MAC80211 MT76 Programming Guide

Version: **4.10**
Release date: **2025.04.25**

Version History

Version	Date	Author	Description
0.1	2021-06-28	Evelyn Tsai	1. Quick Start for MAC80211 driver
0.2	2021-11-01	Hugo Yang	1. Add WPS section 2. Add IGMP Snooping section
0.3	2022-01-14	Hugo Yang	1. Apply the new confidential format
0.4	2022-01-19	Evelyn Tsai Meichia Chiu	1. Update debug tips
0.5	2022-02-11	Evelyn Tsai Shayne Chen	1. Change Firmware Debugging Method
0.6	2022-04-28	Meichia Chiu	1. Update 6G usage
0.7	2022-04-28	William Cheng	1. Update tips for single SKU settings 2. Add the wpa_supplicant stop/start section
1.0	2022-05-10	Hugo Yang	1. Change the version to 1.0 for formal release
1.1	2022-05-23	Meichia Chiu	1. Update 6G in-band/out-of-band discovery protocol
1.2	2022-05-26	Howard Hsu	1. Update hostapd debug log usage
1.3	2022-07-15	StanleyYP Wang	1. Update the usage of BinFile Mode
1.4	2022-07-18	Evelyn Tsai	1. Update the Zero-Wait DFS usage 2. Update FlowOffload usage
1.5	2022-07-25	Howard Hsu	1. Update EDCCA control usage
1.6	2022-08-22	Tom Liu Yi-Chia Hsieh	1. Add HEMU hostapd_cli usage 2. Add Wi-Fi statistics for the hardware offload path
1.7	2022-08-23	StanleyYP Wang	1. Add normal mode pre-calibration usage 2. Add three wire control (PTA) usage
1.8	2022-10-25	StanleyYP Wang	1. Add the DFS command's usage
1.9	2022-11-22	Howard Hsu	1. Update the EDCCA control usage
1.10	2022-11-28	Lian Chen Hugo Yang	1. Add the HW ATF section
2.0	2022-11-29	Hugo Yang	1. Apply the AIP format for the official release
2.1	2022-12-15	Shayne Chen	1. Update the usage guide for the bin file mode
2.2	2022-12-16	Howard Hsu	1. Add thermal protection usage
2.3	2022-12-22	Peter Chiu	1. Add WMM parameters
2.4	2023-01-11	Peter Chiu	1. Add WA Utilization usage
2.5	2023-03-08	Meichia Chiu	1. Update MU usage 2. Add MU stats usage
2.6	2023-04-27	Rex Lu	1. Add the SingleSku table format
2.7	2023-05-11	Allen Ye	1. Add the Wi-Fi7 SingleSku description 2. Add non-native upstream UCI usage
2.8	2023-06-02	Fancy Liu	1. Add Channel/ACS/Scan section
2.9	2023-06-13	StanleyYP Wang	1. Add related txpower commands 2. Add MT7996 normal mode pre-calibration and EEPROM mode support 3. Add firmware mode for Wi-Fi 7 chips
3.0	2023-06-15	Evelyn Tsai	1. Revise the content of the programming guide
3.1	2023-07-06	Jen-Hao Cheng	1. Add LED handling
3.2	2023-07-13	Howard Hsu Allen Ye	1. Add BF and SR debugfs usage 2. Add content of Chapter 13
3.3	2023-07-14	Rohit Kamat	1. Add the KVR cmd format and uci config

Version	Date	Author	Description
3.4	2023-09-25	Howard Hsu	1. Add SCS and SR debugfs support to Node 2. Update Dump MU Stats
3.5	2024-01-05	Howard Hsu	1. Add Thermal re-calibration
3.6	2024-01-12	Allen Ye Evelyn Tsai	1. Add SingleSKU Power Enhancement 2. Revise the content of the Wi-Fi6/7 support readiness report.
3.7	2024-01-23	StanleyYP Wang	1. Add the efuse dump command.
4.0	2024-04-24	Evelyn Tsai	1. Add Wi-Fi7 MLO usage
4.1	2024-04-25	StanleyYP Wang	1. Update firmware mode for Wi-Fi7 chips (MLO single/multiple Wiphy model) 2. Update the DTS file table for the MT7996/MT7992 variants 3. Update MT7992 pre-calibration info (WIP)
4.2	2024-05-02	Evelyn Tsai	1. Change insmod to modprobe 2. Add OWE Transition mode security 3. Add the Legacy mode setting
4.3	2024-05-15	Meichia Chiu	4. Add EMLSR setting
4.4	2024-05-30	Evelyn Tsai	1. Refactor the wording suggested by the AI editor 2. Add a routed station section
4.5	2024-06-11	StanleyYP Wang	1. Add MT7996 PTA support
4.6	2024-08-06	Allen Ye Benjamin Chui-hao Chiu Evelyn Tsai Meichia Chiu Michael-cy Lee StanleyYP Wang	1. Wi-Fi 7 MLO Support – Add advanced MLO features/usage – Post-Setup Feature (A-T2LM/Reconfiguration) – Add MLO with the MBSS setting – Add MLO Debug/Statistics – Link Specific operation over MLD netdev – Add MLO DFS behavior 2. Security Mode Setting – Add AKM24 usage 3. AFC & Wi-Fi6E Power setting 4. Add QoS Mapping Configuration
4.7	2024-08-15	Evelyn Tsai	1. Formal MLO Beta Release Programming Guide 2. Refactor document wording according to the AI editor
4.8	2024-12-20	Evelyn Tsai Meichia Chiu Michael-cy Lee StanleyYP Wang	1. Modify 8.2.1 Mt76 (Only for Wi-Fi 7) to add mt7996_debug_mask usage 2. Modify 8.11 Logs Collection to add debugfs command and log collection method 3. Modify 12.3 Scan/Survey to update scan command and add scan_dfs_relax command 4. Modify 30.2 Single Wiphy Change to add real single wiphy commands 5. Modify 30.2.1 TX/RX Antenna Settings to add the tx/rx antenna configuration 6. Refactor document wording according to the AI editor
4.9	2025-01-20	StanleyYP Wang Meichia Chiu	1. Modify 13.2 Useful Commands to update new commands and configuration for background radar 2. Add 5.16 Fixed Rate debugfs command 3. Add 30.5.1 Constraints for EHT or MLO 4. Add 30.5.3 RSN Overriding (RSNO)
4.10	2025-04-18	Evelyn Tsai Meichia Chiu StanleyYP Wang	1. Add 11vMBSS maximum number constraints 2. Add 30.6.2 Check station's EML capability 3. Update 14 Multiple BSS and MAC Address Rule 4. Update 18.2 WPS Test Examples 5. Update 30.10.2 11v MBSS

Table of Contents

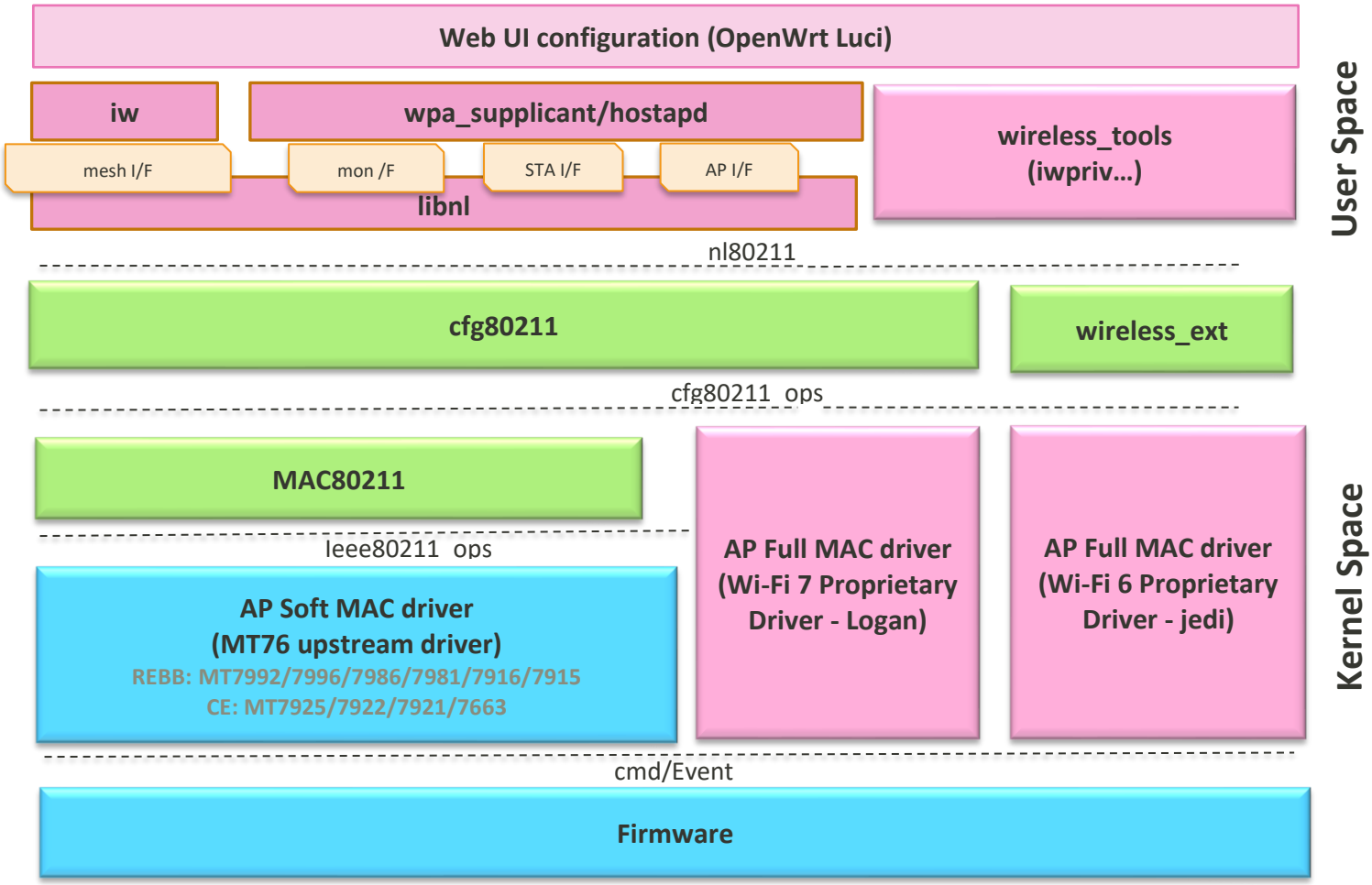
1 Introduction

1.1 Background

The document serves as a software programming guide for the MT76 MAC80211 driver, detailing the configuration of Wi-Fi settings using hostapd and its execution on Openwrt.

The MAC80211 framework is utilized by driver developers to create drivers for SoftMAC wireless devices. Finer control of the hardware and completion of 802.11 frame management in software are enabled by a SoftMAC device. Additionally, it supports both parsing and generation of 802.11 wireless frames, which represent the prevalent type of most devices today.

The cfg80211 callbacks are implemented by MAC80211 for SoftMAC devices. Dependency on cfg80211 is established by MAC80211 for both registration to the networking subsystem and configuration, with configuration being managed by cfg80211 through nl80211 and wireless extensions.



2 Abbreviations

Abbreviation	Full Form
AFC	Automated Frequency Coordination
AP	Access Point
ATF	Air Time Fairness
BSP	Board Support Package
BSS	Basic Service Set
CCK	Complementary Code Keying
CLI	Command Line Interface
CPU	Central Processing Unit
CRDA	Central Regulatory Domain Agent
CSA	Channel Switch Announcement
CTS	Clear to Send
DBDC	Dual-Band Dual-Concurrent
DFS	Dynamic Frequency Selection
DHCP	Dynamic Host Configuration Protocol
DL	Downlink
DMA	Direct Memory Access
DPD	Digital Pre-Distortion
DTIM	Delivery Traffic Information Message
DTS	Device Tree Source
EEPROM	Electrically Erasable Programmable Read-Only Memory
eFuse	Electronic Fuse
EHT	Extremely High Throughput
EMLSR	Enhanced Multi-Link Single Radio
FIIQ	Frequency Independent IQ Imbalance
GHz	Gigahertz
GPIO	General Purpose Inputs-Outputs
ID	Identification
IE	Information Elements
IEEE	Institute of Electrical and Electronics Engineers
IGMP	The Internet Group Management Protocol
IP	Internet Protocol
IPv4	Internet Protocol Version 4
IPv6	Internet Protocol Version 6
LAN	Local Area Network
LED	Light Emitting Diode
LNA	Low Noise Amplifier
MHz	Megahertz
MIMO	Multi-input Multi-output
MLD	Multi-Link Device
MLO	Multi-Link Operations
MWDS	Mixed WDS

Abbreviation	Full Form
OFDM	Orthogonal Frequency Division Multiplexing
OTP	One Time Programmable
OWE	Opportunistic Wireless Encryption
PA	Power Amplifier
PHY	Physical Layer
PIN	Personal Identification Number
PMF	Protected Management Frames
PP	Preamble Puncture
RAM	Random-Access Memory
RSSI	Received Signal Strength Indicator
SAE	Simultaneous Authentication of Equals
SGI	Short Guard Interval
SM	Short Message
SMP	Symmetric Multi-Processing
SR	Spatial Reuse
STA	Station
STBC	Space-Time Block Code
STR	Simultaneous transmit and receive
TCP	Transmission Control Protocol
TKIP	Temporal Key Integrity Protocol
TTLM	TID-To-Link Mapping
TX/RX	Transmit/Receive
UDP	User Datagram Protocol
UL	Uplink
VLAN	Virtual Local Area Network
WDS	Wireless Distribution System
WPS	Wi-Fi Protected Setup

3 Configuration Mode

3.1 /etc/config/wireless

The wireless radio UCI configuration is located in /etc/config/wireless. The wireless function is turned **OFF** by default.

- ***wifi-device***: specifies general radio properties such as channel, driver type, and txpower.
- ***wifi-iface***: defines a wireless network on the top of a Wi-Fi device.

3.1.1 Wi-Fi Device Parameters (Default)

```
config wifi-device 'radio0'
        option type 'mac80211'
        option path '11310000.pcie/pci0001:00/0001:00:00.0/0001:01:00.0'
        option channel '1'
        option band '2g'
        option htmode 'EHT20'
```

3.1.2 Wi-Fi Device Common Options (Upstream)

Name	Type	Required	Default	Description
type	String	Yes	*Autodetected*	The *type* is determined on the first boot during the initial radio device detection (usually not required to be changed). Only support *MAC80211* in the MT76 driver.
phy	String	No/yes	*Autodetected*	Specifies the radio PHY associated with this section If present, it is usually autodetected and should not be changed.
macaddr	MAC address	Yes/No	*Autodetected*	Specifies the radio adapter associated to this section, it is *not* used to change the device MAC but to identify the underlying interface.
disabled	Boolean	No	*0*	Disables the radio adapter if set to *1*. Removing this option or setting it to *0* enables the adapter.
channel	Integer or “auto”	Yes	*Auto*	Specifies the wireless channel. “auto” defaults to the lowest available channel, or utilizes the ACS algorithm depending on hardware/driver support.
channels	List	No	*Regulatory domain specific*	Uses specific channels, when one is in “auto” mode, i.e., to allow hostapd to select one of the provided channels when one should be automatically selected. Channels can be provided in a range using hyphen ('-') or individual channels can be specified by space (' ') among separated values.
hwmode	String	No	*Driver default*	Specifies the hardware mode. The possible values are as follows: • *11b* for 2.4 GHz (used for legacy 11b only) • *11g* for 2.4 GHz (used for 802.11b, 802.11g, and 802.11n) • *11a* for 5 GHz (used for 802.11a, 802.11n, and 802.11ac) Note that 11ng, 11na, 11n, and 11ac are invalid options; this setting is mainly for controlling the frequency band.
htmode	String	No	*Driver default*	Specifies the high throughput mode which is used to control 802.11n (HT), 802.11ac (VHT), and 802.11ax (HE), and 802.11be (EHT). The channel width used for these depends on this configuration. Refer to this article for more details. The Possible values are: *HT20*, *HT40-*, *HT40+*, *HT40*, or *VHT20*, *VHT40*, *VHT80*, *VHT160*, HE20, HE40, HE80, HE160, *NOHT* disables 11n and 11ac. EHT20, EHT40*, EHT160, EHT320*
chanbw	Integer	No	*20*	Specifies a narrow channel width in MHz, possible values are: *5*, *10*, *20*
ht_capab	String	No	*(driver default)*	Specifies the available capabilities of the radio. The values are autodetected. See here for options (check the version of hostapd installed on the router using the “refs” link).

Name	Type	Required	Default	Description
txpower	Integer	No	*(driver default)*	Specifies the *transmission power in dBm* (Deprecated after the Wi-Fi 7 Single Wiphy MLD change; use vif_txpower instead)
rxantenna	Integer	No	*(driver default)*	Specifies the *antenna for receiving*; the value may be driver specific, usually *1* for the first antenna and *2* for the second. Specifying *0* enables automatic selection by the driver if supported. This option has no effect if diversity is enabled.
txantenna	Integer	No	*(driver default)*	Specifies the *antenna for transmitting*; values are identical to *rxantenna.*
country	Varies	No	*(driver default)*	Specifies the country code, which affects the available channels and transmission powers. For type *Broadcom,* a two-letter country code is used (*EN* or *DE*). The *madwifi* driver expects a numeric code, Note: For MediaTek DBDC or TBTC devices, different device options must use the SAME country code. Otherwise, the DFS or channels might be overwritten unexpectedly.
country_ie	Boolean	No	*1:* If the country *is set, 0: Otherwise*	Enables the IEEE 802.11d country IE (information element) advertisement in beacon and probe response frames. This IE contains the country code and channel/power map. Requires *a country*.
noscan	Boolean	No	*0*	Do not scan for overlapping BSSs in HT40+/- mode. Turning this on will violate regulatory requirements!
legacy_rates	Boolean	No	*1 in 19.07, 0 in 21.02*	*0 = Disallow legacy 802.11b data rates, 1 = Allow legacy 802.11b data rates. Legacy or badly behaving devices may require legacy 802.11b rates to interoperate. However, airtime efficiency may be significantly reduced where these are used, so it is recommended to not allow legacy 802.11b rates where possible. The basic_rate and supported_rates options override this option.*
legacy_rates	Boolean	No	*1 in 19.07, 0 in 21.02*	*0 = Disallow legacy 802.11b data rates, 1 = Allow legacy 802.11b data rates. Legacy or badly behaving devices may require legacy 802.11b rates to interoperate, but airtime efficiency may be significantly reduced where these are used. It is recommended to not allow 802.11b rates where possible. The basic_rate and supported_rates options override this option.*
require_mode	String	No	*none*	*Sets the minimum client capability level that connecting clients must support to be allowed to connect. Overrides and sets legacy_rates to 0 to disable legacy 802.11b data rates. Supported values: n = 802.11n, and ac = 802.11ac* *Warning: setting this value to "ac" causes reliability problems from Apple devices, even if they actually support 802.11ac or better.* *For legacy station compatibility, need to set to 1 for carrying '1, 2, 5.5, 11' in supported rate IE*
ht_coex	Integer	No	*0*	*Disable honoring 40 MHz intolerance in coexistence flags of stations. When enabled, the radio will not stop using the 40 MHz channels if the 40 MHz intolerance indication is received from another AP or station.*
log_level	Integer	No	*2*	Set the log_level Supported levels are: • 0 = Verbose Debugging • 1 = Debugging • 2 = Informational messages • 3 = Notification • 4 = Warning
radio	Integer	Yes	*(driver default)*	Specifies the radio index. (This parameter is only supported in versions after OpenWRT 2024/11)

3.1.3 Wi-Fi Device Common Options (MediaTek Vendor-Specific)

Name	Type	Required	Default	Description
mbssid	Boolean	No	*0*	Enable 11vMbssid and ensure that the MAC addresses of multiple access points adhere to the standards.
mu_onoff	Integer	No	*0*	Specifies the uplink or downlink with MIMO or OFDMA (bit map). • 0 = DL OFDMA • 1 = UL OFDMA • 2 = DL MIMO • 3 = UL MIMO
he_twt_responder	Boolean	No	*1*	Enable the TWT feature; by default, it is enabled.
etxbfen	Boolean	No	*1*	Enable explicit beamform; this option will apply other beamform configs.
itxbfen	Boolean	No	*0*	Enable implicit beamform.
sr_enable	Boolean	No	*1*	Enable Spatial Reuse. P.S. This would not affect the hostapd config
sr_enhenced	Boolean	No	*1*	Enable *enhanced* Spatial Reuse (a proprietary feature of MTK). P.S. This would not affect the hostapd config
lpi_enable	Boolean	No	*0*	Enable low power indoors.
beacon_dup	Boolean	No	*1*	Enable beacon duplicate mode
sku_idx	Integer	No	*0*	Specify sku index, used for runtime change single sku table. Default use 0 is to tell driver don't consider sku index and use the first matched country table.

3.1.4 Wi-Fi Interface Parameters (Default)

```
config wifi-iface 'default_radio0'
        option device 'radio0'
        option network 'lan'
        option mode 'ap'
        option ssid 'OpenWrt'
        option encryption 'none'
```

3.1.5 Wi-Fi iFace Common Options (Upstream)

Name	Type	Required	Default	Description
ifname	String	No	*Driver default*	Specifies a custom name for the Wi-Fi interface, which is otherwise automatically named. Maximum length: 15 characters (see Network Basics for more information).
device	String	Yes	*First device ID*	Specifies the used wireless adapter, must refer to one of the defined *wifi-device* sections.
network	String	Yes	*Lan*	Specifies the network interface to attach the wireless network to.
mode	String	Yes	*ap*	Specifies the operation mode of the wireless network interface controller. Possible values are AP, STA, Adhoc, WDS, Monitor, and *Mesh.*
disabled	Boolean	No	*0*	When set to 1, the wireless network is disabled.
ssid	String	Yes	*OpenWrt*	The broadcasted SSID of the wireless network and the managed mode of the SSID of the network to which you are connecting.
bssid	BSSID address	No	*Driver default*	Overrides the BSSID of the network, only applicable in *adhoc* or *sta* mode. In *WDS* mode, it specifies the BSSID of another AP to create a WDS with.
mesh_id	Mesh ID	No	*None*	The Mesh ID as defined in IEEE 802.11s. If set, the wireless interface joins this mesh network when brought up. If not, it is necessary to invoke *iw <iface> mesh join <mesh_id>* to join a mesh after the interface is brought up.
hidden	Boolean	No	*0*	Disables the broadcasting of beacon frames if set to *1* and also hides the ESSID Where the ESSID is hidden, clients may fail to roam, and airtime efficiency may be significantly reduced.

Name	Type	Required	Default	Description
isolate	Boolean	No	*0*	Isolates wireless clients from each other, only applicable in *ap* mode. See this post for details.
doth	Boolean	No	*0*	Enables 802.11h support.
wmm	Boolean	No	*1*	Enables WMM Where Wi-Fi Multimedia (WMM) Mode QoS is disabled, clients may be limited to 802.11a/802.11g rates. Required for 802.11n/802.11ac/802.11ax
encryption	String	No	*None*	Wireless encryption method. Possible values are listed in the encryption values table, right after this table.
key	Integer or string	No	*None*	In any **WPA-PSK** mode, this is a string that specifies the pre-shared passphrase from which the pre-shared key will be derived. The clear text key has to be 8-63 characters long. If a 64-character hexadecimal string is supplied, it is used directly as the pre-shared key instead. In **WEP** mode, this can be an integer specifying which key index to use (*key1*, *key2*, *key3*, or *key4*). Alternatively, it can be a string specifying a passphrase or key directly, as in *key1*. In any **WPA-Enterprise AP** mode, this option has a different interpretation.
key1	String	No	*None*	WEP passphrase or key #1 (selected by the index in *key*). This string is treated as a passphrase from which the WEP key will be derived. If a 10- or 26-character hexadecimal string is supplied, it is used directly as the WEP key instead.
key2	String	No	*None*	WEP passphrase or key #2 (selected by the index in *key*), as in *key1*.
key3	String	No	*None*	WEP passphrase or key #3 (selected by the index in *key*), as in *key1*.
key4	String	No	*None*	WEP passphrase or key #4 (selected by the index in *key*), as in *key1*.
macfilter	String	No	*Disabled*	Specifies the *MAC filter policy*, *disable* to disable the filter, *allow* to treat it as a whitelist, or *deny* to treat it as the blacklist.
maclist	List of MAC addresses	No	*None*	A list of MAC addresses (divided by spaces) to put into the MAC filter.
iapp_interface	String	No	*None*	Specifies a network interface to be used for 802.11f (IAPP) - only enabled when defined.
rsn_preauth	Boolean	No	*0*	Allows pre-authentication for WPA2-EAP networks (and advertises it in WLAN beacons). Only it works if the specified network interface is a bridge.
ieee80211w	Integer	No	*0*	Enables MFP (802.11w) support (0 = disabled, 1 = optional, 2 = required). **Requires the 'full' version of wpad/hostapd and support from the Wi-Fi driver.**
ieee80211w_max_timeout	Integer	No	*Hostapd default*	Specifies the 802.11w Association SA Query maximum timeout
ieee80211w_retry_timeout	Integer	No	*Hostapd default*	Specifies the 802.11w Association SA Query retry timeout
maxassoc	Integer	No	*Hostapd/ driver default*	Specifies the maximum number of clients that can connect.
macaddr	MAC address	No	*Hostapd/ driver default*	This overrides the MAC address used for the Wi-Fi interface **Warning: If the MAC address specified is a multicast address, this override can fail silently. To avoid this problem, ensure that the MAC address specified is a valid unicast MAC address.**
dtim_period	Integer	No	*2 (hostapd default)*	Sets the DTIM (delivery traffic information message) period. There is one DTIM per many-beacon frame. This may be set between 1 and 255. This option only has an effect on *AP* wifi-ifaces.
short_preamble	Boolean	No	*1*	Set the optional use of a short preamble
max_listen_int	Integer	No	*65535 (hostapd default)*	Sets the maximum allowed STA (client) listen interval. Association is refused if an STA attempts to associate with a listen-interval greater than this value. This option only has an effect on *AP* wifi interfaces.
wds	Boolean	No	*0*	Sets 4-address mode.
owe_transition_ssid	String	No	*None*	Opportunistic Wireless Encryption (OWE) transition SSID (only for OPEN and OWE networks).
owe_transition_bssid	BSSID address	No	*None*	Opportunistic Wireless Encryption (OWE) Transition Service Set Identifier (only for OPEN and OWE networks).
sae_pwe	Integer	No	*0 (hostapd default)*	Sets the SAE mechanism for PWE derivation: • 0 = Hunting and pecking only • 1 = hash-to-element only if _ • 2 = both hunting-and-pecking and hash-to-element
start_disabled	Boolean	No	*0*	For an AP, start with beaconing disabled by default (see *start_disabled* in hostapd.conf). Note that if an interface with mode *sta* is also defined on the

Name	Type	Required	Default	Description
				same radio, *start_disabled* is then added in the hostapd configuration, regardless of the value set for the AP.
default_disabled	Boolean	No	*0*	For an STA, add *disabled* to the default wpa_supplicant network block (to prevent it from scanning by default). The network block can still be enabled, for example by using *wpa_cli* (see *disabled* in wpa_supplicant.conf).

3.1.6 Wi-Fi iFace Common Options (MediaTek Vendor-Specific)

Name	Type	Required	Default	Description
beacon_prot	Boolean	No	*0*	Enable beacon protection (management frame protection for Beacon frames). This depends on management frame protection being enabled (iee80211w!= 0)
rssi_reject_assoc_timeout	Integer	No	*30*	The association retry delay in seconds allowed by the STA if the RSSI has not met the threshold (rssi_reject_assoc_rssi and rssi_ignore_probe_request have already in upstream options)
unsol_bcast_probe_resp_interval	Integer	No	*0*	This is for the 6 GHz band only. Unsolicited broadcast probe response transmission settings
fils_discovery_min_interval	Integer	No	*0*	Fast Initial Link Setup (FILS) discovery FILS Discovery frame transmission minimum interval settings
fils_discovery_max_interval	Integer	No	*0*	Fast Initial Link Setup (FILS) discovery FILS Discovery frame transmission maximum interval settings
rnr	Boolean	No	*0*	The Reduced Neighbor Report (RNR) Set it to 1 to enable out-of-band discovery.
sae_groups	Integer	No	*None*	Enable SAE finite cyclic groups
owe_groups	Integer	No	*None*	OWE DH groups
pairwise	String	No	*None*	Set of accepted cipher suites (encryption algorithms) for pairwise keys (unicast packets).
group_cipher	String	No	*None*	Optional override for automatic group cipher selection
vif_txpower	Integer	No	*None*	Specifies the *transmission power* in dBm through interface.
disable_ht	Boolean	No	*0*	For an STA interface, add disable_ht=1 to disable STA's HT capability.
disable_vht	Boolean	No	*0*	For an STA interface, add disable_vht=1 to disable STA's HT capability.
disable_he	Boolean	No	*0*	For an STA interface, add disable_he=1 to disable STA's HT capability.
disable_eht	Boolean	No	*0*	For an STA interface, add disable_eht=1 to disable STA's HT capability.
assoc_phy	Integer	No	*None*	For an STA interface, it is necessary for an MLD Single Wiphy change.

3.1.7 Wi-Fi 7 Multi-Link Device Parameters (MediaTek Vendor-Specific)

Name	Type	Required	Default	Description
Ifname	*String*	Yes	*Driver default*	For **AP MLD** Specifies a custom name for the Wi-Fi interface, which is otherwise automatically named. Maximum length: 15 characters (see Network Basics for more information).
mld_id	Integer	Yes	*0*	For **AP MLD**, MLD Idx (0~16)
mld_addr	String	No	*None*	For **AP MLD**, the MLD Address can be a unique one or the same as one of the link addresses
mld_allowed_links	Integer	No	*0*	For **AP MLD**, *Allowed link bitmap of the AP MLD to which the AP is affiliated* *(Make sure all affiliated APs are set up, and then the management frame is out.)* *For example:* *'1' -> allow 2 GHz* *'2' -> allow 5 GHz* *'3' -> allow 2, 5 GHz* *'4' -> allow 6 GHz* *'5' -> allow 2, 6 GHz*

Name	Type	Required	Default	Description
				'6' -> allow 5, 6 GHz *'7' -> allow 2, 5, 6 GHz*
mld_allowed_phy_bitmap	Integer	Yes	*0*	For **STA MLD** *The bitmap of allowed PHYs that MLD STA can use for connection.* *For example:* *'0' -> legacy STA* *'1' -> allow 2 GHz* *'2' -> allow 5 GHz* *'3' -> allow 2, 5 GHz* *'4' -> allow 6 GHz* *'5' -> allow 2, 6 GHz* *'6' -> allow 5, 6 GHz* *'7' -> allow 2, 5, 6 GHz*
mld_assoc_phy	Integer	Yes	*None*	*For* **STA MLD,** *The PHY number that the MLD STA prefers for connection.* *For example:* *'0' -> phy0,* *'1' -> phy1,* *'2' -> phy2*
encryption & key	string	Yes	*None*	**Both of the AP MLD and STA MLD** Wireless encryption method: Based on the current Hostapd/wpa_supplicant design, the available settings are as follows *MLD STA: Only SAE & OWE mode allowed* *MLD AP:* • *w/o 6GHz: SAE/OWE/WPA2/WPA2&3 Mixed/open/WPA3 transition Mode* • *W 6GHz: SAE/OWE mode*
ssid	string	Yes	*None*	**Both of the AP MLD and STA MLD** The broadcasted SSID of the wireless network and for for managed mode the SSID of the network you're connecting to.
wds	boolean	No	*0*	**Both AP MLD and STA MLD are** This sets the 4-address mode.
eml_disable	boolean	No	*0*	For **AP MLD** '0' -> Enable EMLSR feature '1' -> Disable EMLSR feature
eml_resp	boolean	No	*1*	For **AP MLD** '0' -> AP will not response EML OMN action frame '1' -> AP will response EML OMN action frame

The creation of the Wi-Fi-MLD section on the AP MLD side is essential to guarantee identical SSID and encryption settings for each Multicast Listener Discovery (MLD) device. The script should use the MLD identifier (mld_id) and interface name (ifname) to distinguish among identical AP MLD devices. Moreover, the Wi-Fi interface of MLD groups must be configured with the mld_id parameter. It is crucial to emphasize that Wi-Fi-MLD takes precedence over Wi-Fi interfaces. If there is a conflict in the SSID value between Wi-Fi interfaces and Wi-Fi-MLD, prioritize using the SSID specified in Wi-Fi-MLD.

```
config wifi-mld 'ap_mld_1'
        option ssid 'mt76_mlo'
        option encryption 'sae'
        option key '12345678'
        option ifname 'ap_mld_1'
        option mld_id '1'

config wifi-iface 'default_radio0'
        option device 'radio0'
        option network 'lan'
        option mode 'ap'
        option mld_id '1'

config wifi-iface 'default_radio1'
```

```
        option device 'radio1'
        option network 'lan'
        option mode 'ap'
        option mld_id '1'

config wifi-iface 'default_radio2'
        option device 'radio2'
        option network 'lan'
        option mode 'ap'
        option mld_id '1'
```

Non-AP MLD (STA MLD) side without creating an additional section, you can add the mld_assoc_phy and mld_allowed_phy_bitmap parameters to the existing station's wifi-iface configuration. These parameters will enable MLD support for the STA side without the need for a separate section.

```
config wifi-iface 'wifinet_band0_1'
        option device 'radio0'
        option network 'wwan'
        option ssid 'mt76_mlo'
        option mode 'sta'
        option encryption 'sae'
        option key '12345678'
        option mld_assoc_phy '0'
        option mld_allowed_phy_bitmap '7'
```

3.2 UCI Command Usage

```
# uci
Usage: uci [<options>] <command> [<arguments>]

Commands:
        batch
        export      [<config>]
        import      [<config>]
        changes     [<config>]
        commit      [<config>]
        add         <config> <section-type>
        add_list    <config>.<section>.<option>=<string>
        del_list    <config>.<section>.<option>=<string>
        show        [<config>[.<section>[.<option>]]]
        get         <config>.<section>[.<option>]
        set         <config>.<section>[.<option>]=<value>
        delete      <config>.<section>[.<option>]
        rename      <config>.<section>[.<option>]=<name>
        revert      <config>[.<section>[.<option>]]
        reorder     <config>.<section>=<position>

Options:
        -c <path>  set the search path for config files (default: /etc/config)
        -d <str>   set the delimiter for list values in uci show
        -f <file>  use <file> as input instead of stdin
        -m         when importing, merge data into an existing package
```

```
        -n          name unnamed sections on export (default)
        -N          don't name unnamed sections
        -p <path>   add a search path for config change files
        -P <path>   add a search path for config change files and use as default
        -q          quiet mode (don't print error messages)
        -s          force strict mode (stop on parser errors, default)
        -S          disable strict mode
        -X          do not use extended syntax on 'show'
```

3.2.1 HE80 Access Point Mode

```
uci set wireless.radio1.hwmode=11a
uci set wireless.radio1.country=US
uci set wireless.radio1.channel=36
uci set wireless.radio1.htmode=HE80
uci set wireless.@wifi-iface[1].ssid=OpenWrt
uci set wireless.radio1.disabled=0
uci commit wireless
wifi
```

3.2.2 HE80 Station Mode

Note that MT76 STA's channel and htmode options are not configurable by uci commands. The MT76 STA operates on the same channel as associated AP. A disable_ht (vht/he/eht) option can be used to disable specific PHY mode.

```
uci set wireless.radio1.hwmode=11a
uci set wireless.radio1.country=US
uci set wireless.@wifi-iface[1].device="radio1"
uci set wireless.@wifi-iface[1].network="wwan"
uci set wireless.@wifi-iface[1].mode="sta"
uci set wireless.@wifi-iface[1].encryption="none"
uci set wireless.@wifi-iface[1].ssid="MTK_HARRIER_AP_5G"
uci set wireless.@wifi-iface[1].disable_eht=1
uci set wireless.radio1.disabled=0
uci set network.wwan="interface"
uci set network.wwan.proto="dhcp"
uci commit wireless
wifi
```

3.2.3 Wi-Fi 7 Multi-Link AP Mode

```
#Add mld_id AP interface
uci set wireless.@wifi-iface[0].mld_id=1

#Add mld_id AP interface
uci set wireless.@wifi-iface[1].mld_id=1

#Add mld_id AP interface
uci set wireless.@wifi-iface[2].mld_id=1
```

```
#Add wifi-mld section
uci set wireless.ap_mld_1=wifi-mld
uci set wireless.ap_mld_1.ssid=MT7996_MLD_AP
uci set wireless.ap_mld_1.encryption=sae
uci set wireless.ap_mld_1.key=12345678
uci set wireless.ap_mld_1.mld_id=1
uci set wireless.ap_mld_1.ifname=ap-mld-1

#Save and reload
uci commit wireless
wifi
```

3.2.4 Wi-Fi 7 Multi-Link STA Mode

```
uci set wireless.@wifi-iface[0].device=radio0
uci set wireless.@wifi-iface[0].network=wwan
uci set wireless.@wifi-iface[0].ssid=MT7996_MLD_AP
uci set wireless.@wifi-iface[0].mode=sta
uci set wireless.@wifi-iface[0].encryption=sae
uci set wireless.@wifi-iface[0].key=12345678
uci set wireless.@wifi-iface[0].assoc_phy=1
uci set wireless.@wifi-iface[0].mld_allowed_phy_bitmap=7
uci set wireless.@wifi-iface[1].disabled=1
uci set wireless.@wifi-iface[2].disabled=1
uci set network.wwan="interface"
uci set network.wwan.proto="dhcp"
uci commit wireless
wifi
```

3.3 Hostapd Configuration

Mac80211.sh under *package/kernel/mac80211/files/lib/netifd/wireless/mac80211.sh* converts */etc/config/wireless* into the hostapd configuration. The hostapd is configured by a text file that lists all the configuration parameters. See an example configuration file, hostapd.conf, for detailed information about the configuration format and supported fields.

3.3.1 Bypass netifd Conversion

Some parameters not handled in netifd/wireless/mac80211.sh can be bypassed by the following ways.

- Stop the hostapd process

Connac2 Wi-Fi 6	`ubus call hostapd config_remove '{"iface": "phy0-ap0"}'`
	`ubus call hostapd config_remove '{"iface": "phy1-ap0"}'`
	`ubus call hostapd config_remove '{"iface": "phy2-ap0"}'`
	`iw dev phy0-ap0 del`

Connac3 Wi-Fi 7	`ubus call hostapd config_set '{"phy": "phy0", "radio": 0, "config": "", "prev_config": ""}'`

	ubus call hostapd config_set '{"phy": "phy0", "radio": 1, "config": "", "prev_config": ""}' ubus call hostapd config_set '{"phy": "phy0", "radio": 2, "config": "", "prev_config": ""}'

- Modify the hostapd configuration file

Connac2 Wi-Fi 6	vi /var/run/hostapd-phy0.conf vi /var/run/hostapd-phy1.conf vi /var/run/hostapd-phy2.conf
Connac3 Wi-Fi 7	vi /var/run/hostapd-phy0.0.conf vi /var/run/hostapd-phy0.1.conf vi /var/run/hostapd-phy0.2.conf

- Start the hostapd process

Connac2 Wi-Fi 6	ubus call hostapd config_add '{"iface": "phy0-ap0", "config": "/var/run/hostapd-phy0.conf"}' ubus call hostapd config_add '{"iface": "phy1-ap0", "config": "/var/run/hostapd-phy1.conf"}' ubus call hostapd config_add '{"iface": "phy2-ap0", "config": "/var/run/hostapd-phy2.conf"}'
Connac3 Wi-Fi 7	ubus call hostapd config_set '{"phy": "phy0", "radio": 0, "config": "/var/run/hostapd-phy0.0.conf", "prev_config": ""}' ubus call hostapd config_set '{"phy": "phy0", "radio": 1, "config": "/var/run/hostapd-phy0.1.conf", "prev_config": ""}' ubus call hostapd config_set '{"phy": "phy0", "radio": 2, "config": "/var/run/hostapd-phy0.2.conf", "prev_config": ""}'

- Stop the wpa_supplicant process

Connac2 Wi-Fi 6	ubus call wpa_supplicant config_remove '{"iface": "phy0-sta0"}' iw dev phy0-sta0 del
Connac3 Wi-Fi 7	ubus call wpa_supplicant config_set '{"phy": "phy0", "radio": -1, "config": [] }

- Modify the wpa_supplicant configuration

Connac2 Wi-Fi 6	vi /var/run/wpa_supplicant-phy0-sta0.conf
Connac3 Wi-Fi 7	vi /var/run/wpa_supplicant-phy0-sta0.conf

- Start wpa_supplicant process

Connac2 Wi-Fi 6	ubus call wpa_supplicant config_add '{"driver": "nl80211", "iface": "wlan0", "ctrl": "/var/run/wpa_supplicant", "config": "/var/run/wpa_supplicant-wlan0.conf"}'
Connac3 Wi-Fi 7	ubus call wpa_supplicant config_set '{"config": [{ "ctrl": "\/var\/run\/wpa_supplicant", "iface": "<ifname>", "mode": "sta", "config": "\/var\/run\/wpa_supplicant-phy0-sta0.conf", "macaddr": "<macaddress>", "bridge": "<bridge name>", "4addr": <true|false>, "powersave": <true|false> }]"phy":"phy0", "radio": -1, "num_global_macaddr": "<int value>", "defer": "<0|1>"}

3.4 Hostapd Legacy Operations - Per BSS/Interface Down/up (Wi-Fi 6 and Wi-Fi 7)

Using the hostapd_cli command is strongly recommended over ifconfig down/up to maintain synchronization in the hostapd finite state machine as per the original design of hostapd.

BSS operations

- Remove BSS

```
hostapd_cli -i global raw REMOVE_BSS <interface name>
```

  Then modify BSS configuration as you want (ex: SSID.)

- Add BSS

```
hostapd_cli -i global raw ADD bss_config=phyX:<hostapd_config path>
```

- Enable BSS

```
hostapd_cli -i <interface name> enable_bss
```

- Disable BSS

  (note that it is not allowed to disable the last enabled BSS. Instead, disabling the whole PHY should be used)

```
hostapd_cli -i <interface name> disable_bss
```

3.5 Hostapd MLD Operations – Remove/Add and Down/up (Wi-Fi 7 Only)

- Remove AP MLD

```
hostapd_cli -i global raw REMOVE_MLD <interface name>
```

  Then modify BSS configuration as you want (ex: SSID.)

- Add a single link

```
hostapd_cli -i global raw ADD bss_config=phyX:<hostapd_config path>
```

- Enable AP MLD

```
hostapd_cli -i <interface name> enable_mld
```

- Disable AP MLD

  (note that it is not allowed to disable the last enabled AP MLD. Instead, disabling the whole physicist should be used)

```
hostapd_cli -i <interface name> disable_mld
```

4 iw - Introduction and Applications

iw is a new nl80211 based CLI configuration utility for wireless devices which supports all new drivers that have been added to the kernel recently. Note that the old tool *iwconfig* that uses the Wireless Extensions interface has been deprecated.

4.1 Get Device Capabilities

Use the followings to get device capabilities for all devices, such as band information (2.4 GHz, and 5 GHz) and 802.11n information.

```
iw list
```

4.2 Show Information for All interfaces

```
iwinfo
```

4.3 Scanning (Station Mode)

```
iw dev <devname> scan
```

4.4 Listen to Events

```
iw event
```

4.5 Get Link Status

To determine whether the device is connected to an AP or not and what the last TX rate used is, use the command below.

- Example output when associated to a legacy (non-802.11n) AP:

```
iw dev <devname> link
Connected to 04:21:b0:e8:c8:8b (on wlan0)
        SSID: attwifi
        freq: 2437
        RX: 2272 bytes (18 packets)
        TX: 232 bytes (3 packets)
        signal: -57 dBm
        tx bitrate: 36.0 MBit/s
```

- Example output when associated with an 802.11n AP:

```
iw dev <devname> link
Connected to 68:7f:74:3b:b0:01 (on wlan0)
        SSID: tesla-5g-bcm
        freq: 5745
        RX: 30206 bytes (201 packets)
        TX: 4084 bytes (23 packets)
        signal: -31 dBm
        tx bitrate: 300.0 MBit/s MCS 15 40Mhz short GI
```

4.6 Get Station Statistics

To get the station statistics information, such as the amount of TX/RX bytes, the last TX bitrate (including MCS rate), what to do is as follows.

```
$ iw dev <devname> station dump
Station 12:34:56:78:9a:bc (on wlan0)
        inactive time:  304 ms
        rx bytes:       18816
        rx packets:     75
        tx bytes:       5386
        tx packets:     21
        signal:         -29 dBm
        tx bitrate:     54.0 MBit/s
```

4.7 Get Station Statistics Against a Peer

The following command should be used to obtain specific statistics against a peer that is being communicated with by a station.

In the case below, the <peer-MAC-address> would be the MAC address of the AP.

```
$ iw dev <devname> station get <peer-MAC-address>
```

4.8 Modify Transmit Bitrates

Modifying TX bitrates is supported by iw, allowing adjustments to both legacy and HT Modulation and Coding Scheme (MCS) rates through masking of the permitted bitrates. Users are also able to clear the mask.

※ Currently, bitrates do not support EHT mode adjustment. For commands related to EHT mode, please refer to Chapter 5.16 Fixed Rate.

```
● Connac2 Wi-Fi6/Connac3 Non-MLO
$ iw dev <devname> set bitrates [options]-[band] [value]

● Connac3 Wi-Fi 7 MLO
$ iw dev <devname> set bitrates -l [link] [options]-[band] [value]

[legacy-<2.4|5> <legacy rate in Mbps>*]
[ht-mcs-<2.4|5> <MCS index>*]
[vht-mcs-<2.4|5> [he-mcs-<2.4|5|6> <NSS:MCSx,MCSy... | NSS:MCSx-MCSy>*]
[sgi-2.4|lgi-2.4]
[sgi-5|lgi-5]
[he-gi-<2.4|5|6> <0.8|1.6|3.2>]
[he-ltf-<2.4|5|6> <1|2|4>]
```

- Fix he-ltf and he-gi to 2x LTF (6.4us), 0.8us GI

Connac2 Wi-Fi 6	`iw dev phy0-ap0 set bitrates he-ltf-5 2 he-gi-5 0.8`
Connac3 Wi-Fi 7 MLO	`iw dev ap-mld-1 -l 1 set bitrates he-ltf-5 2 he-gi-5 0.8`

- Fix he-mcs to 2SS MCS 7

Connac2 Wi-Fi 6	`iw dev phy0-ap0 set bitrates he-mcs-5 2:7`
Connac3 Wi-Fi 7 MLO	`iw dev ap-mld-1 -l 1 set bitrates he-mcs-5 2:7`

- Fix SGI

Connac2 Wi-Fi 6	`iw dev phy0-ap0 set bitrates sgi-5`
Connac3 Wi-Fi 7 MLO	`iw dev ap-mld-1 -l 1 set bitrates sgi-5`

- Reset to default

Connac2 Wi-Fi 6	`iw dev phy0-ap0 set bitrates`
Connac3 Wi-Fi 7 MLO	`iw dev ap-mld-1 -l 1 set bitrates`

5 Debugfs Node

The debug commands show under /sys/kernel/debug/ieee80211/phyX/mt76.

Parameter	Information
mib_infoX	Check the phy layer state
trinfo	DMA Tx/Rx Ring Information
txpower_sku	Dump all rate's Tx Power
txpower_level	Set the power level to drop.
txpower_path	Dump the txpower path table
txpower_info	Retrieve the basic TX power settings
tx_stats	Dump tx statistics
wtbl_info/ uwtbl_info	You need to assign a WLAN index to wlan_idx and use wtbl_info to display the WLAN entry. #echo x > wlan_idx #cat wtbl_info #cat uwtbl_info
ple_info	Dumps the PLE/PSE buffer and queue structure information.
pse_info	Dumps the PLE/PSE buffer and queue structure information.
regidx	You need to assign a register index to regidx and cat regval to show the CR value. #echo x > regidx #cat regval
muru_stats	Dump the mu/su ratio. Please enable muru_debug before using this command
agg_infoX	Dump information about the aggregation of MPDUs at the MAC layer.
amsdu_info	Dump information about the aggregation of MSDUs into the MAC layer.
per	Show the MPDU-based TX packet error rate of a station
fw_version	Show the fw_version and A-die information and also the FEM type
sta_info	Dump all stations' MAC address and corresponding Wcid index
sys_recovery	Collect SER state information
token	Show per band software-path packet token
rx_drop_stats	Rx Drop Count
tx_drop_stats	Tx Drop Count

5.1 Dump TX Statistics

```
#cd /sys/kernel/debug/ieee80211/phyX/mt76(phyX is your wifi phy device)
#cat tx_stats
```

```
Example: Get TX Statistics of phy1
#cd /sys/kernel/debug/ieee80211/phy1/mt76
#cat tx_stats
Phy 0, Phy band 0
Length:         1 |   2 - 10 |  11 - 19 |  20 - 28 |  29 - 37 |  38 - 46 |  47 - 55 |  56 -
79 |  80 -103 | 104 -127 | 128 -151 | 152 -175 | 176 -199 | 200 -223 | 224 -247 |
Count:          0 |        0 |        0 |        0 |        0 |        0 |        0 |
0 |        0 |        0 |        0 |        0 |        0 |        0 |        0 |
BA miss count: 0

Tx Beamformer applied PPDU counts: iBF: 0, eBF: 0
Tx Beamformer Rx feedback statistics: All: 0, HE: 0, VHT: 0, HT: 0, BW20, NC: 0, NR: 0
Tx Beamformee successful feedback frames: 0
Tx Beamformee feedback triggered counts: 0
Tx multi-user Beamforming counts: 0
Tx multi-user MPDU counts: 0
Tx multi-user successful MPDU counts: 0
Tx single-user successful MPDU counts: 8093

Tx MSDU statistics:
```

```
AMSDU pack count of 1 MSDU in TXD:       986 (100%)
AMSDU pack count of 2 MSDU in TXD:         0 (  0%)
AMSDU pack count of 3 MSDU in TXD:         0 (  0%)
AMSDU pack count of 4 MSDU in TXD:         0 (  0%)
AMSDU pack count of 5 MSDU in TXD:         0 (  0%)
AMSDU pack count of 6 MSDU in TXD:         0 (  0%)
AMSDU pack count of 7 MSDU in TXD:         0 (  0%)
AMSDU pack count of 8 MSDU in TXD:         0 (  0%)
```

5.2 Dump BF Stats

```
#cd /sys/kernel/debug/ieee80211/phyX/mt76(phyX is your wifi phy device)
#cat tx_stats
```

```
Example: Get TX Statistics of phy1
#cd /sys/kernel/debug/ieee80211/phy1/mt76
#cat tx_stats
Phy phy1, Phy band 1
Length:         1 |   2 - 10 |  11 - 19 |  20 - 28 |  29 - 37 |  38 - 46 |  47 - 55 |  56 -
79 |  80 -103 | 104 -127 | 128 -151 | 152 -175 | 176 -199 | 200 -223 | 224 -247 |
Count:       7542 |        0 |        0 |        0 |        0 |        0 |        0 |
0 |        0 |        0 |        0 |        0 |        0 |        0 |        0 |
BA miss count: 447
Tx attempts:    29208 (MPDUs)
Tx success:     27007 (MPDUs)
Tx PER: 8%
Tx RED drop:         0

Tx Beamformer applied PPDU counts: iBF: 0, eBF: 8594
Tx Beamformer Rx feedback statistics: All: 82, EHT: 0, HE: 82, VHT: 0, HT: 0, BW80, NC:
510, NR: 1213
Tx Beamformee successful feedback frames: 0
Tx Beamformee feedback triggered counts: 0
Tx multi-user Beamforming counts: 0
Tx multi-user MPDU counts: 0
Tx multi-user successful MPDU counts: 0
Tx single-user successful MPDU counts: 30056

Tx MSDU statistics:
AMSDU pack count of 1 MSDU in TXD:      2534 ( 13%)
AMSDU pack count of 2 MSDU in TXD:       989 (  5%)
AMSDU pack count of 3 MSDU in TXD:      1537 (  8%)
AMSDU pack count of 4 MSDU in TXD:       821 (  4%)
AMSDU pack count of 5 MSDU in TXD:       627 (  3%)
AMSDU pack count of 6 MSDU in TXD:       650 (  3%)
AMSDU pack count of 7 MSDU in TXD:     11967 ( 62%)
AMSDU pack count of 8 MSDU in TXD:         1 (  0%)
```

To verify if the beamform feature is functioning properly, two fields can be checked: Tx Beamformer applied PPDU counts and Tx Beamformer Rx feedback statistics. The former indicates the number of PPDU counts that have been applied to the beamform matrix, allowing AP to transmit data frames with beamforming applied if the feature works correctly with associated stations. The latter shows how many sounding feedback frames have been received by AP from beamformee; if no sounding feedback has been received, AP cannot calculate the correct beamform matrix, resulting in no beamforming gain.

