# MD-6000 Detailed Design Document

# Ath10K 802.11ac Structured Mesh™

(Addendum to 2014 Base Draft)

TABLE OF CONTENTS



TABLE OF FIGURES



# 1 Introduction

MeshDynamics delivers third-generation wireless mesh networking solutions for high-performance outdoor data, voice, and video networking. Based on sophisticated dynamic channel-agile networking algorithms, MeshDynamics MD4000 family of Structured Mesh™ wireless nodes deliver very low-latency and low-jitter performance, even over multi-hop topologies where many earlier generation wireless mesh networking products fail.

Software development began in 2001 with United States Defence contracts. Prototypes were rigorously tested by the United States military through 2003-2005. Production shipments began in 2005, providing scalable wireless mesh networking solutions for Defence, Homeland Security, surface and underground mining.

Today, MeshDynamics products are used worldwide in mining and industrial, video surveillance, defence, and outdoor sport event.

MeshDynamics Structured Mesh™ multi-radio mesh network MESH Algorithm provides features which makes it different from other mesh networking implementations. These features include:-
1. Dynamic RF channel management
2. Dynamic scanning for mobility
3. Structured Mesh™ heartbeat transmission and processing
4. Mesh routing table management
5. Self-forming/Self-healing mesh networking

It also includes security components like Wi-Fi-Protected Access (WPA) version 1 and 2, IEEE 802.11i, FIPS 140-2, IEEE 802.1x which makes the MeshDynamics Structured Mesh™ a better choice for users across the Defence and Homeland Security Agencies in US, UK and Canada to provide mission critical video surveillance and perimeter security.

Currently the mesh algorithm implementation is tightly coupled with the underlying Atheros drivers and is based on closed sourced HAL implementation. This architecture is however robust, but to provide more flexibility to the customers, an approach is needed to make the mesh algorithm independent of the HAL and the device drivers. With this approach, more options open to customers to easily choose between the underlying hardware and gain benefits of open source approach.

## 1.1 Objective

# 2 Scope

# 3 Acronyms

| | |
|---|---|
| IMCP | Infrastructure Mesh Control Protocol |
| AP | Access Point |
| BSS | Basic Service Set |
| CTS | Clear To Send |
| MAC | Media Access Control |
| MLME | Media Access Control Sub layer Management Entity |
| POE | Power Over Ethernet |
| RADIUS | Remote Authentication Dial In User Service |
| RTS | Request to Send |
| Rx | Reception |
| Tx | Transmission |
| SSID | Service Set Identifier |
| STA | Station |

# 4 References

- "MD4000 Node Deployment and Trouble Shooting Guide" - MD4000_HWMANUAL.pdf
- "Architecture Overview" - MD-OEM-HARDWARE-INTEGRATION-VOL1-4.pdf
- http://www.campsmur.cat/files/mac80211_intro.pdf
- http://wireless.kernel.org/en/developers/Documentation/mac80211

# 5 Product Models

**2.4GHz Backhaul Products**

1. MD4220-IIxx: 2-Radio module 2.4GHz uplink and downlink Backhaul (BH).
2. MD4320-IIIx: 3-Radio module 2.4GHz sectored BH slots 0,1 and 2.4GHz AP radio in slot 2.
3. MD4325-IIxI: 3-Radio module 2.4GHz BH, Downlink also acts as AP. A 2.4GHz Mobility Scanner in slot3.
4. MD4424-IIII: 4-Radio module 2.4GHz service radios (AP) in all slots. Use with 4 panel antennas.

**5GHz Backhaul Products**

1. MD4250-AAxx: 2-Radio module 5GHz uplink and downlink Backhaul (BH).
2. MD4350-AAIx: 3-Radio module 5GHz BH and 2.4GHz AP radio in slot 2. AP modes may be b, g, or b & g.
3. MD4452-AAIA: 4-Radio module 5GHz BH and 2.4GHz AP radio. Second sectored 5.8GHz downlink in slot 3.
4. MD4454-AAAA: 4-Radio module 5GHz with radios as downlinks. Intended as root with four 90 deg panels.
5. MD4458-AAII: 4-Radio module 5GHz BH and two 2.4GHz AP radios in slots 2, 3 for sectored service.
6. MD4455-AAIA: 4-Radio module 5GHz BH and 2.4GHz AP radio in slot 2. 5GHz mobility scanner in slot 3.

# 6 Hardware Boards

# 7 MAC 80211n and Mac 80211ac Frame formats

# 8 Software Architecture

The MeshDynamics Mac80211 based Mesh Networking architecture defines a design approach where the proprietary Mesh Networking algorithm provided by MeshDynamics will be fully integrated with the Linux based MAC80211 architecture.

It provides the complete abstraction of the proprietary Mesh Networking algorithm (meshap) from the underlying device drivers and therefore the dependency of meshap on any of the underlying device drivers will be removed.

If the underlying device driver changes, the meshap continues to provide services without any impact and no modification is required in the code. The goal is to support all the functionalities provided by the existing MeshDynamics algorithm and there shall be no interface level impact on the existing meshap algorithm.

The block diagram of the Mesh Dynamics' Mac 80211 based Mesh Networking architecture is

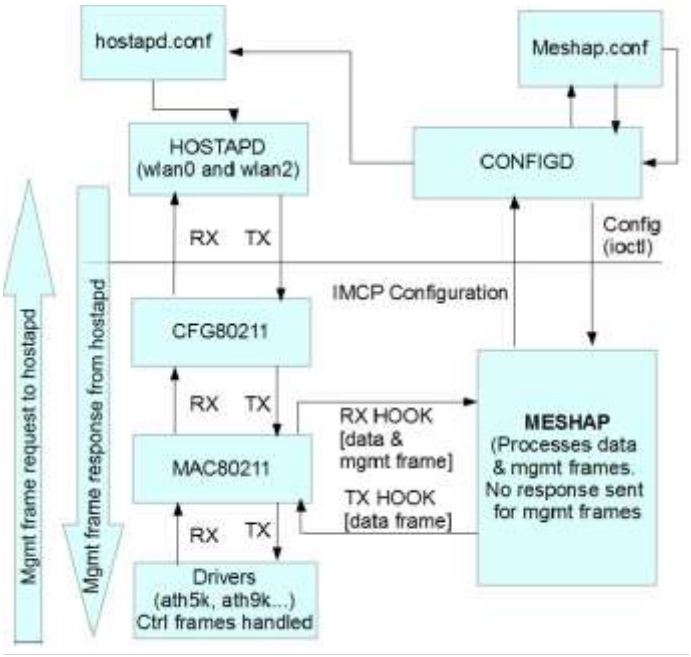



Figure 1 High level architecture

The Figure 1 describes the existing Linux based Mac80211 architecture which is integrated with the Meshap. The diagram shows the packet flow between the blocks via MAC80211 to the Meshap.

2

3

## 8.1 Device Drivers

It is the underlying WLAN device drivers like atheros 5k, atheros 9k in the linux kernel. The packets received by the atheros devices are processed by the device driver layer and handed over to MAC80211 block. These are the standard drivers of linux and are not impacted with the integration of meshap and MAC80211..

## 8.2 MAC80211

The mac80211 is a framework which driver developers can use to write drivers for SoftMAC wireless devices. mac80211 implements the cfg80211 callbacks for SoftMAC devices. Device drivers call the routines of MAC80211 to hand the received packet. The Mac80211 block processes this packet and at one point calls the Rx hook which will redirect the data and management frames to the meshap block for processing. The response from meshap is passed back to mac80211 by calling the Tx hook function. For management frames, one copy is sent to meshap and the other copy is sent to hostapd. The response for these management frames is only sent back from hostapd while meshap only processes the frames and updates its database. For more details on MAC80211 architecture, refer to section **Error! Reference source not found.**.

## 8.3 cfg80211

cfg80211 is the Linux 802.11 configuration API. The netlink 'nl80211' driver is used to configure a cfg80211 device and is used for communication between kernel and userspace. This layer is

responsible to configure MAC80211 based on callbacks from hostapd in userspace. There is no change in this block with integration of meshap with mac80211.

## 8.4 Hostapd

Hostapd is a user space daemon for access point and authentication servers. It configures the wireless interfaces (using netlink sockets) for MAC80211 based system. It implements IEEE 802.11 access point management, IEEE 802.1X/WPA/WPA2/EAP Authenticators, RADIUS client, EAP server, and RADIUS authentication server It depends on hostapd to handle authenticating clients, setting encryption keys, establishing key rotation policy, and other aspects of the wireless infrastructure. Hostapd is run on the interfaces configured as AP (wlan2) and on the downlink interface (wlan0). Hostapd handles the management frames and sends the response for the management frames it receives.

## 8.5 Configd

It is a user space daemon used by Meshap to process configuration requests from Meshap in the user space. The configd daemon is used for the boot-time as well as the runtime configuration of meshap. When meshap boots up, configd daemon configures and start the meshap. On receiving a configuration change parameter configd initiates IOCTL command to configure that parameter to meshap and also applies this change to mac80211 via hostapd by modifying the hostapd.conf configuration file and issuing a SIGHUP to the hostapd daemon.

## 8.6 Meshap

The meshap block is the complete mesh networking core algorithm provided by Mesh Dynamics.

# 9 Configuration Parameters to be added for 802.11n and 802.11ac

This section contains configuration parameters specific to 802.11n standard functionalities.

## 9.1 List of configuration parameters to be considered

| Configuration Parameter | Description |
|---|---|
| **Common Configuration Parameters for 802.11n and ac** | |
| Operation Mode | Configures physical layer operation mode 802.11a/b/g/n/ac |
| Channel Bandwidth | Sets Channel Width to be used for Data Transmission 20/40/80/160 MHz |
| Guard Interval | Used to configure short or long guard interval in 802.11n/ac supported AP |
| Tx-STBC | Enables/disables Tx Space-Time Block Coding (STBC) feature supported in |

| | |
|---|---|
| | 802.11n and ac standards |
| Rx-STBC | Enables/disables Rx Space-Time Block Coding (STBC) feature supported in 802.11n and ac standards |
| Maximum Transmit Rate | Provides flexibility to configure maximum transmit rate of Downlink (AP mode) interface as per 802.11n/ac standard |
| Preamble Type | Used to configure preamble type |
| Tx Frame Aggregation | Enables/disables Tx Frame Aggregation support and type |
| **802.11n(HT) Related Configuration** | |
| Secondary Channel Offset | Configures Secondary Channel is above or below the primary channel in case of bandwidth is 40MHz |
| LDPC Capability | Enables/Disables LDPC Coding Capability |
| SM Power Save | Sets Spatial Multiplexing Power Save Mode type |
| Green Field Mode | This mode forces 802.11n supported AP device to work purely in 802.11n mode |
| Delayed BlockAck | Enables/Disables delayed Block Acknowledgement feature |
| BSS 40MHz DSSS/CCK Mode | Enables/Disables use of DSSS/CCK Mode in 20/40MHz in BSS |
| 40 MHz intolerant | Enables/Disables 40 MHz Transmission support |
| Max-AMPDU Length | Sets Maximum size of an AMPDU Frame length |
| Max-AMSDU Length | Sets Maximum size of an AMSDU Frame length |
| **802.11ac(VHT) Related Configuration** | |
| Max-MPDU Length | Sets Maximum size of a MPDU Frame length |
| SU Beamformer Support | Indicates support for Single User Beamformer |
| SU Beamformee Support | Indicates support for Single User Beamformee |

| | |
|---|---|
| CS - Number of BFA Supported | Beamformee's capability indicating the maximum number of beamformer antennas the beamformee can support when sending compressed beamforming feedback |
| Number of SD | Beamformer's capability indicating the maximum value of the NUM_STS parameter in the TXVECTOR of a VHT NDP |
| MU Beamformer | Indicates support for operation as Multi-User Beamformer |
| MU Beamformee | Indicates support for operation as Multi-User Beamformee |
| VHT TXOP PS | Indicates support for VHT TXOP Power Save Mode |
| HTC-VHT Capable | Indicates whether or not the STA supports receiving a VHT variant HT Control field |
| Maximum A-MPDU Length Exponent | Indicates the maximum length of A-MPDU pre-EOF padding that the STA can receive |
| VHT Link Adaptation Capable | Indicates whether or not the STA supports link adaptation using VHT variant HT Control field |
| Rx Antenna Pattern Consistency | Indicates the possibility of Rx antenna pattern change |
| Tx Antenna Pattern Consistency | Indicates the possibility of Tx antenna pattern change |
| Seg0 Central Frequency | Sets channel center frequency for 80 and 160 MHz VHT BSS and the frequency segment 0 channel center frequency for 80+80 MHz VHT BSS |
| Seg1 Central Frequency | Sets Channel center frequency for 80+80 MHz VHT BSS |

**Note:** Configuration Parameters marked in violet colour are already existing parameters in MD-4000 device.

4
5
6
7
8
9

## 9.1

## 9.2 Representation of above mentioned configuration parameters in hostap.conf

### 9.2.1 Operation Mode

This configuration represented by hw_mode and ieee80211n/ieee80211ac fields in hostapd.conf file. If hw_mode=g and ieee80211n=1, it indicates band type is 2.4GHz and HT mode is enabled.

(Eg: To set operation mode in 802.11n and 5GHz band, hw_mode=a and ieee80211n=1)

### 9.2.2 Channel Width and Secondary Channel Offset

The channel width and secondary channel offset configuration represented as part of ht_capab in hostapd.conf file. If bandwidth is configured as 40MHz then it is indicated as HT40 and secondary channel is below the primary channel then it is indicated by (-) symbol suffixed or if secondary channel is above the primary channel then it is indicated by (+) symbol suffixed.

(Eg: To configure 40MHz and Secondary channel is below primary channel then it would be represented as ht_capab=[HT40-])

### 9.2.3 Maximum Transmit Rate

No corresponding configuration field present in hostapd.conf file

### 9.2.4 LDPC Capability

This configuration represented as part of ht_capab field. To enable this feature include [LDPC] flag as part of ht_capab field otherwise it would be considered as LDPC unsupported.

(Eg: ht_capab=[LDPC])

### 9.2.5 SM Power Save

This configuration represented as part of ht_capab field. To set Static Spatial Multiplexing (SM) Power save mode include [SMPS-STATIC] flag in ht_capab field. To set Dynamic SMPS mode include [SMPS-DYNAMIC] flag in ht_capab field, neither flag set if SMPS is disabled.

(Eg: ht_capab=[SMPS-STATIC])

### 9.2.6 GF (Green Field) Mode

This configuration represented as part of ht_capab field. To enable GF mode include [GF] flag in ht_capab field. GF mode considered as disabled if this is not set.

(Eg: ht_capab=[GF])

### 9.2.7 Guard Interval

This configuration represented as part of ht_capab field. To enable short guard interval support in 20MHz bandwidth include [SHORT-GI-20] flag in ht_capab field and to disable don't include this flag. Similarly to enable short guard interval supports in 40MHz bandwidth include [SHORT-GI-40], to disable don't include this flag.

(Eg: ht_capab=[SHORT-GI-20])

### 9.2.8 Tx-STBC

This configuration represented as part of ht_capab field. To enable Transmission STBC support include [TX-STBC] flag in ht_capab field, to disable don't include this flag.

(Eg: ht_capab=[TX-STBC])

### 9.2.9 Rx-STBC

This configuration represented as part of ht_capab field. To enable Receiving STBC support include [RX-STBC] flag in ht_capab field, to disable don't include this flag.

