RW-WLAN-nX FMAC SW

User Manual

RW-WLAN-nX-FMAC-SW-UM/0.05

Version 0.05

2016-08-03

RivieraWaves
http://www.rivierawaves.com/

Revision History

Version	Date	Revision Description	Author
0.01	2015-02-16	Initial Release	Steven L'Her
0.02	2015-03-10	Added APM messages	Steven L'Her
0.03	2015-05-26	Various updates	Steven L'Her
0.04	2016-03-31	Added new API messages in ME and APM Updated parameter structures	Steven L'Her
0.05	2016-08-03	Added MESH messages	Laurent Trarieux

Changes between a version and the previous one is reflected by the addition of **change bars**, like for the line below:

TBD	Date entered	Description	Status

Items to be determined in the future versions of this document

Table of Contents

List of Figures

List of Tables

1 Overview

1.1 Document overview

1.1.1 Purpose

The purpose of this document is to describe the functionality and the API of the RW-WLAN-nX FMAC SW, as well as guidelines of usage of this API to put the system in the required mode. This document is addressed to both upper MAC developers and any person wishing to have an overview of the FMAC functionality. Note that a part of the API of the FMAC SW is described in the LMAC SW User Manual.

1.1.2 Abbreviations and Acronyms

AC	Access Category
ACK	Acknowledgment
ACM	Access Category Mandatory
AID	Association Identifier
AP	Access Point
APSD	Automatic Power Save Delivery
A-MPDU	Aggregate MAC Protocol Data Unit
A-MSDU	Aggregate MAC Service Data Unit
BA	Block Acknowledgement
BAR	Block Acknowledgement Request
BSSID	Basic Service Set Identifier
CF	Contention Free
CSI	Carrier State Information
CTS	Clear To Send
CW	Contention Window
DCF	Distributed Coordination Function
EDCA	Enhanced Distributed Channel Access
FIFO	First-In-First-Out
FC	Frame Control
FCS	Frame Check Sequence
HCCA	HCF Controlled Channel Access
HT	High Throughput
IBSS	Independent Basic Service Set
ICV	Integrity Check Value
IV	Initialization Vector
LLC	Logical Link Control
L-SIG	Legacy (Non-HT) Signal Field
MAC	Medium Access Control

MBSS	Mesh Basic Service Set
MEM-BAR	Memory Base Address Register
MIB	Management Information Base
MIMO	Multiple Input Multiple Output
MLME	MAC Sub Layer Management Entity
MMPDU	MAC Management Protocol Data Unit
MP	Mesh Point
MPDU	MAC Protocol Data Unit
MPM	Mesh Peering Management Protocol
MSDU	MAC service data unit
NAV	Network Allocation Vector
OS	Operating System
PC	Point Coordinator
PHY	Physical layer
PLME	PHY Sub Layer Management Entity
PS	Power Save
PSMP	Power Save Multi-Poll
QAP	QoS Access Point
QoS	Quality of Service
QSTA	QoS Station
RD	Reverse Direction
RDG	Reverse Direction Grant
RSNA	Robust Security Network Association
RTS	Request to Send
RX	Receiver
SSID	Service Set Identifier
STA	Station
STBC	Space-Time Block Coding
TID	Traffic Identifier
TIM	Traffic Indication Map
TS	Traffic Stream
TU	Time Unit
TX	Transmitter
TX_REQ	Transmit Request message received from UMAC-SW
U-APSD	Unscheduled Automatic Power Save Delivery
WEP	Wired Equivalent Privacy

WLAN	Wireless LAN
WM	Wireless Medium

Table 1: Abbreviations and Acronyms

2 FMAC SW Overview

This section describes major components of FMAC SW and its interfaces with other layers in the embedded system. The figure below shows the different blocks of the FMAC SW.