5.3 Dump Tx/Rx Drop Counts

Tx

```
root@OpenWrt:/# cat /sys/kernel/debug/ieee80211/phy0/mt76/tx_drop_stats
                                        dev       Band0       Band1       Band2
total
Receive from mac80211                5040348         196         261         129
5040934
Send to hw                                 0     3480247     1424062      136625
5040934

Drop due to IN_TESTMODE                    0           0           0           0
0
Drop due to WCID_NOT_INIT                  0           0           0           0
0
Drop due to STOPPED_QUEUE                  0           0           0           0
0
Drop due to RESET_STATE                    0           0           0           0
0
Drop due to GET_TXWI_FAIL                  0           0           0           0
0
Drop due to DMA_FAIL                       0           0           0           0
0
Drop due to AGG_EXCEEDED                   0           0           0           0
0
Drop due to RING_FULL                      0           0           0           0
0
Drop due to INVALID_SKB                    0           0           0           0
0
Drop due to GET_TOKEN_FAIL                 0           0           0           0
0
Drop due to ADDR_TRANS_FAIL                0           0           0           0
0
Drop due to INVALID_WCID                   0           0           0           0
0
Drop due to INVALID_LINK                   0           0           0           0
0
```

Rx

```
root@OpenWrt:/# cat /sys/kernel/debug/ieee80211/phy0/mt76/rx_drop_stats
                                       RXQ4        RXQ5
Drop due to DMAD_RRO_REPEAT              25           0
Drop due to DMAD_RRO_OLDPKT            7334           0
Drop due to DMAD_RRO_PN_CHK_FAIL          0           0
Drop due to DMAD_WO_FRAG                  0           0
Drop due to DMAD_WO_DROP                  0           0
Drop due to DMAD_ADDR_NOT_FOUND           0           0
Drop due to DMAD_TOKEN_NOT_FOUND          0           0
Drop due to DMAD_GET_TOKEN_FAIL           0           0
Drop due to DMAD_GET_RXWI_FAIL            0           0
Drop due to DMAD_NOMEM                    0           0
Drop due to DMAD_DMA_MAPPING_FAIL         0           0
Drop due to FRAG                          0           0
Drop due to BUILD_SKB_FAIL                0           0

                                       Band0       Band1       Band2
Receive from hw                        91232       83747      525134
Send to mac80211                       91232       83747      525134

Drop due to RXD_ERR                       0           0           0
Drop due to STATE_ERR                     0           0           0
Drop due to RFC_PKT                       0           0           0
Drop due to AGG_SN_LESS                   0           0           0
```

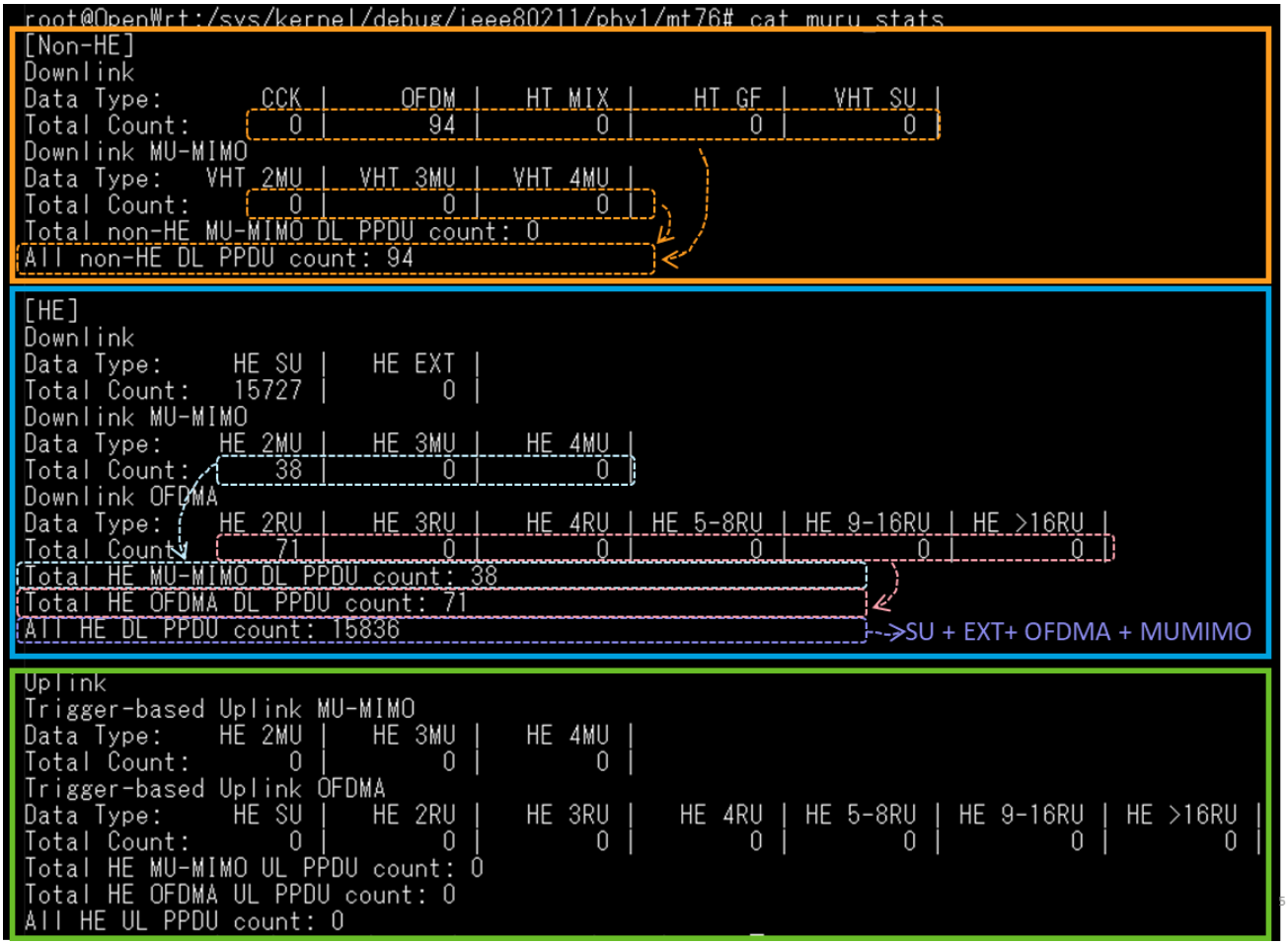
	Drop due to AGG_DUP	0	0	0

5.4 Dump Software Token

show per band token count	root@OpenWrt:/# cat /sys/kernel/debug/ieee80211/phy0/mt76/token Cut through token: 0 (token pending xxx ms) 1 (token pending xxx ms) 2 (token pending xxx ms) 3 (token pending xxx ms) 4 (token pending xxx ms) 5 (token pending xxx ms) 6 (token pending xxx ms) 7 (token pending xxx ms) 8 (token pending xxx ms) 9 (token pending xxx ms) 10 (token pending xxx ms) Band0 consume: xx, free: xx total: xx Band1 consume: xx, free: xx total: xx
Dump Specific Token	#echo <token_idx> > /sys/kernel/debug/ieee80211/phy0/mt76/token_idx #cat /sys/kernel/debug/ieee80211/phy0/mt76/token_txd

5.5 Dump MU Stats

```
#cd /sys/kernel/debug/ieee80211/phyX/mt76(phyX is your wifi phy device)
#echo 1 > muru_debug
#cat muru_stats
```



Note that this method is specifically applicable only to Wi Fi 6 BSP due to firmware design.

5.6 Dump TWT Status

```
#cd /sys/kernel/debug/ieee80211/phyX/mt76 (phyX is your wifi phy device)
#cat twt_stats
```

```
root@OpenWrt:/sys/kernel/debug/ieee80211/phy0/mt76# cat twt_stats
     wcid |       id |    flags |      exp | mantissa | duration |          tsf |
        1 |        0 |     s--- |       10 |     2048 |      255 |   1159725056 |
```

5.7 Enable/Disable Smart Carrier Sense

Smart Carrier Sense (SCS) is a feature that improves the channel access method.

In traditional carrier sense multiple access (CSMA) protocols, such as CSMA/CA used in Wi-Fi, a device listens for a clear channel before transmitting data. However, if multiple devices are trying to access the channel at the same time, collisions can occur, resulting in lost packets and lower network efficiency.

The use of SCS enhances the efficiency of the channel access method, decreasing collision risks and improving overall network performance. This is especially vital in densely populated wireless settings such as airports, stadiums, and conference centers, where multiple devices compete for the same restricted bandwidth.

By default, this feature is **enabled**. If you want to enable or disable this feature, please use the following command.

```
#cd /sys/kernel/debug/ieee80211/phyX/mt76(phyX is your wifi phy device)
#echo <val> > scs_enable
# 0: Disable SCS feature
# 1: Enable SCS feature
```

5.8 Enable/Disable Spatial Reuse feature

Spectrum utilization is maximized by Spatial Reuse (SR) through the utilization of Space Division Multiple Access (SDMA) technology. This technology enables simultaneous communication among multiple devices, thereby increasing both network throughput and capacity. Spatial Reuse strategically assigns devices to distinct areas to prevent interference, thereby enhancing overall system performance. The IEEE incorporates Spatial Reuse into systems to optimize transmissions in congested settings such as airports and malls.

By default, this feature is **enabled**. If you want to enable or disable this feature, please use the following command.

```
#cd /sys/kernel/debug/ieee80211/phyX/mt76(phyX is your wifi phy device)
#echo <val> > sr_enable
# 0: Disable SR feature
# 1: Enable SR feature
```

5.9 Enable/Disable Enhanced Spatial Reuse Feature

Enhanced Spatial Reuse is an exclusive feature developed by MTK. It enables the adaptation of Spatial Reuse in the presence of far-range VHT/HT interference, effectively overcoming the challenge of using Spatial Reuse with legacy traffic.

By default, this feature is **enabled**. To enable or disable this feature, the following command should be used:

```
#cd /sys/kernel/debug/ieee80211/phyX/mt76(phyX is your wifi phy device)
#echo <val> > sr_enhanced_enable
# 0: Disable enhance SR feature
# 1: Enable enhanced SR feature
```

5.10 Dump Spatial Reuse Stats

```
#cd /sys/kernel/debug/ieee80211/phyX/mt76(phyX is your wifi phy device)
#cat sr_stats
```

```
Example: Get TX Statistics of phy1
#cd /sys/kernel/debug/ieee80211/phy1/mt76
#cat sr_stats
[ 2278.829483] mt7996e 0000:01:00.0: Inter PPDU Count = 28352
[ 2278.834970] mt7996e 0000:01:00.0: SR Valid Count = 83
[ 2278.840015] mt7996e 0000:01:00.0: SR Tx Count = 53
[ 2278.846174] mt7996e 0000:01:00.0: SR Tx Acked Count = 53
```

This command shows 4 spatial reuse feature tx statistics.

1. Inter PPDU count: The number of PPDUs sent from one BSS to another.
2. SR Valid Tx Count: The number of PPDUs that meet the criteria for an AP to reuse.
3. SR Tx count: The number of AP SR Tx PPDUs transmitted.
4. SR Tx acked count: The number of acknowledgments that an AP receives for its SR Tx PPDU.

5.11 Dump Information About Aggregation of MPDUs

```
# cd /sys/kernel/debug/ieee80211/phy0/mt76
# cat agg_infoX (X is the index of band of interest.)
```

```
Example: Get the aggregation information of band 1.
# cd /sys/kernel/debug/ieee80211/phy0/mt76
# cat agg_info1
Band 1 AGG Status
================================
AC00 Agg limit = 0      AC01 Agg limit = 0      AC02 Agg limit = 0      AC03 Agg
limit = 0      AC10 Agg limit = 0      AC11 Agg limit = 0      AC12 Agg limit = 0
AC13 Agg limit = 0      AC20 Agg limit = 0      AC21 Agg limit = 0      AC22 Agg
limit = 0      AC23 Agg limit = 0      AC30 Agg limit = 0      AC31 Agg limit = 0
AC32 Agg limit = 0      AC33 Agg limit = 0      ===AMPDU Related Counters===
Tx Agg Range: 1           2~10        11~19       20~28
 Burst count: 0x0         0x0         0x0         0x1
 Burst ratio: (0%)        (0%)        (0%)        (2%)
  MDPU ratio: (0%)        (0%)        (0%)        (0%)
              29~37       38~46       47~55       56~79
              0xf         0x0         0x0         0x0
              (32%)       (0%)        (0%)        (0%)
              (8%)        (0%)        (0%)        (0%)
              80~103      104~127     128~151     152~175
              0x0         0x0         0x0         0x6
              (0%)        (0%)        (0%)        (13%)
              (0%)        (0%)        (0%)        (15%)
              176~199     200~223     224~247     248~1024
              0x12        0x6         0x0         0x0
              (39%)       (13%)       (0%)        (0%)
              (54%)       (20%)       (0%)        (0%)
```

The dumped information is explained below:

- ***Agg limit***: the number of MPDUs allowed to be aggregated in an A-MPDU. If the value is 0, no limit is set, but the hardware limitation is 1024.
- ***Burst count***: the number of A-MPDUs that contain many MPDUs specified by ***the Tx Agg Range***.

5.12 Dump Information About Aggregation of MSDUs

```
# cd /sys/kernel/debug/ieee80211/phy0/mt76
# cat amsdu_info
```

```
Example: Get the MSDU aggregation information.
# cd /sys/kernel/debug/ieee80211/phy0/mt76
# cat amsdu_info
```

```
HW A-MSDU Information:
# of HW A-MSDU containing 1 MSDU: 0x1   (0.1%)
# of HW A-MSDU containing 2 MSDU: 0x23a (99.8%)
# of HW A-MSDU containing 3 MSDU: 0x0   (0.0%)
# of HW A-MSDU containing 4 MSDU: 0x0   (0.0%)
# of HW A-MSDU containing 5 MSDU: 0x0   (0.0%)
# of HW A-MSDU containing 6 MSDU: 0x0   (0.0%)
# of HW A-MSDU containing 7 MSDU: 0x0   (0.0%)
# of HW A-MSDU containing 8 MSDU: 0x0   (0.0%)
PAO A-MSDU Information:
# of PAO A-MSDU containing  1 MSDU: 0x0 (0.0%)
# of PAO A-MSDU containing  2 MSDU: 0x0 (0.0%)
# of PAO A-MSDU containing  3 MSDU: 0x0 (0.0%)
# of PAO A-MSDU containing  4 MSDU: 0x0 (0.0%)
# of PAO A-MSDU containing  5 MSDU: 0x0 (0.0%)
# of PAO A-MSDU containing  6 MSDU: 0x0 (0.0%)
# of PAO A-MSDU containing  7 MSDU: 0x0 (0.0%)
# of PAO A-MSDU containing  8 MSDU: 0x0 (0.0%)
# of PAO A-MSDU containing  9 MSDU: 0x0 (0.0%)
# of PAO A-MSDU containing 10 MSDU: 0x0 (0.0%)
# of PAO A-MSDU containing 11 MSDU: 0x0 (0.0%)
# of PAO A-MSDU containing 12 MSDU: 0x0 (0.0%)
# of PAO A-MSDU containing 13 MSDU: 0x0 (0.0%)
```

5.13 Show MPDU-Based TX Packet Error Rate (PER)

```
# cd /sys/kernel/debug/ieee80211/phy0/mt76
# echo X > wlan_idx (X is the WCID of the station of interest.)
# cat per
```

```
Example 1: Get the PER of the station with WCID 1.
# cd /sys/kernel/debug/ieee80211/phy0/mt76
# echo 1 > wlan_idx
# cat per
WCID 1     TxTotalMpduCount: 4681284     TxFailedMpduCount: 275469     PER: 5.8%

Example 2: Reset the statistical counters, and get the PER during a period of
interest.
# cd /sys/kernel/debug/ieee80211/phy0/mt76
# echo 1 > wlan_idx
# echo 1 > reset_counter
(Wait for a period as one wishes.)
# cat per
WCID 1     TxTotalMpduCount: 43559     TxFailedMpduCount: 2400     PER: 5.5%
```

The statistical counters ***TxTotalMpduCount*** and ***TxFailedMpduCount*** accumulate values without resetting after reading. The reset_counter DebugFS node must be used to reset the counters (see Example 2).

5.14 Trigger Thermal Re-Calibration

To manually trigger thermal re-calibration, use the following commands:

```
# echo 1 > /sys/kernel/debug/ieee80211/phy0/mt76/thermal_recal
# echo 2 > /sys/kernel/debug/ieee80211/phy0/mt76/thermal_recal
```

To disable thermal re-calibration, the following command should be used:

```
# echo 0 > /sys/kernel/debug/ieee80211/phy0/mt76/thermal_recal
```

5.15 Show Firmware Version

Version: [Wi-Fi Generation].[Wi-Fi SKU].[Openwrt MT76 Trunk Version (Year)].[Openwrt MT76 Trunk Version (Version)]

```
# cat /sys/kernel/debug/ieee80211/phy0/mt76/fw_version
Version: 4.3.24.3
Rom Patch Build Time: 20240420003752a

WM Patch Build Time: 20240420003747, Mode: Normal mode
WA Patch Build Time: 20240420003712
DSP Patch Build Time: 20240420003645
Adie 0: ID = 0x7976, Ver = 0x8a20
Adie 1: ID = 0x7977, Ver = 0x8a10
Adie 2: ID = 0x7977, Ver = 0x8a10
FEM type: eFEM
```

Version

Wi-Fi Generation		Wi-Fi SKUs		OpenWRT MT76 Trunk	
Wi-Fi6	2	Connac2	2	2024-03-18	24.02
Wi-Fi7 non-MLO	3	Connac3	3	2024-04-04	24.03
Wi-Fi7 MLO	4			2024-05-17	24.05
				2024-07-13	24.07
				2024-10-11	24.10

5.16 Fixed Rate

This command is per-station debugfs knob, and please ensure that every parameter has a corresponding value.

```
cd /sys/kernel/debug/ieee80211/phy0/netdev:phy0.0-ap0/stations/<STA MAC Address>
echo "[mode] [bw] [mcs] [nss] [gi] [preamble] [stbc] [ldpc] [spe_en] [he_ltf]" > fixed_rate
```

Parameter	Description
Mode	0: CCK, 1: OFDM, 2: HT, 3: GF, 4: VHT, 8: HE SU, 9: HE ER, 15: EHT
BW	0: BW20, 1: BW40, 2: BW80, 3: BW160, 4: BW320
MCS	CCK: 0~4, OFDM: 0~7, HT: 0~32, VHT: 0~9, HE SU: 0~11, HE ER: 0~2, EHT: 0~13
NSS	VHT, HE, EHT: 1~4
GI	HT/VHT – 0: LGI, 1: SGI HE – 0: 0.8us, 1: 1.6us, 2: 3.2us
Preamble	0: Long, 1: Short
STBC	0: OFF, 1: ON
LDPC	0: OFF, 1: ON
SPE_EN	0: OFF, 1: ON

HE_LTF	0: 1xLTF, 1: 2xLTF, 2: 4xLTF

Take an example:

- HE MCS7 BW20 1NSS with 3.2us GI and 4xLTF

```
echo "8 0 7 1 2 0 0 1 0 2" > fixed_rate
```

5.17 MLO Debugfs Knobs (Only for Wi-Fi 7)

Per dev debugfs knob: Under MLD Single-Wiphy architecture, you must use phy0

```
/sys/kernel/debug/ieee80211/phy0/mt76
```

Per band debugfs knob

```
/sys/kernel/debug/ieee80211/phy0/mt76/bandX:
agginfo              pfmu_tag_read       thermal_enable
atf_enable           scs_enable          tx_stats
bf_fbk_rpt           sr_enable           txpower_info
bf_starec_read       sr_enhanced_enable  txpower_level
bf_txsnd_info        sr_scene_cond       txpower_path
hw-queues            sr_stats            txpower_sku
mibinfo              sys_recovery        xmit-queues
```

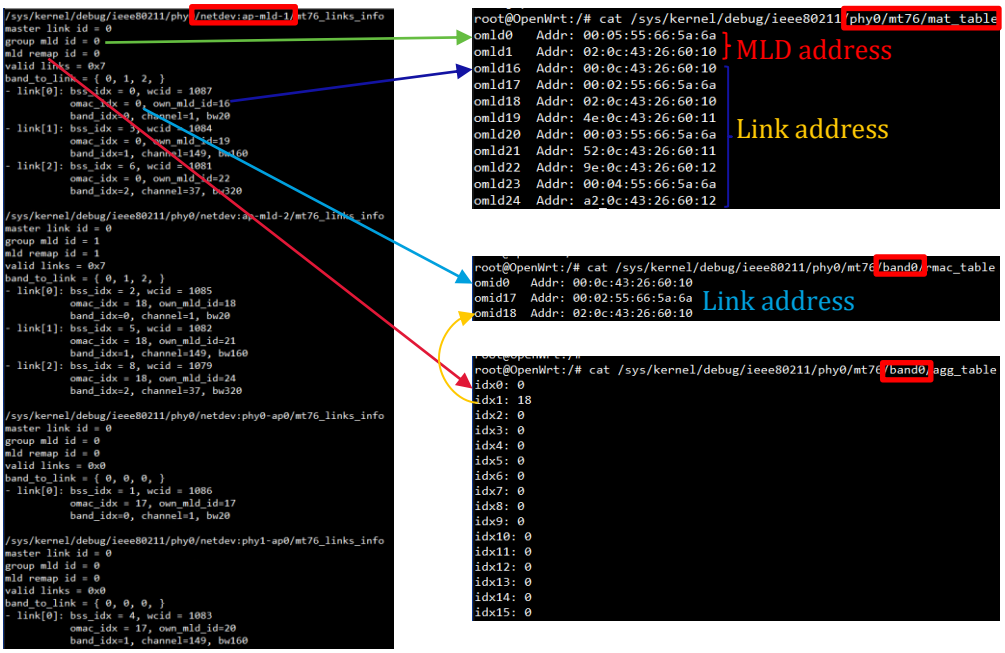
Check MLD Configuration

	Layering	Command
BSS	Mac80211	iw dev
	Mt76	cat /sys/kernel/debug/ieee80211/phy0/<interface>/mt76_links_info cat /sys/kernel/debug/ieee80211/phy0/mt76/mat_table cat /sys/kernel/debug/ieee80211/phy0/mt76/<bandX>/rmac_table cat /sys/kernel/debug/ieee80211/phy0/mt76/<bandX>/agg_table
Station	Mac80211	iw dev <interface> station dump
	Mt76	cd /sys/kernel/debug/ieee80211/phy0/ cat <interface>/stations/<mac address>/**mt76_links_info**
Pre-Link	Mac80211	iw dev <interface> station dump cat /sys/kernel/debug/ieee80211/phy0/<interface>/<link index>/addr
	Mt76	cd /sys/kernel/debug/ieee80211/phy0/ cat <interface>/stations/<mac address>/**<link index>/link_sta_info**

Expected Result:

BSS:

- iw dev: check the number of interfaces, interface types, number of links, and channel.
- mt76_links_info, mat_table, rmac_table, agg_table:



Station:

- iw dev <interface> station dump: check TRx rate, rssi, macaddress, and link num of connected stations.
- mt76_links_info: Get the setup link and WCID of connected stations.

Links:

- iw dev <interface> station dump: check TRx rate, rssi, macaddress, and link num of connected stations.
- Addr: Dump the link address
- link_sta_info: Get the trx rate and rssi of each ring.

5.18 MLO Statistics Knobs (Only for Wi-Fi 7)

Dump per-station (a.k.a. per-MLD) TX/RX information, which includes MSDU-based statistics.

```
# cd /sys/kernel/debug/ieee80211/phy0/netdev\:ap-mld-1/stations/00\:0c\:43\:4b\:ec\:d0/
```

```
# cat mt76_links_info
primary link, link ID = 0
secondary link, link ID = 1
valid links = 0x7
link0: wcid=1, phy=0, link_valid=1
link1: wcid=2, phy=1, link_valid=1
link2: wcid=3, phy=2, link_valid=1
TX MSDU Count: 248
TX MSDU Fails: 21 (PER: 8.4%)
TX MSDU Retries: 21
RX MSDU Count: 105
```

Dump per-link TX/RX information

```
# cd /sys/kernel/debug/ieee80211/phy0/netdev\:ap-mld-1/stations/00\:0c\:43\:4b\:ec\:d0/
# cat link-0/link_sta_info
WCID: 1
Link ID: 0
Link Address: 00:0c:43:4b:ec:d0
Status:
        RSSI: -12 [-14, -15, -61, -62] dBm
        ACK RSSI: -7 [-9, -10, -57, -57] dBm
        ACK SNR: [51, 51, 47, 46] dBm
Rate:
        TX: 103.2 Mbit/s EHT 20MHz MCS 4 NSS 2 GI 0.8us
        RX: 68.8 Mbit/s EHT 20MHz MCS 5 NSS 1 GI 0.8us
Statistics:
        TX:
                Bytes: 17679200
                MPDU Count: 3436
                MPDU Fails: 1019 (PER: 29.6%)
                MPDU Retries: 151
                Airtime: 2218740 (unit: 1.024 us)
        RX:
                Bytes: 37920
                MPDU Count: 4941
                MPDU FCS Errors: 256 (PER: 5.1%)
                Airtime: 189343 (unit: 1.024 us)
```

5.19 System Error Recovery

In wifi 6, ‘sys_recovery’ is in /sys/kernel/debug/ieee80211/phyX/mt76/
In wifi 7, ‘sys_recovery’ is in /sys/kernel/debug/ieee80211/phy0/mt76/bandX.

SER is a highly HW-related feature, it is not recommended to allow user trigger SER by command. Please check with Mediatek contact window for more details.

collect SER state information	**echo 0 > sys_recovery** cat sys_recovery
WA coredump	echo 10 > sys_recovery
WM coredump	echo 11 > sys_recovery
Test trigger L1 SER	echo 1 > sys_recovery
Test trigger L0.5 SER	echo 8 > sys_recovery

6 Firmware Mode (Wi-Fi 7 Must Do)

In Wi-Fi 7 chipsets, due to the limitation of RAM size, we divide WM firmware into two bins: the normal mode WM firmware bin and the test mode WM firmware bin. Therefore, before starting up, **please check that you are loading the normal mode WM firmware bin.**

[Important updates after MLO are supported]

According to the MLO data structure refactor mentioned in Wi-Fi 7 Multi-Link Operation Introduction (MLO), MT76 normal mode has transitioned to the **Single Wiphy Model**; however, testmode will continue to use the legacy **Multiple Wiphy Model** due to monitor interface limitations in the kernel. Therefore, please **disable all the normal mode interfaces** and **set the UCI config to disable** before switching to test mode; otherwise, it will result in a severe **kernel crash**.

For Wi-Fi 6 chipsets, please ignore this chapter.

6.1 Check Firmware Mode

To check the current firmware mode you are using, you can either enter the following command or check the bootup log.

```
#cat /sys/kernel/debug/ieee80211/phy0/mt76/fw_version
```

```
root@OpenWrt:/# cat /sys/kernel/debug/ieee80211/phy0/mt76/fw_version
Version: 3.3.10.0
Rom Patch Build Time: 20230606161755a

WM Patch Build Time: 20230606161745, Mode: Normal mode
WA Patch Build Time: 20230606161652
DSP Patch Build Time: 20230606161623
```

Bootup log:

- Normal mode

```
mt7996e 0000:01:00.0: WM Firmware Version: ____000000, Build Time: 20230606161745
mt7996e 0000:01:00.0: DSP Firmware Version: ____000000, Build Time: 20230606161623
mt7996e 0000:01:00.0: WA Firmware Version: ____000000, Build Time: 20230606161652
```

- Test mode

```
mt7996e 0000:01:00.0: WM_TM Firmware Version: ____000000, Build Time: 20230516165518
mt7996e 0000:01:00.0: DSP Firmware Version: ____000000, Build Time: 20230516165216
mt7996e 0000:01:00.0: WA Firmware Version: ____000000, Build Time: 20230516165241
```

6.2 Enter Normal Mode Firmware

For **Wi-Fi 7 chipsets**, the driver will determine which WM firmware bin to load during bootup based on your EEPROM mode (See 7.5) (flash, eFuse, binfile, or default bin mode), EEPROM fields, and input module parameters. (See 7 eFuse/Flash/BinFile Mode)

To check your current EEPROM mode, please refer to 7.5. To enter test mode firmware mode, please refer to Section 2.1 in "MT76 Test Mode Programming Guide."

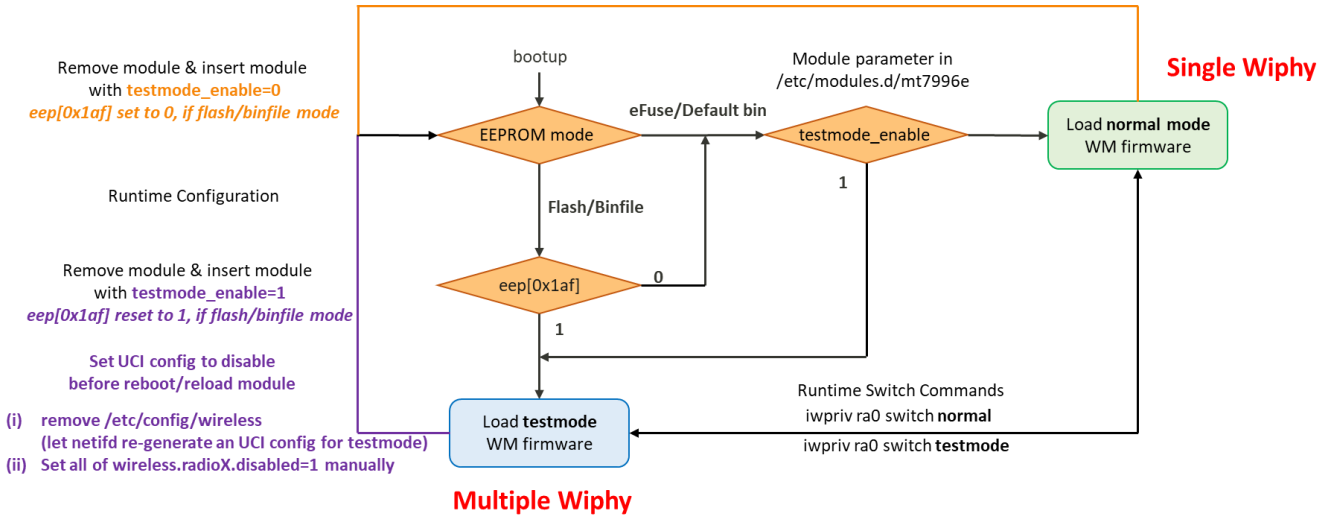
Please refer to the following descriptions or the following flowchart for the MT76 WM firmware loading flow.

(i) Flash & binfile mode:

For flash and binfile mode, MT76 driver first checks EEPROM field 0x1af (eep[0x1af]) to determine which bin to load. If eep[0x1af] = 0, then MT76 driver directly loads normal mode firmware. Otherwise, MT76 driver further checks module parameter "testmode_enable" in /etc/module.d/mt7996e to decide which bin to load.

(ii) eFuse & default bin mode:

For eFuse and default bin mode, MT76 driver only checks module parameter "testmode_enable" in /etc/module.d/mt7996e to decide which bin to load.



There are two ways to **switch firmware modes in runtime**.

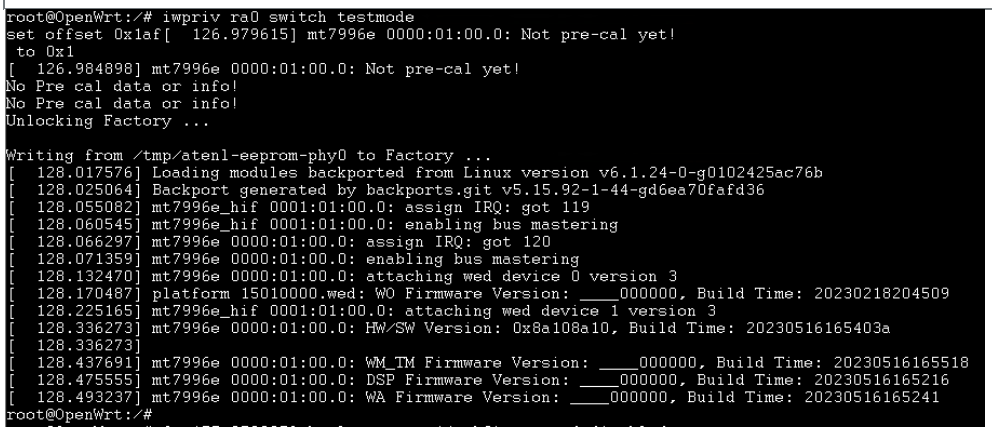
(i) Re-insert modules:

```
#cat /sys/kernel/debug/ieee80211/phy0/mt76/eeprom_mode
## If eeprom mode == flash or binfile
#atenl -i phy0 -c "eeprom set 0x1af=0x<testmode_enable>"
#atenl -i phy0 -c "eeprom precal sync"
#atenl -i phy0 -c "sync eeprom all"

#wifi down
## After Single Wiphy refactor, please set normal mode interface to disabled
## Method 1: Netifd will generate an UCI config based on current firmware
            mode(disabled = 1 for testmode)
#rm /etc/config/wireless
## Method 2: Set UCI manually
#uci set wireless.radio0.disabled=<testmode_enable>
#uci set wireless.radio1.disabled=<testmode_enable>
#uci set wireless.radio2.disabled=<testmode_enable>
#uci commit


#rmmod mt7996e
#rmmod mt76-connac-lib
#rmmod mt76
#rmmod mac80211
#rmmod cfg80211
#rmmod compat
#modprobe compat
```

```
#modprobe cfg80211
#modprobe mac80211
#modprobe mt76
#modprobe mt76-connac-lib
#modprobe mt7996e testmode_enable=<testmode_enable>
#sleep 5
#killall hostapd
#killall netifd
```

(ii) iwpriv wrapper commands (wraps the above commands):

```
#iwpriv ra0 switch <testmode/normal>
```



7 eFuse/Flash/BinFile Mode

7.1 Flash Mode (NAND flash)

When the DTS file is configured, the initiation of MT76 will occur in the flash mode instead of the eFuse mode.
Please refer to the corresponding examples below based on your kernel version, Wi-Fi generation, and Wi-Fi card type (PCIE card or SoC).
For eMMC flash, please refer to 7.3 eMMC Flash Support.
Note that the code should be rebuild after modifying the following DTS file.
To check if the flash mode is successfully entered, please refer to 7.5 Check EEPROM Mode

[Kernel 5.4]

The path of DTS is "./openwrt/lede/target/linux/mediatek/file-5.4/arch/arm64/boot/dts/mediatek/".

The configuration flow is the same for all chips (including Connac 2 & 3 chips); however, the DTS filename might be slightly different. The DTS filename is listed in the following table.

(i) For the PCIE Wi-Fi card, see:
Wi-Fi 6 example: Take the mt7915 chip in the AX8400 as an example.
STEP 1: Add a "slot0" node under "pcie0" node if it doesn't exist. For example, in "**mt7986a.dtsi**" file (for AX6000 (please use mt7986b.dtsi).

```
pcie0: pcie@11280000 {
                                        ⋮
        slot0: pcie@0,0 {
                  reg = <0x0000 0 0 0 0>;
        };
};
```

STEP 2: Add a "slot0" node in DTS files of mt7986 (mt7986a-2500wan-spim-nand-rfb.dts), where 0x5000 is the offset of eeprom data in flash memory.

```
&slot0 {
         mt7915@0,0 {
                  reg = <0x0000 0 0 0 0>;
                  device_type = "pci";
                  mediatek,mtd-eeprom = <&factory 0x5000>;
         };
};
```

Wi-Fi 7 example: Take the MT7996 + MT7988 platform (BE19000) as an example.
STEP 1: Add a "slot0" node under "pcie0" node if it doesn't exist. For example, in "**mt7988.dtsi**" file.

```
pcie0: pcie@11280000 {
                                        ⋮
        slot0: pcie@0,0 {
                  reg = <0x0000 0 0 0 0>;
        };
};
```

STEP 2: Add a "slot0" node in DTS files of mt7988 (mt7988a-gsw-spim-nand.dts or mt7988a-dsa-spim-nand.dts), where 0x5000 is the offset of eeprom data in flash memory.

Select the corresponding DTS file based on the image you use. For example, if you use “openwrt-mediatek-mt7988-mediatek_mt7988a-**gsw**-10g-spim-nand-squashfs-sysupgrade.bin”, please make sure your slot0 is added in mt7988a-**gsw**-spim-nand.dts instead of mt7988a-**dsa**-spim-nand.dts.

```
&slot0 {
        mt7996@0,0 {
                reg = <0x0000 0 0 0 0>;
                device_type = "pci";
                mediatek,mtd-eeprom = <&factory 0x5000>;
        };
};
```

(ii) For SoC:

Take the MT7986 chip in the AX8400 as an example.

Please configure mt7986a(or b)-2500wan-spim-nand-rfb.dts

```
&wbsys {
         mediatek,mtd-eeprom = <&factory 0x0000>;
         status = "okay";
         pinctrl-names = "default", "dbdc";
         pinctrl-0 = <&wf_2g_5g_pins>;
         pinctrl-1 = <&wf_dbdc_pins>;
};
```

Note:

- **For enabling both MT7986 and MT7915, please do (i) & (ii) respectively.**

[Kernel 6.6]

The path of DTS is “./openwrt/openwrt/target/linux/mediatek/file-6.6/arch/arm64/boot/dts/mediatek/”.

Note that only Wi-Fi 7 supports kernel 6.6.

Take the MT7996 + MT7988 spim nand platform (BE19000) as an example.

STEP 1: Add a Factory partition under flash node if it doesn’t exist (mt7988a-rfb-spim-nand-nmbm.dtso).

```
fragment@0 {
    target = <&spi0>;
    __overlay__ {
        flash@0 {
            partitions {
                compatible = "fixed-partitions";
                ...
                factory: partition@180000 {
                    label = "Factory";
                    reg = <0x180000 0x0400000>;
                };
                ...
                };
            };
        };
    };
```

```
};
```

STEP 2: Add a "slot0" node in DTSO files of mt7988 spim nand (mt7988a-rfb-spim-nand-nmbm.dtso).

```
fragment@1 {
        target = <&pcie0>;
        __overlay__ {
                slot0: pcie@0,0 {
                        reg = <0x0000 0 0 0 0>;
                        mt7996@0,0 {
                                compatible = "mediatek,mt76";
                                reg = <0x0000 0 0 0 0>;
                                device_type = "pci";
                                mediatek,mtd-eeprom = <&factory 0x0>;
                        };
                };
        };
};
```

7.2 BinFile Mode

If the DTS file and Makefile are configured, the MT76 will initiate with the BinFile mode rather than the eFuse mode.
Please refer to the corresponding examples below based on your kernel version, Wi-Fi generation, and Wi-Fi card type (PCIE card or SoC).
Note that the code should be rebuild after modifying the DTS file.

To check if the binfile mode is successfully entered, please refer to 7.5 Check EEPROM Mode

[Kernel 5.4]

The path of DTS is “./openwrt/lede/target/linux/mediatek/file-5.4/arch/arm64/boot/dts/mediatek/”.
(i) For the PCIe Wi-Fi card:
Wi-Fi 6 example: Take the mt7915 chip in AX8400, for example.
Step 1: Add a "slot0" node under "pcie0" node if it doesn't exist. For example, in "mt7986a.dtsi" file.

```
pcie0: pcie@11280000 {
                                        ⋮
        slot0: pcie@0,0 {
                reg = <0x0000 0 0 0 0>;
        };
};
```

Step 2: Add a "slot0" node in DTS files of mt7986 (mt7986a-2500wan-spim-nand-rfb.dts). Note that the name of “bin_file_name” is customizable.

```
&slot0 {
        mt7915@0,0 {
                reg = <0x0000 0 0 0 0>;
                device_type = "pci";
                bin_file_name = "mediatek/bin_file_mode_1.bin";
        };
};
```

Wi-Fi 7 example, take MT7996 + MT7988 (BE19000).

Step 1: Add a "slot0" node under "pcie0" node if it doesn't exist. For example, in "mt7988.dtsi" file.

```
pcie0: pcie@11280000 {
                                        ⋮
        slot0: pcie@0,0 {
                reg = <0x0000 0 0 0 0>;
        };
};
```

Step 2: Add a "slot0" node in DTS files of mt7988 (mt7988a-gsw-spim-nand.dts or mt7988a-dsa-spim-nand.dts). Note that the name of “bin_file_name” is customizable.

Select the corresponding DTS file based on the image you use. For example, if you use “openwrt-mediatek-mt7988-mediatek_mt7988a-**gsw**-10g-spim-nand-squashfs-sysupgrade.bin”, please make sure your slot0 is added in mt7988a-**gsw**-spim-nand.dts instead of mt7988a-**dsa**-spim-nand.dts.

```
&slot0 {
        mt7996@0,0 {
                reg = <0x0000 0 0 0 0>;
                device_type = "pci";
                bin_file_name = "mediatek/bin_file_mode_1.bin";
        };
};
```

(ii) For SoC:

Take the mt7986 chip in AX8400, for example.

Please configure mt7986a(or b)-2500wan-spim-nand-rfb.dts

```
&wbsys {
        mediatek,mtd-eeprom = <&factory 0x0000>;
        bin_file_name = "mediatek/bin_file_mode_2.bin";
        status = "okay";
        pinctrl-names = "default", "dbdc";
        pinctrl-0 = <&wf_2g_5g_pins>;
        pinctrl-1 = <&wf_dbdc_pins>;
};
```

Note:

- **For enabling both mt7986 and mt7915, please do (i) & (ii).**

Note that the BinFile mode bin should be placed in the same path as the default bin and be added to the Makefile[1] (for building the code). See below for an example of how to configure the Makefile.

Wi-Fi 6 Makefile

```
                                        ⋮
  cp \
    $(PKG_BUILD_DIR)/firmware/mt7916_eeprom.bin \
```

[1] Path of Makefile: ./openwrt/lede/package/kernel/mt76/

```
    $(PKG_BUILD_DIR)/firmware/mt7915_eeprom.bin \
    $(PKG_BUILD_DIR)/firmware/mt7915_eeprom_dbdc.bin \
    $(PKG_BUILD_DIR)/firmware/mt7916_binfile.bin \
    $(1)/lib/firmware/mediatek
                                        ⋮
define KernelPackage/mt7986-firmware/install
  $(INSTALL_DIR) $(1)/lib/firmware/mediatek
  cp \
    $(PKG_BUILD_DIR)/firmware/mt7986_eeprom_mt7976_dual.bin \
    $(PKG_BUILD_DIR)/firmware/mt7986_eeprom_mt7976.bin \
    $(PKG_BUILD_DIR)/firmware/mt7986_eeprom_mt7976_dbdc.bin \
    $(PKG_BUILD_DIR)/firmware/mt7986_eeprom_mt7975_dual.bin \
    $(PKG_BUILD_DIR)/firmware/mt7986_eeprom_mt7975.bin \
    $(PKG_BUILD_DIR)/firmware/mt7986_binfile_mt7975_dual.bin \
    $(1)/lib/firmware/mediatek
                                        ⋮
```

Wi-Fi 7 Makefile

```
                                        ⋮
# Please find the corresponding define for your chip sku
# For example
#   - mt7996-firmware for MT7996 BE19000
#   - mt7992-firmware for MT7992 BE7200
define KernelPackage/<your sku>/install
  $(INSTALL_DIR) $(1)/lib/firmware/mediatek/mt7996
  cp \
    $(PKG_BUILD_DIR)/firmware/mt7996/xxx.bin \
    ...
    $(PKG_BUILD_DIR)/firmware/mt7996/bin_file_mode_1.bin \
    $(1)/lib/firmware/mediatek/mt7996
endef
                                        ⋮
```

The BinFile mode bin's name should be the same as the bin_file_name set in the DTS file.

To obtain the current RF calibration data in flash/efuse for revising the binfile bin, please enter the following command[2]:

```
#dd if=/sys/kernel/debug/ieee80211/phy0/mt76/eeprom of=xxx.bin bs=1 count=4096
#tftp -p -r xxx.bin TFTP_SERVER_IP
```

SKU	DTSI	DTS
AX6000	mt7986**b**.dtsi	mt7986**b**-2500wan-spim-nand-rfb.dts
AX7800	mt7986**a**.dtsi	mt7986**a**-2500wan-spim-nand-rfb.dts
AX8400	mt7986**a**.dtsi	mt7986**a**-2500wan-spim-nand-rfb.dts
BE19000 (MT7988a + MT7996)	mt7988.dtsi	mt7988**a**-dsa-10g-spim-nand.dts (gsw build is retired in Wi-Fi 7 MT76)
BE14000 (MT7988d + *2adie tri-band* MT7996)		mt7988**d**-dsa-10g-spim-nand.dts (gsw build is retired in Wi-Fi 7 MT76)
BE7200 (MT7988D + MT7992)		

[2] Please replace "phy0" in the following command with your desired interface.

BE5040 (MT7988D + MT7992)		

[Kernel 6.6]

The path of DTS is "./openwrt/openwrt/target/linux/mediatek/file-6.6/arch/arm64/boot/dts/mediatek/".

Step 1: Add a "slot0" node in corresponding DTSO files of your platform. For MT7988A eMMC, please configure the mt7998a-rfb-emmc.dtso. For MT7988A spim nand, please configure the mt7988a-rfb-spim-nand-nmbm.dtso.
Note that the name of "bin_file_name" is customizable.

```
fragment@1 {
    target = <&pcie0>;
    __overlay__ {
        slot0: pcie@0,0 {
            reg = <0x0000 0 0 0 0>;
                mt7996@0,0 {
                    reg = <0x0000 0 0 0 0>;
                    bin_file_name = "mediatek/bin_file_mode_1.bin";
                };
        };
    };
};
```

Note that the BinFile mode bin should be placed in the same path as the default bin and be added to the Makefile (for building the code). Please refer to the examples in Kernel 5.4 on how to configure the Makefile.

7.3 eMMC Flash Support

When the DTS file is configured, the initiation of MT76 will occur in the flash mode instead of the eFuse mode.
Note that the code should be rebuild after modifying the following DTS file.
Additionally, only Wi-Fi 7 chipsets on the kernel 6.6 platform support eMMC flash mode.
To check if the flash mode is successfully entered, please refer to 7.5 Check EEPROM Mode

[Kernel 5.4]

Kernel 5.4 does not support reading EEPROM data from eMMC flash, as it requires the NVMEM API to read it.
Therefore, please use binfile mode in this case. For more information, please refer to 7.2 BinFile Mode

[Kernel 6.6]

The path of DTS is "./openwrt/openwrt/target/linux/mediatek/file-6.6/arch/arm64/boot/dts/mediatek/".

Take the MT7996 + MT7988 eMMC platform (BE19000) as an example.
STEP 1: Add a Factory partition (block-partition-factory) under eMMC node if it doesn't exist (mt7988a-rfb-emmc.dtso), where 0x0 in the reg is the offset of the EEPROM in the eMMC flash, and the 0x1e00 is the EEPROM size.

```
fragment@0 {
    target = <&mmc0>;
    __overlay__ {
        card@0 {
```

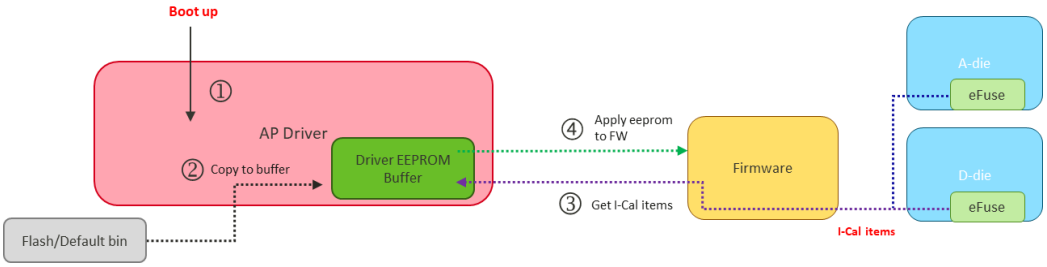
```
            block {
                compatible = "block-device";
                ...
                partitions {
                    ...
                    block-partition-factory {
                        partname = "factory";
                        nvmem-layout {
                            compatible = "fixed-layout";
                            #address-cells = <1>;
                            #size-cells = <1>;

                            eeprom_factory_0: eeprom@0 {
                                reg = <0x0 0x1e00>;
                            };
                        };
                    };
                    ...
                };
            };
        };
    };
};
```

STEP 2: Add a "slot0" node in DTSO files of mt7988 eMMC (mt7988a-rfb-emmc.dtso)
eeprom_factory_0 will serve as a phandle (pointer handle) that points to the EEPROM node in step 1.

```
fragment@1 {
        target = <&pcie0>;
        __overlay__ {
                slot0: pcie@0,0 {
                        reg = <0x0000 0 0 0 0>;
                        mt7996@0,0 {
                                reg = <0x0000 0 0 0 0>;
                                nvmem-cells = <&eeprom_factory_0>;
                                nvmem-cell-names = "eeprom";
                        };
                };
        };
};
```

7.4 Dump eFuse Content

After the FT (Final Test), chip-dependent calibrated data will be written to the eFuse of adie and ddie for better performance. Therefore, as shown in the following figure, the driver should apply those I-cal data during loading EEPROM from either flash memory or any other source; this action is also called **eFuse merge**.



To dump the content of eFuse, please enter the following commands:

```
# hexdump -C /sys/kernel/debug/ieee80211/phy0/mt76/otp
```

Note that OTP stands for One Time Programmable.

7.5 Check EEPROM Mode

To check your current EEPROM mode, please enter the following commands:

```
# cat /sys/kernel/debug/ieee80211/phy0/mt76/eeprom_mode
```

```
root@OpenWrt:/# cat /sys/kernel/debug/ieee80211/phy0/mt76/eeprom_mode
Current eeprom mode:
   flash mode
   mtd name = Factory
   flash offset = 0x0
```

8 Useful Debug Tips

8.1 Enable Telnet Service

```
# telnetd -l /bin/ash
```

8.2 Check Debug Log

8.2.1 Mt76 (Only for Wi-Fi 7)

The default value of the mt7996 debug mask is 0x1f, which means enabling all logs except TXRX.

- Check the value of the debug mask

```
# cat /sys/module/mt7996e/parameters/mt7996_debug_mask
```

- There are two methods to modify the value
  - vim /etc/modules.d/mt7996e

```
# mt7996e mt7996_debug_mask=<new value>
# reboot
```

  - Reload the mt7996e module with the new debug mask value

```
# modprobe mt7996e mt7996_debug_mask=<new value>
```

Type	Value
MT7996_DBG_DEV	BIT(0)
MT7996_DBG_BSS	BIT(1)
MT7996_DBG_STA	BIT(2)
MT7996_DBG_CHAN	BIT(3)
MT7996_DBG_MLD	BIT(4)
MT7996_DBG_TXRX	BIT(5)
MT7996_DBG_ALL	0xffffffff

8.2.2 Hostapd/Wpa_Supplicant

Logread on the router can read the ring buffer records, stream them to a file, and transmit them to a remote system via a TCP/UDP socket.

```
#logread -f

Sat Jan  1 00:04:24 2000 daemon.notice hostapd: Configuration file:
/var/run/hostapd-phy0.conf (phy wlan0) --> new PHY
Sat Jan  1 00:04:24 2000 daemon.notice hostapd: wlan0: interface state
UNINITIALIZED->HT_SCAN
Sat Jan  1 00:04:25 2000 daemon.err hostapd: Using interface wlan0 with hwaddr
00:0c:43:2a:05:85 and ssid "OpenWrt"
Sat Jan  1 00:04:25 2000 daemon.notice hostapd: wlan0: interface state HT_SCAN-
>ENABLED
Sat Jan  1 00:04:25 2000 daemon.notice hostapd: wlan0: AP-ENABLED
```

8.3 Debug Level Change - Hostapd/Wpa_Supplicant

There are two kinds of debug log functions in hostapd, wpa_msg() series and hostapd_logger(), respectively. Below we will introduce how to enable them.

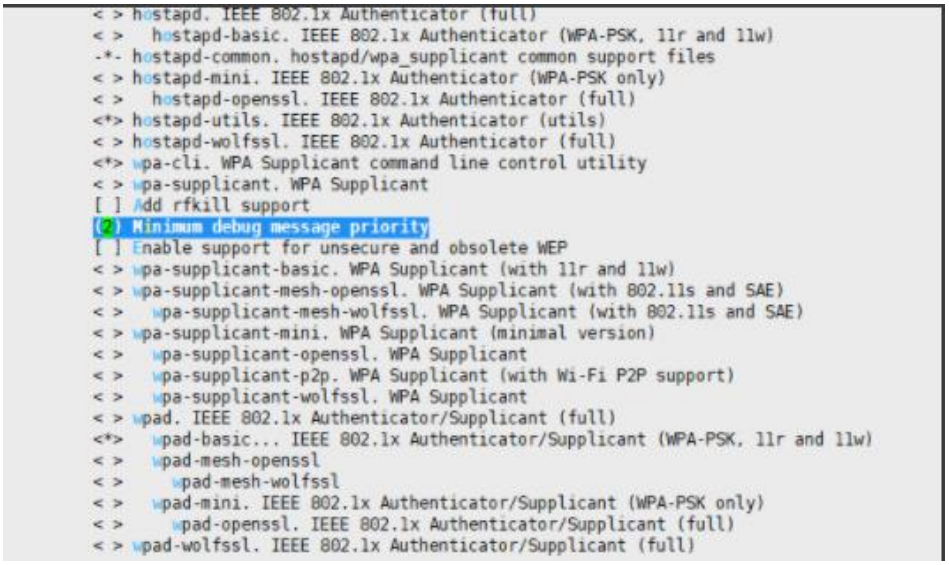
8.3.1 wpa_msg()

The wpa_msg() series include wpa_msg(), wpa_dbg(), wpa_printf(), wpa_hexdump(), etc. If you want to print out the debug log down to specific level by wpa_msg() series, you must lower the configuration CONFIG_WPA_MSG_MIN_PRIORITY and “wpa_debug_level” to equal to that level.