(Eg: ht_capab=[RX-STBC])

### 9.2.10 Delayed BlockAck

This configuration represented as part of ht_capab in hostapd.conf file. To enable this feature include [DELAYED-BA] flag as part of ht_capab field value. To disable don't include this flag.

(Eg: ht_capab= [DELAYED-BA])

### 9.2.11 Preamble Type

This configuration represented in preamble field. To enable short preamble set preamble to 1.

(Eg: preamble=1)

### 9.2.12 BSS 40MHz DSSS/CCK Mode

This configuration represented as part of ht_capab field. To allow DSSS-CCK in 40MHz, include flag [DSSS_CCK-40] in ht_capab field. Don't include this flag to disable DSSS-CCK in 40MHz.

(Eg: ht_capab=[DSSS_CCK-40]

### 9.2.13 40 MHz intolerant

This configuration represented as part of ht_capab field. To enable 40MHz intolerant advertisement, include flag [40-INTOLERANT] in ht_capab field. Don't include this flag to disable advertisement of 40MHz intolerant.

(Eg: ht_capab=[40-INTOLERANT]

### 9.2.14 Tx Frame Aggregation

No field present in hostap.conf file to enable Frame aggregation in Transmission Path

### 9.2.15 Max-AMSDU Length

This configuration represented as part of ht_capab in hostapd.conf. To set maximum AMSDU length as 7935 octets update ht_capab field with MAX-AMSDU-7935.

(Eg: ht_capab=[ MAX-AMSDU-7935]

## 9.3 Representation of above mentioned configuration parameters in meshap.conf

In meshap.conf file, 802.11n HT related configurations shall be represented as below,

```
wlan0 {
      medium_type=w,
      medium_sub_type=an,
      usage_type=wm,
      channel=149,
      bonding=s,
      ht_capab {
            ldpc=enabled,
            ht_bandwidth=40+,
            smps=static,
            gi_20=short,
            gi_40=auto,
            tx_stbc=enabled,
            rx_stbc=enabled,
            delayed_ba=disabled,
            max_amsdu_len=1,
            dsss_cck_40=allow,
            40_intolerant=disabled,
            l-sig_txop=disabled,
      },
      frame_aggregation {
            tx_ampdu_enable=1,
            tx_ampdu_max_len=64KB,
      },
      vht_capab {
            max_mpdu_len=3895,
            supported_channel_width=0,
            rx_ldpc=enabled,
            gi_80=auto,
            gi_160=auto,
            tx_stbc=enabled,
            rx_stbc=enabled,
            su_beamformer_cap=yes,
            su_beamformee_cap=yes,
            beamformee_sts_count=1,
            sounding_dimensions=2,
            mu_beamformer_cap=yes,
            mu_beamformee_cap=yes,
            vht_txop_ps=yes,
            htc_vht_cap=yes,
            max_ampdu_len=7,
            vht_link_adaptation_cap=yes,
            rx_ant_pattern_consistency=yes,
            tx_ant_pattern_consistency=yes,
            vht_oper_bandwidth=1,
            seg0_center_freq=42,
            seg1_center_freq=0
      }
      essid=StructuredMesh,
      .....
      .....
      beacon_interval=100,
```

```
        dca=1,
        txpower=100,
        txrate=0,
        preamble_type=long,
        gfmode=disabled,
       ..........
  .    .........
}
```

### 9.3.1 Operation Mode

This configuration represented as medium_sub_type field in meshap.conf file. For each interface configuration this medium_sub_type field is present.

Valid values for *medium_sub_type* field are listed in below table,

| Value | Description |
|---|---|
| a | Supports only 802.11a standard, operates in 5GHz band |
| b | Supports only 802.11b standard, operates in 2.4GHz band |
| g | Supports only 802.11g standard , operates in 2.4GHz band |
| n | Supports only 802.11n standard, operates in 2.4GHz and 5GHz band based on device interface support |
| ac | Supports only 802.11ac standard, operates in 5GHz band |
| bg | Supports both 802.11b and 802.11g standard , operates in 2.4GHz band |
| bgn | Supports 802.11b, 802.11g and 802.11n standards, operates in 2.4GHz band |
| an | Supports 802.11a and 802.11n standard, operates in 5GHz band |
| anac | Supports 802.11a, 802.11n and 802.11ac, operates in 5GHz band |
| x | This value indicates interface medium is Wired Ethernet and don't support wireless standards |

**Default Value:** *bgn* for 2.4GHz interface and *anac* for 5GHz interface

### 9.3.2 Channel Width and Secondary Channel Offset

The channel width and secondary channel offset configuration represented as part of ht_capab field for each wireless interface configuration in meshap.conf file. The sub-field ht_bandwidth in ht_capab field represents bandwidth and secondary channel offset configured by user.

Valid values for sub-field *ht_bandwidth* in *ht_capab* field are listed in below table,

| Value | Description |
|---|---|
| 20 | Bandwidth configured as 20MHz |
| 40+ | Bandwidth configured as 40MHz and secondary channel offset is above the primary channel |

| | |
|---|---|
| 40- | Bandwidth configured as 40MHz and secondary channel offset is below the primary channel |

**Default Value:** *20* (20MHz bandwidth)

### 9.3.3 Maximum Transmit Rate

This configuration represented as txrates for each interface configuration in meshap.conf file.

When Operation Mode of 2.4GHz Downlink interface configured as b/g/n (i.e., medium_sub_type=bgn) and the operation mode of 5GHz Downlink interface configured as a/n (i.e., medium_sub_type=an) then the possible maximum transmit rate values are 6, 9, 12, 18, 24, 36, 48, 54, Auto. All these values are in Mbps.

**Default Value:** Auto

When Operation Mode of 2.4GHz or 5GHz Downlink interface configured as only 802.11n (i.e., medium_sub_type=n) then the maximum transmit rate values will vary based on Channel Bandwidth and Guard interval configurations. Here is the table lists the possible values as per Bandwidth and Guard interval values.

| **20 MHz Bandwidth** | **40 MHz Bandwidth** | | |
|---|---|---|---|
| **Long Guard Interval** | **Short Guard Interval** | **Long Guard Interval** | **Short Guard Interval** |
| 6.5 Mbps | 7.2 Mbps | 13.5 | 15 |
| 13 Mbps | 14.4 Mbps | 27 | 30 |
| 19.5 Mbps | 21.7 Mbps | 40.5 | 45 |
| 26 Mbps | 28.9 Mbps | 54 | 60 |
| 39 Mbps | 43.3 Mbps | 81 | 90 |
| 52 Mbps | 57.8 Mbps | 108 | 120 |
| 58.5 Mbps | 65 Mbps | 121.5 | 135 |
| 65 Mbps | 72.2 Mbps | 135 | 150 |
| Auto | Auto | Auto | Auto |

**Default Value:** Auto

### 9.3.4 LDPC Capability

This configuration represented as part of ht_capab field for each wireless interface configuration in meshap.conf file. The sub-field ldpc under ht_capab filed indicates LDPC supported or not.

Valid values for *ldpc* sub-filed under *ht_capab* field: *enabled* or *disabled*.

**Default Value:** enabled

### 9.3.5 SM (Saptial Mutliplexing) Power Save

**Saptial Multiplexing Power Save** (SMPS) configuration represented as part of ht_capab field for each wireless interface in meshap.conf file. The sub-field *smps* under ht_capab field indicates SMPS supported or not, If supported what is the mode it supports.

Valid values for sub-field *smps* in *ht_capab* field: *disabled* or *static* or *dynamic.*

**Default Value:** disabled

### 9.3.6 GF (Green Field) Mode

This configuration represented as *gfmode* field for each wireless interface configuration in meshap.conf file.

Valid values for gfmode field: *enabled* or *disabled*.

**Default Value:** disabled

### 9.3.7 Guard Interval

This configuration represented as part of ht_capab field for each wireless interface configuration in meshap.conf file. The sub-fields *gi_20* and *gi_40* (under *ht_capab* field) indicates short guard interval in 20MHz and 40MHz bandwidth is enabled or disabled.

Valid values for gi_20 and gi_40 sub-fields: *auto* or *long*.

**Default Value:** auto

### 9.3.8 Tx-STBC

This configuration represented as part of ht_capab field for each wireless interface configuration in meshap.conf file. The sub-field tx_stbc under ht_capab field indicates STBC (Spatial Time Block Coding) feature supported in Transmission path or not.

Valid values for *tx_stbc* sub-field under *ht_capab* field: *enabled* or *disabled*.

**Default Value:** enabled

### 9.3.9 Rx-STBC

This configuration represented as part of ht_capab field for each wireless interface configuration in meshap.conf file. The sub-field rx_stbc under ht_capab field indicates STBC (Spatial Time Block Coding) feature supported in Reception path or not.

Valid values for *rx_stbc* sub-field under *ht_capab* field: *enabled* or *disabled*.

**Default Value:** enabled

### 9.3.10 Delayed BlockAck

This configuration represented as part of ht_capab for each wireless interface configuration in meshap.conf file. The sub-field *delayed_ba* in field ht_capab indicates delayed block acknowledgement supported or not.

Valid values for *delayed_ba* sub-field: *enabled* or *disabled*.

**Default Value:** enabled

### 9.3.11 Preamble Type

This configuration represented in *preamble_type* field for each wireless interface in meshap.conf file.

Valid values for *preamble_type* field: *auto or short or long*.

**Default Value:** auto

### 9.3.12 BSS 40MHz DSSS/CCK Mode

This configuration represented as part of ht_capab field for each wireless interface configuration in meshap.conf file. The sub-field *dsss_cck_40* in ht_capab field indicates DSSS/CCK (Direct Sequence Spread-Spectrum/ Complementary Code Keying) mode in 40MHz is allowed or denied.

Valid values for dsss_cck_40 sub-field under ht_capab field: *allow* or *deny*.

**Default Value:** allow

### 9.3.13 40 MHz intolerant

This configuration represented as part of ht_capab field for each wireless interface configuration in meshap.conf file. The sub-field *40_intolerant* in ht_capab field indicates transmission in 40MHz bandwidth is supported in the current 2.4GHz BSS or not.

Valid values for *40_intolerant* sub-field: *enabled* or *disabled*.

**Default Value:** disabled

### 9.3.14 Tx Frame Aggregation

This configuration represented as part of frame_aggregation field for each interface in meshap.conf file. The sub-field tx_ampdu_enable under frame_aggregation field indicates aggregation is enabled in Transmission Path or not.

Valid values for *tx_ampdu_enable* sub-field: *enabled* or *disabled.*

**Default Value:** enabled

### 9.3.15 Max-AMPDU Length

This configuration represented as part of frame_aggregation field for each interface in meshap.conf file. The sub-field tx_ampdu_max_len under frame_aggregation field indicates maximum length of an AMPDU Frame (aggregated frame size) in Transmission Path.

Valid values for tx_ampdu_max_len sub-field: *8KB, 16KB, 32KB, and 64KB*.

**Default Value:** 64KB

### 9.3.16 Max-AMSDU Length

This configuration represented as part of ht_capab field in meshap.conf file. The sub-field *max_amsdu_len* under ht_capab field indicates maximum length of an AMSDU frame it can handle.

Valid values for *max_amsdu_len* sub-field are 0 and 1. Here 0 implies to 3839 octets and 1 implies 7935 octets.

**Default Value:** 1

### 9.3.17 Max MPDU Length

This configuration represented as part of vht_capab field in meshap.conf file. The sub-field *max_mpdu_len* under vht_capab field indicates maximum length of an MPDU frame it can handle.

Valid values for *max_mpdu_len* sub-field are 3895, 7991 and 11454.

**Default Value:** 3895

### 9.3.18 Supported Channel Width Set

This configuration represented as part of vht_capab field in meshap.conf file. The sub-field *supported_channel_width* under vht_capab field indicates channel bandwidth set.

Valid values for *supported_channel_width* sub-field are 0, 1 and 2.

Here 0=> doesn't support 160 and 80+80

1 => Supports 160 MHz

2 => Supports 160 MHz and 80+80MHz

**Default Value:** 0

### 9.3.19 Rx LDPC

This configuration represented as part of vht_capab field in meshap.conf file. The sub-field *rx_ldpc* under vht_capab field indicates support for receiving LDPC encoded packets.

**Default Value:** 1

### 9.3.20 GI 80MHz

This configuration represented as part of vht_capab field in meshap.conf file. The sub-field *gi_80* under vht_capab field indicates support for short guard interval in 80MHz channel bandwidth.

Valid values for gi_80 sub-field under vht_capab field: auto or long

**Default Value:** auto

### 9.3.21 GI 160MHz and 80+80MHz

This configuration represented as part of vht_capab field in meshap.conf file. The sub-field *gi_160* under vht_capab field indicates support for short guard interval in 160MHz or 80+80 MHz channel bandwidth.

Valid values for gi_160 sub-field under vht_capab field: auto or long

**Default Value:** auto

### 9.3.22 SU Beamformer Capable

This configuration represented as part of vht_capab field in meshap.conf file. The sub-field *su_beamformer_capab* under vht_capab field indicates support for operation as a single user beamformer.

Valid values for *su_beamformer_capab* sub-field under vht_capab field: yes or no

**Default Value:** yes

### 9.3.23 SU Beamformee Capable

This configuration represented as part of vht_capab field in meshap.conf file. The sub-field *su_beamformee_capab* under vht_capab field indicates support for operation as a single user beamformer.

Valid values for *su_beamformee_capab* sub-field under vht_capab field: yes or no

**Default Value:** yes

### 9.3.24 Beamformee STS Capability

This configuration represented as part of vht_capab field in meshap.conf file. The sub-field *beamformee_sts_count* under vht_capab field indicates the maximum number of beamformer antennas the beamformee can support when sending compressed beamforming feedback.

Valid values for *beamformee_sts_count* under vht_capab field: 1, 2, 3

**Default Value:** 1

### 9.3.25 Number of Sounding Dimensions

This configuration represented as part of vht_capab field in meshap.conf file. The sub-field *sounding_dimensions* under vht_capab field indicates the maximum value of the NUM_STS parameter in the TXVECTOR of a VHT NDP.

Valid values for *sounding_dimensions* under vht_capab field: 1, 2, 3

**Default Value:** 1

### 9.3.26 MU Beamformer Capable

This configuration represented as part of vht_capab field in meshap.conf file. The sub-field *mu_beamformer_capab* under vht_capab field indicates support for operation as an MU beamformer.

Valid values for *mu_beamformer_capab* under vht_capab field: yes or no

**Default Value:** yes

### 9.3.27 MU Beamformee Capable

This configuration represented as part of vht_capab field in meshap.conf file. The sub-field *mu_beamformee_capab* under vht_capab field indicates support for operation as an MU beamformee.

Valid values for *mu_beamformee_capab* under vht_capab field: yes or no

**Default Value:** yes

### 9.3.28 VHT TXOP PS

This configuration represented as part of vht_capab field in meshap.conf file. The sub-field *vht_txop_ps* under vht_capab field indicates whether or not the AP supports VHT TXOP Power Save Mode.

Valid values for *vht_txop_ps* under vht_capab field: yes or no

**Default Value:** yes

### 9.3.29 HTC-VHT

This configuration represented as part of vht_capab field in meshap.conf file. The sub-field *htc_vht_cap* under vht_capab field indicates whether or not the STA supports receiving a VHT variant HT Control field.

