Easy-SoC-libs WiFi library

libwifi-6.so

Generated by Doxygen 1.8.11

Contents

Chapter 1

Architecture and Design goals

The easy-soc-libs is a collection of libraries (Linux shared objects), which provide well defined, abstract and hardware agnostic APIs for different subsystems like WiFi, DSL, Ethernet etc.The APIs provide interfaces to the underlying platform/hardware for setting parameters and getting status/statistics information.

Users of the easy-soc-libs can focus on the application logic and not bother about the nitty-gritty nuances of a platform/hardware.

See IopsysWrt design and architecture documents to know more about easy-soc-libs.

This document focuses only on the easy-soC-libs's WiFi library, which is called **libwifi.so**.

Chapter 2

WiFi Objects

Every WiFi module creates atleast one Linux network interface.

Users through this interface can set/get parameters like ssid, bssid, channel, encryption etc. of the WiFi device. It is the WiFi module's MAC (or layer2) interface.

This interface can function in one of the various WiFi modes that a WiFi module supports viz. AP (or Master), Client (or managed), Monitor, AdHoc etc.

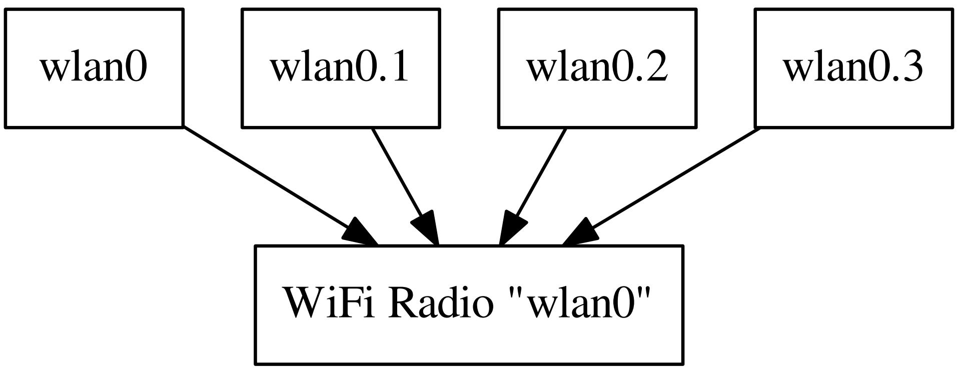
Since, IOPSYSWRT is a Router/AP/Gateway software, the WiFI interfaces which function in either AP or Client modes are of interest and can be managed through the easy-soc-libs's "libwifi" library.

Any 'real' network interface must also have a PHY associated with it for it to communicate with the world. In WiFi, this PHY device is the WiFi's Radio interface. The Radio interface has its own set of registers, fifos, states and status. It represents Layer1 of the WiFI device.

Thus, a WiFi device can be represented as a Radio interface plus a MAC interface.

For simplicity, the MAC interface is called only interface (i.e. without the MAC part), and the radio interface is called radio.

Libwifi's API header file "wifi.h" defines data structures that map to a WiFi device's radio and ap-interface - "struct wifi_radio" and "struct wifi_ap" respectively.



In the above figure, the first (or main) interface name is "wlan0", which is the same as the radio name "wlan0". Additional (virtual) interfaces have names wlan0.1, wlan0.2 etc. and so on.

Chapter 3

Add support for a new WiFi module

This chapter describes how to easily add support for a "new_wifi" WiFi module.

3.1 Implement libwifi APIs for the new WiFi module

It is broadly a four step process:

Step 1. Create a new file "new_wifi_driver.c" within the 'modules' directory. This file will implement radio and ap related operations for the new wifi. Define structure instance for the new_wifi driver's operations as follows -

```
struct wifi_driver new_wifi = {
        .name = "new",  /* new_wifi driver creates interface names starting with this */
        .radio.info = new_wifi_radio_info,
        .ap.get_ssid = new_wifi_get_ssid,
        .get_channel = new_wifi_get_channel,

         /* Add others operations as necessary */

         /* See 'nlwifi.c' within the 'modules' folder for implementation of nl/cfg80211 drivers. */
};
```

Step 2. Add "new_wifi" in drivers.c -

```
const struct wifi_driver *wifi_drivers[] = {
        :
        :
        .
#ifdef NEW_WIFI_MODULE
        &new_wifi,
#endif
};
```

Step 3. Add in drivers.h file the following lines -

```
        :
        :
        .
#ifdef NEW_WIFI_MODULE
extern const struct wifi_driver new_wifi;
#endif
```

Step 4. Finally include "new_wifi" to the build -

Add in the Makefile

```
        :
        .
objs_lib += modules/new_wifi_driver.o
```

After successfully building the package with the new_wifi module, a couple of .so files will be generated -

libwifi-X.so.a.b.c
libwifi-6.so.4
libwifi-6.so
[where X = is based on the wifi.h file's version implementation,
a, b, c = major, minor and revision number of the libwifi-X.so.a]

This chapter describes how to easily add support for a "new_wifi" WiFi module.

3.2 Implement libwifi APIs for the new WiFi module

It is broadly a four step process:

Step 1. Create a new file "new_wifi_driver.c" within the 'modules' directory. This file will implement radio and ap related operations for the new wifi. Define structure instance for the new_wifi driver's operations as follows -

```
struct wifi_driver new_wifi = {
        .name = "new",  /* new_wifi driver creates interface names starting with this */
        .radio.info = new_wifi_radio_info,
        .ap.get_ssid = new_wifi_get_ssid,
        .get_channel = new_wifi_get_channel,

         /* Add others operations as necessary */

         /* See 'nlwifi.c' within the 'modules' folder for implementation of nl/cfg80211 drivers. */
};
```

Step 2. Add "new_wifi" in drivers.c -

```
const struct wifi_driver *wifi_drivers[] = {
        :
        :
        .
#ifdef NEW_WIFI_MODULE
        &new_wifi,
#endif
};
```

Step 3. Add in drivers.h file the following lines -

```
        :
        :
        .
#ifdef NEW_WIFI_MODULE
extern const struct wifi_driver new_wifi;
#endif
```

Step 4. Finally include "new_wifi" to the build -

Add in the Makefile

```
        :
        .
objs_lib += modules/new_wifi_driver.o
```

After successfully building the package with the new_wifi module, a couple of .so files will be generated -

libwifi-X.so.a.b.c
libwifi-6.so.a
libwifi-6.so
[where X = is based on the wifi.h file's version implementation,
a, b, c = major, minor and revision number of the libwifi-X.so.a]

Chapter 4

Add support for receiving events in new WiFi module

4.1 Register, receive and dispatch events in the new WiFi module

This section describes how to easily add support for receiving (f.e. from a new netlink family/group) and dispatching of events in the "new_wifi" module.

Step 1. Implement the events' registration and receive functions -

In new_wifi_driver.c file, implement "register_event" and "recv_event" operations -

```
struct wifi_driver new_wifi = {
        :
        :
        .
        .register_event = new_wifi_register_event,
        .recv_event = nlwifi_recv_event,
        :
};

int new_wifi_register_event(const char *ifname, struct event_struct *req,
                                                void **handle)
{
        /* handle new_wifi vendor events, if any */
        if (!strncmp(req->family, "nl80211", 7) &&
                !(strncmp(req->group, "vendor", 6))) {

                req->override_cb = new_wifi_handle_vendor_event;
        }

        return nlwifi_register_event(ifname, req, handle);
}

int new_wifi_handle_vendor_event(struct event_struct *ev)
{
        struct nlwifi_event_vendor_resp *r =
                        (struct nlwifi_event_vendor_resp *)ev->resp.data;

        if (r->oui != OUI_NEW_WIFI)
                return 0;       /* discard as not ours */

        /* 'r->subcmd' holds vendor specific commands for handling */
                :
                :
                .
        /* dispatch event through 'ev->cb()' after any processing etc. */
        if (ev->cb) {
                return ev->cb(ev);
        }

        return 0;
}
```

Libwifi's internal API 'nlwifi_recv_event' is used here receive the new_wifi driver's "nl80211" vendor specific events. Obviously, any netlink famiy/group can be easily supported by implementing the 'register_event' and 'recv_event' functions.

Chapter 5

Using libwifi APIs

5.1 Functions and APIs

Making use of the libwifi APIs is easy. Users simply include the library header "wifi.h" in their main application code, and build by linking against the library .so file with the "-lwifi-6" flag.

