SIMCOM_ANDROID_RIL 适配文档 V1.21

目录

Simcom Android ril 适配 1.19

（在本文中，android7,8,9.）

日期	Histoty	
2016-06-07	First release	
2018-05-21	Version 1.13	
2020-06-04	Version 1.17	

一、SIMCOM 模块 USB 相关描述：

SIM 系列:

7100/SIM7200/SIM7230/SIM7250/7500/A7600/7800 /7906/8200

A 系列:

A7600C/A7600E/A7620/A7906

模块 USB：

VID：0x1E0E，

PID：0x9001,9011

BCD：0x0414,0x0318,0100,…

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 serial	data interface
6	USB serial	Android add debug port

SIM7100/7500/7600/7800 系列 可以支持 NDIS、RNDIS、PPP 方式拨号！但默认的情况下均采用 NDIS 或者 RNDIS 拨号方式；

SIM8200 系列默认支持 QMI 拨号；

A 系列支持 RNDIS 和 PPP 拨号方式；

下面适配的步骤中请依据拨号方式操作！

以 Android7.1为示例:

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、驱动配置

模块的 USB 口连接到 linux/windows 系统后，会虚拟出多个口，用于不同的功能使用。这些设备口用于相同的设备 ID（VID、PID），并且会按照一定的顺序排列，这个顺序由模块的驱动决定；这个顺序也是我们用于查找特定端口的依据。

2.1、串口驱动

Linux 内核已经有了默认的串口驱动，这个驱动是标准的，我们在添加模块的驱动时，只需要在原有的驱动配置文件里面，把我们的设备 ID 添加进去就可以了。

PPP 和 NDIS 拨号都需要以下配置：
CONFIG_USB_SERIAL=y
CONFIG_USB_SERIAL_WWAN=y
CONFIG_USB_SERIAL_OPTION=y

配置预留 NDIS 串口：

文件路径：kernel/drivers/usb/serial/option.c；

在 static const struct usb_device_id option_ids[]数组中添加我们的串口信息，其中包括设备 ID（VID、PID），以及需要跳过的设备。

需要跳过的设备：在同一个设备 ID 下面，可能存在多个设备（同一个物理 USB 口虚拟出多个模拟端口）；设置一个黑名单，可以把不需要配置成串口的模拟口屏蔽掉，这样就不会被映射为串口。

串口配置：kernel/drivers/usb/serial/option.c；

Android 配置中需要个人添加 blacklist 结构（可以只添加自己需要的 PID、VID）：

```
/*SimCom products(vendor_id and product_id)*/
#define SIMCOM_PRODUCT_VID_1E0E         0x1E0E
#define SIMCOM_PRODUCT_PID_9001         0x9001
#define SIMCOM_PRODUCT_PID_9011         0x9011/*SIM7600,A7600*/
#define SIMCOM_PRODUCT_PID_902B         0x902B

#define SIMCOM_PRODUCT_VID_05C6         0x05C6
#define SIMCOM_PRODUCT_PID_9000         0x9000

/*==================SIMCOM CONFIG END============================ */
```

以下为添加 blacklist android7,8,9, 稍微有些变化：
Android7:

```
/*SIMCOM*/
/*==================SIMCOM CONFIG START============================ */
static const struct option_blacklist_info simcom_blacklist_9000 = {
    .reserved = BIT(4) | BIT(5) | BIT(6) | BIT(7),
};

static const struct option_blacklist_info simcom_blacklist_9001 = {
    .reserved = BIT(4) | BIT(5) | BIT(6) | BIT(7),
};

static const struct option_blacklist_info simcom_blacklist_9011 = {
    .reserved = BIT(0) | BIT(1) | BIT(6) | BIT(7),
};

static const struct option_blacklist_info simcom_blacklist_902B = {
    .reserved = BIT(4) | BIT(5) | BIT(6) | BIT(7),
};
/*==================SIMCOM CONFIG END============================ */
```

```
/*============================SIMCOM ADD================================*/
    {USB_DEVICE(SIMCOM_PRODUCT_VID_05C6, SIMCOM_PRODUCT_PID_9000),
        .driver_info = (kernel_ulong_t)&simcom_blacklist_9000 },

    {USB_DEVICE(SIMCOM_PRODUCT_VID_1E0E, SIMCOM_PRODUCT_PID_9001),
        .driver_info = (kernel_ulong_t)&simcom_blacklist_9001 },

    {USB_DEVICE(SIMCOM_PRODUCT_VID_1E0E, SIMCOM_PRODUCT_PID_9011),
        .driver_info = (kernel_ulong_t)&simcom_blacklist_9011 },

    {USB_DEVICE(SIMCOM_PRODUCT_VID_1E0E, SIMCOM_PRODUCT_PID_902B),
        .driver_info = (kernel_ulong_t)&simcom_blacklist_902B },

/*============================SIMCOM ADD================================*/
```

Android8.0 以上版本的 blacklist 稍微有点变化：里面的数值和上面的 blacklist 中对应：

```
/*================================SIMCOM_CONFIG START============================*/
   { USB_DEVICE(SIMCOM_PRODUCT_VID_05C6, SIMCOM_PRODUCT_PID_9000),
     .driver_info = RSVD(4) | RSVD(5) | RSVD(6) | RSVD(7) },
   { USB_DEVICE(SIMCOM_PRODUCT_VID_1E0E, SIMCOM_PRODUCT_PID_9001),
     .driver_info = RSVD(4) | RSVD(5) | RSVD(6) | RSVD(7) },
   { USB_DEVICE(SIMCOM_PRODUCT_VID_1E0E, SIMCOM_PRODUCT_PID_9011),
     .driver_info = RSVD(0) | RSVD(1) | RSVD(6) | RSVD(7) },
   { USB_DEVICE(SIMCOM_PRODUCT_VID_1E0E, SIMCOM_PRODUCT_PID_902B),
     .driver_info = RSVD(4) | RSVD(5) | RSVD(6) | RSVD(7) },
/*================================SIMCOM_CONFIG END============================*/
```

