

Simcom Android ril 适配 1.13

	Histoty
2016-06-07	First release
2018-05-21	Version 1.13

一 SIMCOM 模块 USB 相关描述

SIM7100/SIM7200/SIM7230/SIM7250/7500/7600/7800 系列模块的 USB VID 是 0x1E0E PID 是 0x9001

SIM5360/SIM6320/SIM5320 的 USB VID: 0x05C6 PID:0x9000

SIM7100 系列作为 Slave USB 设备，配置如下表

Interface number		
0	USB serial	Diagnostic Interface
1	USB serial	GPS NMEA Interface
2	USB serial	AT port Interface
3	USB serial	Modem port Interface
4	USB serial	USB Audio Interface
5	USB Net	NDIS wwan interface
6	USB adb	Android add debug port

SIM7100/7500/7600/7800 系列可以支持 NDIS 方式拨号！但默认的情况下均采用 ppp 拨号方式。

下面适配的步骤中请依据拨号方式操作！下列配置除指明 ndis 使用外，默认 ppp 和 ndis 都需要配置。

Android7.1 由于 selinux 权限限制，拨号脚本无法被调用，建议使用 ndis 拨号，如需 ppp 拨号请添加 selinux 策略。

二 USB 串口驱动使用

PPP 和 NDIS 拨号都需如下配置。

1 USB Serial 的内核配置支持

CONFIG_USB_SERIAL=y

CONFIG_USB_SERIAL_WWAN=y

CONFIG_USB_SERIAL_OPTION=y

如果是适配 SIM5360/SIM6320/SIM5320 等模块： 因为一般的 linux 内核都有预置这些模块的 VID 和 PID。 所以。做完这步可以直接跳到第四步 RIL 库应用

2 修改驱动代码增加 SIM7100 的 VID/PID

找到内核源码文件 option.c(一般情况下，路径在 drivers/usb/serial/option.c)

- 如果是较新的内核版本（V3.2 以上）

```
#define SIMCOM_SIM7100_VID          0x1E0E
#define SIMCOM_SIM7100_PID          0x9001

//for SIM7100 modem for NDIS
static const struct option_blacklist_info simcom_sim7100_blacklist = {
    .reserved = BIT(5),
};
在 option_ids 列表中增加
... ..

//for SIM7100 modem for NDIS
{ USB_DEVICE(SIMCOM_SIM7100_VID, SIMCOM_SIM7100_PID),
    .driver_info = (kernel_ulong_t)& simcom_sim7100_blacklist
},
... ..
```

- 如果是较低的内核版本，

```
#define SIMCOM_SIM7100_VID          0x1E0E
#define SIMCOM_SIM7100_PID          0x9001

在 option_ids 列表中增加

{ USB_DEVICE(SIMCOM_SIM7100_VID, SIMCOM_SIM7100_PID)}, /*SIM7100 */
```

3. 预留 NDIS 口：

如果采用 PPP 拨号方式： 可以不做这一步。

如果采用 NDIS 拨号方式， 需要这一步步骤：

在 option.c 中的 option_probe 添加下面一段代码。

```
/* sim7100 */
if (serial->dev->descriptor.idVendor == SIMCOM_SIM7100_VID &&
```

```
serial->dev->descriptor.idProduct == SIMCOM_SIM7100_PID &&  
serial->interface->cur_altsetting->desc.bInterfaceNumber == 5 )  
return -ENODEV;
```

