Realtek Wi-Fi SDK for Android JB 4.1
ver. 1.3.0

Contents

Release History

0.0.0 beta	2012/08/22	1. Support STA+P2P concurrent and SoftAP mode for Android 4.1 (JB)
1.0.0	2012/11/30	1. First normal release
1.1.0	2013/01/17	1. Update realtek_wifi_SDK_for_android_JB_4.1_20130117.tar.gz 1.1. Modify patch for WifiP2pService.java 2. Update wpa_supplicant_8_jb_4.1_rtw_r6358_20130117.tar.gz
1.2.0	2013/02/06	1. Update realtek_wifi_SDK_for_android_JB_4.1_20130206.tar.gz 1.1. Modify wifi_realtek.c 1.2. Modify patch for WifiP2pService.java
1.3.0		1. Update realtek_wifi_SDK_for_android_JB_4.1_20130306.tar.gz 1.1. Modify patch for WifiP2pService.java 1.2. Add patch for WifiMonitor.java 2. Update wpa_supplicant_8_jb_4.1_rtw_r6890.20130306.tar.gz
1.4.0	2013/03/27	1. Update wpa_supplicant_8_jb_4.1_rtw_r6986.20130327.tar.gz

SDK packages

- hardware/realtek/*

Folder to store config file, private code from Realtek.

- hardware/libhardware_legacy/wifi/Android.mk
  Reference codes for applying wifi_realtek.c

- frameworks/base/wifi/*
  Reference codes for service of Wi-Fi

※ For wpa_supplicant_8_jb_4.1_rtw_r6986.20130327.tar.gz or newer version, see wpa_supplicant_hostapd folder of our SW release package or consult our contact window.

Introduction

This document provides a simple guide to help engineers to apply Realtek Wi-Fi solution onto their Android 4.1 (JB) system. For now, we have supported the following two scenarios:

- **STA/AP** – Switch between STA mode and AP mode
- **(STA+P2P)/AP** – Switch between STA+P2P(Wi-Fi Direct) concurrent mode and AP mode

To port Realtek Wi-Fi driver onto Android 4.1 platform, you can go through the following guide with reference codes within our driver package's realtek_wifi_SDK_for_android_JB_4.1_20130306.tar.gz.

Because Android's SDK may differ from platform to platform, our reference codes may not be applied on every platform without modifications. You should check if our reference code is suitable for you to use.

1. Copy Necessary Files into SDK

You need to copy the following folder into your target Android SDK folder:

- hardware/realtek/

2. Platform Related Files

2.1. BoardConfig.mk

To apply Realtek Wi-Fi solution onto your Android JB system, define the following compile-time variables in BoardConfig.mk of your platform:

```
BOARD_WIFI_VENDOR := realtek
ifeq ($(BOARD_WIFI_VENDOR), realtek)
    WPA_SUPPLICANT_VERSION := VER_0_8_X
    BOARD_WPA_SUPPLICANT_DRIVER := NL80211
    CONFIG_DRIVER_WEXT :=y
    BOARD_WPA_SUPPLICANT_PRIVATE_LIB := lib_driver_cmd_rtl
    BOARD_HOSTAPD_DRIVER            := NL80211
    BOARD_HOSTAPD_PRIVATE_LIB       := lib_driver_cmd_rtl

    BOARD_WLAN_DEVICE := rtl8192cu
    #BOARD_WLAN_DEVICE := rtl8192du
    #BOARD_WLAN_DEVICE := rtl8192ce
    #BOARD_WLAN_DEVICE := rtl8192de
    #BOARD_WLAN_DEVICE := rtl8723as
    #BOARD_WLAN_DEVICE := rtl8723au
    #BOARD_WLAN_DEVICE := rtl8189es

    WIFI_DRIVER_MODULE_NAME     := "wlan"
    WIFI_DRIVER_MODULE_PATH     := "/system/lib/modules/wlan.ko"
    WIFI_DRIVER_MODULE_ARG      := "ifname=wlan0 if2name=p2p0"

    WIFI_FIRMWARE_LOADER        := ""
    WIFI_DRIVER_FW_PATH_STA     := ""
    WIFI_DRIVER_FW_PATH_AP      := ""
    WIFI_DRIVER_FW_PATH_P2P     := ""
    WIFI_DRIVER_FW_PATH_PARAM := ""
endif
```

- **BOARD_WIFI_VENDOR := realtek**

To distinguish the platform Wi-Fi device from products of other companies, we define variable BOARD_WIFI_VENDOR as realtek. This is for compile-time choices to be applied for Realtek Wi-Fi solutions.