serial->interface->cur_altsetting->desc.bInterfaceNumber 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。在函数 static int option_probe(struct usb_serial *serial,const struct usb_device_id *id)中参照以下内容添加：

```
static struct usb_serial_driver * const serial_drivers[] = {
    &option_1port_device, NULL
};

module_usb_serial_driver(serial_drivers, option_ids);

static int option_probe(struct usb_serial *serial,
            const struct usb_device_id *id)
{
    struct usb_interface_descriptor *iface_desc =
                &serial->interface->cur_altsetting->desc;
    struct usb_device_descriptor *dev_desc = &serial->dev->descriptor;
    const struct option_blacklist_info *blacklist;

/*=========================SIMCOM CONFIG START=======================*/
    if (serial->dev->descriptor.idVendor == SIMCOM_PRODUCT_VID_1E0E  &&
        serial->dev->descriptor.idProduct == SIMCOM_PRODUCT_PID_9001 &&
        serial->interface->cur_altsetting->desc.bInterfaceNumber >= 4 )
            return -ENODEV;

    if (serial->dev->descriptor.idVendor == SIMCOM_PRODUCT_VID_05C6  &&
        serial->dev->descriptor.idProduct == SIMCOM_PRODUCT_PID_9000 &&
        serial->interface->cur_altsetting->desc.bInterfaceNumber >= 4 )
            return -ENODEV;

     if (serial->dev->descriptor.idVendor == SIMCOM_PRODUCT_VID_1E0E  &&
        serial->dev->descriptor.idProduct == SIMCOM_PRODUCT_PID_9011)
    {
        if(serial->interface->cur_altsetting->desc.bInterfaceNumber <= 1 ||
        serial->interface->cur_altsetting->desc.bInterfaceNumber >= 6 )
            return -ENODEV;
    }

    if (serial->dev->descriptor.idVendor == SIMCOM_PRODUCT_VID_1E0E  &&
        serial->dev->descriptor.idProduct == SIMCOM_PRODUCT_PID_902B &&
        serial->interface->cur_altsetting->desc.bInterfaceNumber >= 4 )
            return -ENODEV;

/*=========================SIMCOM CONFIG END=========================*/
```

如果以上信息都正确配置，那么当 android 启动插入设备就可以看到/dev/ttyUSB0~5（旧8200模块为 ttyUSB0和 ttyUSB1）。

设置 USB 权限：system/core/rootdir/ueventd.rc

```
#simcom modify ttyUSB0-4
/dev/tyyUSB*                0666   root       root
```

2.2、网卡驱动

如果是 PPP 拨号方式，直接跳过这一步，进入2.3 配置 telephony；

网卡驱动是用来拨号上网的，这个网络端口同样由 USB 口虚拟得来，需要特定的驱动程序来驱动这个口。

Ndis 拨号需在 android 系统中的/system/build.prop（源码文件可以根据内容自行搜索） 中设置 rild.simcom.ndis=1

2.2.1、 USB Serial 的内核配置支持(android 中大多数默认已经配置好了)

内核3.4.1及以上版本，还需要打开一下三项：

CONFIG_USB_WDM=y

CONFIG_USB_USBNET=y

CONFIG_USB_NET_QMI_WWAN=y

内核3.4.1以下打开一下两项：
CONFIG_USB_WDM=y
CONFIG_USB_USBNET=y

2.2.2、 添加驱动文件

4G 模块驱动文件：simcom_wwan.c
5G 模块驱动文件：qmi_wwan_simcom_1e0e_9001.c

将我们的驱动代码放到 USB 驱动目录 kernel/drivers/net/usb/下。

2.2.3、 添加到编译脚本

“kernel/drivers/net/usb/Makefile”

```
29 obj-$(CONFIG_USB_USBNET)     += usbnet.o qmi_wwan_simcom_1e0e_9001.o
30 obj-$(CONFIG_USB_NET_INT51X1)    += int51x1.o
31 obj-$(CONFIG_USB_CDC_PHONET)     += cdc-phonet.o
32 obj-$(CONFIG_USB_NET_KALMIA)     += kalmia.o
33 obj-$(CONFIG_USB_IPHETH)     += ipheth.o
34 obj-$(CONFIG_USB_SIERRA_NET)     += sierra_net.o
35 obj-$(CONFIG_USB_NET_CX82310_ETH)    += cx82310_eth.o
36 obj-$(CONFIG_USB_NET_CDC_NCM)    += cdc_ncm.o
37 obj-$(CONFIG_USB_NET_HUAWEI_CDC_NCM)     += huawei_cdc_ncm.o
38 obj-$(CONFIG_USB_VL600)      += lg-vl600.o
39 #obj-$(CONFIG_USB_NET_QMI_WWAN) += qmi_wwan.o
```