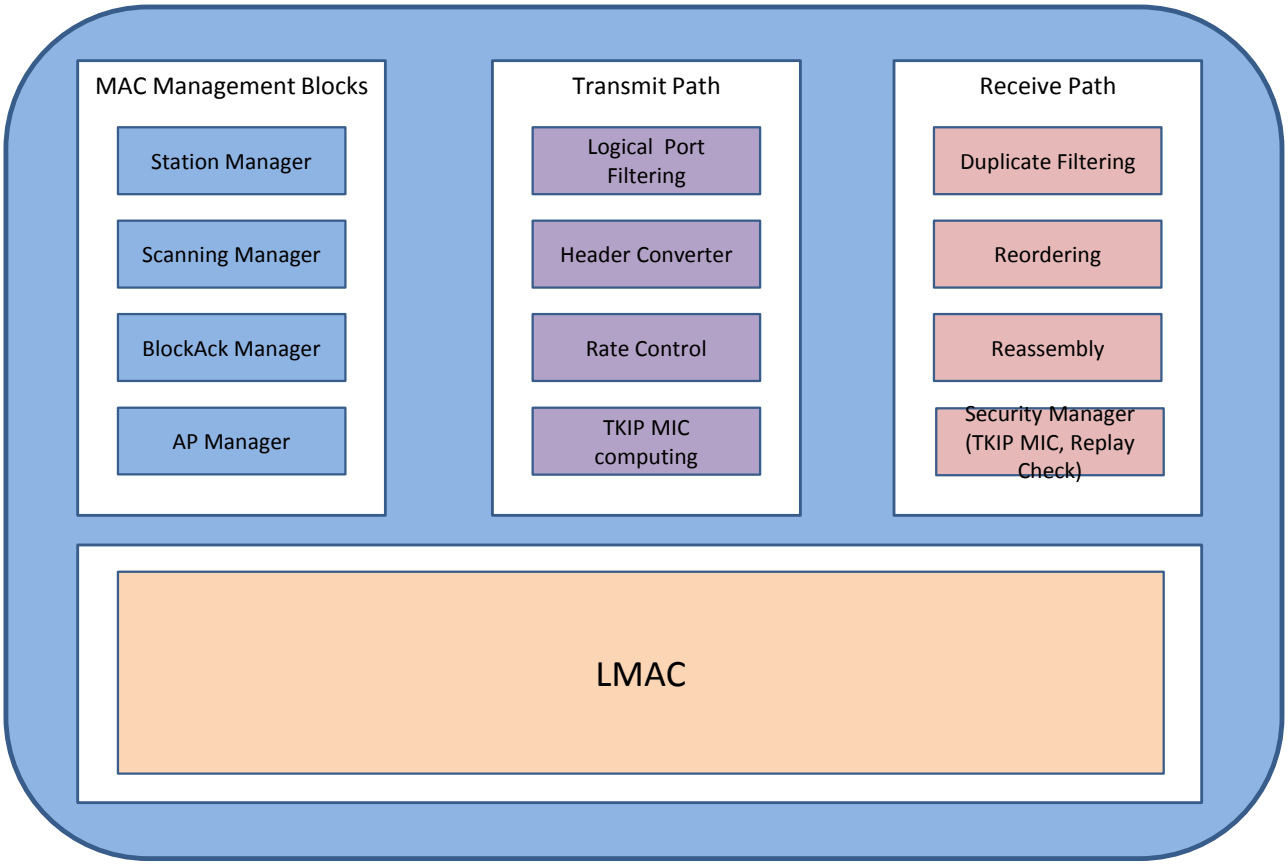

Figure 1: FMAC SW block diagram

3 API description

3.1 Configuration interface

This interface is used for the configuration of the FMAC SW. The procedures that can be started using this API include:

- Scanning for peer devices
- Connecting to Access Points
- Starting an Access Point
- Starting a Mesh Point

This interface is controlled via messages that are sent/received to/from the FMAC SW. These messages have to be addressed to the correct module of the FMAC, using a unique 16-bit identifier (see Table 2).

LMAC Module	Identifier
Scanning Manager (SCANU)	0x0003
MAC Entity Manager (ME)	0x0004
Station Manager (SM)	0x0005
AP Manager (APM)	0x0006
MESH Manager (MESH)	0x0008

Table 2: LMAC module identifiers

A naming convention is adopted, in order to easily differentiate the direction of each message:

- xxxx_REQ: Configuration request. It is sent from upper MAC to FMAC.
- xxxx_CFM: Confirmation of a previous configuration request. It is sent from FMAC to upper MAC.
- yyyy_IND: Asynchronous indication sent from FMAC to upper MAC.

The general format of such messages is given in the below section.

3.1.1 Control Message format

A control message has to follow the following format:

name	msg_id	dest_id	src_id	param_len	parameters
size	2	2	2	2	0 - n

Figure 2: Control Message Format

The fields of a control message are the following:

- msg_id: Unique identifier of the message (see below chapters for details about this field)
- dest_id: Destination of the message (see Table 2)
- src_id: Reserved for internal LMAC use. Has to be put to 0x64 in all messages.
- param_len: Length, in bytes, of the parameters field (0 if no parameters)
- parameters: Parameters of the message, if any. This field varies depending on the value of msg_id.