3 内核调试信息打印

如果驱动正确编译到内核，内核开机找到模块后，会打印如下信息

```
usb 1-1: new high speed USB device using rt3xxx-ehci and address 2  
option 1-1:1.0: GSM modem (1-port) converter detected  
usb 1-1: GSM modem (1-port) converter now attached to ttyUSB0  
option 1-1:1.1: GSM modem (1-port) converter detected  
usb 1-1: GSM modem (1-port) converter now attached to ttyUSB1  
option 1-1:1.2: GSM modem (1-port) converter detected  
usb 1-1: GSM modem (1-port) converter now attached to ttyUSB2  
option 1-1:1.3: GSM modem (1-port) converter detected  
usb 1-1: GSM modem (1-port) converter now attached to ttyUSB3  
option 1-1:1.4: GSM modem (1-port) converter detected  
usb 1-1: GSM modem (1-port) converter now attached to ttyUSB4
```

dev/ttyUSB0~4 就会生成，上层应用就可以通过这些设备和模块交互了（发送 AT 命令等）。

三 USB NDIS NET 使用

如果是 PPP 拨号方式，直接跳过这一步，进入第四步 RIL 库应用

Ndis 拨号需在 android 的/system/build.prop 中设置 rild.simcom.ndis=1

1 内核配置支持

7100 NDIS 拨号使用驱动 qmi_wwan.c，7500/7600 NDIS 拨号使用驱动 simcom_wwan.c（与 ril 打包在一起）

（1）Linux 从 3.4.1 开始已经把 QMI WWAN 驱动包含到源码里。
因此，如果内核版本高于或者等于 3.4.1，只需将这三个配置项打开就可。

```
CONFIG_USB_WDM=y  
CONFIG_USB_USBNET=y  
CONFIG_USB_NET_QMI_WWAN=y
```

并且在 qmi_wwan.c 里增加 SIM7100 的 VID/PID,设置端口号 5

```
{ QMI_FIXED_INTF (0x1e0e, 0x9001,5)},    /* SIM7100 Modem Device */
```

(2)如果内核版本低于 3.4.1, 将这两个配置项打开, 并使用我们提供的驱动。

```
CONFIG_USB_WDM=y
```

```
CONFIG_USB_USBNET=y
```

(3) qmi_wwan.c 放在 drivers/net/usb 目录下, 并且修改 Makefile

```
obj-$(CONFIG_USB_USBNET) += usbnet.o qmi_wwan.o
```

simcom_wwan.c 放在 drivers/net/usb 目录下, 并且修改 Makefile

```
obj-$(CONFIG_USB_USBNET) += usbnet.o simcom_wwan.o
```

驱动正确编译到内核, 内核开机后连到模块会打印如下信息

```
qmi_wwan 1-1:1.5: cdc-wdm0: USB WDM device
qmi_wwan 1-1:1.5: wwan0: register 'qmi_wwan' at usb-rt3xxx-1, Qualcomm Gobi
wwan/QMI device, d6:d8:6c:10:b0:0e
```