2.2.4、 添加 cdc-wdm0

在/system/core/init/devices.cpp 中添加设备节点 cdc-wdm0，**一定注意添加位置**。以下为参照实例：

```
            if (uevent->device_name) {
+-----  3 lines: if (!assemble_devpath(devpath, "/dev", uevent->device_name))
                }
                else {
                    /* This imitates the file system that would be created
                     * if we were using devfs instead.
                     * Minors are broken up into groups of 128, starting at "001"
                     */
                    int bus_id = uevent->minor / 128 + 1;
                    int device_id = uevent->minor % 128 + 1;
                    /* build directories */
                    make_dir("/dev/bus", 0755);
                    make_dir("/dev/bus/usb", 0755);
                    snprintf(devpath, sizeof(devpath), "/dev/bus/usb/%03d", bus_id);
                    make_dir(devpath, 0755);
                    snprintf(devpath, sizeof(devpath), "/dev/bus/usb/%03d/%03d", bus_id, device_id);
                }
/*==========================SIMCOM START============================*/
        //add by simcom for mknod /dev/cdc-wdm0
        } else if (!strcmp(uevent->subsystem, "usbmisc") && uevent->device_name) {
            snprintf(devpath, sizeof(devpath), "/dev/%s", uevent->device_name);

/*==========================SIMCOM END============================*/

        } else {
            /* ignore other USB events */
            return;
        }
    } else if (!strncmp(uevent->subsystem, "graphics", 8)) {
        base = "/dev/graphics/";
        make_dir(base, 0755);
    } else if (!strncmp(uevent->subsystem, "drm", 3)) {
        base = "/dev/dri/";
        make_dir(base, 0755);
```

android9以上:

```
void DeviceHandler::HandleDeviceEvent(const Uevent& uevent) {
    if (uevent.action == "add" || uevent.action == "change" || uevent.action == "online") {
        FixupSysPermissions(uevent.path, uevent.subsystem);
    }

    // if it's not a /dev device, nothing to do
    if (uevent.major < 0 || uevent.minor < 0) return;

    std::string devpath;
    std::vector<std::string> links;
    bool block = false;

    if (uevent.subsystem == "block") {
        block = true;
        devpath = "/dev/block/" + Basename(uevent.path);

        if (StartsWith(uevent.path, "/devices")) {
            links = GetBlockDeviceSymlinks(uevent);
        }
    } else if (uevent.subsystem == "usb") {
        if (!uevent.device_name.empty()) {
            devpath = "/dev/" + uevent.device_name;
        } else {
            // This imitates the file system that would be created
            // if we were using devfs instead.
            // Minors are broken up into groups of 128, starting at "001"
            int bus_id = uevent.minor / 128 + 1;
            int device_id = uevent.minor % 128 + 1;
            devpath = StringPrintf("/dev/bus/usb/%03d/%03d", bus_id, device_id);
        }

    } else if (StartsWith(uevent.subsystem, "usb")) {
        // add by sicmom for mknod /dev/cdc-wdm0
        if (uevent.subsystem == "usbmisc" && !uevent.device_name.empty()) {
            devpath = "/dev/" + uevent.device_name;
        } else {
            // ignore other USB events
            return;
        }

    } else if (const auto subsystem =
                   std::find(subsystems_.cbegin(), subsystems_.cend(), uevent.subsystem);
               subsystem != subsystems_.cend()) {
        devpath = subsystem->ParseDevPath(uevent);
    } else {
        devpath = "/dev/" + Basename(uevent.path);
    }

    mkdir_recursive(Dirname(devpath), 0755);

    HandleDevice(uevent.action, devpath, block, uevent.major, uevent.minor, links);
}
```

```
// add by simcom for mknod /dev/cdc-wdm0
    } else if (uevent.subsystem == "usbmisc" && !uevent.device_name.empty()) {
        devpath = "/dev/" + uevent.device_name;
    } else {
        // ignore other USB events
        return;
    }
```

2.3、配置 telephony 的编译

Telephony 涉及的就是打电话、短信等一些列的功能，这部分功能默认是没有编译的。

1. 修改 telephony 编译脚本，加入短信模块的编译

build/target/product/telephony.mk

```
PRODUCT_PACKAGES := \
    CarrierConfig \
    Dialer \
    CallLogBackup \
    CellBroadcastReceiver \
    EmergencyInfo \
    messaging \
    rild

PRODUCT_COPY_FILES := \
```

2. 将 telephony 编译脚本加入到主编译环境中

device/rockchip/rk3399/product.mk (Android7)

```
#
PRODUCT_RUNTIMES := runtime_libart_default

$(call inherit-product, device/rockchip/rk3399/device.mk)

$(call inherit-product, $(SRC_TARGET_DIR)/product/core_64_bit.mk)

#============================== SIMCOM START==============================
$(call inherit-product, $(SRC_TARGET_DIR)/product/telephony.mk)
#============================== SIMCOM END==============================
```

3. device/rockchip/rk3399/rk3399.mk(android8)

```
#============================== SIMCOM START==============================
$(call inherit-product, $(SRC_TARGET_DIR)product/telephony.mk)
$(call inherit-product, $(SRC_TARGET_DIR)product/full_base.mk)
#============================== SIMCOM END==============================
include device/rockchip/rk3399/BoardConfig.mk
```

4. device/rockchip/rk3399/ rk3399_mid.mk(android8)

```
#============================== SIMCOM START==============================
$(call inherit-product, $(SRC_TARGET_DIR)/product/full_base.mk)
$(call inherit-product, $(SRC_TARGET_DIR)/product/telephony.mk)
#============================== SIMCOM START==============================
```

5. device/rockchip/rk3399/nanopc-t4/overlay/frameworks/base/core/res/res/values/config.xml

(**nanopc-t4:**每个公司起名不一样。)

```
    <bool name="config_avoidGfxAccel">true</bool>

    <!-- When true use the linux /dev/input/event subsystem to detect the switch changes
         on the headphone/microphone jack. When false use the older uevent framework. -->
    <bool name="config_useDevInputEventForAudioJack">true</bool>

    <!-- Whether safe headphone volume is enabled or not (country specific). -->
    <bool name="config_safe_media_volume_enabled">true</bool>

    <!-- Boolean indicating if current platform supports BLE peripheral mode -->
    <bool name="config_bluetooth_le_peripheral_mode_supported">true</bool>

    <bool name="config_voice_capable">true</bool>
```