User application can use the libwifi_supports() API to check if a specific API is implemented for the WiFi module.

5.2 Receiving Events

Receiving events through libwifi is also easy. The user application first has to initialize the struct event_struct with information about the event of interest. It then calls wifi_register_event() to register for the event, passing a 'void∗ handle' as the last argument to the function.

In order to receive events, the application has to call wifi_recv_event(), again passing the same 'void ∗handle' pointer that it passed to the register function.

```
int app_register_and_recv_event(struct app_private *priv, ...)
{
        :
        int ret;
        int err;
        void *handle;
        struct event_struct event;
        :
        .
        /* prepare event_struct for registration */

        memset(&event, 0, sizeof(struct event_struct));
        strncpy(event.ifname, ifname, 16);  /* interface name */
        strncpy(event.family, family, 32);  /* netlink family name */
        strncpy(event.group, group, 32);    /* netlink group name */
        event.priv = priv;                  /* application private data */
        event.cb = app_event_cb;            /* callback function after recv event */

        /* setup response buffer */
        event.resp.data = calloc(512, sizeof(uint8_t));
        if (event.resp.data == NULL)
                return -ENOMEM;


        :
        .
        ret = wifi_register_event((char *)ifname, &event, &handle);
        if (ret)
                return ret;   /* handle error */
        /* receive events */
        for (;;) {
                err = wifi_recv_event((char *)ifname, handle);
                if (err < 0)
                        fprintf(stderr, "Error: %s\n", __func__);
        }

        return 0;
}
```

and

```
int app_event_cb(struct event_struct *e)
{
        struct app_private *priv = (struct app_private *)e->priv;
        struct event_response *resp = &e->resp;
        char evtbuf[512] = {0};

        switch (resp->type) {
        case WIFI_EVENT_SCAN_START:
                /* handle events */

                /* resp holds event response buffer, if any */
                break;
        case WIFI_EVENT_SCAN_END:
                :
                .
        }

        :
}
```

Chapter 6

Data Structure Index

6.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 7

Data Structure Documentation

7.1 acs_param Struct Reference

struct acs_param - auto channel sel arguments

7.1.1 Detailed Description

struct acs_param - auto channel sel arguments

7.2 fbt_keys Struct Reference

Data Fields

- uint8_t **ap_address** [6]
- uint8_t r1kh_id [FT_R1KH_ID_LEN]

  bssid
- uint8_t **s1kh_id** [6]
- uint8_t pmk_r0_name [WPA_PMK_NAME_LEN]

  mac address of sta
- uint8_t **pmk_r1** [PMK_LEN]
- uint8_t **pmk_r1_name** [WPA_PMK_NAME_LEN]
- uint8_t **r0kh_id** [FT_R0KH_ID_MAX_LEN]
- uint8_t **r0kh_id_len**
- uint16_t **pairwise**

7.3 mimo_rate Struct Reference

for phyrate calculation

Data Fields

- uint8_t mcs

  MCS value.
- uint8_t bw

  Bandwidth in Mhz.
- uint8_t sgi

  = 1 if SGI enabled; else 0
- uint8_t nss

  Number of SS.

7.3.1 Detailed Description

for phyrate calculation

7.4 nbr Struct Reference

Data Fields

- uint8_t bssid [6]

  Bssid.
- uint32_t bssid_info

  as in IEEE 802.11-2016 9.4.2.37
- uint8_t reg

  regulatory region
- uint8_t channel

  channel
- uint8_t phy

  of enum wifi_phytype

7.5 nbr_header Struct Reference

struct nbr_header - meta data for 'struct nbr'

Data Fields

- uint32_t **flags**

7.5.1 Detailed Description

struct nbr_header - meta data for 'struct nbr'

7.6 rrm_measurement_beacon_request Struct Reference

Data Fields

- uint8_t oper_class

  Operating Class.
- uint8_t channel

  Channel Number.
- uint16_t rand_interval

  Randomization Interval (in TUs)
- uint16_t duration

  Measurement Duration (in TUs)
- uint8_t mode

  Measurement Mode.
- uint8_t bssid [6]

  BSSID.
- uint8_t variable [0]

  Optional Subelements.

7.7 scan_param Struct Reference

struct scan_param - scan request parameters

Data Fields

- char ssid [33]

  ssid specific scan
- uint8_t bssid [6]

  scan bssid
- uint32_t channel

  channel to scan
- uint8_t type

  auto (= 0), active (= 1), passive (=2)

7.7.1 Detailed Description

struct scan_param - scan request parameters

7.8 sta_nbr Struct Reference

Data Fields

- uint8_t **bssid** [6]
- int8_t **rssi**
- int8_t **rsni**

7.9 vendor_ie Struct Reference

struct vendor_ie - vendor ie struct

Data Fields

- struct {
  __u8 **eid**
  __u8 **len**
  } **ie_hdr**
- __u8 **oui** [OUI_LEN]
- __u8 **data** [ ]

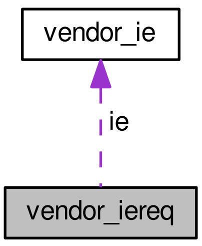
7.9.1 Detailed Description

struct vendor_ie - vendor ie struct

7.10 vendor_iereq Struct Reference

struct vendor_iereq - vendor specific ie request struct

Collaboration diagram for vendor_iereq:



Data Fields

- __u32 mgmt_subtype

  bitmap of management frame subtypes
- struct vendor_ie ie

  vendor ie structure

7.10.1 Detailed Description

struct vendor_iereq - vendor specific ie request struct

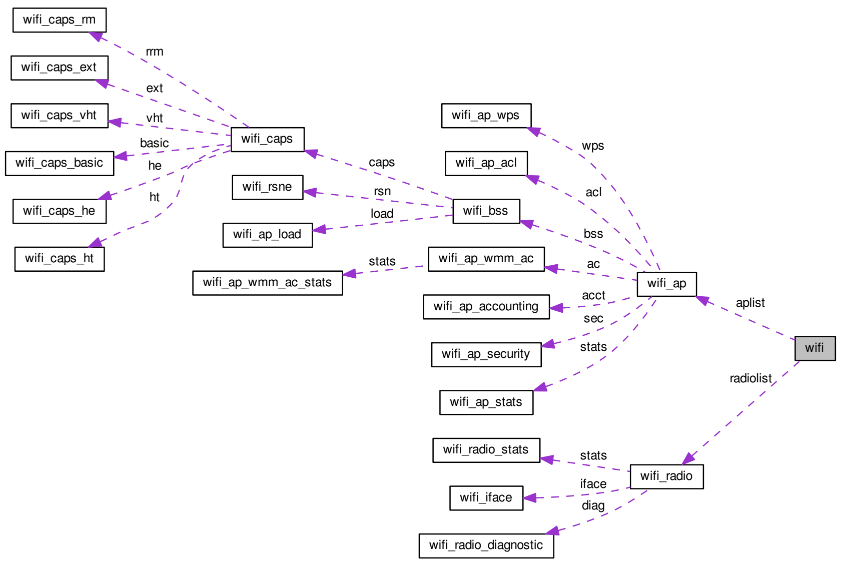
7.11 vlan_param Struct Reference

Data Fields

- uint8_t **dir**
- uint8_t **pcp**
- uint16_t **vid**

7.12 wifi Struct Reference

Collaboration diagram for wifi:



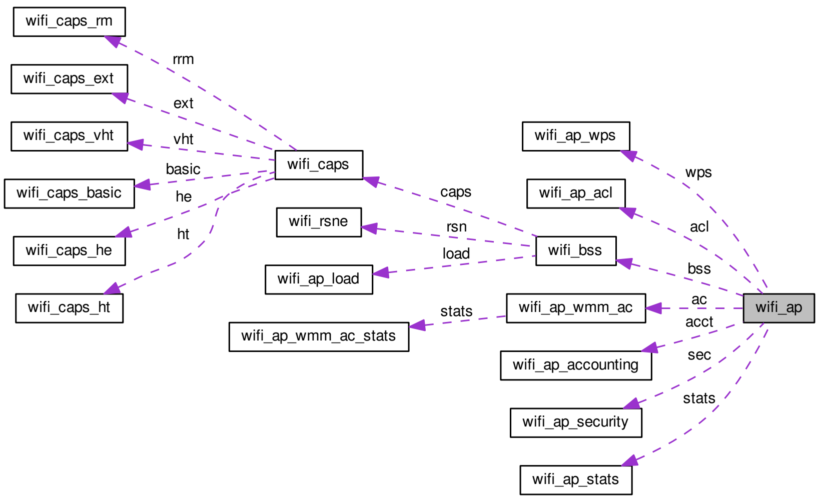
Data Fields

- uint32_t **num_radio**
- uint32_t **num_ap**
- struct wifi_radio ∗ **radiolist**
- struct wifi_ap ∗ aplist

    points to struct wifi_radio array

7.13 wifi_ap Struct Reference

Collaboration diagram for wifi_ap:



Data Fields

- struct wifi_bss **bss**
- enum wifi_ap_confstatus **confstatus**
- ifopstatus_t **opstatus**
- bool **ssid_advertised**
- bool **wmm_cap**
- bool **uapsd_cap**
- bool **wmm_enabled**
- bool **uapsd_enabled**
- uint32_t **assoclist_max**
- bool **isolate_enabled**
- struct wifi_ap_acl **acl**
- struct wifi_ap_security **sec**
- struct wifi_ap_wps **wps**
- struct wifi_ap_accounting **acct**
- struct wifi_ap_wmm_ac **ac** [WIFI_NUM_AC]
- struct wifi_ap_stats **stats**
- uint32_t **assoclist_num**
- void ∗ **assoclist**

7.14 wifi_ap_accounting Struct Reference

Data Fields

- bool **enable**
- ipaddress_t **server** [WIFI_NUM_RADIUS]
- uint32_t **server_port** [WIFI_NUM_RADIUS]
- char **secret** [WIFI_NUM_RADIUS][128]
- uint32_t **intm_interval**

7.15 wifi_ap_acl Struct Reference

Data Fields

- bool **acl_enabled**
- enum acl_policy **policy**
- void ∗ **allowlist**
- void ∗ denylist

  points to array of STA macaddress

7.16 wifi_ap_load Struct Reference

struct wifi_ap_load - Bss load

Data Fields

- uint16_t sta_count

  number of STAs connected
- uint8_t utilization

  channel utilization [0..255]
- uint16_t available

  available admission capacity

7.16.1 Detailed Description

struct wifi_ap_load - Bss load

7.17 wifi_ap_security Struct Reference

Data Fields

- uint32_t **supp_modes**
- uint32_t curr_mode

  bitmap of supported WIFI_SECURITY_∗
- uint8_t wepidx

  from wifi_rsnie in beacon/probe-resp
- uint8_t **wep104** [WIFI_NUM_WEPKEYS][13]
- uint8_t **wep40** [WIFI_NUM_WEPKEYS][5]
- uint8_t **psk** [32]
- char **passphrase** [64]
- uint32_t **rekey_int**
- ipaddress_t **radius_server** [WIFI_NUM_RADIUS]
- uint32_t **radius_port** [WIFI_NUM_RADIUS]
- char **radius_secret** [WIFI_NUM_RADIUS][128]
- enum wifi_mfp_config **mfp**

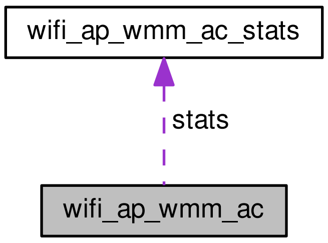
7.18 wifi_ap_stats Struct Reference

Data Fields

- unsigned long **tx_bytes**
- unsigned long **rx_bytes**
- unsigned long **tx_pkts**
- unsigned long **rx_pkts**
- uint32_t **tx_err_pkts**
- uint32_t **tx_rtx_pkts**
- uint32_t **tx_rtx_fail_pkts**
- uint32_t **tx_retry_pkts**
- uint32_t **tx_mretry_pkts**
- uint32_t **ack_fail_pkts**
- uint32_t **aggr_pkts**
- uint32_t **rx_err_pkts**
- unsigned long **tx_ucast_pkts**
- unsigned long **rx_ucast_pkts**
- uint32_t **tx_dropped_pkts**
- uint32_t **rx_dropped_pkts**
- unsigned long **tx_mcast_pkts**
- unsigned long **rx_mcast_pkts**
- unsigned long **tx_bcast_pkts**
- unsigned long **rx_bcast_pkts**
- unsigned long **rx_unknown_pkts**

7.19 wifi_ap_wmm_ac Struct Reference

Collaboration diagram for wifi_ap_wmm_ac:



Data Fields

- enum wmm_ac_type **ac**
- uint8_t **aifsn**
- uint8_t **cwmin**
- uint8_t **cwmax**
- uint8_t **txop**
- bool **ack_policy**
- struct wifi_ap_wmm_ac_stats **stats**

7.20 wifi_ap_wmm_ac_stats Struct Reference

Data Fields

- uint64_t **tx_bytes**
- uint64_t **rx_bytes**
- uint32_t **tx_pkts**
- uint32_t **rx_pkts**
- uint32_t **tx_err_pkts**
- uint32_t **rx_err_pkts**
- uint32_t **tx_rtx_pkts**

7.21 wifi_ap_wps Struct Reference

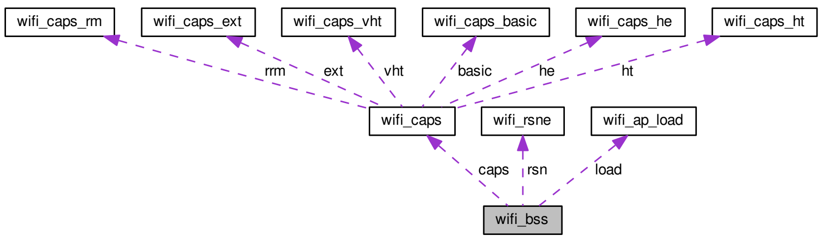
Data Fields

- bool **enable**
- uint32_t **supp_methods**
- enum wps_method en_method

  bitmap of enum wps_method
- enum wps_status **status**
- uint32_t **version**
- char **pin** [8]

7.22 wifi_bss Struct Reference

Collaboration diagram for wifi_bss:



Data Fields

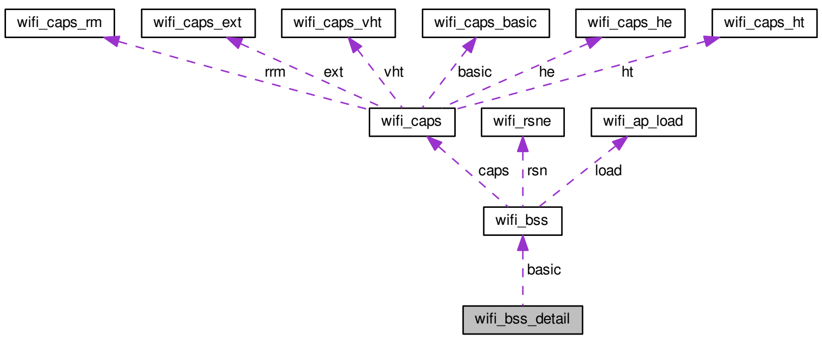
- uint8_t **ssid** [33]
- uint8_t **bssid** [6]
- enum wifi_bss_mode **mode**
- uint8_t **channel**
- enum wifi_bw **curr_bw**
- enum wifi_band **band**
- uint8_t **supp_std**
- uint8_t **oper_std**
- int **rssi**
- int **noise**
- struct wifi_rsne **rsn**
- uint32_t **auth**
- uint32_t **enc**
- uint32_t security

  bitmap of enum wifi_security
- uint32_t **beacon_int**
- uint32_t **dtim_period**
- struct wifi_ap_load **load**
- struct wifi_caps **caps**
- uint8_t cbitmap [16]

  bitmap for enum wifi_capflags

7.23 wifi_bss_detail Struct Reference

Collaboration diagram for wifi_bss_detail:



Data Fields

- struct wifi_bss **basic**
- uint32_t **ielen**
- uint8_t **ie** [1024]