3.1.2 MAC Entity Manager API description

3.1.2.1 Overview

Table 3 gives an overview of the requests/confirmations composing the MM API:

Request	Confirmation	Description
ME_CONFIG_REQ	ME_CONFIG_CFM	Configure the HT/VHT capabilities of the device
ME_CHAN_CONFIG_REQ	ME_CHAN_CONFIG_CFM	Configure the channel list available for scanning and connection
ME_SET_CONTROL_PORT_REQ	ME_SET_CONTROL_PORT_CFM	Open/Close the control port for a given station
ME_STA_ADD_REQ	ME_STA_ADD_CFM	Add a station to the list of known devices
ME_STA_DEL_REQ	ME_STA_DEL_CFM	Delete a station from the list of known devices
ME_UAPSD_TRAFFIC_IND_REQ	ME_UAPSD_TRAFFIC_IND_CFM	Indicate to the FW that there is UAPSD traffic buffered on host
ME_RC_STATS_REQ	ME_RC_STATS_CFM	Get Rate Control statistics for the given station
ME_RC_SET_RATE_REQ	n/a	Set a fixed rate for the transmissions to the given station

Table 3: MAC Entity Control messages

Some indications can also be received asynchronously from the ME:

Indication	Description
ME_TKIP_MIC_FAILURE_IND	Reports a TKIP MIC failure
ME_TX_CREDITS_UPDATE_IND	Indicates a modification in the number of credits allocated to a RA/TID

Table 4: MAC Entity Indication messages

3.1.2.2 Detailed description

3.1.2.2.1 ME_CONFIG_REQ

This request configures the HT and VHT parameters of the FMAC SW.

Msg_id	0x1000
Confirmation message	ME_CONFIG_CFM

Table 5: ME_CONFIG_REQ description

3.1.2.2.1.1 Parameters

Name	Type	Description

ht_cap	struct mac_htcapability	HT capabilities of the device. The format of this structure is defined in the mac.h file.
vht_cap	struct mac_vhtcapability	VHT capabilities of the device. The format of this structure is defined in the mac.h.
ht_supp	u8	0: HT is not supported and the ht_cap parameter is reserved 1: HT is supported and the ht_cap parameter is valid
vht_supp	u8	0: VHT is not supported and the vht_cap parameter is reserved 1: VHT is supported and the vht_cap parameter is valid
ps_on	u8	0: Disallow Power Save mode 1: Allow Power Save mode

Table 6: ME_CONFIG_REQ parameters

3.1.2.2.2 ME_CONFIG_CFM

This message confirms the completion of the handling of ME_CONFIG_REQ message.

Msg_id	0x1001
Confirmation message	n/a

Table 7: ME_CONFIG_CFM description

3.1.2.2.3 ME_CHAN_CONFIG_REQ

This request sets the list of channels that are supported by the device.

msg_id	0x1002
Confirmation message	ME_CHAN_CONFIG_CFM

Table 8: ME_CHAN_CONFIG_REQ description

3.1.2.2.3.1 Parameters

Name	Type	Description
chan2G4	struct scan_chan_tag[14]	List of the supported 2.4GHz band channels. The format of this structure is defined in file scan.h.
chan5G	struct scan_chan_tag[28]	List of the supported 5GHz band channels. The format of this structure is defined in file scan.h.
chan2G4_cnt	u8	Number of 2.4GHz channels available in the list
chan5G_cnt	u8	Number of 5GHz channels available in the list

Table 9: ME_CHAN_CONFIG_REQ parameters

3.1.2.2.4 ME_CHAN_CONFIG_CFM

This message confirms the completion of the handling of ME_CHAN_CONFIG_REQ message.

Msg_id	0x1003
Confirmation message	n/a

Table 10: ME_CHAN_CONFIG_CFM description

3.1.2.2.5 ME_SET_CONTROL_PORT_REQ

This message opens or closes the control port for the given station. It has to be sent by the upper layers once the security keys have been set to the LMAC to allow the data to be transmitted/received to/from this peer station.