2 使用 ifconfig 查看网卡信息, 默认处于 down 状态。

```
wwan0      Link encap:Ethernet  HWaddr D6:D8:6C:10:B0:0E
            BROADCAST MULTICAST  MTU:1500  Metric:1
            RX packets:0 errors:0 dropped:0 overruns:0 frame:0
            TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
            RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

四 RIL 库应用

1. 解压 simcom_rilXX_XXXXXXX.tar.gz

里面文件:

init.rc

rild

libril.so

libreference-ril.so

init.gprs-pppd (PPP 拨号用)

3gdata_call.conf (PPP 拨号用)

qmi_wwan.c (7100 NDIS 拨号用)

simcom_wwan.c (7500/7600/7800 NDIS 拨号用)

gps.simcom.so (GPS 库)

chat (ppp 拨号用)

2. 默认情况下 RIL 支持的一些功能是关闭的，如果客户有需要相关的功能，需要在 android 系统上添加一些属性。

目前支持的功能：

GPS: rild.simcom.gps=1

1: 启用 GPS, 0 或不设置: 不启用

USSD: rild.simcom.ussd=1

STK: rild.simcom.stk=1

NDIS: rild.simcom.ndis=1

(NDIS 支持 7100 系列及后续系列)

STOPGPS: rild.simcom.stopgps=1

Android 屏幕关闭的时候 GPS 关闭。

GPSLOG: rild.simcom.gpsloglevel=1

默认的情况下，GPS 库会关闭大部分 log，如果客户遇到 GPS 定位相关问题的时候，可以将此属性设置为 1，这样可以抓更完整的 log，便于分析问题。

NETCLOSE: rild.simcom.netclose=1

有部分 6320 客户需要用内部协议栈和 android 同时拨号上网，内部协议栈拨号成功后 android 会无法成功拨号。需要先关闭内部协议中。将此属性设为 1，android 拨号前会先关闭内部协议栈拨号

CLVL: rild.simcom.clvl=* (*: 0~7)

有客户需要通过 ril 修改模块音量的，可以设置此属性，ril 会在 sim 卡 ready 后将此值通过 AT+CLVL=* 设置到模块

CSDVC rild.simcom.csdcv=*

部分客户需要通过 ril 设置 CSDVC 的，可以通过设置此属性，ril 会在 Sim 卡 ready 后将此值通过 AT+CSDVC=* 设置到模块

设置成功后，开机通过 adb shell getprop 可以查看到这些属性

(客户调试的时候可以直接修改/system/build.prop 这个文件，这样可以不需要重新编译系统，灵活调整开关)

Android 4.0 4.2:

这两个版本的 android 目前没有统一 ril 版本，需要上述功能的话，需要单独另外提供

3. Init.rc 文件一般放在 android 源码的 /devices/\$vendor_name/\$product_name/ 下面。具体到各个项目可能会有所区别。如果看起来不是很明显的，可以先用 adb shell 看一下设备根目录下的 init.rc，再反推一下源码所用的文件。

根据拨号方式不同, init.rc 所需要修改的有所区别。

(android6.0 与其他版本不同)

NDIS 拨号:

下面截图是 simcom 提供的 init.rc 中的一处截图，将这部分移植到客户源码 init.rc

```
#modified by simcom
#-----
service ril-daemon /system/bin/rild -l /system/lib/libreference-ril.so
    class main
        socket rild stream 660 root radio
        socket rild-debug stream 660 radio system
#---- if need gps feature, unmark next line -----
#    socket rild-gps stream 660 radio system
    user root
    group radio cache inet misc audio sdcard_rw
```

如果需要支持 GPS， 把 GPS 那行打开

客户如果采用的是 **NDIS** 拨号，则操作到此步后，进入第 4 步（更新 **rild** 及库文件）

Android4/5/7PPP 拨号:

Init.rc 中可能存在修改权限不成功，需要手动修改权限。

```
#add by simcom
# change init.gprs-pppd for recovery mode
    chmod 0777 /etc/init.gprs-pppd
    chmod 0777 /etc/3gdata_call.conf
```

```

#modified by simcom
#-----
service ril-daemon /system/bin/rild -l /system/lib/libreference-ril.so
    class main
        socket rild stream 660 root radio
        socket rild-debug stream 660 radio system
#---- if need gps feature, unmark next line -----
#    socket rild-gps stream 660 radio system
    user root
    group radio cache inet misc audio sdcard_rw

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

```

Android6.0 拨号:

```

#add by simcom
# change init.gprs-pppd for recovery mode
    chmod 0777 /etc/init.gprs-pppd
    chmod 0777 /etc/3gdata_call.conf

#modified by simcom
#-----
service ril-daemon /system/bin/rild -l /system/lib/libreference-ril.so
    class main
        socket rild stream 660 root radio system
        socket rild-debug stream 660 radio system
        socket rild-ppp stream 660 radio system
    user root
    group radio cache inet misc audio sdcard_rw

service pppd_gprs /etc/init.gprs-pppd
#class main
    user root
    group radio cache inet misc
    disabled
    oneshot

```

上面截图是 simcom 提供的 init.rc 中的两处截图。客户请将这两部分移植到你们设备源码 init.rc 中

部分客户遇到的问题点:

a. 源码 init.rc 中一般已经自带了一部分上面配置, 按照上面截图修改。不要新增。(不能同时出现两个 service ril-daemon 或 service pppd_gprs)

b. 有些客户的设备很多路径是 read-only 的。此时 chmod 777

/etc/init.gprs-pppd 不能成功修改权限。可以将脚本指定到其他地方
此时假定路径修改为: /system/bin/ 则同时需要修改下面几处:

```
chmod 777 /system /system/bin/init.gprs-pppd  
service pppd_gprs /system/bin/init.gprs-pppd
```

c. 如果修改了 3gdata_call.conf 的路径, 那么修改修改 init.gprs-pppd 文件本身, init.gprs-pppd 里面指定了 3gdata_call.conf 的路径。需要修改这个路径

4. 修改 init.rc-pppd 脚本, 将延迟时间修改为 500

```
#/system/sbin/pppd $*  
# pppd was put into /system/bin instead of /system/sbin after SDK1.6  
/system/bin/pppd user $PPPD_USERNAME password $PPPD_PASSWORD connect 'chat -v -s -r "/var/log/chat.log" -  
f "/etc/3gdata_call.conf"' disconnect 'chat -r "/var/log/chat.log" -t 30 -e -v "" +++ATH "NO CARRIER"'  
$PPPD_DATAPORT 115200 mru 1280 mtu 1280 nodetach debug dump defaultroute usepeerdns novj novjccomp  
noipdefault ipcp-accept-local ipcp-accept-remote connect-delay 500 linkname ppp0
```

5. 添加 PPP 拨号脚本

将 init.gprs-pppd 和 3gdata_call.conf 预置到 init.rc 所指定的路径下.
并修改权限

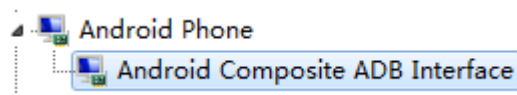
```
adb push init.gprs-pppd /etc/  
adb push 3gdata_call.conf /etc/  
adb shell chmod 777 /etc/init.gprs-pppd
```

如果 android 系统 /system/bin/ 目录下没有 chat 文件, 可以将我们提供的库文件包中的 chat 文件 push 到 /system/bin/ 目录下

```
Adb push chat /system/bin/  
Adb shell chmod 777 /system/bin/chat
```

部分客户遇到的问题点:

- adb 全称叫做 Android Debug Bridge. 主要用于桥接 PC 和 android 设备。
我们一般用 adb 配合 logcat 打印 log、PC 和 android 设备间传送文件。
adb 文件包可以到网络上下载。
Android 设备接到 PC 并安装安装好 adb 驱动后, 设备管理器里面有下面这个设备



此时可以使用 adb push 等命令

关于 adb 网络上可以找到大量详细资料及各种问题的解决方案。

- 部分设备在用 adb push 之前需要调用 adb remount
- 部分客户设备没有 USB 口, 而是通过串口连接数据。此时可用 sd 卡将文件 copy 到相应的位置

d. 部分客户设备无法执行 cp 命令 (read-only), 此时可在设备 shell 目录下执行 `mount -o remount,rw /system`

6. 更新 ril 及库文件

```
adb push ril /system/bin/  
adb shell chmod 777 /system/bin/ril  
adb push libreference-ril.so /system/lib/  
adb push libril.so /system/lib/
```

7. 关于 android5.0 之后的版本信号图标显示惊叹号问题

Android 会通过 captive_portal_detection 方式检测网络。通过构造一个 http 请求发往一个服务器 (默认 <http://connectivitycheck.android.com/>) 由于国内无法访问这个服务器, 所以会有几个惊叹号。

此外还会造成一个问题: android 在没有其他网络数据收发情况下, android 会认为当前网络有问题, 以致重新拨号, 或者重新初始化 ril 包括模块。

解决方法:

方法 1: 在 android 源码

frameworks/base/packages/SettingsProvider/res/values/defaults.xml 中添加一个属性:

```
<bool name="def_captive_portal_detection_enabled">false</bool>
```

此方法会关闭网络检测功能。如果客户希望保留网络检测功能,

方法 2:

在 defaults.xml 中添加一个属性: captive_portal_server

属性值是一个可访问的服务器地址。 或者直接在

NetworkMonitor.java 中直接修改指定服务器。

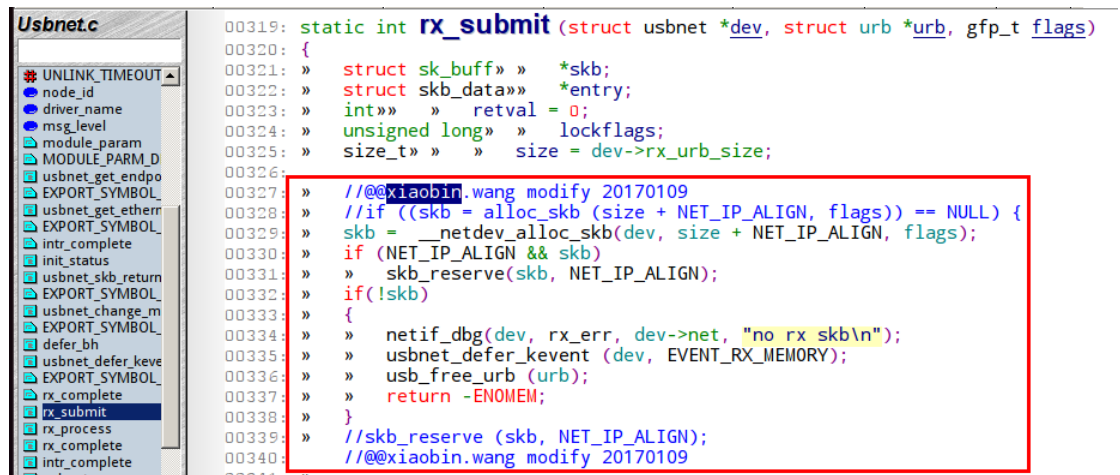
8. GPS 支持

我们提供的 gps 库文件名是 gps.simcom.so, 客户需要根据自己的系统修改一下名称, 一般来说可以进入 /system/lib/hw 查看一下以前的库文件名, 替换一下即可

```
Adb push gps.simcom.so /system/lib/hw/gps.xxxxxx.so
```

9. 关于 kernel3.0 以下版本 IP 获取问题:

将客户 usbnet.c 与我们提供的 usbnet.c 对比, 一般可进入 \kernel\drivers\net\usb\usbnet.c 中查看, 让客户按照图中红圈部分修改。