- **WPA_SUPPLICANT_VERSION := VER_0_8_X**

For Android JB, please set WPA_SUPPLICANT_VERSION as VER_0_8_X to use wpa_supplicant_8.

- **BOARD_WPA_SUPPLICANT_DRIVER := NL80211**

- **BOARD_WPA_SUPPLICANT_PRIVATE_LIB := lib_driver_cmd_ rtl**
- **BOARD_HOSTAPD_DRIVER := NL80211**
- **BOARD_HOSTAPD_PRIVATE_LIB := lib_driver_cmd_rtl**

We use NL80211 as the driver interface for wpa_supplicant and hostapd to communicate with driver and provide lib_driver_cmd_rtl as the private processing library.

- **BOARD_WLAN_DEVICE**

Realtek provide a variety of Wi-Fi solutions to choose. For now, BOARD_WLAN_DEVICE is not used for any purpose but we suggest setting this variable for your Wi-Fi solution you used.

- **WIFI_DRIVER_MODULE_NAME**
- **WIFI_DRIVER_MODULE_PATH**
- **WIFI_DRIVER_MODULE_ARG**

These three variables will be used in libhardware_legacy (wifi.c/wifi_realtek.c) to do insmod and remmod. The value of WIFI_DRIVER_MODULE_NAME should match the value of MODULE_NAME specified in our driver's Makefile at compile-time. Please refer to "Platform Setting Section in Detail" of:

document/Quick_Start_Guide_for_Driver_Compilation_and_Installation.pdf

- **WIFI_FIRMWARE_LOADER :=""**
- **WIFI_DRIVER_FW_PATH_STA :=""**
- **WIFI_DRIVER_FW_PATH_AP :=""**
- **WIFI_DRIVER_FW_PATH_P2P :=""**
- **WIFI_DRIVER_FW_PATH_PARAM :=""**

Because our driver has FW embedded inside, and will automatically load FW at NIC initialization process, there is no need to set these 5 variables, just keep them empty.

2.2. init.xxx.rc

For Wi-Fi to operate properly, we need some daemons to be defined as service inside init.xxx.rc. Please refer to the service definitions below:

- **wpa_supplicant**

```
service rtw_suppl_con /system/bin/wpa_supplicant \
    -ip2p0 -Dnl80211 -c /data/misc/wifi/p2p_supplicant.conf -e/data/misc/wifi/entropy.bin -N \
    -iwlan0 -Dnl80211 -c/data/misc/wifi/wpa_supplicant.conf
    class main
    socket wpa_wlan0 dgram 660 wifi wifi
    disabled
    oneshot

service rtw_suppl /system/bin/wpa_supplicant -iwlan0 -Dnl80211
-c/data/misc/wifi/wpa_supplicant.conf
    socket wpa_wlan0 dgram 660 wifi wifi
    class main
    disabled
    oneshot
```

l dhcpcd

```
service dhcpcd_wlan0 /system/bin/dhcpcd -aABKL
    class main
    disabled
    oneshot

service dhcpcd_p2p /system/bin/dhcpcd -aABKL
    class main
    disabled
    oneshot

service iprenew_wlan0 /system/bin/dhcpcd -n
    class main
    disabled
    oneshot

service iprenew_p2p /system/bin/dhcpcd -n
    class main
    disabled
    oneshot
```

2.3. Others

l Set wifi.interface

To specify the wifi interface name in Android, a system property named "wifi.interface" is used. For Realtek wifi driver, wifi interface name is assigned with "wlan%d". In general, you should set wifi.interface as "wlan0". For example:

```
PRODUCT_PROPERTY_OVERRIDES += \
    wifi.interface=wlan0
```

l Add android.hardware.wifi.direct.xml

If you want to use Wi-Fi Direct (P2P) functionality, please add the rule in the PRODUCT_COPY_FILES variable for your device platform related file to copy the permission definition file of Wi-Fi Direct to the system/etc/permissions/ folder of your system image. For example:

```
PRODUCT_COPY_FILES += \
frameworks/native/data/etc/android.hardware.wifi.direct.xml:system/etc/permissions/android.hard
ware.wifi.direct.xml
```

With this action, the Wi-Fi Direct UI and the related service will be enabled for your system.