Valid values for *htc_vht_cap* under vht_capab field: yes or no

**Default Value:** yes

### 9.3.30 Maximum A-MPDU Length Exponent

This configuration represented as part of vht_capab field in meshap.conf file. The sub-field *max_ampdu_len* under vht_capab field indicates the maximum length of A-MPDU pre-EOF padding that the STA can receive. The length defined by this field is equal to 2 pow (13 + Maximum A-MPDU Length Exponent) -1 octets.

Valid values for *max_ampdu_len* under vht_capab field: 0 - 7

**Default Value:** 7

### 9.3.31 VHT Link Adaptation Capable

This configuration represented as part of vht_capab field in meshap.conf file. The sub-field *vht_link_adaptation_cap* under vht_capab field indicates STA supports link adaptation using VHT variant HT Control field.

Valid values for *vht_link_adaptation_cap* under vht_capab field: yes or no

**Default Value:** yes

### 9.3.32 Rx Antenna Pattern Consistency

This configuration represented as part of vht_capab field in meshap.conf file. The sub-field *rx_ant_pattern_consistency* under vht_capab field indicates the possibility of Rx antenna pattern change.

Valid values for *rx_ant_pattern_consistency* under vht_capab field: yes or no

**Default Value:** yes

### 9.3.33 Tx Antenna Pattern Consistency

This configuration represented as part of vht_capab field in meshap.conf file. The sub-field *tx_ant_pattern_consistency* under vht_capab field indicates the possibility of Tx antenna pattern change.

Valid values for *tx_ant_pattern_consistency* under vht_capab field: yes or no

**Default Value:** yes

### 9.3.34 vht_oper_bandwidth

This configuration represented as part of vht_capab field in meshap.conf file. The sub-field *vht_oper_bandwidth* under vht_capab field indicates operating channel bandwidth in VHT mode.

Valid values for *vht_oper_bandwidth* under vht_capab field: 0, 1, 2, 3

**Default Value:** 1

### 9.3.35 Seg0 Central Frequency

This configuration represented as part of vht_capab field in meshap.conf file. The sub-field *seg0_center_freq* under vht_capab field indicates channel center frequency for 80 and 160 MHz VHT BSS and the frequency segment 0 channel center frequency for 80+80 MHz.

Valid values for *seg0_center_freq* under vht_capab field: 0, 42, 58, 155

Value for seg0_center_feq could vary based on country domain code configuration.

**Default Value:** 42

### 9.3.36 Seg1 Central Frequency

This configuration represented as part of vht_capab field in meshap.conf file. The sub-field *seg1_center_freq* under vht_capab field indicates channel center frequency for 80 and 160 MHz VHT BSS and the frequency segment 0 channel center frequency for 80+80 MHz.

Valid values for *seg1_center_freq* under vht_capab field: 0, 42, 58, 155

Value for seg1_center_feq could vary based on country domain code configuration.

**Default Value:** 0

## 9.4 Openwrt commands to be used to configure above mentioned parameters

**Operation Mode:**

This configuration parameter sets current operation mode of the interface.

**Command to configure operation mode in 802.11n/ac:**

*iwpriv athN mode AUTO|11NAHT20|11NAHT40PLUS|11AC|…*

**Channel Width:**

This parameter configures Channel Width should be set on the interface for the current operating channel.

**Command to configure Channel Width:**

*iwpriv athN chwidth 0|1|2*

*Here , 0 => Use the device settings*

*1 => 20 MHz*

*2 => 40 MHz*

**AMPDU Frame Aggregation:**

AMPDU Frame Aggregation can be configured as enable or disable. Also maximum size of an AMPDU frame can be configured.

**Command to configure AMPDU Frame Aggregation:**

*iwpriv athN AMPDU 1|0*

*This command can be used to enable/disable Tx AMPDU aggregation on an interface.*

**AMPDU Frame Aggregation Factors:**

Frame aggregation limit can be configured based on number of sub-frames to place into an AMPDU Frame or based on number of bytes included in an AMPDU Frame.

**Command to configure AMPDU Frame size based on maximum number of sub-frames:**

*iwpriv athN AMPDUFrames 24*

**Command to configure AMPDU Frame size based on maximum number of bytes in an AMPDU Frame:**

*iwpriv athN AMPDULim 48000*

**GF (Green Field) Mode:**

Green Field Mode enables high throughput in WLAN network where all devices (AP and STA) support 802.11n standard. This field used to force 802.11n supported AP device to work purely in 802.11n mode and can't communicate with 802.11 a/b/g supported devices.

**Command to configure GF Mode:**

*iwpriv athN puren 1|0*

**Guard Interval:**

Guard Interval is intended to avoid signal loss from multipath effect. In 802.11n and 802.11ac mode guard interval can be configured as short interval.

If 802.11n/ac AP device have to support 802.11 a/b/g STA devices then, default guard interval (Long interval i.e., 800ns) should need to be set.

**Command to configure Guard Interval:**

*iwpriv athN shortgi 1*

**40MHz Intolerant:**

This parameter configures operation of the interface in both 20 and 40 MHz bandwidth. Setting HT20/HT40 Coexistence support for 802.11n supported interface indicates it can operate on 40MHz bandwidth in 2.4GHz band.

**Command to configure HT20/HT40 coexistence support:**

*iwpriv athN disablecoext 0|1*

**Physical rate:**

This parameter configures maximum number of Tx/Rx stream support. This configuration can override device settings.

**Command to configure Tx and Rx streams:**

*iwpriv athN chainmasksel 1 ( This command sets automatic chainmask selection enabled or disabled)*

*iwpriv athN txchainmask 0x05 (This command sets Tx chainmask value)*

*iwpriv athN rxchainmask 0x05 (This command sets Tx chainmask value)*

**STBC:**

This parameter enables/disables the Space-Time Block Coding (STBC) feature. This feature can be enabled only in Tx or Rx or both traffic.

**Command to set STBC feature:**

*iwpriv athN txstbc 0|1*

*iwpriv athN rxstbc 0|1*

## 9.5 Mapping of configuration parameters with NMS configuration field names

The below table list outs configuration parameters required to support 802.11n and 802.11ac with respect to NMS controller interface.

| **Basic Wireless Configurations** | | |
|---|---|---|
| **Common Wireless Configuration for 802.11n and ac** | | |
| **Configuration Parameters** | **Configuration Parameter Name in NMS** | **Remarks** |
| Operation Mode | Supported | Refer Section 9.5.1 |

| | | |
|---|---|---|
| | Protocol | |
| Channel Bandwidth | Channel Bandwidth | Refer Section 9.5.2 |
| Secondary Channel Offset | Secondary Channel Position | This field value has dependency of Channel Bandwidth value Refer Section 9.5.3 |
| Maximum Transmit Rate | Max Transmit Rate | This field value has dependency on Supported Protocol field Refer Section 9.5.4 |
| **Advanced Wireless Configurations** | | |
| **Common Wireless Configuration for 802.11n and ac** | | |
| **Configuration Parameters** | **Configuration Parameter Name in NMS** | **Remarks** |
| Preamble Type | Preamble | Refer Section 9.5.5 |
| Guard Interval | Guard Interval | Refer Section 9.5.6 |
| Tx Frame Aggregation | Frame Aggregation | Refer Section 9.5.7 |
| Max-AMPDU Length | Max A-MPDU | This field value has effect based on Frame Aggregation value Refer Section 9.5.8 |
| Max-AMSDU Length | Max A-MSDU | This field value has effect based on Frame Aggregation value Refer Section 9.5.9 |
| Tx-STBC | Tx-STBC | Refer Section 9.5.10 |
| Rx-STBC | Rx-STBC | Refer Section 9.5.11 |
| LDPC Capability | LDPC | Refer Section 9.5.12 |
| Green Field Mode | GF Mode | This field value has effect based on Supported Protocol value Refer Section 9.5.13 |
| **Wireless Configuration specific to only 802.11n** | | |
| 40 MHz intolerant | 20/40MHz coexistence | This field value has effect based on |

| | | |
|---|---|---|
| | | Channel bandwidth value Refer Section 9.5.14 |
| **Wireless Configuration specific to only 802.11ac** | | |
| Max-MPDU Length | Max MPDU | Refer Section 9.5.15 |

### 9.5.1 Supported Protocol

For 5GHz Wireless interface the below represents possible supported protocol values,

| Value | Description |
|---|---|
| a | Set to support only 802.11a standard |
| n | Set to support only 802.11n standard |
| ac | Set to support only 802.11ac standard |
| an | Set to support 802.11a and 802.11n standard |
| anac | Set to support 802.11a, 802.11n and 802.11ac |

**Default Value:** a-n-ac

For 2.4GHz Wireless interface the below represents possible supported protocol values,

| Value | Description |
|---|---|
| b | Set to support only 802.11b standard |
| g | Set to support only 802.11g standard |
| n | Set to support only 802.11n standard |
| bg | Set to support both 802.11b and 802.11g standards |
| bgn | Set to support 802.11b, 802.11g and 802.11n standards |

**Default Value:** b-g-n

### 9.5.2 Channel Bandwidth

| Value | Description |
|---|---|
| 20 MHz | Channel Bandwidth configured as 20MHz |
| 40 MHz | Channel Bandwidth configured as 40MHz |
| 80 MHz | Channel Bandwidth configured as 80MHz |

**Default Value:** 20MHz

### 9.5.3 Secondary Channel Position

This parameter used to indicate secondary channel position is above or below the (primary) channel when channel bandwidth set to 40MHz. When bandwidth is not 40MHz then this parameter don't have to be set (it can be none).

**Default Value:** none

### 9.5.4 Max Transmit Rate

| 20 MHz Bandwidth | 40 MHz Bandwidth | | |
|---|---|---|---|
| **Long Guard Interval** | **Short Guard Interval** | **Long Guard Interval** | **Short Guard Interval** |
| 6.5 Mbps | 7.2 Mbps | 13.5 | 15 |
| 13 Mbps | 14.4 Mbps | 27 | 30 |
| 19.5 Mbps | 21.7 Mbps | 40.5 | 45 |
| 26 Mbps | 28.9 Mbps | 54 | 60 |
| 39 Mbps | 43.3 Mbps | 81 | 90 |
| 52 Mbps | 57.8 Mbps | 108 | 120 |
| 58.5 Mbps | 65 Mbps | 121.5 | 135 |
| 65 Mbps | 72.2 Mbps | 135 | 150 |
| Auto | Auto | Auto | Auto |

**Default Value:** auto

### 9.5.5 Preamble

Valid values for *preamble* field: *auto or short or long*.

**Default Value:** auto

### 9.5.6 Guard Interval

Valid values for Guard interval field: *auto* or *long*.

**Default Value:** auto

### 9.5.7 Frame Aggregation

Valid values for Frame Aggregation field: *enabled* or *disabled.*

**Default Value:** enabled

### 9.5.8 Max A-MPDU

Valid values for Max A-MPDU field: *8KB, 16KB, 32KB, and 64KB*.

**Default Value:** 64KB

### 9.5.9 Max A-MSDU

Valid values for Max A-MSDU field: 3839 bytes or 7935 bytes.

**Default Value:** 3839 bytes

### 9.5.10 Tx-STBC

Valid values for Tx-STBC field: *enabled* or *disabled*.

**Default Value:** enabled

### 9.5.11 Rx-STBC

Valid values for Rx-STBC field: *enabled* or *disabled*.

**Default Value:** enabled

### 9.5.12 LDPC

Valid values for LDPC field: *enabled* or *disabled*.

**Default Value:** enabled

### 9.5.13 GF Mode

Valid values for GF Mode field: *enabled* or *disabled*.

**Default Value:** disabled

### 9.5.14 20/40MHz coexistence

Valid values for 20/40MHz coexistence field: yes or no.

**Default Value:** yes

### 9.5.15 Max MPDU

Valid values for Max MPDU field: 3895 or 7991 or 11454 bytes

**Default Value:** 3895 bytes

# 10 Packet Handling for 802.11n and 802.11ac

## 10.1 Information IE's to be handled

11
12
13
14
15
16
17
17.1

### 17.1.1 HT Capabilities element

An HT STA declares that it is an HT STA by transmitting the HT Capabilities element. The HT Capabilities element contains a number of fields that are used to advertise optional HT capabilities of an HT STA.

The HT Capabilities element is present in

- Beacon,
- Association Request,
- Association Response,
- Reassociation Request,
- Reassociation Response,

- Probe Request,
- Probe Response frames.

HT Capabilities element is defined in Figure Below

**FIGURE 10.18** HT Capabilities Element format

| Element ID | Length | HT Capabilities Info | A-MPDU Parameters | Supported MCS Set | HT Extended Capabilities | Transmit Beamforming Capabilities | ASEL Capabilities |
|---|---|---|---|---|---|---|---|
| Bytes: 1 | 1 | 2 | 1 | 16 | 2 | 4 | 1 |

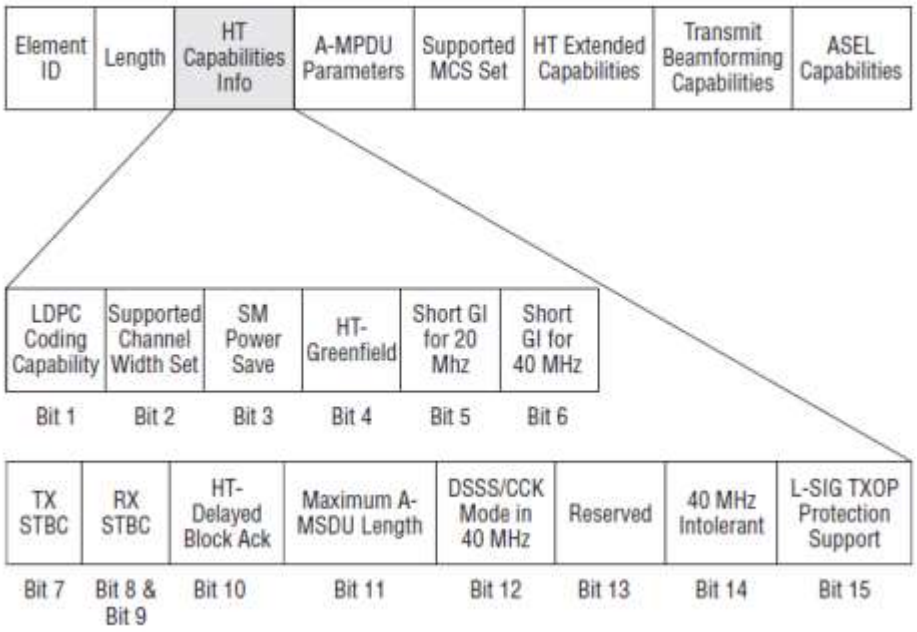
 

#### *17.1.1.1 HT Capabilities Info field*

The HT Capabilities Info field of the HT Capabilities element is 2 octets in length, and contains capability information bits.

The structure of this field as defined

**FIGURE 10.19** HT Capabilities Info field format




##### 17.1.1.1.1 LDPC Coding Capability (low-density parity check)

Indicates support for receiving LDPC coded packets

```
LDPC coding capability: [LDPC] = supported
```

##### 17.1.1.1.2 Supported Channel Width Set

0: if only 20 MHz operation is supported
1: if both 20 MHz and 40 MHz operation is supported

```
Supported channel width set:
        [HT40-] = both 20 MHz and 40 MHz with secondary channel below the
primary channel; secondary_channel = -1
        [HT40+] = both 20 MHz and 40 MHz with secondary channel above the
primary channel: secondary_channel = 1
        (20 MHz only if neither is set)

There are limits on which channels can be used with HT40- and HT40+.
```

Please note that 40 MHz channels may switch their primary and secondary channels if needed or creation of 40 MHz channel maybe rejected based on overlapping BSSes. These changes are done automatically when setting up the 40 MHz channel.