For example, if you want to print out all the level 1 logs, which represents as MSG_MSGDUMP in hostapd source code, you must lower the two configurations to 1.

Lower the CONFIG_WPA_MSG_MIN_PRIORITY:
In current mt76 implementation, the default value of CONFIG_WPA_MSG_MIN_PRIORITY is 2.

You can use $ *make menuconfig* to modify the MIN Priority debug level and then override the **CONFIG_WPA_MSG_MIN_PRIORITY** definition as the following picture.



Lower the “wpa_debug_level”:

In current mt76 implementation, the default value of “wpa_debug_level” is 3. You can add one “-d” to the hostapd/wpa_supplicant command line arguments to lower “wpa_debug_level” by 1. In other words, you can add “-dd” to lower “wpa_debug_level” by 2.

In openWRT, the wifi command is typically used to execute hostapd/wpa_supplicant instead of directly running the hostapd/wpa_supplicant command. It is recommended to adjust the wpad init script so that the wifi command executes hostapd/wpa_supplicant with your desired command line arguments.

```
# vim /etc/init.d/wpad
Modify the line 14 and 31 in the following picture to add the command line arguments
for lowering the wpa_debug_level of hostapd and wpa_supplicant respectively.
# /etc/init.d/wpad restart
```

```
#!/bin/sh /etc/rc.common

START=19
STOP=21

USE_PROCD=1
NAME=wpad

start_service() {
        if [ -x "/usr/sbin/hostapd" ]; then
                mkdir -p /var/run/hostapd
                chown network:network /var/run/hostapd
                procd_open_instance hostapd
                procd_set_param command /usr/sbin/hostapd -s -g /var/run/hostapd/global -dd
                procd_set_param respawn 3600 1 0
                procd_set_param limits core="unlimited"
                [ -x /sbin/ujail -a -e /etc/capabilities/wpad.json ] && {
                        procd_add_jail hostapd
                        procd_set_param capabilities /etc/capabilities/wpad.json
                        procd_set_param user network
                        procd_set_param group network
                        procd_set_param no_new_privs 1
                }
                procd_close_instance
        fi

        if [ -x "/usr/sbin/wpa_supplicant" ]; then
                mkdir -p /var/run/wpa_supplicant
                chown network:network /var/run/wpa_supplicant
                procd_open_instance supplicant
                procd_set_param command /usr/sbin/wpa_supplicant -n -s -g /var/run/wpa_supplicant/global -dd
                procd_set_param respawn 3600 1 0
                procd_set_param limits core="unlimited"
                [ -x /sbin/ujail -a -e /etc/capabilities/wpad.json ] && {
                        procd_add_jail wpa_supplicant
                        procd_set_param capabilities /etc/capabilities/wpad.json
                        procd_set_param user network
                        procd_set_param group network
                        procd_set_param no_new_privs 1
```

Increase the system log size:
If you lower the wpa_debug_level, the default system log size might not be enough. You can then modify the system log size in /etc/config/system. After that, you need to reboot the device.

```
# vim /etc/config/system
Modify the line 6 in the following picture to change the system log size.
# reboot
```

```

config system
        option hostname 'OpenWrt'
        option timezone 'UTC'
        option ttylogin '0'
        option log_size '4096'
        option urandom_seed '0'

config timeserver 'ntp'
        option enabled '1'
        option enable_server '0'
        list server '0.openwrt.pool.ntp.org'
        list server '1.openwrt.pool.ntp.org'
        list server '2.openwrt.pool.ntp.org'
        list server '3.openwrt.pool.ntp.org'

config rngd
        option enabled '0'
        option device '/dev/urandom'

```

8.3.2 hostapd_logger

If you want to print out the debug log down to specific level by hostapd_logger(), you must lower the configuration "logger_syslog_level" to equal to that level you want to print.

 # vim /etc/config/wireless
Please include the log level according to the information in the provided picture.

8.4 Firmware - Replace the Build-In

The firmware location in the kernel path is under */lib/firmware/mediatek.*

```
#cd /lib/firmware/mediatek
#tftp -g -r xxx.bin xxx.xxx.xxx.xxx
#reboot
```

Check the build time of the firmware when booting.

```
mt7915e 0001:01:00.0: HW/SW Version: 0x8a108a10, Build Time: xxxxxxxxxxxxx
mt7915e 0001:01:00.0: WM Firmware Version: ____000000, Build Time: xxxxxxxxxxxxx
mt7915e 0001:01:00.0: WA Firmware Version: DEV_000000, Build Time: xxxxxxxxxxxxx
```

8.5 Firmware - Enable Firmware Log

```
#cd /sys/kernel/debug/ieee80211/phy0/mt76

WM FW
# fw log to host uart (BIT1)
echo 2 > fw_debug_wm
# fw log to WM uart (BIT0)
echo 1 > fw_debug_wm
# fw log to host and WM uart
echo 3 > fw_debug_wm

WA FW
# WA log to host
echo 2 > fw_debug_wa

# WA CPU utlz (need to enable WA log)
cat fw_util_wa
```

```
# iwpriv ra0 set cr4_set=4:1:100
echo 0x640104 > wa_debug_set

# iwpriv ra0 set cr4_query=b (need to enable WA log)
echo 0x0b > wa_debug_query
```

8.6 Firmware - Firmware Parser Tool

The MT76 supports firmware log parser tools to dump the firmware log for debugging.
The corresponding person in charge of MediaTek should be contacted for further information.

```
# fw log to Ethernet (FW log)
mt76-test phy0 fwlog <pc_ip_addr> 15   (FW version date after MT7915 20210330 &
MT7986 & MT7916 & MT7996)
mt76-test phy0 fwlog <pc_ip_addr> 7   (FW version date before MT7915 20210330)

#Note that phy should be the index of DBDC band0 even if you want to log band1,
since the relay file is only created in debugfs of band0.
```

Set the firmware module or the debug level

```
echo 73 > fw_debug_module
echo 1 > fw_debug_level
```

8.7 Set/Dump Control & RF Register

Dump Control Register

```
#cd /sys/kernel/debug/ieee80211/phy0/mt76
#echo 0xXXXXXXXX > regidx
#cat regval
```

Set Control Register

```
#cd /sys/kernel/debug/ieee80211/phy0/mt76
#echo 0xXXXXXXXX > regidx
#echo 0xXXXXXXXX > regval
```

Dump RF Register

```
#cd /sys/kernel/debug/ieee80211/phy0/mt76

#regidx combines with two parts: WF selection [31:24] and offset [23:0]
#echo 0xXXXXXXXX > regidx

#cat rf_regval
```

Set RF Register

```
#cd /sys/kernel/debug/ieee80211/phy0/mt76

#regidx combines with two parts: WF selection [31:24] and offset [23:0]
#echo 0xXXXXXXXX > regidx

#echo 0xXXXXXXXX > rf_regval
```

8.8 Partial Build/Replace Kernel Module of Wi-Fi Driver

Partial Build

```
#  make package/kernel/mt76/compile -j1 V=s
```

Wi-Fi 6 (MT7915/MT7916/MT7986)	Wi-Fi 7 (MT7992/MT7996)
Replace and reload Wi-Fi driver ## reload.sh tftp -gr mt7915e.ko 192.168.x.x mv ./mt7915e.ko /lib/modules/$(uname -r) rmmod mt7915e rmmod mt76-connac-lib rmmod mt76 rmmod mac80211 rmmod cfg80211 modprobe cfg80211 modprobe mac80211 modprobe mt76 modprobe mt76-connac-lib modprobe mt7915e sleep 5 killall hostapd killall netifd	Replace and reload Wi-Fi driver ## reload.sh tftp -gr mt7996e.ko 192.168.x.x mv ./mt7996e.ko /lib/modules/$(uname -r) rmmod mt7996e rmmod mt76-connac-lib rmmod mt76 rmmod mac80211 rmmod cfg80211 modprobe cfg80211 modprobe mac80211 modprobe mt76 modprobe mt76-connac-lib modprobe mt7996e sleep 5 killall hostapd killall netifd

8.9 Runtime Config BA Session

```
#cd /sys/kernel/debug/ieee80211/phy1/netdev:phy1-
ap0/stations/xx:xx:xx:xx:xx:xx

Set “tx/rx start/stop timeout=xx TID” > agg_status
Ex:
#echo "tx start 1" > agg_status (Send Add_BA TID=1 of INITIATOR)
#echo "rx stop 0" > agg_status (Send Del_BA TID=0 of RECIPIENT)

Get
Ex:
#cat agg_status
```

8.10 Adjust WMM Parameters

- *By Hostapd configuration*

The user can modify the WMM parameters by setting the hostapd configuration.

Configuration format:

```
tx_queue_<AC queue>_<parameters>
    -   AC queue: data0, data1, data2, data3
    -   parameters: aifs, cwmin, cwmax, burst
```

For example, if users want to set the AC_BE TXOP, set **tx_queue_data2_burst** and the TXOP would be:

```
(tx_queue_data2_burst * 1000 + 16) >> 5 (unit: 32us)
```

So, the user can set tx_queue_data2_burst=5.9 to set TXOP=0xb8. For more detail, please refer to hostapd configuration.

- *By debugfs configuration (WiFi7 Only)*

```
# Configuration
## Usage:
echo [ap|sta] tx_queue_data2_burst > /sys/kernel/debug/ieee80211/phy0/mt76/txop
## Example:
echo ap 0.0 > /sys/kernel/debug/ieee80211/phy0/mt76/txop
echo sta 5.9 > /sys/kernel/debug/ieee80211/phy0/mt76/txop

# Dump
## Usage:
cat /sys/kernel/debug/ieee80211/phy0/mt76/txop
## Example
root@OpenWrt:/# # TXOP 0
root@OpenWrt:/# echo ap 0.0 > /sys/kernel/debug/ieee80211/phy0/mt76/txop
root@OpenWrt:/# echo sta 0.0 > /sys/kernel/debug/ieee80211/phy0/mt76/txop
root@OpenWrt:/# cat /sys/kernel/debug/ieee80211/phy0/mt76/txop
Band    AP (WMM0)               STA (WMM3)
0       0x00000000 (0.0)        0x00000000 (0.0)
1       0x00000000 (0.0)        0x00000000 (0.0)
2       0x00000000 (0.0)        0x00000000 (0.0)
root@OpenWrt:/#
```

```
root@OpenWrt:/# # TXOP 2
root@OpenWrt:/# echo ap 2.0 > /sys/kernel/debug/ieee80211/phy0/mt76/txop
root@OpenWrt:/# echo sta 2.0 > /sys/kernel/debug/ieee80211/phy0/mt76/txop
root@OpenWrt:/# cat /sys/kernel/debug/ieee80211/phy0/mt76/txop
Band    AP (WMM0)               STA (WMM3)
0       0x003f0000 (2.0)        0x003f0000 (2.0)
1       0x003f0000 (2.0)        0x003f0000 (2.0)
2       0x003f0000 (2.0)        0x003f0000 (2.0)
root@OpenWrt:/#
root@OpenWrt:/# # TXOP 6
root@OpenWrt:/# echo ap 5.9 > /sys/kernel/debug/ieee80211/phy0/mt76/txop
root@OpenWrt:/# echo sta 5.9 > /sys/kernel/debug/ieee80211/phy0/mt76/txop
root@OpenWrt:/# cat /sys/kernel/debug/ieee80211/phy0/mt76/txop
Band    AP (WMM0)               STA (WMM3)
0       0x00b80000 (5.8)        0x00b80000 (5.8)
1       0x00b80000 (5.8)        0x00b80000 (5.8)
2       0x00b80000 (5.8)        0x00b80000 (5.8)
```

8.11 Logs Collection

It provides a step-by-step guide on how to collect and report logs when encountering an issue. The process involves starting log collection, continuously gathering runtime logs, and stopping log collection to save and send the logs for analysis.

Wi-Fi 7 example:

```
## Introduction

## Steps to Report an Issue

1. **Adjust Log Size and Hostapd Level:**
    #sed -i "s/log_size '64'/log_size '32768'/g" /etc/config/system
    #service log restart
    #hostapd_cli -i ap-mld-1 log_level DEBUG # This value can be configured by
hostapd_log_level

2. **Start Recording Firmware Log:**
    This value can be configured by enable_fw_logs
    #killall mt76-test
    Enable FW log
    dev1: Enable ICS on band [0, 1, 2]
    #echo -n 0x820e705c > /sys/kernel/debug/ieee80211/phy0/mt76/regidx; echo -n 0x01ff0000
    > /sys/kernel/debug/ieee80211/phy0/mt76/regval
    #echo -n 0x820e7060 > /sys/kernel/debug/ieee80211/phy0/mt76/regidx; echo -n 0x058000d0
    > /sys/kernel/debug/ieee80211/phy0/mt76/regval
    #echo -n 0x820e4120 > /sys/kernel/debug/ieee80211/phy0/mt76/regidx; echo -n 0x00000001
    > /sys/kernel/debug/ieee80211/phy0/mt76/regval
    #echo -n 0x820e50d0 > /sys/kernel/debug/ieee80211/phy0/mt76/regidx; echo -n 0x00000101
    > /sys/kernel/debug/ieee80211/phy0/mt76/regval
    #echo -n 0x820f705c > /sys/kernel/debug/ieee80211/phy0/mt76/regidx; echo -n 0x01ff0006
    > /sys/kernel/debug/ieee80211/phy0/mt76/regval
```

```
#echo -n 0x820f7060 > /sys/kernel/debug/ieee80211/phy0/mt76/regidx; echo -n 0x058000d0
> /sys/kernel/debug/ieee80211/phy0/mt76/regval
#echo -n 0x820f4120 > /sys/kernel/debug/ieee80211/phy0/mt76/regidx; echo -n 0x00000001
> /sys/kernel/debug/ieee80211/phy0/mt76/regval
#echo -n 0x820f50d0 > /sys/kernel/debug/ieee80211/phy0/mt76/regidx; echo -n 0x00000101
> /sys/kernel/debug/ieee80211/phy0/mt76/regval
#echo -n 0x830e705c > /sys/kernel/debug/ieee80211/phy0/mt76/regidx; echo -n 0x01ff0006
> /sys/kernel/debug/ieee80211/phy0/mt76/regval
#echo -n 0x830e7060 > /sys/kernel/debug/ieee80211/phy0/mt76/regidx; echo -n 0x058000d0
> /sys/kernel/debug/ieee80211/phy0/mt76/regval
#echo -n 0x830e4120 > /sys/kernel/debug/ieee80211/phy0/mt76/regidx; echo -n 0x00000001
> /sys/kernel/debug/ieee80211/phy0/mt76/regval
#echo -n 0x830e50d0 > /sys/kernel/debug/ieee80211/phy0/mt76/regidx; echo -n 0x00000101
> /sys/kernel/debug/ieee80211/phy0/mt76/regval
Starting log to eth
#mt76-test phy0 fwlog <PC IP> 15 &
```

3. **Runtime Logs. Capture per 5~10 seconds**

The following commands are executed continuously at runtime to collect various logs:

Ethernet

```
#cat /proc/mtketh/esw_cnt
#cat /proc/mtketh/dbg_regs
#cat /proc/net/nf_conntrack | grep OFFLOAD
#cat /sys/kernel/debug/mtk_ppe/entries
#cat /sys/kernel/debug/mtk_ppe/bind
```

WED

```
#cat /sys/kernel/debug/wed0/txinfo
#cat /sys/kernel/debug/wed0/rxinfo
```

Statistic

```
#cat /sys/kernel/debug/ieee80211/phy0/mt76/tx_drop_stats
#cat /sys/kernel/debug/ieee80211/phy0/mt76/rx_drop_stats
#cat /sys/kernel/debug/ieee80211/phy0/mt76/amsdu_info
```

Queue

```
#cat /sys/kernel/debug/ieee80211/phy0/mt76/token
#cat /sys/kernel/debug/ieee80211/phy0/mt76/tr_info
#cat /sys/kernel/debug/ieee80211/phy0/mt76/ple_info
#cat /sys/kernel/debug/ieee80211/phy0/mt76/pse_info
#cat /sys/kernel/debug/ieee80211/phy0/mt76/band0/mibinfo
#cat /sys/kernel/debug/ieee80211/phy0/mt76/band1/mibinfo
#cat /sys/kernel/debug/ieee80211/phy0/mt76/band2/mibinfo
```

4. **Save Firmware Log and Store Log Information in Files**

Version

```
#cat /sys/kernel/debug/ieee80211/phy0/mt76/fw_version
#cat /sys/kernel/debug/ieee80211/phy0/mt76/fw_wm_info
#free
```

BSS

```
#ifconfig
#cat /etc/config/wireless
#for i in `ls /var/run/hostapd-phy*.conf`; do echo $i; cat $i; done;
```

```
#iw dev
#cat /sys/kernel/debug/ieee80211/phy0/netdev:ap-mld-1/mt76_links_info
#cat /sys/kernel/debug/ieee80211/phy0/mt76/mat_table
#for i in `ls /sys/kernel/debug/ieee80211/phy0/mt76/band*/rmac_table`; do echo $i; cat
$i; done;
#for i in `ls /sys/kernel/debug/ieee80211/phy0/mt76/band*/agg_table`; do echo $i; cat
$i; done;
#cat /sys/kernel/debug/ieee80211/phy0/mt76/txop

Station
#cat /sys/kernel/debug/ieee80211/phy0/mt76/sta_info
#iw dev ap-mld-1 station dump
#iw dev ap-mld-1.sta1 station dump
#for i in `ls /sys/kernel/debug/ieee80211/phy0/netdev:ap-mld-
1/stations/*/mt76_links_info`; do echo $i; cat $i; done;
#for i in `ls /sys/kernel/debug/ieee80211/phy0/netdev:ap-mld-1/stations/*/link-
*/link_sta_info`; do echo $i; cat $i; done;

WTBL
#iwpriv ap-mld-1 show wtbl=1
#iwpriv ap-mld-1 show wtbl=2
#iwpriv ap-mld-1 show wtbl=3
#iwpriv ap-mld-1 show wtbl=4

Logs
#cat /sys/kernel/debug/tracing/trace
#echo > /sys/kernel/debug/tracing/trace # clear trace
#logread
#/etc/init.d/log restart # clear logread buffer

WED
#cat /sys/kernel/debug/wed0/cfg
```

9 SingleSKU Power Limit Table and Power Backoff Table

9.1 SingleSKU and Backoff Table Setting

SingleSKU can configure the TX power upper limit for different channels, bandwidths, and MCS rates. It can also provide more details than the Wireless Regulatory Domain, which will be introduced in the next chapter.

Backoff can configure power limits for different channel, tx, antenna, data stream, and beamform on/off status.

```
&slot0 {
    mt7915@0,0 {
        reg = <0x0000 0 0 0 0>;
    1   power-limits {
        2   r0 {
            3   regdomain = "fcc";
            4   txpower-5g {
                5   r1 {
                        channels = <36 48>;
                        txs delta = <0 0 0>;
                    6   rates-ofdm = <20 20 20 20 20 20 20 20>;
                        rates-mcs = <1 12 12 12 12 12 12 12 12 12>,
                                    <3 14 14 14 14 14 14 14 14 14>;
                        rates-ru = <3 11 11 11 11 11 11 11 11 11 11 11>,
                                   <4 8 8 8 8 8 8 8 8 8 8 8>;
                    7   paths-cck = <30 20 35 27>;
                        paths-ofdm = <36 33 39 27>;
                        paths-ofdm-bf = <28 21 39>;
                        paths-ru = <1 12 12 12 12 12 12 12 12 12>,
                                   …;
                        paths-ru-bf = <1 12 12 12 12 12 12 12 12 12>,
                                      …;

                    };
                    r2 {
                        channels = <100 181>;
                        txs delta = <0 0 0>;
                        ……
                    };
                };
            };
        };
    };
};
```

1. The table starts from “power-limits” node

2. A block for a regulatory domain or country
   - r0, r1, …

3. Choose a regd or country
   - regdomain = “fcc”
   - country = “US”

4. Choose a band
   - txpower-[256]g

5. A block of per-rate power limit for a channel range

6. Per-rate power setting with the unit of 0.5 dBm

7. Pre-path power setting with the unit of 0.5dBm

9.1.1 Step 1: Convert SingleSKU Table to Power-limits Node

For the dat file format:

- Open that file we offered with Excel.
- Change the values to suit your requirement.
- Save as a new file and set the file type as **CSV**; this will separate every value with a comma. And our parsing script only accepts this type of format.

Please refer to the scripts provided by MediaTek for the implementation of different models.
Parsing script execute environment:

- Window10 or further
- Python 3.9.10
    1) Numpy 1.26.2
    2) Parse 1.19.0

Version:

1.x.x	Support MT7915, MT7986
1.6.0	Support merge two sku tables
2.0.0	Support MT7996
2.1.0	Support both wifi6 and 7
2.2.0~2.3.0	Fix parsing error

```
# MT7615: perl single-sku.pl xxxx regdomain=FCC xxxx.dat
# MT7915: python.exe single-sku_${version}.py --sku mt7915d-sku.dat --scalar 2
# MT7986: python.exe single-sku-${version}.py --sku sku1-test-table-2g-5g-6g.dat --scalar 2
python.exe single-sku-${version}.py --sku2 Backoff_01.dat --scalar 2
# MT7996: python.exe single-sku-${version}.py --sku sku1-test-table-2g-5g-6g.dat --scalar 2 --wifi 7
python.exe single-sku-${version}.py --sku2 Backoff_01.dat --scalar 2 --wifi 7


# The script support translates two sku tables at the same time, need to specify two sku files
# note: MT7986 backoff table don't support 6G.
# python.exe single-sku-2.1.0.py --sku sku1-test-table-2g-5g.dat --sku2 Backoff_01.dat --scalar 2
# python.exe single-sku-2.1.0.py --sku Wifi7_SKU_table.dat --sku2 Wifi7_SKU_backoff_table.dat --scalar 2 --wifi 7

#Copy the execute result
txpower-2g {
        r0 {
                channels = <1 14>;
                txs_delta = <0 0 0>;
                rates-cck = <26 26 26 26>;
                rates-ofdm = <26 26 26 26 26 26 26 26>;
                rates-mcs =
                <4 26 26 26 26 26 26 26 26 26 26>;
                rates-ru =
                <7 26 26 26 26 26 26 26 26 26 26 26 26>;
        };
};
txpower-5g {
        r0 {
                channels = <184 196>;
                txs_delta = <0 0 0>;
                rates-ofdm = <24 24 24 24 24 24 24 24>;
                rates-mcs =
                <4 24 24 24 24 24 24 24 24 24 24>;
                rates-ru =
                <4 24 24 24 24 24 24 24 24 24 24 24 24>,
                <1 20 20 20 20 20 20 20 20 20 20 20 20>,
                <1 24 24 24 24 24 24 24 24 24 24 24 24>,
                <1 20 20 20 20 20 20 20 20 20 20 20 20>;
        };
        r1 {
                channels = <8 144>;
```

```
                txs_delta = <0 0 0>;
                rates-ofdm = <26 26 26 26 26 26 26 26>;
                rates-mcs =
                <4 26 26 26 26 26 26 26 26 26 26>;
                rates-ru =
                <7 26 26 26 26 26 26 26 26 26 26 26 26>;
        };
        r2 {
                channels = <149 181>;
                txs_delta = <0 0 0>;
                rates-ofdm = <32 32 32 32 32 32 32 32>;
                rates-mcs =
                <4 32 32 32 32 32 32 32 32 32 32>;
                rates-ru =
                <7 32 32 32 32 32 32 32 32 32 32 32 32>;
        };
};
```

9.1.2 Step 2: Check Wi-Fi Device Interface Name

- MT7615: pcie
- MT7915: slot0
- MT7986: wbsys
- MT7996: slot0

9.1.3 Step 3: Add the Power-limits Node to DTS File

Check the RFB requirement. Map the DTS file to this path.

- Path:
  - kernel 5.4: target/linux/mediatek/files-5.4/arch/arm64/boot/dts/mediatek
  - kernel 6.6: target/linux/mediatek/files-6.6/arch/arm64/boot/dts/mediatek

  mcs = <bandwidth idx, ht/vht mcs0, ht/vht mcs1…., ht/vht mcs9>

  bandwidth idx from BW20/40/80/160
- MT76 doesn't support VHT MCS 10 and 11; we will ignore these values.
- Ru = <ru idx, he mcs0, he mcs1…., he mcs 11>

  ru idx from 26/52/106/242/484/996*2,

  Ru 242 represents HE BW20

  Ru 484 represents HE BW40

  RU 996 represents HE BW80
- **Set a different regulatory domain or country by adding a new node under power-limits.**
- **SKU-The index is used to specify the SKU table runtime, starting from 1 to 255. 0 is reserved for default and legacy** and should not be used for this feature.

```
#Example 1: MT7915(target/linux/mediatek/dts/mt7622-rfb1-ubi.dts)
&slot0 {
    mt7915@0,0 {
```

```
        reg = <0x0000 0 0 0 0>;
        power-limits {
            r0 {
                regdomain = "fcc";
                sku-index = "1";
                txpower-5g {
                    r1 {
                        channels = <36 48>;
                        txs_delta = <0 0 0>;
                        rates-ofdm = <20 20 20 20 20 20 20 20>;
                        rates-mcs = <1 12 12 12 12 12 12 12 12 12 12>,
                        <3 14 14 14 14 14 14 14 14 14 14>;
                        rates-ru = <3 11 11 11 11 11 11 11 11 11 11 11>,
                        <4 8 8 8 8 8 8 8 8 8 8 8>;
                        paths-ofdm = <29 24 23 36>;
                        paths-ofdm-bf = <0 25 21 23>;
                        paths-ru = <1 20 32 37 33 38 37 21 25 39 30>,
                        <1 21 37 20 21 26 38 26 35 21 33>,
                        <1 38 27 33 32 34 34 24 23 25 25>,
                        <1 30 29 31 35 39 24 22 28 32 33>,
                        <1 24 38 36 26 33 23 36 31 28 37>,
                        <1 22 20 22 24 21 21 29 29 39 28>,
                        <1 24 20 39 23 33 36 24 37 36 38>;
                        paths-ru-bf = <1 38 35 32 31 32 24 28 26 28 34>,
                        <1 28 29 33 23 28 28 35 29 38 30>,
                        <1 21 37 33 34 29 36 26 38 26 22>,
                        <1 0 39 25 23 32 37 36 33 38 30>,
                        <1 0 34 29 23 20 20 39 30 35 23>,
                        <1 0 31 21 35 20 37 29 27 36 32>,
                        <1 0 23 27 24 20 25 22 22 34 30>;
                    };
                    r2 {
                        channels = <100 100>;
                        txs-delta = <0 0 0>;
                        rates-ofdm = <26 26 26 26 26 26 26 26>;
                        rates-mcs = <4 26 26 26 26 26 26 26 26 26 26>;
                        rates-ru = <7 26 26 26 26 26 26 26 26 26 26 26>;
                        paths-ofdm = <30 30 30 39>;
                        paths-ofdm-bf = <0 27 39 26>;
                        paths-ru = <1 27 25 28 37 21 32 33 24 35 20>,
                        <1 32 25 22 31 39 26 28 26 24 35>,
                        <1 34 36 24 26 28 33 20 29 30 25>,
                        <1 33 31 29 30 34 35 39 21 31 32>,
                        <1 34 34 26 37 24 35 35 21 28 32>,
                        <1 23 22 35 26 21 25 36 32 35 21>,
                        <1 20 35 26 34 38 22 31 28 36 23>;
                        paths-ru-bf = <1 31 24 21 38 28 26 24 20 37 34>,
                        <1 32 29 22 31 37 28 23 22 29 39>,
                        <1 34 24 26 37 37 20 25 27 30 39>,
                        <1 0 25 20 32 34 20 37 25 28 29>,
                        <1 0 27 36 33 31 32 20 26 20 37>,
                        <1 0 36 32 23 25 30 32 31 36 32>,
```

```
                        <1 0 25 24 27 39 32 31 39 23 36>;
                    };
                };
            };
        };
    };
};
```

```
#Example 2: MT7986b(target/linux/mediatek/files-5.4/arch/arm64/boot/dts/mediatek/mt7986b-2500wan-spim-nand-rfb.dts)
&wbsys {
    mediatek,mtd-eeprom = <&factory 0x0000>;
    status = "okay";
    power-limits {
        r0 {
            regdomain = "fcc";
            sku-index = "1";
            txpower-5g {
                r1 {
                    channels = <36 64>;
                    txs-delta = <0 0 0>;
                    rates-ofdm = <26 26 26 26 26 26 26 26>;
                    rates-mcs = <4 26 26 26 26 26 26 26 26 26 26>;
                    rates-ru = <7 26 26 26 26 26 26 26 26 26 26 26 26>;
                    paths-ofdm = <29 24 23 36>;
                    paths-ofdm-bf = <0 25 21 23>;
                    paths-ru = <1 20 32 37 33 38 37 21 25 39 30>,
                    <1 21 37 20 21 26 38 26 35 21 33>,
                    <1 38 27 33 32 34 34 24 23 25 25>,
                    <1 30 29 31 35 39 24 22 28 32 33>,
                    <1 24 38 36 26 33 23 36 31 28 37>,
                    <1 22 20 22 24 21 21 29 29 39 28>,
                    <1 24 20 39 23 33 36 24 37 36 38>;
                    paths-ru-bf = <1 38 35 32 31 32 24 28 26 28 34>,
                    <1 28 29 33 23 28 28 35 29 38 30>,
                    <1 21 37 33 34 29 36 26 38 26 22>,
                    <1 0 39 25 23 32 37 36 33 38 30>,
                    <1 0 34 29 23 20 20 39 30 35 23>,
                    <1 0 31 21 35 20 37 29 27 36 32>,
                    <1 0 23 27 24 20 25 22 22 34 30>;
                };
                r2 {
                    channels = <100 100>;
                    txs-delta = <0 0 0>;
                    rates-ofdm = <26 26 26 26 26 26 26 26>;
                    rates-mcs = <4 26 26 26 26 26 26 26 26 26 26>;
                    rates-ru = <7 26 26 26 26 26 26 26 26 26 26 26 26>;
                    paths-ofdm = <30 30 30 39>;
                    paths-ofdm-bf = <0 27 39 26>;
                    paths-ru = <1 27 25 28 37 21 32 33 24 35 20>,
                    <1 32 25 22 31 39 26 28 26 24 35>,
                    <1 34 36 24 26 28 33 20 29 30 25>,
```

```
                    <1 33 31 29 30 34 35 39 21 31 32>,
                    <1 34 34 26 37 24 35 35 21 28 32>,
                    <1 23 22 35 26 21 25 36 32 35 21>,
                    <1 20 35 26 34 38 22 31 28 36 23>;
                    paths-ru-bf = <1 31 24 21 38 28 26 24 20 37 34>,
                    <1 32 29 22 31 37 28 23 22 29 39>,
                    <1 34 24 26 37 37 20 25 27 30 39>,
                    <1 0 25 20 32 34 20 37 25 28 29>,
                    <1 0 27 36 33 31 32 20 26 20 37>,
                    <1 0 36 32 23 25 30 32 31 36 32>,
                    <1 0 25 24 27 39 32 31 39 23 36>;
                };
            };
        };
    };
};
```

```
#Example 3: MT7996a
• kernel 5.4(target/linux/mediatek/files-5.4/arch/arm64/boot/dts/mediatek/mt7988a-dsa-10g-spim-nand.dts)
• kernel 6.6(target/linux/mediatek/files-6.6/arch/arm64/boot/dts/mediatek/mt7988a-rfb-spim-nand-nmbm.dtso)
&slot0 {
    mt7996@0,0 {
        reg = <0x0000 0 0 0 0>;
        device_type = "pci";
        mediatek,mtd-eeprom = <&factory 0x0>;
        power-limits {
            r0 {
                regdomain = "fcc";
                sku-index = "1";
                txpower-2g {
                    r0 {
                        channels = <1 1>;
                            txs-delta = <0 0 0>;
                            rates-cck = <22 25 27 24>;
                            rates-ofdm = <27 22 21 29 28 24 35 29>;
                            rates-mcs =
                            <1 22 26 20 22 32 30 30 30 28 30>,
                            <1 26 30 34 30 20 22 40 22 32 28>,
                            <2 126 126 126 126 126 126 126 126 126 126>;
                            rates-ru =
                            <1 20 36 32 30 32 38 40 32 20 32 28 30>,
                            <1 36 32 24 34 30 38 20 30 34 20 26 26>,
                            <1 30 22 38 32 34 34 38 20 24 22 26 24>,
                            <1 22 26 20 22 32 30 30 30 28 30 32 20>,
                            <1 26 30 34 30 20 22 40 22 32 28 36 34>,
                            <2 126 126 126 126 126 126 126 126 126 126 126 126>;
                            rates-eht-ru =
                            <1 20 36 32 30 32 38 40 32 20 32 28 30 38 26 26 22>,
                            <1 36 32 24 34 30 38 20 30 34 20 26 26 34 20 28 28>,
                            <1 30 22 38 32 34 34 38 20 24 22 26 24 32 20 22 30>,
```

```
                        <1 22 26 20 22 32 30 30 30 28 30 32 20 24 22 24 20>,
                        <1 26 30 34 30 20 22 40 22 32 28 36 34 36 20 26 40>,
                        <3 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126>,
                        <1 38 26 28 34 38 28 24 26 34 24 24 38 38 26 30 40>,
                        <1 20 30 34 22 36 32 34 26 34 30 40 40 34 34 22 22>,
                        <6 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126>;
                        paths-cck = <30 20 35 27 25>;
                        paths-ofdm = <36 33 39 27 36>;
                        paths-ofdm-bf = <28 21 39 36>;
                        paths-ru =
                        <1 25 36 35 35 21 21 29 22 23 28 39 20 36 36 24>,
                        <1 26 34 28 24 20 28 40 36 24 22 30 36 34 32 26>,
                        <1 30 26 20 34 20 36 28 30 22 20 40 28 22 32 34>,
                        <1 34 32 26 40 38 34 36 40 36 32 38 22 40 34 40>,
                        <1 24 26 32 34 32 22 20 36 26 26 26 32 24 22 34>,
                        <1 28 32 20 26 26 20 20 32 38 38 36 24 28 34 26>,
                        <1 26 22 30 36 20 40 26 22 22 32 28 20 24 24 28>,
                        <9 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126>;
                        paths-ru-bf =
                        <1 20 22 38 36 25 24 30 23 22 21 28 21 29 30 32>,
                        <1 36 26 30 40 24 38 30 34 30 40 34 36 32 38 36>,
                        <1 30 28 34 32 34 40 30 40 26 24 32 36 30 22 34>,
                        <1 32 32 30 32 24 26 40 26 40 38 20 24 22 40 28>,
                        <1 38 30 32 34 38 22 22 38 22 32 26 30 22 32 36>,
                        <1 28 40 38 20 22 26 38 36 28 28 36 40 26 40 40>,
                        <1 22 24 36 34 32 24 30 28 32 36 40 32 26 34 32>,
                        <9 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126 126>;
                    };
                };
            };
            # for other domains or countries need to add new node.
            r1 {
                regdomain = "etsi";
                ......
            };
        };
    };
};

#Example 4: specify sku-index in hostapd config with the same country code: sku_idx=1 or 2
power-limits {
    r0 {
        country = "US";
        sku-index = "1";
        ......
    };
    r1 {
        country = "US";
        sku-index = "2";
        ......
    };
};
```

9.1.4 Step 4: Check txpower_sku Value

Enable the Access Point (AP) and set the correct domain.

```
#cat /sys/kernel/debug/ieee80211/phy0/mt76/txpower_sku
```

9.2 Useful TX Power-related Commands

9.2.1 Dump TX Power Info

1. Retrieve the basic TX power settings.

```
#cat /sys/kernel/debug/ieee80211/phy0/mt76/txpower_info

======================== BASIC INFO ========================
    Band Index: 0, Channel Band: 0
    PA Type: ePA
    LNA Type: eLNA
------------------------------------------------------------
    SKU: enable
    Percentage Control: disable
    Power Drop: 0 [dBm]
    Backoff: disable
    TX Front-end Loss:  0, 0, 0, 0
    RX Front-end Loss:  2, 2, 2, 2
    MU TX Power Mode:  auto
    MU TX Power (Auto / Manual): 34 / 0 [0.5 dBm]
    Thermal Compensation:  disable
    Theraml Compensation Value: 0
```

2. Dump the current TX Power sku table and BBP TX power

```
#cat sys/kernel/debug/ieee80211/phy0/mt76/txpower_sku

Phy 0 TX Power Table (Channel 1)
                          1m     2m     5m    11m
CCK (TMAC)           :    47     47     47     47
                          6m     9m    12m    18m    24m    36m    48m    54m
OFDM (TMAC)          :    47     47     47     47     47     47     47     47
                       mcs0   mcs1   mcs2   mcs3   mcs4   mcs5   mcs6   mcs7
HT20 (TMAC)          :    47     47     47     47     47     47     47     46
                       mcs0   mcs1   mcs2   mcs3   mcs4   mcs5   mcs6   mcs7  mcs32
HT40 (TMAC)          :    47     47     47     47     47     47     47     46     47
                       mcs0   mcs1   mcs2   mcs3   mcs4   mcs5   mcs6   mcs7   mcs8   mcs9  mcs10  mcs11
VHT20 (TMAC)         :    47     47     47     47     47     47     47     46     45     45      0      0
VHT40 (TMAC)         :    47     47     47     47     47     47     47     46     45     45      0      0
VHT80 (TMAC)         :    47     47     47     47     47     47     47     47     47     47      0      0
VHT160 (TMAC)        :    47     47     47     47     47     47     47     47     47     47      0      0
HE26 (TMAC)          :    47     47     47     47     47     47     47     46     45     45     44     44
HE52 (TMAC)          :    47     47     47     47     47     47     47     46     45     45     44     44
HE106 (TMAC)         :    47     47     47     47     47     47     47     46     45     45     44     44
HE242 (TMAC)         :    47     47     47     47     47     47     47     46     45     45     44     44
```

```
HE484 (TMAC)          :     47     47     47     47     47     47     47     46     45     45     44     44
HE996 (TMAC)          :     47     47     47     47     47     47     47     47     47     47     47     47
HE2x996 (TMAC)        :     47     47     47     47     47     47     47     47     47     47     47     47

                                                    :

ePA Gain: 0
Max Power Bound: 127
Min Power Bound: -128
BBP TX Power (target power from TMAC)  :     47 [0.5 dBm]
BBP TX Power (target power from RMAC)  :    127 [0.5 dBm]
BBP TX Power (TSSI module power input)  :     47 [0.5 dBm]
```

3. Dump the TxPower path table

```
cat /sys/kernel/debug/ieee80211/phy1/mt76/txpower_path

                       1T1S/2T1S/3T1S/4T1S/5T1S/2T2S/3T2S/4T2S/5T2S/3T3S/4T3S/5T3S/4T4S/5T4S/5T5S
CCK (TMAC)             :   48   48   48   48   48
OFDM (TMAC)            :   30   30   30   39   20
BF-OFDM (TMAC)         :   27   39   26   20
RU26 (TMAC)            :   27   25   28   37   24   21   32   33   34   24   35   38   20   37   34
BF-RU26 (TMAC)         :   31   24   21   38   32   28   26   24   25   20   37   22   34   31   39
RU52 (TMAC)            :   20   22   30   32   22   36   22   32   28   20   36   32   40   30   22
BF-RU52 (TMAC)         :   28   36   20   34   34   20   34   30   40   26   34   22   30   22   22
RU26_52 (TMAC)         :   34   32   38   40   36   32   24   30   36   24   30   26   40   32   32
BF-RU26_52 (TMAC)      :   30   34   28   30   40   34   36   32   38   40   36   38   26   22   38
RU106 (TMAC)           :   30   22   30   20   22   40   26   28   20   30   34   30   32   24   34
BF-RU106 (TMAC)        :   36   30   26   20   24   24   28   36   38   38   24   24   20   26   28
RU106_52 (TMAC)        :   32   30   24   22   26   40   38   28   20   30   38   34   28   26   24
BF-RU106_52 (TMAC)     :   30   40   26   34   40   38   20   30   32   40   40   20   22   26   34
BW20/RU242 (TMAC)      :   40   20   24   24   26   26   36   30   40   20   26   36   40   26   38
BF-BW20/RU242 (TMAC)   :   22   36   24   34   36   26   36   24   34   26   40   40   22   30   20
BW40/RU484 (TMAC)      :   26   22   34   20   26   40   22   40   26   38   30   38   22   32   30
BF-BW40/RU484 (TMAC)   :   28   24   32   20   28   20   26   26   40   30   38   26   28   30   20
RU242_484 (TMAC)       :   30   26   20   34   24   20   32   30   22   22   24   34   38   26   26
BF-RU242_484 (TMAC)    :   34   38   32   34   38   36   22   34   30   32   30   20   40   22   22
BW80/RU996 (TMAC)      :   30   30   30   20   36   36   28   36   28   38   22   32   38   32   40
BF-BW80/RU996 (TMAC)   :   30   30   30   22   40   38   28   40   24   36   24   26   24   40   32
RU484_996 (TMAC)       :   24   24   24   34   28   22   34   34   38   26   32   20   40   34   38
BF-RU484_996 (TMAC)    :   38   34   26   24   20   36   24   20   22   22   30   40   32   26   40
RU242_484_996 (TMAC)   :   20   22   28   28   24   38   34   38   20   38   20   40   40   28   26
BF-RU242_484_996 (TMAC):   26   24   30   34   38   30   32   36   22   36   34   20   28   32   32
BW160/RU996x2 (TMAC)   :   32   20   20   36   28   20   26   30   26   30   36   26   40   20   22
BF-BW160/RU996x2 (TMAC):   40   26   40   22   40   28   20   40   20   26   36   24   32   38   34
RU484_996x2 (TMAC)     :   48   48   48   48   48   48   48   48   48   48   48   48   48   48   48
BF-RU484_996x2 (TMAC)  :   48   48   48   48   48   48   48   48   48   48   48   48   48   48   48
RU996x3 (TMAC)         :   48   48   48   48   48   48   48   48   48   48   48   48   48   48   48
BF-RU996x3 (TMAC)      :   48   48   48   48   48   48   48   48   48   48   48   48   48   48   48
RU484_996x3 (TMAC)     :   48   48   48   48   48   48   48   48   48   48   48   48   48   48   48
BF-RU484_996x3 (TMAC)  :   20   40   30   34   28   34   38   30   34   36   24   30   38   36   20
BW320/RU996x4 (TMAC)   :   48   48   48   48   48   48   48   48   48   48   48   48   48   48   48
BF-BW320/RU996x4 (TMAC):   48   48   48   48   48   48   48   48   48   48   48   48   48   48   48
```

9.2.2 Change the TX Power Value

1. Set the power level drop.

```
#echo ${drop_level} > /sys/kernel/debug/ieee80211/phy0/mt76/txpower_level
```

The meaning of ${drop_level} is different for connac 2 and connac 3 chips.

For MT7915/MT7916/MT7986

Drop Level	Actual Power Drop (dB)
(0, 9]	12
(9, 15]	9
(15, 30]	6
(30, 60]	3
(60, 90]	1
Otherwise	0

For MT7996, the drop level value is the same as the real power drop value of 0.5 dBm.

2. Set the fixed power using the iw command

```
#iw dev phy0-ap0 set txpower fixed 2000
```

The term 'input power' refers to the specified 4TX power measured in millibels per milliwatt (mBm). Consequently, it is necessary to deduct the antenna combination gain when checking single-path tx power via the txpower_sku command. The antenna combination gain is listed in the following table.

NSS	Antenna Combination Gain (dB)
1T	0
2T	3
3T	4.6
4T	6

9.2.3 Enable/Disable sku table by runtime

The default power control from user space is disabled to follow the maximum power from eFuse. If you would like to enable power-relevant features (e.g., SingleSKU/iw set Tx Power), make sure to set 'sku_idx' to zero for a single SKU table or to any positive number for the index of SKU tables you want in the hostapd configuration to enable it.

You can double-check whether the value under the following path is 0.
Default sku is disabled.

```
# 0 is enabled.
echo 0 > /sys/kernel/debug/ieee80211/phy0/mt76/sku_disable
# Check current value
cat /sys/kernel/debug/ieee80211/phy0/mt76/sku_disable
```

[kernel 6.6]

```
# Set sku_idx >=0 to enable sku for the radio in hostapd config
sku_idx=0
```

9.3 Power enhancement (Only for Wi-Fi 6E)

9.3.1 LPI mode

MT76 support enhance management frame power under LPI mode, Due to sometimes the STA can't receive the management frames send from AP. Set hostapd config *(lpi_enable or lpi_psd)* to enable the feature and MT76 will check whether the country setting is US, KR, BR, CL, MY, if yes, the AP would compensate the sku rate table in firmware algo with the following table; else would change nothing. The debugfs table will not show the compensation, but it still can be observed that power bbp value is limited by the sku table OFDM 6m value plus compensate value.

Current Bandwidth (MHz)	Compensate Power [dBm]
40	3
80	6
160	9

Wi-Fi 6 vs. Wi-Fi 7 Support Readiness and Configuration

	Wi-Fi 6	Wi-Fi 7
Supported Version	Firmware build time is after **20231215**	MLO Beta Release **202408xx**
Usage	*lpi_enable=1*	*lpi_psd=1*

9.3.2 Beacon Duplicate Mode

When the beacon duplicate is enabled with hostapd config (*beacon_dup or* lpi_bcn_enhance), the AP may use a wider bandwidth to send management frames. If the power value is not changed, then the range to transmit the management frames will become shorter. To avoid this defect, MT76 will use a higher power value when sending management frames. In detail, MT76 will use the power in the sku table of the current bandwidth value to replace the OFDM power value when beacon duplicate is enabled.

The table shows the combined result of LPI and duplicate enhancement.

LPI_enable	is psd country	beacon_dup	Enhancement
0	0	0	No, use the original sku table
X	0	1	Enhance sku1/2
1	1	1	FW enhances SKU1; MT76 enhances SKU2

Wi-Fi 6 vs. Wi-Fi 7 Support Readiness and Configuration

	Wi-Fi 6	Wi-Fi 7
Supported Version	Firmware build time is after **20231215**	MLO Beta Release **202408xx**
Usage	*beacon_dup=1*	*lpi_bcn_enhance=1*

10 TX/RX Antenna

To set the TX/RX antenna, there are two ways to do it: UCI setting and the iw command.
For UCI settings, the commands remain the same for both Wi-Fi 6 (Connac 2) and Wi-Fi 7 (Connac 3, including single wiphy).
However, some user might not use UCI to setup the interface. For those users, please use iw command to configure the antenna and ensure that **the antenna is configured before the interface starts (i.e. before ifconfig <intf> up)**. Note that the TX and RX antenna masks should be the same due to hardware limitations.

10.1 UCI Settings

The following UCI commands set the TX and RX antenna mask of <radio_name> to <bitmap>.

```
uci set wireless.<radio_name>.txantenna=<bitmap>
uci set wireless.<radio_name>.rxantenna=<bitmap>
uci commit wireless
wifi
```

10.2 iw Command

The following iw command set the TX and RX antenna of <phyname> to <bitmap>.

```
iw phy <phyname> set antenna <bitmap> | all | <tx bitmap> <rx bitmap>
```

For single wiphy, please refer to Section 30.2 Single Wiphy Change to check out the new definition of the bitmap.

10.3 Check the Configured Antenna

To check the configured antenna bitmap, please enter the following command.

```
iw <phyname> info | grep Ant
```

The “Available Antennas” indicates the maximum capability of the TX/Rx path, and the “Configured Antennas” indicates the user-configured antenna bitmap.

```
root@OpenWrt:/# iw phy0 info | grep Ant
        Available Antennas: TX 0xf RX 0xf
        Configured Antennas: TX 0x1 RX 0x1
root@OpenWrt:/# iw phy1 info | grep Ant
        Available Antennas: TX 0xf RX 0xf
        Configured Antennas: TX 0xf RX 0xf
```

11 Wireless Regulatory Domain (wireless-regdb)

cfg80211 provides full regulatory support which is done through wireless-regdb and CRDA. The Linux wireless regulatory domain provides this framework as a safety net for regulatory considerations to account for changes and updates on the regulatory rules worldwide.

For MediaTek DBDC or TBTC devices, only one country setting is available across different bands. Otherwise, the DFS or channels might be overwritten unexpectedly.

An API allows drivers to export their own regulatory restrictions. The regulatory infrastructure consists of three major components:

- Kernel integration - cfg80211
- User space - Central Regulatory Domain Agent (CRDA)
- wireless-regdb

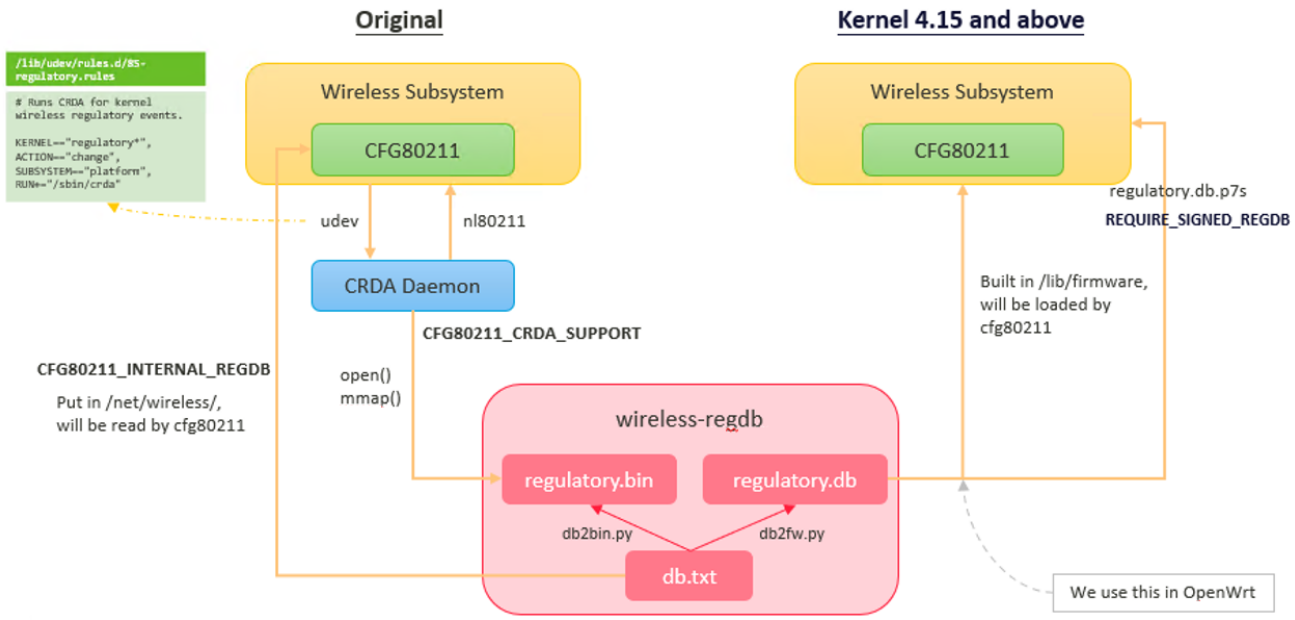


Figure 11-1. Wireless regulatory domain

The wireless-regdb utility functions as the regulatory database for Linux, producing regulatory.bin using data from db.txt.
Below is an example related to the country code of the US.

```
country US: DFS-FCC
        (2402 - 2472 @ 40), (30)
        (5170 - 5250 @ 80), (17), AUTO-BW
        (5250 - 5330 @ 80), (23), DFS, AUTO-BW
        (5490 - 5730 @ 160), (23), DFS
        (5735 - 5835 @ 80), (30)
```

Users can modify *wireless-regdb/db.txt* by different requirements

12 Wi-Fi Channel Management

MT76 chipsets provide several distinct radio frequency bands for use in Wi-Fi communications: 2.4 GHz, 5 GHz and 6 GHz. Each range is divided into a multitude of channels. In the standards, channels are numbered at 5 MHz spacing within a band, and the number linearly relates to the center frequency of the channel. Although channels are numbered at 5 MHz spacing, transmitters generally occupy at least 20 MHz, and standards allow for channels to be bonded together to form wider channels for faster throughput.

Countries apply their own regulations to the allowable channels, allowed users, and maximum power levels within these frequency ranges.