When you enable this, make sure your driver is configured for STA+P2P concurrent mode or you may encounter error when you open the Wi-Fi. Please refer to **"7. Driver Configurations for Android JB"**

3. System Resource Configurations

We should set the following three resource configurations of your platform to configure the network function and enable the corresponding UI interface. In general you can set the following configurations in your platform dependent config.xml file. Take panda board for example:

device/ti/panda/overlay/frameworks/base/core/res/res/values/config.xml

Or the global config.xml file:

frameworks/base/core/res/res/values/config.xml

l networkAttributes

To define the system's available network interfaces, make sure the wifi and wifi_p2p interface items is defined in the networkAttributes resource configuration. For example:

```
<string-array translatable="false" name="networkAttributes">
    <item>"wifi,1,1,1,-1,true"</item>
    <item>"bluetooth,7,7,0,-1,true"</item>
    <item>"ethernet,9,9,2,-1,true"</item>
    <item>"wifi_p2p,13,1,0,-1,true"</item>
</string-array>
```

- **radioAttributes**

To define the system's available network interfaces, we need to define interface items for wifi in the networkAttributes resource configuration. For example:

```
<string-array translatable="false" name="radioAttributes">
    <item>"1,1"</item>
    <item>"7,1"</item>
    <item>"9,1"</item>
</string-array>
```

- **config_tether_wifi_regexs**

The interfaces set here are used as the interfaces for Wi-Fi LAN port. We use 'wlan0' by default when our Wi-Fi is set as softap mode. So it needs to set 'wlan0' here for system to recognized 'wlan0' as Wi-Fi LAN port. For example:

```
<string-array translatable="false" name="config_tether_wifi_regexs">
    <item>"wlan0"</item>
</string-array>
```

- **config_tether_upstream_types**

The connection types set here are used as the interfaces for WAN port to connect to internet. You could declared an entry in your platform dependent config.xml file to override the global definition. For example, adding wifi and ethernet:

```
<integer-array translatable="false" name="config_tether_upstream_types">
    <item>1</item>
    <item>9</item>
</integer-array>
```

At least one item should be declared here to enable the "Tehtering&portable hotspot" option of WirelessSettings in Settings.apk.

To know the definition and set other upstream connection types, please refer to frameworks/base/core/java/android/net/ConnectivityManager.java.

4. libhardware_legacy

The libhardware_legacy library includes functionality for Wi-Fi to operate. We have made modifications and extensions for our Wi-Fi solutions. To apply this, please go through the following instructions:

4.1. Apply wifi_realtek.c

Modify hardware/libhardware_legacy/wifi/Android.mk to include wifi_realtek.c instead of wifi.c into LOCAL_SRC_FILES. For example:

```
ifeq ($(BOARD_WIFI_VENDOR), realtek)
LOCAL_SRC_FILES += ../realtek/wlan/libhardware_legacy/wifi/wifi_realtek.c
else
LOCAL_SRC_FILES += wifi/wifi.c
endif
```

5. wpa_supplicant_8

We provide wpa_supplicant_8_jb_4.1_rtw_r6986.20130327.tar.gz or newer version in the wpa_supplicant_hostapd/ of our SW release package. You can:

- **Compare and merge with your own wpa_supplicant_8**

  Compare and merge from wpa_supplicant_8_jb_4.1_rtw by your own. For both external/wpa_suppliant_8/wpa_supplicant/Android.mk and external/wpa_suppliant_8/hostapd/Android.mk, you should notice that the two macros REALTEK_WIFI_VENDOR and ANDROID_P2P should be added into L_CFLAGS. For example:

```
ifeq ($(BOARD_WLAN_DEVICE), bcmdhd)
L_CFLAGS += -DANDROID_P2P
endif

ifeq ($(BOARD_WIFI_VENDOR), realtek)
L_CFLAGS += -DREALTEK_WIFI_VENDOR
L_CFLAGS += -DANDROID_P2P
Endif

# Use Android specific directory for control interface sockets
```