6. device/rockchip/rk3399/nanopc_t4.mk 功能：复制相应的文件到 vendor/simcom

（nanopc-t4）

```
#================================SIMCOM CONFIG START=========================
$(call inherit-product, $(SRC_TARGET_DIR)/product/core.mk)
include device/rockchip/rk3399/nanopc-t4/BoardConfig.mk

# Quectel
#$(call inherit-product-if-exists, vendor/quectel/ec20/device-partial.mk)
$(call inherit-product, $(SRC_TARGET_DIR)/product/telephony.mk)
#================================SIMCOM CONFIG END===========================
```

```
#================================SIMCOM CONFIG START=========================
PRODUCT_COPY_FILES += vendor/simcom/rild:vendor/bin/hw/rild
PRODUCT_COPY_FILES += vendor/simcom/libril.so:vendor/lib64/libril.so
PRODUCT_COPY_FILES += vendor/simcom/libreference-ril.so:vendor/lib64/libreference.so

PRODUCT_PROPERTY_OVERRIDES += \
    persist.demo.hdmirotates=true \
    rild.libpath = /vendor/lib64/libreference-ril.so

#================================SIMCOM CONFIG END===========================
```

2.4、配置 Android 功能（电话、短信功能）

device/rockchip/common/overlay/frameworks/base/core/res/res/values/config.xml

```
@@ -20,7 +20,7 @@
 <resources>

     <!-- This device is not "voice capable"; it's data-only. -->
-    <bool name="config_voice_capable">false</bool>
+    <bool name="config_voice_capable">true</bool>

     <!-- Flag indicating whether we should enable the automatic brightness
in Settings.
          config_hardware_automatic_brightness_available is not set, so we w
ill use software implementation -->
@@ -164,7 +164,7 @@
     <bool name="config_enableMultiUserUI">true</bool>

     <!-- This device allows sms service. -->
-    <bool name="config_sms_capable">false</bool>
+    <bool name="config_sms_capable">true</bool>

     <bool name="config_ui_enableFadingMarquee">true</bool>
```

2.5、配置 SElinux 权限（ppp）

Android7以及以下版本的 selinux 添加:

如果使用 ndis 和 rndis 可以不做 selinux；

2.5.1、 设置 system/sepolicy/file_contexts

```
@@ -218,6 +218,7 @@
 /system/bin/idmap u:object_r:idmap_exec:s0
 /system/bin/update_engine        u:object_r:update_engine_exec:s0
 /system/bin/bspatch              u:object_r:update_engine_exec:s0
+/system/etc/init.gprs-pppd       u:object_r:pppd_gprs_exec:s0
```

2.5.2、 设置 system/sepolicy/property.te

```
@@ -19,6 +19,7 @@ type ctl_dumpstate_prop, property_type;
 type ctl_fuse_prop, property_type;
 type ctl_mdnsd_prop, property_type;
 type ctl_rildaemon_prop, property_type;
+type ctl_pppd_gprs_prop, property_type;
 type ctl_bugreport_prop, property_type;
 type ctl_console_prop, property_type;
 type audio_prop, property_type, core_property_type;
```

2.5.3、 设置 system/sepolicy/property_contexts

```
@@ -81,6 +81,7 @@ ctl.dumpstate                u:object_r:ctl_dumpstate_prop:s0
 ctl.fuse_                   u:object_r:ctl_fuse_prop:s0
 ctl.mdnsd                   u:object_r:ctl_mdnsd_prop:s0
 ctl.ril-daemon              u:object_r:ctl_rildaemon_prop:s0
+ctl.pppd_gprs               u:object_r:ctl_pppd_gprs_prop:s0
 ctl.bugreport               u:object_r:ctl_bugreport_prop:s0
 ctl.console                 u:object_r:ctl_console_prop:s0
 ctl.                        u:object_r:ctl_default_prop:s0
```

2.5.4、 设置 system/sepolicy/rild.te

(若是新的进程则在对应的*.te 中添加)

```
@@ -30,6 +30,7 @@ set_prop(rild, net_radio_prop)
 set_prop(rild, system_radio_prop)
 auditallow rild net_radio_prop:property_service set;
 auditallow rild system_radio_prop:property_service set;
+allow rild ctl_pppd_gprs_prop:property_service set;

 # Read/Write to uart driver (for GPS)
 allow rild gps_device:chr_file rw_file_perms;
```

2.5.5、 添加一个新文件 *.te

添加文件：system/sepolicy/pppd_gprs.te

```
# pppd_gprs
type pppd_gprs, domain, domain_deprecated;
type pppd_gprs_exec, exec_type, file_type;

init_daemon_domain(pppd_gprs)
net_domain(pppd_gprs)

# property service
set_prop(pppd_gprs, radio_prop)
set_prop(pppd_gprs, net_radio_prop)
set_prop(pppd_gprs, system_radio_prop)
```

Andorid8及以上 selinux 设置

如果使用 ndis 和 rndis 可以不做 selinux；

2.5.6、设置 system/sepolicy/private/file_contexts 和 system/sepolicy/prebuilts/api/*/private/file_context。

（或者只添加在 system/sepolicy/vendor/file_contexts。切记不能和以上两个路径同时添加，否则会报错。）

```
@@ -218,6 +218,7 @@
 /system/bin/idmap u:object_r:idmap_exec:s0
 /system/bin/update_engine        u:object_r:update_engine_exec:s0
 /system/bin/bspatch              u:object_r:update_engine_exec:s0
+/system/etc/init.gprs-pppd       u:object_r:pppd_gprs_exec:s0
```