The following configuration specifies channel 6/149/37 separately for 2.4g/5g/6g interfaces, and specifies the country to 'US'.

```
config wifi-device 'radio0'
        option type 'mac80211'
        option path '11300000.pcie/pci0000:00/0000:00:00.0/0000:01:00.0'
        option band '2g'
        option country 'US'
        option noscan '1'
        option htmode 'EHT40'
        option channel '6'
        option disabled '0'

config wifi-iface 'default_radio0'
        option device 'radio0'
        option network 'lan'
        option mode 'ap'
        option encryption 'none'
        option macaddr '00:00:55:66:bd:10'
        option ssid 'mt76_2G'

config wifi-device 'radio1'
        option type 'mac80211'
        option path '11300000.pcie/pci0000:00/0000:00:00.0/0000:01:00.0+1'
        option band '5g'
        option htmode 'EHT80'
        option country 'US'
        option channel '149'
        option disabled '0'

config wifi-iface 'default_radio1'
        option device 'radio1'
        option network 'lan'
        option mode 'ap'
        option macaddr '00:01:55:66:bd:10'
        option ssid 'mt76_5G'
        option encryption 'psk2+ccmp'
        option key '12345678'

config wifi-device 'radio2'
        option type 'mac80211'
```

```
        option path '11300000.pcie/pci0000:00/0000:00:00.0/0000:01:00.0+2'
        option band '6g'
        option country 'US'
        option channel '37'
        option htmode 'EHT320-'
        option disabled '0'

config wifi-iface 'default_radio2'
        option device 'radio2'
        option network 'lan'
        option mode 'ap'
        option encryption 'sae'
        option macaddr '00:02:55:66:76:30'
        option key '12345678'
        option ssid 'mt76_6G'
```

Check *hostapd-phy.conf* for channel-related settings.

```
country_code=US
ieee80211d=1
ieee80211h=1
hw_mode=a
beacon_int=100
channel=36
ht_coex=0
ht_capab=[HT40+][..]
ieee80211ac=1
vht_oper_chwidth=2
vht_oper_centr_freq_seg0_idx=50
vht_capab=[VHT160][..]
ieee80211ax=1
he_oper_chwidth=2
he_oper_centr_freq_seg0_idx=50
ieee80211be=1
eht_oper_chwidth=2
eht_oper_centr_freq_seg0_idx=50
```

12.1 Channel Switch

12.1.1 Commands for Channel Switch

The provided commands can be used to switch channel or bandwidth when the access point (AP) is running.

1. Initiate the channel switch announcement

Connac2 Wi-Fi 6/ Connac3 Non-MLO	`hostapd_cli -i <devname> chan_switch <cs_count> <freq> [sec_channel_offset=offset] [center_freq1=cfreq1] [center_freq2=cfreq2] [bandwidth=MHz][blocktx] [ht|vht|he|eht] [skip_cac]`
Connac3 Wi-Fi 7 MLO	`hostapd_cli -i <devname> -l <link> chan_switch <cs_count> <freq> [sec_channel_offset=offset] [center_freq1=cfreq1] [center_freq2=cfreq2] [bandwidth=MHz][blocktx] [ht|vht|he|eht] [skip_cac]`

This command will initiate a channel switch announcement and will inform the connected station that the AP is changing the channel. The parameters specify the following information:

Parameter	Description
cs_count	The first mandatory parameter tells the AP to switch the channel after cs_count beacon frames.
freq	The second mandatory parameter specifies the frequency of the new channel (in MHz).
sec_channel_offset	Secondary channel offset for HT40 * 0 = HT40 disabled; * -1 = HT40 enabled, secondary channel below primary; * 1 = HT40 enabled, secondary channel above primary.
center_freq1	Segment 0 center frequency in MHz is valid for both HT and VHT.
cent_freq2	Segment 1 center frequency in MHz, non-zero only for bandwidth 80 and an 80+80 channel
bandwidth	Channel bandwidth, in MHz (20, 40, 80, 160).
blocktx	Whether to prevent the AP from transmitting packets to the internet when announcing a channel switch
ht|vht|he|eht	Whether to enable HT, VHT, HE, or EHT mode.
skip_cac	Whether to skip CAC when switching to the DFS channel

Here are some examples:

```
#2.4G CH5 EHT40-
hostapd_cli -i phy0-ap0 chan_switch 0 2432 center_freq1=2422 sec_channel_offset=-1  bandwidth=40 ht vht he eht
#2.4G CH1 EHT40+
hostapd_cli -i phy0-ap0 chan_switch 0 2412 center_freq1=2422 sec_channel_offset=1  bandwidth=40 ht vht he eht

#5G CH149, 80M
hostapd_cli -i phy1-ap0 chan_switch 10 5745 center_freq1=5775 sec_channel_offset=1  bandwidth=80 ht vht he eht
#5G CH40, 160M
hostapd_cli -i phy1-ap0 chan_switch 10 5200 center_freq1=5250 sec_channel_offset=-1  bandwidth=160 ht vht he eht

#6G CH65, EHT160
hostapd_cli -i phy2-ap0 chan_switch 0 6275 sec_channel_offset=1 center_freq1=6345 bandwidth=160 he eht
#6G CH69, EHT320
hostapd_cli -i phy2-ap0 chan_switch 0 6295 sec_channel_offset=-1 center_freq1=6425 bandwidth=320 he eht
```

Below are sampled UCI commands related to channel settings.

```
uci set wireless.radio1.hwmode=11a
uci set wireless.radio1.country=US
uci set wireless.radio1.channel=36
uci set wireless.radio1.htmode=HE80
uci commit wireless
wifi
```

12.1.2 Debug – Fail to Switch Channel

If channel config or command apply failed, you can check the hostapd log file to get the failure reason. The failure is generally caused by channel/cen_ch/bandwidth/sec_channel_offset unmatching, or channel not in support list.

Check the channel running info on each interface.

```
root@OpenWrt:/# iw dev
phy#2
        Interface phy2-ap0
                ifindex 9
                wdev 0x200000002
                addr 00:02:55:66:e8:29
                ssid mt76_6G
                type AP
                channel 65 (6275 MHz), width: 160 MHz, center1: 6345 MHz
                txpower 12.00 dBm
                multicast TXQ:
                        qsz-byt qsz-pkt flows   drops   marks   overlmt hashcol tx-bytes        tx-packets
                        0       0       0       0       0       0       0       0               0
phy#1
        Interface phy1-ap0
                ifindex 10
                wdev 0x100000002
                addr 00:01:55:66:d1:c9
                ssid mt76_5G
                type AP
                channel 149 (5745 MHz), width: 80 MHz, center1: 5775 MHz
                txpower 29.00 dBm
                multicast TXQ:
                        qsz-byt qsz-pkt flows   drops   marks   overlmt hashcol tx-bytes        tx-packets
                        0       0       0       0       0       0       0       0               0
phy#0
        Interface phy0-ap0
                ifindex 11
                wdev 0x2
                addr 00:00:55:66:d1:c9
                ssid mt76_2G
                type AP
                channel 6 (2437 MHz), width: 40 MHz, center1: 2447 MHz
                txpower 30.00 dBm
                multicast TXQ:
                        qsz-byt qsz-pkt flows   drops   marks   overlmt hashcol tx-bytes        tx-packets
                        0       0       196     0       0       0       0       34296           197
```

Check the supported channel list on each band:

```
root@OpenWrt:/# iw phy
Wiphy phy2
..
                Frequencies:
                        * 5955 MHz [1] (12.0 dBm)
                        * 5975 MHz [5] (12.0 dBm)
                        * 5995 MHz [9] (12.0 dBm)
                        * 6015 MHz [13] (12.0 dBm)
                        * 6035 MHz [17] (12.0 dBm)
                        * 6055 MHz [21] (12.0 dBm)
                        * 6075 MHz [25] (12.0 dBm)
                        * 6095 MHz [29] (12.0 dBm)
                        * 6115 MHz [33] (12.0 dBm)
                        * 6135 MHz [37] (12.0 dBm)
                        * 6155 MHz [41] (12.0 dBm)
                        * 6175 MHz [45] (12.0 dBm)
                        * 6195 MHz [49] (12.0 dBm)
                        * 6215 MHz [53] (12.0 dBm)
                        * 6235 MHz [57] (12.0 dBm)
                        * 6255 MHz [61] (12.0 dBm)
                        * 6275 MHz [65] (12.0 dBm)
                        * 6295 MHz [69] (12.0 dBm)
                        * 6315 MHz [73] (12.0 dBm)
                        * 6335 MHz [77] (12.0 dBm)
```

```
                        * 6355 MHz [81] (12.0 dBm)
                        * 6375 MHz [85] (12.0 dBm)
                        * 6395 MHz [89] (12.0 dBm)
                        * 6415 MHz [93] (12.0 dBm)
                        * 6435 MHz [97] (12.0 dBm)
                        * 6455 MHz [101] (12.0 dBm)
                        * 6475 MHz [105] (12.0 dBm)
                        * 6495 MHz [109] (12.0 dBm)
                        * 6515 MHz [113] (12.0 dBm)
                        * 6535 MHz [117] (12.0 dBm)
                        * 6555 MHz [121] (12.0 dBm)
                        * 6575 MHz [125] (12.0 dBm)
                        * 6595 MHz [129] (12.0 dBm)
                        * 6615 MHz [133] (12.0 dBm)
                        * 6635 MHz [137] (12.0 dBm)
                        * 6655 MHz [141] (12.0 dBm)
                        * 6675 MHz [145] (12.0 dBm)
                        * 6695 MHz [149] (12.0 dBm)
                        * 6715 MHz [153] (12.0 dBm)
                        * 6735 MHz [157] (12.0 dBm)
                        * 6755 MHz [161] (12.0 dBm)
                        * 6775 MHz [165] (12.0 dBm)
                        * 6795 MHz [169] (12.0 dBm)
                        * 6815 MHz [173] (12.0 dBm)
                        * 6835 MHz [177] (12.0 dBm)
                        * 6855 MHz [181] (12.0 dBm)
                        * 6875 MHz [185] (12.0 dBm)
                        * 6895 MHz [189] (12.0 dBm)
                        * 6915 MHz [193] (12.0 dBm)
                        * 6935 MHz [197] (12.0 dBm)
                        * 6955 MHz [201] (12.0 dBm)
                        * 6975 MHz [205] (12.0 dBm)
                        * 6995 MHz [209] (12.0 dBm)
                        * 7015 MHz [213] (12.0 dBm)
                        * 7035 MHz [217] (12.0 dBm)
                        * 7055 MHz [221] (12.0 dBm)
                        * 7075 MHz [225] (12.0 dBm)
                        * 7095 MHz [229] (12.0 dBm)
                        * 7115 MHz [233] (12.0 dBm)
Wiphy phy1
..
                Frequencies:
                        * 5180 MHz [36] (23.0 dBm)
                        * 5200 MHz [40] (23.0 dBm)
                        * 5220 MHz [44] (23.0 dBm)
                        * 5240 MHz [48] (23.0 dBm)
                        * 5260 MHz [52] (24.0 dBm) (radar detection)
                        * 5280 MHz [56] (24.0 dBm) (radar detection)
                        * 5300 MHz [60] (24.0 dBm) (radar detection)
                        * 5320 MHz [64] (24.0 dBm) (radar detection)
                        * 5500 MHz [100] (24.0 dBm) (radar detection)
                        * 5520 MHz [104] (24.0 dBm) (radar detection)
                        * 5540 MHz [108] (24.0 dBm) (radar detection)
                        * 5560 MHz [112] (24.0 dBm) (radar detection)
                        * 5580 MHz [116] (24.0 dBm) (radar detection)
                        * 5600 MHz [120] (24.0 dBm) (radar detection)
                        * 5620 MHz [124] (24.0 dBm) (radar detection)
                        * 5640 MHz [128] (24.0 dBm) (radar detection)
                        * 5660 MHz [132] (24.0 dBm) (radar detection)
                        * 5680 MHz [136] (24.0 dBm) (radar detection)
                        * 5700 MHz [140] (24.0 dBm) (radar detection)
                        * 5720 MHz [144] (24.0 dBm) (radar detection)
                        * 5745 MHz [149] (29.0 dBm)
                        * 5765 MHz [153] (29.0 dBm)
                        * 5785 MHz [157] (29.0 dBm)
                        * 5805 MHz [161] (29.0 dBm)
                        * 5825 MHz [165] (29.0 dBm)
                        * 5845 MHz [169] (27.0 dBm) (no IR)
                        * 5865 MHz [173] (27.0 dBm) (no IR)
                        * 5885 MHz [177] (27.0 dBm) (no IR)
Wiphy phy0
..
                Frequencies:
                        * 2412 MHz [1] (30.0 dBm)
                        * 2417 MHz [2] (30.0 dBm)
```

```
                        * 2422 MHz [3] (30.0 dBm)
                        * 2427 MHz [4] (30.0 dBm)
                        * 2432 MHz [5] (30.0 dBm)
                        * 2437 MHz [6] (30.0 dBm)
                        * 2442 MHz [7] (30.0 dBm)
                        * 2447 MHz [8] (30.0 dBm)
                        * 2452 MHz [9] (30.0 dBm)
                        * 2457 MHz [10] (30.0 dBm)
                        * 2462 MHz [11] (30.0 dBm)
                        * 2467 MHz [12] (disabled)
                        * 2472 MHz [13] (disabled)
                        * 2484 MHz [14] (disabled)
```

12.2 Auto Channel Selection (ACS)

ACS algorithms and implementations facilitate interfaces in determining the appropriate channel configuration automatically for initiating communication in any operational mode that emits signals on an AP or Mesh network.

The survey-based algorithm relies on the nl80211 survey API command to query the interface for channel active time, channel busy time, channel tx time, and noise. The active time consists of the amount of time the radio has spent on a channel. The busy time consists of the amount of time the radio has spent on the channel but noticed the channel was busy and could not initiate communication if it wanted to. The tx time is the amount of time the radio has spent on the channel transmitting data. The survey algorithm relies on these values to build an interference factor to give a sense of how much interference was detected on the channel and then picks the channel with the lowest interference factor.

The interference factor is defined as the ratio of the observed busy time over the time we spent on the channel, this value corresponds to:

```
10^(chan_nf/5) + (busy time - tx time) / (active time - tx time) * 2^(10^(chan_nf/10) + 10^(band_min_nf/10))
```

To enable hostapd to perform ACS when the interface is up, ensure that your hostapd.conf contains either:

```
channel=0
```

or

```
channel=acs_survey
```

Or use the UCI command to set the channel to zero.

```
root@OpenWrt:/# uci set wireless.radio0.channel=0
root@OpenWrt:/# uci commit wireless
root@OpenWrt:/# wifi
```

To limit the ACS channel list through hostapd config:

```
channel=0
freqlist=2412-2427,2447-2467
```

Or

```
channel=0
chanlist=1-4,8-12
```

Retrieve the log prompt and the final channel selection result:

```
hostapd: ACS: Automatic channel selection has started, this may take a bit
hostapd: phy0-ap0: interface state COUNTRY_UPDATE->ACS
hostapd: phy0-ap0: ACS-STARTED
```

```
netifd: radio0 (6611): Setup SMP Affinity
netifd: Wireless device 'radio0' is now up
hostapd: phy0-ap0: ACS-COMPLETED freq=2427 channel=4
root@OpenWrt:/# iw dev
phy#0
        Interface phy0-ap0
                ifindex 16
                wdev 0x4
                addr 00:00:55:66:87:aa
                ssid mt76_2G
                type AP
                channel 4 (2427 MHz), width: 40 MHz, center1: 2437 MHz
                txpower 30.00 dBm
                multicast TXQ:
                        qsz-byt qsz-pkt flows   drops   marks   overlmt hashcol tx-bytes        tx-packets
                        0       0       0       0       0       0       0       0               0
```

12.3 Scan/Survey

The functionality for scanning or surveying is provided by the 'iw' commands.

According to the DFS requirements for the ETSI domain (Section 4.2.6.1.4 in ETSI EN 301 893 V2.1.1), “If no radar was detected on an operating channel by the in-service monitoring but the RLAN stops operating on that channel, then the channel **becomes an available channel again.**”

This means that once a DFS channel is marked as available after the CAC, it will remain in the available state even when switching to a different operating channel. This is also referred to as **Pre-CAC allowed** in mac80211.

However, this rule is not explicitly stated in the FCC's DFS requirements. For the non-ETSI region, if the AP leaves the DFS channel, the state of the DFS channel will be reset to usable. Then, the AP must perform another CAC again on this DFS channel to operate on it after scanning is completed.

Therefore, for non-ETSI regulatory domains, scanning is prohibited in native mac80211 when the operating channel is a DFS channel. To relax the DFS restriction for scanning, please enter the following commands before triggering scanning or Remain on Channel (offchannel):

Relax the DFS restrictions:

```
echo 1 > /sys/kernel/debug/ieee80211/phyX/scan_dfs_relax
```

Check if the DFS restrictions are relaxed or not.

```
cat /sys/kernel/debug/ieee80211/phyX/scan_dfs_relax
```

Trigger the scan.

```
iw <devname> scan trigger [freq <freq>*] [duration <dur>] [ies <hex as 00:11:..>]
[lowpri,flush,ap-force] [ssid <ssid>*|passive]
```

Dump the current scan results. If -u is specified, print unknown data in the scan results.

```
iw <devname> scan dump [-u]
```

The following command is a combination of the previous two commands:

```
iw <devname> scan [-u] [freq <freq>*] [duration <dur>] [ies <hex as 00:11:..>]
[lowpri,flush,ap-force] [ssid <ssid>*|passive]
```

The optional parameters specify the following information:

Parameter	Description
freq	Scan on the given frequencies and probe for the given SSIDs (or wildcard if not given) unless passive scanning is requested.
duration	How long to listen on each channel, in TUs (1024 µs). If duration_mandatory is not set, this is the maximum dwell time and the actual dwell time may be shorter.
ies	Specified vendor IEs, which will be appended to the probe request when the scan mode is active.
lowpri	Scan request has low priority
flush	Flush cache before scanning.
ap-force	force a scan even if the interface is configured as AP and the beaconing has already been configured.
ssid	SSIDs to scan for (active scan only).
passive	Passive scan, only listen and do not send out probe request.

To trigger a scan on the 2.4 GHz AP:

```
root@OpenWrt:/# iw phy0-ap0 scan
BSS 8c:be:be:31:aa:c3(on phy0-ap0)
        last seen: 3069.132s [boottime]
        TSF: 1457355327907 usec (16d, 20:49:15)
        freq: 2412
        beacon interval: 100 TUs
        capability: ESS Privacy ShortSlotTime RadioMeasure (0x1411)
        signal: -70.00 dBm
        last seen: 728 ms ago
        Information elements from Probe Response frame:
        SSID: \xe5\xa4\xa7\xe7\xb1\xb3\xe5\x92\x8c\xe5\xb0\x8f\xe7\xb1\xb3@*\xe2\x99\xa0741/+++
        Supported rates: 1.0* 2.0* 5.5* 11.0* 18.0 24.0 36.0 54.0
        DS Parameter set: channel 1
        ERP: <no flags>
        ERP D4.0: <no flags>
        RSN:     * Version: 1
                 * Group cipher: TKIP
                 * Pairwise ciphers: CCMP TKIP
                 * Authentication suites: PSK
                 * Capabilities: 16-PTKSA-RC 1-GTKSA-RC (0x000c)
        Extended supported rates: 6.0 9.0 12.0 48.0
        HT capabilities:
                Capabilities: 0x19bc
                        HT20
                        SM Power Save disabled
                        RX Greenfield
                        RX HT20 SGI
                        TX STBC
                        RX STBC 1-stream
                        Max AMSDU length: 7935 bytes
                        DSSS/CCK HT40
                Maximum RX AMPDU length 65535 bytes (exponent: 0x003)
                Minimum RX AMPDU time spacing: 8 usec (0x06)
                HT RX MCS rate indexes supported: 0-15
                HT TX MCS rate indexes are undefined
        HT operation:
                 * primary channel: 1
                 * secondary channel offset: no secondary
                 * STA channel width: 20 MHz
                 * RIFS: 1
                 * HT protection: nonmember
                 * non-GF present: 0
                 * OBSS non-GF present: 1
                 * dual beacon: 0
                 * dual CTS protection: 0
                 * STBC beacon: 0
                 * L-SIG TXOP Prot: 0
                 * PCO active: 0
                 * PCO phase: 0
        Overlapping BSS scan params:
                 * passive dwell: 20 TUs
```

```
                * active dwell: 10 TUs
                * channel width trigger scan interval: 300 s
                * scan passive total per channel: 200 TUs
                * scan active total per channel: 20 TUs
                * BSS width channel transition delay factor: 5
                * OBSS Scan Activity Threshold: 0.25 %
        Extended capabilities:
                * HT Information Exchange Supported
                * Extended Channel Switching
                * Operating Mode Notification
        WPA:    * Version: 1
                * Group cipher: TKIP
                * Pairwise ciphers: CCMP TKIP
                * Authentication suites: PSK
        WMM:    * Parameter version 1
                * u-APSD
                * BE: CW 15-1023, AIFSN 3
                * BK: CW 15-1023, AIFSN 7
                * VI: CW 7-15, AIFSN 2, TXOP 3008 usec
                * VO: CW 3-7, AIFSN 2, TXOP 1504 usec
        RM enabled capabilities:
                Capabilities: 0x72 0x08 0x01 0x00 0x00
                        Neighbor Report
                        Beacon Passive Measurement
                        Beacon Active Measurement
                        Beacon Table Measurement
                        Statistics Measurement
                        AP Channel Report
                Nonoperating Channel Max Measurement Duration: 0
                Measurement Pilot Capability: 0
…
```

The operating channel should be left and proceeded to the designated channel for a temporary period.

```
iw <devname> offchannel <freq> <duration>
```

To go to channel 8 and dwell for 200 TUs and then dump all channel stats.

```
root@OpenWrt:/# iw phy0-ap0 offchannel 2447 200
root@OpenWrt:/# iw phy0-ap0 scan dump
root@OpenWrt:/# iw phy0-ap0 survey dump
Survey data from phy0-ap0
        frequency:                      2412 MHz [in use]
        noise:                          -71 dBm
        channel active time:            3362785 ms
        channel busy time:              1879668 ms
        channel receive time:           1861373 ms
        channel BSS receive time:       1660853 ms
        channel transmit time:          123 ms
Survey data from phy0-ap0
        frequency:                      2417 MHz
        noise:                          -73 dBm
        channel active time:            111 ms
        channel busy time:              30 ms
        channel receive time:           30 ms
        channel BSS receive time:       22 ms
        channel transmit time:          0 ms
Survey data from phy0-ap0
        frequency:                      2422 MHz
        noise:                          -69 dBm
        channel active time:            111 ms
        channel busy time:              39 ms
        channel receive time:           39 ms
        channel BSS receive time:       21 ms
        channel transmit time:          0 ms
Survey data from phy0-ap0
        frequency:                      2427 MHz
        noise:                          -79 dBm
        channel active time:            111 ms
        channel busy time:              54 ms
        channel receive time:           54 ms
        channel BSS receive time:       29 ms
        channel transmit time:          0 ms
Survey data from phy0-ap0
```

```
        frequency:                      2432 MHz
        noise:                          -86 dBm
        channel active time:            111 ms
        channel busy time:              37 ms
        channel receive time:           37 ms
        channel BSS receive time:       25 ms
        channel transmit time:          0 ms
Survey data from phy0-ap0
        frequency:                      2437 MHz
        noise:                          -73 dBm
        channel active time:            115 ms
        channel busy time:              54 ms
        channel receive time:           54 ms
        channel BSS receive time:       49 ms
        channel transmit time:          0 ms
Survey data from phy0-ap0
        frequency:                      2442 MHz
        noise:                          -75 dBm
        channel active time:            111 ms
        channel busy time:              37 ms
        channel receive time:           37 ms
        channel BSS receive time:       17 ms
        channel transmit time:          0 ms
Survey data from phy0-ap0
        frequency:                      2447 MHz
        noise:                          -86 dBm
        channel active time:            203 ms
        channel busy time:              92 ms
        channel receive time:           92 ms
        channel BSS receive time:       48 ms
        channel transmit time:          0 ms
Survey data from phy0-ap0
        frequency:                      2452 MHz
        noise:                          -61 dBm
        channel active time:            111 ms
        channel busy time:              52 ms
        channel receive time:           52 ms
        channel BSS receive time:       23 ms
        channel transmit time:          0 ms
Survey data from phy0-ap0
        frequency:                      2457 MHz
        noise:                          -81 dBm
        channel active time:            111 ms
        channel busy time:              50 ms
        channel receive time:           50 ms
        channel BSS receive time:       41 ms
        channel transmit time:          0 ms
Survey data from phy0-ap0
        frequency:                      2462 MHz
        noise:                          -71 dBm
        channel active time:            127 ms
        channel busy time:              71 ms
        channel receive time:           71 ms
        channel BSS receive time:       66 ms
        channel transmit time:          0 ms
```

12.4 Frequency Limit (dt-bindings: Common IEEE 802.11 Frequency Limit Properties)

ieee80211-freq-limit is a list of supported frequency ranges in KHz. This can be used for devices in given configuration that support fewer channels than normal. A chipset can support a wide range of wireless bands but it is limited by the antennas or a power amplifier used. An example case is a tri-band wireless router with two identical chipsets used for two different 5GHz sub-bands. Incorrect uses can lead to malfunction or noticeable performance decreases.

```
pcie@0,0 {
        reg = <0x0000 0 0 0 0>;
        wifi@0,0 {
```

```
                reg = <0x0000 0 0 0 0>;
                ieee80211-freq-limit = <2402000 2482000>,
                                       <5170000 5250000>;
        };
};
```

13 Dynamic Frequency Selection (DFS)

In many countries, operating Wi-Fi devices on some or all channels in the 5GHz band requires radar detection and DFS (Dynamic Frequency Selection). If a channel is defined in the wireless configuration that requires DFS according to the country regulations, the 5GHz radio device then does not start up unless the firmware image is able to provide the DFS support (i.e., it is both included and enabled). More technical details of the Linux implementation can be found here.

DFS works as follows on Linux that the driver detects radar pulses and reports to nl80211 where the information is processed. If a series of pulses matches one of the defined radar patterns, it is then reported to the user-space application (e.g., hostapd) which in turn reacts by switching to another channel. The following configuration specifies channel 104 which needs DFS support as implicitly stated with country code DE.

```
config   wifi-device              'radio0'
         option   type            'mac80211'
         option   channel         '104'
         option   hwmode          '11a'
         option   path            'pci0000:00/0000:00:00.0'
         option   htmode          'HT20'
         option   country         'DE'

config   wifi-iface
         option   device          'radio0'
         option   network         'lan'
         option   mode            'ap'
         option   ssid            'OpenWrt'
         option   encryption      'none'
```

The country code on the Wi-Fi card can be obtained using the following command.

```
iw reg get
```

To enhance clarity, review hostapd-phy.conf to verify the presence of specified values and ensure that the country code is configured.

```
country_code=DE
ieee80211n=1
ieee80211d=1
ieee80211h=1
hw_mode=a
```

If the radar detection is working, DFS channels shall show up as below (the example here is for Belgium, *iw phy1 info* output trimmed).

```
Frequencies:
* 5220 MHz [44] (17.0 dBm)
* 5240 MHz [48] (17.0 dBm)
* 5260 MHz [52] (20.0 dBm) (radar detection)
DFS state: usable (for 2155257 sec)
DFS CAC time: 60000 ms
* 5280 MHz [56] (20.0 dBm) (radar detection)
DFS state: usable (for 2155257 sec)
DFS CAC time: 60000 ms
```

Note that when DFS is on, there is a delay before the interface is enabled (e.g., after reboot). During this time period[3] (often 60 seconds and determined by local regulations) which is to detect the presence of other signals in the channel (Channel Availability Check Time), Luci shall report that the interface is disabled. This process can be monitored with:

```
logread -f
```

Once DFS is triggered, it randomly switches to the next available channel. See the hostapd log below.

```
hostapd: wlan1: DFS-CAC-START freq=5500 chan=100 sec_chan=1, width=1, seg0=106,
seg1=0, cac_time=60s
hostapd: wlan1: DFS-CAC-COMPLETED success=0 freq=5500 ht_enabled=0 chan_offset=0
chan_width=3 cf1=5530 cf2=0
hostapd: wlan1: DFS-RADAR-DETECTED freq=5500 ht_enabled=0 chan_offset=0
chan_width=3 cf1=5530 cf2=0
hostapd: wlan1: DFS-NEW-CHANNEL freq=5660 chan=132 sec_chan=1
hostapd: wlan1: interface state DFS->DFS
hostapd: wlan1: DFS-CAC-START freq=5660 chan=132 sec_chan=1, width=1, seg0=138,
seg1=0, cac_time=60s
```

13.1 Zero-Wait DFS

Enable in /etc/config/wireless configuration

```
config wifi-device 'radio0'
        option type 'mac80211'
        option path '11280000.pcie/pci0000:00/0000:00:00.0/0000:01:00.0'
        option channel '52'
        option band '5g'
        option htmode 'HE80'
        option country 'US'
        option background_radar '1'

config wifi-iface 'default_radio0'
        option device 'radio0'
        option network 'lan'
        option mode 'ap'
        option ssid 'OpenWrt_5G'
        option encryption 'none'
```

[3] The name of the web application.

Check hostapd configuration

```
vi /var/run/hostapd-phy0.conf
…
enable_background_radar=1
```

Check hostapd log

During the initial 60 seconds, the 5 GHz band operates on a non-DFS channel, while the background radar performs Channel Availability Check (CAC) in the chosen DFS channel.

```
root@OpenWrt:/# iwinfo
wlan0     ESSID: "OpenWrt_5G"
          Access Point: 00:0C:43:2A:94:DD
          Mode: Master  Channel: 149 (5.745 GHz)
          Center Channel 1: 155 2: unknown
          Tx-Power: 27 dBm  Link Quality: unknown/70
          Signal: unknown  Noise: -92 dBm
          Bit Rate: unknown
          Encryption: none
          Type: nl80211  HW Mode(s): 802.11bgnacax
          Hardware: 14C3:7915 14C3:7915 [MediaTek MT7915E]
          TX power offset: none
          Frequency offset: none
          Supports VAPs: yes  PHY name: phy0
```

daemon.notice hostapd: wlan0: AP-ENABLED

daemon.notice hostapd: wlan0: DFS-CAC-START freq=5260 chan=52 chan_offset=0 width=3 seg0=5290 seg1=0 cac_time=60s (background)

After 60 seconds, 5 GHz back to the selected DFS channel.

```
hostapd: wlan0: DFS-CAC-COMPLETED success=1 freq=5260 ht_enabled=0 chan_offset=0
chan_width=3 cf1=5290 cf2=0
hostapd: wlan0: DFS-NEW-CHANNEL freq=5260 chan=52 sec_chan=1
hostapd: wlan0: DFS-CAC-START freq=5500 chan=100 chan_offset=0 width=3 seg0=5530
seg1=0 cac_time=60s (background)
```

Trigger a Radar Event by using the manual mode

Please refer to DFS Useful Commands.

13.2 Useful Commands

Some useful commands for debugging Distributed File System (DFS) are provided below.

1. Get the current DFS channel status

```
cat /sys/kernel/debug/ieee80211/phyX/dfs_status
```

The command displays the DFS state of all DFS channels in the current band. If the PHY is not in the 5G band, then it will... would be empty.

The description of each DFS state defined in cfg80211/mac80211 is listed in the following table.

DFS State	Description
Usable	The channel can be used, but a channel availability check (CAC) must be performed before using it for AP or IBSS.
Unavailable	A radar has been detected on this channel, and it is therefore marked as not available. It must wait for a non-occupancy period (NOP) before becoming "usable."
Available	The channel has been CAC checked and is available to watch.

Here are some examples:

a. When Wi-Fi is down (or Wi-Fi is up but the operating channel is not in the DFS channel)

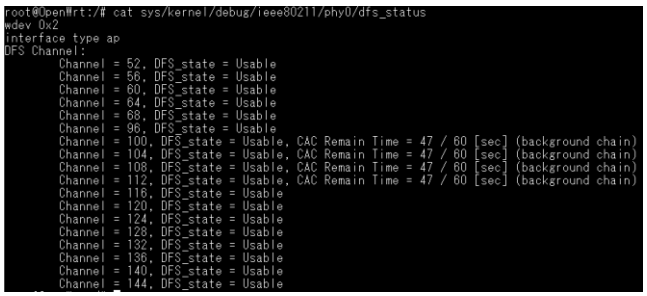
```
root@OpenWrt:/# cat /sys/kernel/debug/ieee80211/phy0/dfs_status
DFS Channel:
        Channel = 52, DFS_state = Usable
        Channel = 56, DFS_state = Usable
        Channel = 60, DFS_state = Usable
        Channel = 64, DFS_state = Usable
        Channel = 68, DFS_state = Usable
        Channel = 96, DFS_state = Usable
        Channel = 100, DFS_state = Usable
        Channel = 104, DFS_state = Usable
        Channel = 108, DFS_state = Usable
        Channel = 112, DFS_state = Usable
        Channel = 116, DFS_state = Usable
        Channel = 120, DFS_state = Usable
        Channel = 124, DFS_state = Usable
        Channel = 128, DFS_state = Usable
        Channel = 132, DFS_state = Usable
        Channel = 136, DFS_state = Usable
        Channel = 140, DFS_state = Usable
        Channel = 144, DFS_state = Usable
```

b. Wi-Fi is up (channel = 100, 80 Hz) during the CAC

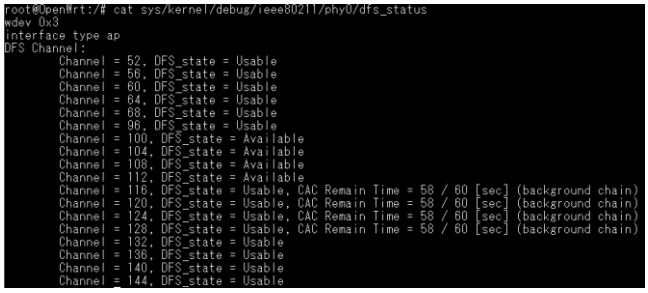
```
root@OpenWrt:/# cat /sys/kernel/debug/ieee80211/phy0/dfs_status
DFS Channel:
        Channel = 52, DFS_state = Usable
        Channel = 56, DFS_state = Usable
        Channel = 60, DFS_state = Usable
        Channel = 64, DFS_state = Usable
        Channel = 68, DFS_state = Usable
        Channel = 96, DFS_state = Usable
        Channel = 100, DFS_state = Usable, CAC Remain Time = 56 [sec]
        Channel = 104, DFS_state = Usable, CAC Remain Time = 56 [sec]
        Channel = 108, DFS_state = Usable, CAC Remain Time = 56 [sec]
        Channel = 112, DFS_state = Usable, CAC Remain Time = 56 [sec]
        Channel = 116, DFS_state = Usable
        Channel = 120, DFS_state = Usable
        Channel = 124, DFS_state = Usable
        Channel = 128, DFS_state = Usable
        Channel = 132, DFS_state = Usable
        Channel = 136, DFS_state = Usable
        Channel = 140, DFS_state = Usable
        Channel = 144, DFS_state = Usable
```

c. Wi-Fi is up (channel = 100, 80 Hz & **ZWDFS enabled**) during CAC

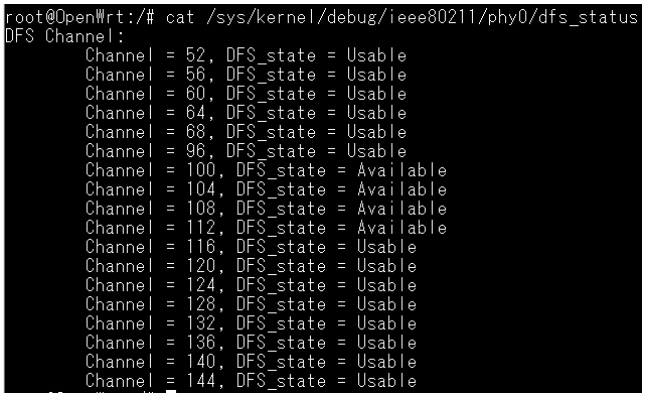
i. The background chain is listening to channel 100 (CAC), and the AP channel is operating on a non-DFS channel.



ii. The CAC of the background chain is finished, and AP switches to channel 100, 80Hz. The background chain starts to listen to another DFS channel (randomly picked).



d. Wi-Fi is up (channel = 100, 80 Hz) after the CAC count down



e. Wi-Fi is up (channel = 100, 80 Hz), but radar is detected in channel = 100, 80 Hz

f. Multiple interfaces (including STA & AP)

```
root@OpenWrt:/sys/kernel/debug/ieee80211/phy1# cat dfs_status
wdev 0x4
interface type ap
DFS Channel:
        Channel = 52, DFS_state = Usable
        Channel = 56, DFS_state = Usable
        Channel = 60, DFS_state = Usable
        Channel = 64, DFS_state = Usable
        Channel = 68, DFS_state = Usable
        Channel = 96, DFS_state = Usable
        Channel = 100, DFS_state = Usable, CAC Remain Time = 43 [sec]
        Channel = 104, DFS_state = Usable, CAC Remain Time = 43 [sec]
        Channel = 108, DFS_state = Usable, CAC Remain Time = 43 [sec]
        Channel = 112, DFS_state = Usable, CAC Remain Time = 43 [sec]
        Channel = 116, DFS_state = Usable
        Channel = 120, DFS_state = Usable
        Channel = 124, DFS_state = Usable
        Channel = 128, DFS_state = Usable
        Channel = 132, DFS_state = Usable
        Channel = 136, DFS_state = Usable
        Channel = 140, DFS_state = Usable
        Channel = 144, DFS_state = Usable
wdev 0x3
interface type station
DFS Channel:
        Channel = 52, DFS_state = Usable
        Channel = 56, DFS_state = Usable
        Channel = 60, DFS_state = Usable
        Channel = 64, DFS_state = Usable
        Channel = 68, DFS_state = Usable
        Channel = 96, DFS_state = Usable
        Channel = 100, DFS_state = Usable
        Channel = 104, DFS_state = Usable
        Channel = 108, DFS_state = Usable
        Channel = 112, DFS_state = Usable
        Channel = 116, DFS_state = Usable
        Channel = 120, DFS_state = Usable
        Channel = 124, DFS_state = Usable
        Channel = 128, DFS_state = Usable
        Channel = 132, DFS_state = Usable
        Channel = 136, DFS_state = Usable
        Channel = 140, DFS_state = Usable
        Channel = 144, DFS_state = Usable
```

2. Skip CAC

(i) *Skip all the AP's CAC*

```
echo 1 > /sys/kernel/debug/ieee80211/phyX/dfs_skip_cac
```

(ii) *Skip background chain's (ZWDFS) CAC*

```
echo 2 > /sys/kernel/debug/ieee80211/phyX/dfs_skip_cac
```

(iii) *Skip all CAC (including AP & background chain)*

```
echo 3 > /sys/kernel/debug/ieee80211/phyX/dfs_skip_cac
```

3. Skip All NOP

```
echo 1 > /sys/kernel/debug/ieee80211/phyX/dfs_skip_nop
```

4. Trigger a Radar Event in Manual Mode

Note: Only chips with Dedicated Rx can support ZW-DFS

Wi-Fi 6:

```
echo 1 > /sys/kernel/debug/ieee80211/phyX/mt76/radar_trigger (5GHz)
echo 2 > /sys/kernel/debug/ieee80211/phyX/mt76/radar_trigger (Dedicated Rx)
```

Wi-Fi 7:

```
echo 1 > /sys/kernel/debug/ieee80211/phy0/mt76/band1/radar_trigger (5GHz)
echo 2 > /sys/kernel/debug/ieee80211/phy0/mt76/band1/radar_trigger (Dedicated Rx)
```

5. Disable the channel switch after receiving radar

```
# The channel switches normally after receiving radar for both AP and dedicated RX
hostapd_cli -i <interface> raw DFS_DETECT_MODE 0
# Disable channel switch after receiving radar for AP
```

```
hostapd_cli -i <interface> raw DFS_DETECT_MODE 1
# Disable channel switch after receiving radar for dedicated RX (background chain, ZWDFS)
hostapd_cli -i <interface> raw DFS_DETECT_MODE 2
# Disable channel switch after receiving radar for both AP and dedicated RX
hostapd_cli -i <interface> raw DFS_DETECT_MODE 3
```

6. Background radar off-chain control

```
# Get the status of background radar off-chain
hostapd_cli -i <interface> [-l <link_id>] get_offchain
# Set the channel of the background radar off-chain and trigger CAC on it
hostapd_cli -i <interface> [-l <link_id>] set_offchain chan=<control channel>
[bandwidth=<bw>] [is_temp_ch=<0/1>] [expand=<0/1>]
```

Notes

1. Please make sure the interface contains 5G radios. If it is an AP MLD, then link id is required.
2. If bandwidth is not specified in the command, then the bandwidth of the main radio chain is used for the setting.
3. If the **is_temp_ch** flag is set, then after the background radar completes the CAC of the specified channel, the main radio chain will switch to that channel. Additionally, the background radar will select another channel to perform CAC on.
4. If the **expand** flag is set, then after the background radar completes the CAC of the specified channel, the main radio chain will try to combine the channel of background radar and itself to expand its channel width. If the expanding process fails, then the main radio chain will switch to the channel of background radar (just like is_temp_ch). As usual, the background radar will select another channel to perform CAC on.

7. Background radar standalone mode

```
vi /var/run/hostapd-phyX.conf
…
background_auto_ctrl=0
```

To enable background radar standalone mode, please set the background_auto_ctrl flag in hostapd configuration to 0 (default is 1).

When the standalone mode is enabled, then background radar off-chain can only be controlled by the hostapd_cli command above. Users can decide the behavior of the background radar based on their own preference. For example, if radar is detected on the background radar, the background radar will become inactive instead of switching to a randomly selected DFS channel to perform CAC on.

13.3 Debug – Fail to Switch Channel

If there is no DFS-NEW-CHANNEL message in the log, it means that hostapd failed to select a new channel.

```
hostapd: wlan1: DFS-RADAR-DETECTED freq=5500 ht_enabled=1 chan_offset=0
chan_width=1 cf1=5500 cf2=0
hostapd: dfs_downgrade_bandwidth: no DFS channels left, waiting for NOP to finish
```

hostapd checks available channels from the chanlist parameters which are specified in the hostapd configuration. If the channel does not exist in the chanlist, DFS shall fail to switch channels. This parameter can be modified by users.

```
# Channel list restriction. This option allows hostapd to select one of the
# provided channels when a channel should be automatically selected.
# Channel list can be provided as range using hyphen ('-') or individual
```

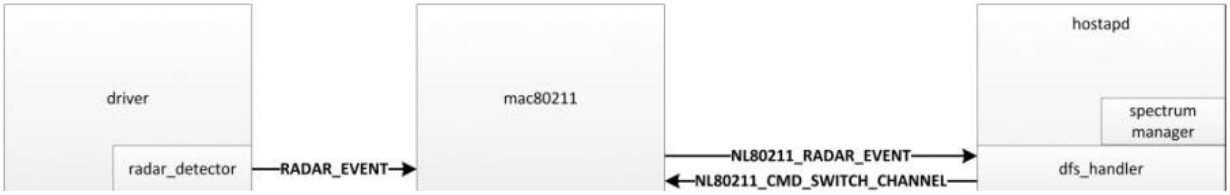
```
# channels can be specified by space (' ') separated values
# Default: all channels allowed in selected hw_mode
#chanlist=100 104 108 112 116
#chanlist=1 6 11-13
```

13.4 Debug – Fail to Start CAC

Selecting a channel that requires DFS in the country and enabling HT40 may result error **DFS start_dfs_cac() failed** (visible with *logread*). To resolve the error, change the configuration to HT20.

```
Configuration file: /var/run/hostapd-phy1.conf
wlan1: interface state UNINITIALIZED->COUNTRY_UPDATE
wlan1: interface state COUNTRY_UPDATE->HT_SCAN
wlan1: interface state HT_SCAN->DFS
wlan1: DFS-CAC-START freq=5680 chan=136 sec_chan=-1, width=0, seg0=0, seg1=0,
cac_time=60s
DFS start_dfs_cac() failed, -1
Interface initialization failed
wlan1: interface state DFS->DISABLED
wlan1: AP-DISABLED
hostapd_free_hapd_data: Interface wlan1 wasn't started
```

MT76 is a hardware-based detector driver that reports the occurrence of a radar pattern on a channel. A radar event is passed to MAC80211 and forwarded to hostapd, which handles DFS channel states based on the radar events and manages the spectrum (including initiating CSA, moving the channel, etc.).



14 Multiple BSS and MAC Address Rule

The default access point (AP) MAC address is verified during the initial boot-up to determine if the board is calibrated. If so, the default AP MAC address will utilize the EEPROM MAC address, which can be verified using a specific command.

```
cat /sys/class/ieee80211/phyX/macaddress
```

Otherwise, we will make a random mac address in UCI to prevent the same mac addresses from conflicting.
User can config mac address manually by uci option, and the mac address would consider have highest priority to apply. However, if config mbss and set mac address to some (not all) interfaces may counter same mac address issue. Therefore, please **set ALL interfaces or don't** set mac address in uci option.

For different situations, there are different rules. These rules of legacy mbss are applied to interfaces that do not set an UCI MAC address.

Wi-Fi 7 BSP MAC Address Rule

Radio	Address	Legacy mbss	11vmbss 5 bsses
0	**Frist BSS of radio 0** Usually MLD#1 link 0 address	Use **physical MAC address** (base_addr) 00:xx:xx:xx:xx:1e (radio 0 idx 0)	

	# MLDs
	MLD#2..16
	radio 0 link address
	or
	# legacy APs/STAs
	phy0.0-ap1,
	phy0.0-ap2 ...
		// addr = base_addr
		// addr[5] ^= idx
		00:xx:xx:xx:xx:1f (idx 1),
		00:xx:xx:xx:xx:1c (idx 2),
		00:xx:xx:xx:xx:1d (idx 3),
		00:xx:xx:xx:xx:1a (idx 4),
		00:xx:xx:xx:xx:1b (idx 5), …
			// Follow the rules in 14.3.2
			00:xx:xx:xx:xx:1f (idx 1),
			00:xx:xx:xx:xx:18 (idx 2),
			00:xx:xx:xx:xx:19 (idx 3),
			00:xx:xx:xx:xx:1a (idx 4)

1	**Frist BSS of radio 1**	// addr = base_addr
		// addr[5] ^= (idx + radio idx * 16)
		// addr[5] ^= 16
		00:xx:xx:xx:xx:2e (radio 1 idx 0)	

	# MLDs
	MLD#2..16
	radio 1 link address
	or
	# legacy APs/STAs
	phy0.1-ap1,
	phy0.1-ap2 ...
		// addr = base_addr
		// addr[5] ^= (idx + radio idx * 16)
		// addr[5] ^= (idx + 16)
		
		00:xx:xx:xx:xx:0f(idx 1),
		00:xx:xx:xx:xx:0c(idx 2),
		00:xx:xx:xx:xx:0d(idx 3),
		00:xx:xx:xx:xx:0a(idx 4),
		00:xx:xx:xx:xx:0b(idx 5), …
			// Follow the rules in 14.3.2
			00:xx:xx:xx:xx:2f (idx 1),
			00:xx:xx:xx:xx:28 (idx 2),
			00:xx:xx:xx:xx:29 (idx 3),
			00:xx:xx:xx:xx:2a (idx 4)

2	**Frist BSS of radio 2**	// addr = base_addr
		// addr[5] ^= (idx + radio idx * 16)
		// addr[5] ^= 32
		00:xx:xx:xx:xx:3e (radio 2 idx 0)	

	# MLDs
	MLD#2..16
	radio 2 link address
	or
	# legacy APs/STAs
		// addr = base_addr
		// addr[5] ^= (idx + radio idx * 16)
		// addr[5] ^= (idx + 32)
		
		00:xx:xx:xx:xx:3f(idx 1),
			// Follow the rules in 14.3.2
			00:xx:xx:xx:xx:3f (idx 1),
			00:xx:xx:xx:xx:38 (idx 2),
			00:xx:xx:xx:xx:39 (idx 3),
			00:xx:xx:xx:xx:3a (idx 4)

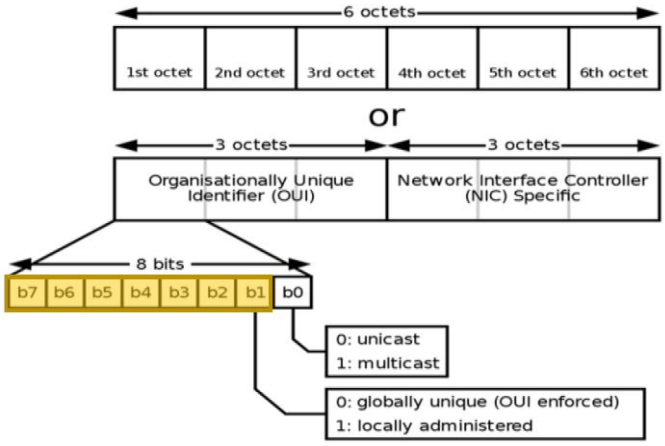
	phy0.2-ap1, phy0.2-ap2 ...	00:xx:xx:xx:xx:3c(idx 2), 00:xx:xx:xx:xx:3d(idx 3), 00:xx:xx:xx:xx:3a(idx 4), 00:xx:xx:xx:xx:3b(idx 5), …	

-	**MLD address #1**	// addr = base_addr
		// addr[5] ^= idx
		// addr[5] ^= 60
		00:xx:xx:xx:xx:22 (idx 60, mld idx 1)	

	MLD address #2, 3..,
	16
		// addr = base_addr
		// addr[5] ^= idx
		10:xx:xx:xx:xx:22 (mld idx 2),
		20:xx:xx:xx:xx:22 (mld idx 3),
		30:xx:xx:xx:xx:22 (mld idx 4), ...
		F0:xx:xx:xx:xx:22 (mld idx 16)	

14.1 Legacy MBSS

The mbss of mt76 is controlled by the hostapd daemon. So, the user can use hostapd/hostapd_cli to create the virtual bss with security mode, ssid, and other AP parameters. We use the local bit to present multiple Aps; the local bit is bit1 in the first byte.



The primary access point (AP) will utilize the EEPROM MAC address. The other access points (APs) will set the local bit to 1 and use the remaining bits 2-7 for presentation.

For example:

- MBSS number is 3, 3-2 (<= 21, bit mask:1), using b2, <b1>
- 00:xx, 02:xx, 06:xx
- MBSS number is 8, 8-2 (<= 23, bit mask:3), using b4, b3, b2, <b1>
- 00:xx, 02:xx, 06:xx, 0a:xx, 0e:xx, 12:xx

14.1.1 Adding a Virtual BSS Can Be Accomplished by the Following Steps

Here is an example to add a virtual bss

```
uci set wireless.phy0_ap1='wifi-iface'
```

```
uci set wireless.phy0_ap1.disabled=0
uci set wireless.phy0_ap1.device=radio0
uci set wireless.phy0_ap1.mode=ap
uci set wireless.phy0_ap1.ssid= 2.4G_mbss1
uci set wireless.phy0_ap1.network=lan
```

Other AP parameters can also be added, as preferred.

```
uci set wireless.phy0_ap1.encryption=psk2+ccmp
uci set wireless.phy0_ap1.ieee80211w=0
uci set wireless.phy0_ap1.key=12345678
```

14.2 Legacy Station

Mostly the same as Legacy MBSS, except the station interface will use an EEPROM MAC address. All the station interfaces will set the local bit to 1.

14.3 11v MBSS

For the 11v MBSS feature, we set up a special BSS called *Transmit BSS and others called Non-Transmit BSS*.
One or more special IEs - MBSSID IEs - will be included in the beacon by Transmit BSS to display other Non-Transmit Basic Service Set Information.

Non-Transmitting BSS will not send out a beacon.

Note that due to standard limitations, WPA security mode is not supported by the 11v MBSS.
Note that due to upstream limitations, Hidden SSID is not supported by the 11v MBSS.

14.3.1 How to Configure 11v MBSS

The first step is to create a normal bss.

And then the special parameters for 11v mbss

```
Case 1: uci set wireless.radio0.mbssid=1
Case 2: uci set wireless.phy0_ap1.macaddr=00:00:43:80:63:20
```

Case 1:
The mbssid=1 means the radio will open the 11v mbss mode.
In this mode, the main BSS is just the *Transmit BSS,* and the other BSSs in the same band are all *Non-Transmit BSSs.*

Case 2:
There is a special rule for the bssid of 11v mbss – chapter 13.2.2

So, users need to check and configure the right mac address for all 11v mbss.

14.3.2 11v MBSS MAC Rule

According to generating 11v MBSS bssid in Section 9.4.2.45, all the MAC addresses of non-transmitted bss are based on transmitted bss.

REF BSSID: The transmit BSSID

If REF BSSID equals MacAddr = 00:11:22:33:44:11, n = **4** *(Max 16 MBSS)*

The BSSID for i is equal to BSSID_A or BSSID_B.