```
00319: static int rx_submit (struct usbnet *dev, struct urb *urb, gfp_t flags)
00320: {
00321:     struct sk_buff *skb;
00322:     struct skb_data *entry;
00323:     int retval = 0;
00324:     unsigned long lockflags;
00325:     size_t size = dev->rx_urb_size;
00326:
00327:     /*@@xiaobin.wang modify 20170109
00328:     //if ((skb = alloc_skb (size + NET_IP_ALIGN, flags)) == NULL) {
00329:     skb = __netdev_alloc_skb(dev, size + NET_IP_ALIGN, flags);
00330:     if (NET_IP_ALIGN && skb)
00331:         skb_reserve(skb, NET_IP_ALIGN);
00332:     if (!skb)
00333:     {
00334:         netif_dbg(dev, rx_err, dev->net, "no rx skb\n");
00335:         usbnet_defer_kevent (dev, EVENT_RX_MEMORY);
00336:         usb_free_urb (urb);
00337:         return -ENOMEM;
00338:     }
00339:     //skb_reserve (skb, NET_IP_ALIGN);
00340:     /*@@xiaobin.wang modify 20170109
```

10. 抓 ril log

如果移植后发现异常，提供 ril log:

adb logcat -b radio -v time >radio.txt

adb logcat -v time >main.txt

如果设备和 PC 间是串口连接， 进入串口 shell 模式后执行

Logcat -b radio -v time

Logcat -v time

把打印出的 log 存成文件

五 客户遇到的一些问题:

1. apn 没有设置:

如果按照上面所述步骤操作后，出现有网络信号，但是不能成拨号上网，很有可能是 apn 没有设置， apn 的设置使用基本是 android 上的操作，这部分本身是需要客户自行调试他们的系统设置的。但是目前很多客户，尤其是电信卡使用客户都反馈这个问题， 所以整理了一个大致分析解决此问题的流程，请客户参考:

- 首先大致判断一下网络模式， 主要判断一下是否是 CDMA/EVDO 模式
如果不是 7100CE/6320 并且使用电信卡的话，不用考虑 CDMA/EVDO 模式
- 非 CDMA 模式: 进入 android 设置界面下，查看一下是否有 apn， 是否处于激活状态。 如果没有设置或未激活则添加激活一下即可。
- CDMA/EVDO 模式: 此时原生态的 android 设置一般都不显示 apn 设置菜单。
此时可以从 radio log 大致判断一下是否为 apn 未设置 (所有网络模式都可以以此判断)。用 UltraEdit 打开 radio log, 搜索字符串 apn， 把所有带 apn 的行过滤出来，看最后如果仍然显示 null 的话，就是 apn 没有设置好
(目前只有 6320 或 7100CE 使用电信卡的时候可能处在此模式下)

- d. 如果已经确认是 apn 未设置:

此时需要在 android 系统/etc/apns-conf.xml 中修改或添加相应的设置, 再删除数据库文件: /data/data/com.android.providers.telephony/databases/telephony.db 重启 android, 正常情况下 android 会重新设置 apn 菜单。

注意电信卡目前大致有两种(46003 46011) 46011 一般是 4G 卡, 需要根据自己的卡添加, 建议两种都添加。此外, 一定要添加 username 和 password 如果是公网卡, 设置为 card card 即可, 专网卡的话填写运营商提供的用户名密码

- e. 如果已操作 d 步骤, 但是 radio log 仍然显示 null, 那么需要检查一下设置是否成功。Android 管理使用 apn, 并不是直接读取或操作 apns-conf.xml. 而是通过读写数据库(telephony.db). 所以我们可以直接查看数据库是否已经有我们添加修改的东西:

导出 telephony.db 到 PC 上用 SQLite Expert 查看

不导出文件, 直接使用 sqlite3 数据库操作命令查看。

2. 端口属性问题 (ttyUSB* 没有写权限)

部分客户的 android 系统将 ttyUSB* 的权限设置的比较低, 导致 ttyUSB 口无法正常使用。

这个一般在 android 源码的 device 子目录下的某个 rc 文件中。可以直接在 device 目录下执行 grep -rn ttyUSB. 应该可以找到某行显示一个 rc 文件, 并且有显示权限值, 正常的应该是 666。如果不是, 找到相应文件, 把值修改为 666

/dev/spidev2.0	0000	system	system
/dev/ttyUSB*	0666	radio	radio
/dev/ttySAC0	0666	system	system

3. Android 设置添加网络模式设置菜单

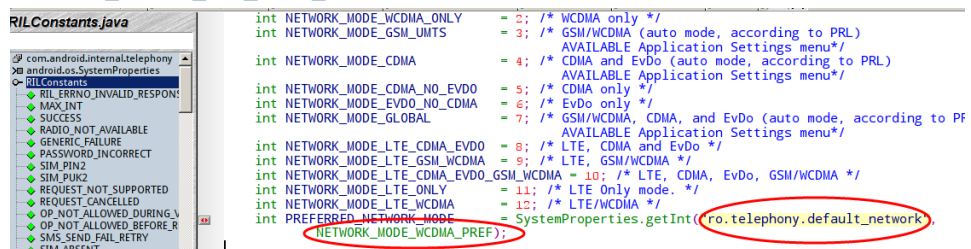
部分客户 android 网络模式设置菜单只有 3G 和 2G 两个菜单项, 没有 4G 的。

- a. 打开 android 源码 packages/services/Telephony/res/values/config.xml, 将其中 config_enabled_lte 的值改为 true。这样设置菜单中就会有 4G 选项出现

- b. 修改 android 默认网络模式为 4G: 打开 android 源码

frameworks/base/telephony/java/com/android/internal/telephony/RILConstants.java

将其中的变量 PREFERRED_NET_MODE 修改为 NETWORK_MODE_LTE_GSM_WCDMA



- c. 网络设置模式设置的 java 源码:

packages/services/Telephony/src/com/android/phone/MobileNetworkSettings.java

客户如果通过 a, b 步骤仍然有问题的, 可以去分析这个代码找一下问题

4. APN 设置相关问题

有些 android 设备会发起彩信的连接(非用户主动, 可能和 android 本身相关) RIL 接到彩信连接之后需要断开正常数据连接, 再重新通过彩信的 apn, 拨号上网。结束后等待 android 重新下发数据网络连接。由于拨号上网需要一定的时间, 所以会造成网络断开一阵的现象。

如果客户并不需要彩信, 邮件的功能, 我们建议去掉彩信, 邮件等 apn 设置很多 android 设备默认预置了彩信, 邮件的 APN。(进入 apn 设置界面, 可以看到好几个选项)。我们建议保留一个即可:

中国移动: 保留 cmnet

中国电信: 保留 3gnet

中国电信: 保留 ctnet

下面的修改方式供参考:

修改前:

```
<apn carrier="China Mobile" mcc="460" mnc="00" apn="cmnet" type="default,supl" />
<apn carrier="China Mobile" mcc="460" mnc="02" apn="cmnet" type="default,supl" />
<apn carrier="中国移动 (China Mobile) GPRS" mcc="460" mnc="07" user="cmnet" password="cmnet"
<apn carrier="China Mobile MMS" mcc="460" mnc="00" apn="cmwap" proxy="10.0.0.172" port="80" m
<apn carrier="China Mobile MMS" mcc="460" mnc="02" apn="cmwap" proxy="10.0.0.172" port="80" m
<apn carrier="中国移动彩信 (China Mobile)" mcc="460" mnc="07" apn="cmwap" proxy="10.0.0.172"
<apn carrier="China Mobile CMWAP" apn="CMWAP" mcc="460" mnc="00" user="wap" password="wap" s
<apn carrier="China Mobile CMWAP" apn="CMWAP" mcc="460" mnc="02" user="wap" password="wap" s
<apn carrier="China Mobile CMWAP" apn="CMWAP" mcc="460" mnc="07" user="wap" password="wap" s
<apn carrier="China Unicom 3G" mcc="460" mnc="01" apn="3gnet" port="80" type="default,supl" /
<apn carrier="中国联通 3g 彩信 (China Unicom)" mcc="460" mnc="01" apn="3gwap" mmsc="http://mm
<apn carrier="China Unicom MMS" mcc="460" mnc="01" apn="uniwap" mmsc="http://mmsc.myuni.com.c
<apn carrier="China Telecom" apn="CTNET" mcc="460" mnc="03" user="ctnet@mycdma.cn" password=
<apn carrier="China Telecom wap" apn="CTWAP" mcc="460" mnc="03" user="ctwap@mycdma.cn" pass
<apn carrier="China Telecom Mms" apn="CTWAP" mcc="460" mnc="03" user="ctwap@mycdma.cn" passw
```

将红色下面部分和红色 X 部分去掉

修改后:

```
<apn carrier="China Mobile_00" mcc="460" mnc="00" apn="cmnet" type="default" />
<apn carrier="China Mobile_02" mcc="460" mnc="02" apn="cmnet" type="default" />
<apn carrier="China Unicom_01" mcc="460" mnc="01" apn="3gnet" type="default" />
<apn carrier="China Unicom_06" mcc="460" mnc="06" apn="3gnet" type="default" />
<apn carrier="China Telecom03" apn="ctnet" mcc="460" mnc="03" user="ctnet@mycdma.cn" password="vne
<apn carrier="China Telecom11" apn="ctnet" mcc="460" mnc="11" user="ctnet@mycdma.cn" password="vne
```