Here is the description of the specific macros:

MACRO	Description	
ANDROID_P2P	Android's wpa_supplicant_8 patch.	Must
REALTEK_WIFI_VENDOR	General purpose patch made by Realtek.	Must

- **Use the wpa_supplicant_8_jb_4.1_rtw instead of the original**
  A. Backup and remove the original external/wpa_supplcant_8/ folder
  B. Extract and copy the wpa_supplicant_8_jb_4.1_rtw tar file to the external/ folder of your Android SDK.
  C. Rename wpa_supplicant_8_jb_4.1_rtw as wpa_supplicant_8

※ We have enabled the ANDROID_P2P and REALTEK_WIFI_VENDOR macros by default.

6. Patches for Android's frameworks/base/wifi/

6.1. Patch for WifiP2pService.java

Add or modify the following code segment in WifiP2pService.java. For the specific line number and code segments, please reference our reference code:

frameworks/base/wifi/java/android/net/wifi/p2p/WifiP2pService.java

- **Line 851:**

```
        case WifiMonitor.P2P_FIND_STOPPED_EVENT:
            /*=== Realtek delete start ===*/
            // When discovery stops in inactive state, flush to clear
            // state peer data
            //mWifiNative.p2pFlush();
            /*=== Realtek delete end ===*/
            mServiceDiscReqId = null;
            sendP2pDiscoveryChangedBroadcast(false);
            break;
```

- **Line 878:**

```
/*=== Realtek add start ===*/
case WifiMonitor.P2P_GROUP_STARTED_EVENT:
    mGroup = (WifiP2pGroup) message.obj;
    if (DBG) logd(getName() + " group started");
    if (mGroup.isGroupOwner()) {
        startDhcpServer(mGroup.getInterface());
        if (mAutonomousGroup)
            mWifiNative.setP2pGroupIdle(mGroup.getInterface(), 0);
    } else {
        // Set group idle only for a client on the group interface to speed up
        // disconnect when GO is gone. Setting group idle time for a group owner
        // causes connectivity issues for new clients
        mWifiNative.setP2pGroupIdle(mGroup.getInterface(), GROUP_IDLE_TIME_S);
        mDhcpStateMachine = DhcpStateMachine.makeDhcpStateMachine(mContext,
                P2pStateMachine.this, mGroup.getInterface());
        mDhcpStateMachine.sendMessage(DhcpStateMachine.CMD_START_DHCP);
        WifiP2pDevice groupOwner = mGroup.getOwner();
        mPeers.updateStatus(groupOwner.deviceAddress, WifiP2pDevice.CONNECTED);
        sendP2pPeersChangedBroadcast();
    }
    mSavedPeerConfig = null;
    transitionTo(mGroupCreatedState);
    break;
/*=== Realtek add end ===*/
default:
    return NOT_HANDLED;
```

Line 1065:

```
                    /*=== Realtek add start ===*/
                    case WifiMonitor.P2P_PROV_DISC_FAILURE_EVENT:
                        loge("provision discovery failed");
                        handleGroupCreationFailure();
                        transitionTo(mInactiveState);
                        break;
                    /*=== Realtek add end ===*/
                    default:
                        return NOT_HANDLED;
```

Line 1095:

```
        case WifiMonitor.P2P_GROUP_STARTED_EVENT:
            mGroup = (WifiP2pGroup) message.obj;
            if (DBG) logd(getName() + " group started");
            if (mGroup.isGroupOwner()) {
                startDhcpServer(mGroup.getInterface());
                /*=== Realtek add start ===*/
                if (mAutonomousGroup)
                    mWifiNative.setP2pGroupIdle(mGroup.getInterface(), 0);
                /*=== Realtek add end ===*/
            } else {
```

Line 1460:

```
                logd("Stopped Dhcp server");

            /*=== Realtek add start ===*/
            try {
                mNwService.clearInterfaceAddresses(mGroup.getInterface());
                mNwService.disableIpv6(mGroup.getInterface());
            } catch (Exception e) {
                loge("Failed to clear addresses or disable ipv6" + e);
            }
            /*=== Realtek add end ===*/
        }
```