BSSID_A = 00:11:22:33:44:1**0** bit 48 ~ bit n => copy from reference BSSID, bit (n-1) ~ bit 0 => set to 0

BSSID_B = 00:00:00:00:00:0**X** bit 48 ~ bit n => set to 0, bit (n-1) ~ bit 0 => (i + n LSBs of REF_BSSID) % 2^n

Example:

BSSID(1) = 00:11:22:33:44:1**2**

BSSID(2) = 00:11:22:33:44:1**3**

…

BSSID(14) = 00:11:22:33:44:1**F**

BSSID(15) = 00:11:22:33:44:1**0**

Some cases of the main AP MAC address are being considered.

1. Use the UCI option MAC address as the transmitted BSS mac address.
2. If UCI is not set, we will apply the legacy MBSS rule; if we have STA, then set the local bit. Otherwise, AP just uses the EEPROM MAC address.
3. Follow the spec rule to generate a non-transmitted BSS MAC address. Hence, the non-transmitted BSS MAC address is determined in the spec. In this case, the parsing script will ignore the UCI MAC address setting for non-transmitted BSS.

15 Routed Station Mode

The routed (NAT) client mode is the most generic wireless option. It is supported by all chipsets and drivers and requires no special modifications. The downside of routed client mode is the inability to bridge network segments or relay broadcast traffic. A fully routed setup allows access by using the client router's WWAN IP address as the gateway for the client network behind it.

You can refer to link to understand the problem using bridged client mode issue.

Step1: Create station’s wifi interface section

```
vi /etc/config/wireless
…
config wifi-iface 'default_radio0'
        option device 'radio0'
        option network wwan
        option mode 'sta'
        option ssid 'OpenWrt'
        option encryption 'none'
```

Step2: Modify the br-lan ip address.

```
ifconfig br-lan 192.168.2.1
```

Step3: Enable DHCP for wwan.

```
vi /etc/config/network
…
config interface 'wan6'
        option device 'eth1'
        option proto 'dhcpv6'
config interface 'wwan'
        option proto 'dhcp'
```

15.1 Bridged Client Mode (with relayd)

It is possible to achieve a bridge-like client mode setup with the help of relayd.
https://openwrt.org/docs/guide-user/network/wifi/relay_configuration

You need to ask the vendor to apply corresponding patches about supporting hardware paths to gain high throughput.
Limitation: Only one wwan interface can be added into relay-bridge. You need to select *ONLY* one backhaul radio at the same time.

16 Wireless Distribution System (WDS)

MT76 drivers support AP-to-STA WDS mode to use wireless bridging to form one common broadcast domain based on executing a normal association procedure (authentication request/response, association request/response).

In MWDS (Mixed WDS) mode, the management/control frames are transmitted using 3-address, and the data frame types are transmitted using 4-address.

16.1 WDS-AP Setting

Only one change is needed from a normal wireless access point configuration.

```
vi /etc/config/wireless
…
config wifi-iface 'default_radio0'
        option device 'radio0'
        option network 'lan'
        option mode 'ap'
        option ssid 'OpenWrt'
        option encryption 'none'
        option wds '1'
```

16.2 WDS-STA Setting

Specific options differ based on hardware. The SSID, channel, encryption type, and password must match the access point. Enable WDS mode.

Enable WDS Options

```
vi /etc/config/wireless
…
config wifi-iface 'default_radio0'
        option device 'radio0'
        option network 'lan'
        option mode 'sta'
        option wds '1'
        option ssid 'OpenWrt'
        option encryption 'none'
```

The DHCP server on the LAN interface should be disabled.

```
vi /etc/config/dhcp
…
config dhcp 'lan'
        option interface 'lan'
        option start '100'
        option limit '150'
        option leasetime '12h'
        option dhcpv4 'server'
        option dhcpv6 'disabled'
        option ra 'server'
        option ra_slaac '1'
        list ra_flags 'managed-config'
        list ra_flags 'other-config'
        option ignore '1'
```

Set a static IP address

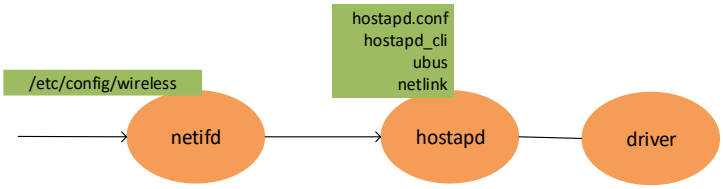
```
vi /etc/config/network
…
config interface 'lan'
        option device 'br-lan'
        option proto 'static'
        option ipaddr '192.168.1.188'
        option netmask '255.255.255.0'
        option ip6assign '60'
```

The device should reboot and automatically connect wirelessly to the access point. **Wait until the client bridge device associates with the access point. This can take 1-2 minutes for the association to happen.** After connecting, the “iwinfo” command should show a new device name (e.g., “wlan.staN” (where N is a number)) in addition to the base “wlan0” wireless interface device at **WDS-AP**.

```
root@OpenWrt:/# iwinfo
wlan0     ESSID: "OpenWrt"
          Access Point: 00:0C:43:28:A2:A7
          Mode: Master  Channel: 157 (5.785 GHz)
          ...
wlan0.sta1 ESSID: unknown
          Access Point: 00:0C:43:28:A2:A7
          Mode: Master (VLAN)  Channel: 157 (5.785 GHz)
          …
```

17 Encryption Modes

The IEEE 8021X/WPA/WPA2/WAP3 authenticator is executed by hostapd in the MT76 platform. Hostapd operates as a user-space daemon connected to the driver, providing a control interface to users. UCI for wireless simplifies wireless interface setup.



In most cases, wireless UCI is sufficient for configuring wireless security options; however, for some new features, setting them through the hostapd control interface directly is necessary.

17.1 Security Options

Most security options are located in the *wifi-iface* or *wifi-mld* section of the wireless UCI, such as encryption, key, *rsn_preauth*, encryption modes, WPA enterprise, etc. See the configuration examples below, and a way to check these security options is by *hostapd_cli*. Note that the security options should be setup on *wifi-mld* section if an AP MLD is configured.

UCI Usage:

```
uci set wireless.<wifi-iface>.encryption=<security mode>+<cipher>
# no need for open/owe mode
uci set wireless.<wifi-iface>.key=<key value>
# set Hash-to-Element (H2E)
uci set wireless.<wifi-iface>.sae_pwe=2

# set RSNO Override
# AP (BSS or MLD link)
uci set wireless.<wifi-iface>.encryption_rsno=<security mode>+<cipher>
uci set wireless.<wifi-iface>.encryption_rsno_2=<security mode>+<cipher>
# STA
uci set wireless.<wifi-iface>.rsn_overriding=<0|1|2>
```

For example:

```
# WPA3 Personal (PSK) mode with cipher CCMP-128
uci set wireless.phy0-ap0.encryption=sae (cipher can be ignored when setting CCMP-128)
# OWE (Opportunistic Wireless Encryption) with cipher CCMP-256
uci set wireless.phy2-ap1.encryption=owe+ccmp256
# WPA2/WPA3 Personal (PSK/SAE mixed mode) with cipher GCMP-128
uci set wireless.ap-mld-1.encryption=sae-mixed+gcmp128
```

Usable security mode values:

Type	Value
open	none
WPA2 Personal (PSK)	psk2
OWE (Opportunistic Wireless Encryption)	owe
WPA3 Personal (PSK) mode	sae
WPA2/WPA3 Personal (PSK/SAE mixed mode)	sae-mixed
AKM24	sae-ext

Usable cipher values:

Note that this cipher is valid only for WPA3

Type	Value
CCMP-128	ccmp or N/A
CCMP-256	ccmp256
GCMP-128	gcmp128
GCMP 256	gcmp256

17.1.1 Security Options for Hostapd/Wpa_supplicant Configuration Files

Here is an example of hostapd and wpa_supplicant security configuration that uses WPA3 Personal (PSK) mode with cipher CCMP-128, and a brief description.

AP Side (Hostapd Configuration)	STA Side (WPAD Configuration)

…
# fixed value (0 for open mode) 	
wpa=2
auth_algs=1
 	
# security mode & password 	
wpa_key_mgmt=SAE
wpa_passphrase=12345678
 	
# cipher 	
wpa_pairwise=CCMP
group_mgmt_cipher=AES-128-CMAC
 	
# mgmt. & beacon protection 	
ieee80211w=2
sae_require_mfp=1
beacon_prot=1
 	
# SAE mechanism for PWE derivation 	
sae_pwe=2 /*0 =looping, 1=H2E, 2=Mixed mode*/
…
	network={
	….
	# fixed value (N/A for open mode)
	proto=RSN
	
	# security mode
	key_mgmt=SAE
	
	# password
	# psk for psk2, sae_password for WPA3
	psk="12345678"
	sae_password="12345678"
	
	# cipher
	pairwise=CCMP
	group_mgmt=AES-128-CMAC
	
	# mgmt. & beacon protection
	ieee80211w=2
	beacon_prot ‘1’
	}

Usable wpa_key_mgmt/key_mgmt value:

Type	Value
open	N/A
WPA2 Personal (PSK)	**# PMF off** wpa_key_mgmt=WPA-PSK **# PMF on** wpa_key_mgmt=WPA-PSK WPA-PSK-SHA256
OWE (Opportunistic Wireless Encryption)	OWE
WPA3 Personal (PSK) mode	SAE
WPA2/WPA3 Personal (PSK/SAE mixed mode)	WPA-PSK WPA-PSK-SHA256 SAE
AKM24	WPA-EXT-KEY

Usable wpa_pairwise/pairwise and group_mgmt_cipher/group_mgmt values

Type	Value	
	wpa_pairwise	group_mgmt_cipher
CCMP-128	CCMP	AES-128-CMAC
CCMP-256	CCMP-256	BIP-CMAC-256
GCMP-128	GCMP-128	BIP-GMAC-128
GCMP 256	GCMP-256	BIP-GMAC-256

17.1.2 OWE Transition Mode in AP Side

AP Side (UCI Configuration)	AP Side (Hostapd Configuration)
# Add hidden OWE BSS uci set wireless.<wifi-iface>.ssid=OpenWrt-owe uci set wireless.<wifi-iface>.encryption=owe uci set wireless.<wifi-iface>.hidden=1 uci set wireless.<wifi-iface>.ifname=ap_owe uci set wireless.<wifi-iface>.owe_transition_ifname=ap_open # Add Open BSS uci set wireless.<open-wifi-iface>=wifi-iface uci set wireless.<open-wifi-iface>.device=<radio_number> uci set wireless.<open-wifi-iface>.network=lan uci set wireless.<open-wifi-iface>.mode=ap uci set wireless.<open-wifi-iface>.ssid=OpenWrt-Open uci set wireless.<open-wifi-iface>.encryption=none uci set wireless.<open-wifi-iface>.ifname=ap_open uci set wireless.<open-wifi-iface>owe_transition_ifname=ap_owe	owe_transition_ifname=ap_open auth_algs=1 wpa=0 ssid= OpenWrt-owe ignore_broadcast_ssid=1 bss=ap_open owe_transition_ifname=ap_owe auth_algs=1 wpa=0 ssid= OpenWrt-Open ignore_broadcast_ssid=0

17.2 Protected Management Frames

PMF (Protected Management Frames), or 802.11w, is used to enhance the security by providing data confidentiality of management frames, mechanisms to enable data integrity, data origin authenticity, and replay protection. It is controlled by wireless UCI option “ieee80211w” as follows.

UCI Usage:

```
uci set wireless.<wifi-iface>.ieee80211w=2
```

Name	Type	Required	Default	Description
ieee80211w	Integer	No	*0*	Enables MFP (802.11w) to support (0 = disabled, 1 = optional, 2 = required). Requires the 'full' version of wpad/hostapd and support from the Wi-Fi driver

18 Wi-Fi Protected Setup (WPS)

The Wi-Fi Protected Setup (WPS) is also known as WLAN Simple Configuration. It is a standard feature that simplifies the process of setting up protected connections for various WLAN devices connecting to APs. Although WPS is not yet popular, it is a very useful configuration method that helps users avoid manually configuring SSID and security to create a WLAN.

18.1 WPS Related Settings

The WPS related parameters are in /etc/config/wireless (refer to OpenWrt/WPS options).

Name	Type	Required	Default	Description
wps_config	List	No	*None*	The list of configuration methods Currently supported methods: *push_button*
wps_device_name	String	No	*LEDE AP*	The user-friendly description of the device; up to 32 octets encoded in UTF-8
wps_device_type	String	No	*6-0050F204-1*	The primary device type Examples: *1-0050F204-1* (Computer/PC), *1-0050F204-2* (Computer/Server), *5-0050F204-1* (Storage / NAS), *6-0050F204-1* (Network Infrastructure/AP)
wps_label	Boolean	No	*0*	Enables *the label* configuration method.
wps_manufacturer	String	No	*lede-project.org*	The manufacturer of the device (up to 64 ASCII characters)
wps_pin	String	No	none	The PIN to use with WPS-PIN (only in external registrar mode)
wps_pushbutton	Boolean	No	*0*	Enables *the push-button* configuration method

Please note that enabling hostapd-*utils is done through make menuconfig,* and CONFIG_WPS should also be enabled in hostapd's.conf file.

18.2 WPS Test Examples

~~**Currently, WPS can only be conducted on AP MLD's first link.**~~

The code versions following MLD MP4.1 (202504) are capable of supporting AP MLD on either the 2.4GHz or 5GHz bands. Please note that WPS setup cannot be performed on the 6GHz band due to specification restrictions.

18.2.1 Push-Button Mode

Configuration:

1. Set up related WPS parameters in /etc/config/wireless.

```
# /etc/config/wireless
...
# On legacy BSS
config wifi-iface 'default_radio0'
        option device 'radio0'
        option mode 'ap'
        option ssid 'My-WiFi-Home'
        option network 'lan'
        option encryption 'psk2'
        option key 'WiFipassword'
```

```
        option ieee80211w '0'
        option wps_pushbutton '1'


# On AP MLD
config wifi-mld 'ap_mld_1'
        option ssid 'My-WiFi-Home'
        option encryption 'sae'
        option key 'WiFipassword'
        option ieee80211w '2'
        option wps_pushbutton '1'
        option mld_id '1'
```

2. Restart the network service by commanding: /etc/init.d/network restart

Runtime Command:

AP Side (hostapd_cli)	STA Side (Wpad Configuration)
# legacy BSS hostapd_cli -i <interface> wps_pbc # MLD link hostapd_cli -i <interface> **-l 0** wps_pbc	wpa_cli -i <interfcace> wps_pbc

18.2.2 Pin Mode

Configuration:

1. Set up related WPS parameters in /etc/config/wireless.

```
# /etc/config/wireless
...
# On legacy BSS
config wifi-iface 'default_radio0'
        option device 'radio0'
        option mode 'ap'
        option ssid 'My-WiFi-Home'
        option network 'lan'
        option encryption 'psk2'
        option key 'WiFipassword'
        option ieee80211w '0'
        option wps_label '1'

# On AP MLD
config wifi-mld 'ap_mld_1'
        option ssid 'My-WiFi-Home'
        option encryption 'sae'
        option key 'WiFipassword'
        option ieee80211w '0'
        option mld_id '1'
        option wps_label '1'
```

2. Restart the network service by commanding: /etc/init.d/network restart

Runtime Command:

AP Side (hostapd_cli)	STA Side (WPAD Configuration)

# legacy BSS 	
hostapd_cli -i <interface> wps_pin any $PIN
 	
# MLD link 	
hostapd_cli -i <interface> **-l 0** wps_pin any $PIN
	wpa_cli -i <interface> wps_pin <BSSID> [PIN]
	Start the WPS PIN method (returns the PIN if not hardcoded).
	
	Example
	wpa_cli -I <interface> wps_pin get
	//generate WPS PIN
	62467803
	wpa_cli -i <interface> wps_pin any 62467803
	//configure PIN and enter this PIN to registrar
	
	BSSID-based scan and connection
	wpa_cli -I <interface> scan
	
	wpa_cli -i <interface> scan_results
	//find the bssid, to connect to
	
	wpa_cli -i <interface> wps_pin 02:0c:43:59:ad:3f 62467803 //configure the PIN and enter this PIN into the registrar

19 IGMP Snooping & Multicast-to-Unicast Conversion

In this section, a brief introduction to IGMP (Internet Group Management Protocol) snooping and multicast-to-unicast conversion, abbreviated as M2U, are presented, followed by how to configure these settings.

19.1 Introduction

IGMP querier periodically sends IGMP membership queries to determine which multicast addresses are of interest to the systems it serves.

IGMP Snooping is the process of listening to IGMP conversation between IGMP querier and hosts, and maintaining a map of which ports need which IP multicast transmission. Based on the map, multicast traffic may be filtered from the ports that do not need it, thus conserving bandwidth on these ports.

Multicast-to-unicast conversion is the process of converting multicast traffic to unicast one. It has many advantages, including better performance because unicast traffic typically can be transmitted at higher data rates compared to multicast one.

19.2 Configuration

In this subsection, how to enable, disable, and show status of multicast querier, IGMP snooping, and multicast-to-unicast conversion are elaborated.

19.2.1 Multicast Querier

Multicast querier is a feature that allows a bridge to act as an IGMP querier. If there is no IGMP querier (e.g. multicast router) in the network, one can enable this feature to ensure that IGMP group membership information is continuously updated and properly maintained.

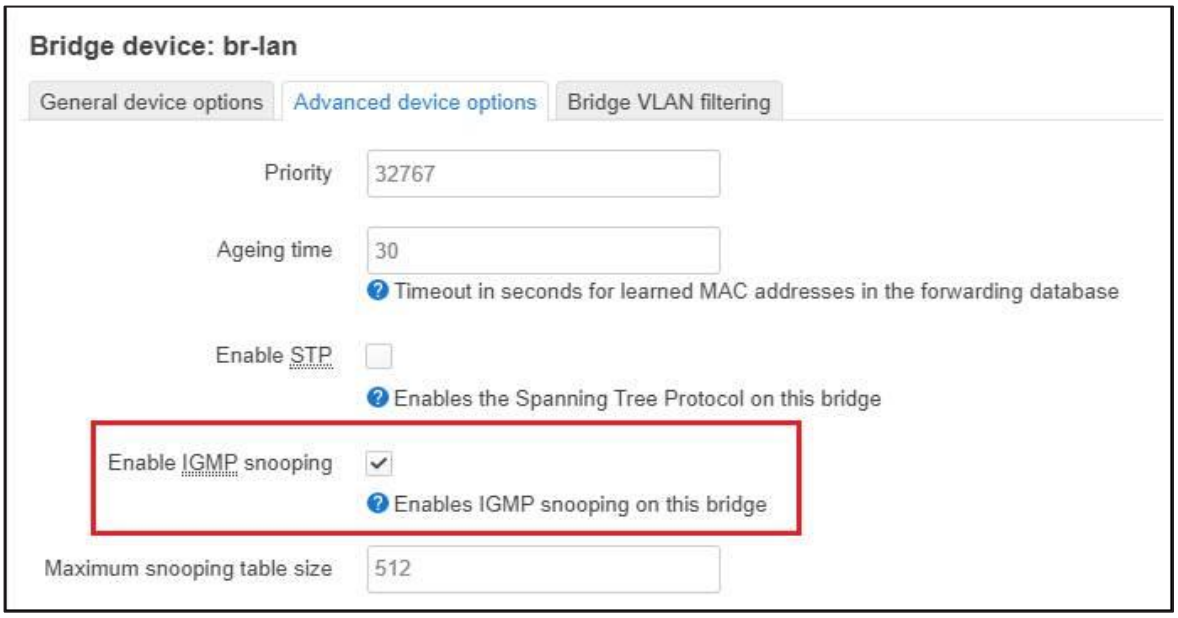
Configure multicast querier on bridge via ***sysfs***.

```
# Enable
echo 1 > /sys/class/net/br-lan/bridge/multicast_querier

# Disable
echo 0 > /sys/class/net/br-lan/bridge/multicast_querier

# Show current setting
cat /sys/class/net/br-lan/bridge/multicast_querier
```

19.2.2 IGMP Snooping

IGMP snooping relies on IGMP group membership information to determine which ports need which multicast traffic. Without such information, the switch cannot perform snooping properly. Hence, for IGMP snooping to work effectively, it is highly recommended to enable multicast querier on the bridge to ensure the presence of IGMP messages.

Enable/disable IGMP snooping via ***LuCI***.



Configure IGMP snooping via network configuration file.

```
# Enable
uci set network.@device[0].igmp_snooping=1
uci commit network
reload_config

# Disable
uci set network.@device[0].igmp_snooping=0
uci commit network
reload_config

# Show current setting
cat /etc/config/network
...
config device
        option name 'br-lan'
        option type 'bridge'
        list ports 'lan0'
        list ports 'lan1'
        list ports 'lan2'
        list ports 'lan3'
        list ports 'lan4'
        list ports 'lan5'
        option igmp_snooping '0'

config interface 'lan'
        option device 'br-lan'
        option proto 'static'
        option ipaddr '192.168.1.1'
        option netmask '255.255.255.0'
...
```

Configure IGMP snooping via ***sysfs***.

```
# Enable
echo 1 > /sys/class/net/br-lan/bridge/multicast_snooping

# Disable
echo 0 > /sys/class/net/br-lan/bridge/multicast_snooping

# Show current setting
cat /sys/class/net/br-lan/bridge/multicast_snooping
```

19.2.3 Multicast-to-Unicast Conversion

Multicast-to-unicast conversion works on top of IGMP snooping of the bridge. Specifically, the conversion takes place only if the destination host signalizes that it is interested in the multicast traffic via IGMP membership report. In other words, for M2U to work effectively, it is necessary to enable IGMP snooping on the bridge. For more details, refer to bridge: multicast to unicast commit.

Configure multicast-to-unicast conversion on interface via ***sysfs***.

```
# Enable
echo 1 > /sys/class/net/br-lan/brif/<interface>/multicast_to_unicast

# Disable
echo 0 > /sys/class/net/br-lan/brif/<interface>/multicast_to_unicast

# Show current setting
cat /sys/class/net/br-lan/brif/<interface>/multicast_to_unicast
```

19.3 Patches

The kernel should be checked to determine if the following patch has been applied or is included.

- 021-bridge-multicast-to-unicast.patch

Check the proxy package (IGMPv3 and MLDv2 Multicast Proxy)

- https://openwrt.org/packages/pkgdata_owrt18_6/omcproxy
- https://openwrt.org/docs/guide-user/network/wan/udp_multicast

20 High-Efficiency Wi-Fi 6E Access Point (AP) and Station (STA)

20.1 6G Security

The Wi-Fi Alliance will require **WPA3 security** for Wi-Fi 6E devices that will operate in the 6 GHz band, and the hostapd upstream solution follows the Wi-Fi Alliance. Therefore, please enable the 6G interface with encryption.

20.2 AP Setting

Step 1: Set the 6G interface.

```
vi /etc/config/wireless
…
config wifi-device 'radio2'
        option type 'mac80211'
        option path 'platform/18000000.wbsys+1'
        option channel '37'
        option band '6g'
        option htmode 'HE80'
        option country 'DE'
config wifi-iface 'default_radio2'
        option device 'radio2'
        option network 'lan'
        option mode 'ap'
        option ssid '000OpenWrtGGGGGG'
        option encryption 'sae'
        option key '12345678'
```

Step 2: Enable the 6G interface.

```
wifi
```

Step 3: Check the result.

```
iw dev
phy#2
        Interface wlan2
                ifindex 17
                wdev 0x200000002
                addr 82:0c:43:49:a9:49
                ssid 000OpenWrtGGGGGG
                type AP
                channel 37 (6135 MHz), width: 80 MHz, center1: 6145 MHz
                txpower 23.00 dBm
                multicast TXQ:
                        qsz-byt qsz-pkt flows   drops   marks   overlmt hashcol tx-bytes
tx-packets
                        0       0       99      0       0       0       0       13476
99
```

20.3 STA Setting

Step 1:

```
vi /etc/config/wireless
…
config wifi-device 'radio2'
        option type 'mac80211'
        option path 'platform/18000000.wbsys+1'
        option htmode 'HE80'
        option country 'DE'
        option channel '37'
        option band '6g'
config wifi-iface 'default_radio2'
        option device 'radio2'
        option network 'lan'
        option mode 'ap'
        option ssid 'OpenWrt'
        option encryption 'sae'
        option key '12345678'
config wifi-iface 'wifinet3'
        option device 'radio2'
        option mode 'sta'
        option network 'wwan'
        option ssid '000OpenWrtGGGGGG'
        option encryption 'sae'
        option key '12345678'
```

Note:

- The SSID in 'wifinet3' is the desired SSID that the STA intends to connect to.

Step 2: Modify the br-lan ip address.

```
ifconfig br-lan 192.168.2.1
```

Step 3: Enable DHCP for wwan.

```
vi /etc/config/network
…
config interface 'wan6'
        option device 'eth1'
        option proto 'dhcpv6'
config interface 'wwan'
        option proto 'dhcp'
```

Step 4: Enable the 6G interface.

```
wifi
```

Step 5: Check the result.

```
ifconfig
br-lan    Link encap:Ethernet  HWaddr 00:0C:43:49:A9:40
          inet addr:192.168.2.1  Bcast:192.168.2.255  Mask:255.255.255.0
          inet6 addr: fd0d:d0ed:f29a::1/60 Scope:Global
          inet6 addr: fe80::20c:43ff:fe49:a940/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1134 errors:0 dropped:0 overruns:0 frame:0
          TX packets:744 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:102457 (100.0 KiB)  TX bytes:74876 (73.1 KiB)
…
wlan2     Link encap:Ethernet  HWaddr 82:0C:43:49:A9:3F
          inet addr:192.168.1.244  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::800c:43ff:fe49:a93f/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:500 errors:0 dropped:0 overruns:0 frame:0
          TX packets:509 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:40251 (39.3 KiB)  TX bytes:40497 (39.5 KiB)
wlan2-1   Link encap:Ethernet  HWaddr 86:0C:43:49:A9:3F
          inet6 addr: fe80::840c:43ff:fe49:a93f/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:81 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:15784 (15.4 KiB)
```

```
iw dev
…
phy#2
        Interface wlan2-1
                ifindex 18
                wdev 0x200000003
                addr 86:0c:43:49:a9:3f
                ssid OpenWrt
                type AP
                channel 37 (6135 MHz), width: 80 MHz, center1: 6145 MHz
                txpower 23.00 dBm
                multicast TXQ:
                        qsz-byt qsz-pkt flows   drops   marks   overlmt hashcol
tx-bytes      tx-packets
                        0       0       0       0       0       0       0       0
0
        Interface wlan2
                ifindex 17
                wdev 0x200000002
                addr 82:0c:43:49:a9:3f
                ssid 000OpenWrtGGGGGG
                type managed
                channel 37 (6135 MHz), width: 80 MHz, center1: 6145 MHz
                txpower 23.00 dBm
                multicast TXQ:
                        qsz-byt qsz-pkt flows   drops   marks   overlmt hashcol
tx-bytes      tx-packets
```

```
                                0       0       0       0       0       0       0       0
0
```

Note:

- wlan2: Station mode; wlan2-1: AP mode

20.4 In-band Discovery Protocol

MT76 drivers support the in-band discovery protocol in the 6G band. To speed up the discovery process, there are two passive scan methods for in-band discovery.

Passive scan in the 6G band:

1. Unsolicited probe response
2. Fast Initial Link Setup (FILS) discovery

Note:

- Either an unsolicited probe response or a FILS discovery

20.4.1 Unsolicited Probe Response

The hostapd configuration needs to be enabled.

```
vi /var/run/hostapd-phy2.conf
…
unsol_bcast_probe_resp_interval=20
fils_discovery_min_interval=0
```

Sniffer check

```
0.019612    82:0c:43:49:a9:49  Broadcast  802.11    292 Beacon frame, SN=3751, FN=0, Flags=........, BI=100, SSID=OpenWrt123
0.040707    82:0c:43:49:a9:49  Broadcast  802.11    286 Probe Response, SN=3752, FN=0, Flags=........, BI=100, SSID=OpenWrt123
0.061121    82:0c:43:49:a9:49  Broadcast  802.11    286 Probe Response, SN=3753, FN=0, Flags=........, BI=100, SSID=OpenWrt123
0.082077    82:0c:43:49:a9:49  Broadcast  802.11    286 Probe Response, SN=3754, FN=0, Flags=........, BI=100, SSID=OpenWrt123
0.102107    82:0c:43:49:a9:49  Broadcast  802.11    286 Probe Response, SN=3755, FN=0, Flags=........, BI=100, SSID=OpenWrt123
0.121903    82:0c:43:49:a9:49  Broadcast  802.11    292 Beacon frame, SN=3756, FN=0, Flags=........, BI=100, SSID=OpenWrt123
```

20.4.2 Fast Initial Link Setup (FILS)

Enabling it in the hostapd configuration is required.

```
vi /var/run/hostapd-phy2.conf
…
fils_discovery_max_interval=20
```

Sniffer check

```
0.028736    82:0c:43:49:a9:49  Broadcast  802.11    292 Beacon frame, SN=478, FN=0, Flags=........, BI=100, SSID=OpenWrt123
0.049380    82:0c:43:49:a9:49  Broadcast  802.11    118 FILS Discovery, BI=100
0.069798    82:0c:43:49:a9:49  Broadcast  802.11    118 FILS Discovery, BI=100
0.090220    82:0c:43:49:a9:49  Broadcast  802.11    118 FILS Discovery, BI=100
0.110644    82:0c:43:49:a9:49  Broadcast  802.11    118 FILS Discovery, BI=100
0.131020    82:0c:43:49:a9:49  Broadcast  802.11    292 Beacon frame, SN=483, FN=0, Flags=........, BI=100, SSID=OpenWrt123
```

20.5 Out-of-band Discovery

Out-of-band discovery is completely controlled by hostapd. To make it work, you have to enable its prerequisite, the Reduced Neighbor Report (RNR), in the hostapd config file.

```
vi /var/run/hostapd-phyX.conf
…
rnr=1
```

20.6 EDCCA Control

MT76 drivers support configuring Energy Detect Clear Channel Assessment (EDCCA). The default EDCCA feature is enabled. If you want to disable EDCCA, add the following configuration to the hostapd config file.

```
vi /var/run/hostapd-phyX.conf
…
edcca_enable=0
```

You can also adjust the default EDCCA threshold value.

```
vi /var/run/hostapd-phyX.conf
…
edcca_threshold=aa bb cc dd
```

The first one (denoted as aa) is used to configure the BW20 threshold, the second (denoted as bb) is for BW40, the third (denoted as cc) is for BW40, and the last one (denoted as dd) is for BW160. The range of the threshold is from -126 to 0 dBm. Please note that each threshold is separated by a space. If you did not add the EDCCA threshold in hostapd configuration, the firmware will decide on the default EDCCA threshold value.

The bandwidth of 160 MHz only needs to be set in Wi-Fi 7 BSP; it is not supported by Wi-Fi 6 series.

MT76 drivers support adding the compensation value to the BW20 EDCCA threshold for the 6G band only. In other words, you can adjust the BW20 EDCCA threshold by configuring this compensation value. Change the compensation value by adding the following configuration in the hostapd config file:

```
vi /var/run/hostapd-phyX.conf
…
edcca_compensation=-6
```

This parameter applies only to Wi-Fi 6. The default compensation value is -6 dBm. To adjust the EDCCA threshold, use -6 dBm as the base value.

If you're running regulatory certification (FCC/ETSI) on Wi-Fi6 chips, you need to disable firmware auto edcca mechanism and the threshold will be exactly the same as your configuration.

```
#echo 0 > /sys/kernel/debug/ieee80211/phyX/mt76/sw_aci
```

The configuration mentioned above can be set before bringing the interface up. For dynamic adjustments during runtime, please consult 32.4 EDCCA Feature Control for detailed instructions.

20.7 Automated Frequency Coordination (AFC)

MT76 driver support AFC feature since Wi-Fi 7; to configure hostapd to run AFC procedure, the device type should be set as standard power. The device type varies, as below:

```
HE_REG_INFO_6GHZ_AP_TYPE_INDOOR          = 0,
HE_REG_INFO_6GHZ_AP_TYPE_SP                = 1,
HE_REG_INFO_6GHZ_AP_TYPE_VLP               = 2,
HE_REG_INFO_6GHZ_AP_TYPE_INDOOR_ENABLED    = 3,
HE_REG_INFO_6GHZ_AP_TYPE_INDOOR_SP         = 4,
```

```
# hostapd config to set standard power device type
he_6ghz_reg_pwr_type=1
or
he_6ghz_reg_pwr_type=4
```

20.7.1 Configuration Settings in Hostapd for an AFC Request

According to the AFC spec, the AFC request should include some device information, and this information should be filled in the hostapd config. Below is an example of one of these configs.

```
# AFC daemon connection socket
afcd_sock=/var/run/afcd.sock

# AFC request identification parameters
afc_request_version=1.4
afc_request_id=0
afc_serial_number=SN000
afc_cert_ids=US_47_CFR_PART_15_SUBPART_E:CID000
#
# AFC location type:
# 0 = ellipse
# 1 = linear polygon
# 2 = radial polygon
afc_location_type=0
#
# AFC ellipse or linear polygon coordinations
afc_linear_polygon=-121.98586998164663:37.38193354300452
#
# AFC radial polygon coordinations
afc_radial_polygon=118.8:92.76,76.44:87.456,98.56:123.33
#
# AFC ellipse major/minor axis and orientation
afc_major_axis=150
afc_minor_axis=150
afc_orientation=0
#
# AFC device elevation parameters
afc_height=15
afc_height_type=AGL
afc_vertical_tolerance=2
#
```

```
# AFC minimum desired TX power (dbm)
afc_min_power=24
#
# AFC request frequency ranges
afc_freq_range=5925:6425,6525:6875
#
# AFC request operation classes
afc_op_class=131,132,133,134,136,137
#
# AFC maximum timeout time between two queries (second)
afc_max_timeout=180

# sku index for default or standard power used, value should > 0
sku_idx=1
# sku index for lpi mode used, value should > 0
lpi_sku_idx=2
```

20.7.2 AFC Daemon (afcd)

afcd is a daemon used to help hostapd to build http request and communicate with AFC server. Therefore, we need to run afcd and specify the query address before hostapd start to query AFC response. Below is an example to send request to harness. Before we send the http request we should put the certification into the device. Note that if the AP can't query the IP address from DNS server, we can directly set it in to local dns table.

```
# Put .pem in /etc/ssl/cert.pem
# Put .crt in /etc/ssl/certs/ca-certificates.crt

$ echo "[AFC server IP address] testserver.wfatestorg.org" >> /etc/hosts
$ afcd -u https://testserver.wfatestorg.org/afc-simulator-
api/availableSpectrumInquiry -d &
```

20.7.3 Example

An example of an AFC request:

```
root@OpenWrt:/# Received request: { "version": "1.4", "availableSpectrumInquiryRequests": [ { "requestId": "0", "d
eviceDescriptor": { "serialNumber": "SN000", "certificationId": [ { "rulesetId": "US_47_CFR_PART_15_SUBPART_E", "i
d": "CID000" } ] }, "location": { "ellipse": { "center": { "longitude": -121.98586998164663, "latitude": 37.381933
543004521 }, "majorAxis": 150, "minorAxis": 150, "orientation": 0 }, "elevation": { "height": 15.0, "heightType":
"AGL", "verticalUncertainty": 2 }, "indoorDeployment": 1 }, "minDesiredPower": 24, "inquiredFrequencyRange": [ { "
lowFrequency": 5925, "highFrequency": 6425 }, { "lowFrequency": 6525, "highFrequency": 6875 } ], "inquiredChannels
": [ { "globalOperatingClass": 131, "channelCfi": [ 1, 5, 9, 13, 17, 21, 25, 29, 33, 37, 41, 45, 49, 53, 57, 61, 6
5, 69, 73, 77, 81, 85, 89, 93, 97, 101, 105, 109, 113, 117, 121, 125, 129, 133, 137, 141, 145, 149, 153, 157, 161,
 165, 169, 173, 177, 181, 185, 189, 193, 197, 201, 205, 209, 213, 217, 221, 225, 229, 233 ] }, { "globalOperatingC
lass": 132, "channelCfi": [ 3, 11, 19, 27, 35, 43, 51, 59, 67, 75, 83, 91, 99, 107, 115, 123, 131, 139, 147, 155,
163, 171, 179, 187, 195, 203, 211, 219, 227, 235 ] }, { "globalOperatingClass": 133, "channelCfi": [ 7, 23, 39, 55
, 71, 87, 103, 119, 135, 151, 167, 183, 199, 215, 231 ] }, { "globalOperatingClass": 134, "channelCfi": [ 15, 47,
79, 111, 143, 175, 207, 239 ] }, { "globalOperatingClass": 136, "channelCfi": [ 2 ] }, { "globalOperatingClass": 1
37, "channelCfi": [ 61, 93, 125, 157, 189, 221 ] } ] } ] }
```

An example of an AFC server response:

```
Received reply: {"availableSpectrumInquiryResponses": [{"response": {"responseCode": 0, "shortDescription": "SUCCE
SS"}, "availableFrequencyInfo": [{"frequencyRange": {"highFrequency": 5965, "lowFrequency": 5945}, "maxPsd": 19.2}
, {"frequencyRange": {"highFrequency": 5985, "lowFrequency": 5965}, "maxPsd": 13.9}, {"frequencyRange": {"highFreq
uency": 6005, "lowFrequency": 5985}, "maxPsd": 19.4}, {"frequencyRange": {"highFrequency": 6025, "lowFrequency": 6
005}, "maxPsd": 16.4}, {"frequencyRange": {"highFrequency": 6045, "lowFrequency": 6025}, "maxPsd": 14.1}, {"freque
ncyRange": {"highFrequency": 6065, "lowFrequency": 6045}, "maxPsd": 20.6}, {"frequencyRange": {"highFrequency": 60
85, "lowFrequency": 6065}, "maxPsd": 11.4}, {"frequencyRange": {"highFrequency": 6105, "lowFrequency": 6085}, "max
Psd": 15.8}, {"frequencyRange": {"highFrequency": 6285, "lowFrequency": 6265}, "maxPsd": 20.6}, {"frequencyRange":
 {"highFrequency": 6305, "lowFrequency": 6285}, "maxPsd": 19.9}, {"frequencyRange": {"highFrequency": 6325, "lowFr
equency": 6305}, "maxPsd": 11.4}, {"frequencyRange": {"highFrequency": 6345, "lowFrequency": 6325}, "maxPsd": 18.8
}, {"frequencyRange": {"highFrequency": 6365, "lowFrequency": 6345}, "maxPsd": 17.6}, {"frequencyRange": {"highFre
quency": 6385, "lowFrequency": 6365}, "maxPsd": 16.7}, {"frequencyRange": {"highFrequency": 6405, "lowFrequency":
6385}, "maxPsd": 21.4}, {"frequencyRange": {"highFrequency": 6425, "lowFrequency": 6405}, "maxPsd": 19.3}], "avail
ableChannelInfo": [{"channelCfi": [1, 5, 9, 13, 17, 21, 25, 29, 65, 69, 73, 77, 81, 85, 89, 93], "globalOperatingC
lass": 131, "maxEirp": [32.2, 26.9, 32.4, 29.4, 27.1, 33.6, 24.4, 28.8, 33.6, 32.9, 24.4, 31.8, 30.6, 29.7, 34.4,
32.3]}, {"channelCfi": [3, 11, 19, 27, 67, 75, 83, 91], "globalOperatingClass": 132, "maxEirp": [29.9, 32.4, 30.1,
 27.4, 35.9, 27.4, 32.7, 35.3]}, {"channelCfi": [7, 23, 71, 87], "globalOperatingClass": 133, "maxEirp": [32.9, 30
.4, 30.4, 35.7]}, {"channelCfi": [15, 79], "globalOperatingClass": 134, "maxEirp": [33.4, 33.4]}], "requestId": "0
", "availabilityExpireTime": "2024-08-08T09:16:24Z", "rulesetId": "US_47_CFR_PART_15_SUBPART_E"}], "version": "1.4
"}
```


Dump the current afc table. The power value of 127 means the channel or the bandwidth is invalid, according to the AFC response.

```
$ cat /sys/kernel/debug/ieee80211/phy0/mt76/afc_table
bw/ru :    20    40    80   160  320-1 320-2    26    52    78   106   132   726
1480  1772  2476  2988  3472
ch   1:   127   127   127   127   127   127   127   127   127   127   127   127
127   127   127   127   127
ch   5:   127   127   127   127   127   127   127   127   127   127   127   127
127   127   127   127   127
ch   9:   127   127   127   127   127   127   127   127   127   127   127   127
127   127   127   127   127
ch  13:   127   127   127   127   127   127   127   127   127   127   127   127
127   127   127   127   127
ch  17:   127   127   127   127   127   127   127   127   127   127   127   127
127   127   127   127   127
ch  21:   127   127   127   127   127   127   127   127   127   127   127   127
127   127   127   127   127
ch  25:   127   127   127   127   127   127   127   127   127   127   127   127
127   127   127   127   127
ch  29:   127   127   127   127   127   127   127   127   127   127   127   127
127   127   127   127   127
ch  33:    30    36    42    48   127    54    10    16    20    22    24    40
46    47    50    52    53
ch  37:    52    36    42    48   127    54    32    38    42    44    46    40
46    47    50    52    53
ch  41:    36    42    42    48   127    54    16    22    26    28    30    40
46    47    50    52    53
ch  45:    48    42    42    48   127    54    28    34    38    40    42    40
46    47    50    52    53
ch  49:    56    60    50    48   127    54    36    42    46    48    50    48
46    47    50    52    53
ch  53:    54    60    50    48   127    54    34    40    44    46    48    48
46    47    50    52    53
ch  57:    38    44    50    48   127    54    18    24    28    30    32    48
46    47    50    52    53
ch  61:    50    44    50    48   127    54    30    36    40    42    44    48
46    47    50    52    53
```

```
ch  65:    52    38    44    50   127    54    32    38    42    44    46    42
48    49    50    52    53
ch  69:    32    38    44    50   127    54    12    18    22    24    26    42
48    49    50    52    53
ch  73:    50    56    44    50   127    54    30    36    40    42    44    42
48    49    50    52    53
ch  77:    52    56    44    50   127    54    32    38    42    44    46    42
48    49    50    52    53
ch  81:    38    44    50    50   127    54    18    24    28    30    32    48
48    49    50    52    53
ch  85:    42    44    50    50   127    54    22    28    32    34    36    48
48    49    50    52    53
ch  89:    54    56    50    50   127    54    34    40    44    46    48    48
48    49    50    52    53
ch  93:    50    56    50    50   127    54    30    36    40    42    44    48
48    49    50    52    53
......
ch 233:   127   127   127   127   127   127   127   127   127   127   127   127
127   127   127   127   127
```

21 Extender Mode Behavior Description

21.1 Configuration Setup

When an extender is configured with both AP and STA interfaces on the same radio, the AP interface is set up first, followed by the STA interface. During the STA interface setup, the AP interface is temporarily disabled, as indicated by the following log:

Please note that the behavior of AP and STA concurrent is mainly maintained by OpenWRT Hostapd Patches:
Wi-Fi 6: Hostapd/wpa_supplicant through reload_config (by socket). Fi 6: Hostapd/wpa_supplicant through reload_config (by socket)

OpenWRT Hostapd Patches	MTK Internal Patches
340-reload_freq_change.patch 360-ctrl_iface_reload.patch 370-ap_sta_support.patch 450-scan_wait.patch 700-wifi-reload.patch	mtk-xxx-hostapd-mtk-Add-channel-information-for-hostapd-relo.patch mtk-xxx-hostapd-mtk-Avoid-setting-beacon-after-wpa_supplican.patch mtk-xxx-hostapd-mtk-Fix-unexpected-AP-beacon-state-transitio.patch mtk-xxx-hostapd-mtk-Fix-CCA-issue.patch mtk-xxx-hostapd-mtk-Add-sta-assisted-DFS-state-update-mechan.patch mtk-xxx-hostapd-mtk-Fix-auto-ht-issue-when-switching-to-DFS-.patch mtk-xxx-hostapd-mtk-add-log-in-extender-mode.patch

Wi-Fi 6: line 629

```
628 Thu Oct 26 11:47:28 2023 daemon.notice netifd: Network device 'phy0-sta0' link is up
629 Thu Oct 26 11:47:28 2023 daemon.notice hostapd: mtk: stop iface for phy0-ap0
630 Thu Oct 26 11:47:28 2023 daemon.notice wpa_supplicant[2858]: phy0-sta0: Associated with 00:0c:43:2f:9d:12
631 Thu Oct 26 11:47:28 2023 kern.debug kernel: [  243.768596] phy0-sta0: moving STA 00:0c:43:2f:9d:12 to state 4
632 Thu Oct 26 11:47:28 2023 daemon.notice wpa_supplicant[2858]: phy0-sta0: CTRL-EVENT-CONNECTED - Connection to 00:0c:43:2f:9d:12 completed [id=0 id_str=]
633 Thu Oct 26 11:47:28 2023 kern.info kernel: [  243.774717] IPv6: ADDRCONF(NETDEV_CHANGE): phy0-sta0: link becomes ready
634 Thu Oct 26 11:47:28 2023 kern.info kernel: [  243.781581] br-lan: port 8(phy0-sta0) entered blocking state
635 Thu Oct 26 11:47:28 2023 kern.info kernel: [  243.787258] br-lan: port 8(phy0-sta0) entered forwarding state
636 Thu Oct 26 11:47:28 2023 daemon.notice hostapd: mtk: update iface for phy0-ap0
637 Thu Oct 26 11:47:28 2023 daemon.notice hostapd: mtk: new channel information: channel=36, secondary_channel=1, center_segment0=42, center_segment1=0, op_class=128
638 Thu Oct 26 11:47:28 2023 daemon.notice wpa_supplicant[2858]: phy0-sta0: CTRL-EVENT-SUBNET-STATUS-UPDATE status=0
```

Wi-Fi 7: Hostapd/wpa_supplicant through ucode Fi 7: Hostapd/wpa_supplicant through ucode

OpenWRT Hostapd Patches	MTK Internal Patches
hostapd: add ucode support, use ucode for the main ubus object hostapd: reimplement AP/STA support via ucode	xx-hostapd-ucode-mtk-add-parsing-eht_oper-from-hostapd.patch xx-hostapd-ucode-mtk-synchronize-bandwidt.patch mtk-xx-hostapd-mtk-ucode-add-support-for-ucode-to-parse-BW3.patch mtk-xx-hostapd-mtk-synchronize-bandwidth-in-AP-STA-support.patch

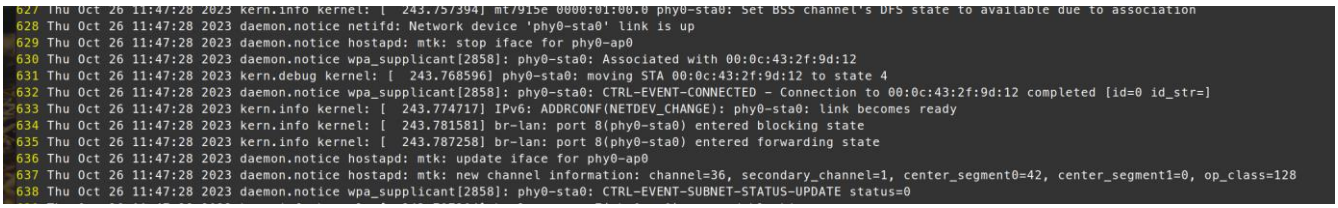
Wi-Fi 7: line 1096

```
1094 Thu Sep 14 11:01:36 2023 daemon.notice hostapd:     * csa: null
1095 Thu Sep 14 11:01:36 2023 daemon.notice hostapd: Interface phy1-ap0 already disabled
1096 Thu Sep 14 11:01:36 2023 daemon.notice hostapd: ucode: mtk: stop iface for phy1-ap0
1097 Thu Sep 14 11:01:36 2023 daemon.debug wpa_supplicant[1734]: Not configuring frame filtering - BSS 00:00:00
1098 Thu Sep 14 11:01:36 2023 daemon.debug wpa_supplicant[1734]: nl80211: Authenticate (ifindex=7)
```

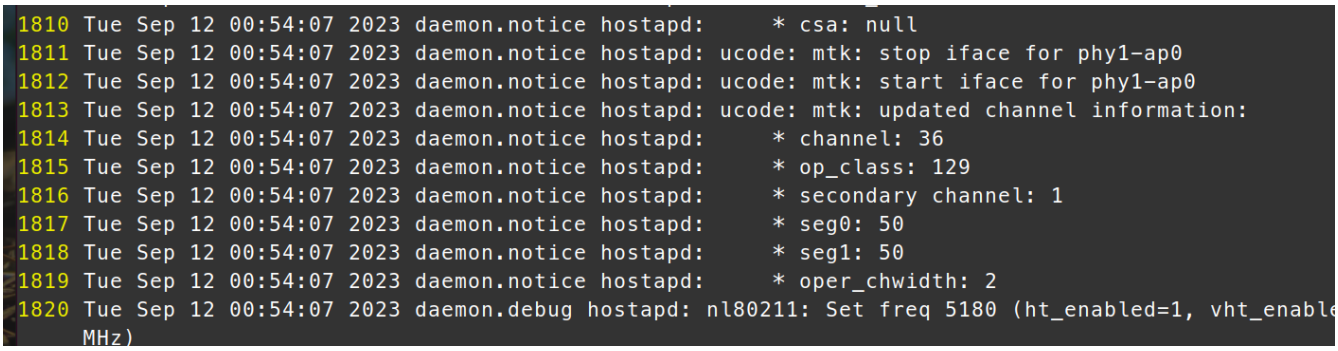
21.2 AP and STA Channel Synchronization Behavior

After connecting to the Root AP, the STA interface will restart the AP interface and reload it with the same channel information as the Root AP. This includes the control channel, center channel, secondary channel offset, and operating class. However, the PHY mode is not included, and the AP interface will operate in the PHY mode specified by the configuration. The following log indicates that the STA has successfully restarted the AP and loaded the channel information from the Root AP:

Wi-Fi 6: lines 636 and 637



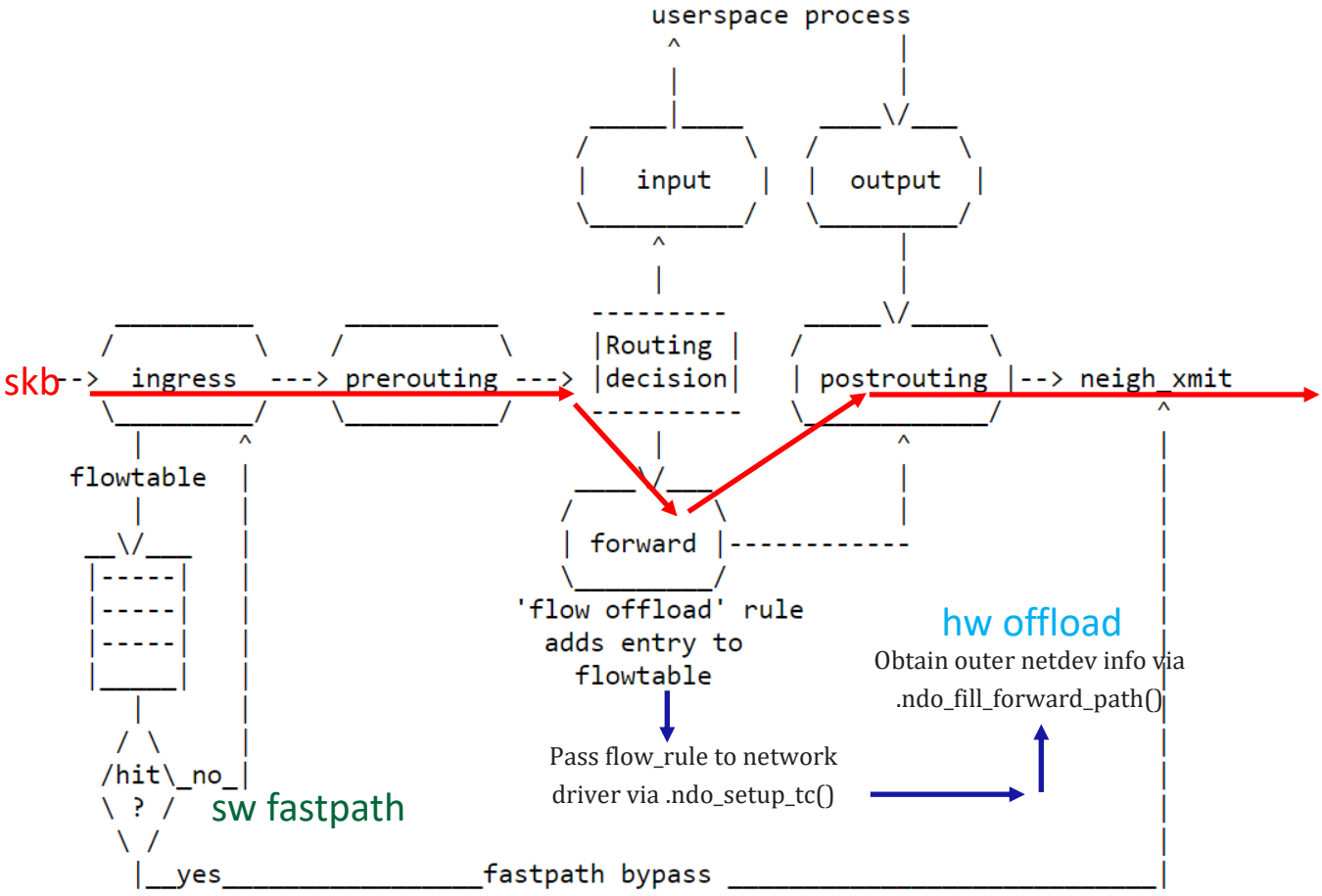
Wi-Fi 7: lines 1812 to 1819



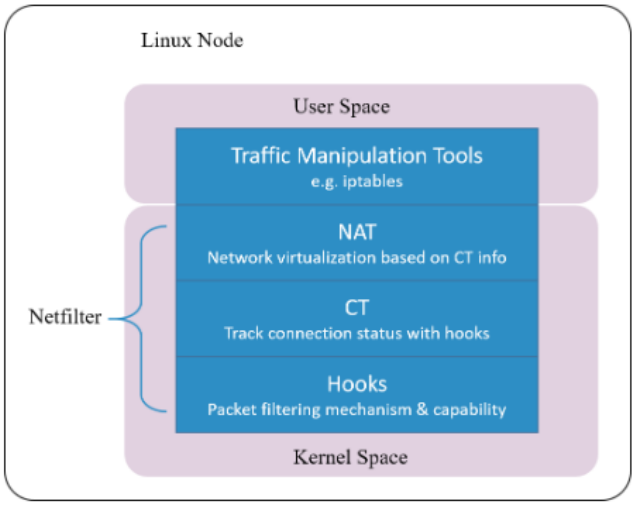
22 Netfilter Hardware Offload Infrastructure (FlowOffload)

22.1 Software FastPath vs. Hardware FastPath

MediaTek use standard Netfilter framework for ethernet and Wi-Fi hardware offload. For Netfilter flowtable infrastructure, refer to: https://docs.kernel.org/networking/nf_flowtable.html.