Msg_id	0x1004
Confirmation message	ME_SET_CONTROL_PORT_CFM

Table 11: ME_SET_CONTROL_PORT_REQ description

3.1.2.2.5.1 Parameters

Name	Type	Description
sta_idx	u8	Index of the station for which the control port has to be modified.
control_port_open	u8	0: Control port is closed 1: Control port is opened

Table 12: ME_SET_CONTROL_PORT_REQ parameters

3.1.2.2.6 ME_SET_CONTROL_PORT_CFM

This message confirms the completion of the handling of ME_SET_CONTROL_PORT_REQ message.

Msg_id	0x1005
Confirmation message	n/a

Table 13: ME_SET_CONTROL_PORT_CFM description

3.1.2.2.7 ME_TKIP_MIC_FAILURE_IND

This message indicates that TKIP MIC failure has been detected when receiving a packet from the peer station passed as parameter. The upper layers (typically WPA Supplicant) should then trigger the TKIP counter measure procedures.

msg_id	0x1006

Confirmation message	n/a

Table 14: ME_TKIP_MIC_FAILURE_IND description

3.1.2.2.7.1 Parameters

Name	Type	Description
addr	u8[6]	MAC address of the sending STA
tsc	u64	TSC of the received packet
ga	u8	0: The received frame is unicast frame 1: The received frame is a group addressed frame
keyid	u8	KeyId of the key that was used to compute the received MIC
vif_idx	u8	Index of the VIF this peer STA is attached to

Table 15: ME_TKIP_MIC_FAILURE_IND parameters

3.1.2.2.8 ME_STA_ADD_REQ

This message is sent by the host to add a new station to the FMAC list of known devices. This operation will then allow the transmission and reception to/from this STA. It is typically called in AP mode, upon a new station association to the AP.

Msg_id	0x1007
Confirmation message	ME_STA_ADD_CFM

Table 16: ME_STA_ADD_REQ description

3.1.2.2.8.1 Parameters

Name	Type	Description
mac_addr	u8[6]	MAC address of the new STA.
rate_set	struct mac_rateset	Legacy rate set supported by the new STA.
ht_cap	struct mac_htcapability	HT capabilities of the new STA (valid only if the *flags* parameter indicates HT support).
Vht_cap	struct mac_vhtcapability	VHT capabilities of the new STA (valid only if the *flags* parameter indicates VHT support).
Flags	u32	Bitfield containing additional information about the new STA: bit0: When set, it indicates that the STA supports the QoS. bit1: When set, it indicates that the STA is HT capable. bit2: When set, it indicates that the STA is VHT capable.
aid	u16	AssociationID that was given to the new STA by the host application driving the AP (typically hostapd).

uapsd_queues	u8	Bitfield indicating which AC queues are subject to UAPSD for this STA: bit0: When set, it indicates UAPSD for VOICE queue bit1: When set, it indicates UAPSD for VIDEO queue bit2: When set, it indicates UAPSD for BACKGROUND queue bit3: When set, it indicates UAPSD for BEST-EFFORT queue
vif_idx	u8	Index of the VIF the new STA is attached to.

Table 17: ME_STA_ADD_REQ parameters

3.1.2.2.9 ME_STA_ADD_CFM

This message confirms the addition of a new STA upon the ME_STA_ADD_REQ message reception.

Msg_id	0x1008
Confirmation message	n/a

Table 18: ME_STA_ADD_CFM description

3.1.2.2.9.1 Parameters

Name	Type	Description
sta_idx	u8	Index of the STA given by the FMAC SW. This index will then be used for the frame transmissions and other procedures.
Status	u8	Status indicating whether the STA has been correctly added or not: 0x00: The STA has been successfully added. 0x01: The STA was not added due to insufficient resources.
pm_state	u8	Indicates whether the station has been detected as going to PS prior to the reception of this message. 0: The station can be considered as active 1: The station has indicated it is in PS mode

Table 19: ME_STA_ADD_CFM parameters

3.1.2.2.10 ME_STA_DEL_REQ

This message is sent by the host to delete a station from the FMAC list of known devices. This operation is typically used in AP mode, upon a station disassociation from the AP.

Msg_id	0x1009
Confirmation message	ME_STA_DEL_CFM

Table 20: ME_STA_DEL_REQ description

3.1.2.2.10.1 Parameters

Name	Type	Description
sta_idx	u8	Index of the STA that needs to be deleted.

Table 21: ME_STA_DEL_REQ parameters

3.1.2.2.11 ME_STA_DEL_CFM

This message confirms the deletion of the STA upon the ME_STA_DEL_REQ message reception.