#### 17.1.1.1.3 Spatial Multiplexing Power Save

Indicates the spatial multiplexing power save mode

Set to 0 for static SM power save mode
Set to 1 for dynamic SM power save mode
Set to 3 for SM Power Save disabled

```
Spatial Multiplexing (SM) Power Save:
        [SMPS-STATIC] or [SMPS-DYNAMIC]
        (SMPS disabled if neither is set)
```

#### 17.1.1.1.4 High Throughput Green-Field

Indicates support for the reception of PPDUs (Protocol data units) with HT-greenfield format:

0: if not supported
1: if supported

```
HT-greenfield: [GF] (disabled if not set)
```

#### 17.1.1.1.5 Short GI for 20 MHz

Indicates short GI support for the reception of packets transmitted with TXVECTOR parameter CH_BANDWIDTH set to HT_CBW20

```
Short GI for 20 MHz: [SHORT-GI-20] (disabled if not set)
```

#### 17.1.1.1.6 Short GI for 40 MHz

Indicates short GI support for the reception of packets transmitted with TXVECTOR parameter CH_BANDWIDTH set to HT_CBW40

```
Short GI for 40 MHz: [SHORT-GI-40] (disabled if not set)
```

#### 17.1.1.1.7 Tx STBC (space-time block coding)

Indicates support for the transmission of PPDUs using STBC

```
Tx STBC: [TX-STBC] (disabled if not set)
```

#### 17.1.1.1.8 Rx STBC

Indicates support for the reception of PPDUs using STBC

```
Rx STBC:
        [RX-STBC1] (one spatial stream)
        [RX-STBC12] (one or two spatial streams)
        [RX-STBC123] (one, two, or three spatial streams)
        Rx STBC disabled if none of these set
```

#### 17.1.1.1.9 HT-Delayed Block Ack

Indicates support for HTdelayed Block Ack operation

```
HT-delayed Block Ack: [DELAYED-BA] (disabled if not set)
```

#### 17.1.1.1.10 Maximum A-MSDU Length

Indicates maximum AMSDU length as per 9.7c
0 for 3839 octets
1 for 7935 octets

```
Maximum A-MSDU length: [MAX-AMSDU-7935] for 7935 octets
        (3839 octets if not set)
```

9.7c A-MSDU operation (some key points)
An A-MSDU shall contain only MSDUs whose DA and SA parameter values map to the same RA and TA values.
The constituent MSDUs of an A-MSDU shall all have the same priority parameter value from the corresponding MA-UNITDATA.request.
An A-MSDU shall be carried, without fragmentation, within a single QoS data MPDU.

#### 17.1.1.1.11 DSSS/CCK Mode in 40 MHz

Indicates use of DSSS/CCK mode in a 20/40 MHz BSS based on "11.14 20/40 MHz BSS operation"

```
DSSS/CCK Mode in 40 MHz: [DSSS_CCK-40] = allowed (not allowed if not set)
```

#### 17.1.1.1.12 Forty MHz Intolerant (20/40MHz switching allowed or not)

1=prohibit use for 40MHz channel in 2.4GHz
Indicates whether APs receiving this information or reports of this information are required to prohibit 40 MHz transmissions based on "11.14.12 Switching between 40 MHz and 20 MHz"

```
40 MHz intolerant [40-INTOLERANT] (not advertised if not set)
```

#### 17.1.1.1.13 L-SIG TXOP Protection Support

Indicates support for the LSIG TXOP protection mechanism "9.13.5 L-SIG TXOP protection"

```
L-SIG TXOP protection support: [LSIG-TXOP-PROT] (disabled if not set)
```

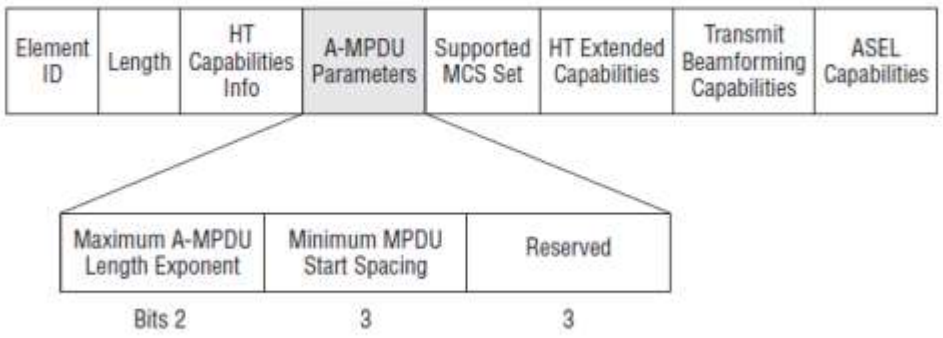
 

### 17.1.1.2 *A-MPDU Parameters field*

The structure of the A-MPDU Parameters field of the HT Capabilities element is shown below

**FIGURE 10.34** A-MPDU Parameters Field format




#### 17.1.1.2.1 Maximum A-MPDU Length Exponent

Indicates the maximum length of A-MPDU that the STA can receive.

This field is an integer in the range 0 to 3. The length defined by this field is equal to $2^{(13 + \text{Maximum A-MPDU Length Exponent})} - 1$ Octets

#### 17.1.1.2.2 Minimum MPDU Start Spacing

Determines the minimum time between the start of adjacent MPDUs within an AMPDU that the STA can receive, measured at the PHY-SAP

These value doesn't seem to be set by the hostapd, value is set by default in ath10k.

Ath10k code

*"ath10k_mac_setup_ht_vht_cap ()"*

*ht_cap.ampdu_factor = IEEE80211_HT_MAX_AMPDU_64K;*

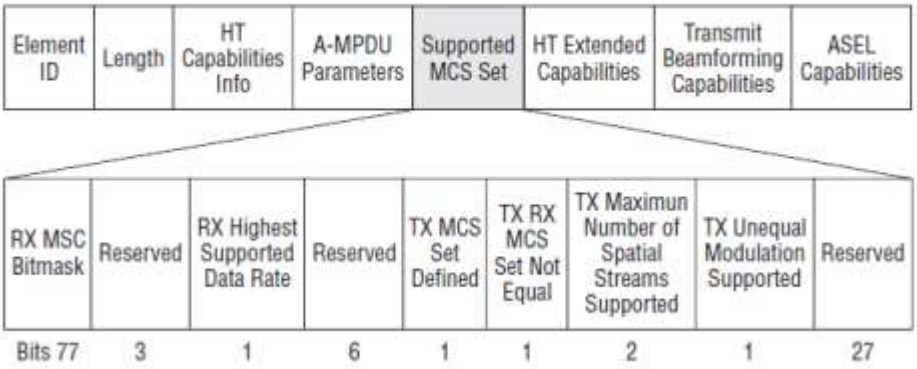
*ht_cap.ampdu_density = IEEE80211_HT_MPDU_DENSITY_8;*

### 17.1.1.3 *Supported MCS Set field*

The Supported MCS Set field of the HT Capabilities element indicates which MCSs a STA supports.

**FIGURE 10.38** Supported MCS Set format




#### 17.1.1.3.1 RX MCS Bitmask

one bit for each 77 (0-76) MCSs, few are mandatory for HT-STA

#### 17.1.1.3.2 RX Highest Supported Rate

Define the highest data rate that STA support, however a STA is not required to provide that information & may set to 0.

#### 17.1.1.3.3 TX Supported MCS set

TX & RX MCS set

TX Max Spatial Stream Supported

#### 17.1.1.3.4 TX Unequal Modulation

**TABLE 10.13** TX modulation set encoding

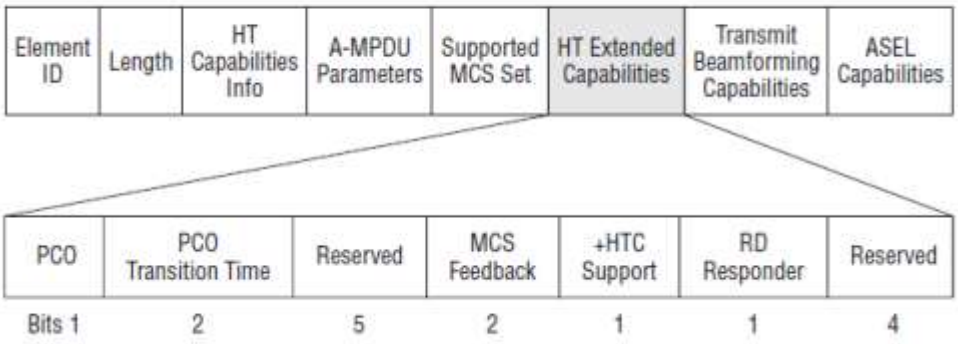
| Condition | TX MCS set defined | TX RX MCS set not equal | TX maximum spatial streams supported | TX unequal modulation supported |
|---|---|---|---|---|
| No TX MCS set is defined. | 0 | 0 | 0 | 0 |
| The TX MCS set is defined to be equal to the RX MCS set. | 1 | 0 | 0 | 0 |
| The TX MCS set may differ from the RX MCS set. | 1 | 1 | 0 = 1 spatial stream<br>1 = 2 spatial streams<br>2 = 3 spatial streams<br>3 = 4 spatial streams | 0 = UEQM not supported<br>1 = UEQM supported |

### 17.1.1.4 *HT Extended Capabilities*

The structure of the HT Extended Capabilities field of the HT Capabilities element is defined as below

**FIGURE 10.40** HT Extended Capabilities field format




#### 17.1.1.4.1 PCO (phased coexistence operation

Indicates Support for PCO.

*3A.43 phased coexistence operation (PCO):* A basic service set (BSS) mode with alternating 20 MHz and 40 MHz phases controlled by an access point (AP).

#### 17.1.1.4.2 PCO Transition Time

When **transmitted by a non-AP STA**: indicates that the STA can switch between 20 MHz channel width and 40 MHz channel width within the specified time.

When **transmitted by an AP**: indicates the PCO Transition Time to be used during PCO operation

#### 17.1.1.4.3 MCS Feedback

Indicates whether the STA can provide MFB (MCS FEED BACK)

Set to 0 (No Feedback) if the STA does not provide MFB

Set to 2 (Unsolicited) if the STA provides only unsolicited MFB

Set to 3 (Both) if the STA can provide MFB in response to MRQ
(either Delayed or Immediate, see 9.18.1) as well as unsolicited MFB

#### 17.1.1.4.4 +HTC Support

Indicates support of the HT Control field as per "9.7a HT Control field operation"

#### 17.1.1.4.5 RD Responder

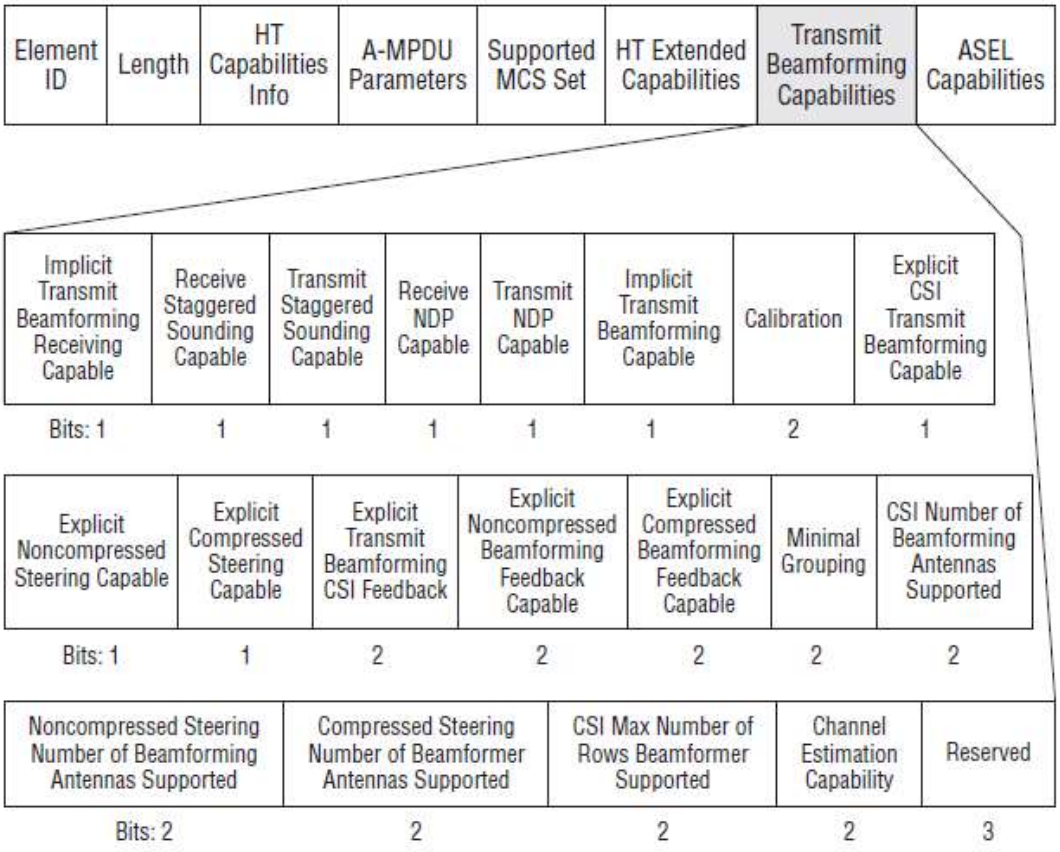
direction responder, i.e., the STA may use an offered RDG to transmit data to an RD initiator using the Reverse Direction Protocol described in "9.15 Reverse Direction Protocol"

### *17.1.1.5 Transmit Beamforming Capabilities*

The structure of the Transmit Beamforming Capabilities field of the HT Capabilities element is defined as below

**FIGURE 10.42** Transmit Beamforming Capabilities field format




Transmit beamforming and Chain support seems to be handled by the hostapd in 11AC configuration

#### *17.1.1.6* *ASEL Capability field*

The structure of the ASEL Capability field of the HT Capabilities element is defined in figure below

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Antenna Selection Capable | Explicit CSI Feedback Based Transmit ASEL Capable | Antenna Indices Feedback Based Transmit ASEL Capable | Explicit CSI Feedback Capable | Antenna Indices Feedback Capable | Receive ASEL Capable | Transmit Sounding PPDUs Capable | Reserved |

### 17.1.2 HT Operation element

This element id is present in the beacon, probe response, re-association frames

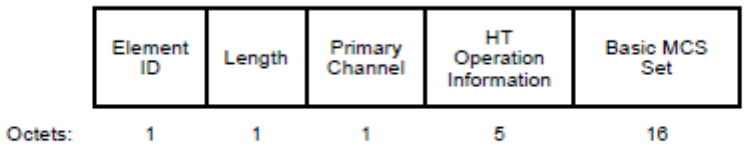



**Figure 8-255—HT Operation element format**

#### *17.1.2.1 Primary Channel*

Indicates the channel number of the primary channel.

#### *17.1.2.2 HT Operation information*

The structure of the HT Operation Information field is shown in Figure 8-256.