2.5.7、设置 system/sepolicy/private/property_contexts 和 system/sepolicy/prebuilts/api/*/private/property_contexts

```
@@ -81,6 +81,7 @@ ctl.dumpstate                u:object_r:ctl_dumpstate_prop:s0
 ctl.fuse_               u:object_r:ctl_fuse_prop:s0
 ctl.mdnsd               u:object_r:ctl_mdnsd_prop:s0
 ctl.ril-daemon          u:object_r:ctl_rildaemon_prop:s0
+ctl.pppd_gprs           u:object_r:ctl_pppd_gprs_prop:s0
 ctl.bugreport           u:object_r:ctl_bugreport_prop:s0
 ctl.console             u:object_r:ctl_console_prop:s0
 ctl.                    u:object_r:ctl_default_prop:s0
```

2.5.8、设置 system/sepolicy/public/property.te 和 system/sepolicy/prebuilts/api/*/public/property.te

```
@@ -19,6 +19,7 @@ type ctl_dumpstate_prop, property_type;
 type ctl_fuse_prop, property_type;
 type ctl_mdnsd_prop, property_type;
 type ctl_rildaemon_prop, property_type;
+type ctl_pppd_gprs_prop, property_type;
 type ctl_bugreport_prop, property_type;
 type ctl_console_prop, property_type;
 type audio_prop, property_type, core_property_type;
```

2.5.9、设置 system/sepolicy/public/rild.te 和 system/sepolicy/prebuilts/api/*/public/rild.te

(若是新的进程则在对应的*.te 中添加)

```
@@ -30,6 +30,7 @@ set_prop(rild, net_radio_prop)
 set_prop(rild, system_radio_prop)
 auditallow rild net_radio_prop:property_service set;
 auditallow rild system_radio_prop:property_service set;
+allow rild ctl_pppd_gprs_prop:property_service set;

 # Read/Write to uart driver (for GPS)
 allow rild gps_device:chr_file rw_file_perms;
```

2.5.10、设置 system/sepolicy/public/domain.te 和 system/sepolicy/prebuilts/api/*/public/domain.te

（具体位置根据版本不同仔细查看，或者根据编译报错提示位置添加。）

```
diff --git a/public/domain.te b/public/domain.te
index f5c72cc..ed86524 100644
--- a/public/domain.te
+++ b/public/domain.te
@@ -724,6 +724,7 @@ full_treble_only(`
         -coredomain
         -appdomain
         -rild
+        -pppd_gprs
         -vendor_executes_system_violators
     } {
         exec_type
```

2.5.11、添加一个新文件 pppd_gprs.te

设置 system/sepolicy/public/pppd_gprs.te 和 system/sepolicy/prebuilts/api/*/public/pppd_gprs.te

```
# type_transition must be private policy the domain_trans rules could stay
# public, but conceptually should go with this
type pppd_gprs, domain, coredomain;

net_domain(pppd_gprs)

set_prop(pppd_gprs, radio_prop)
set_prop(pppd_gprs, net_radio_prop)
set_prop(pppd_gprs, system_radio_prop)
set_prop(pppd_gprs, system_file)
```

2.5.12、添加一个新文件 pppd_gprs.te

设置 system/sepolicy/vendor/pppd_gprs.te

2.6、脚本配置

2.6.1、ppp 拨号配置

添加拨号文件：

复制 init.gprs-pppd 和3gdata_call.conf 脚本到 andorid 系统的/system/etc/目录。

8200模块：

Adb push 8200option /etc/ppp/peers/；

Adb push 8200-chat /etc/ppp/chat/；

如果路径不存在，则需要自己创建(拨号命令 sudo pppd call 8200option)。

把对应文件的权限修改为777：

Adb shell chmod 777 /system/etc/init.gprs-pppd

Adb shell chmod 777 /system/etc/3gdata_call.conf

8200模块：

Adb shell chmod -R 777 /system/etc/chat

Adb shell chmod -R 777 /system/etc/chat

2.6.2、启动脚本

启动脚本的主要作用就是使指定程序可以开机自动启动，我们需要在其中添加 rild 和 ppp 拨号的服务。

源码位置：/system/core/rootdir/init.rc（主） device/vendor/**/init.**.rc（厂商相关，编译时会被主脚本调用合并）

注意：

启动脚本需要编译到 boot 里面，所以不能直接 push 到系统里面，否则不能生效。其中编译之后在系统根目录可以找到这个脚本，可以在这个位置检查脚本是否编译成功。

在源码位置：/system/core/rootdir/init.rc 和/hardware/ril/rild.rc 开机脚本中添加以下内容，-l 选项为 libreference-ril.so 库的位置, 源码 init.rc 中如果已经自带了一部分上面配置，按照下面截图修改。不要新增。（不能同时出现两个 service ril-daemon 或 service pppd_gprs）。

添加脚本权限：

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

8200模块（需要再添加下面两行）：

```
    chmod 0777 -R /system/etc/ppp/chat
    chmod 0777 -R /system/etc/ppp/peers
```

添加开机启动，开机启动的 ril 进程和库的路径一定要和 push 时的文件路径对应，如果需要 GPS，请打开 GPS 那一行；如果不是 ppp 拨号，则可以不添加 service pppd_gprs…. 相关的脚本语句。

```
#================================SIMCOM START=====================================
service ril-daemon /vendor/bin/hw/rild -l /vendor/lib64/libreference-ril.so
    class main
    socket rild stream 660 root radio
    socket sap_uim_socket1 stream 660 bluetooth bluetooth
    socket rild-debug stream 660 radio system
    user root
    group radio cache inet misc audio sdcard_rw log

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

#================================SIMCOM end=======================================
```

Android6.0(与其他不同)

```
#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
```