Msg_id	0x100A
Confirmation message	n/a

Table 22: ME_STA_DEL_CFM description

3.1.2.2.12 ME_TX_CREDITS_UPDATE_IND

This message indicates that the credits allocated to the specified RA/TID needs to be updated with the value passed as parameter. It is typically sent by the FMAC SW upon a BlockAck agreement addition/deletion.

msg_id	0x100B
Confirmation message	n/a

Table 23: ME_TX_CREDITS_UPDATE_IND description

3.1.2.2.12.1 Parameters

Name	Type	Description
sta_idx	u8	Index of the STA for which the credits are updated.
tid	u8	TID for which the credits are updated.
credits	s8	Offset to be applied to the TX credit count for this STA/TID pair. This offset can be negative.

Table 24: ME_TX_CREDITS_UPDATE_IND parameters

3.1.2.2.13 ME_UAPSD_TRAFFIC_IND_REQ

This message is sent by the host to indicate to the FMAC FW the availability or unavailability of UAPSD buffered traffic on host. Based on this information the FW knows how to react upon UAPSD trigger frame reception.

Msg_id	0x100C
Confirmation message	ME_UAPSD_TRAFFIC_IND_CFM

Table 25: ME_UAPSD_TRAFFIC_IND_REQ description

3.1.2.2.13.1 Parameters

Name	Type	Description

sta_idx	u8	Index of the STA for which traffic is available or unavailable.
tx_avail	u8	Flag indicating if traffic is available or not: 0: No traffic available on host. The FW shall simply replies with QoS-NULL frames with EOSP bit set upon reception of UAPSD trigger frames from the peer device. 1: Buffered traffic available on host. The FW shall request this traffic using the MM_TRAFFIC_REQ_IND message upon reception of UAPSD trigger frames.

Table 26: ME_UAPSD_TRAFFIC_IND_REQ parameters

3.1.2.2.14 ME_UAPSD_TRAFFIC_IND_CFM

This message confirms the execution of the ME_UAPSD_TRAFFIC_IND_REQ message reception.

Msg_id	0x100D
Confirmation message	n/a

Table 27: ME_UAPSD_TRAFFIC_IND_CFM description

3.1.2.2.15 ME_RC_STATS_REQ

This message allows getting Rate Control statistics for the station given as parameter.

Msg_id	0x100E
Confirmation message	ME_RC_STATS_CFM

Table 28: ME_RC_STATS_REQ description

3.1.2.2.15.1 Parameters

Name	Type	Description
sta_idx	u8	Index of the STA for which RC statistics are requested.

Table 29: ME_RC_STATS_REQ parameters

3.1.2.2.16 ME_RC_STATS_CFM

This message confirms the addition of a new STA upon the ME_STA_ADD_REQ message reception.

Msg_id	0x100F
Confirmation message	n/a

Table 30: ME_RC_STATS_CFM description

3.1.2.2.16.1 Parameters

Name	Type	Description

sta_idx	u8	Index of the STA given by the FMAC SW. This index will then be used for the frame transmissions and other procedures.
no_samples	u16	Number of samples used in the RC algorithm
ampdu_len	u32	Number of MPDUs transmitted (per sampling interval)
ampdu_packets	u32	Number of AMPDUs transmitted (per sampling interval)
avg_ampdu_len	u32	Average number of MPDUs in each AMPDU frame (EWMA)
sw_retry_step	u8	Current step 0 of the retry chain
sample_wait	u8	Trial transmission period
retry	struct step[4]	Retry chain steps
rate_stats	struct rc_rate_stats[10]	RC statistics for each rate currently in the table
tp	u32[10]	User throughput for each rate currently in the table

Table 31: ME_RC_STATS_CFM parameters

3.1.2.2.17 ME_RC_SET_RATE_REQ

This message is sent by the host assign fixed TX rate to the given station.

Msg_id	0x1010
Confirmation message	n/a

Table 32: ME_RC_SET_RATE_REQ description

3.1.2.2.17.1 Parameters

Name	Type	Description
sta_idx	u8	Index of the STA the rate of which needs to be fixed.
fixed_rate_cfg	u16	Index of the rate to be used

Table 33: ME_RC_SET_RATE_REQ parameters

3.1.3 Scan Manager API description

3.1.3.1 Overview

Table 3 gives an overview of the requests/confirmations composing the SCANU API:

Request	Confirmation	Description
SCANU_START_REQ	SCANU_START_CFM	Start a scanning procedure on a given set of channels.