7.24 wifi_btmreq Struct Reference

Data Fields

- uint8_t mode

  bitmap of WIFI_BTMREQ_∗
- uint16_t disassoc_tmo

  in tbtts when DISASSOC_IMM is set
- uint8_t validity_int

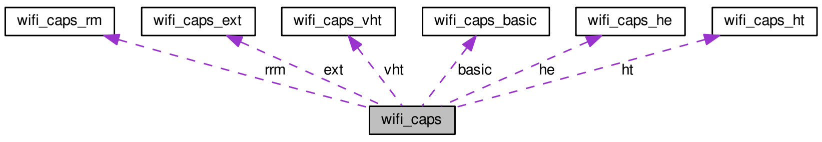
  in tbtts until candidate list is valid
- uint16_t bssterm_dur

  bss termination duration in minutes

7.25 wifi_caps Struct Reference

struct wifi_caps - wifi device/interface capabilities

Collaboration diagram for wifi_caps:



Data Fields

- uint32_t **valid**
- struct wifi_caps_basic basic

  bitmap of caps available and valid
- struct wifi_caps_ext **ext**
- struct wifi_caps_ht **ht**
- struct wifi_caps_vht **vht**
- struct wifi_caps_rm **rrm**
- struct wifi_caps_he **he**

7.25.1 Detailed Description

struct wifi_caps - wifi device/interface capabilities

7.26 wifi_caps_basic Struct Reference

Data Fields

- union {
  uint8_t **byte** [2]
  uint16_t **cap**
  };

7.27 wifi_caps_ext Struct Reference

Data Fields

- uint8_t **byte** [16]

7.28 wifi_caps_he Struct Reference

Data Fields

- uint8_t **id_ext**
- uint8_t **byte_mac** [6]
- uint8_t **byte_phy** [11]
- uint8_t **byte_opt** [46]

7.29 wifi_caps_ht Struct Reference

Data Fields

- union {
  uint8_t **byte** [2]
  uint16_t **cap**
  };
- uint8_t **ampdu_params**
- uint8_t **supp_mcs** [16]
- union {
  uint8_t **byte_ext** [2]
  uint16_t **cap_ext**
  };
- uint8_t **txbf** [4]
- uint8_t **asel**

7.30 wifi_caps_rm Struct Reference

Data Fields

- uint8_t **byte** [5]

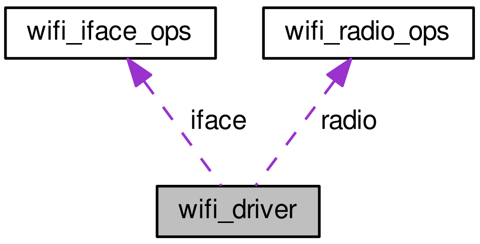
7.31 wifi_caps_vht Struct Reference

Data Fields

- union {
  uint8_t **byte** [4]
  uint32_t **cap**
  };
- uint8_t **supp_mcs** [8]

7.32 wifi_driver Struct Reference

Collaboration diagram for wifi_driver:



Data Fields

- const char ∗ **name**
- const char ∗∗(∗ **get_apis**)(const char ∗name)
- int(∗ **info**)(const char ∗name, struct wifi_metainfo ∗info)
- struct wifi_radio_ops **radio**
- struct wifi_iface_ops **iface**
- int(∗ **register_event**)(const char ∗ifname, struct event_struct ∗ev, void ∗∗evhandle)
- int(∗ **unregister_event**)(const char ∗ifname, void ∗evhandle)
- int(∗ **recv_event**)(const char ∗ifname, void ∗evhandle)
- const char ∗(∗ **get_version**)(void)

7.33 wifi_iface Struct Reference

struct wifi_iface - interface per wifi radio

Data Fields

- char **name** [16]
- enum wifi_mode **mode**

7.33.1 Detailed Description

struct wifi_iface - interface per wifi radio

7.34 wifi_iface_ops Struct Reference

WiFi interface related operations.

Data Fields

- int(∗ **start_wps**)(const char ∗ifname, struct wps_param wps)
- int(∗ **stop_wps**)(const char ∗ifname)
- int(∗ **get_wps_status**)(const char ∗ifname, enum wps_status ∗s)
- int(∗ **get_wps_pin**)(const char ∗ifname, unsigned long ∗pin)
- int(∗ **set_wps_pin**)(const char ∗ifname, unsigned long pin)
- int(∗ **get_wps_device_info**)(const char ∗ifname, struct wps_device ∗info)
- int(∗ **get_caps**)(const char ∗ifname, struct wifi_caps ∗caps)
- int(∗ **get_mode**)(const char ∗ifname, enum wifi_mode ∗mode)
- int(∗ **get_security**)(const char ∗ifname, uint32_t ∗auth, uint32_t ∗enc)
- int(∗ **add_vendor_ie**)(const char ∗ifname, struct vendor_iereq ∗req)
- int(∗ **del_vendor_ie**)(const char ∗ifname, struct vendor_iereq ∗req)
- int(∗ **get_vendor_ies**)(const char ∗ifname, struct vendor_ie ∗ies, int ∗num_ies)
- int(∗ **get_param**)(const char ∗ifname, const char ∗param, int ∗len, void ∗val)
- int(∗ **set_param**)(const char ∗ifname, const char ∗param, int len, void ∗val)
- int(∗ **vendor_cmd**)(const char ∗ifname, uint32_t vid, uint32_t subcmd, uint8_t ∗in, int inlen, uint8_t ∗out, int ∗outlen)
- int(∗ **subscribe_frame**)(const char ∗ifname, uint8_t type, uint8_t stype)
- int(∗ **set_4addr**)(const char ∗ifname, bool enable)
- int(∗ **set_vlan**)(const char ∗ifname, struct vlan_param vlan)
- int(∗ **ap_info**)(const char ∗name, struct wifi_ap ∗ap)
- int(∗ **get_bssid**)(const char ∗ifname, uint8_t ∗bssid)
- int(∗ **get_ssid**)(const char ∗ifname, char ∗ssid)
- int(∗ **get_stats**)(const char ∗ifname, struct wifi_ap_stats ∗s)
- int(∗ **get_beacon_ies**)(const char ∗ifname, uint8_t ∗ies, int ∗len)
- int(∗ **get_assoclist**)(const char ∗ifname, uint8_t ∗stas, int ∗num_stas)
- int(∗ **get_sta_info**)(const char ∗ifname, uint8_t ∗addr, struct wifi_sta ∗info)
- int(∗ **get_sta_stats**)(const char ∗ifname, uint8_t ∗addr, struct wifi_sta_stats ∗s)
- int(∗ **disconnect_sta**)(const char ∗ifname, uint8_t ∗sta, uint16_t reason)
- int(∗ **restrict_sta**)(const char ∗ifname, uint8_t ∗sta, int enable)

- int(∗ **monitor_sta**)(const char ∗ifname, uint8_t ∗sta, void ∗outdata)
- int(∗ **get_monitor_stas**)(const char ∗ifname, struct wifi_monsta ∗stas, int ∗num)
- int(∗ **add_neighbor**)(const char ∗ifname, struct nbr nbr)
- int(∗ **del_neighbor**)(const char ∗ifname, unsigned char ∗bssid)
- int(∗ **get_neighbor_list**)(const char ∗ifname, struct nbr ∗nbr, int ∗nr)
- int(∗ **req_beacon_report**)(const char ∗ifname, uint8_t ∗sta)
- int(∗ **get_beacon_report**)(const char ∗ifname, uint8_t ∗sta, struct sta_nbr ∗snbr, int ∗nr)
- int(∗ **req_bss_transition**)(const char ∗ifname, unsigned char ∗sta, int bsss_nr, unsigned char ∗bsss, unsigned int tmo)
- int(∗ **req_btm**)(const char ∗ifname, unsigned char ∗sta, int bsss_nr, uint8_t ∗bsss, struct wifi_btmreq ∗b)
- int(∗ **get_11rkeys**)(const char ∗ifname, unsigned char ∗sta, uint8_t ∗r1khid)
- int(∗ **set_11rkeys**)(const char ∗ifname, struct fbt_keys ∗fk)
- int(∗ **sta_info**)(const char ∗name, struct wifi_sta ∗sta)
- int(∗ **sta_get_stats**)(const char ∗ifname, struct wifi_sta_stats ∗s)
- int(∗ **sta_get_ap_info**)(const char ∗ifname, struct wifi_bss ∗info)
- int(∗ **sta_disconnect_ap**)(const char ∗ifname, uint32_t reason)

7.34.1 Detailed Description

WiFi interface related operations.