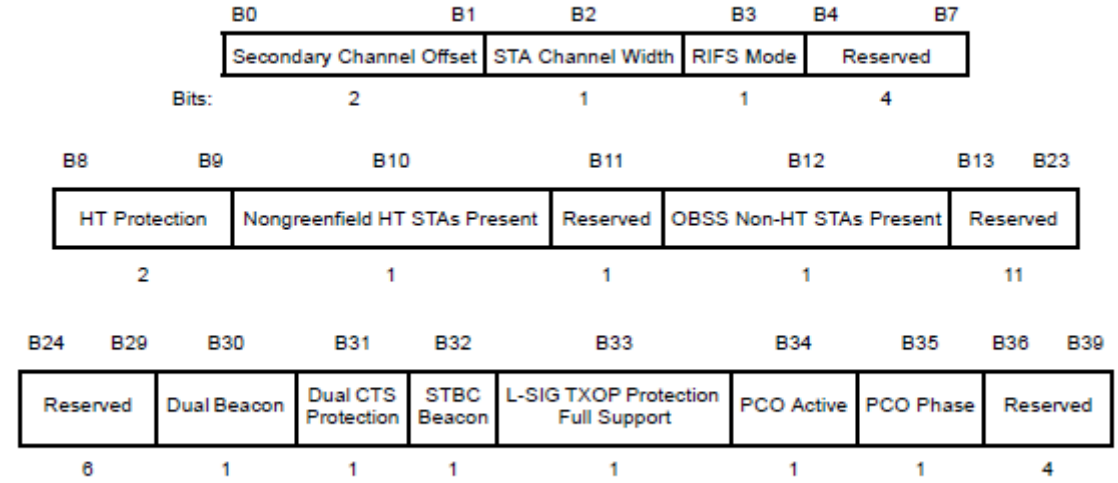



**Figure 8-256—HT Operation Information field**

**17.1.2.2.1** Secondary Channel Offset

Indicates the offset of the secondary channel relative to the primary channel

1 - if the secondary channel is above the primary channel

3 - if the secondary channel is below the primary channel

0 - if there is no secondary channel

**17.1.2.2.2** Station Channel Width

0 - for a 20 Mhz channel

1 - allows use of any channel width in the Supported channel width set

#### 17.1.2.2.3 Reduced Interframe support (RIFS) mode

Indicates whether the use of reduced interframe space is permitted within the BSS. RIFS can be used only when a HT Greenfield network is in place.

0 – RIFS is prohibited

1 – RIFS is permitted

#### 17.1.2.2.4 HT Protection

Indicates protection requirements of HT transmissions.

0 – no protection mode, all the stations detected are HT supported STA & are 20 or 40MHz BW

1 – non-member protection mode, a non HT STA is detected & is not known by the transmitting STA to be a member of the BSS

2 – 20 MHz protection mode, all the stations detected are HT STA, BSS is 20/40 MHz supported and there is one station connected with 20MHz and rest all are 40 MHz

3 – non-HT mixed mode, if non off the above

#### 17.1.2.2.5 Nongreenfield HT STAs Present

AP indicates if any HT STAs that are not HT-greenfield capable have associated

0 – if all HT STAs that are associated are HT-greenfield capable
1 - if one or more HT STAs that are not HT-greenfield capable are associated

#### 17.1.2.2.6 OBSS Non-HT STAs Present

Indicates if the use of protection for non-HT STAs by overlapping BSSs is determined to be desirable. The OBSS Non-HT STAs Present field allows HT APs to report the presence of non-HT STAs that are not members of its BSS in the primary channel, the secondary channel, or in both primary and secondary channels.

1 – use of protection for non-HT STAs by OBSSs is determined to be desirable.
0 – otherwise

#### 17.1.2.2.7 Dual Beacon

Indicates whether the AP transmits an STBC beacon.
0 – no STBC beacon is transmitted
1 – STBC beacon is transmitted by the AP

#### 17.1.2.2.8 Dual CTS Protection

Dual CTS protection is used by the AP to set a NAV at STAs that do not support STBC and at STAs that can associate solely through the STBC beacon.

If the Dual CTS Protection field of the HT Operation element has value 1 in the Beacon frames transmitted by its AP, a non-AP HT STA shall start every TXOP with an RTS addressed to the AP.

0 – dual CTS protection is not required
1 – dual CTS protection is required

#### 17.1.2.2.9 STBC Beacon

Indicates whether the beacon containing this element is a primary or an STBC beacon. The STBC beacon has half a beacon period shift relative to the primary beacon. Defined only in a Beacon transmitted by an AP. Otherwise reserved.

0 – primary beacon

1 – STBC beacon

#### 17.1.2.2.10 L-SIG TXOP Protection Full Support

Indicates whether all HT STA in the BSS support L-SIG TXOP protection

0 – one or more HT STA in the BSS do not support L-SIG TXOP protection

1 – all HT STA in the BSS support L-SIG TXOP protection

#### 17.1.2.2.11 PCO Active

Indicates whether PCO is active in the BSS. Present in Beacon/Probe Response frames transmitted by an AP. Otherwise reserved. Non-PCO STAs regard the BSS as a 20/40 MHz BSS and may associate with the BSS without regard to this field

0 – PCO is not active in the BSS
1 – PCO is active in the BSS

#### 17.1.2.2.12 PCO Phase

Indicates the PCO phase of operation Defined only in a Beacon and Probe Response frames when PCO Active is 1. Otherwise reserved.

0 – indicates switch to or continue 20 MHz phase
1 – indicates switch to or continue 40 MHz phase

### *17.1.2.3 Basic MCS Set*

Indicates the MCS values that are supported by all HT STAs in the BSS.

The Basic MCS Set is a bitmap of size 128 bits Bit 0 corresponds to MCS 0. A bit is set to 1 to indicate support for that MCS and 0 otherwise.

This field has a similar format to the RX MCS Bitmask subfield in the supported MCS Set field of the HT Capabilities Element. Unlike the RX MCS Set subfield, which shows the MCS supported by the AP, the Basic MCS Set field shows only MCS values that are supported by all HT STAs within the BSS.

## 17.1.3 20/40 BSS Coexistence element

The 20/40 BSS Coexistence element is formatted as shown

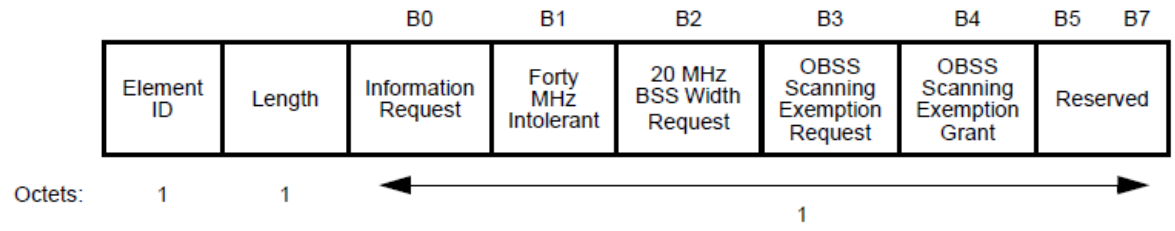



**Figure 7-95o27—20/40 BSS Coexistence element format**

#### *17.1.3.1 Information Request –*

The Information Request field is used to indicate that a transmitting STA is requesting the recipient to transmit a 20/40 BSS Coexistence Management frame with the transmitting STA as the recipient

#### *17.1.3.2 Forty MHz intolerant –*

1 - prohibit an AP that receives this information or reports of this information from operating a 20/40 MHz BSS.

0 - it does not prohibit a receiving AP from operating a 20/40 MHz BSS. This field is used for inter-BSS communication

#### *17.1.3.3 20Mhz BSS width request*

1 - prohibit a receiving AP from operating its BSS as a 20/40 MHz BSS. This field is used for intra-BSS communication

#### *17.1.3.4 OBSS Scanning Exemption Request*

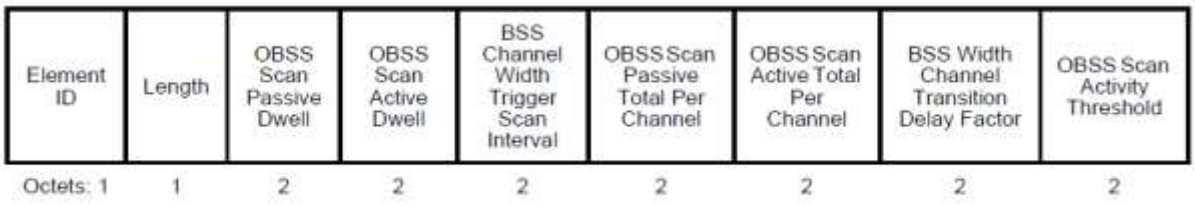
1- indicate that the transmitting non-AP STA is requesting the BSS to allow the STA to be exempt from OBSS scanning

#### *17.1.3.5 OBSS Scanning Exemption Grant field*

1 - by an AP to indicate that the receiving STA is exempted from performing OBSS Scanning

### 17.1.4 Overlapping BSS Scan Parameters element



| Element ID | Length | OBSS Scan Passive Dwell | OBSS Scan Active Dwell | BSS Channel Width Trigger Scan Interval | OBSS Scan Passive Total Per Channel | OBSS Scan Active Total Per Channel | BSS Width Channel Transition Delay Factor | OBSS Scan Activity Threshold |
|---|---|---|---|---|---|---|---|---|
| Octets: 1 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |

**Figure 7-95o26—Overlapping BSS Scan Parameters element format**

#### *17.1.4.1 OBSS Scan Passive Dwell*

Value in TUs, encoded as an unsigned integer, that a receiving STA uses to set its dot11OBSSScanPassiveDwell MIB variable

#### *17.1.4.2 OBSS Scan Active Dwell*

Value in TUs, encoded as an unsigned integer, that a receiving STA uses to set its dot11OBSSScanActiveDwell MIB variable

#### *17.1.4.3 BSS Channel Width Trigger Scan Interval*

Value in seconds, encoded as an unsigned integer, that a receiving STA uses to set its dot11BSSWidthTriggerScanInterval MIB variable

#### *17.1.4.4 OBSS Scan Passive Total Per Channel*

Value in TUs, encoded as an unsigned integer, that a receiving STA uses to set its Dot11OBSSScanPassiveTotalPerChannel MIB variable

#### *17.1.4.5 OBSS Scan Active Total Per Channel*

Value in TUs, encoded as an unsigned integer, that a receiving STA uses to set its dot11OBSSScanActiveTotalPerChannel MIB variable

#### *17.1.4.6 BSS Width Channel Transition Delay Factor*

Value that a receiving STA uses to set its dot11BSSWidthChannelTransitionDelayFactor MIB variable

#### *17.1.4.7 OBSS Scan Activity Threshold*

Value in hundredths of percent, encoded as an unsigned integer, that a receiving STA uses to set its dot11OBSSScanActivityThreshold MIB

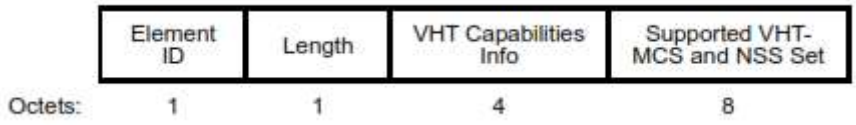
 

## 17.1.5 VHT Capabilities Element

### *17.1.5.1 VHT Capabilities element structure*

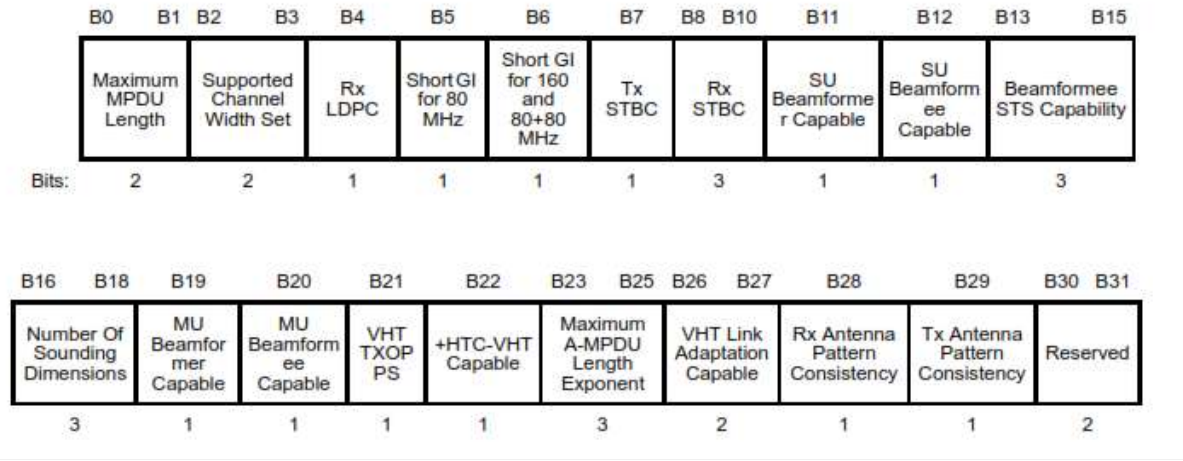
A VHT STA declares that it is a VHT STA by transmitting the VHT Capabilities element.
The VHT Capabilities element contains a number of fields that are used to advertise VHT capabilities of a VHT STA.

The VHT Capabilities element is defined in Figure as below,




### *17.1.5.2 VHT Capabilities Info field*

The structure of the VHT Capabilities Info field is defined in Figure below,




#### 17.1.5.2.1 Maximum MPDU Length

Indicates the maximum MPDU length. Here is the list of possible values encoded in this sub-field,
Set to 0 for 3895 octets.
Set to 1 for 7991 octets.
Set to 2 for 11 454 octets.
The value 3 is reserved

#### 17.1.5.2.2 Supported Channel Width Set

Indicates the channel widths supported by the STA. Here is the list of possible values encoded in this sub-field,
Set to 0 if the STA does not support either 160 or 80+80 MHz.
Set to 1 if the STA supports 160 MHz.
Set to 2 if the STA supports 160 MHz and 80+80 MHz.
The value 3 is reserved.

#### 17.1.5.2.3 Rx LDPC

Indicates support for receiving LDPC encoded packets. Valid values that can be encoded in this sub-field are,
Set to 0 if not supported.
Set to 1 if supported.

#### 17.1.5.2.4 Short GI for 80 MHz

Indicates short GI support for the reception of packets transmitted with TXVECTOR parameters FORMAT equal to VHT and CH_BANDWIDTH equal to CBW80.
Set to 0 if not supported.
Set to 1 if supported.

#### 17.1.5.2.5 Short GI for 160 and 80+80 MHz

Indicates short GI support for the reception of packets transmitted with TXVECTOR parameters FORMAT equal to VHT and CH_BANDWIDTH equal to CBW160 or CBW80+80.
Set to 0 if not supported.
Set to 1 if supported.

#### 17.1.5.2.6 Tx STBC

Indicates support for the transmission of at least 2x1 STBC.
Set to 0 if not supported.
Set to 1 if supported.

#### 17.1.5.2.7 Rx STBC

Indicates support for the reception of PPDUs using STBC.
Set to 0 for no support.
Set to 1 for support of one spatial stream.
Set to 2 for support of one and two spatial streams.
Set to 3 for support of one, two, and three spatial streams.
Set to 4 for support of one, two, three, and four Spatial streams.
The values 5, 6, 7 are reserved.

#### 17.1.5.2.8 SU Beamformer Capable

Indicates support for operation as an SU beamformer.
Set to 0 if not supported.
Set to 1 if supported.

#### 17.1.5.2.9 SU Beamformee Capable

Indicates support for operation as an SU beamformee.
Set to 0 if not supported.
Set to 1 if supported.