2.7、使 Android 默认4G 优先

2.7.1、检查 Android 下发的请求：

REQUEST_SET_PREFERRED_NETWORK_TYPE 请求的值如果不是9或者其他功能更适当的值（Android10为：26），则默认不是请求的4G 或者5G 网络，需要修改 Android 源码，配置默认为4G 优先。

a.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。
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 或者5G 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_NETWORK_MODE 修改为
4G：NETWORK_MODE_LTE_GSM_WCDMA。
5G：NETWORK_MODE_NR_LTE_GSM_WCDMA（android10及以上）。

```
int PREFERRED_NETWORK_MODE      = SystemProperties.getInt("ro.telephony.default_network",
        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。

2.7.2、检查模块的配置

AT+CNMP?，返回的值不是2（自动），则需要重新配置模块的参数。

2.8、APN 配置

APN 是用于拨号时的必要信息。Android 默认预置了部分 APN，但是仍然有部分 APN 是没有预置的，因此可能会导致某些 SIM 卡不能拨号的问题，此时就需要添加对应的 APN。
添加方式：

2.8.1、 UI 界面添加 APN：

在 Android 的 setting 里面对应的 SIM 卡配置里面可以找到 APN 的配置。

2.8.2、 配置文件添加 APN：

路径：

/etc/apns-conf.xml（系统）

vendor/rockchip/common/phone/etc/apns-full-conf.xml（源码）

修改系统中的配置文件后，必须删除掉对应的 databass，然后重启才会生效。因为这个 databass 只会在不存在的时候，才会从配置文件中加载数据，并生成 databass，如果已经存在，就不会再加载配置文件中的数据了。或者在 Android 的 APN 配置界面选择恢复默认值，也会更新 databass。

Databass 路径：

/data/data/com.android.providers.telephony/databases/telephony.db

2.8.3、 编译环境确保可以正确编译到 APN 的配置文件：

PRODUCT_COPY_FILES += vendor/rockchip/common/phone/etc/apns-full-conf.xml:system/etc/apns-conf.xml。

2.8.4、 APN 问题相对较多，可以跳到 “客户问题”中关于 APN 遇到的一些问题。

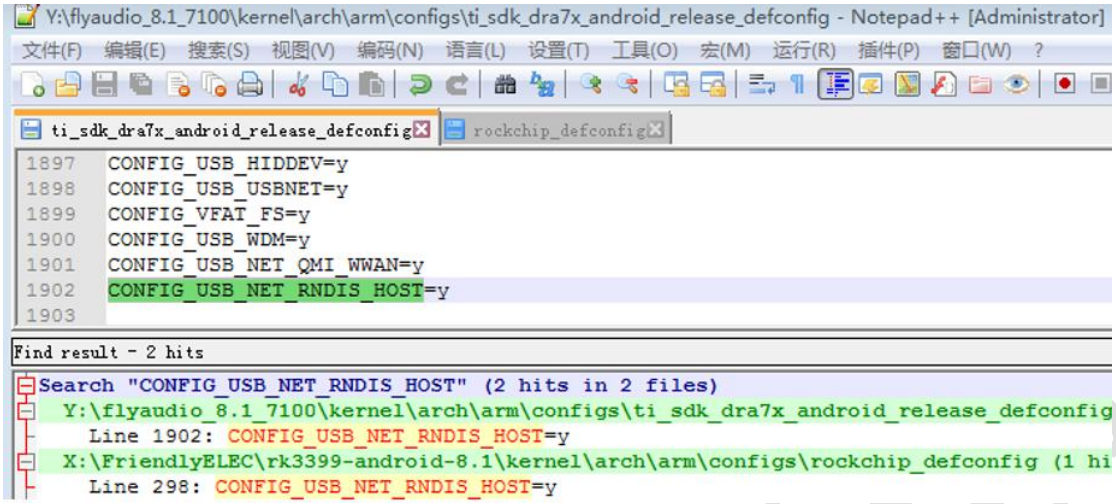
2.9、Android 系统的配置 RNDIS

RNDIS: rild.simcom.rndis=1

2.9.1、问题描述

使用中发现不能发起 RNDIS 的拨号。（PID=9011模式下面）

2.9.2、问题分析



这个 deconfig 的文件一般在 kernel 里面，需要放到 CONFIG_USB_USBNET=1的后面。

2.10、编译的配置选项

2.10.1、Ril 初始化后与 Android 没有交互

(1) 查看是否缺 socket rild
adb 进到 android dev/socket 下查看需要有 rild、rild-debug 等

(2) telephony 的有关信息没有编译到