BSS/STA interface operations are handled through this structure.

int (∗start_wps)(const char ∗ifname, struct wps_param wps)
Start WPS registration

Parameters

in	*ifname*	interface name
in	*wps*	wps_param structure

int (∗stop_wps)(const char ∗ifname)
Stop ongoing WPS registration

Parameters

in	*ifname*	interface name

int (∗get_wps_status)(const char ∗ifname, enum wps_status ∗s)
Get latest wps registration status

Parameters

in	*ifname*	interface name
out	*s*	wps_param structure

int (∗get_wps_pin)(const char ∗ifname, unsigned long ∗pin)
Get AP's (i.e. own) WPS pin

Parameters

in	*ifname*	interface name
out	*pin*	wps pin value

int (∗set_wps_pin)(const char ∗ifname, unsigned long pin)
Set AP's (i.e. own) WPS pin

Parameters

in	*ifname*	interface name
in	*pin*	wps pin value

int (∗get_wps_device_info)(const char ∗ifname, struct wps_device ∗s)
Get WPS device information

Parameters

in	*ifname*	interface name
out	*s*	wps_device structure

int (∗get_caps)(const char ∗ifname, struct wifi_caps ∗caps)
Get capabilities

Parameters

in	*ifname*	interface name
out	*caps*	wifi_caps structure

int (∗get_mode)(const char ∗ifname, enum wifi_mode ∗mode)
Get WiFi mode

Parameters

in	*ifname*	interface name
out	*mode*	WiFi mode f.e. WIFI_MODE_AP or WIFI_MODE_STA etc.

int (∗get_security)(const char ∗ifname, uint32_t ∗auth, uint32_t ∗enc)
Get security info

Parameters

in	*ifname*	interface name
out	*auth*	authtication type
out	*enc*	encryption type

int (∗add_vendor_ie)(const char ∗ifname, struct vendor_iereq ∗req)
Add vendor specific ie element

Parameters

in	*ifname*	interface name
in	*req*	vendor_iereq structure

int (∗del_vendor_ie)(const char ∗ifname, struct vendor_iereq ∗req)
Delete vendor specific ie element

Parameters

in	*ifname*	interface name
in	*req*	vendor_iereq structure

int (∗get_vendor_ies)(const char ∗ifname, struct vendor_ie ∗ies, int ∗num_ies)
Get list of vendor information elements

Parameters

in	*ifname*	interface name
out	*ies*	array of struct vendor_ie
out	*num*	array size (number of elements)

int (∗get_param)(const char ∗ifname, const char ∗param, int ∗len, void ∗val)
Get AP parameter value(s).

Parameters

in	*ifname*	interface name
in	*param*	parameter name
out	*len*	length of the returned value
out	*val*	parameter value

int (∗set_param)(const char ∗ifname, const char ∗param, int len, void ∗val)
Set AP parameter value(s).

Parameters

in	*ifname*	interface name
in	*param*	parameter name
in	*len*	length of the parameter
in	*val*	value of parameter

int (∗vendor_cmd)(const char ∗ifname, uint32_t vid, uint32_t subcmd, uint8_t ∗in, int inlen, uint8_t ∗out, int ∗outlen)
Vendor specific command

Parameters

in	*ifname*	interface name
in	*vid*	vendor id (OUI)

Parameters

in	*subcmd*	(sub)command
in	*in*	input parameter
in	*inlen*	length of the input parameter
out	*out*	output parameter
out	*outlen*	length of the output parameter

int (∗subscribe_frame)(const char ∗ifname, uint8_t type, uint8_t stype)
Subscribe for received frames

Parameters

in	*name*	interface name
in	*type*	frame type as in IEEE802.11 Std.
in	*stype*	frame sub-type as in IEEE802.11 Std.

int (∗set_4addr)(const char ∗ifname, bool enable)
Enable or disable 4-address WDS mode.

Parameters

in	*ifname*	interface name
in	*enable*	enable = 1, else disable.

int (∗set_vlan)(const char ∗ifname, struct vlan_param vlan)
Set VLAN link.

Parameters

in	*ifname*	interface name
in	*vlan*	vlan parameters

int (∗ap_info)(const char ∗ifname, struct wifi_ap ∗ap)
Get detailed AP information

Parameters

in	*ifname*	interface name
out	*ap*	ap information

int (∗get_bssid)(const char ∗ifname, uint8_t ∗bssid)
Get BSSID

Parameters

in	*ifname*	interface name
out	*bssid*	BSSID buffer (6 bytes)

int (∗get_ssid)(const char ∗ifname, char ∗ssid)

Get SSID

Parameters

in	*ifname*	interface name
out	*ssid*	SSID buffer

int (∗get_stats)(const char ∗ifname, struct wifi_ap_stats ∗s)
Get statistics

Parameters

in	*ifname*	interface name
out	*s*	wifi_ap_stats structure

int (∗get_beacon_ies)(const char ∗ifname, uint8_t ∗ies, int ∗len)
Get Beacon frame information elements

Parameters

in	*ifname*	interface name
out	*ies*	information elements buffer
out	*len*	length of information elements buffer

int (∗get_assoclist)(const char ∗ifname, uint8_t ∗stas, int ∗num_stas)
Get STA association list

Parameters

in	*ifname*	interface name
out	*stas*	macaddresses of STAs
out	*num_stas*	number of STAs

int (∗get_sta_info)(const char ∗ifname, uint8_t ∗addr, struct wifi_sta ∗info)
Get STA information

Parameters

in	*ifname*	interface name
in	*addr*	macaddress of STA
out	*info*	STA information

int (∗get_sta_stats)(const char ∗ifname, uint8_t ∗addr, struct wifi_sta_stats ∗s)
Get STA statistics

Parameters

in	*ifname*	interface name
in	*addr*	macaddress of STA
out	*s*	STA counters

int (∗disconnect_sta)(const char ∗ifname, uint8_t ∗sta)
Disconnect STA

Parameters

in	*ifname*	interface name
in	*sta*	macaddress of STA
in	*reason*	disconnect reason code as in IEEE802.11 Std

int (∗restrict_sta)(const char ∗ifname, uint8_t ∗sta, int enable)
Assoc-control STA

Parameters

in	*ifname*	interface name
in	*sta*	macaddress of STA
in	*enable*	enable (= 1) or disable (= 0) assoc-control

int (∗monitor_sta)(const char ∗ifname, uint8_t ∗sta, void ∗outdata)
Monitor STA frames

Parameters

in	*ifname*	interface name
in	*sta*	macaddress of STA
out	*outdata*	monitored data

int (∗get_monitor_stas)(const char ∗ifname, struct wifi_monsta ∗stas, int ∗num)
Get monitored STA information

Parameters

in	*ifname*	interface name
out	*stas*	array of struct wifi_monsta
out	*num*	array size (number of wifi_monsta elements)

int (∗add_neighbor)(const char ∗ifname, struct nbr nbr)
Add a 802.11k neighbor entry

Parameters

in	*ifname*	interface name
in	*nbr*	nbr structure

int (∗del_neighbor)(const char ∗ifname, unsigned char ∗bssid)
Delete a 802.11k neighbor entry

Parameters

in	*ifname*	interface name
in	*bssid*	Bssid of the neighbor

int (∗get_neighbor_list)(const char ∗ifname, struct nbr ∗nbr, int ∗nr)
Get 802.11k neighbor list

Parameters

in	*ifname*	interface name
out	*nbr*	array of struct nbr
out	*nr*	number of array entries

int (∗req_beacon_report)(const char ∗ifname, uint8_t ∗sta)
Request 802.11k Beacon Report from a STA

Parameters

in	*ifname*	interface name
in	*sta*	macaddress of the STA

int (∗get_beacon_report)(const char ∗ifname, uint8_t ∗sta, struct sta_nbr ∗snbr, int ∗nr)
Get 802.11k Beacon Report received from a STA

Parameters

in	*ifname*	interface name
in	*sta*	macaddress of the STA
out	*snbr*	array of sta_nbr structures
out	*nr*	number of array entries

int (∗req_bss_transition)(const char ∗ifname, unsigned char ∗sta, int bsss_nr, unsigned char ∗bsss, unsigned int tmo)
[Deprecated] Request 802.11v BSS transition to a STA

Parameters

in	*ifname*	interface name
in	*sta*	macaddress of the STA
in	*bsss_nr*	number of neighbor bssids
in	*bsss*	array of neighbor bssids
in	*tmo*	timeout (secs) until this request is valid

int (∗req_btm)(const char ∗ifname, unsigned char ∗sta, int bsss_nr, unsigned char ∗bsss, struct wifi_btmreq ∗b)
Request 802.11v BSS transition to a STA

