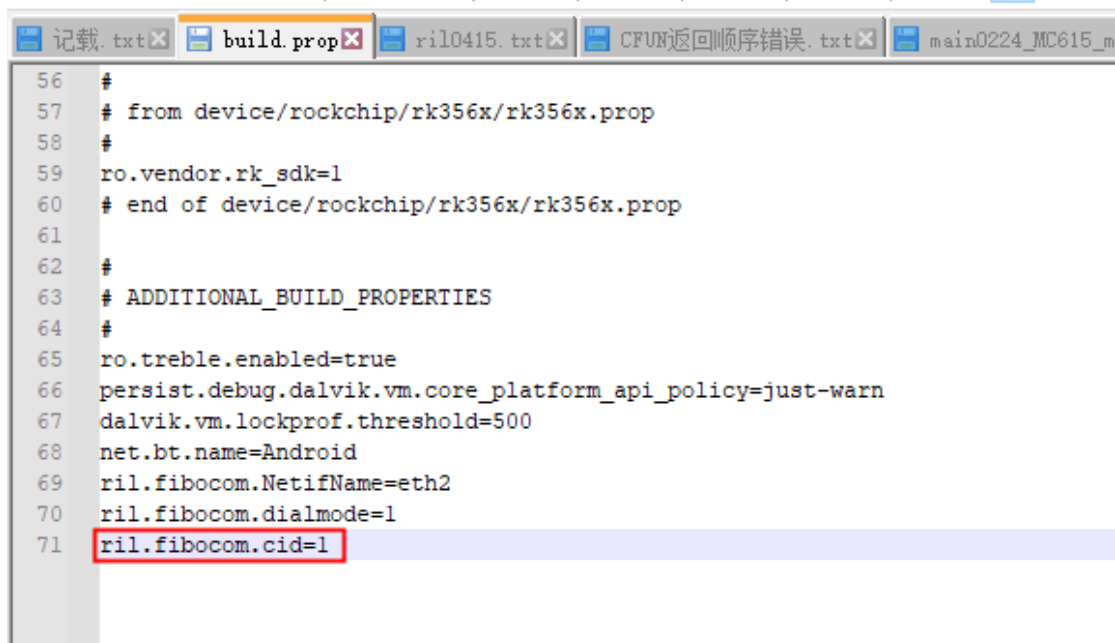
9.4.1 作用

自定义模块使用哪一路的 cid 进行拨号。

9.4.2 取值范围

该属性的取值范围是[1,7]，默认值是 1。

9.4.3 示例



```
56 #
57 # from device/rockchip/rk356x/rk356x.prop
58 #
59 ro.vendor.rk_sdk=1
60 # end of device/rockchip/rk356x/rk356x.prop
61
62 #
63 # ADDITIONAL_BUILD_PROPERTIES
64 #
65 ro.treble.enabled=true
66 persist.debug.dalvik.vm.core_platform_api_policy=just-warn
67 dalvik.vm.lockprof.threshold=500
68 net.bt.name=Android
69 ril.fibocom.NetifName=eth2
70 ril.fibocom.dialmode=1
71 ril.fibocom.cid=1
```

9.5 ril.fibocom.cgdcont0

9.5.1 作用

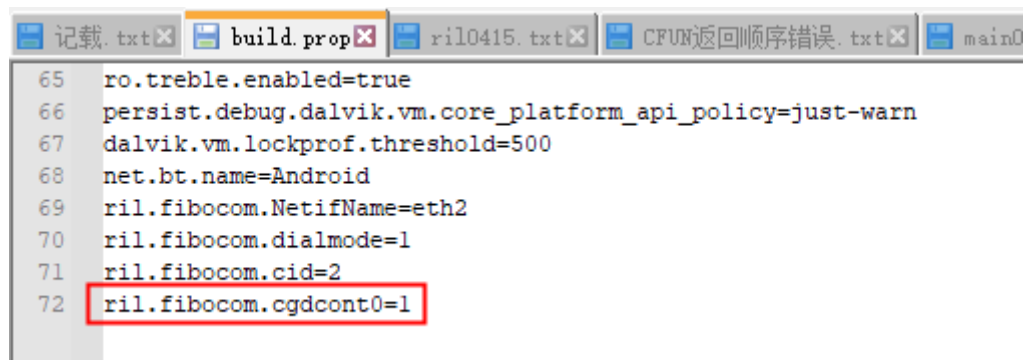
该属性用于自定义是否设置 cid0 的 APN。

9.5.2 取值范围

0: 不设置 cid0 的 APN(默认)

1: 设置 cid0 的 APN

9.5.3 示例



```
65 ro.treble.enabled=true
66 persist.debug.dalvik.vm.core_platform_api_policy=just-warn
67 dalvik.vm.lockprof.threshold=500
68 net.bt.name=Android
69 ril.fibocom.NetifName=eth2
70 ril.fibocom.dialmode=1
71 ril.fibocom.cid=2
72 ril.fibocom.cgdcont0=1
```

10 常见问题

10.1 无法正常的进行数据业务

1. 查询拨号状态，是否获取了 local ip, primary dns, second dns。
2. 如果设备不拨号，请检测 Android 设置是否设置了 APN, APN 的设置可以通过 Andorid UI 界面设置，一般流程如下：

Setings -> Network& Internet -> More -> Mobile network -> Advanced -> Access Point Names（见章节 [6](#)）。