adb 进到 system/priv-app 中查看需要有 Dialer、TelephonyProvider、Telecom、TeleServiceDialer、TelephonyProvider 在 build/target/product/telephony.mk 中编译，Telecom、TeleService 在 build/target/product/core.mk 中编译

Dialer、TelephonyProvider、Telecom、TeleSevice 这4个是之前跟 Android 交互一定要用到的

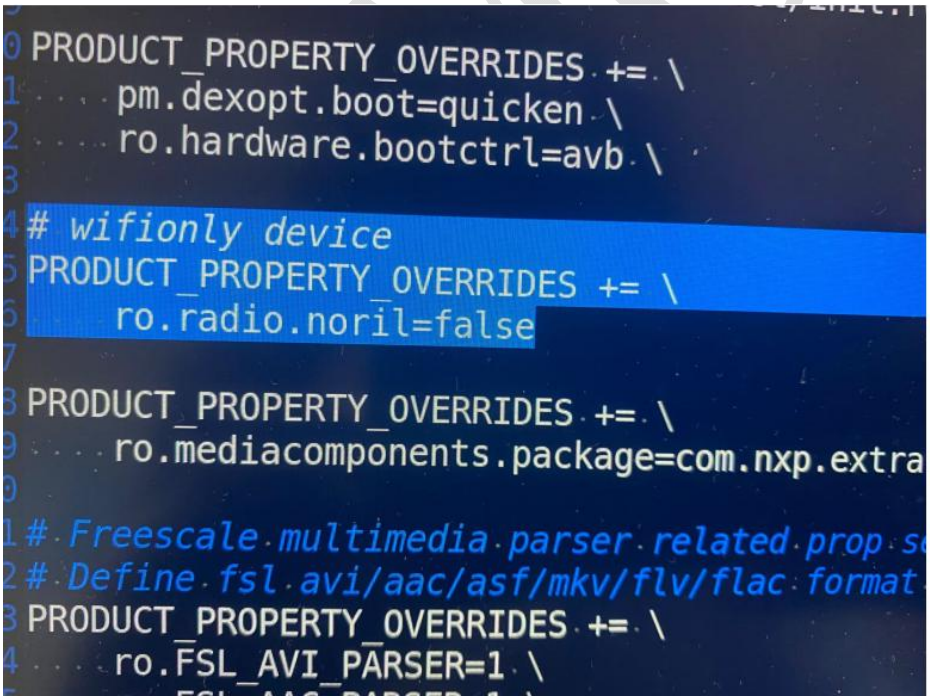
2.11、config 的配置文件修改

修改这个路径下的配置文件：
device/rockchip/rk3399/nanopc-t4/overlay/frameworks/base/core/res/res/values

修改文件内容如下：

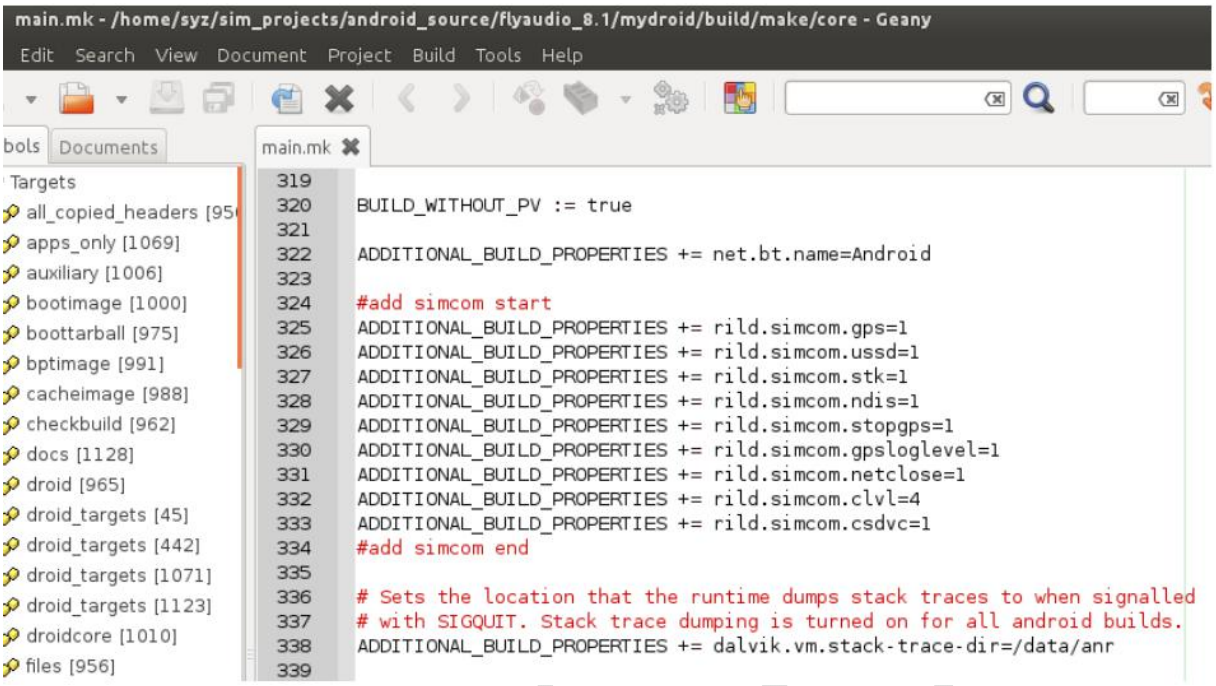
```
<!-- This string array should be overridden by the device to present a list of network
       attributes.  This is used by the connectivity manager to decide which networks can coexist
       based on the hardware -->
  <!-- An Array of "[Connection name],[ConnectivityManager connection type],
       [associated radio-type],[priority],[restoral-timer(ms)],[dependencyMet]  -->
  <!-- the 5th element "resore-time" indicates the number of milliseconds to delay
       before automatically restore the default connection.  Set -1 if the connection
       does not require auto-restore. -->
  <!-- the 6th element indicates boot-time dependency-met value. -->
  <string-array translatable="false" name="networkAttributes">
      <item>"wifi,1,1,1,-1,true"</item>
      <item>"mobile,0,0,0,-1,true"</item>
      <item>"bluetooth,7,7,1,-1,true"</item>
      <item>"ethernet,9,9,0,-1,true"</item>
  </string-array>
```

2.12、解决 WIFI ONLY 的问题



2.13 属性的设置文件

在代码段修改 build.prop 文件



2.14 slot1的默认配置修改

04-22 14:20:18.194 W/hwservicemanager(2939): getTransport: Cannot find entry android.hardware.radio@1.0::IRadio/slot1 In either framework or device manifest

hwservicemanage 出问题了
需要查：system/manifest.xml 和 vendor/manifest.xml