#### 17.1.5.2.10 Beamformee STS Capability

The maximum number of space-time streams that the STA can receive in a VHT NDP, the maximum value for *NSTS total* that can be sent to the STA in a VHT MU PPDU if the STA is MU beamformee

capable, and the maximum value of *Nr* that the STA transmits in a VHT Compressed Beamforming frame.

If SU beamformee capable, set to maximum number of space-time streams that the STA can receive in a VHT NDP minus 1. Otherwise, reserved.

#### 17.1.5.2.11 Number of Sounding Dimension

Beamformer's capability indicating the maximum value of the TXVECTOR parameter NUM_STS for a VHT NDP

#### 17.1.5.2.12 MU Beamformer Capable

Indicates support for operation as an MU beamformer.
Set to 0 if not supported or if SU Beamformer Capable is set to 0 or if sent by a non-AP STA.
Set to 1 if supported and SU Beamformer Capable is set to 1.

#### 17.1.5.2.13 MU Beamformee Capable

Indicates support for operation as an MU beamformee.
Set to 0 if not supported or if SU Beamformee Capable is set to 0 or if sent by an AP.
Set to 1 if supported and SU Beamformee Capable is set to 1.

#### 17.1.5.2.14 VHT TXOP PS

Indicates whether the AP supports VHT TXOP Power Save Mode or whether the non-AP STA has enabled VHT TXOP Power Save mode.
Set to 0 if the AP does not support TXOP Power Save Mode.
Set to 1 if the AP supports TXOP Power Save Mode.
Set to 0 if the non-AP STA does not enable TXOP Power Save Mode.
Set to 1 if the non-AP STA enables TXOP Power Save Mode.

#### 17.1.5.2.15 +HTC VHT Capable

Indicates whether the STA supports receiving a VHT variant HT Control field.
Set to 0 if not supported.
Set to 1 if supported.

#### 17.1.5.2.16 Maximum A-MPDU Length Exponent

Indicates the maximum length of A-MPDU that the STA can receive.EOF padding is not included in this limit.
This field is an integer in the range of 0 to 7.
The length defined by this field is equal to 2 (13+ Maximum A-MPDU Length Exponent) octets.

#### 17.1.5.2.17 VHT Link Adaptation Capable

Indicates whether the STA supports link adaptation using VHT variant HT Control field.
If +HTC-VHT Capable is 1:

Set to 0 (No Feedback) if the STA does not provide VHT MFB.
Set to 2 (Unsolicited) if the STA provides only unsolicited VHT MFB.

Set to 3 (Both) if the STA can provide VHT MFB in response to VHT MRQ and if the STA provides unsolicited VHT MFB.

The value 1 is reserved.

Reserved if +HTC-VHT Capable is 0

#### 17.1.5.2.18 RX Antenna Pattern Consistency

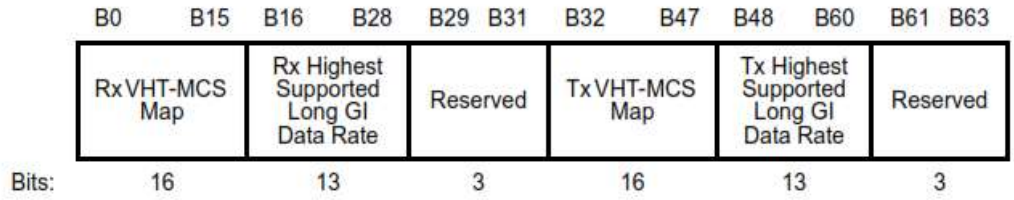
Indicates the possibility of a receive antenna pattern change.
Set to 0 if receive antenna pattern might change during the lifetime of the current association.
Set to 1 if receive antenna pattern does not change during the lifetime of the current association.

#### 17.1.5.2.19 Tx Antenna Pattern Consistency

Indicates the possibility of a transmit antenna pattern change.
Set to 0 if the transmit antenna pattern might change during the lifetime of the current association.
Set to 1 if the transmit antenna pattern does not change during the lifetime of the current association.

### *17.1.5.3 Supported VHT-MCS and NSS Set field*

The Supported VHT-MCS and NSS Set field is used to convey the combinations of VHT-MCSs and spatial streams that a STA supports for reception and the combinations that it supports for transmission. The structure of the field is shown in below figure,

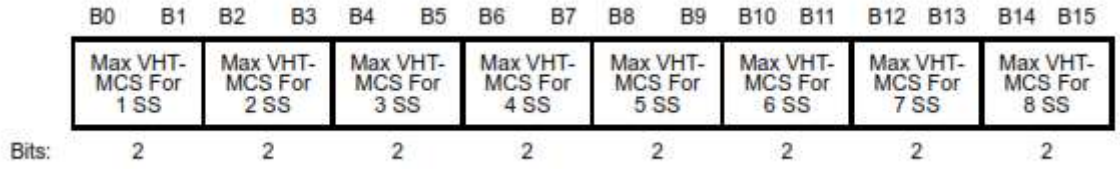



The Supported VHT-MCS and NSS Set subfields are defined in below Table,

| Subfield | Definition | Encoding |
|---|---|---|
| Rx VHT-MCS Map | Indicates the maximum value of the RXVECTOR parameter MCS of a PPDU that can be received at all channel widths supported by this STA for each number of spatial streams. | The format and encoding of this subfield are defined in Figure 8-401bs and the associated description. |
| Rx Highest Supported Long GI Data Rate | Indicates the highest long GI VHT PPDU data rate that the STA is able to receive. | The largest integer value less than or equal to the highest long GI VHT PPDU data rate in Mb/s the STA is able to receive (see 9.7.11.1).<br><br>The value 0 indicates that this subfield does not specify the highest long GI VHT PPDU data rate that the STA is able to receive. |
| Tx VHT-MCS Map | Indicates the maximum value of the TXVECTOR parameter MCS of a PPDU that can be transmitted at all channel widths supported by this STA for each number of spatial streams. | The format and encoding of this subfield are defined in Figure 8-401bs and the associated description. |
| Tx Highest Supported Long GI Data Rate | Indicates the highest long GI VHT PPDU data rate that the STA is able to transmit at. | The largest integer value less than or equal to the highest long GI VHT PPDU data rate in Mb/s that the STA is able to transmit (see 9.7.11.2).<br><br>The value 0 indicates that this subfield does not specify the highest long GI VHT PPDU data rate that the STA is able to transmit. |

The Rx VHT-MCS Map subfield and the Tx VHT-MCS Map subfield have the structure shown in below figure,

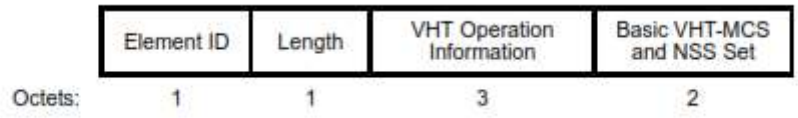



The Max VHT-MCS For *n* SS subfield (where *n* = 1, ..., 8) is encoded as follows:
— 0 indicates support for VHT-MCS 0-7 for *n* spatial streams
— 1 indicates support for VHT-MCS 0-8 for *n* spatial streams
— 2 indicates support for VHT-MCS 0-9 for *n* spatial streams
— 3 indicates that *n* spatial streams is not supported
NOTE—A VHT-MCS indicated as supported in the VHT-MCS Map fields for a particular number of spatial streams might not be valid at all bandwidths (see 22.5) and might be limited by the declaration of Tx Highest Supported Long GI Data Rates and Rx Highest Supported Long GI Data Rates and might be affected by 9.7.11.3.

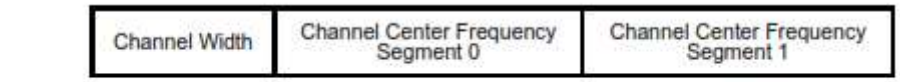
 

### 17.1.5.4 *VHT Operation element*

The operation of VHT STAs in the BSS is controlled by the HT Operation element and the VHT Operation element. The format of the VHT Operation element is defined in below figure,




The structure of the VHT Operation Information field is defined in below figure,




The VHT STA gets the primary channel information from the HT Operation element. The subfields of the VHT Operation Information field are defined in below table.

The Basic VHT-MCS and NSS Set field indicates the VHT-MCSs for each number of spatial streams in VHT PPDUs that are supported by all VHT STAs in the BSS (including IBSS and MBSS). The Basic VHTMCS
and
NSS Set field
is a bitmap
of
size 16
bits;
each
2 bits indicates
the supported
VHT-MCS
set
for
*N*
from 1 to 8. The Basic VHT-MCS and NSS Set field is defined in Figure

| Field | Definition | Encoding |
|---|---|---|
| Channel Width | This field, together with the HT Operation element STA Channel Width field, defines the BSS operating channel width (see 10.39.1). | Set to 0 for 20 MHz or 40 MHz operating channel width.<br>Set to 1 for 80 MHz operating channel width.<br>Set to 2 for 160 MHz operating channel width.<br>Set to 3 for 80+80 MHz operating channel width.<br>Values in the range 4 to 255 are reserved. |
| Channel Center Frequency Segment 0 | Defines the channel center frequency for an 80 and 160 MHz VHT BSS and the frequency segment 0 channel center frequency for an 80+80 MHz VHT BSS. See 22.3.14. | For 80 MHz or 160 MHz operating channel width, indicates the channel center frequency index for the 80 MHz or 160 MHz channel on which the VHT BSS operates.<br>For 80+80 MHz operating channel width, indicates the channel center frequency index for the 80 MHz channel of frequency segment 0 on which the VHT BSS operates.<br>Reserved otherwise. |
| Channel Center Frequency Segment 1 | Defines the frequency segment 1 channel center frequency for an 80+80 MHz VHT BSS. See 22.3.14. | For an 80+80 MHz operating channel width, indicates the channel center frequency index of the 80 MHz channel of frequency segment 1 on which the VHT BSS operates. Reserved otherwise. |

## 17.2 Management Frame Handling

### 17.2.1 Management Frame Handling in Mac802.11

### 17.2.2 Management Frame Handling in Meshap

#### *17.2.2.1 Getting the support HT Capabilites for the underlying driver (ath10k):*

The ath10k drivers exposes its supported configurations which is to be fetched by making NL socket call.

Ath10k  set the following capabilities:

AMPDU Parameter

IEEE80211_HT_CAP_SUP_WIDTH_20_40

IEEE80211_HT_CAP_DSSSCCK40

IEEE80211_HT_CAP_SGI_20

IEEE80211_HT_CAP_SGI_40

IEEE80211_HT_CAP_TX_STBC

IEEE80211_HT_CAP_RX_STBC

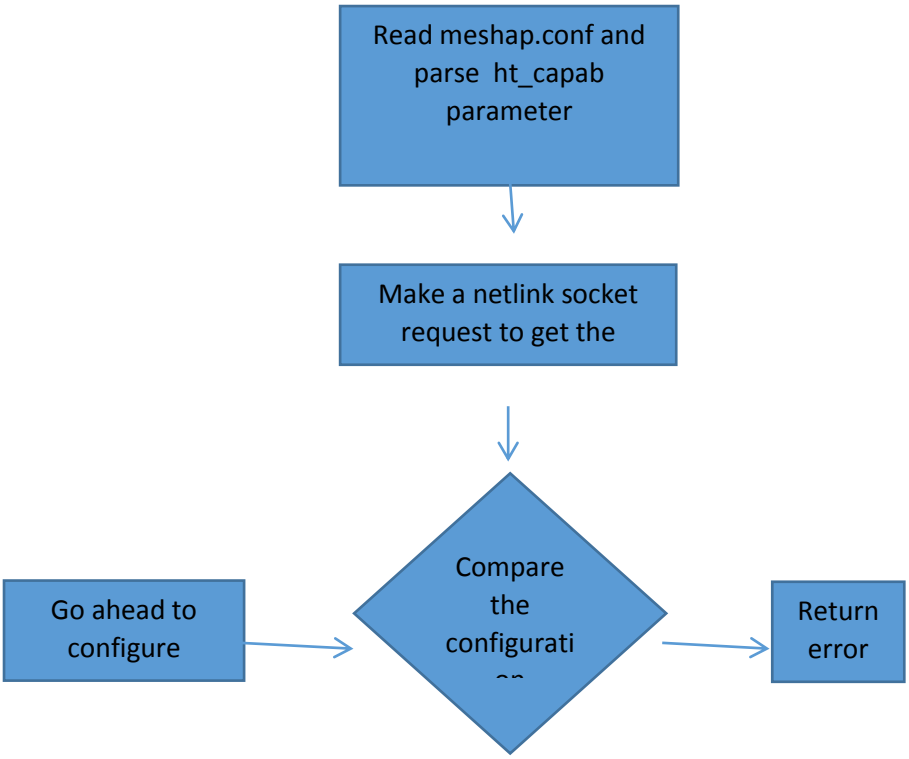
IEEE80211_HT_CAP_LDPC_CODING

IEEE80211_HT_CAP_MAX_AMSDU

IEEE80211_HT_MCS_TX_DEFINED

Meshap can get this configuration from the driver and set those values in the beacon frames.




#### *17.2.2.2 AP Side -- Updating of HT Elements in Beacon Frames*

This is done by ieee802_11_set_beacon. This is the same API currently we are using to set beacon. Need to add HT Parameters in the beacon frames.

Function : ieee802_11_set_beacon()

```
#ifdef CONFIG_IEEE80211N
    tailpos = hostapd_eid_ht_capabilities(hapd, tailpos);
    tailpos = hostapd_eid_ht_operation(hapd, tailpos);
#endif  /* CONFIG_IEEE80211N */
```

Snippet:

```
cap->ht_capabilities_info = host_to_le16(hapd->iconf->ht_capab);
cap->a_mpdu_params = hapd->iface->current_mode->a_mpdu_params;
os_memcpy(cap->supported_mcs_set, hapd->iface->current_mode->mcs_set,
                  16);
```

```
/* TODO: ht_extended_capabilities (now fully disabled) */
/* TODO: tx_bf_capability_info (now fully disabled) */
/* TODO: asel_capabilities (now fully disabled) */
```

#### *17.2.2.3 Station Side – Sending Association Request:*

Function - ieee80211_mgd_assoc:

After processing beacon frames, Station prepares association request frame which contains a sub-set of HT capabilities present in Beacon frame or HT is completely disabled:

HT is disabled in Association Request Frames for the following cases:

IEEE802.11n does not allow TKIP/WEP as pairwise ciphers in HT mode. Still associate in non-HT mode (11a/b/g) if any one of these ciphers is configured as pairwise.

Code snippet:

```
for (i = 0; i < req->crypto.n_ciphers_pairwise; i++)
{
   if (req->crypto.ciphers_pairwise[i] == WLAN_CIPHER_SUITE_WEP40 ||
       req->crypto.ciphers_pairwise[i] == WLAN_CIPHER_SUITE_TKIP ||
       req->crypto.ciphers_pairwise[i] == WLAN_CIPHER_SUITE_WEP104)
   {
         ifmgd->flags |= IEEE80211_STA_DISABLE_HT;
         ifmgd->flags |= IEEE80211_STA_DISABLE_VHT;
         netdev_info(sdata->dev, "disabling HT/VHT due to WEP/TKIP use\n");
   }
}
```

Disable HT if Station does not support it. Or AP does not use WMM.