10.2 电话打不通

如果电话打不通，确认 SIM 卡是否支持语音功能，如果支持需要向 Fibocom 技术支持人员确认模块在当前网络制式下是否支持电话功能。

10.3 无法发送短信

如果无法发送短信，确认 SIM 卡是否支持短信功能，如果支持需要向 Fibocom 技术支持人员确认模块在当前网络制式下是否支持短信功能。

10.4 RIL 没有正常工作

有很多原因导致 Android 设备无法启动 RIL，如果出现这类现象请按照以下流程进程排查：

1. RIL 库没有被正确的加载

进入 Android shell 终端：输入 `cat init.rc | grep ril` ；期待"service ril-daemon /vendor/bin/hw/rild -l /system/lib64/libreference-ril.so -- -w 0" 出现，如果没有请修改对应项目的 init.rc 以便加载正确的库文件。



适用于 Android7 及以下，Android8 以后的安卓版本不需要修改 init.rc 文件。详见章节 [3.5](#)。

2. RILD 是否正在运行

进入 Android shell 终端：输入 `getprop | grep init.svc.ril-daemon`；检测 RIL 的运行状态，期待的结果：

[init.svc.ril-daemon]: [running]

3. USB 串口是否被枚举

进入 Android shell 终端：输入 `ls /dev/ttyUSB* -l` 以检测 usb 串口是否被枚举。如果没有枚举端口，RIL 也无法正常工作。

10.5 设备不支持 PPP 拨号

使用如下步骤来编译、安装 PPP：

- 加入内核支持

按照 [3.2](#) 节的步骤开始编译内核，输入 “make menuconfig” 命令后，选择如下组件：

Device Drivers --- >

Network device support --- >

<*> PPP(point-to-point protocol) support

[*] PPP multilink support (EXPERIMENTAL)

<*> PPP support for async serial ports

<*> PPP support for sync tty ports

<*> PPP Deflate compression

<*> PPP BSD-Compress compression

具体步骤如下图所示：

```
boot options --->
Userspace binary formats --->
Power management options --->
CPU Power Management --->
[*] Networking support --->
[ ] Device Drivers --->
    Firmware Drivers --->
[ ] ACPI (Advanced Configuration and Power Interface) Support ----
File systems --->
[ ] Virtualization ----
    Kernel hacking --->
    Security options --->
-*- Cryptographic API --->
    Library routines --->
```



```

-*> Device Tree and Open Firmware support --->
<> Parallel port support ----
[*] Block devices --->
<*> NVM Express block device
    Misc devices --->
    SCSI device support --->
<*> Serial ATA and Parallel ATA drivers (libata) --->
[*] Multiple devices driver support (RAID and LVM) --->
<> Generic Target Core Mod (TCM) and ConfigFS Infrastructure ----
[ ] Fusion MPT device support ----
    IEEE 1394 (FireWire) support --->
[*] Network device support --->
[ ] Open-Channel SSD target support ----
    Input device support --->
    Character devices --->
    I2C support --->
[*] SPI support --->
<> SPMI support ----
<> HSI support ----
    PPS support --->
    PTP clock support --->

```

```

[ ] HIPPI driver support
-*> PHY Device support and infrastructure --->
<> Micrel KS8995MA 5-ports 10/100 managed Ethernet switch
<*> PPP (point-to-point protocol) support
<*>     PPP BSD-Compress compression
<*>     PPP Deflate compression
[*]     PPP filtering
<*>     PPP MPPE compression (encryption)
[*]     PPP multilink support
<*>     PPP over Ethernet
<*>     PPP over L2TP
<*>     PPP on L2TP Access Concentrator
<*>     PPP on PPTP Network Server
<*>     PPP support for async serial ports
<*>     PPP support for sync tty ports
<*>     SLIP (serial line) support
[*]     CSLIP compressed headers
[ ]     Keepalive and linefill
[*]     Six bit SLIP encapsulation
<*>     USB Network Adapters --->
[*]     Wireless LAN --->

```



1. 按“ENTER”键进入对应目录，按“y”选择对应选项，按“n”取消选择对应选项。
2. “PPP(point-to-point) support”下之前所提的选项必选，此例中为全部选上，然后编译进内核。

完成这些步骤后，进行内核的编译。将生成的内核映像文件下载到板子上。内核启动后，会在 /dev 目录下生成 ppp 设备节点。如：

```
rk3399pro:/dev # ls ppp -l
crw-rw---- 1 radio vpn 108,  0 2020-11-29 13:23 ppp
rk3399pro:/dev #
```

- 下载 PPP

使用如下命令得到 PPP 相关文件：

```
wget -c https://ftp.samba.org/pub/ppp/ppp-2.4.5.tar.gz
```

然后使用解压命令解压获取到的文件，生成目录 ppp-2.4.5：

```
tar -zxvf ppp-2.4.5.tar.gz
```

进入目录 ppp-2.4.5 后,执行以下命令进行交叉编译：

```
# ./configure
# make CC=arm-linux-gcc
```



若执行“make CC=arm-linux-gcc”时提示 “arm-linux-gcc：命令未找到” 错误，可能是因为 Linux 上没有安装 arm-linux-gcc 工具或该工具配置错误，客户可上网寻找解决办法或询问我司相关人员。

将 ppp-2.4.5 目录下 pppd , chat , pppdump , pppstats 下生成的可执行程序 pppd , chat, pppdump, pppstats 拷贝到开发板/usr/sbin 目录下。

按照 [5.1](#) 节的内容，在指定位置修改 ip-up, ip-down 文件的内容。至此，发起 PPP 拨号前的准备工作完成。

11 附录 A 参考文件

表 6. 术语和缩写

缩写词	描述
GSM	全球移动通信系统
VID	供应商 ID
PID	产品识别码
RIL	无线接口层
RILD	无线接口层守护进程
APN	接入点名称