Table 34: Scan Manager Control messages

Some indications can also be received asynchronously from the SCAN Manager:

Indication	Description
SCANU_RESULT_IND	Indication of a scanning result

Table 35: Scan Manager Indication messages

3.1.3.2 Detailed description

3.1.3.2.1 SCANU_START_REQ

This message requests the FMAC SW to start a scanning procedure on the given set of channels.

Msg_id	0x0C00
Confirmation message	SCANU_START_CFM

Table 36: SCANU_START_REQ description

3.1.3.2.1.1 Parameters

Name	Type	Description
chan	struct scan_chan_tag[42]	List of channels to be scanned. 2.4GHz and 5GHz must not be interleaved within the list.
ssid	struct mac_ssid[2]	List of SSIDs that need to be scanned.
bssid	u8[6]	Reserved for internal use
add_ies	u32	Physical address, in host memory, of the additional information elements to put in the ProbeReq frame while scanning actively.
add_ie_len	u16	Length of the additional IEs (0 if none).
vif_idx	u8	Index of the VIF that will scan
chan_cnt	u8	Number of channels present in the chan list
ssid_cnt	u8	Number of SSIDs present in the SSID list. Can be 0 if no specific SSIDs need to be found

no_cck	u8	Flag indicating whether CCK rates are allowed for the ProbeReq transmission. It is used mainly in case of a P2P discovery during which only OFDM rates can be used. 0: CCK rates are allowed 1: CCK rates are disallowed

Table 37: SCANU_SCAN_REQ parameters

3.1.3.2.2 SCANU_START_CFM

This message confirms the completion of the scanning procedure requested using the SCANU_START_REQ message.

Msg_id	0x0C01
Confirmation message	n/a

Table 38: SCANU_START_CFM description

3.1.3.2.3 SCANU_RESULT_IND

This message indicates a scanning result (i.e beacon or probe response) obtained during the scanning procedure.

msg_id	0x0C02
Confirmation message	n/a

Table 39: SCANU_RESULT_IND description

3.1.3.2.3.1 Parameters

Name	Type	Description
length	u16	Length of the received beacon or probe response.
framectrl	u16	FrameCtrl field of the received beacon or probe response.
center_freq	u16	Center frequency (in MHz) of the channel on which the beacon or probe response was received
band	u8	WiFi band on which the beacon or probe response was received: 0: 2.4GHz 1: 5GHz
sta_idx	u8	Index of the beacon or probe response transmitter if known. 0xFF otherwise.
inst_nbr	u8	Index of the VIF that received the beacon or probe response
rssi	s8	RSSI of the received beacon or probe response
payload	u8[512]	Array containing the payload of the received beacon or probe response

Table 40: SCANU_RESULT_IND parameters

3.1.4 Station Manager API description

3.1.4.1 Overview

Table 3 gives an overview of the requests/confirmations composing the SM API:

Request	Confirmation	Description
SM_CONNECT_REQ	SM_CONNECT_CFM	Start a connection procedure.
SM_DISCONNECT_REQ	SM_DISCONNECT_CFM	Disconnect from an AP

Table 41: Station Manager Control messages

Some indications can also be received asynchronously from the SM:

Indication	Description
SM_CONNECT_IND	Indication of a completed connection procedure
SM_DISCONNECT_IND	Indication of a disconnection

Table 42: MAC Entity Indication messages

3.1.4.2 Detailed description

3.1.4.2.1 SM_CONNECT_REQ

This message requests the FMAC SW to start a connection procedure to the given AP.

Msg_id	0x1400
Confirmation message	SM_CONNECT_CFM

Table 43: SM_CONNECT_REQ description

3.1.4.2.1.1 Parameters

Name	Type	Description
ssid	struct mac_ssid	SSID of the network to connect to.
bssid	u8[6]	BSSID of the AP to connect to. If this information is not available, this parameter shall be set to the wildcard BSSID, i.e. FF:FF:FF:FF:FF:FF.
chan	struct scan_chan_tag	Channel on which the connection shall be performed. If this information is not available, the *freq* field of the structure shall be set to -1.