Parameters

in	*ifname*	interface name
in	*sta*	macaddress of the STA
in	*bsss_nr*	number of neighbor bssids
in	*bsss*	array of neighbor bssids
in	*b*	additional request parameters

int (∗get_11rkeys)(const char ∗ifname, unsigned char ∗sta, uint8_t ∗r1khid)
Get 802.11r keys

Parameters

`in`	*ifname*	interface name
`in`	*sta*	macaddress of the STA
`out`	*rikhid*	R1KHID

int (∗set_11rkeys)(const char ∗ifname, struct fbt_keys ∗fk)
Set 802.11r keys

Parameters

`in`	*ifname*	interface name
`in`	*fk*	fbt_keys struct

int (∗sta_info)(const char ∗ifname, struct wifi_sta ∗sta)
Get detailed STA information

Parameters

`in`	*ifname*	interface name
`out`	*sta*	STA information

int (∗sta_get_stats)(const char ∗ifname, struct wifi_sta_stats ∗s)
Get STA interface statistics

Parameters

`in`	*ifname*	interface name
`out`	*s*	STA interface statistics

int (∗sta_get_ap_info)(const char ∗ifname, struct wifi_bss ∗info)
Get BSS information of the STA's AP

Parameters

`in`	*ifname*	interface name
`out`	*info*	BSS information of STA's AP

int (∗sta_disconnect_ap)(const char ∗ifname, uint32_t reason)
Disconnect from STA's AP

Parameters

`in`	*ifname*	interface name
`in`	*reason*	disconnection reason code as in IEEE802.11 Std.

7.35 wifi_metainfo Struct Reference

struct wifi_metainfo - meta information about wifi module

Data Fields

- char vendor_id [8]

  0xvvvv
- char device_id [8]

  0xdddd
- char drv_data [128]

  driver name + version info
- char fw_data [128]

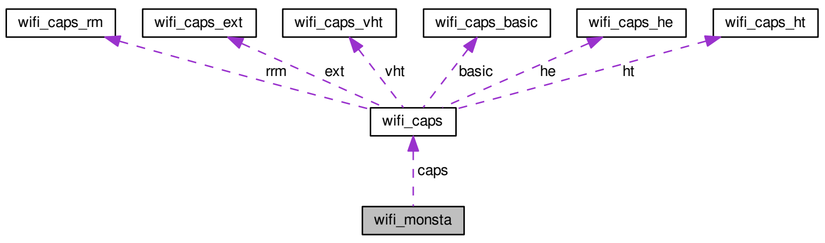
  firmware name + version

7.35.1 Detailed Description

struct wifi_metainfo - meta information about wifi module

7.36 wifi_monsta Struct Reference

Collaboration diagram for wifi_monsta:



Data Fields

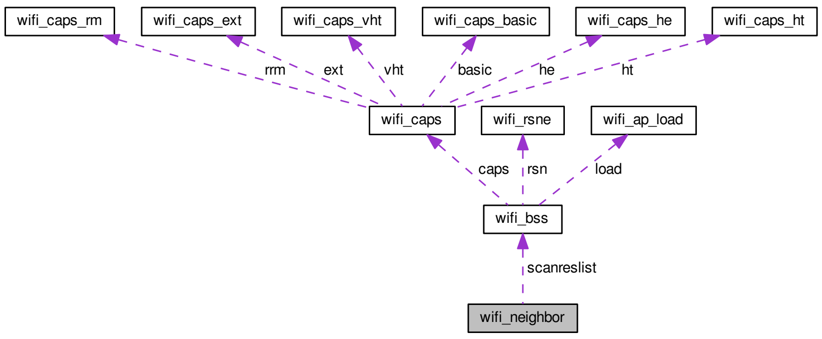
- uint8_t **macaddr** [6]
- int8_t rssi [WIFI_NUM_ANTENNA]

  latest rssi in dBm
- struct wifi_caps caps

  capabilities

7.37 wifi_neighbor Struct Reference

Collaboration diagram for wifi_neighbor:



Data Fields

- char **radio** [16]
- uint32_t num_result

  scanning wifi radio device name
- struct wifi_bss ∗ scanreslist

  num of scanned APs

7.38 wifi_opchannel Struct Reference

struct wifi_opchannel - channel definition in operating class

Data Fields

- int8_t txpower

  max txpower in dBm
- uint8_t **num**
- uint8_t **ch** [WIFI_NUM_CHANNEL_IN_OPCLASS]

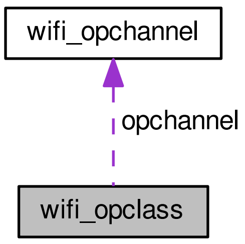
7.38.1 Detailed Description

struct wifi_opchannel - channel definition in operating class

7.39 wifi_opclass Struct Reference

struct wifi_opclass - operating class

Collaboration diagram for wifi_opclass:



Data Fields

- uint32_t **opclass**
- uint32_t **g_opclass**
- enum wifi_band **band**
- enum wifi_bw **bw**
- struct wifi_opchannel **opchannel**

7.39.1 Detailed Description

struct wifi_opclass - operating class

7.40 wifi_oper_he Struct Reference

struct wifi_oper_he - HE operational element

Data Fields

- uint8_t **id_ext**
- uint8_t **param** [3]
- uint8_t **color**
- uint8_t **basic_mcs** [2]

7.40.1 Detailed Description

struct wifi_oper_he - HE operational element

7.41 wifi_oper_ht Struct Reference

struct wifi_oper_ht - HT operation element

Data Fields

- uint8_t **channel**
- uint8_t **info** [5]
- uint8_t **basic_mcs** [16]

7.41.1 Detailed Description

struct wifi_oper_ht - HT operation element

7.42 wifi_oper_vht Struct Reference

struct wifi_oper_vht - VHT operation element

Data Fields

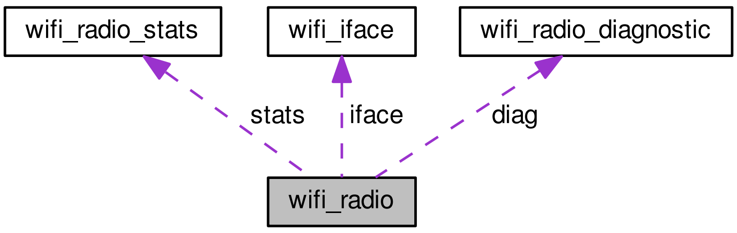
- uint8_t **channel_width**
- uint8_t **freq_mid_seg0**
- uint8_t **freq_mid_seg1**
- uint8_t **basic_mcs** [2]

7.42.1 Detailed Description

struct wifi_oper_vht - VHT operation element

7.43 wifi_radio Struct Reference

Collaboration diagram for wifi_radio:



Data Fields

- bool **enabled**
- uint8_t **tx_streams**
- uint8_t **rx_streams**
- uint32_t **max_bitrate**
- enum wifi_band oper_band

  exactly one band from supp_band
- uint8_t supp_band

  bitmap of wifi frequency bands
- uint8_t supp_std

  bitmap of wifi_std
- uint8_t oper_std

  bitmap of wifi_std from supp_std
- uint8_t channel

  current primary (ctrl) channel
- uint8_t **supp_channels** [64]
- uint8_t oper_channels [64]

  in use channels
- bool **acs_capable**
- bool **acs_enabled**
- uint32_t acs_interval

  in secs
- uint32_t supp_bw

  bitmap of wifi_bw
- enum wifi_bw **curr_bw**
- enum wifi_chan_ext extch

  current extension channel
- enum wifi_guard **gi**
- int8_t txpower

  -1 for auto; else in %-age
- int8_t txpower_dbm

  in dBm
- bool **dot11h_capable**
- bool **dot11h_enabled**
- char regdomain [4]

  countrycode + "O" | "I" + NUL
- uint8_t srl

  short retry limit
- uint8_t lrl

  long retry limit
- uint32_t **frag**
- uint32_t **rts**
- uint32_t beacon_int

  in msecs
- uint32_t **dtim_period**
- bool **aggr_enable**
- enum wifi_preamble **pr**
- uint32_t **basic_rates** [32]
- uint32_t **oper_rates** [32]
- uint32_t **supp_rates** [32]
- struct wifi_radio_stats **stats**

- struct wifi_radio_diagnostic **diag**
- uint8_t **max_iface_ap**
- uint8_t **max_iface_sta**
- uint8_t **num_iface**
- struct wifi_iface **iface** [WIFI_IFACE_MAX_NUM]

7.44 wifi_radio_diagnostic Struct Reference

struct wifi_radio_diagnostic - radio diagnostic data in latest second

Data Fields

- uint32_t **channel_busy**
- uint32_t tx_airtime

  in usecs
- uint32_t rx_airtime

  in usecs
- uint32_t obss_airtime

  in usecs
- uint32_t cca_time

  in usecs
- uint32_t false_cca_count

  in usecs

7.44.1 Detailed Description

struct wifi_radio_diagnostic - radio diagnostic data in latest second

7.45 wifi_radio_ops Struct Reference

wifi radio related operations.