```
165    <hal format="hidl">
166        <name>android.hardware.radio</name>
167        <transport>hwbinder</transport>
168        <version>1.0</version>
169        <interface>
170            <name>IRadio</name>
171            <instance>slot1</instance>
172        </interface>
173    </hal>
174    <hal format="hidl">
175        <name>android.hardware.radio.deprecated</name>
176        <transport>hwbinder</transport>
177        <version>1.0</version>
178        <interface>
179            <name>IOemHook</name>
180            <instance>slot1</instance>
181        </interface>
```

```
147    <hal format="hidl">
148        <name>android.hardware.radio</name>
149        <transport>hwbinder</transport>
150        <version>1.0</version>
151        <interface>
152            <name>IRadio</name>
153            <instance>default</instance>
154        </interface>
155    </hal>
156    <hal format="hidl">
157        <name>android.hardware.radio.deprecated</name>
158        <transport>hwbinder</transport>
159        <version>1.0</version>
160        <interface>
161            <name>IOemHook</name>
162            <instance>default</instance>
163        </interface>
```

三、GPS

GPS: rild.simcom.gps=1

1:启用 GPS, 0 或不设置:不启用;

STOPGPS: rild.simcom.stopgps=1

Android 屏幕关闭的时候 GPS 关闭;

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

四、抓 log

手动抓 log 保存到文件：

RIL 的 log：

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

GPS/pppd/CM：

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、RIL 库应用

5.1 ril 库的应用

解压 simcom_rilXX_XXXXXXXX.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)

qmi_wwan_simcom_1e0e_9001.c(SIM8200(5G) 拨号用)

gps.simcom.so (GPS 库)

chat （ppp 拨号用）

8200option （8200以上模块）放到/etc/ppp/peers 目录

8200-chat （8200以上模块）放到/etc/ppp/chat 目录

（拨号命令 sudo pppd call 8200option;）

把 ril 库的相关文件推送到 android 对应的目录（添加开机脚本时一定要对应路径）

若是手动 push 到系统，首先要重新挂载设备：

adb root && adb remount （低版本 android 或者 mount -o remount,rw /system）

然后 push 文件：

adb push ./rild /vendor/bin/hw/

adb shell chmod 777 /vendor/bin/rild

adb push ./libril.so /vendor/bin/hw/

adb push ./libreference-ril.so /vendor/bin/hw/

这些库也可以直接添加到源码生成文件中，直接生成*. img 包。

5.2、功能配置

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.csdvc=*

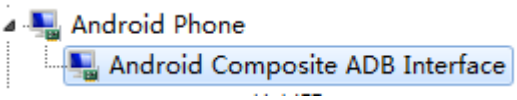
部分客户需要通过 ril 设置 CSDVC 的，可以通过设置此属性， ril 会在 Sim 卡 ready 后将此值通过 AT+CSDVC=* 设置到模块设置成功后，开机通过 adb shell getprop 可以查看到这些属性（客户调试的时候可以直接修改/system/build.prop 这个文件，这样可以不需要重新编译系统， 灵活调整开关）

Android 每个版本的 ril 库都有部分差异，没有同意的 ril 库版本，需要上述功能的话，需要单独另外提供。

六、客户遇到的问题

6.1、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。

6.2、 部分设备在用 adb push 之前需要调用 adb root && adb remount;

6.3、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 相应的位置;

6.4、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;

6.5、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的情况下，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>

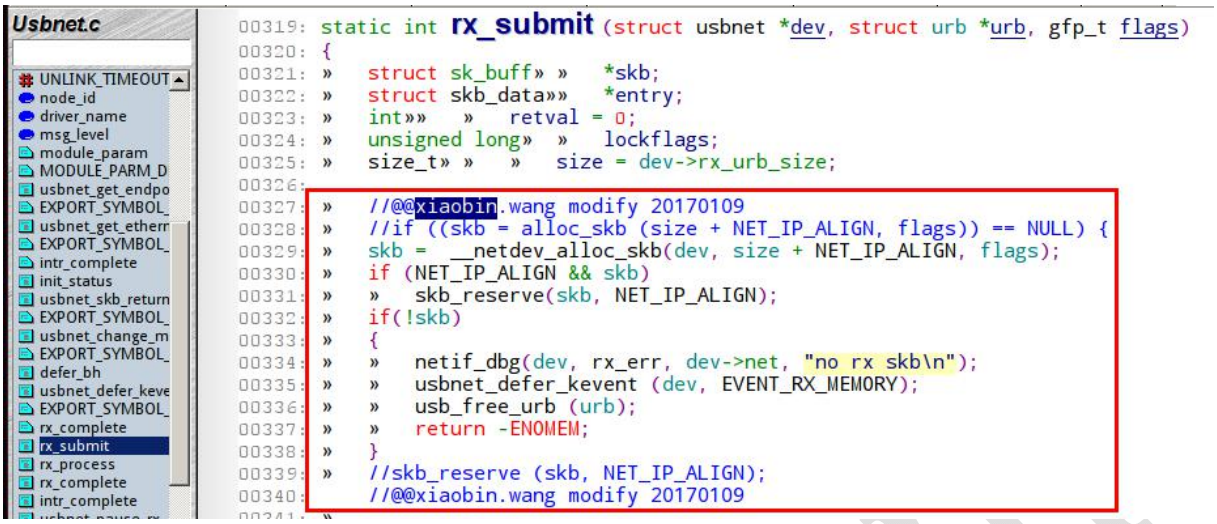
此方法会关闭网络检测功能。如果客户希望保留网络检测功能;

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。

6.6、关于 kernel3.0 以下版本 IP 获取问题：

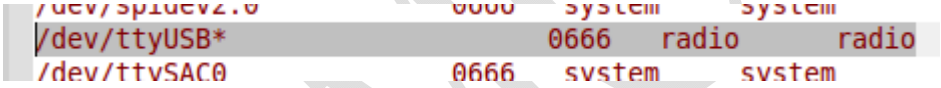
将客户 usbnet.c 与我们提供的 usbnet.c 对比，一般可进入\kernel\derivers\net\usb\usbnet.c 中查看，让客户按照图中红圈部分修改。



6.7、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。



6.8、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：

a.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。

b.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。

c.CDMA/EVDO 模式：此时原生态的 android 设置一般都是不显示 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。

6.9 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
```