Code snippet:

```
if (!sband->ht_cap.ht_supported ||
   local->hw.queues < IEEE80211_NUM_ACS || !bss->wmm_used ||
   ifmgd->flags & IEEE80211_STA_DISABLE_WMM)
{
    ifmgd->flags |= IEEE80211_STA_DISABLE_HT;
    if (!bss->wmm_used &&!(ifmgd->flags &
IEEE80211_STA_DISABLE_WMM))
        netdev_info(sdata->dev, "disabling HT as WMM/QoS is not
supported by the AP\n");
}
```

Supported Channnel Width: If 40Mhz is disabled, then associate as if Station is not capable of 40Mhz

#### 17.2.2.4 *AP Side: Processing Association request:*

If mandatory HT is configured, then reject association. (require_ht = 1)

Code snippet:

```
if (hapd->iconf->ieee80211n && hapd->iconf->require_ht &&
          !(sta->flags & WLAN_STA_HT)) {
               hostapd_logger(hapd, sta->addr,
HOSTAPD_MODULE_IEEE80211, HOSTAPD_LEVEL_INFO, "Station does not
support " "mandatory HT PHY - reject association");
       return WLAN_STATUS_ASSOC_DENIED_NO_HT;
```

AP updates the ht_capablities info of STA which tries to associate.

#### 17.2.2.5 *AP Side: Sending Association Response:*

AP sends the association response back to station.

### 17.2.3 Mapping of Management Frames Field against Configuration Parameters

Mapping of HT Capability Info Fields against Configuration Parameters:

| HT Capability Field | Configuration Parameter |
|---|---|
| LDPC Coding Capability | LDPC Capability |
| Supported Channel Width Set | Channel Width |
| SM Power Save | SM Power Save Mode |
| HT-Greenfield | GF Mode |
| Short GI for 20MHz | Guard Interval |
| Short GI for 40MHz | Guard Interval |
| Tx STBC | Tx-STBC |
| Rx STBC | Rx-STBC |
| HT-Delayed Block Ack | Delyaed BlockAck |
| Maximum A-MSDU Length | Max-AMSDU Length |
| DSSS/CCK Mode in 40 MHz | BSS 40MHz DSSS/CCK Mode |
| 40 MHz intolerant | 40 MHz intolerant |
| L-SIG TXOP Protection | <NA> |

**Mapping of A-MPDU Parameters Field against Configuration Parameters:**

| A-MPDU Parameters Field | Configuration Parameter |
|---|---|
| Maximum A-MPDU Length Exponent | Max-AMPDU Length |
| Minimum MPDU Start Spacing | <NA> |

**Mapping of Supported MCS Set field against Configuration Parameters:**

| Supported MCS Set Field | Configuration Parameter |
|---|---|
| The Tx MCS set is defined to be equal to the Rx MCS set | Physical Rate |
| The Tx MCS set may differ from the Rx MCS set | Physical Rate |

## 17.3 Data Frame Handling

### 17.3.1 Data Frame Handling in mac80211

### 17.3.2 Data Frame Handling in Meshap

# 11 IMCP Packet Changes

# 12 Coupling APIs changes for 802.11n and 802.11ac

Meshap has exported quite a few symbols, which are currently used by the ath5K drivers. The sections below list down the various symbols exported by meshap and how they will be used w.r.t. mac80211 code

## 12.1 meshap_get_board_temp

This function is unimplemented in meshap code. It returns 0 by default. Hence, this API will not be called from mac80211 code.

## 12.2 meshap_get_board_voltage

This function is unimplemented in meshap code. It returns 0 by default. Hence, this API will not be called from mac80211 code

## 12.3 meshap_set_led_on

This function calls the led_brightness_set api of linux stack with argument LED_FULL. Meshap will continue to use this API. In all the code paths where LED is set to ON, the same function will be called in the mac80211 code path also.

## 12.4 meshap_set_led_off

This function calls the led_brightness_set api of linux stack with argument LED_OFF. Meshap will continue to use this API. In all the code paths where LED is set to OFF, the same function will be called in the mac80211 code path also.

## 12.5 meshap_set_led_blink

This function calls the led_blink_set api of linux stack with the 'delay' argument specifying the frequency of blink as 1000. Meshap will continue to use this API. In all the code paths where LED is set to BLINK, the same function will be called in the mac80211 code path also.

### 12.6 meshap_set_led_blink_fast

This function calls the led_blink_set api of linux stack with the 'delay' argument specifying the frequency of blink as 200. Meshap will continue to use this API. In all the code paths where LED is set to BLINK, the same function will be called in the mac80211 code path also.

### 12.7 meshap_set_led_blink_once

This function calls the led_brightness_set api of linux stacks with argument as LED_OFF and modifies the timer to expire after 1000 hz. Meshap will continue to use this API. In all the code paths where LED is set to BLINK, the same function will be called in the mac80211 code path also.

### 12.8 meshap_enable_reset_generator

This function is not implemented in the meshap code and hence won't be called from mac80211 code as well.

### 12.9 meshap_strobe_reset_generator

This function is not implemented in meshap code and hence won't be called from mac80211 code as well.

### 12.10 meshap_get_gpio

This function is unimplemented in meshap code. It returns 0 by default. Hence, this API will not be called from mac80211 code.

### 12.11 meshap_set_gpio

This function is unimplemented in meshap code. It returns 0 by default. Hence, this API will not be called from mac80211 code.

### 12.12 meshap_get_gps_info

This function gets the gps location of meshap. Meshap will continue to use this API.

### 12.13 meshap_set_gps_info

This function sets the gps location of meshap. Meshap will continue to use this API.

### 12.14 meshap_process_mgmt_frame

This api is called by the atheros driver when it receives a management frame and is used to pass the frame to meshap for processing. With the mac80211 code, a hook will be added which will generate the copy of the frame and give the frame to meshap for processing. Meshap will then call meshap_core_process_mgmt_frame and process the management frames.

### 12.15 meshap_process_data_frame

This api is called by the atheros driver when it receives a data frame and is used to pass the frame to meshap for processing. With the mac80211 code, a hook will be added give the frame to meshap for processing. Meshap will then call meshap_core_process_data_frame and process the data frames.

### 12.16 meshap_on_link_notify

The api is called from meshap code from the following code paths

a. When a station gets associated / disassociated, then this meshap hook gets called in case of relay node processing for the uplink interface. This is because the uplink interface of the mesh node acts like a station.

b. Meshap registers with the netdev notifier to get updates about the state of the link. Any change in the state of the link is notified via meshap callback handler which ends up calling the above API.

c. In the previous code, the driver has a callback registered for on_phy_link_notify_watchdog.

### 12.17 meshap_on_net_device_create

This api is called when a mip0 device creation notification is received from linux. It creates the core_net_if_t structure when a mip0 device is registered (on receiving NETDEV_REGISTER event) or on receiving NETDEV_CHANGE event.

For wireless interfaces, this api is currently called by the atheros_attach api.

For integration with mac80211, meshap_on_net_device_create will be called for wireless interfaces on trigger from configd.

### 12.18 meshap_on_net_device_destroy

This api is called when a mip0 device is un-registered (on receiving NETDEV_UNREGISTER event). Meshap removes the entry for this device from core_net_if_t structure.

### 12.19 meshap_get_sta_info

Wireless Network drivers shall call this function to obtain information about a destination mac-address from the meshap LKM. The most common use of this function is in the "**hard_header**" handler for the **net_device**. The Kernel calls the **hard_header** handler, when sending packets from the stack directly through the network device.

With mac80211 integration, it is not required to call this API as the drivers get this information directly from mac80211.

### 12.20 meshap_reboot_machine

This function is called by the atheros_tx routine of meshap when the TX buffer pool overflows. In this case meshap needs to be restarted. For integration with mac80211, this would not be required.

### 12.21 torna_hw_id_get_address

This api is unimplemented in meshap. It is used to set the mac address of the device. The setting of mac addresses of the devices will be handled by standard wireless drivers.

### 12.22 torna_get_product_oui_id

This api is unimplemented in meshap and not called by meshap. . Hence, this API will not be called from mac80211 code.

### 12.23 torna_get_generic_id

This api is unimplemented in meshap and not called by meshap. Hence, this API will not be called from mac80211 code.

### 12.24 torna_put_reboot_info

This api is unimplemented in meshap. It is called from _meshap_panic_event and meshap_die_event APIs. . Hence, this API will not be called from mac80211 code

### 12.25 torna_get_reboot_info

This API populates the '"reboot"  proc entry.

### 12.26 Meshap hook functions registered with the driver

In the current code, meshap registers various hook functions with the drivers, which are then called by the driver explicitly to notify meshap of the various events. The sections below list down the various hook functions, their functionality in the current code and how the same functionality will be achieved with the mac80211

### 12.27 round_robin_hook

The hook is registered by meshap to the driver, so that the driver can inform meshap when it receives  beacons from the AP. This is set by the function _meshap_net_dev_set_round_robin_notify_hook().

### 12.28 probe_request_hook

The hook is registered by meshap to the driver, so that driver can inform meshap whenever the station sends the probe request to the AP. This will be required in the case of relay node on the uplink interface, where the interface acts like a station. It is set by the function

_meshap_net_dev_set_probe_request_notify_hook()

### 12.29 radar_hook

### 12.30 set_hw_addr

In the current implementation a vector for set_hw_addr is registered in the meshap_net_dev_t structure. This vector calls the ath_hal_setmac api to set the mac address in the device.

With the mac80211 code, each device registers with net_dev a function to set the mac address.

set_hw_addr will be modified to call the ndo_set_mac_address from the net_dev directly.

### 12.31 associate

In the current implementation, when the meshap wants to send the associate request to the parent, it calls ath5K routine to send the associate request.

***ret = atheros_sta_fsm_join(instance,essid,length,bssid,channel,ie_in,ie_in_length,ie_out);***

However, now since the intention is to make the code independent of the ath5k specific drivers, the code will be changed to call the cfg_80211 routines to generate the association request. During the boot time, drivers call the ieee80211 specific initializations where they register the ops specific  to mac80211. Hence, those ops will be called to handle send the association request from a particular interface.

***err = rdev->ops->assoc(&rdev->wiphy, dev, &req);***

## 12.32 dis_associate

In the current implementation, when the meshap wants to send the dis-associate request to the parent, it calls ath5K routine to send the dis-associate request

***ret = atheros_sta_fsm_leave(instance,1,WLAN_REASON_UNSPECIFIED);***

However, now since the intention is to make the code independent of the ath5k specific drivers, the code will be changed to call the cfg_80211 routines to generate the association request. During the boot time, drivers call the ieee80211 specific initializations where they register the ops specific to mac80211. Hence, those ops will be called to handle send the association request from a particular interface.

***rdev->ops->disassoc(&rdev->wiphy, dev, &req, wdev);***

## 12.33 get_bssid

In the current implementation, when the meshap wants to get the bssid, it copies same from the driver's instance structure associated with the wireless device.

The code used to do the following:-

memcpy(bssid,instance->current_bssid,ETH_ALEN);

Now the code, will be modified to get the value from the ieee80211_ptr associated with the netdev.

It will be fetched using the following:-

struct wireless_dev *wdev = dev->ieee80211_ptr;

memcpy(bssid, wdev->current_bss->pub.bssid, bssid, ETH_ALEN) ;

## 12.34 scan_access_points

In the current code, there is a function written for the scanning, the result of which is used for the determining the active parent for the relay node. In this code, the thread sleeps till the response for message sent arrives and the response is used for figuring out the results.

In mac80211, the scan code is quite different. Mac80211 provides various APIs to start the scanning and notify when the call-back gets completed. The scan code in Meshap needs to be changed to be able to get integrated with the mac80211 code.

## 12.35 scan_access_points_active

In the current code, there is a function written for the scanning, the result of which is used for the determining the active parent for the relay node. In this code, the thread sleeps till the response for message sent arrives and the response is used for figuring out the results.

In mac80211, the scan code is quite different. Mac80211 provides various APIs to start the scanning and notify when the call-back gets completed. The scan code in Meshap needs to be changed to be able to get integrated with the mac80211 code

## 12.36 scan_access_points_passive

In the current code, there is a function written for the scanning, the result of which is used for the determining the active parent for the relay node. In this code, the thread sleeps till the response for message sent arrives and the response is used for figuring out the results.

In mac80211, the scan code is quite different. Mac80211 provides various APIs to start the scanning and the notify them, when the call-back gets completed. The scan code in Meshap needs to be changed to be able to get integrated with the mac80211 code

### 12.37 get_last_beacon_time

In the existing code, this is implemented by fetching the value from the driver instance structure.

However, the “op” is not getting called from meshap. Hence, the implementation of this op will return 0.

#### 12.37.1 set_mesh_downlink_round_robin_time

This is used to set the configuration parameter, for round_robin_time. This is being used by the driver to set the time interval of sending message to various stations connected to the Access point.

Now, since the messages to the various stations will be triggered by hostapd, meshap won’t have any role to play for this. This will be stubbed out in the current meshap code.

### 12.38 add_downlink_round_robin_child

This is used by the meshap code, to add the station to the driver. This is done as part of the processing of the association messages received from the stations. In the current code, meshap is directly adding the child to the driver.

However, now informing about the child to the driver will be managed by hostapd and meshap doesn’t need to worry about it.

Hence, this function will be stubbed out.

### 12.39 remove_downlink_round_robin_child

This is used by the meshap code, to remove the station from the driver. This is done as part of the processing of dissociate message, or for any other reason meshap was to disassociate the child.

However, now informing about the child to the driver will be managed by hostapd and meshap doesn’t need to worry about it.

Hence, this function will be stubbed out.

### 12.40 virtual_associate

### 12.41 get_duty_cycle_info

### 12.42 set_rate_ctrl_parameters

### 12.43 reset_rate_ctrl

### 12.44 get_rate_ctrl_info

### 12.45 set_round_robin_notify_hook

### 12.46 enable_ds_verification_opertions

### 12.47 dfs_scan

### 12.48 set_radar_notify_hook

### 12.49 get_mode

In the current implementation, this function is used to fetch the mode from the instance structure. This is used by meshap during the init time.

With the mac80211 code, the implementation of the API, will be changed to fetch the value from the meshap's private structure stored in the ieee80211_hw structure.

### 12.50 set_mode

In the current implementation, this function is used to set the mode for each of the interfaces in the instance structure associated with the driver. This is done during the init time.

With the mac80211 code, the mode will be computed and set in the driver's `ieee80211_hw structure`

### 12.51 get_essid

This function returns the stored essid in the atheros_instance_t structure.When an association frame is received by meshap, it compares the receive ssid information element with the value returned by get_essid. If the ssid matches, association is allowed. Meshap will continue using this API.

### 12.52 set_essid

This function sets the ssid for the interface. This information is obtained from the configuration file and stored in the atheros_instance_t structure. Meshap will continue using this API.

### 12.53 get_rts_threshold

This function returns the rts threshold value which is present in the atheros_instance_t structure. Meshap uses this value while transmitting a packet.

For integration, mac80211 will transmit the packets, and takes care of rts threshold handling.

### 12.54 set_rts_threshold

This function sets the rts threshold value in the atheros_instance_t structure. Meshap will store this information.

### 12.55 get_frag_threshold

This function returns the fragmentation threshold value which is present in the atheros_instance_t structure. In the current implementation the _atheros_setup_packet function gets the frag threshold from the api _atheros_get_fragment_size. Meshap then fragments the packet.

### 12.56 set_frag_threshold

This function stores the value of fragmentation threshold in the atheros_instance_t structure of meshap. This function is called while applying the configuration at startup or in the IMCP message handling.

In mac80211, If the device does the fragmentation itself then “set_frag_threshold” defined by the driver will be called by mac80211 to do fragmentation else mac80211 will itself fragment the packets.