Data Fields

- int(∗ **info**)(const char ∗name, struct wifi_radio ∗radio)
- int(∗ **get_supp_band**)(const char ∗name, uint32_t ∗bands)
- int(∗ **get_oper_band**)(const char ∗name, enum wifi_band ∗band)
- int(∗ **get_caps**)(const char ∗name, struct wifi_caps ∗caps)
- int(∗ **get_supp_stds**)(const char ∗name, uint8_t ∗std)
- int(∗ **get_oper_stds**)(const char ∗name, uint8_t ∗std)
- int(∗ **get_country**)(const char ∗name, char ∗alpha2)
- int(∗ **get_channel**)(const char ∗ifname, uint32_t ∗channel, enum wifi_bw ∗bw)
- int(∗ **set_channel**)(const char ∗ifname, uint32_t channel, enum wifi_bw bw)
- int(∗ **get_supp_channels**)(const char ∗name, uint32_t ∗chlist, int ∗num, const char ∗alpha2, enum wifi_↩
band f, enum wifi_bw b)

- int(∗ **get_oper_channels**)(const char ∗name, uint32_t ∗chlist, int ∗num, const char ∗alpha2, enum wifi_band f, enum wifi_bw b)
- int(∗ **get_supp_opclass**)(const char ∗name, int ∗num_opclass, struct wifi_opclass ∗o)
- int(∗ **get_curr_opclass**)(const char ∗name, struct wifi_opclass ∗o)
- int(∗ **get_bandwidth**)(const char ∗name, enum wifi_bw ∗bw)
- int(∗ **get_maxrate**)(const char ∗name, unsigned long ∗rate_kbps)
- int(∗ **get_basic_rates**)(const char ∗name, int ∗num, uint32_t ∗rates_kbps)
- int(∗ **get_oper_rates**)(const char ∗name, int ∗num, uint32_t ∗rates_kbps)
- int(∗ **get_supp_rates**)(const char ∗name, int ∗num, uint32_t ∗rates)
- int(∗ **get_stats**)(const char ∗ifname, struct wifi_radio_stats ∗s)
- int(∗ **scan**)(const char ∗name, struct scan_param ∗p)
- int(∗ **get_scan_results**)(const char ∗name, struct wifi_bss ∗bsss, int ∗num)
- int(∗ **get_bss_scan_result**)(const char ∗name, uint8_t ∗bssid, struct wifi_bss_detail ∗b)
- int(∗ **get_noise**)(const char ∗ifname, int ∗noise)
- int(∗ **acs**)(const char ∗name, struct acs_param ∗p)
- int(∗ **get_param**)(const char ∗name, const char ∗param, int ∗len, void ∗val)
- int(∗ **set_param**)(const char ∗name, const char ∗param, int len, void ∗val)
- int(∗ **add_iface**)(const char ∗name, enum wifi_mode m, char ∗argv[ ])
- int(∗ **del_iface**)(const char ∗name, const char ∗ifname)

7.45.1 Detailed Description

wifi radio related operations.

All radio/device specific operations are handled through this structure. In order to support a new wifi chipset, struct wifi_radio_ops alongwith struct wifi_iface_ops must be implemented by its driver module.

Unless otherwise mentioned, the following functions return 0 on Success, and -1 on Failure.

int (∗info)(const char ∗name, struct wifi_radio ∗radio).
Get information about the radio interface.

Parameters

in	*name*	radio interface name
out	*radio*	struct wifi_radio

int (∗get_supp_band)(const char ∗name, uint32_t ∗bands)
Get supported WiFi bands in bands param.

Parameters

in	*name*	radio interface name
out	*bands*	bitmap of bands from struct wifi_band

int (∗get_oper_band)(const char ∗name, enum wifi_band ∗band)
Get current operating WiFi band.

Parameters

in	*name*	radio interface name
out	*band*	band struct wifi_band type

int (∗get_caps)(const char ∗name, struct wifi_caps ∗caps)
Get WiFi radio capabilities.

Parameters

`in`	*name*	radio interface name
`out`	*caps*	capabilities struct wifi_caps

int (∗get_supp_stds)(const char ∗name, uint8_t ∗std)
Get WiFi supported standards.

Parameters

`in`	*name*	radio interface name
`out`	*std*	bitmap of #enum wifi_std

int (∗get_oper_stds)(const char ∗name, uint8_t ∗std)
Get WiFi operational standards.

Parameters

`in`	*name*	radio interface name
`out`	*std*	bitmap of enum wifi_std

int (∗get_country)(const char ∗name, char ∗alpha2)
Get operating country information.

Parameters

`in`	*name*	radio interface name
`out`	*alpha2*	country code

int (∗get_channel)(const char ∗ifname, uint32_t ∗channel, enum wifi_bw ∗bw)
Get operating channel information.

Parameters

`in`	*ifname*	radio interface name
`out`	*channel*	primary control channel
`out`	*bw*	channel bandwidth

int (∗set_channel)(const char ∗ifname, uint32_t channel, enum wifi_bw bw)
Set operating channel with bandwidth.

Parameters

`in`	*ifname*	radio interface name
`out`	*channel*	primary control channel
`out`	*bw*	channel bandwidth

int (∗get_supp_channels)(const char ∗name, uint32_t ∗chlist, int ∗num, const char ∗alpha2, enum wifi_band f, enum wifi_bw bw)

Get supported channels.

Parameters

in	*name*	radio interface name
out	*chlist*	array of channels
out	*num*	number of channels in chlist array
in	*alpha2*	country code
in	*f*	frequency band #enum wifi_band
in	*bw*	channel bandwidth enum wifi_bw

int (∗get_oper_channels)(const char ∗name, uint32_t ∗chlist, int ∗num, const char ∗alpha2, enum wifi_band f, enum wifi_bw b)

Get operating channels.

Parameters

in	*name*	radio interface name
out	*chlist*	array of channels
out	*num*	number of channels in chlist array
in	*alpha2*	country code
in	*f*	frequency band #enum wifi_band
in	*bw*	channel bandwidth enum wifi_bw

int (∗get_supp_opclass)(const char ∗name, int ∗num, struct wifi_opclass ∗o)

Get supported operating classes.

Parameters

in	*name*	radio interface name
out	*num*	number of operating classes supported
out	*o*	array of struct wifi_opclass elements

int (∗get_curr_opclass)(const char ∗name, int ∗num, struct wifi_opclass ∗o)

Get current operating class(es).

Parameters

in	*name*	radio interface name
out	*num*	number of current operating classes
out	*o*	array of struct wifi_opclass elements

int (∗get_bandwidth)(const char ∗name, enum wifi_bw ∗bw)

Get operating channel bandwidth.

Parameters

in	*name*	radio interface name
out	*bw*	bandwidth #enum wifi_bw

int (∗get_maxrate)(const char ∗name, unsigned long ∗rate)
Get maximum supported phy rate.

Parameters

`in`	*name*	radio interface name
`out`	*rate*	rate in Mbps

int (∗get_basic_rates)(const char ∗name, int ∗num, uint32_t ∗rates)
Get basic phy rates.

Parameters

`in`	*name*	radio interface name
`out`	*num*	number of elements in rates array
`out`	*rates*	array of rates in Mbps

int (∗get_oper_rates)(const char ∗name, int ∗num, uint32_t ∗rates)
Get operational phy rates.

Parameters

`in`	*name*	radio interface name
`out`	*num*	number of elements in rates array
`out`	*rates*	array of rates in Mbps

int (∗get_supp_rates)(const char ∗name, int ∗num, uint32_t ∗rates)
Get supported phy rates.