22.2 Iptables - User Space Tool to Config Netfilter



22.2.1 Hardware Offload Rule Using iptables Commands

	IPv4	IPv6
TCP	iptables -I FORWARD -p tcp -m conntrack --ctstate RELATED,ESTABLISHED -j FLOWOFFLOAD --hw	ip6tables -I FORWARD -p tcp -m conntrack --ctstate RELATED,ESTABLISHED -j FLOWOFFLOAD --hw
UDP	iptables -I FORWARD -p udp -j FLOWOFFLOAD --hw	ip6tables -I FORWARD -p udp -j FLOWOFFLOAD --hw

22.2.2 Dump iptables Rules

Function	Command
List iptables rules	iptables -L FORWARD -n --line-number
Clear iptables rules	iptables -D FORWARD xx

22.3 Connection Tracking Debug

Check the Hardware Offload entries
cat /proc/net/nf_conntrack | grep OFFLOAD

22.4 Check Netsys Packet Processing Engine (PPE)

Dump All Entries	Dump Binding
cat /sys/kernel/debug/mtk_ppe/entries	cat /sys/kernel/debug/mtk_ppe/bind

22.5 Netsys Per-flow Accounting

When traffic is running in H/W path, traffic no longer goes through CPU and ETH/Wi-Fi drivers, and thus we can't do statistic such as byte counts transmitted, byte counts received, etc.

A new mechanism has been proposed to solve this problem when hardware offloading is enabled.

22.5.1 Statistics in the conntrack

```
root@OpenWrt:/# cat /proc/net/nf_conntrack | grep OFFLOAD
ipv4     2 tcp      6 src=192.168.1.5 dst=192.168.1.15 sport=63283 dport=60114 packets=1905369
bytes=2885182803 src=192.168.1.15 dst=192.168.1.5 sport=60114 dport=63283 packets=107218
bytes=6432610 [HW_OFFLOAD] mark=0 zone=0 use=3
```

22.5.2 Statistics in the PPE Bind Entries

```
root@OpenWrt:/# cat /sys/kernel/debug/mtk_ppe/bind
01230 BND IPv4 5T ppe=0 orig=192.168.1.5:55940->192.168.1.15:61891 new=192.168.1.5:55940-
>192.168.1.15:61891 eth=00:0c:43:49:a2:d8
->54:05:db:e2:7b:68 etype=0008 vlan=0,0 bytes=251904471 packets=166357
01231 BND IPv4 5T ppe=0 orig=192.168.1.15:61891->192.168.1.5:55940 new=192.168.1.15:61891-
>192.168.1.5:55940 eth=00:0c:43:49:a2:d8
->d0:37:45:2c:d6:00 etype=0101 vlan=0,0 bytes=605160 packets=10086
```

22.6 Wi-Fi-Ethernet-Dispatch Configuration

For Wi-Fi 6 series chipsets (MT7915/MT7916/MT7986)

Module Parameter Initialization (/etc/modules.d/mt7915e)	
WED On (Default)	WED Off
mt7915e wed_enable=1 or mt7915e	mt7915e wed_enable=0
RunTime Configuration	
WED On (Default)	WED Off
rmmod mt7915e rmmod mt76-connac-lib rmmod mt76 rmmod mac80211 rmmod cfg80211 modprobe cfg80211 modprobe mac80211 modprobe mt76 modprobe mt76-connac-lib modprobe mt7915e wed_enable=1 sleep 5 killall hostapd killall netifd	rmmod mt7915e rmmod mt76-connac-lib rmmod mt76 rmmod mac80211 rmmod cfg80211 modprobe cfg80211 modprobe mac80211 modprobe mt76 modprobe mt76-connac-lib modprobe mt7915e wed_enable=0 sleep 5 killall hostapd killall netifd
Check the Module Parameter of MT76	
cat /sys/module/mt7915e/parameters/wed_enable	

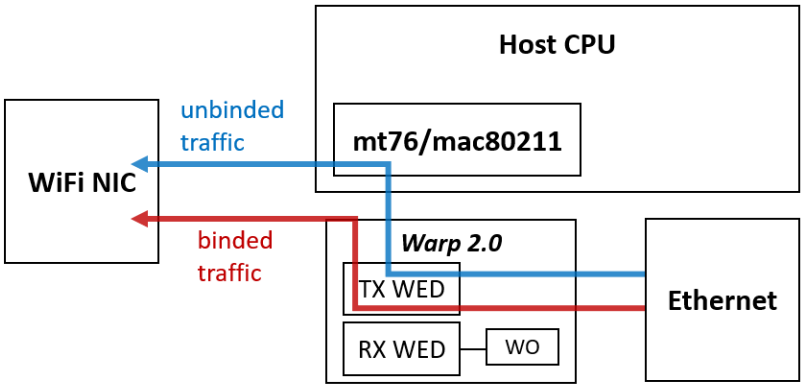
For Wi-Fi 7 series chipsets (MT7996/MT7992)

Module Parameter Initialization (/etc/modules.d/mt7996e)	
WED On (Default)	WED Off
mt7996e wed_enable=1 or mt7996e	mt7996e wed_enable=0
RunTime Configuration	
WED On (Default)	WED Off
rmmod mt7996e rmmod mt76-connac-lib rmmod mt76 rmmod mac80211 rmmod cfg80211 modprobe cfg80211 modprobe mac80211 modprobe mt76 modprobe mt76-connac-lib modprobe mt7996e wed_enable=1 sleep 5 killall hostapd killall netifd	rmmod mt7996e rmmod mt76-connac-lib rmmod mt76 rmmod mac80211 rmmod cfg80211 modprobe cfg80211 modprobe mac80211 modprobe mt76 modprobe mt76-connac-lib modprobe mt7996e wed_enable=0 sleep 5 killall hostapd killall netifd
Check the Module Parameter of MT76	
cat /sys/module/mt7996e/parameters/wed_enable	

22.7 Wi-Fi Statistic for Hardware Offload Path

When traffic is running in software path, packets go through CPU and driver (MT76/MAC80211), and thus we can do statistic such as byte counts transmitted, byte counts received, etc.

When WED is enabled and traffic is bound to the hardware path, the traffic is no longer visible.
The driver disrupts the operation of software statistics. A new mechanism is required to resolve this issue.
Hardware offloading is disabled.



22.7.1 Statistics for H/W Tx Path

Tx Status (TXS) one kind of packet that generated from Wi-Fi H/W to give a status about the transmission. There are 2 types of TXS: MPDU TXS and PPDU TXS. With PPDU TXS, we can get the Tx byte counts, fail counts and failed counts. To get PPDU TXS, we need to enable register PPDU_TXS2H_EN_B0/1. With that being set, we'll get PPDU TXS sending up from H/W.

PPDU TXS

- Set **CR PPDU_TXS2H_EN_B0/1**
- Gets TX bytes, retry_cnt, etc…
- TXSFM=2

22.7.2 Statistics for H/W Rx Path

The receive path, on the other hand, uses the RXCNT_INFO event from the WO mcu to get the Rx statistics. When H/W offload is enabled, MT76 will send a WO mcu command RXCNT_CTRL, which requests WO to send the RXCNT_INFO event periodically. mtk_wed will get the event and use the registered hook function to update the Rx statistics in the MT76 driver. The command to see Wi-Fi statistics.

Connect a wireless client to DUT, then generate traffic and check binded or not
cat /sys/kernel/debug/mtk_ppe/bind
once binded, use iw command to get statistic

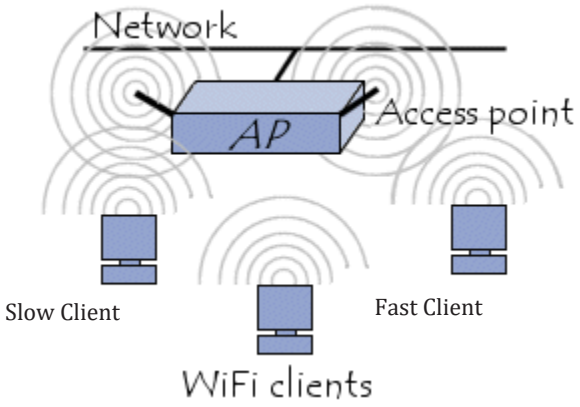
iw [ifname] station dump

e.g.

```
root@OpenWrt:/# iw wlan1 station dump
Station 00:0c:43:2a:90:3d (on wlan1)
        inactive time:  2088 ms
        rx bytes:       2455373
        rx packets:     45456
        tx bytes:       169899332
        tx packets:     113244
        tx retries:     13008
        tx failed:      13191
…
```

23 Hardware Airtime Fairness (HW ATF)

When slow-speed clients are connected to a high-capacity network, too much time is taken to transmit, leading to a decrease in overall throughput.



To increase overall throughput, slow client should not get large airtime. A solution to this problem is “Airtime Fairness”.

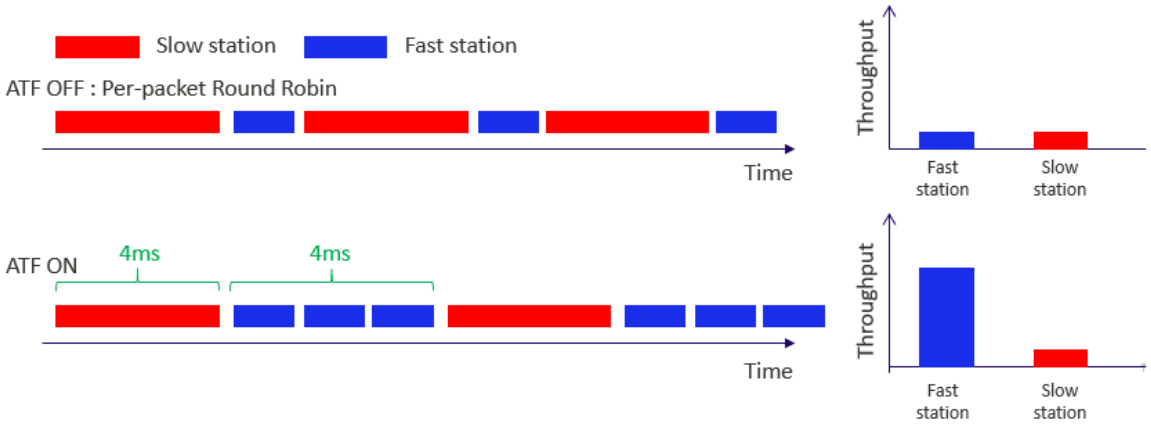
23.1 Airtime Fairness (ATF)

23.1.1 Introduction of ATF

Station airtime fairness operates within the same WMM AC in a group. There are two purposes:

- To make all stations have the same airtime usage in the same WMM AC
- To avoid that, a slow client uses too much airtime and decreases the total throughput

In other words, we change station’s Tx packet schedule to achieve airtime fairness among stations.



23.1.2 Usage

Supported DebugFS Commands

MT7986

Different bands have a shared DebugFS knob:

Band0/1/2: /sys/kernel/debug/ieee80211/phy0/mt76/vow

root@OpenWrt:/sys/kernel/debug/ieee80211/phy0/mt76# cat vow

======== Control ==========

vow_atf_en=<0/1> 0:disable, 1:enable

vow_watf_en=<0/1> 0:disable, 1:enable

vow_watf_quantum=<level>-<quantum> unit 256us

======== Station table ==========

vow_watf_set_entry=<qid>-<mac address>

vow_dwrr_max_wait_time=<time> 256us

======== Debug ==========

show_vow_info

vow_show_sta=<STA num>

vow_show_en=<0/1> 0:dieable, 1:enable

show_vow_sta_conf=<STA num> 0:all

MT7996

vow_info: Show the VoW configuration.

```
root@OpenWrt:/sys/kernel/debug/ieee80211/phy0/mt76# cat vow_info
VoW ATF Configuration:
ATF: enabled
WATF: disabled
Airtime Quantums (unit: 256 us)
        L0: 6
        L1: 12
        L2: 16
        L3: 20
        L4: 24
        L5: 28
        L6: 32
        L7: 36
Max Airtime Deficit: 64 (unit: 256 us)
```

Enable/Disable ATF

MT7986

Turn ON ATF by command (default on):

cd /sys/kernel/debug/ieee80211/phy0/mt76/

echo vow_atf_en=1 > vow

echo vow_dwrr_max_wait_time=64 > vow

Turn OFF ATF using the command:

echo vow_atf_en=0 > vow

echo vow_dwrr_max_wait_time=1 > vow

Note: Please make sure vow_dwrr_max_wait_time=1 when ATF disable.

MT7996

atf_enable: enable/disable ATF.

Enable ATF (default on):

```
root@OpenWrt:/sys/kernel/debug/ieee80211/phy0/mt76# echo 1 > atf_enable
```

Disable ATF:

```
root@OpenWrt:/sys/kernel/debug/ieee80211/phy0/mt76# echo 0 > atf_enable
```

Show the airtime information of the stations

MT7986

echo vow_show_sta=3 > vow

echo vow_show_en=1 > vow

```
[ 2701.924990] mt7996e 0000:01:00.0: sta1:00:10:94:20:00:04 tx → 496657)
[ 2701.931531] mt7996e 0000:01:00.0: sta2:00:10:94:00:10:01 tx → 495648)
[ 2702.948990] mt7996e 0000:01:00.0: sta1:00:10:94:20:00:04 tx → 496322)
[ 2702.955527] mt7996e 0000:01:00.0: sta2:00:10:94:00:10:01 tx → 496694)
[ 2703.972989] mt7996e 0000:01:00.0: sta1:00:10:94:20:00:04 tx → 500589)
[ 2703.979527] mt7996e 0000:01:00.0: sta2:00:10:94:00:10:01 tx → 492794)
[ 2704.996987] mt7996e 0000:01:00.0: sta1:00:10:94:20:00:04 tx → 495489)
[ 2705.003518] mt7996e 0000:01:00.0: sta2:00:10:94:00:10:01 tx → 497027)
[ 2706.020988] mt7996e 0000:01:00.0: sta1:00:10:94:20:00:04 tx → 500306)
[ 2706.027514] mt7996e 0000:01:00.0: sta2:00:10:94:00:10:01 tx → 487364)
[ 2707.044989] mt7996e 0000:01:00.0: sta1:00:10:94:20:00:04 tx → 498864)
[ 2707.051528] mt7996e 0000:01:00.0: sta2:00:10:94:00:10:01 tx → 498943)
[ 2708.068999] mt7996e 0000:01:00.0: sta1:00:10:94:20:00:04 tx → 496003)
[ 2708.075539] mt7996e 0000:01:00.0: sta2:00:10:94:00:10:01 tx → 491512)
[ 2709.092988] mt7996e 0000:01:00.0: sta1:00:10:94:20:00:04 tx → 496043)
[ 2709.099516] mt7996e 0000:01:00.0: sta2:00:10:94:00:10:01 tx → 501600)
[ 2710.116989] mt7996e 0000:01:00.0: sta1:00:10:94:20:00:04 tx → 500274)
[ 2710.123529] mt7996e 0000:01:00.0: sta2:00:10:94:00:10:01 tx → 493188)
[ 2711.140989] mt7996e 0000:01:00.0: sta1:00:10:94:20:00:04 tx → 496276)
[ 2711.147517] mt7996e 0000:01:00.0: sta2:00:10:94:00:10:01 tx → 493188)
[ 2711.140989] mt7996e 0000:01:00.0: sta1:00:10:94:20:00:04 tx → 496276)
[ 2711.147517] mt7996e 0000:01:00.0: sta2:00:10:94:00:10:01 tx → 496526)
```

MT7996

airtime: Show cumulative airtime information (read clear) for all stations.

```
root@OpenWrt:/sys/kernel/debug/ieee80211/phy0/mt76# cat airtime
VoW Airtime Information:
76:5e:de:86:26:ce WCID: 1 BandIdx: 1 OmacIdx: 0x0       TxAirtime: 414881
9a:3c:05:8b:60:ff WCID: 2 BandIdx: 0 OmacIdx: 0x0       TxAirtime: 163342
3e:3e:21:18:48:63 WCID: 3 BandIdx: 1 OmacIdx: 0x11      TxAirtime: 411440
fa:d8:d8:71:ba:2a WCID: 4 BandIdx: 0 OmacIdx: 0x12      TxAirtime: 152318
```

23.2 Weighted ATF (WATF)

23.2.1 Introduction of WATF

For enhanced airtime management, mt76 provides support for weighted airtime fairness.

- We will divide the stations into 4 groups of 4. In different groups, we give different airtime quantums to each other.
  - Level 0 – airtime quantum is 1.5ms.
  - Level 1 – airtime quantum is 3ms.
  - Level 2 – airtime quantum is 4ms.
  - Level 3 – airtime quantum is 5ms.
- In WATF mode, we can service the VIP station by setting it up in a high-level group.
- The lowest level group can be used to “guest station”.

- If you don’t set MAC address in any level, we will use Level 0’s airtime quantum by default.

23.2.2 Usage

Enable/Disable WATF

MT7986

Turn ON WATF by command (default off):

cd /sys/kernel/debug/ieee80211/phy0/mt76/

echo vow_watf_en=1 > vow

Turn off ATF by using the command:

echo vow_watf_en =0 > vow

Note: WATF is one of ATF’s configurations. So before enable WATF, you must enable ATF first.

MT7996

WATF is not supported by FW at this time.

Configure the airtime quantum level group for the station.

MT7986

vow_watf_set_entry=<qid>-<mac address>

echo vow_watf_add_entry=3-00:10:94:20:00:04 > vow

Show the station's quantum level

echo show_vow_sta_conf=<wlanidx>

echo show_vow_sta_conf=1 > vow

```
root@OpenWrt:/# echo show_vow_sta_conf=1 > ./sys/kernel/debug/ieee80211/phy1/mt7
6/vow
[ 6562.798655] mt7996e 0000:01:00.0: mt7996_show_vow_sta_conf: ****** sta1: 00:10:94:20:00:04******
[ 6562.807500] mt7996e 0000:01:00.0: Ac0 ⟶ 5120us(3)
[ 6562.812373] mt7996e 0000:01:00.0: Ac1 ⟶ 5120us(3)
[ 6562.817241] mt7996e 0000:01:00.0: Ac2 ⟶ 5120us(3)
[ 6562.822108] mt7996e 0000:01:00.0: Ac3 ⟶ 5120us(3)
```

24 Normal Mode Pre-calibration

24.1 Pre-calibration introduction

In MT76, pre-calibration (pre-cal) could be classified as two types in terms of its functionality, including Group pre-calibration and DPD/Flatness pre-calibration. For more information, please refer to the following table.

<table>
<tr><th>Pre-cal Type</th><th>Purpose</th><th>Items</th><th colspan="7">Size [Bytes]</th></tr>
<tr><td rowspan="4">Group cal</td><td rowspan="4">Applied during power on
1. Save boot-up time
2. Meet RF regulations by avoiding TX emissions</td><td rowspan="4">RX DCOC
RSSI DCOC
TX TSSI DCOC
TX LPFG
TX FDIQ
TX DCIQ
RX FDIQ
RX FIIQ
ADCDCOC</td><td>Band</td><td>MT7915</td><td>MT7916</td><td>MT7986</td><td colspan="2">MT7996</td></tr>
<tr><td>2 + 5</td><td>49K + 16</td><td>54K + 16</td><td rowspan="2">94K + 16</td><td colspan="2" rowspan="2">--</td></tr>
<tr><td>2 + 6</td><td rowspan="2">--</td><td>94K + 16</td></tr>
<tr><td>2 + 5 + 6</td><td>--</td><td>--</td><td colspan="2">174K + 52</td></tr>
<tr><td rowspan="5">DPD/Flatness cal</td><td rowspan="5">Applied during a channel switch
1. Save channel switch time
2. Meet RF regulations by avoiding TX emissions</td><td rowspan="5">TSSI DNL
TX DNL
TX DPD
TX Flatness</td><td rowspan="2">6G</td><td rowspan="2">--</td><td colspan="2" rowspan="2">198K
(3K/channel)</td><td>Legacy</td><td>354K
(6K/channel)</td></tr>
<tr><td>memDPD</td><td>195K
(15K/channel)</td></tr>
<tr><td rowspan="2">5G</td><td rowspan="2">48K
(2K/channel)</td><td rowspan="2">93K
(3K/channel)</td><td>Legacy</td><td>168K
(6K/channel)</td></tr>
<tr><td>memDPD</td><td>45K
(15K/channel)</td></tr>
<tr><td>2G</td><td>6K
(2K/channel)</td><td colspan="2">9K
(3K/channel)</td><td colspan="2">18K
(6K/channel)</td></tr>
</table>

For generating pre-cal data, refer to the MT76 Test Mode Programming Guide.

24.2 Setting

To enable pre-cal, first of all, please make sure you are currently in flash mode or binfile mode (refer to Chapter 7 for entering flash or binfile mode). Second, please make sure your pre-cal data is written in flash memory and the pre-cal indication bit is set in EEPROM.

24.2.1 Pre-cal indication bit

The pre-cal indication bit is used to indicate whether pre-cal is performed or not. As shown in the following figure, pre-cal uses the first n bits to store the information. For example, eeprom[0x62] = 0x0d means group cal & DPD 2G / 6G is performed. Therefore, the driver would apply group cal & DPD 2G/6G cal to the firmware.

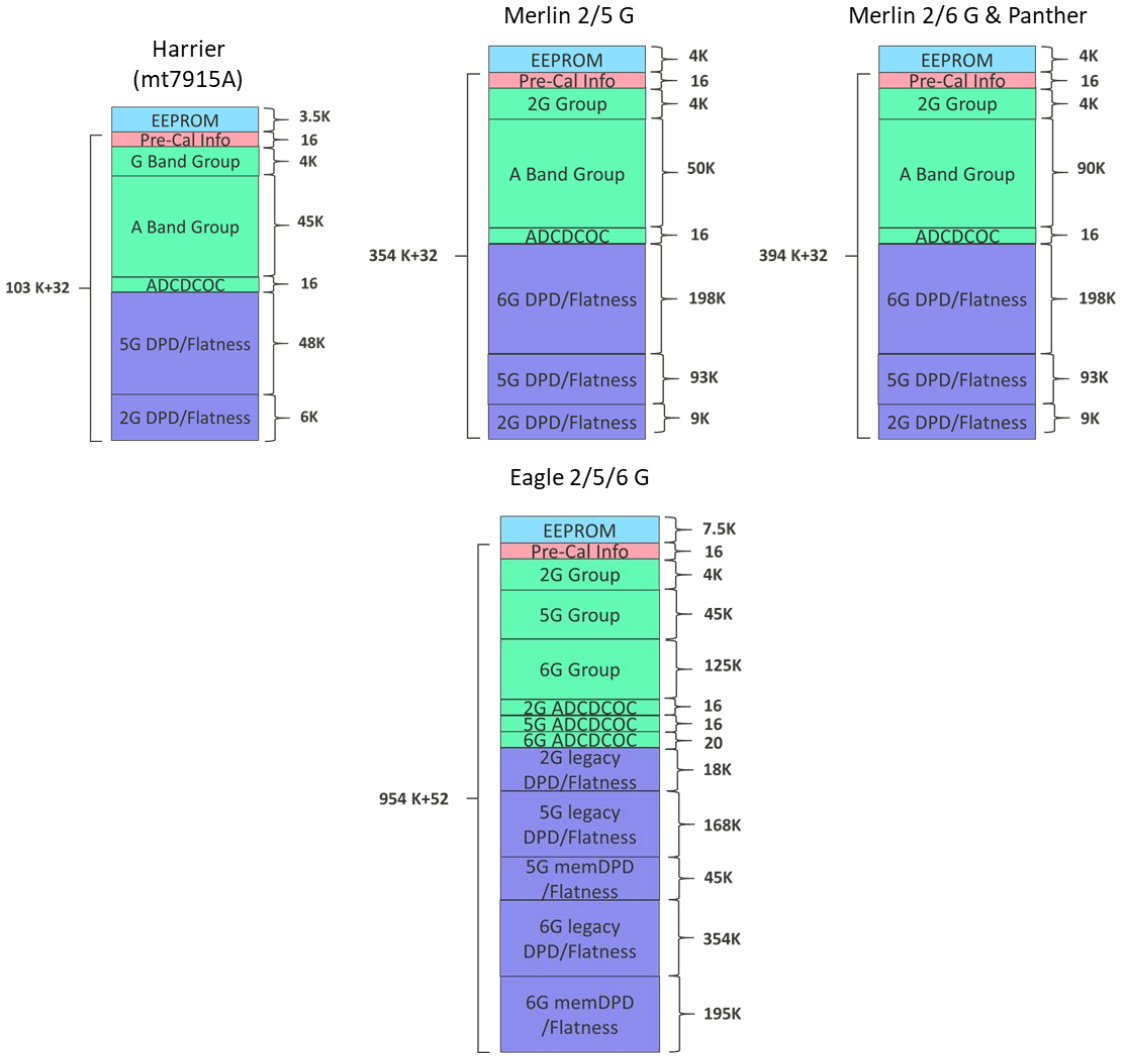
If you use the MT76 test mode command to generate or erase pre-cal data, the pre-cal indication bit will be set properly. To check the indication bit, please use the following command (Take MT7986, for example).

atenl -i phy0 -c "eeprom read 0x19a" or iwpriv phy0 e2p 19a

Chip	Pre-cal Indication Bit EEPROM Offset
MT7915	0x62
MT7916 MT7986	0x19a
NT7996	0x1a5

24.2.2 Pre-cal Data Layout

The data layouts for pre-calibration in flash memory are depicted in the following figures.



Ensure that the data written in flash conforms to the specified layout. Utilize the test mode command to generate pre-calibration data and store it correctly in flash memory.

Note:

- Pre-Cal Info is only used by the driver to store pre-calibration information in test mode. Please leave a blank for pre-cal info if you want to write your pre-calibration data directly to flash instead of using test mode commands.
- For AX8400 (or AX7800), please make sure the flash memory offset is set enough for enabling pre-calibration in both 7986 & 7915(or 7916). For example, 7986s' EEPROM and pre-cal data start from 0x0 in flash memory and 7915s' EEPROM and pre-cal data start from 0x70000 in flash memory (since 7986s' EEPROM & pre-cal data would occupy 0x0 to 0x63820 in flash memory). You can enter the following command to write second chip's EEPROM data to flash memory.

Write eeprom.bin to the memory address 0x70000 using the mtd command for factory settings.

24.3 Usage

Pre-calibration is disabled in upstream due to the fact that using random EEPROM or pre-cal data would lead to Severe throughput drops significantly. After applying all the internal patches, pre-calibration should be enabled and applied automatically.

Note:

- The pre-cal data generated by test mode commands will not lead to severe throughput drops.
- If pre-calibration is applied to firmware and you want to redo the pre-calibration data in test mode, please first erase the pre-calibration data and reboot.

25 Thermal Protection

MT76 provides a thermal protection mechanism. When the chip temperature exceeds the trigger temperature, the chip is protected by lowering the Tx duty cycle to allow the chip to cool down, which causes a reduction in throughput.

When the chip temperature falls below the restore temperature, the Tx duty cycle is increased to restore the throughput.

Adjusting the Tx duty cycle is done by switching the levels on and off. There are currently 4 levels of Tx duty cycle available. Each level is half the Tx duty cycle of the previous level.

The current initial settings for mt76's Tx duty cycle range from level 0 to level 4 with values set at 100, 50, 25, and 12, respectively. Following an update triggered by temperature changes, there will be a delay of 10 seconds before the system reassesses the temperature and readjusts the level.

In addition to the thermal protection mechanism, we also support the user's ability to adjust the Tx duty cycle directly. The user can change the Tx duty cycle of level 0, and the rest of the levels will automatically be half of the Tx duty cycle of the previous level.

The following explains how to change the above-mentioned configuration:

Read the chip's current temperature:

```
# cat /sys/class/hwmon/hwmonX/temp1_input
```

Read/Write Restore temperature:

```
# echo <temp> > /sys/class/hwmon/hwmonX/temp1_crit
# cat /sys/class/hwmon/hwmonX/temp1_crit
```

Read/Write Trigger Temperature:

```
# echo <temp> > /sys/class/hwmon/hwmonX/temp1_max
# cat /sys/class/hwmon/hwmonX/temp1_max
```

Read/Write cooling cycle set by the user:

```
# echo <state> > /sys/class/thermal/cooling_deviceX/cur_state
# cat /sys/class/thermal/cooling_deviceX/cur_state
```

Note that <state> is the cooling state, not the Tx duty cycle value. The sum of cooling state and Tx duty cycle will be 100. If you set 10 to this state, it means that Tx duty cycle is 90%.

Read the current Chip TX duty cycle:

```
# cat /sys/class/hwmon/hwmonX/throttle1
```

26 LED Handling

Currently, Linux has its own LED subsystem for LED device. In MT76, LEDs are configured as LED class devices, we could control them via /sys/class/leds/. The LEDs don't support brightness adjustment, so they will only be turned on for non-zero brightness settings.

A LED classdev is registered for each band by MT76 through the allocation and completion of a LED classdev structure, followed by calling led_classdev_register.

LED registration API

```
static inline int led_classdev_register(struct device *parent,
                                        struct led_classdev *led_cdev)
```

The LED class supports various types of LED triggers. The default trigger is a throughput-based LED blink trigger. The system alters the blink frequency based on the current throughput.

```
struct ieee80211_tpt_blink {
        int throughput;
        int blink_time;
};
```

The following example illustrates the procedure to modify or verify the current configuration.

Check whether the LED class has been instantiated:

```
root@OpenWrt:/# ll /sys/class/leds/
drwxr-xr-x    2 root     root             0 Jan  1  1970 ./
drwxr-xr-x   37 root     root             0 Jan  1  1970 ../
lrwxrwxrwx    1 root     root             0 Apr  9 23:20 mt76-phy0 -> ../../devices/platform/11300000.pcie/pci0000:00/0000:00:00.0/0000:01:00.0/leds/mt76-phy0/
lrwxrwxrwx    1 root     root             0 Apr  9 23:20 mt76-phy1 -> ../../devices/platform/11300000.pcie/pci0000:00/0000:00:00.0/0000:01:00.0/leds/mt76-phy1/
lrwxrwxrwx    1 root     root             0 Apr  9 23:20 mt76-phy2 -> ../../devices/platform/11300000.pcie/pci0000:00/0000:00:00.0/0000:01:00.0/leds/mt76-phy2/
```

Get the current trigger configuration:

```
root@OpenWrt:/# cat /sys/class/leds/mt76-phy0/trigger
none mmc0 timer heartbeat default-on netdev phy0rx phy0tx phy0assoc phy0radio [phy0tpt] phy1rx phy1tx phy1assoc phy1radio phy1tpt phy2rx phy2tx phy2assoc phy2radio phy2tpt
root@OpenWrt:/# cat /sys/class/leds/mt76-phy1/trigger
none mmc0 timer heartbeat default-on netdev phy0rx phy0tx phy0assoc phy0radio phy0tpt phy1rx phy1tx phy1assoc phy1radio [phy1tpt] phy2rx phy2tx phy2assoc phy2radio phy2tpt
root@OpenWrt:/# cat /sys/class/leds/mt76-phy2/trigger
none mmc0 timer heartbeat default-on netdev phy0rx phy0tx phy0assoc phy0radio phy0tpt phy1rx phy1tx phy1assoc phy1radio phy1tpt phy2rx phy2tx phy2assoc phy2radio [phy2tpt]
root@OpenWrt:/#
```

Change the LED trigger to a throughput-based LED blink trigger:

```
# echo phy0tpt > /sys/class/leds/mt76-phy0/trigger
# echo phy1tpt > /sys/class/leds/mt76-phy1/trigger
# echo phy2tpt > /sys/class/leds/mt76-phy2/trigger
```

Set the LED brightness to a non-zero value when the LED trigger is none

```
# echo 1 > /sys/class/leds/mt76-phy0/brightness
```

27 802.11k/v/r

27.1 802.11k (Radio Resource Management)

Measurement of neighboring APs using Beacon requests is supported by Mt76.

The Beacon request/report pair enables a STA to request from another STA a list of APs it can receive on a particular specified channel or channels This measurement may be done in an active mode (like active scan) or in a passive mode Modes such as passive scan mode or beacon table modes are available.

The beacon report is used to collect pre-handoff information, which can drastically speed up handoffs between cells within the same ESS.

The following hostapd_cli cmd command mentioned the configuration parameters:

Beacon request hostapd_cli cmd format:

1. Connect an 11k supported 2G STA to APUT
2. hostapd_cli –i<2G interface> req_beacon + STA's Mac + param to send beacon req to STA

/*Params used to send Request:

* Operating Class (1), Channel Number (1), Randomization Interval (2),
* Measurement Duration (2), Measurement Mode (1), BSSID (6),
* Optional Subelements (variable)

*/

Beacon Request Measurement mode:

Active mode:

2G_11K_MeasMode_active_ch0

a) SSID input in beacon request and Wildcard BSSID:

hostapd_cli -iphy0-ap0 req_beacon c6:db:dc:e3:1b:78 req_mode=00 51000000320001ffffffffffff000b50616e746865725f4d424f

b) Wildcard SSID and Wildcard BSSID:

hostapd_cli -iphy0-ap0 req_beacon c6:db:dc:e3:1b:78 req_mode=00 51000000320001ffffffffffff

c) BSSID input:

hostapd_cli -iphy0-ap0 req_beacon c6:db:dc:e3:1b:78 req_mode=00 510000000320001000c432660e8

2G_11K_MeasMode_active_ch255

a) SSID input and Wildcard BSSID:

hostapd_cli -iphy0-ap0 req_beacon c6:db:dc:e3:1b:78 req_mode=00 51ff0000320001ffffffffffff000b50616e746865725f4d424f

b) Wildcard SSID and Wildcard BSSID:

hostapd_cli -iphy0-ap0 req_beacon c6:db:dc:e3:1b:78 req_mode=00 51ff0000320001ffffffffffff

Beacon Table mode:

5G_11K_MeasMode_table_ch48

a) SSID input and Wildcard BSSID:

hostapd_cli -iphy1-ap0 req_beacon 82:c5:9c:13:9b:30 req_mode=00
73000000320002ffffffffffff000b50616e746865725f4d424f

b) Wildcard SSID and Wildcard BSSID:
hostapd_cli -iphy1-ap0 req_beacon 82:c5:9c:13:9b:30 req_mode=00 73000000320002ffffffffffff

Passive mode:
5G_11K_MeasMode_passive_ch48
a) Wildcard SSID and Wildcard BSSID:
hostapd_cli -iphy1-ap0 req_beacon 82:c5:9c:13:9b:30 req_mode=00
73300000700000ffffffffffff000b50616e746865725f4d424f
b) SSID input and Wildcard BSSID:
hostapd_cli -iphy1-ap0 req_beacon 82:c5:9c:13:9b:30 req_mode=00 73300000700000ffffffffffff

27.2 802.11V (BSS Transition Management)

BTM (BSS Transition Management):
A WNM type request frame is sent by the AP as a BTM request, containing the neighbor report element of its own BSS and nearby AP.

With a preference value to direct connected STAs to connect to preferred APs.
This is beneficial for traffic management, load balancing, noise interference, and when the AP needs to terminate.
The services are in BSS.

5G 11V BTM Req & Rsp command format:

```
hostapd_cli –i<5G interface>  bss_tm_req + STA's Mac to send btm req to STA
    /*
     * BSS Transition Candidate List Entries - Neighbor Report elements
     * neighbor=<BSSID eg: 00:20:30:40:50:60>,<BSSID Information - e.g 000>,<Operating Class - e.g 73>,
     * <Channel Number - e.g 24>,<PHY Type - 09>[,<Optional Subelements as pref - e.g - 0301ff>]
     */
```

5G_11V_diassoc_imminent bit is enabled for BTM request:
hostapd_cli -iphy1-ap0 bss_tm_req 82:c5:9c:13:9b:30 disassoc_timer=50 valid_int=200 bss_term=1,2
neighbor=02:0C:43:36:60:80,000,115,36,09,0301ff pref=1 abridged=1 disassoc_imminent=1

5G_11V_diassoc_imminent bit disabled at btm request:
hostapd_cli -iphy1-ap0 bss_tm_req 82:c5:9c:13:9b:30 valid_int=200
neighbor=02:0C:43:36:60:80,000,115,36,09,0301ff

5G_11V_neighbor pref
hostapd_cli -iphy1-ap0 -p /var/run/hostapd bss_tm_req 82:c5:9c:13:9b:30 valid_int=200
neighbor=02:0C:43:36:60:B0,000,115,36,09,0301ff neighbor=00:01:55:66:53:EB,000,73,24,09,030100

AP UCI config for 11KV - 2G
uci set wiruci set wireless.radio0.htmode=EHT40
uci set wireless.radio0.channel=6
uci set wireless.@wifi-iface[0].mode=ap
uci set wireless.@wifi-iface[0].ssid=MT76-mbo_2G
uci set wireless.@wifi-iface[0].network=lan
uci set wireless.@wifi-iface[0].ieee80211w=0
uci set wireless.@wifi-iface[0].encryption=psk2+ccmp
uci set wireless.@wifi-iface[0].key=12345678
uci set wireless.@wifi-iface[0].interworking=1
uci set wireless.radio0.disabled=0
uci set wireless.@wifi-iface[0].rrm_beacon_report=1
uci set wireless.@wifi-iface[0].bss_transition=1
uci commit wireless

27.3 802.11R (Fast Transition)

80211R:
Fast transition utilizes the ESS capability of connected BSS with backbone LAN. It reduces the roam time by sharing the previous generated PMK to target AP over the LAN where its derived actual PTK using the r0kh and r1kh intermediate key holder.

11R hostapd config:

AP1 (current AP)
nas_identifier=ap.mtk.com
r1_key_holder=<AP0's HW Addr> (Ex: 000C43280761)
mobility_domain=0101
r0_key_lifetime=10000
r1kh=<AP1's HW Addr> <AP1's HW Addr> 000102030405060708090a0b0c0d0e0f
r0kh=<AP1's HW Addr> <AP1's nas_identifier> 000102030405060708090a0b0c0d0e0f
pmk_r1_push=1

ft_over_ds=0
reassociation_deadline=1000
wpa_key_mgmt=FT-PSK
rsn_pairwise=CCMP

AP1 (Target AP)
nas_identifier=ap1.mtk.com
r1_key_holder=<AP1's HW Addr> (Ex: 000C43264698)
mobility_domain=5254
r0_key_lifetime=10000
r1kh=<AP0's HW Addr> < AP0's HW Addr > 000102030405060708090a0b0c0d0e0f
r0kh=<AP0's HW Addr> <AP0's nas_identifier> 000102030405060708090a0b0c0d0e0f
ft_over_ds=0
reassociation_deadline=1000
wpa_key_mgmt=FT-PSK
rsn_pairwise=CCMP

11R uci-conf:
AP1 : 00:0C:43:4A:D2:B9

option macaddr '00:0C:43:4A:D2:B9'
option ssid 'MT76-FT'
option ieee80211w '2'
option key '12345678'
option ieee80211r '1'
option mobility_domain '0101'
option pmk_r1_push '1'
option nasid 'ap1.mtk.com'
option r1_key_holder '000C434AD2B9'
list r0kh' 00:01:55:66:C4:43,ap2.mtk.com,000102030405060708090a0b0c0d0e0f'
listr1kh'00:01:55:66:C4:43,00:01:55:66:C4:43,000102030405060708090a0b0c0d0e0f'

option r0_key_lifetime '10000'
option reassociation_deadline '1000'
option ft_over_ds '0'
option ft_psk_generate_local '0'
option encryption 'sae'

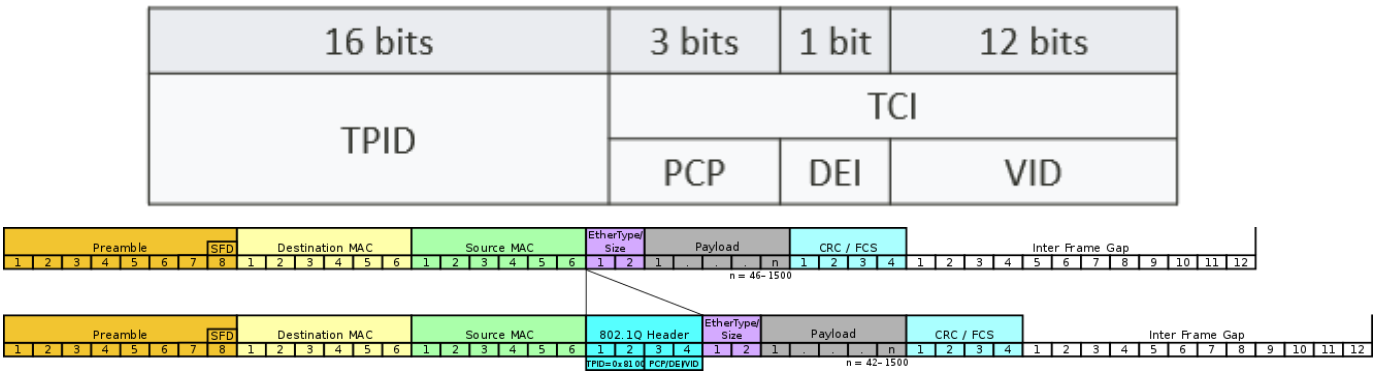
AP2 : 00:01:55:66:C4:43

option macaddr '00:01:55:66:C4:43'
option ssid 'MT76-FT'
option ieee80211w '2'
option key '12345678'
option ieee80211r '1'
option mobility_domain '0101'

```
        option pmk_r1_push '1'
        option nasid 'ap2.mtk.com'
        option r1_key_holder '00015566C443'
        list r0kh
'00:0C:43:4A:D2:B9,ap1.mtk.com,000102030405060708090a0b0c0d0e0f'
        list r1kh
'00:0C:43:4A:D2:B9,00:0C:43:4A:D2:B9,000102030405060708090a0b0c0d0e0f'
        option r0_key_lifetime '10000'
        option reassociation_deadline '1000'
        option ft_over_ds '0'
        option ft_psk_generate_local '0'
        option encryption 'sae'
```

28 VLAN

MT76 only supports transmission and receiving VLAN packet transparently; in other words, the VLAN packets with 802.1Q header is kept through Wi-Fi interfaces. If ingress packets do not include 802.1Q, the egress packets are still without 802.1Q.



802.1Q Header Format

Ensure that you configure the Wi-Fi VLAN interface and add it to the bridge when VLAN packets pass through the Wi-Fi interface, such as "phy1-ap0". The steps are as follows:

Add a VLAN interface to the Wi-Fi interface:

- Usage

```
vconfig add <interface> <vlan_id>
ifconfig <interface>.<vlan_id> up
```

- Example

```
vconfig add phy1-ap0 33
ifconfig phy1-ap0.33 up
```

Rename VLAN interface name (Optional)

- Usage

```
ip link set dev <old_name> name <new_name>
```

Add the VLAN interface to the bridge

```
brctl addif br-lan phy1-ap0.33
```

Check VLAN configuration

```
cat /proc/net/vlan/config
VLAN Dev name    | VLAN ID
Name-Type: VLAN_NAME_TYPE_RAW_PLUS_VID_NO_PAD
phy1-ap0.33    | 33  | phy1-ap0
```

After executing these steps, your platform will be configured with a WIFI VLAN interface. You can confirm the success or failure of the configuration by checking the VLAN configuration:

29 Preamble Puncture

29.1 Static Puncturing

MT76 EHT mode supports static puncturing with two configurations, mode (vendor) and bitmap (native). Static puncturing will apply the puncture bitmap rule when the ap is started.

Enable Static Preamble Puncture (e.g. 5 GHz, Channel 36, Bandwidth 160)
uci set wireless.radio1.channel=36 uci set wireless.radio1.htmode=EHT160 # hex format (0xc) is also acceptable at pp_bitmap option uci set wireless.radio1.pp_bitmap=12 uci set wireless.radio1.pp_mode=2
pp_mode=2 punct_bitmap=12

Disable Preamble Puncture
uci set wireless.radio1.pp_mode=0
pp_mode=0

29.2 Configuration

The valid bitmap pattern follows the EHT MRU pattern, and the primary channel always can't be punctured. We list all the valid puncture bitmap pattern here, and 0 is also valid.

Bandwidth (MHz)	Valid Bitmap
80	0x8, 0x4, 0x2, 0x1
160	0x80, 0x40, 0x20, 0x10, 0x8, 0x4, 0x2, 0x1, 0xc0, 0x30, 0xc, 0x3
320	0xc000, 0x3000, 0xc00, 0x300, 0xc0, 0x30, 0xc, 0x3, 0xf000, 0xf00, 0xf0, 0xf, 0xfc00, 0xf300, 0xf0c0, 0xf030, 0xf00c, 0xf003, 0xc00f, 0x300f, 0xc0f, 0x30f, 0xcf, 0x3f

pp_mode is used to tell the driver which mode to operate in preamble puncture.

Mode	Meaning
0	DISABLE, to disable the pp feature
1	FW_MODE, fw operates the bitmap
2	USR_MODE, user config the pp bitmap

We can check the current mode and bitmap values stored in the hostapd layer with the following command.

```
hostapd_cli -I <ifname> [-I <link id>] get_pp
```

30 Wi-Fi 7 Multi-Link Operation Introduction (MLO)

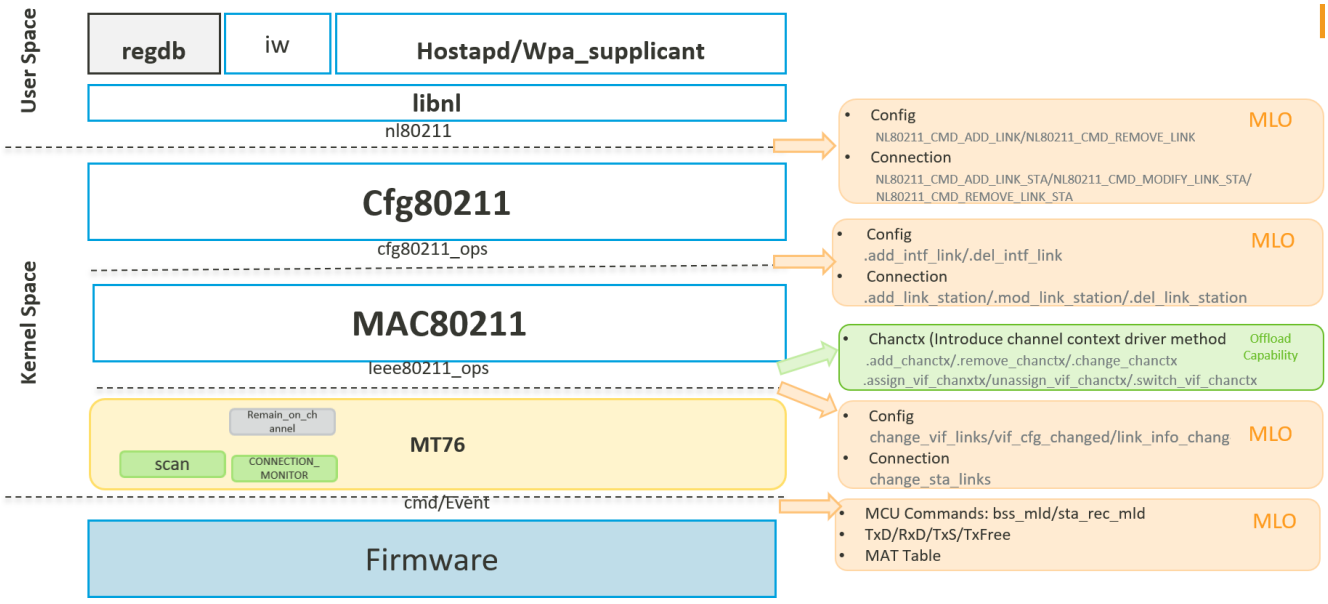
Multi-Link Operation (MLO) in Wi-Fi7 802.11be enables devices to establish and manage multiple links simultaneously, allowing for improved throughput, reliability, and seamless handover between different network connections. The MT76 BSP plays a critical role in enabling Multi-Link Operation features and ensuring compatibility with the Wi-Fi 7 standard.

Terminology
In this programming guide, the term "non-AP MLD" as defined in the IEEE 802.11be standard will be referred to interchangeably as "STA MLD" or "MLD STA." These terms are used synonymously and represent the same concept throughout this document.

30.1 Standard MLO Operations Between Layers

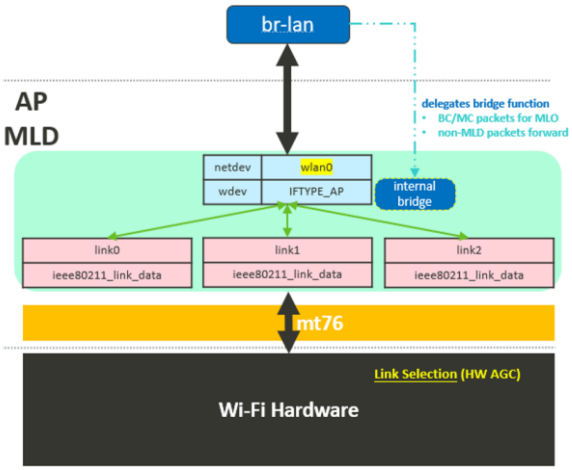
To support Multi-Link Operation in the Wi-Fi7 802.11be standard, the MT76 BSP should provide implementation for the following standard commands related to MLD Discovery and Setup.

In addition to standard commands, the MT76 BSP should include callback functions to handle events and notifications related to MLD discovery and setup. The following callback functions are essential for supporting Multi-Link Operation:



30.2 Single Wiphy Change

the Linux-wireless Multi-Link Device (MLD) design, Single MLD utilizes one netdev (network device) to represent multiple links, ensuring backward compatibility with legacy stations. An MLD Access Point (AP) is designed to have only one interface and netdev, which simplifies the representation of multiple links within the network.