For integration, meshap can send the packet to mac80211 and mac80211 will take care of fragmenting the packet if required.

### 12.57 get_beacon_interval

Meshap never handles the probe response and it never sends the beacon interval. This information is maintained by meshap in the core_net_if structure.

### 12.58 set_beacon_interval

Meshap stores this information in core_net_if structure. Meshap will not send any beacons.

### 12.59 get_default_capabilities

On startup, meshap sets the default capabilities in the “default_capability” field of atheros_instance_t structure. The capability information is used in beacon transmissions to advertise the network's capabilities. Capability Information is also used in Probe Request and Probe Response frames.

Since meshap is not sending beacons, capability information is not used by it.

### 12.60 get_capabilities

On startup, meshap sets the default capabilities in the “capability” field of atheros_instance_t structure. The capability information is used in beacon transmissions to advertise the network's capabilities. Capability Information is also used in Probe Request and Probe Response frames.

Since meshap is not sending beacons, capability information is not used by it.

### 12.61 get_slot_time_type

This field is the part of the capability field. This is only used for beacon frames.

### 12.62 set_slot_time_type

This field is the part of the capability field. This is only used for beacon frames.

### 12.63 get_erp_info

This api returns the erp value. This is not being called in the current code.

### 12.64 set_erp_info

This field is the Effective Radiated power. This field is set only for WLAN_PHY_MODE_802_11_G and WLAN_PHY_MODE_802_11_PURE_G. When meshap receives a beacon, it processes it and gets this information from received beacon. It then compares this information with the stored in the extended_rate_phy_info field of the instance structure. If the value is changed, meshap updates extended_rate_phy_info field the value received in the beacon frame.

After integration, the beacon frames will be received by hostapd and meshap and meshap will handle the erp info . Meshap will not respond to this beacon.

### 12.65 set_beacon_vendor_info

This parameter is set by meshap which is sent in beacon frame. If the node is a root node, then the vendor id used is the DS mac of the root node, if the node is relay node, the vendor id used is parent's bssid. Hostapd will configure this value in the parameter “vendor_elements”

### 12.66 enable_wep

This is used by meshap to set the flag field in instance structure with value ATHEROS_INSTANCE_FLAGS_WEP_ENABLE and if this value is set then meshap updates the 'capability' field of instance structure with WLAN_CAPABILITY_PRIVACY flag.

This field is applicable in the context of sending beacon frames, and since meshap will not send out beacons, it will not be required while integrating with mac80211.

### 12.67 disable_wep

This is used by meshap to clear the flag field in instance structure with value ATHEROS_INSTANCE_FLAGS_WEP_ENABLE.

### 12.68 set_rsn_ie

This is used by meshap to set the flag field in instance structure with value ATHEROS_INSTANCE_FLAGS_RSN_IE_ENABLE and if this value is set then meshap updates the 'capability' field of instance structure with WLAN_CAPABILITY_PRIVACY flag.

This field is application in the context of sending beacon frames, and since meshap will not send out beacons, it will not be required while integrating with mac80211.

### 12.69 set_security_key

This is configured at init time. It will be configured using hostapd/iwconfig. Meshap uses this to get the key during the init time.

Since, initializations will be taken over by the mac80211/hostaps, since meshap is not supposed to configure this.

However, when this routine is called meshap will store the key in its local structure to be able to use the same in processing later on.

### 12.70 release_security_key

In the current meshap code, it is called when the key needs to be released from the driver. Now, this functionality should be handled by mac80211 and/or hostapd. Meshap shouldn't have any role to play in this.

However, when this routine is called meshap will remove the key info from its local structure.

### 12.71 get_security_key_data

In the current meshap code, it is called in the processing of the auth frames.

Now, this functionality will be handled by mac80211 and/or hostapd. However, since meshap will also receive the auth frames, process it and send the auth response, it will do so using the key info stored in the local structure.

However, the auth frames, will be blocked in the end from being transmitted.

The implementation will fetch the values from the local structures.

### 12.72 set_ds_security_key

In the current meshap code, it is called on the relay node to set the encryption key based on the data received from the parent.

With the current code, the mac80211 APIs will be called directly to set the key in the driver.

### 12.73 get_supported_rates

Meshap initializes the supported rates during its initialization. Meshap uses this value to setup beacon and respond to probe response contents. Meshap also receives this value in probe req and response messages.

Since, the response of the messages will be sent by mac80211 and not meshap, meshap doesn't need this api.

At the same time, every driver sets up this value at the init time. Hence, the data can be fetched if required.

### 12.74 get_extended_rates

Meshap initialized the extended rates during initialization. It then puts this value in the association response. Meshap will not send the management response so no mac80211 api is required. Meshap will just not transmit the association response..

### 12.75 get_bit_rate

Meshap sends the bit rate information in the heartbeat IMCP message. Meshap needs to retrieve this information from its database

### 12.76 set_bit_rate

This is the txrate parameter of each WLAN interface defined by the meshap.conf file. This is a configuration parameter and set via hostapd or iwconfig.

### 12.77 get_rate_table

### 12.78 get_tx_power

Meshap is not calling this.

### 12.79 set_tx_power

On transmission, meshap sets the TX power in the driver. It can be configured using iwconfig when needed

### 12.80 get_channel_count

This api is not being called by meshap.

### 12.81 get_channel

This function gets the current operating channel on an interface. Meshap can call rdev->ops->get_channel which maps to ieee80211_wiphy_get_channel. This will return the current operating channel.

### 12.82 set_channel

This function sets the current operating channel on an interface. Meshap can call rdev->ops->set_channel which maps to ieee80211_set_channel. This will set the current operating channel.

### 12.83 get_phy_mode

This function returns the current phy mode of the device. When the phy mode is 80211G or 80211BG. Meshap uses this information to set the preamble time and erp info for this phy mode.

### 12.84 set_phy_mode

This is being set during the init time to configure the drivers appropriately. However, now the drivers should get set using hostapd/mac80211. Hence, meshap is not supposed to set these fields in the driver.

However, meshap will maintain a local structure and then store the information in the local structure for reference in various scenarios.

### 12.85 get_preamble_type

This is not getting called in the current meshap code. However, we will maintain the value in a local structure in cre_net_if. The value can be fetched using that.

### 12.86 set_preamble_type

In the current meshap code, this is being called during the init time and the value is being set directly into the Atheros structure. Since, this is the part of initialization, the mac80211/hostapd code takes care of setting the appropriate values in the driver. Hence, from the meshap perspective it doesn't require it. However, still the value will be stored in a local structure maintained in core_net_if.

### 12.87 set_dev_token

In the current meshap code, a token is being set into the instance structure of the driver. This structure is used to fetch the local representation of dev from the instance and get all the information quickly.

The token will be set in the ieee80211_hw structure.

### 12.88 set_essid_info

In the current implementation a vector for set_ssid_info is registered in the meshap_net_dev_t structure. The information is later used by the driver to respond to the beacon requests.

However, with the mac80211 code, the beacons will be handled by mac80211 code and meshap will not sending them out. Hence, this function doesn't have much used. However, we will store the information in the device specific structure, so that if any need arises, meshap can easily fetch the value.

### 12.89 get_ack_timeout

In the current implementation meshap does not call this api . Hence, this will be stubbed out.

### 12.90 set_ack_timeout

In the current implementation a vector for set_ack_timeout is registered in the meshap_net_dev_t structure. This vector calls the ath_hal_setacktimeout api to set the ack timeout in the device.

With the mac80211 code, set_ack_timeout will be modified to call the set_coverage_class to set the ack timeout value.

### 12.91 get_reg_domain_info

In the current meshap implementation this is not being called.

### 12.92 set_reg_domain_info

It sets various parameters in the meshap, it is being set during init time.

Need to check whether these parameters can be configured with hostapd and iwconfig n it is being set during init time.

### 12.93 get_supported_channels

In the current implementation get_supported_channels is used to get the supported channel based on phy_mode, It is being called for mesh_imcp_send_packet_supported_channels_info to send the supported channel information.

### 12.94 get_hide_ssid

This is not being called in meshap design.

### 12.95 set_hide_ssid

In the current implementation set_hide_ssid is used to set the hide_ssid is enable or disable into the device. Now the parameter will be set through hostapd and this is moved out from meshap.

In the mac code set_hide_ssid can be done through calling **ieee80211_start_ap(struct wiphy *wiphy, struct net_device *dev, struct cfg80211_ap_settings *params)**

### 12.96 set_dot11e_category_info

In the current implementation set_dot11e_category_info is used to set various parameters such as category, acwmin, acwmax, aifsn, disable_backoff and burst_time.

Now this is being moved out from mesh and set it through hostapd.

### 12.97 set_tx_antenna

In the current implementation set_tx_antenna is set during init time and this is now moved out of meshap and set it through hostapd.

Not able to get the exact parameter in hostapd .

### 12.98 set_radio_data

Not being called in meshap

### 12.99 radio_diagnostic_command

### 12.100 set_probe_request_notify_hook

This is a hook function which meshap will register with mac80211 and mac80211 code will be modified to call this hook whenever probe request will generate.

### 12.101 set_virtual_mode

This is meshap specific function used to set Master and Infra mode for the virtual interface. Now, we will create virtual interfaces explicitly and enable hostapd/mac80211 to run on the virtual interface itself. Hence, from the mac80211 perspective, it doesn't require the virtual mode.

However, from meshap perspective the current processing will remain. The mode for the interface will be set in the core_net_if structure locally.

### 12.102 set_device_type

In the current meshap code, this is used to set the device type as virtual and the device_mode as MIXED. Since, for the mac80211 code, there will be separate interfaces created and it handles the virtual interfaces, there is no need to set the device type.

However, for the meshap purposes, device_type is required to be set and will be used to determine the interface on which the packet arrived. The device-type and mode will be stored directly in the core_net_if structure, instead of the instance structure as happening currently.

### 12.103 get_device_type

In the current implementation, this is used to get the device type associated with the device on which the packet arrived. Based on the type of device, the physical or the virtual core_net_if is used for the packet processing.

The new implementation will fetch the value from the core_net_if structure and let the callers of the functions take appropriate decisions.

### 12.104 initialize_mixed_mode

In the current implementation, this is used to set the mode in the Atheros driver and reinitialise the driver.

However, with mac80211, this will be managed via hostapd/mac80211 directly and meshap doesn't need to do anything about this. Hence, the implementation of this function will be stubbed out.

### 12.105 enable_beaconing_uplink

In the current meshap code, through the imcp messages, the uplinks can be enabled to send beacons. This is something which is special to meshap and will be handled specifically for meshap. Hence, in the mac80211 code, the changes will be done to allow the beacons coming from uplink to be forwarded to meshap for processing. However, mac80211 won't process those beacons.

### 12.106 disable_beaconing_uplink

In the current meshap code, through the imcp messages, the uplinks can be disabled to send beacons. This is something which is special to meshap and will be handled specifically for meshap.

### 12.107 set_action_hook

The set_action_hook is used to process action frames in meshap, so will register set_action_hook with mac80211 to handle these frames.

### 12.108 send_action

The meshap will call Send_action hook to transmit the action frames. The current implementation of the send_action hook calls the atheros driver routine to transmit the frame. Now the implementation will be modified to call the mac80211 transmit routines.

### 12.109 `set_ht_cababilites`

HT and VHT configurations are defined in meshap.conf file. On startup, the 'read_mesh_configuration' function will read the HT and VHT parameters from the meshap.conf file and update it to the global `mesh_config` structure.

```
struct ht_element_id
   {
      Uint16_t ht_capa;
      Uint8_t ampdu;
      Uint8_t supported_mcs_set[16];
       .

       .
```

}

Coupling function "`set_ht_cababilites`" is called to validate and verify if the driver supports the configured configurations. The ht_element_id will be passed as reference.

Psuedo code for coupling function

```
void set_ht_cababilites(struct ht_capab *);
{
    1>Call the wrapper to fetch the driver capable
configurations Validate against the configured in meshap.conf
        2>                                          If success,
          update the instance & return, else return error to
          network viewer
}

void meshap_set_vht_parameters(struct vht_capab *);
{
  /*Call the coupling function to verify if the requested
         configuration is supported by the driver*/
  core_net_if->dev_info->set_vht_cababilites(vht_capab);

}
```

Similarly, the coupling functions to set the other HT Capabilities IE – HT Operations, 20/40 BSS Co-existence, Overlapping BSS Scan will be there to verify and update the corresponding parameters in the meshap_instance_t structure.

This coupling function will get the HT/VHT capabilities from the driver using and compares the input configuration. If input configuration is not supported by the driver, then return error.

```
if ((ht_capab& HT_CAP_INFO_LDPC_CODING_CAP) &&
      !(hw & HT_CAP_INFO_LDPC_CODING_CAP)) {
          printk("Driver does not support configured HT
                 capability [LDPC]");
          return -1;
```

}

When all the configurations are verified, the coupling function stores the configuration in the meshap_instance_t structure and returns SUCCESS.

Starting Access point: Setting up Beacon frame.

When Access point initializes, it sets up the beacon buffer by calling '_meshap_init_beacon_buffer(instance)' function.

While creating the beacon frame, Meshap shall add the HT/VHT Elements in the beacon frame by directly getting the HT/VHT values from the meshap_instance_t structure which was stored as shown in the previous step.

### 12.110 meshap_sta_fsm_get_ht_override

On station side, the meshap STA FSM is running. At first, the _process_beacon function is called to get the list of available APs and find the best parent.

The _process_beacon function has to get the HT and VHT Element ids if present and store it in the parent structure.

HT/VHT paraments will also be considered to select best parent selection. Eg If station is 11n capable, then it shall prefer an 11n AP.

STA FSM joins the best parent by invoking the association request by calling the cfg80211 operation rdev->ops->assoc(wdev->wiphy, instance->dev, &req);

This callback function internally calls ieee80211_mgd_assoc function which takes care of deriving the sub-set of supported HT/VHT capabilities with which the station shall associate with the AP. This sub-set is stored in the global structure of struct ieee80211_if_managed *ifmgd.

Coupling function "meshap_sta_fsm_get_ht_override" is called by STA FSM is assoc returns success. This coupling will get the global ifmgd flags set by mac80211 and store it in the meshap_instance_t structure of station interface.

```
Meshap_sta_fsm_get_ht_overide(meshap_instance_t *instance)
{
     Call the wrapper function to get the ifmgd flag
     Update the instance->ht_capab structure with the returned
     values of ifmgd->ht_capa
}
```

1. Access Point receives the Association Request from Station, and sends Assoc Response. AP processes the association request in al_process_assoc, and calls the `meshap_update_sta_ht_state` function to update the ht_capabilities of the STA.

```
void meshap_update_sta_ht_state(access_point_sta_entry_t *sta, struct
ht_capab*)
{
       if ((ht_capab & HT_CAP_INFO_SUPP_CHANNEL_WIDTH_SET) == 0)
       {
            if (!sta->ht_20mhz_set)
               sta->ht_20mhz_set = 1;
       }
       if ((ht_capab & HT_CAP_INFO_GREEN_FIELD) == 0) {
                  if (!sta->no_ht_gf_set) {
                           sta->no_ht_gf_set = 1;
            }
       }
      if (ht_capab & HT_CAP_INFO_40MHZ_INTOLERANT)
            ht40_intolerant_add(hapd->iface, sta);
}
```

# 13 QoS Handling

# 14 900MHz radio devices handling

# 15 Single Radio Model support

## 15.1 Support for virtual Interfaces

# 16 900MHz USB support