Parameters

`in`	*name*	radio interface name
`out`	*num*	number of elements in rates array
`out`	*rates*	array of rates in Mbps

int (∗get_stats)(const char ∗ifname, struct wifi_radio_stats ∗s)
Get radio statistics.

Parameters

`in`	*ifname*	radio interface name
`out`	*s*	radio stats and counters

int (∗scan)(const char ∗name, struct scan_param ∗p)
Trigger scanning.

Parameters

`in`	*name*	radio interface name
`in`	*p*	scan request parameters

int (∗get_scan_results)(const char ∗name, struct wifi_bss ∗bsss, int ∗num)
Get scan results.

Parameters

in	*name*	radio interface name
out	*bsss*	array of scanned APs
out	*num*	number of scanned APs

int (∗get_bss_scan_result)(const char ∗name, uint8_t ∗bssid, struct wifi_bss_detail ∗b)
Get scan result details of a specific AP.

Parameters

in	*name*	radio interface name
in	*bssid*	bssid of a scanned AP
out	*b*	scan result including IE details

int (∗get_noise)(const char ∗ifname, int ∗noise); Get current noise value.

Parameters

in	***name***	**radio interface name**
out	***noise***	**noise value in dBm**

int (∗acs)(const char ∗name, struct acs_param ∗p)
Trigger ACS (auto channel selection).

Parameters

in	***name***	**radio interface name**
in	***p***	**ACS request parameters**

int (∗get_param)(const char ∗name, const char ∗param, int ∗len, void ∗val)
Get radio parameter value(s).

Parameters

in	***name***	**radio interface name**
in	***param***	**radio parameter name**
out	***len***	**length of the returned parameter value**
out	***val***	**parameter value**

int (∗set_param)(const char ∗name, const char ∗param, int len, void ∗val)
Set radio parameter value(s).

Parameters

in	***name***	**radio interface name**
in	***param***	**radio parameter name**

Parameters

`in`	***len***	**length of the parameter**
`in`	***val***	**value of parameter**

int (∗add_iface)(const char ∗name, enum wifi_mode m, char ∗argv[])
Create a WiFi interface on this radio.

Parameters

`in`	***name***	**radio interface name**
`in`	***m***	**wifi mode f.e. WIFI_MODE_AP, WIFI_MODE_STA etc.**
`in`	***argv***	**string arguments array of wifi attributes and values**

int (∗del_iface)(const char ∗name, const char ∗ifname)
Delete a WiFi interface on this radio.

Parameters

`in`	***name***	**radio interface name**
`in`	***ifname***	**wifi interface name to be deleted**

7.46 wifi_radio_stats Struct Reference

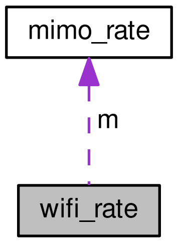
Data Fields

- unsigned long **tx_bytes**
- unsigned long **rx_bytes**
- unsigned long **tx_pkts**
- unsigned long **rx_pkts**
- uint32_t **tx_err_pkts**
- uint32_t **rx_err_pkts**
- uint32_t **tx_dropped_pkts**
- uint32_t **rx_dropped_pkts**
- uint32_t **rx_plcp_err_pkts**
- uint32_t **rx_fcs_err_pkts**
- uint32_t **rx_mac_err_pkts**
- uint32_t **rx_unknown_pkts**
- int **noise**

7.47 wifi_rate Struct Reference

struct wifi_rate - holds rate information

Collaboration diagram for wifi_rate:



Data Fields

- uint32_t rate

    rate in Mbps
- struct mimo_rate m

    of type struct mimo_rate
- enum wifi_phytype phy

    of type struct #wifi_phytype

7.47.1 Detailed Description

struct wifi_rate - holds rate information

7.48 wifi_rsne Struct Reference

Data Fields

- uint16_t wpa_versions

    bitmap of WPA_VERSION∗
- uint32_t group_cipher

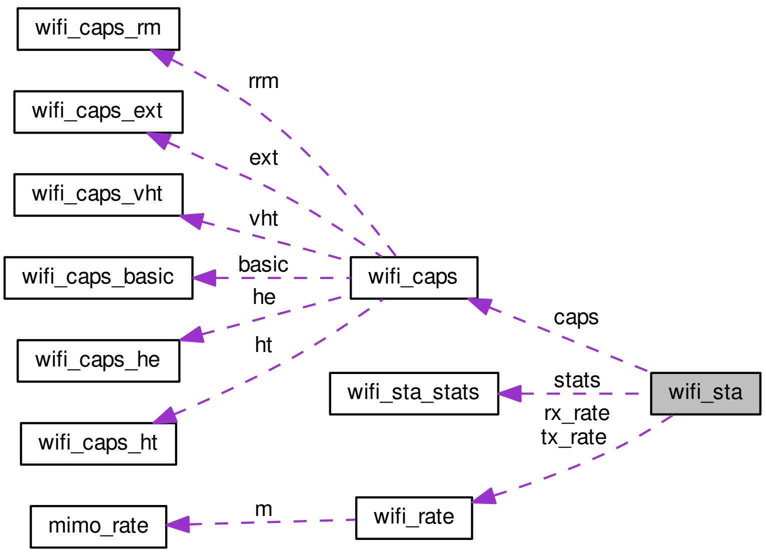
    one of WIFI_CIPHER_∗
- uint32_t pair_ciphers

    bitmap of WIFI_CIPHER_∗
- uint32_t akms

    bitmap of WIFI_AKM_∗
- uint16_t **rsn_caps**

7.49 wifi_sta Struct Reference

Collaboration diagram for wifi_sta:



Data Fields

- uint8_t **macaddr** [6]
- uint8_t sbitmap [4]

  bitmap of enum wifi_statusflags
- uint8_t cbitmap [16]

  bitmap for enum wifi_capflags
- struct wifi_caps caps

  capabilities
- uint8_t oper_std

  bitmap of wifi_std from supp_std
- uint32_t maxrate

  max phy operational rate in Mbps
- struct wifi_rate rx_rate

  latest rate: from AP -> this STA
- struct wifi_rate tx_rate

  latest rate: this STA -> AP
- uint32_t rx_thput

  AP -> this STA instant throughput.
- uint32_t tx_thput

  this STA -> AP instant throughput
- int8_t rssi_avg

  average rssi
- int8_t rssi [WIFI_NUM_ANTENNA]

latest rssi in dBm per-chain

- int8_t noise_avg

  average phy noise in dBm
- int8_t noise [WIFI_NUM_ANTENNA]

  latest noise in dBm
- struct wifi_sta_stats **stats**
- uint64_t tx_airtime

  Tx airtime(msecs) in the last second.
- uint64_t rx_airtime

  Rx airtime(msecs) in the last second.
- int8_t airtime

  airtime in %-age in the last second
- uint32_t conn_time

  time in secs since connected
- uint32_t idle_time

  inactive time in secs

7.50 wifi_sta_stats Struct Reference

Data Fields

- uint64_t **tx_bytes**
- uint64_t **rx_bytes**
- uint32_t **tx_pkts**
- uint32_t **rx_pkts**
- uint32_t **tx_err_pkts**
- uint32_t **tx_rtx_pkts**
- uint32_t **tx_rtx_fail_pkts**
- uint32_t **tx_retry_pkts**
- uint32_t **tx_mretry_pkts**
- uint32_t **tx_fail_pkts**
- uint64_t **rx_fail_pkts**

7.51 wps_device Struct Reference

Data Fields

- char **name** [32]
- char **manufacturer** [64]
- char **modelname** [32]
- char **modelnum** [32]
- char **serialnum** [32]

7.52 wps_param Struct Reference

struct wps_param - WPS parameter to be used during registration : enrollee, registrar or proxy.

Data Fields

- enum wps_role role

  bitmap of wps_role
- enum wps_method method

  bitmap of wps_method
- union {
  unsigned long pin
  pin value for PIN method
  };

7.52.1 Detailed Description

struct wps_param - WPS parameter to be used during registration : enrollee, registrar or proxy.

Bitmap of WPS_ENROLLEE, WPS_REGISTRAR, WPS_PROXY etc. : WPS configuration method, i.e. one of enum wps_method : pin value if wps_method 'PIN' is used for registration

Index