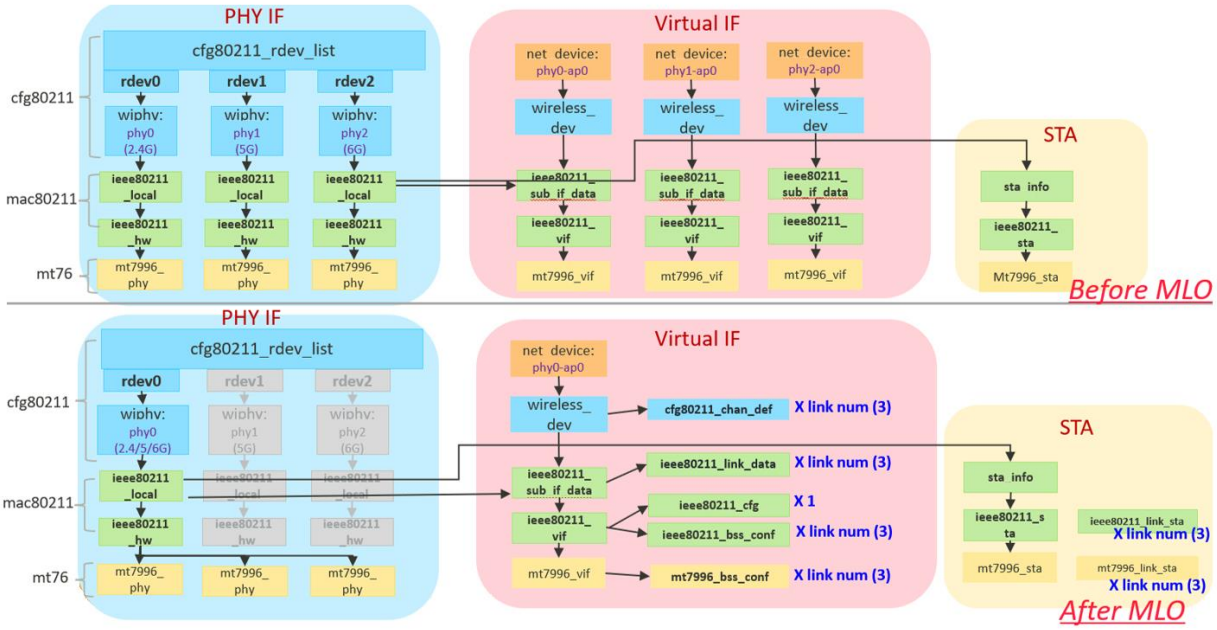


To achieve this goal, data structure changes in the kernel and Hostapd are needed to accommodate the switch to the **Single Wiphy** approach for both Legacy and MLD modes in MT76. Below are the details of the data structure changes:

CFG80211/MAC80211/MT76

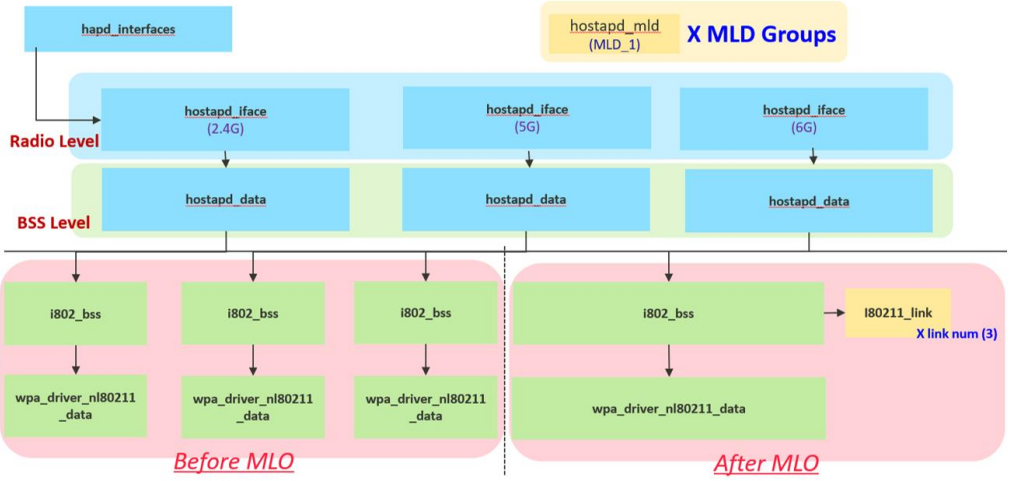
The kernel data structures need to be modified to support the Single Wiphy approach for both Legacy and MLD modes in the MT76. The following changes are required:

- **The Alpha MLD (202404) and Beta MLD (202408) version still keeps the dummy wiphy1/2, but we shall avoid using wiphy1/2 to create interfaces. Otherwise, numerous kernel panic issues will occur. It is a temporary stage and has been removed in the MLD MP (202412) version.**
- **The code before MLD MP (202412) versions is fake single wiphy, and the versions after MLD MP (202412) are real single wiphy.**



Hostapd

Similarly, the Hostapd data structures also need to be updated to align with the Single Wiphy approach for both Legacy and MLD modes in MT76. The following changes are necessary:



30.2.1 TX/RX Antenna Settings

For single wiphy, it is required to set the antenna bitmap of all the radios under the wiphy before the wiphy starts up. The UCI configuration for TX/RX antenna settings remains the same (handling the bitmap conversion in mac80211.sh). However, the definition of antenna bitmap in the iw command must change from the bitmap for a single radio to the bitmap of all radios under the Wireless PHY (wiphy). The iw command format remains unchanged.

For example, assume a single 3-radios wiphy device with a 5x5x5 TX/RX path.

As shown in the table below, the default bitmap will be 0x7fff, where bits 0 to 4 represent the antenna bitmap for radio 0, bits 5 to 9 represent the antenna bitmap for radio 1, and bits 10 to 14 represent the antenna bitmap for radio 2.
To set the antenna of radio 1 to 0x7, then the user should set the bitmap to 0x7cff, where 0x7CFF = 0x1f << 0 | 0x7 << 5 | 0x1f << 10.

Setting	Radio 0 Bit [4:0]	Radio 1 Bit [9:5]	Radio 2 Bit [14:10]	The Whole Bitmap
Default	0x1f	0x1f << 5 = 0x3e0	0x1f << 10 = 0x7c00	0x7fff
Configured	0x1f	**0x7** << 5 = 0xe0	0x1f << 10 = 0x7c00	0x7cff

The antenna mask of each radio can be checked by the following command:

```
iw phy0 info
```

```
Supported wiphy radios:
        * Idx 0:
                antenna_mask: 0x0000000f
                Frequency Range: 2400 MHz - 2500 MHz
                Radio's valid interface combinations:
                        * #{ AP, mesh point } <= 16, #{ managed } <= 19,
                          total <= 19, #channels <= 1, STA/AP BI must match, radar detect widths: { 20 MHz (no HT), 20 MHz, 40 MHz, 80 MHz, 160 MHz }

        * Idx 1:
                antenna_mask: 0x000000f0
                Frequency Range: 5000 MHz - 5900 MHz
                Radio's valid interface combinations:
                        * #{ AP, mesh point } <= 16, #{ managed } <= 19,
                          total <= 19, #channels <= 1, STA/AP BI must match, radar detect widths: { 20 MHz (no HT), 20 MHz, 40 MHz, 80 MHz, 160 MHz }

        * Idx 2:
                antenna_mask: 0x00000f00
                Frequency Range: 5925 MHz - 7200 MHz
                Radio's valid interface combinations:
                        * #{ AP, mesh point } <= 16, #{ managed } <= 19,
                          total <= 19, #channels <= 1, STA/AP BI must match, radar detect widths: { 20 MHz (no HT), 20 MHz, 40 MHz, 80 MHz, 160 MHz }
```

30.2.2 Example of Legacy AP and Legacy STA

Legacy AP

```
root@OpenWrt:~# iw dev | grep -E "Interface|ssid|type|channel|Radios"
        Interface phy0.2-ap0
                ssid OpenWrt-6g
                type AP
                channel 37 (6135 MHz), width: 320 MHz, center1: 6105 MHz
                Radios: 2
        Interface phy0.1-ap0
                ssid OpenWrt-5g
                type AP
                channel 36 (5180 MHz), width: 80 MHz, center1: 5210 MHz
                Radios: 1
        Interface phy0.0-ap0
                ssid OpenWrt-2g
                type AP
                channel 1 (2412 MHz), width: 40 MHz, center1: 2422 MHz
                Radios: 0
```

Legacy STA

```
root@OpenWrt:/# iw dev | grep -E "Interface|ssid|type|channel|Radios"
phy#0
        Interface phy0.2-sta0
                ssid test_6g
                type managed
                channel 37 (6135 MHz), width: 320 MHz, center1: 6265 MHz
                Radios: 2
        Interface phy0.1-sta0
                ssid test_5g
                type managed
                channel 36 (5180 MHz), width: 160 MHz, center1: 5250 MHz
                Radios: 1
        Interface phy0.0-sta0
                ssid test_2g
                type managed
                channel 6 (2437 MHz), width: 40 MHz, center1: 2447 MHz
                Radios: 0
```

30.2.3 Scan

In single wiphy, an AP MLD can specify the radio to scan using the following commands:

Trigger scan

```
iw <devname> scan trigger [freq <freq>*] [duration <dur>] [ies <hex as 00:11:..>]
[lowpri,flush,ap-force] [ssid <ssid>*|passive] [radios all|<id>[,<id>…]]
```

Scan & dump result

```
iw <devname> scan [-u] [freq <freq>*] [duration <dur>] [ies <hex as 00:11:..>]
[lowpri,flush,ap-force] [ssid <ssid>*|passive] [radios all|<id>[,<id>…]]
```

For example:

- Scan all the valid radios for the MLD interface (If no radio is specified, then all valid radios will be scanned).

iw <devname> scan trigger **radios all**
iw <devname> scan trigger

- Scan radio 0 & 1 for the MLD interface

iw <devname> scan trigger **radios 0,1**

- Scan radio 2 for the MLD interface

iw <devname> scan trigger **radios 2**

30.3 Porting Awareness When Transitioning to MT76 MLO BSP

Interface/Channel

- All AP_VIF and STA_VIF need to be created under wiphy0 (phy0)

iw phy **phy0** interface add <name> type <type>

- Create one MLD with all bands radio.

iw phy **phy0** interface add **ap-mld-1** type managed radios **all**

- Create one MLD with 2 GHz, 5 GHz bands radio.

iw phy **phy0** interface add **ap-mld-1** type managed radios **0,1**

- Create a legacy interface with a 5GHz bands radio.

iw phy **phy0** interface add **phy0.1-ap0** type managed radios **1**

- Create a legacy interface with 6 GHz band radio.

iw phy **phy0** interface add **phy0.2-ap0** type managed radios **2**

- STA_VIF's scan channels need to be specified in wpa_supp.conf (freq_list) in legacy mode. Otherwise, it might scan 2.4G/5G/6G channels

Band0 Scan List	2412 2417 2422 2427 2432 2437 2442 2447 2452 2457 2462 2467 2472
Band1 Scan List	5180 5200 5220 5240 5260 5280 5300 5320 5500 5520 5540 5560 5580 5600 5620 5640 5660 5680 5700 5720 5745 5765 5785 5805 5825 5845 5865 5885
Band2 Scan List	5955 5975 5995 6015 6035 6055 6075 6095 6115 6135 6155 6175 6195 6215 6235 6255 6275 6295 6315 6335 6355 6375 6395 6415 6435 6455 6475 6495 6515 6535 6555 6575 6595 6615 6635 6655 6675 6695 6715 6735 6755 6775 6795 6815 6835 6855 6875 6895

Debugfs

No matter which link (2.4G/5G/6G), we shall use the debugfs under /sys/kernel/debug/ieee80211/**phy0**

Power

In Legacy mode, the power setting for each link cannot be controlled through the WiFi device level or the 'iw phy phyX set txpower fixed xxxx' commands. It is necessary to specify the power under the wifi-iface interface and use 'iw dev phyX-ap0 set txpower fixed xxxx'.

30.4 AP MLD and STA MLD Information

MLO enables a STA MLD to discover, authenticate, associate with, and set up multiple links with an AP MLD. Each link enables channel access and frame exchanges between the STA MLD and the AP MLD based on the supported capabilities exchanged during association. The supported types are as follows:

Type	Mode	AP Support	STA Support
MLSR	MLSR	Y	N
	EMLSR	Y	N
MLMR	STR	Y	Y
	NSTR	N	N
	EMLMR	N	N

30.4.1 AP MLD and STA MLD Setting

30.4.1.1 AP MLD Setting

Just like mentioned in Wi-Fi 7 Multi-Link Device Parameters (MediaTek Vendor-Specific), takes OWE mode for example

```
config wifi-mld 'ap_mld_1'
        option ssid 'mt76_mlo_owe'
        option encryption 'owe'
        option key '12345678'
        option ifname 'ap_mld_1'
        option sae_pwe '2'
        option ieee80211w '2'
        option mld_id '1'

config wifi-iface 'default_radio0'
        option device 'radio0'
        option network 'lan'
        option mode 'ap'
        option mld_id '1'

config wifi-iface 'default_radio1'
        option device 'radio1'
        option network 'lan'
        option mode 'ap'
        option mld_id '1'

config wifi-iface 'default_radio2'
```

```
        option device 'radio2'
        option network 'lan'
        option mode 'ap'
        option mld_id '1'
```

30.4.1.2 STA MLD Setting

```
config wifi-iface 'sta_mld_1'
        option ifname 'sta_mld_1'
        option device 'radio0'
        option network 'wwan'
        option mode 'sta'
        option ssid 'mt76_mlo_owe'
        option encryption 'owe'
        option ieee80211w '2'
        option mld_assoc_phy '0'
        option mld_allowed_phy_bitmap '7'
```

30.4.2 WDS-AP MLD and WDS-STA MLD Setting

30.4.2.1 WDS-AP MLD Setting

Only one change is needed from a normal MLO configuration. (reference: Wi-Fi 7 Multi-Link Device Parameters (MediaTek Vendor-Specific))

```
config wifi-mld 'ap_mld_1'
        option ssid 'mt76_mlo'
        option encryption 'sae'
        option key '12345678'
        option ifname 'ap_mld_1'
        option mld_id '1'
        option wds '1'
```

30.4.2.2 WDS-STA MLD setting

Specific options differ based on hardware. The SSID, channel, encryption type, and password must match the access point. Enable WDS mode.

Enable WDS Options

```
vi /etc/config/wireless
…
config wifi-iface 'wifinet_band0_1'
        option device 'radio0'
        option ssid 'mt76_mlo'
        option encryption 'sae'
        option key '12345678'
        option mld_assoc_phy '0'
        option mld_allowed_phy_bitmap '7'
```

```
        option network 'lan'
        option mode 'sta'
        option wds '1'
```

The DHCP server on the LAN interface should be disabled.

```
vi /etc/config/dhcp
…
config dhcp 'lan'
        option interface 'lan'
        option start '100'
        option limit '150'
        option leasetime '12h'
        option dhcpv4 'server'
        option dhcpv6 'disabled'
        option ra 'server'
        option ra_slaac '1'
        list ra_flags 'managed-config'
        list ra_flags 'other-config'
        option ignore '1'
```

Set a static IP address

```
vi /etc/config/network
…
config interface 'lan'
        option device 'br-lan'
        option proto 'static'
        option ipaddr '192.168.1.188'
        option netmask '255.255.255.0'
        option ip6assign '60'
```

The device should reboot and automatically connect wirelessly to the access point. **Wait until the client bridge device associates with the access point. This can take 1-2 minutes for the association to happen.** After connecting, the “iwinfo” command should show a new device name (e.g., “ap_mld_1.staN” (where N is a number)) in addition to the base “ap_mld_1” wireless interface device at **WDS-AP**.

```
root@OpenWrt:/# iwinfo
ap_mld_1  ESSID: "OpenWrt"
          Access Point: 00:0C:43:28:A2:A7
          Mode: Master  Channel: 157 (5.785 GHz)
          ...
ap_mld_1.sta1 ESSID: unknown
          Access Point: 00:0C:43:28:A2:A7
          Mode: Master (VLAN)  Channel: 157 (5.785 GHz)
          …
```

30.4.3 An Example of AP MLD and STA MLD

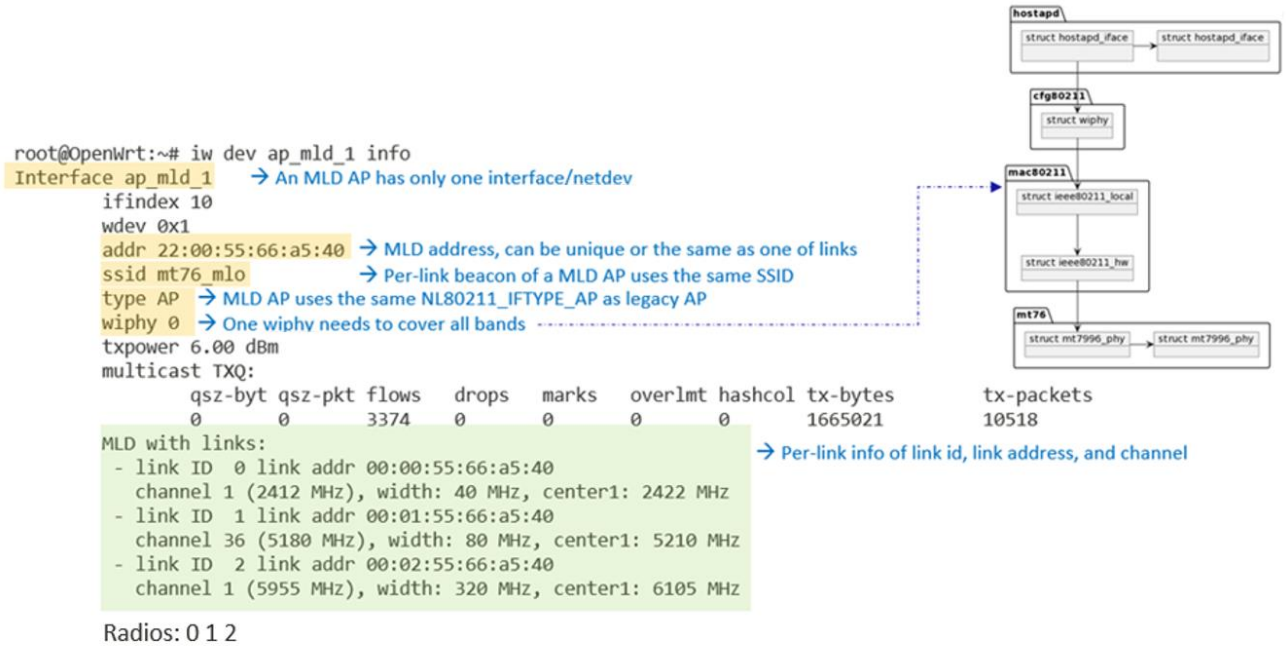
- Example: MLD AP

```
root@OpenWrt:~# iw dev
phy#0
        Interface ap-mld-1
                ifindex 13
                wdev 0x4
                addr 00:22:55:66:a1:c7
                ssid MT76_AP_MLD
                type AP
                multicast TXQ:
                        qsz-byt qsz-pkt flows   drops   marks   overlmt hashcol tx-bytes        tx-packets
                        0       0       0       0       0       0       0       0               0
                MLD with links:
                 - link ID  0 link addr 00:10:55:66:a1:c7
                   channel 1 (2412 MHz), width: 40 MHz, center1: 2422 MHz
                   txpower 30.00 dBm
                 - link ID  1 link addr 00:11:55:66:a1:c7
                   channel 36 (5180 MHz), width: 80 MHz, center1: 5210 MHz
                   txpower 23.00 dBm
                Radios: 0 1 2
```

active radios

allowed radios

- Example: MLD AP information



- Example: MLD STA information

```
root@OpenWrt:~# iw dev phy0-sta0 link
Connected to 22:0c:43:62:e2:8c (on phy0-sta0) → An MLD STA has only one interface/netdev
        SSID: mt76_mlo_ss
        Link 1 BSSID 08:0c:43:62:e2:8c
                freq: 5180.0
        Link 0 BSSID 00:0c:43:62:e2:8c
                freq: 2437.0
        Link 2 BSSID 0c:0c:43:62:e2:8c
                freq: 6135.0
MLD 22:0c:43:62:e2:8c stats:
        RX: 1046558697 bytes (8802867 packets)
        TX: 407789562 bytes (271163 packets)
        signal: -21 dBm
        tx bitrate: 1.0 MBit/s
        bss flags:
        dtim period: 1
        beacon int: 0
```

Currently MAC80211 only supports statistics of per-STA bytes, signal, and rate.
Driver prints per-link info in debugfs.

30.5 MLO Security

30.5.1 Constraints for EHT or MLO

The following content is quoted from the Chapter 11.3 of the WFA WPA3 v3.4 specification.
When an AP is operating a BSS with EHT or MLO enabled, the AP shall abide by the constraints for that BSS:

- The AP's BSS Configuration shall not allow any PSK AKM or 802.1X SHA-1 AKM to be used in an association that negotiates use of EHT or MLO.
  NOTE: The AP's BSS Configuration is still allowed to advertise these AKMs and might still enable these AKMs.
- The AP's BSS Configuration shall not enable legacy open
  NOTE: Wi-Fi Enhanced Open might be used for unauthenticated access with EHT or MLO.
- The AP's BSS Configuration shall not allow Wi-Fi Enhanced Open Transition Mode (i.e., where the OWE Transition Mode element is included in Beacons and Probe responses).

30.5.2 Configuration for EHT or MLO

The MLO Security mode must be one of the following security modes. Current Hostapd/Wpa_supplicant security requirement under MLD. You can refer to Encryption Modes for more detail setting.

Security Type	Encryption
w/ RSN Overriding	psk2 + sae + sae-ext
w/o RSN Overriding	
w/ 6GHz	*AP MLD*: sae/owe/ sae-ext (WPA3 only) *Non-AP MLD*: sae/owe/ sae-ext (WPA3 only)
w/o 6GHz	*AP MLD*: psk2/sae/owe/sae-ext (WPA3/WPA2) *Non-AP MLD*:: sae/owe/ sae-ext (WPA3 only)

Currently, MLO + 11FT-SAE (AKM9/AKM25) is partially supported. The full implementation for the WPA3 MLO FT test case is not yet completed, and patches to support this are expected to be released around June 2025.

30.5.3 RSN Overriding (RSNO)

Extensions to the RSNE and RSNXE have resulted in issues with previously deployed STAs being unable to complete connection when the AP is enabling newer functionality, e.g., when advertising multiple AKM suite selectors. Since software updates to fix these issues might not be available for some previously deployed STAs, some network deployments depend on other mechanisms to avoid known interoperability issues.

RSN overriding provides such a mechanism in a manner that allows an AP to advertise limited RSN parameters in the RSNE and the RSNXE (or fully omitting the RSNXE), so that the previously deployed STAs would not be exposed to the extensions that have resulted in issues.

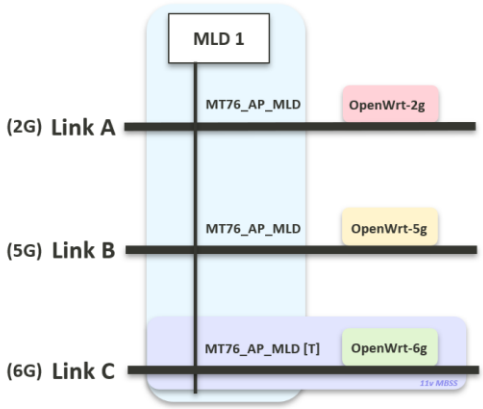
For versions after **2025/02/05**, AP has **RSNO enabled** by default. The default security settings for MLD BSS are as follows:

	Category	2GHz/5GHz	6GHz
MLD BSS	encryption	psk2	sae
	encryption_rsno	sae	
	encryption_rsno 2	sae-ext	sae-ext

Confidential C

For versions after 2025/02/05 (autobuild P66), AP has RSNO enabled by default

cat /etc/config/wireless




```
config wifi-device 'radio0'
        option type 'mac80211'
        option path 'soc/11300000...
        option radio '0'
        option band '2g'
        option channel '1'
        option htmode 'EHT40'
        option country 'US'
        option disabled '0'
        option noscan '1'
        option rnr '1'
        option tx_burst '0.0'

config wifi-iface 'default_radio0_mld'
        option device 'radio0'
        option network 'lan'
        option mode 'ap'
        option mld_id '1'
        option ieee80211w '1'
        option encryption 'psk2'
        option encryption_rsno 'sae'

config wifi-iface 'default_radio0'
        option device 'radio0'
        option network 'lan'
        option mode 'ap'
        option ssid 'OpenWrt-2g'
        option encryption 'none'
        option mbo '0'
```

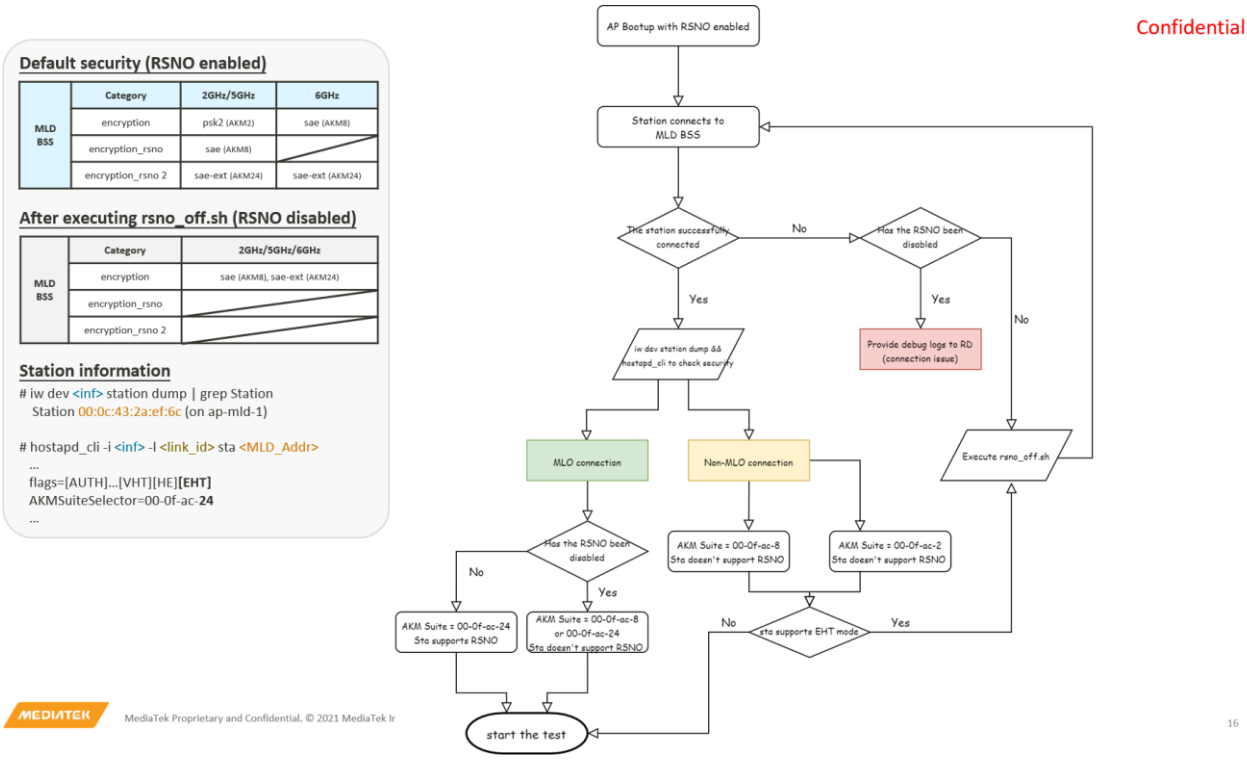
```
config wifi-device 'radio1'
        option type 'mac80211'
        option path 'soc/11300000...
        option radio '1'
        option band '5g'
        option channel '36'
        option htmode 'EHT160'
        option country 'US'
        option disabled '0'
        option noscan '1'
        option rnr '1'
        option background_radar '1'
        option tx_burst '0.0'

config wifi-iface 'default_radio1_mld'
        option device 'radio1'
        option network 'lan'
        option mode 'ap'
        option mld_id '1'
        option ieee80211w '1'
        option encryption 'psk2'
        option encryption_rsno 'sae'

config wifi-iface 'default_radio1'
        option device 'radio1'
        option network 'lan'
        option mode 'ap'
        option ssid 'OpenWrt-5g'
        option encryption 'none'
        option mbo '0'
```

```
config wifi-device 'radio2'
        option type 'mac80211'
        option path 'soc/11300000...
        option radio '2'
        option band '6g'
        option channel '37'
        option htmode 'EHT320'
        option country 'US'
        option disabled '0'
        option noscan '0'
        option mbssid '1'
        option tx_burst '0.0'

config wifi-iface 'default_radio2_mld'
        option device 'radio2'
        option network 'lan'
        option mode 'ap'
        option mld_id '1'
        option ieee80211w '2'
        option encryption 'sae'

config wifi-iface 'default_radio2'
        option device 'radio2'
        option network 'lan'
        option mode 'ap'
        option ssid 'OpenWrt-6g'
        option encryption 'sae'
        option mbo '1'
        option key '12345678'
        option sae_pwe '2'
        option ieee80211w '2'
```

	Category	2GHz/5GHz	6GHz
MLD BSS	encryption	psk2	sae
	encryption_rsno	sae	
	encryption_rsno 2	sae-ext	sae-ext

```
config wifi-mld 'ap_mld_1'
        option ifname 'ap-mld-1'
        option mld_id '1'
        option ssid 'MT76_AP_MLD'
        option encryption_rsno_2 'sae-ext'
        option key '12345678'
        option sae_pwe '2'
```

If you encounter issues connecting to the station after enabling RSNO, you can refer to the following flowchart.



- **Check the station information.**

```
root@OpenWrt:~# iw dev ap-mld-1 station dump | grep Station
Station 00:0c:43:2a:ef:6c (on ap-mld-1)
root@OpenWrt:~# hostapd_cli -i ap-mld-1 sta 00:0c:43:2a:ef:6c
00:0c:43:2a:ef:6c
flags=[AUTH][ASSOC][AUTHORIZED][SHORT_PREAMBLE][WMM][MFP][HT][HE][EHT]
aid=4
capability=0x430
listen_interval=5
supported_rates=02 04 0b 16 0c 12 18 24 30 48 60 6c
timeout_next=NULLFUNC POLL
dot11RSNAStatsSTAAddress=00:0c:43:2a:ef:6c
dot11RSNAStatsVersion=1
dot11RSNAStatsSelectedPairwiseCipher=00-0f-ac-9
dot11RSNAStatsTKIPLocalMICFailures=0
dot11RSNAStatsTKIPRemoteMICFailures=0
wpa=2
AKMSuiteSelector=00-0f-ac-8 AKM8
hostapdWPAPTKState=11
hostapdWPAPTKGroupState=0
hostapdMFPR=1
rx_packets=0
tx_packets=31
rx_bytes=576
tx_bytes=1216
inactive_msec=20050
signal=-55
rx_rate_info=60
tx_rate_info=10
ht_mcs_bitmask=ffffffff000000000000
last_ack_signal=-59
connected_time=2570
sae_group=19
sae_rejected_groups=
supp_op_classes=535153547374757677787 97b7c7d7e7f808183848586890082808785
he_capab=0108001a40002270ce920d03f3164e3f00aaffaaff1b1cc7711cc771
eht_capab=8201e801031e2860000000444444
ht_caps_info=0x09ef
ext_capab=04004a0201404040000120
peer_addr[0]=00:0c:43:2a:f0:6c
peer_addr[1]=00:0c:43:2a:f1:6c
peer_addr[2]=00:0c:43:2a:f2:6c
max_simul_links=2
```

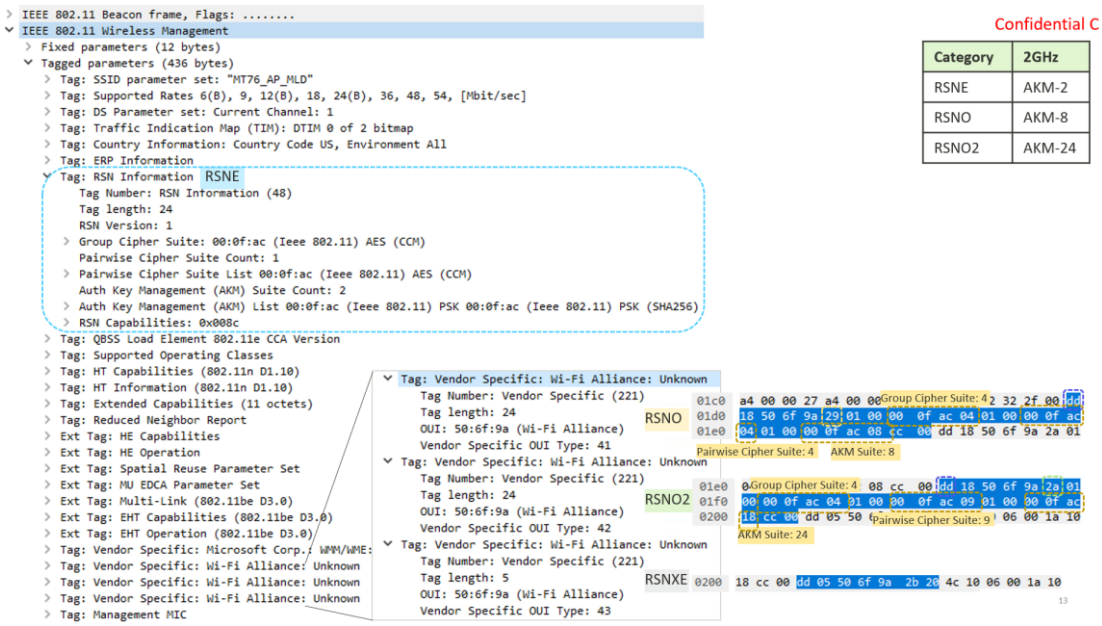
(This station uses AKM8 to connect to AP)

30.5.3.1 AP RSNO Setting

Take three-link MLD BSS as an example.

```
#new phy0 AP interface
uci set wireless.default_radio0_mld=wifi-iface
uci set wireless.default_radio0_mld.device=radio0
uci set wireless.default_radio0_mld.network=lan
uci set wireless.default_radio0_mld.mode=ap
uci set wireless.default_radio0_mld.mld_id=1
uci set wireless.default_radio0_mld.ieee80211w=1
uci set wireless.default_radio0_mld.encryption=psk2
uci set wireless.default_radio0_mld.encryption_rsno=sae

#new phy1 AP interface
```

```
uci set wireless.default_radio1_mld=wifi-iface
uci set wireless.default_radio1_mld.device=radio1
uci set wireless.default_radio1_mld.network=lan
uci set wireless.default_radio1_mld.mode=ap
uci set wireless.default_radio1_mld.mld_id=1
uci set wireless.default_radio1_mld.ieee80211w=1
uci set wireless.default_radio1_mld.encryption=psk2
uci set wireless.default_radio1_mld.encryption_rsno=sae


#new phy2 AP interface
uci set wireless.default_radio2_mld=wifi-iface
uci set wireless.default_radio2_mld.device=radio2
uci set wireless.default_radio2_mld.network=lan
uci set wireless.default_radio2_mld.mode=ap
uci set wireless.default_radio2_mld.mld_id=1
uci set wireless.default_radio2_mld.ieee80211w=2
uci set wireless.default_radio2_mld.encryption=sae

#MLD
uci set wireless.ap_mld_1=wifi-mld
uci set wireless.ap_mld_1.ssid=MT7996_MLD_AP
uci set wireless.ap_mld_1.encryption_rsno_2=sae-ext
uci set wireless.ap_mld_1.key=12345678
uci set wireless.ap_mld_1.mld_id=1
uci set wireless.ap_mld_1.mld_addr='00:0c:33:44:12:9a'
uci set wireless.ap_mld_1.ifname=ap-mld-1

uci set wireless.ap_mld_1.wds=1

#Disable other bands and default interfaces
uci set wireless.radio0.disabled=0
uci set wireless.radio1.disabled=0
uci set wireless.radio2.disabled=0
uci set wireless.default_radio0.disabled=1
uci set wireless.default_radio1.disabled=1
uci set wireless.default_radio2.disabled=1

#Save and reload
uci commit wireless
wifi
```



Confidential C

Category	2GHz
RSNE	AKM-2
RSNO	AKM-8
RSNO2	AKM-24

The 2G beacon frame with RSNO enabled.

30.5.3.1 STA RSNO Setting

Take three-link MLD STA as an example.

The MT76 Station can only use RSNO2 to connect to the MT76 AP for MLO.

```
#new phy1 STA interface
uci set wireless.wifinet_band0_1=wifi-iface
uci set wireless.wifinet_band0_1.device=radio1
uci set wireless.wifinet_band0_1.network=lan
uci set wireless.wifinet_band0_1.ssid=MT7996_MLD_AP
uci set wireless.wifinet_band0_1.mode=sta
uci set wireless.wifinet_band0_1.encryption=sae-ext
uci set wireless.wifinet_band0_1.key=12345678
uci set wireless.wifinet_band0_1.mld_assoc_phy=1
uci set wireless.wifinet_band0_1.mld_allowed_phy_bitmap=7

uci set wireless.wifinet_band0_1.wds=1

#Disable other bands and default interfaces
uci set wireless.radio0.disabled=1
uci set wireless.radio1.disabled=0
uci set wireless.radio2.disabled=1
uci set wireless.default_radio0.disabled=1
uci set wireless.default_radio1.disabled=1
uci set wireless.default_radio2.disabled=1


#Save and reload
uci commit wireless
wifi
```

30.6 Enhanced Multi-link Single Radio (EMLSR) Operation

30.6.1 Setting

The EMLSR feature is currently supported only by the AP. STA mode does not support EMLSR.

There are two options available for EMLSR configuration:

1. **eml_disable**: Disable or enable the EMLSR feature
   - For **AP** MLD
     - '0' -> Enable EMLSR feature
     - '1' -> Disable EMLSR feature
2. **eml_resp**: Disable/Enable EML Operating Mode Notification
   - For **AP** MLD
     - '0' -> AP will not response EML OMN action frame
     - '1' -> AP will response EML OMN action frame

UCI command to set eml_disable/eml_resp:

uci set wireless.ap_mld_1.eml_disable=<value>

uci set wireless.ap_mld_1.eml_resp=<value>

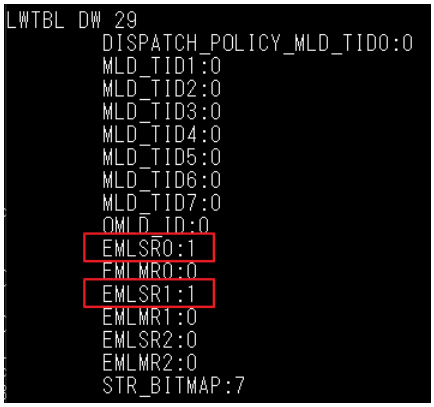
Setting in etc/config/wireless:

```
config wifi-mld 'ap_mld_1'
        option ssid 'mt76-mld'
        option encryption 'sae'
        option key '12345678'
        option mld_id '1'
```

```
        option ifname 'ap-mld-1'
        option eml_disable '0'
        option eml_resp '1'
```

After the station sends an EML OMN action frame to the AP to enter EMLSR mode, we can check the EMLSR value by WTBL on the AP:

```
echo <wlan_idx> > sys/kernel/debug/ieee80211/phy0/mt76/wlan_idx
cat sys/kernel/debug/ieee80211/phy0/mt76/wtbl_info
```



30.6.2 Check station's EML capability

Once the station is connected to the AP, we can use the hostapd_cli command to verify the station's EML capability.

```
hostapd_cli -i <interface> -l <link_id> sta <MLD address>
e.g. hostapd_cli -i ap-mld-1 -l 0 sta 06:0c:43:2a:ee:f3
 ...
 peer_addr[0]=06:0c:43:2a:ee:f2
 peer_addr[1]=06:0c:43:2a:ee:f3
 nstr_bitmap=0x05
 peer_addr[2]=06:0c:43:2a:ee:f4
 nstr_bitmap=0x03
 max_simul_links=2
 emlsr_support=0
 emlmr_support=0
```

30.7 MLO Reconfiguration

During link reconfiguration, the AP MLD must already exist for adding or removing an affiliated AP of itself. **It should be noted that the creation and destruction of AP MLD is NOT supported** in the MLO link reconfiguration.

30.7.1 Add Link

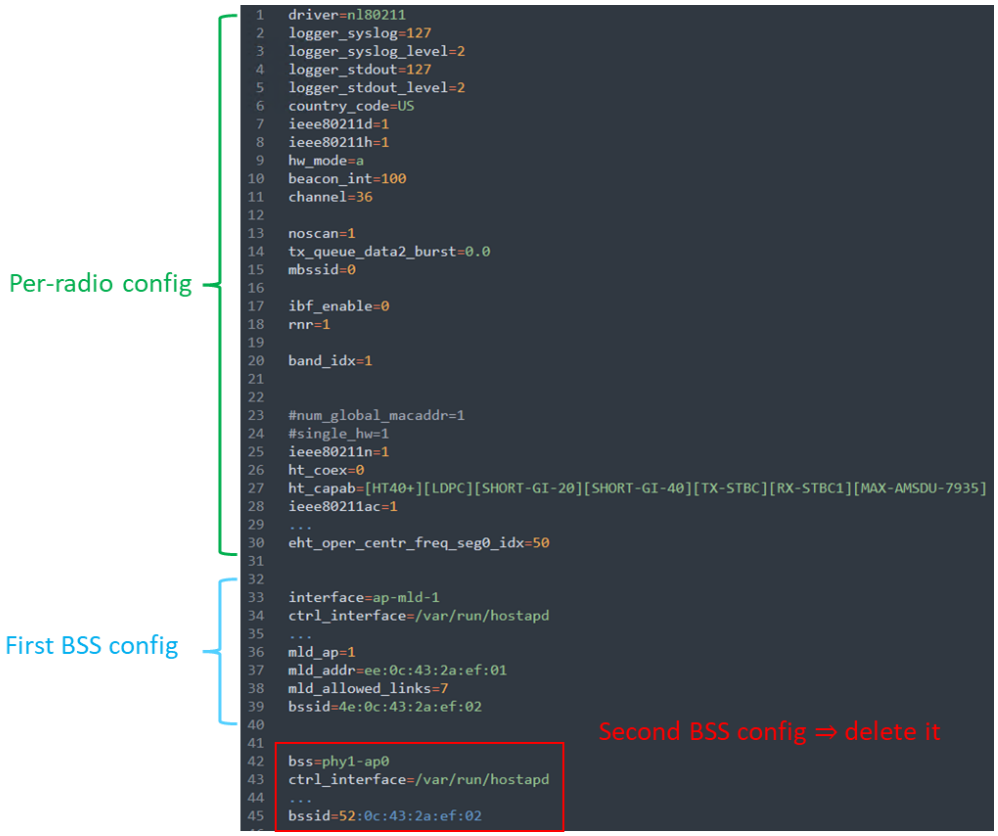
To add an affiliated AP to the AP MLD, please enter the following command:

MLD AP	MLD STA
hostapd_cli -i <interface> link_add bss_config=phyX:<bss_config>	**The link reconfiguration request frame is NOT supported yet.**

	MT76 STA will **trigger reconnection** as a workaround when it identifies that an affiliated AP has been added to its associated AP MLD.

Note that the <bss_config> is a "**Single BSS**" hostapd config that user should generate on their own; meaning only one BSS can be included in the config.

For UCI users, the template of hostapd config can be found at /var/run/hostapd-phyX.conf(.prev).
If the template you get contains multiple BSS, you can simply delete the part shown in the following figure to get a "Single BSS" hostapd config.



Also, please ensure that there is **no existing legacy AP on the same band as the added link**. Otherwise, it would fail due to the limitation of hostapd (AP MLD should be the first BSS of a band).

30.7.2 Remove Link

To remove an affiliated AP from the AP MLD, please enter the following command:

MLD AP	MLD STA
hostapd_cli -i <interface> -l <link_id> link_remove count=<count>	When an MLD STA receives the reconfiguration ML IE from the MLD AP, it will remove the connected link automatically after the timeout of the removal timer.

Note that the < count > specifies the countdown number, in terms of beacons, before the link is removed.
Due to firmware limitations, the total countdown time must be a multiple of seconds.
That is, (count * beacon interval [ms]) % 1000 should be 0.

30.8 MLO Link Management

30.8.1 Advertised TTLM (Adv-TTLM)

For example, a 3-link AP MLD is about to disable its 5GHz link (link ID=1) by Adv-TTLM, which will be establish in 3 seconds and last for 10 seconds

MLD AP	MLD STA
#Usage hostapd_cli -i <interface> set_attlm disabled_links=<link_bitmap> switch_time=<switch time in ms> duration=<duration in ms> #Example hostapd_cli -i ap-mld-1 set_attlm disabled_links=2 switch_time=3000 duration=10000	When an MLD STA receives the beacon with TTLM from the MLD AP, it will disable the specified link after the switch time and enable the specified link after the duration expires.

30.8.2 Negotiated TTLM (Neg-TTLM)

Currently, Negotiated TTLM request by AP MLD is not supported.

For example, a 3-link STA MLD is about to disable its 5GHz link (link ID=1) by Neg-TTLM, and then the Neg-TTLM is tore down by AP MLD or STA MLD

Command	MLD AP	MLD STA
Neg-TTLM setup	N/A	#Usage wpa_cli -i <ifname> neg_ttlm_setup bidi <tid0 bitmap> <tid1 bitmap> <tid2 bitmap> <tid3 bitmap> <tid4 bitmap> <tid5 bitmap> <tid6 bitmap> <tid7 bitmap> #Example wpa_cli -i sta-mld-1 neg_ttlm_setup bidi 5 5 5 5 5 5 5 5
Neg-TTLM teardown	hostapd_cli -i <ifname> neg_ttlm_teardown <STA's MLD address>	wpa_cli -i <ifname> neg_ttlm_teardown
Dump Neg-TTLM status	hostapd_cli -i <ifname> get_neg_ttlm <STA's MLD address>	N/A

30.9 Pre-MLD and Per-Link Wi-Fi Statistics

MT76 can provide the per-MLD and also per-link information (signal/bitrate/bytes count) shown as below; you can refer to MLO Statistics Knobs (Only for Wi-Fi 7) for more information.

Assuming a wireless client is connected to the DUT, use the following command to get its per-station (a.k.a. per-MLD) and per-link information. **iw [ifname] station dump** e.g.

```
root@OpenWrt:/# iw ap-mld-1 station dump
Station 00:0c:43:4b:ec:d0 (on ap-mld-1)
        inactive time:  5908 ms
        rx bytes:       37920
        rx packets:     14970
        tx bytes:       17679424
        tx packets:     313136
        tx retries:     151
        tx failed:      1020
        rx drop misc:   5
        signal:         -9 [-11, -13, -59, -60] dBm
        signal avg:     -10 [-12, -13, -59, -59] dBm
        tx bitrate:     206.4 MBit/s 40MHz EHT-MCS 4 EHT-NSS 2 EHT-GI 0
        tx duration:    10457513 us
        rx bitrate:     68.8 MBit/s EHT-MCS 5 EHT-NSS 1 EHT-GI 0
        rx duration:    952316 us
        last ack signal:-9 dBm
        avg ack signal: -6 dBm
        airtime weight: 256
        authorized:     yes
        authenticated:  yes
        associated:     yes
        preamble:       short
        WMM/WME:        yes
        MFP:            yes
        TDLS peer:      no
        DTIM period:    0
        beacon interval:0
        connected time: 356 seconds
        associated at [boottime]:       258.660s
        associated at:  1699274033138 ms
        current time:   1699274389257 ms
        *** MLD Information ***
        ***** Link ID:  0 *****
        Link addr: 00:0c:43:4b:ed:d0
        rx bytes:       37920
        rx mpdus:       279
        rx fcs errors: 5
        tx bytes:       17679424
        tx retries:     151
        tx failed:      1020
        signal:         -29 [-36, -71, -29, -68] dBm
        signal avg:  -27 [-34, -69, -27, -68] dBm
        tx bitrate:     206.4 MBit/s 40MHz EHT-MCS 4 EHT-NSS 2 EHT-GI 0
        tx duration:    2246792 us
```

```
rx bitrate:     68.8 MBit/s EHT-MCS 5 EHT-NSS 1 EHT-GI 0
rx duration:    192810 us
last ack signal:-9 dBm
avg ack signal: -6 dBm
DTIM period:    2
beacon interval:100
***** Link ID:  1 *****
Link addr: 00:0c:43:4b:ee:d0
rx bytes:       462560
rx mpdus:       3403
rx fcs errors: 61
tx bytes:       0
tx retries:     0
tx failed:      0
signal:         -38 [-38, -59, -66, -63] dBm
signal avg:  -37 [-37, -65, -65, -64] dBm
tx bitrate:     6.0 MBit/s
tx duration:    418 us
rx bitrate:     1297.0 MBit/s 160MHz EHT-MCS 4 EHT-NSS 3 EHT-GI 0
rx duration:    336805 us
last ack signal:-27 dBm
avg ack signal: -26 dBm
DTIM period:    2
beacon interval:100
***** Link ID:  2 *****
Link addr: 00:0c:43:4b:ef:d0
rx bytes:       512960
rx mpdus:       3774
rx fcs errors: 68
tx bytes:       440461472
tx retries:     3351
tx failed:      2487
signal:         -46 [-46, -84, -84, -84] dBm
signal avg:  -43 [-43, -83, -83, -83] dBm
tx bitrate:     2882.3 MBit/s 320MHz EHT-MCS 13 EHT-NSS 1 EHT-GI 0
tx duration:    8210303 us
rx bitrate:     2882.3 MBit/s 320MHz EHT-MCS 13 EHT-NSS 1 EHT-GI 0
rx duration:    422701 us
last ack signal:-37 dBm
avg ack signal: -36 dBm
DTIM period:    2
beacon interval:100
```

More counters can be dumped under debugfs, please refer to 5.18 MLO Statistics Knobs (Only for Wi-Fi 7).

30.10 MLO with MBSS

The **total number of AP Wi-Fi interfaces (Legacy AP + AP MLD)** per band: **up to 16 MBSS.**

The **total number of Station interfaces (Legacy STA + STA MLD)** per band: **up to 1 STA.** (Either the Legacy STA or the STA MLD)

- **STA WDS mode:** Interfaces can be initiated on all three bands.
- **STA NAT mode:** Only one band interface can be selected for the legacy interface.

STA					
Connection mode	**Scenario**	**Interface type**	**2G**	**5G**	**6G**
WDS	Scenario 1	Legacy interface	O	O	O
		MLD interface			
	Scenario 2	Legacy interface			
		MLD interface	O	O	O
NAT	Scenario 3	Legacy interface	Δ	Δ	Δ
		MLD interface			
	Scenario 4	Legacy interface			
		MLD interface	O	O	O

O: Both can exist simultaneously or partially,

Δ: Only one band can be selected out of the three bands.

The following are examples of the number of MBSS:

- The number of MBSS per band for AP ≤ 16, and the number of STA per band for STA ≤1 (valid)

1)

Type	Mode	2G	5G	6G
AP	**Legacy interface**	3	3	3
	MLD interface (3 links, 2+5+6)	1	1	1
	Total MBSS	**4**	**4**	**4**
STA	**Legacy interface**	1	0	0
	Total STA	**1**	**0**	**0**

2)

Type	Mode	2G	5G	6G
AP	**Legacy interface**	14	10	10
	MLD interface (3 links, 2+5+6)	1	1	1
	MLD interface (2 links, 2+5)	1	1	0
	MLD interface (2 links, 5+6)	0	1	1
	Total MBSS	**16**	**13**	**12**
STA	**MLD interface (2 links, 2+5)**	1	1	0
	Total STA	**1**	**1**	**0**

- The number of MBSS per band for AP >16, and the number of STA per band for STA > 1 (invalid)

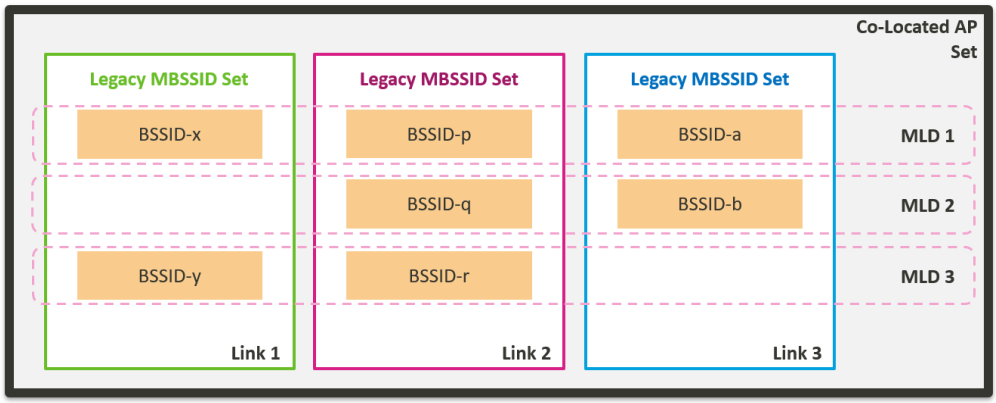
1)

Type	Mode	2G	5G	6G
AP	**Legacy interface**	16	15	15
	MLD interface (3 links, 2+5+6)	1	1	1
	Total MBSS	**17**	**16**	**16**
STA	**Legacy interface**	1	1	1
	MLD interface (2 links, 2 + 5)	1	1	0
	Total STA	**2**	**2**	**1**

2)

Type	Mode	2G	5G	6G
AP	Legacy interface	14	10	10
	MLD interface (3 links, 2 + 5 + 6)	1	1	1
	MLD interface (2 links, 2+5)	1	1	0
	MLD interface (2 links, 5+6)	0	1	1
	MLD interface (2 links, 2+6)	1	0	1
	Total MBSS	17	13	13
STA	MLD interface (3 links, 2+5+6)	1	1	1
	MLD interface (2 links, 2+5)	1	1	0
	Total STA	2	2	1

30.10.1 Legacy MBSSID



Legacy BSS with MLD groups

Here is an example of adding a Legacy MBSS with multiple MLDs.

```
# Add Legacy MBSS
uci set wireless.phy0_ap1='wifi-iface'
uci set wireless.phy0_ap1.disabled=0
uci set wireless.phy0_ap1.device=radio0
uci set wireless.phy0_ap1.mode=ap
uci set wireless.phy0_ap1.network=lan
# Set the MLD ID for the Legacy MBSS
uci set wireless.phy0_ap1.mld_id=2
# Set the MLD group configuration
uci set wireless.ap_mld_2='wifi-mld'
uci set wireless.ap_mld_2.ifname=ap_mld_2
uci set wireless.ap_mld_2.mld_id=2
uci set wireless.ap_mld_2.ssid=Mld_group_2
uci set wireless.ap_mld_2.encryption=sae
uci set wireless.ap_mld_2.ieee80211w=2
uci set wireless.ap_mld_2.key=7pKG1yDy34YC1nLnV
```

An example is provided for setting up the second 3-link AP MLD based on our default setting.