6.2. Patch for WifiMonitor.java

Add or modify the following code segment in WifiMonitor.java. For the specific line number and code segments, please reference our reference code:

frameworks/base/wifi/java/android/net/wifi/WifiMonitor.java

- **Line 217:**

```
    private static final String P2P_PROV_DISC_SHOW_PIN_STR = "P2P-PROV-DISC-SHOW-PIN";
    /*=== Realtek add start ===*/
    /* P2P-PROV-DISC-FAILURE p2p_dev_addr=42:fc:89:e1:e2:27 */
    private static final String P2P_PROV_DISC_FAILURE_STR = "P2P-PROV-DISC-FAILURE";
    /*=== Realtek add end ===*/
```

- **Line 323:**

```
    public static final int P2P_SERV_DISC_RESP_EVENT                = BASE + 38;
    /*=== Realtek add start ===*/
    public static final int P2P_PROV_DISC_FAILURE_EVENT             = BASE + 39;
    /*=== Realtek add end ===*/
```

- **Line 614:**

```
    } else if (dataString.startsWith(P2P_PROV_DISC_SHOW_PIN_STR)) {
        mStateMachine.sendMessage(P2P_PROV_DISC_SHOW_PIN_EVENT,
                new WifiP2pProvDiscEvent(dataString));
    /*=== Realtek add start ===*/
    } else if (dataString.startsWith(P2P_PROV_DISC_FAILURE_STR)) {
        mStateMachine.sendMessage(P2P_PROV_DISC_FAILURE_EVENT);
    /*=== Realtek add end ===*/
    } else if (dataString.startsWith(P2P_SERV_DISC_RESP_STR)) {
```

7. Driver Configurations for Android JB

Android JB support two scenarios for Wi-Fi solution:

- **STA/AP – Switch between STA and AP mode**
- **(STA+P2P)/AP – Switch between STA+P2P concurrent and AP mode**

The configuration of driver to fit the requirement of each scenario, see the following table:

MACRO	STA /AP	(STA+P2P)/AP	Kernel ver.
CONFIG_IOCTL_CFG80211	Defined	Defined	ver. >= 2.6.35
RTW_USE_CFG80211_STA_EVENT	Defined	Defined	ver. >= 3.2.0
CONFIG_CONCURRENT_MODE	Undefined	Defined	-
CONFIG_P2P_IPS	Don't Care	Defined	-

To use RTW_USE_CFG80211_STA_EVENT on the system with kernel version between 3.0 and 3.2, please refer to the patch file:

linux-3.0.42_STATION_INFO_ASSOC_REQ_IES.diff

Please modify both the include/autoconf.h and the specific autoconf file(needed for compound driver release) for your Wi-Fi product.

Chip type	Autoconf file to modify
RTL8192CU-series	autoconf_rtl8192c_usb_linux.h
RTL8192CE-series	autoconf_rtl8192c_pci_linux.h
RTL8192DU-series	autoconf_rtl8192d_usb_linux.h
RTL8192DE-series	autoconf_rtl8192d_pci_linux.h
RTL8723AS-series	autoconf_rtl8723a_sdio_linux.h
RTL8723AU-series	autoconf_rtl8723a_usb_linux.h
RTL8189ES-series	autoconf_rtl8189e_sdio_linux.h
RTL8188EU-series	autoconf_rtl8188e_usb_linux.h

For example, if you want to configure RTL8192CU-series driver (ex: RTL8188CUS, RTL8192CU) to fit the scenario of (STA+P2P)/AP, make sure the macros: CONFIG_IOCTL_CFG80211, RTW_USE_CFG80211_STA_EVENT and CONFIG_CONCURRENT_MODE, CONFIG_P2P_IPS in both include/autoconf.h and autoconf_rtl8192c_usb_linux.h(needed for compound driver release) are defined. As following:

```
#define CONFIG_IOCTL_CFG80211
#ifdef CONFIG_IOCTL_CFG80211
        #define RTW_USE_CFG80211_STA_EVENT
        //#define CONFIG_CFG80211_FORCE_COMPATIBLE_2_6_37_UNDER
        //#define CONFIG_DEBUG_CFG80211 1
#endif
…
#define CONFIG_CONCURRENT_MODE
…
#define CONFIG_P2P_IPS
```