flags	u32	Bit field representing some flags associated to the new connection. The bits are encoded as follows: bit0: Flag indicating whether the control port is controlled by the host or not (1: host controlled, 0 otherwise). bit1: Flag indicating whether the control port frames shall be sent unencrypted (1: unencrypted, 0 otherwise). bit2: Flag indicating whether HT shall be disabled or not (1: disabled, 0 otherwise). bit3: Flag indicating whether WPA, WPA2 or WPI authentication is in use or not (1: in use, 0 otherwise);
ctrl_port_ethertype	u16	Value of the Ethertype of the control port frames.
ie_len	u16	Length of the additional IEs to be added to the association request (e.g. for the RSN or WiFi Direct IEs).
listen_interval	u16	Listen interval to be used for this connection
dont_wait_bcmc	u8	Flag indicating whether the FW should wait for Broadcast/Multicast frame reception after beacon reception or go back to sleep immediately. 0: Listen for Broadcast/Multicast 1: Go back to sleep immediately after beacon reception
auth_type	u8	Authentication type to be put in the AUTH frame: 0x00: Open 0x01: Shared
uapsd_queues	u8	Bit field indicating on which access category the UAPSD frame delivery shall be enabled. bit0: When set, UAPSD is enabled for VOICE queue bit1: When set, UAPSD is enabled for VIDEO queue bit2: When set, UAPSD is enabled for BACKGROUND queue bit3: When set, UAPSD is enabled for BEST-EFFORT queue
vif_idx	u8	Index of the VIF that is attempting to connect
ie_buf	u32[32]	Buffer containing the additional IEs to be put in the association request.

Table 44: SM_CONNECT_REQ parameters

3.1.4.2.2 SM_CONNECT_CFM

This message confirms the start of the connection procedure requested using the SM_CONNECT_REQ message.

Msg_id	0x1401
Confirmation message	n/a

Table 45: SM_CONNECT_CFM description

3.1.4.2.2.1 Parameters

Name	Type	Description
status	u8	Status of the procedure: CO_OK: The connection procedure has been successfully started. CO_BUSY: The SM module is currently in a procedure and is not available to handle the new procedure. CO_OP_IN_PROGRESS: The VIF for which the connection is requested is already connected to an AP, or not acting as a STA.

Table 46: SM_CONNECT_CFM parameters

3.1.4.2.3 SM_CONNECT_IND

This message indicates the completion of the connection procedure that was started using the SM_CONNECT_REQ message.

Msg_id	0x1402
Confirmation message	n/a

Table 47: SM_CONNECT_IND description

3.1.4.2.3.1 Parameters

Name	Type	Description
status_code	u16	Status of the procedure: 0x0000: Successful. 0x0001-0xFFFF: These values indicate an unsuccessful completion of the procedure. In case the failure is coming from an error status received from the peer device (in either the AUTH frame or the ASSOC_RSP frame), then this field is set to the value of the status code received (see [1], chapter 8.4.1.9 for details on those codes). If this status is unknown (because the failure has not been detected by a received AUTH or ASSOC_SP packet), then this field is set to 0x0001.
Bssid	u8[6]	BSSID of the network we just connected to.
Roamed	bool	Boolean indicated whether the SM_CONNECT_IND results from a host request or an internal roaming procedure.
Vif_idx	u8	Index of the VIF on which the connection procedure was handled.
Ap_idx	u8	Index of the STA entry that was allocated internally to save the information about the AP. This parameter has to be saved by the host in order to be reused later for the procedures where it is required.

Ch_idx	u8	Index of the channel context that has been associated to the VIF for this connection.
Qos	bool	Boolean indicating whether the AP we associated to is supporting QoS or not.
Acm	u8	ACM bits set in the AP WMM parameter element
assoc_req_ie_len	u16	Length of the Information Elements present in the ASSOC_REQ we sent.
Assoc_rsp_ie_len	u16	Length of the Information Elements present in the ASSOC_RSP we received.
Assoc_ie_buf	u8[800]	Buffer containing the information elements of both the transmitted ASSOC_REQ and the received ASSOC_RSP. The elements of the ASSOC_RSP are located at an offset of *assoc_req_ie_len* in the buffer.

Table 48: SM_CONNECT_IND parameters

3.1.4.2.4 SM_DISCONNECT_REQ

This message requests the FMAC SW to disconnect from the AP we are connected to.