- AP

 # Add Legacy MBSS
 # Keep the default configuration files (phy0-ap0, phy1-ap0, phy2-ap0), and Add Legacy MBSS (phy0-ap1, phy1-ap1, phy2-ap1)
 # Interface phy0_ap1 configuration
uci set wireless.phy0_ap1='wifi-iface'
uci set wireless.phy0_ap1.disabled=0
uci set wireless.phy0_ap1.device=radio0
uci set wireless.phy0_ap1.mode=ap
uci set wireless.phy0_ap1.network=lan
 # Interface phy1_ap1 configuration
uci set wireless.phy1_ap1='wifi-iface'
uci set wireless.phy1_ap1.disabled=0
uci set wireless.phy1_ap1.device=radio1
uci set wireless.phy1_ap1.mode=ap
uci set wireless.phy1_ap1.network=lan
 # Interface phy2_ap1 configuration
uci set wireless.phy2_ap1='wifi-iface'
uci set wireless.phy2_ap1.disabled=0
uci set wireless.phy2_ap1.device=radio2
uci set wireless.phy2_ap1.mode=ap
uci set wireless.phy2_ap1.network=lan
 # MT76 interfaces configuration for MLD_ID1
uci set wireless.phy0_ap0.mld_id=1
uci set wireless.phy1_ap0.mld_id=1
uci set wireless.phy2_ap0.mld_id=1
 # MT76 interfaces configuration for MLD_ID2
uci set wireless.phy0_ap1.mld_id=2
uci set wireless.phy1_ap1.mld_id=2
uci set wireless.phy2_ap1.mld_id=2
 # MT76 AP MLD: MLD_ID1 configuration
uci set wireless.ap_mld_1='wifi-mld'
uci set wireless.ap_mld_1.ifname=ap_mld_1
uci set wireless.ap_mld_1.mld_id=1
uci set wireless.ap_mld_1.ssid=Mld_group_1
uci set wireless.ap_mld_1.encryption=owe
uci set wireless.ap_mld_1.ieee80211w=2
 # MT76 AP MLD: MLD_ID2 configuration
uci set wireless.ap_mld_2='wifi-mld'
uci set wireless.ap_mld_2.ifname=ap_mld_1
uci set wireless.ap_mld_2.mld_id=2
uci set wireless.ap_mld_2.ssid=Mld_group_2
uci set wireless.ap_mld_2.encryption=sae
uci set wireless.ap_mld_2.ieee80211w=2
uci set wireless.ap_mld_2.key=7pKG1yDy34YC1nLnV

#commit uci config
uci commit wireless
wifi

- STA

 # station can only connect to a single AP MLD at most.
 # Connects to MLD_ID1
 # MT76 STA MLD: sta_mld_1 configuration
uci set wireless.sta_mld_1='wifi-iface'
uci set wireless.sta_mld_1.ifname=sta_mld_1
uci set wireless.sta_mld_1.device=radio0
uci set wireless.sta_mld_1.network=wwan
uci set wireless.sta_mld_1.mode=sta
uci set wireless.sta_mld_1.ssid=Mld_group_1
uci set wireless.sta_mld_1.encryption=owe
uci set wireless.sta_mld_1.ieee80211w=2
uci set wireless.sta_mld_1.mld_assoc_phy=0
uci set wireless.sta_mld_1.mld_allowed_phy_bitmap=7
uci set wireless.sta_mld_1.disabled=0

#commit uci config
uci commit wireless
wifi

or

 # Connects to MLD_ID2
 # MT76 STA MLD: sta_mld_1 configuration
uci set wireless.sta_mld_1='wifi-iface'
uci set wireless.sta_mld_1.ifname=sta_mld_1
uci set wireless.sta_mld_1.device=radio0
uci set wireless.sta_mld_1.network=wwan
uci set wireless.sta_mld_1.mode=sta
uci set wireless.sta_mld_1.ssid=Mld_group_2
uci set wireless.sta_mld_1.encryption=sae
uci set wireless.sta_mld_1.ieee80211w=2
uci set wireless.sta_mld_1.key=7pKG1yDy34YC1nLnV
uci set wireless.sta_mld_1.mld_assoc_phy=0
uci set wireless.sta_mld_1.mld_allowed_phy_bitmap=7
uci set wireless.sta_mld_1.disabled=0

#commit uci config
uci commit wireless
wifi

30.10.2 11v MBSS

Currently, 11v MBSS with MLD groups does not support A-TTLM link management.

Follow the rules to setup a legacy MBSSID in 30.10.1, and setup the 11v MBSS option on a radio.

```
uci set wireless.radio0.mbssid=1
```

According to the MMPDU (Management Frame Protocol Data Unit) size limitation, the beacon length has its bottleneck, and the specification should be less than 1820 bytes.

For reference, in the current OpenWrt WiFi configuration with an SDK build, a multi-link device (MLD) with the 11v MBSS (Multiple BSSID) case should **have a maximum number of MBSS less than 6 when considering the inheritance of the RSNO vendor information element (IE), or up to 10 MBSS if RSNO is turned off.** This number may vary if the hostapd configuration enables feature sets related to beacon size.

30.11 Link-Specific Operation over MLD netdev

30.11.1 Hostapd Control Interface

```
root@OpenWrt:/# hostapd_cli -i
hostapd_cli: option requires an argument: -i

usage: hostapd_cli [-p<path>] [-i<ifname>] [-l<link_id>] [-hvBr] [-a<path>] \
            [-P<pid file>] [-G<ping interval>] [command..]

Options:
  -h          help (show this usage text)
  -v          shown version information
  -p<path>    path to find control sockets (default: /var/run/hostapd)
  -s<dir_path> dir path to open client sockets (default: /var/run/hostapd)
  -a<file>    run in daemon mode executing the action file based on events
              from hostapd
  -r          try to reconnect when client socket is disconnected.
              This is useful only when used with -a.
  -B          run a daemon in the background
  -i<ifname>  Interface to listen on (default: first interface found in the
              socket path)
  -l<link_id>  Link ID of the interface in case of Multi-Link
              Operation
```

MLD Level	hostapd_cli -i ap-mld-1 stat	name=ap-mld-1 mld_address=00:22:55:66:a0:82 num_links=3 LINK INFORMATION link_id=0 link_addr=00:10:55:66:a0:82 link_id=1 link_addr=00:11:55:66:a0:82 link_id=2 link_addr=00:12:55:66:a0:82
Link Level	hostapd_cli -i ap-mld-1 -l0 stat	phy=phy0 freq=2412 num_links=3 link_id=0 link_addr=00:10:55:66:a0:82 partner_link[1]=00:11:55:66:a0:82 partner_link[2]=00:12:55:66:a0:82 bss[0]=ap-mld-1 bssid[0]=00:10:55:66:a0:82 ssid[0]=MT76_AP_MLD num_sta[0]=0 mld_addr[0]=00:22:55:66:a0:82 mld_id[0]=0 mld_link_id[0]=0 bss[1]=phy0-ap0 bssid[1]=00:00:55:66:a0:82 ssid[1]=OpenWrt-2g num_sta[1]=0
	hostapd_cli -i ap-mld-1 -l1 stat	phy=phy1 freq=5180 num_links=3 link_id=1 link_addr=00:11:55:66:a0:82 partner_link[0]=00:10:55:66:a0:82 partner_link[2]=00:12:55:66:a0:82 bss[0]=ap-mld-1 bssid[0]=00:11:55:66:a0:82 ssid[0]=MT76_AP_MLD num_sta[0]=0 mld_addr[0]=00:22:55:66:a0:82 mld_id[0]=0 mld_link_id[0]=1 bss[1]=phy1-ap0 bssid[1]=00:01:55:66:a0:82 ssid[1]=OpenWrt-5g
	hostapd_cli -i ap-mld-1 -l2 stat	phy=phy2 freq=6135 num_links=3 link_id=2 link_addr=00:12:55:66:a0:82 partner_link[0]=00:10:55:66:a0:82 partner_link[1]=00:11:55:66:a0:82 bss[0]=ap-mld-1

		bssid[0]=00:12:55:66:a0:82 ssid[0]=MT76_AP_MLD num_sta[0]=0 mld_addr[0]=00:22:55:66:a0:82 mld_id[0]=0 mld_link_id[0]=2 bss[1]=phy2-ap0 bssid[1]=00:02:55:66:a0:82 ssid[1]=OpenWrt-6g

30.11.2 Scan

```
AP:

# trigger scan
$ iw dev ap_mld_1 scan

# check the survey results
$ iw ap_mld_1 survey dump

Station:

Method 1:
# trigger scan
$ wpa_cli -i sta_mld_1 scan

# get the scan result
$ wpa_cli -i sta_mld_1 scan_results

Method 2:
# trigger scan
$ iw dev sta_mld_1 scan
```

30.11.3 Offchannel Scan

```
iw <devname> offchannel [-l link_id] <freq> <duration>
```

30.11.4 Channel Switch

```
hostapd_cli -i <devname> -l <link_id>
chan_switch <cs_count> <freq> [sec_channel_offset=offset] [center_freq1=cfreq1]
[center_freq2=cfreq2] [bandwidth=MHz][blocktx] [ht|vht|he|eht] [skip_cac]
```

Currently, a channel switch of one link during the progress of the channel switch of another link affiliated with the same AP MLD is NOT allowed.

For DFS radar detection during MLD channel switch, the radar detection event will be blocked until the channel switch is done.

30.11.5 Power

Make sure the txpower_sku is enabled, the value shall be 0 when check /sys/kernel/debug/ieee80211/phy0/mt76/sku_disable

```
# Set txpower per link [1dBm = 100 mBm]
iw dev <devname> set txpower [-l <link_id>] <auto|fixed|limit> [<tx power in mBm>]

# Note that the txpower will select the minimum value, so we should set a value less than
the current txpower to take effect.
```

30.11.6 Auto Channel Selection

One Link or Multi-Links ACS over MLD on the channel configuration within each radio in the hostapd settings.
In Hostapd configuration file. (Per-Radio)

```
channel=0
or
channel=acs_survey
```

In UCI (/etc/config/wireless) configuration file

```
#Radio0
uci set wireless.radio0.channel=0

#Three-Links/Radios need to perform CAC
uci set wireless.radio0.channel=0
uci set wireless.radio1.channel=0
uci set wireless.radio2.channel=0
```

30.12 MLO DFS Behavior

In this section, the behavior of DFS radar detection (CAC) under MLD will be introduced in two parts: bootup CAC and CAC after channel switch.

Note that the behavior of CAC under MLD for MT76 is **not the same** as that of the proprietary driver Logan.
For more information of the basic DFS mechanism, please refer to Dynamic Frequency Selection (DFS).

Using the 3 link MT7996 as a reference, the following examples will illustrate the connection status of the AP MLD and the non-AP MLD (STA MLD) during the CAC process.

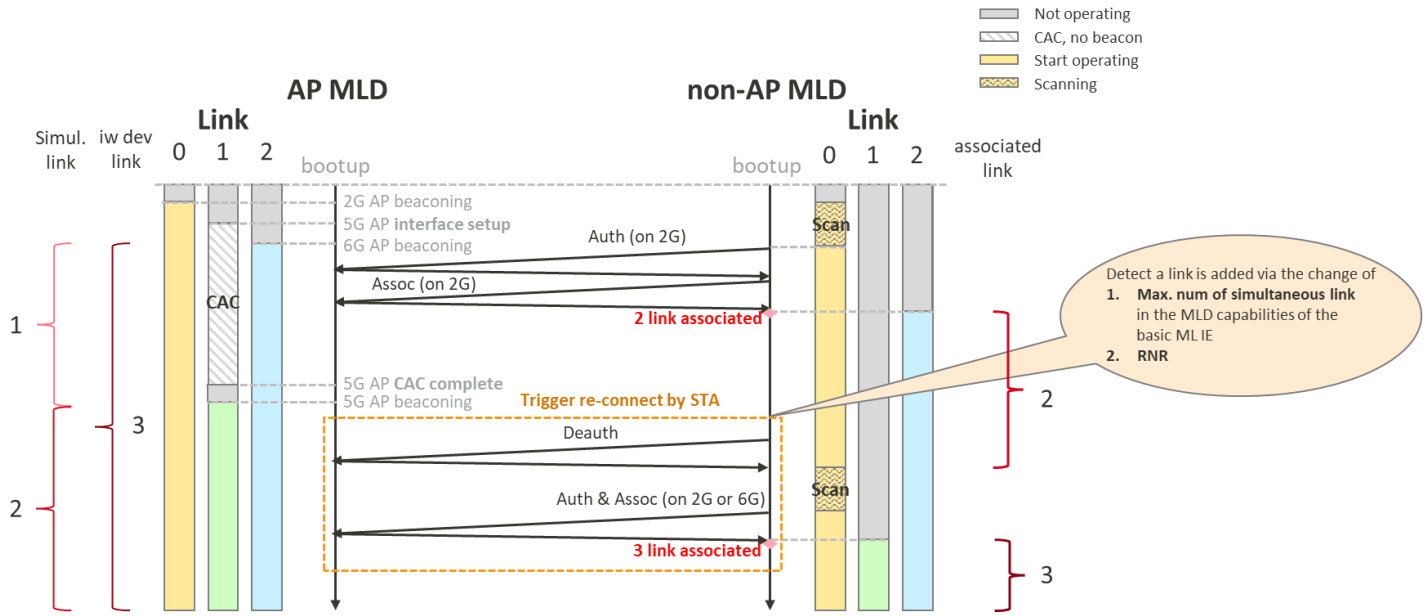
30.12.1 Bootup Radar Detection (CAC)

- **Without background radar (ZWDFS, or called dedicated RX)**

After the bootup of the AP MLD interface, the 5G link will perform CAC before beaconing if the 5G channel configured by the user contains a USABLE DFS channel. Note that the information of the 5G affiliated AP of this MLD will not be included in the RNR of the beacon of its partner links. Therefore, the STA MLD will only connect to 2G and 6G links when the 5G AP is performing the CAC. **After the CAC is completed, STA MLD will detect that an affiliated AP has been added to its associated AP MLD from either the change in the max number of simultaneous link field in the MLD capabilities of the Basic ML IE or the change in the RNR**. Then, a **re-connection request** will be triggered automatically by the STA if the added link matches the capability of the STA for now. After that, the STA will be connected to three links.

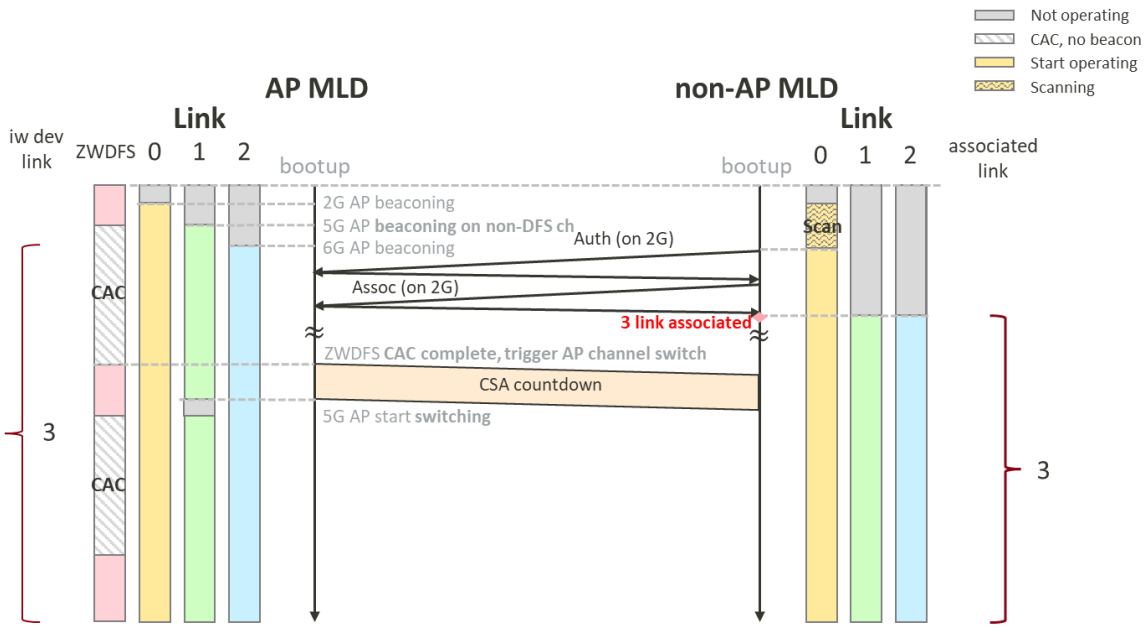
Also, it is worth noting that during the CAC period of 5G link, the link number of the AP MLD shows in “iw dev” command will be 3 instead of 2.

Note that the reconnection behavior here is a **temporary workaround**, as link reconfiguration to the ML setup is not yet supported (Wi-Fi 7 Cert R2). Once it is supported, the STA will trigger link reconfiguration by sending a link reconfiguration request frame.



- **With background radar**

With the help of background radar, the 5G link will operate on a randomly selected non-DFS channel instead of the DFS channel set by the user. And the work of performing the CAC of the user-configured DFS channel will be offloaded to the background radar (ZWDFS). Thus, the STA MLD can connect to **all three links** during the auth/assoc process. After the CAC is completed by the background radar, the 5G link will be triggered to switch to the AVAILABLE DFS channel, and the background radar will continue to perform the CAC on the other randomly selected DFS channel, as shown in the figure below.

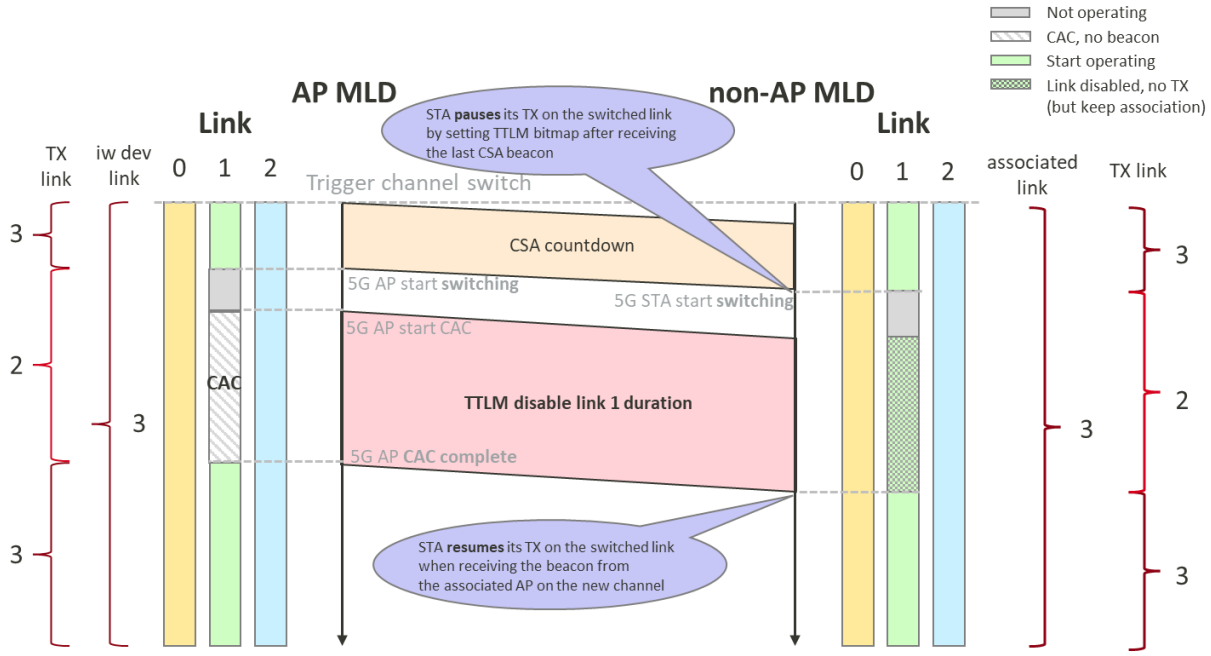


30.12.2 Radar Detection (CAC) after Channel Switch

The channel switch here could be a manually triggered channel switch or the channel switch right after radar is detected in the DFS channel. Note that if the target switching channel is a non-DFS channel, then no further action will be taken, as the AP will switch to it directly. So, the following examples assume that the target switching channel is a DFS channel.

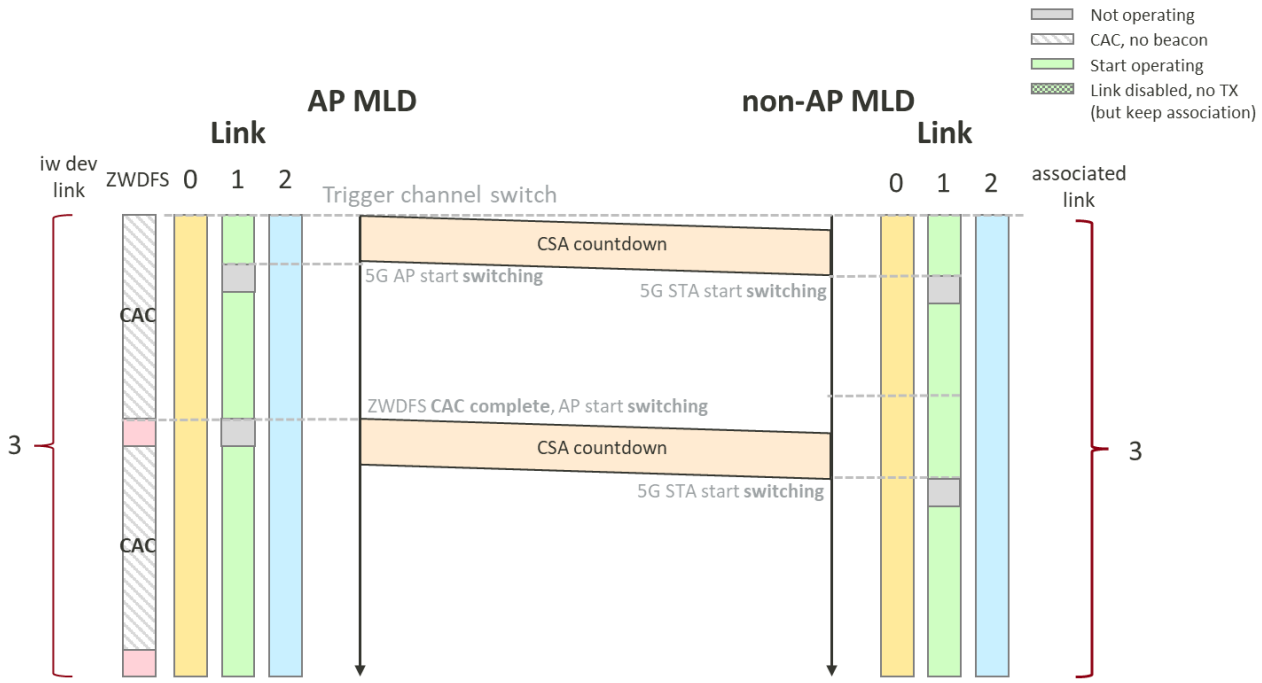
- **Without background radar (MLO Beta Release)**

When switching to the DFS channel, the AP should perform CAC of the target DFS channel after the CSA countdown. Due to DFS regulatory restrictions, both AP and STA shall not transmit any data. Therefore, the STA MLD will pause its TX on the switched link by setting TTLM bitmap after receiving the last CSA beacon. **Until 5G AP completes CAC, the 5G link will remain associated but disabled (no TX)**, as shown in the figure below (TX link number is 2 during the CAC period). Upon CAC completion, the STA MLD will automatically **resume the TX on the switched link when it receives a beacon** from the associated AP on target DFS channel.



- **With background radar**

With the help of background radar, the 5G link will switch to a randomly selected non-DFS channel instead of the DFS channel set by the user. And the work of performing the CAC of the target DFS channel will be offloaded to the background radar (ZWDFS). Thus, the impact on the traffic could be fully minimized, with only a slight disturbance in performance during the progress of the actual channel switch, as depicted by the gray area in the figure below. After the CAC is completed by the background radar, the 5G link will be triggered to switch to the target DFS channel, and the background radar will continue to perform the CAC on the other randomly selected DFS channel.



30.13 EPCS Priority Access

In this section, a brief introduction to EPCS (Emergency Preparedness Communication Services) Priority Access is presented, followed by how to configure and debug EPCS via ***hostapd_cli*** commands.

30.13.1 Introduction

EPCS Priority Access provides prioritized access to resources on IEEE 802.11 networks for authorized devices to increase their probability of successful communication during periods of network congestion. The authorized devices normally represent a small fraction of the overall devices associated with a BSS. The prioritization is carried out by using EDCA parameters that result in higher precedence.

An AP or STA MLD may request to enable EPCS by sending an EPCS Priority Access Enable Request, and the peer MLD may reply with an EPCS Priority Access Enable Response to enable EPCS. Note that EPCS will be enabled on all links between AP and STA MLDs. After EPCS is enabled, the AP MLD may update the EPCS EDCA parameters by sending an EPCS Priority Access Enable Response in unsolicited mode. Lastly, either MLD may send an EPCS Priority Access Teardown to disable existing EPCS service.

30.13.2 Configuration

** EPCS Priority Access for STA MLD is not yet supported by MT76. The following configuration is for AP MLD only.*

Authorize a STA MLD to use EPCS Priority Access.

```
hostapd_cli -i <interface> epcs authorize <station_mac_address>
```

- After authorization, the following EPCS EDCA parameters by default will be set for each link of the STA MLD.

AC	AIFSN	ECWmin	ECWmax	TXOP Limit
0 (BE)	3	4	9	0
1 (BK)	7	4	9	0
2 (VI)	2	3	4	188
3 (VO)	2	2	3	102

Set EPCS EDCA parameters link-wisely for an authorized STA MLD.

- If the STA MLD hasn't been authorized before, it will automatically be authorized.
- The EDCA parameters are set locally, and are not used by both AP and STA MLD yet. To apply the EDCA parameters, use ***send_update*** command described below.
- Format of EDCA parameters
  - **BE**_AIFSN, **BE**_ECWmin, **BE**_ECWmax, **BE**_TXOP,
  - **BK**_AIFSN, **BK**_ECWmin, **BK**_ECWmax, **BK**_TXOP,
  - **VI**_AIFSN, **VI**_ECWmin, **VI**_ECWmax, **VI**_TXOP,
  - **VO**_AIFSN, **VO**_ECWmin, **VO**_ECWmax, **VO**_TXOP

```
hostapd_cli -i <interface> epcs authorize <station_mac_address> link=<link_id> params=<edca_parameters>

# E.g. hostapd_cli -i ap-mld-1 epcs authorize 00:0c:43:2a:ef:25 link=0 params=1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16 link=1 params=16,15,14,13,12,11,10,9,8,7,6,5,4,3,2,1
```

Deauthorize a STA MLD to use EPCS Priority Access.

```
hostapd_cli -i <interface> epcs deauthorize <station_mac_address>
```

Send EPCS Priority Access Enable Request frame to an authorized STA MLD.

```
hostapd_cli -i <interface> epcs send_request <station_mac_address>
```

Send EPCS Priority Access Enable Response frame in unsolicited mode to an authorized STA MLD to update EDCA parameters.

- The EPCS EDCA parameters previously set by ***authorize*** command is used for update.

```
hostapd_cli -i <interface> epcs send_update <station_mac_address>
```

Send EPCS Priority Access Teardown frame to an authorized STA MLD.

```
hostapd_cli -i <interface> epcs send_teardown <station_mac_address>
```

30.13.3 Debug

Show EPCS Priority Access information of all authorized STA MLDs.

```
hostapd_cli -i <interface> epcs show
logread | grep “EPCS authorized-STA list” -A <lines>
# Use grep -A option to adjust number of lines of EPC information to show
# E.g. logread | grep “EPCS authorized-STA list” -A 50
```

- Sample output:

```
EPCS authorized-STA list:
        STA 06:0c:43:4b:ec:d1
                State: torn down
                Dialog token: 0
                EDCA parameters:
                        Link 0:
                                WMM index: 1
                                AC      AIFSN   ECWmin  ECWmax  TXOP
                                0       1       2       3       4
                                1       5       6       7       8
                                2       9       10      11      12
                                3       13      14      15      16
                        Link 1:
                                WMM index: 2
                                AC      AIFSN   ECWmin  ECWmax  TXOP
                                0       11      12      13      14
                                1       15      16      17      18
                                2       19      10      9       8
                                3       7       6       5       4
                        Link 2:
                                WMM index: 0
                                AC      AIFSN   ECWmin  ECWmax  TXOP
                                0       3       4       9       0
                                1       7       4       9       0
                                2       2       3       4       188
                                3       2       2       3       102
        STA 12:34:56:78:90:12
                Not connected to ap-mld-1
```

31 QoS Mapping Configuration

QoS map is used to translate IP DSCP (Differentiated Services Code Point) into IEEE 802.1D UP (User Priority), which is also known as IEEE 802.11e TID (Traffic Identifier). TID is then mapped to IEEE 802.11e AC (Access Category) for QoS in wireless LANs.

By default, hostapd MT76 uses the following QoS maps, ordered by priority:

1. User-defined map: defined in hostapd configuration files.
2. Recommended map: defined in section 4 of RFC8325.
   - Only used after wifi: cfg80211: Update the default DSCP-to-UP mapping was merged into Backports v6.8.
3. Default map: the three most significant bits of DSCP, defined in section 2.3 of RFC8325.

Hence, the default QoS map in Wi-Fi 6/7 MT76 is as follows:

DSCP	UP/TID	AC
0	0	BE
1	1	BK
2 to 16	0	BE
17	2	BK
18 to 22	3	BE
23	2	BK
24 to 39	4	VI
40 to 43	5	VI
44 to 46	6	VO
47	5	VI
48 to 63	7	VO

Set up a new user-defined QoS map

```
# Command: hostapd_cli -i <interface> set_qos_map_set <QoS map>
# <QoS map>: from left to right, two consecutive numbers separated by a comma are considered a combination. And all combinations are classified into the following two categories:
# (1) The last 8 combinations specify the DSCP range for each UP in order. E.g. DSCP 2~16 are mapped to UP 0, and DSCP 48~56 are mapped to UP 7. If both values in a combination are 255, the corresponding UP is not used in this map.
# (2) The remaining leading combinations specify the DSCP-to-UP exceptions, which overwrite the DSCP ranges previously described. E.g. DSCP 0 is mapped to UP 0. There can be up to 21 exceptions.
hostapd_cli -i phy0-ap0 set_qos_map_set 0,0,2,16,1,1,255,255,18,22,24,38,40,40,44,46,48,56
```

32 Vendor Specific Command

32.1 Single SKU Enable/Disable by Runtime

```
# usage 1 – hostapd configuration file initialization (only for WiFi7)
# vi hostapd-phyX.conf
# Add “sku_idx=[val]” without quotation, the val shall be greater than 0
sku_idx=0
```

```
# usage 2 – through debugfs (both for WiFi6/7)
# echo 1 > /sys/kernel/debug/ieee80211/phy0/mt76/sku_disable
# wifi
```

32.2 MU Enable/Disable – Hostapd Control

```
# Default: mu_onoff = 15 (bitmap- UL MU-MIMO: 1, DL MU-MIMO: 1, UL OFDMA: 0, DL
OFDMA: 1)

# usage 1 – runtime configuration
# hostapd_cli -i wlanX set_mu onoff [val]
(val = bitmap- UL MU-MIMO(bit3), DL MU-MIMO(bit2), UL OFDMA(bit1), DL OFDMA(bit0)
#  hostapd_cli -i wlanX get_mu onoff

# examples:
# hostapd_cli -i wlan1 set_mu onoff 15 (all enabled)
# hostapd_cli -i wlan1 set_mu onoff 0 (all disabled)
# hostapd_cli -i wlan1 set_mu onoff 12 (MU-MIMO UL & DL enabled)
# hostapd_cli -i wlan1 get_mu onoff
[hostapd_cli] = UL MU-MIMO: 1, DL MU-MIMO: 1, UL OFDMA: 0, DL OFDMA: 0

# usage 2 – hostapd configuration file initialization
# vi hostapd-phyX.conf
# Add “mu_onoff=[val]” without quotation
(val = bitmap- UL MU-MIMO(bit3), DL MU-MIMO(bit2), UL OFDMA(bit1), DL OFDMA(bit0)
```

Please add -i <ifname> to set the mu status at this band. (per-interface control)

32.3 Maximum Stations/BSS Capability Dump

```
# usage
# mt76-vendor <ifname> dump phy_capa

# take mt7915 as an example
# mt76-vendor wlan1 dump phy_capa
[vendor] Max Supported BSS=16.  Max Supported STA=268
```

32.4 EDCCA Feature Control

```
# usage
# hostapd_cli -i wlanX raw SET_EDCCA [enable|threshold|compensation] [value]
# hostapd_cli -i wlanX raw GET_EDCCA [enable|threshold|compensation]
```

The first command is used to configure the EDCCA mode, threshold, or compensation in run time. The second command is used to get EDCCA mode, threshold, or compensation at run time.

If you want to set the EDCCA mode, the value can be 0 or 1. 0 means to disable EDCCA, and 1 means auto mode. Below This is an example to disable the EDCCA feature on the wlan1 interface:

```
# usage
# hostapd_cli -i wlan1 raw SET_EDCCA enable 0
```

Here is an example of how to retrieve the current mode of the EDCCA feature on the wlan1 interface.

```
# usage
# hostapd_cli -i wlanX raw GET_EDCCA enable
```

If you want to set the EDCCA threshold, the value shall be in the format below:

```
# usage
# hostapd_cli -i wlanX raw SET_EDCCA threshold BandwithID:Threshold
```

- BandwidthId
  - 0: BW20, 1:BW40, 2:BW 80, 3:BW 160
- The range of the threshold is from -126 to 0 dBm.
- 3:BW 160 only for Wi-Fi 7

The following example retrieves all current EDCCA thresholds on the wlan1 interface.

```
# usage
# hostapd_cli -i wlan1 raw GET_EDCCA threshold
```

Here is an example of setting the BW40 EDCCA threshold to -60 dBm.

```
# usage
# hostapd_cli -i wlanX raw SET_EDCCA threshold 1:-60
```

Below is an example of setting the compensation value -2 dBm on the wlan1 interface:

```
# usage
# hostapd_cli -i wlan1 raw SET_EDCCA compensation -2
```

The compensation value range extends from -126 dBm to 126 dBm.

The following example demonstrates how to retrieve the current compensation value of the EDCCA feature on the wlan1 interface.

```
# usage
# hostapd_cli -i wlan1 raw GET_EDCCA compensation
```

The command parameter “compensation” is only needed on Wi-Fi 6 BSP

The following presents a scenario for configuring the EDCCA feature.
Enabling a virtual interface activates the EDCCA feature by default. The following command can be used:
to check the EDCCA threshold value

```
# hostapd_cli -i wlan1 raw GET_EDCCA threshold
Threshold BW20: 0xbf, BW40: 0xc2, BW80: 0xc5, BW160: 0xc8
```

If you want to change the BW20 EDCCA threshold to -66 dBm, you can use the following command to modify the threshold.

```
# hostapd_cli -i wlan1 raw SET_EDCCA threshold 0:-66
OK
```

Use the following command to check the result. The output will be as shown below.

```
# hostapd_cli -i wlan1 raw GET_EDCCA threshold
Threshold BW20: 0xBE, BW40: 0xC2, BW80: 0xC5, BW160: 0xC8
```

If you want to configure EDCCA threshold as -66 dBm, -62 dBm, -59 dBm and -57 dbmfor BW20, BW40, BW 80 and BW 160
When the virtual interface is up, the following line needs to be added to the hostapd configuration.

```
# vim /var/run/hostapd-phyX.conf
edcca_threshold=-66 -62 -59 -57
```

How to use this new hostapd configuration to start hostapd, please refer to 3.3.1 Bypass netifd Conversion.

32.5 AMPUD and AMSDU Enable/Disable – Hostapd Control

```
# Default: AMSDU = 1, AMPDU=1

# usage – hostapd configuration file initialization
# vi hostapd-phyX.conf
# Add “ampdu=[val]” without quotation
# Add “amsdu=[val]” without quotation
(val = 1 : enable, val = 0 : disable)
(Note: AMSDU setting is only effective when AMPDU is enabled)

# usage – runtime get
# hostapd_cli -i wlanX get_aggr

# examples:
# hostapd_cli -i wlan1 get_aggr
[hostapd_cli] AMPDU:1, AMSDU: 1
```

32.6 Three Wire Control (PTA)

32.6.1 Setting

First, if you are using MTK reference board (RFB), please switch the mode of MT7986's external source in GPIO by entering the following commands.

```
# MT7986:
regs w 0x1001f350 0x22221111
regs w 0x1001f360 0x11111122
# MT7996:
mwctl phy phy0 mac 70005310=11111111
```

If a customer board is being used, the HW engineer should be contacted to ensure that the GPIO is set correctly.

To enable PTA, please add "three wire enable=<val>" to the hostapd configuration file, as shown in the figure below. Please refer to Hostapd Configuration for more information.

```
snoop_iface=br-lan
qos_map_set=0,0,2,16,1,1,1,255,255,18,22,24,38,40,40,44,46,48,56
config_id=dd14ec6a7c90e36831b63d55e3b34a8b
bssid=00:0c:43:26:60:10
three_wire_enable=2
```

The modes supported by PTA are listed in the table below. The default value of three_wire_enable is 0 if three_wire_enable is not configured in the hostapd config file.

Value	Ext0	Ext1
0	Disabled	Disabled
1	Enabled	Disabled
2 (MT7986 only)	Disabled	Enabled
3 (MT7986 only)	Enabled	Enabled

32.6.2 Usage

Note that firmware supports PTA runtime disablement. As a result, you can switch the mode of PTA without reloading the wifi by following the steps in Hostapd Configuration.

32.7 Get BSS Color and Available BSS Color Bitmap

Retrieve the he_bss_color of a BSS

```
# usage
# hostapd_cli -i <ifname> get_bss_color

# example
# hostapd_cli -i phy1-ap0 get_bss_color
BSS Color=30
```

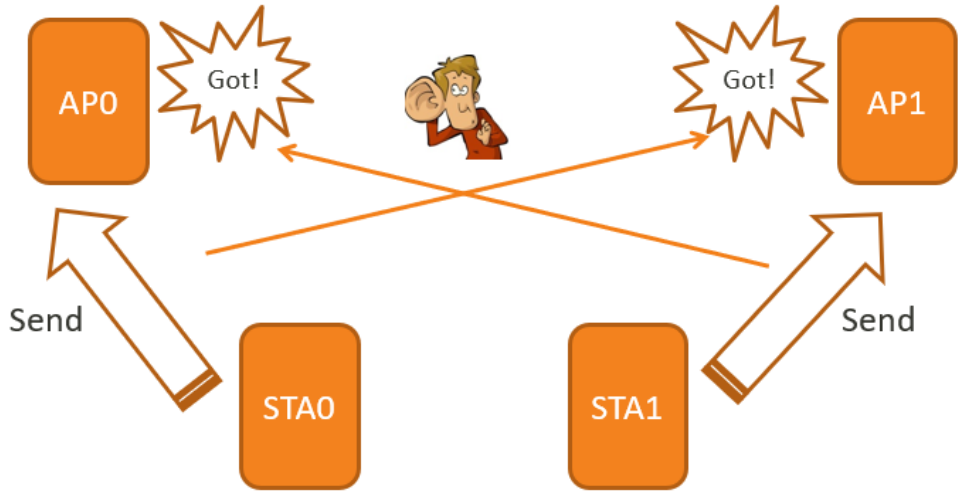
Get available color bitmap a BSS detects. The color bitmap takes results from mac80211 and WNM event report. Our own BSS color is not reflected on the bitmap. [0/1]: occupied/available

```
# usage
# hostapd_cli -i <ifname> get_color_bmp

# example
# hostapd_cli -i phy1-ap0 get_color_bmp
available color bitmap=0xff7ffeffffeffff
 0: 1 1 1 1 1 1 1 1
 8: 1 1 1 1 1 1 1 1
16: 1 1 1 1 0 1 1 1
24: 1 1 1 1 1 1 1 1
32: 1 1 1 1 1 1 1 1
40: 0 1 1 1 1 1 1 1
48: 1 1 1 1 1 1 1 0
56: 1 1 1 1 1 1 1 1
```

32.8 Add/ Remove and Dump Air Monitor Entries

The air monitor is just like a sniffer function that is used to eavesdrop on the packets of unassociated stations with specified MACs. The station MAC can be inserted into the list using the commands below. This feature can get information like RSSI and Last Seen.

For example, Station 1 is connected to AP1, and if AP0 wants to listen to packets between AP1 and Station 1, AP0 can use the air monitor.



MT7986

Add or remove an air monitor entry. A maximum of 16 entries can be added to a specific band.

```
# usage
# Add Entry
# mt76-vendor <interface> set amnt <index> <mac addr>
root@OpenWrt:/# /usr/sbin/mt76-vendor wlan1 set amnt 0 90:8C:43:DD:0D:72
root@OpenWrt:/# /usr/sbin/mt76-vendor wlan1 set amnt 1 11:22:33:44:55:66
```

Air Monitor Entries can be dumped for a station at a specific index or for all stations that are included in the air monitor list.

```
# usage
```

```
# Dump RSSI for single station in air monitor list
# mt76-vendor <interface> dump amnt <monitor index>
# Dump RSSI for all stations in the list
# mt76-vendor <interface> dump amnt 0xFF

# Example
# mt76-vendor wlanX dump amnt <index>
root@OpenWrt:/# /usr/sbin/mt76-vendor wlan1 dump amnt 0
[vendor] amnt_idx: 0, addr=90:8c:43:dd:d:72, rssi=-48/-70/17/17, last_seen=4572
root@OpenWrt:/#
root@OpenWrt:/# /usr/sbin/mt76-vendor wlan1 dump amnt 1
[vendor] amnt_idx: 1, addr=11:22:33:44:55:66, rssi=0/0/0/0, last_seen=139880
root@OpenWrt:/#
root@OpenWrt:/# /usr/sbin/mt76-vendor wlan1 dump amnt 0xff
[vendor] amnt_idx: 0, addr=90:8c:43:dd:d:72, rssi=-51/-73/17/17, last_seen=7316
[vendor] amnt_idx: 1, addr=11:22:33:44:55:66, rssi=0/0/0/0, last_seen=145404
```

MT7996

Add or remove an air monitor entry. A maximum of 16 entries can be added to a specific band.

```
# usage
# Add Entry
# hostapd_cli -i <interface> set_amnt <monitor index> <station mac address>
# Delete Entry
# hostapd_cli -i <interface> set_amnt <monitor index> 00:00:00:00:00:00

# Example
# hostapd_cli -i phy0-ap0 set_amnt 1 56:BF:C5:4E:53:A6
OK
# hostapd_cli -i phy0-ap0 set_amnt 1 00:00:00:00:00:00
OK
```

Air monitor entries can be dumped for a specific station based on its index or for all stations that are included in the air monitor list.

```
# usage
# Dump RSSI for single station in air monitor list
# hostapd_cli -i <interface> dump_amnt <monitor index>
# Dump RSSI for all stations in the list
# hostapd_cli -i <interface> dump_amnt 0xFF

# Example
# hostapd_cli -i phy0-ap0 dump_amnt 1
[hostapd_cli] amnt_idx: 1, addr=56:BF:C5:4E:53:A6, rssi=-45/-42/-56/-40, last_seen=35
# hostapd_cli -i phy1-ap0 dump_amnt 0xff
[hostapd_cli] amnt_idx: 0, addr=2:c:43:46:60:30, rssi=-33/-36/-38/-51, last_seen=76
[hostapd_cli] amnt_idx: 1, addr=2:c:43:56:60:30, rssi=-33/-35/-38/-51, last_seen=72
[hostapd_cli] amnt_idx: 2, addr=2:c:43:86:60:30, rssi=-33/-35/-38/-51, last_seen=60
[hostapd_cli] amnt_idx: 3, addr=2:c:43:76:60:30, rssi=-33/-35/-38/-51, last_seen=64
[hostapd_cli] amnt_idx: 4, addr=2:c:43:66:60:30, rssi=-33/-36/-38/-51, last_seen=68
[hostapd_cli] amnt_idx: 5, addr=0:a:d3:a:dd:55, rssi=-38/-38/-43/-63, last_seen=68
[hostapd_cli] amnt_idx: 6, addr=2:a:d3:1a:dd:55, rssi=-39/-38/-43/-63, last_seen=64
[hostapd_cli] amnt_idx: 7, addr=2:a:d3:2a:dd:55, rssi=-39/-38/-43/-63, last_seen=60
[hostapd_cli] amnt_idx: 8, addr=2:a:d3:3a:dd:55, rssi=-38/-38/-43/-63, last_seen=56
[hostapd_cli] amnt_idx: 9, addr=f2:1e:59:8c:d7:65, rssi=-48/-50/-59/-76, last_seen=7960
[hostapd_cli] amnt_idx: 10, addr=58:ef:68:bf:6c:4a, rssi=-54/-65/-57/-82, last_seen=92
```

```
[hostapd_cli] amnt_idx: 11, addr=0:c:43:26:11:90, rssi=-54/-52/-50/-73, last_seen=4
[hostapd_cli] amnt_idx: 12, addr=2:c:43:36:50:50, rssi=-66/-75/-64/-80, last_seen=68
[hostapd_cli] amnt_idx: 13, addr=58:ef:68:bf:6c:4a, rssi=-66/-74/-67/-83, last_seen=446924
[hostapd_cli] amnt_idx: 14, addr=2:c:43:26:50:50, rssi=-65/-73/-63/-80, last_seen=172
```

32.9 Channel State Information (CSI)

Please refer to the following documents.

Connac2 (Wi-Fi 6)	Connac2_CSI_Spec_Requirement_release_v1.1
Connac3 (Wi-Fi 7)	Connac3_CSI_Spec_Requirement_release_v1.2

33 Conduct Testing with Equipment

Users need to configure specific settings for the APUT when conducting tests using virtual multiple station equipment, such as **C50**, **Octoscope**, and **Veriwave**.

Configuration	Command	Note
ARP issue	Perform a background ping to the stations before starting the traffic *for i in $(seq 150 189); do ping -i 5 192.168.1.$i > /dev/null & done*	Solve ARP issue
Binding issue	• *iptables -D FORWARD -p tcp -m conntrack --ctstate RELATED,ESTABLISHED -j FLOWOFFLOAD --hw* • *iptables -I FORWARD -p tcp -j FLOWOFFLOAD --hw*	Modify the iptables rules to trigger binding forcibly
addba_resp_timer timeout issue	*echo 20 > /sys/module/mac80211/parameters/addba_resp_wait_count*	BA establishment with **C50**-station fail needs to extend the BA response timer in MAC80211 from 2 sec to 20 sec
Connection issue (*VeriWave)	*dmesg -n1*	Reduce the log level to prevent excessive logging, which may cause the test device to fail to connect successfully
BA buffer size issue	*mwctl <ifname> set ap_wireless add_ba_req_bufsize=64*	**C50** cannot handle a BA buffer size of 256, which will cause C50 to repeatedly disconnect and rebuild BA

33.1 Veriwave on 2.4 GHz

Due to the following reasons, MT76 does not support 256-QAM modulation (VHT MCS8~9). Please use the HT mode as testing option.

- MT76 strictly follows IEEE 802.11 standards, and HT 256-QAM is not included in the standards. Rather, HT 256-QAM is a proprietary feature developed during the transition from IEEE 802.11n to IEEE 802.11ac Wi-Fi generation.
- Code commits related to HT 256-QAM and aiming for upstreaming are rejected by the OpenWrt community. That is, the OpenWrt operating system does not support HT 256-QAM, either.

When testing 2G throughput in HT mode with frame sizes less than 1,024 bytes, do not enable TX A-MSDU in Veriwave. Otherwise, Veriwave stations will generate conceivably erroneous A-MSDUs for DUT.

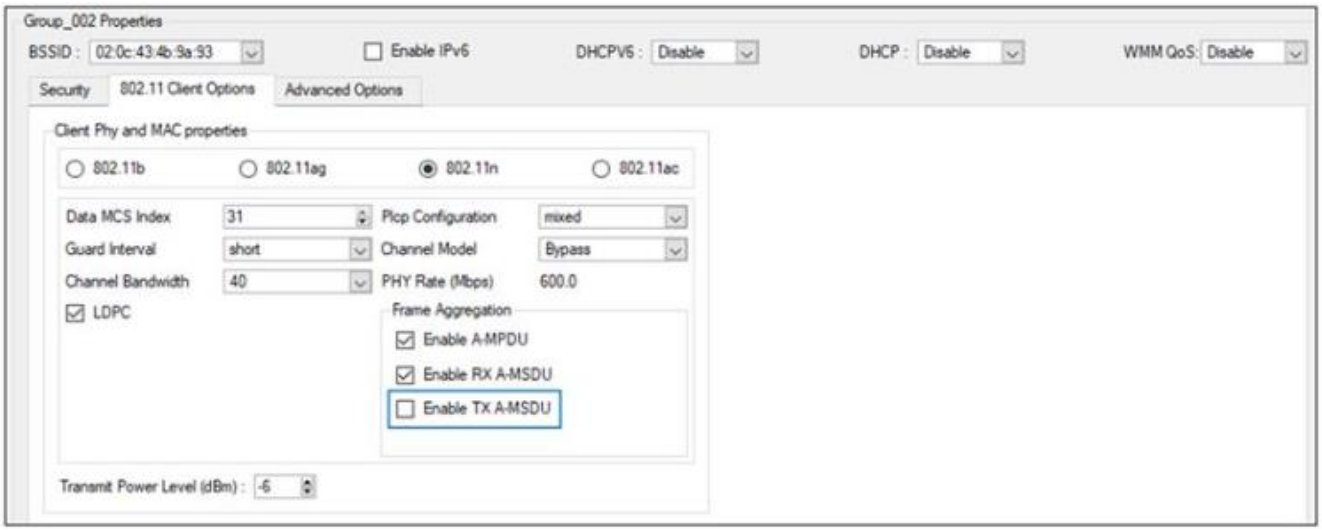


Exhibit 1 Terms and Conditions

Your access to and use of this document and the information contained herein (collectively this “Document”) is subject to your (including the corporation or other legal entity you represent, collectively “You”) acceptance of the terms and conditions set forth below (“T&C”). By using, accessing or downloading this Document, You are accepting the T&C and agree to be bound by the T&C. If You don’t agree to the T&C, You may not use this Document and shall immediately destroy any copy thereof.

This Document contains information that is confidential and proprietary to MediaTek Inc. and/or its affiliates (collectively “MediaTek”) or its licensors and is provided solely for Your internal use with MediaTek’s chipset(s) described in this Document and shall not be used for any other purposes (including but not limited to identifying or providing evidence to support any potential patent infringement claim against MediaTek or any of MediaTek’s suppliers and/or direct or indirect customers). Unauthorized use or disclosure of the information contained herein is prohibited. You agree to indemnify MediaTek for any loss or damages suffered by MediaTek for Your unauthorized use or disclosure of this Document, in whole or in part.

MediaTek and its licensors retain titles and all ownership rights in and to this Document and no license (express or implied, by estoppels or otherwise) to any intellectual propriety rights is granted hereunder. This Document is subject to change without further notification. MediaTek does not assume any responsibility arising out of or in connection with any use of, or reliance on, this Document, and specifically disclaims any and all liability, including, without limitation, consequential or incidental damages.

THIS DOCUMENT AND ANY OTHER MATERIALS OR TECHNICAL SUPPORT PROVIDED BY MEDIATEK IN CONNECTION WITH THIS DOCUMENT, IF ANY, ARE PROVIDED “AS IS” WITHOUT WARRANTY OF ANY KIND, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE. MEDIATEK SPECIFICALLY DISCLAIMS ALL WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, FITNESS FOR A PARTICULAR PURPOSE, COMPLETENESS OR ACCURACY AND ALL WARRANTIES ARISING OUT OF TRADE USAGE OR OUT OF A COURSE OF DEALING OR COURSE OF PERFORMANCE. MEDIATEK SHALL NOT BE RESPONSIBLE FOR ANY MEDIATEK DELIVERABLES MADE TO MEET YOUR SPECIFICATIONS OR TO CONFORM TO A PARTICULAR STANDARD OR OPEN FORUM.

Without limiting the generality of the foregoing, MediaTek makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does MediaTek assume any liability arising out of the application or use of any product, circuit or software. You agree that You are solely responsible for the designing, validating and testing Your product incorporating MediaTek’s product and ensure such product meets applicable standards and any safety, security or other requirements.

The above T&C and all acts in connection with the T&C or this Document shall be governed, construed and interpreted in accordance with the laws of Taiwan, without giving effect to the principles of conflicts of law.