Msg_id	0x1403
Confirmation message	SM_DISCONNECT_CFM

Table 49: SM_CONNECT_REQ description

3.1.4.2.4.1 Parameters

Name	Type	Description
reason_code	u16	Code indicating the reason of the disconnection. This code, the value of which has to be taken in [1], chapter 8.4.1.7, is inserted in the DEAUTH frame sent to the AP.
Vif_idx	u8	Index of the VIF that has to disconnect.

Table 50: SM_DISCONNECT_REQ parameters

3.1.4.2.5 SM_DISCONNECT_CFM

This message confirms the start of the disconnection procedure requested using the SM_DISCONNECT_REQ message.

Msg_id	0x1404
Confirmation message	n/a

Table 51: SM_DISCONNECT_CFM description

3.1.4.2.6 SM_DISCONNECT_IND

This message indicates the completion of the disconnection procedure that was started using the SM_DISCONNECT_REQ message, a disconnection requested by the AP, or a disconnection detected internally (e.g. detected by the connection monitoring module).

Msg_id	0x1405
Confirmation message	n/a

Table 52: SM_DISCONNECT_IND description

3.1.4.2.6.1 Parameters

Name	Type	Description
reason_code	u16	Code indicating the reason of the disconnection: 0x0000: Value used in case of disconnection initiated by the host using the SM_DISCONNECT_REQ message. 0x0001-0xFFFF: Value of the reason code field received in the DEAUTH or DISASSOC message (see [1], chapter 8.4.1.7 for more details). In case the disconnection is detected locally, a value of 0x0001 (Reason Unspecified) is used.
Vif_idx	u8	Index of the VIF that has disconnected.

Table 53: SM_DISCONNECT_IND parameters

3.1.5 Access Point Manager API description

3.1.5.1 Overview

Table 3 gives an overview of the requests/confirmations composing the APM API:

Request	Confirmation	Description
APM_START_REQ	APM_START_CFM	Start an Access Point.
APM_STOP_REQ	APM_STOP_CFM	Stop a started Access Point
APM_START_CAC_REQ	APM_START_CAC_CFM	Start the Channel Availability Check procedure
APM_STOP_CAC_REQ	APM_STOP_CAC_CFM	Stop the Channel Availability Check procedure

Table 54: Station Manager Control messages

3.1.5.2 Detailed description

3.1.5.2.1 APM_START_REQ

This message requests the FMAC SW to start an Access Point on the VIF passed as parameter.

Msg_id	0x1800
Confirmation message	APM_START_CFM

Table 55: APM_START_REQ description

3.1.5.2.1.1 Parameters

Name	Type	Description
basic_rates	struct mac_rateset	Basic rates the AP will advertise.
chan	struct scan_chan_tag	Channel on which the AP shall be started.
center_freq1	u32	Center frequency of the first segment (in MHz). This parameter allows indicating to the FMAC SW whether the secondary channel is above or below the primary one. Ex: If the started AP has to be of 40MHz bandwidth, and the secondary channel is above the primary one, then the value of this field shall be 10MHz greater than the *freq* field of the *chan* parameter.
center_freq2	u32	Center frequency of the second segment (in MHz). This parameter is applicable only in 80+80 configuration.
ch_width	u8	Channel width: 0x00: 20MHz 0x01: 40MHz 0x02: 80MHz 0x03: 160MHz or 80+80MHz

bcn_addr	u32	Physical address, in host memory, of the beacon template. This address will be used to download the beacon buffer in embedded memory.
bcn_len	u16	Length of the beacon template.
tim_oft	u16	Offset of the TIM IE in the beacon template.
bcn_int	u16	Beacon interval to be used (in TUs).
flags	u32	Bit field representing some flags associated to the new connection. The bits are encoded as follows: bit0: Flag indicating whether the control port is controlled by the host or not (1: host controlled, 0 otherwise). bit1: Flag indicating whether the control port frames shall be sent unencrypted (1: unencrypted, 0 otherwise). bit2: Flag indicating whether HT shall be disabled or not (1: disabled, 0 otherwise). bit3: Flag indicating whether WPA, WPA2 or WPI authentication is in use or not (1: in use, 0 otherwise);
ctrl_port_ethertype	u16	Value of the Ethertype of the control port frames.
tim_len	u8	Length of the TIM information element present in the beacon template.
vif_idx	u8	Index of the VIF that is attempting to connect

Table 56: APM_START_REQ parameters

3.1.5.2.2 APM_START_CFM

This message confirms the start of AP requested using the APM_START_REQ message. In case the AP is not started, the status parameter indicates the failure of the procedure.

Msg_id	0x1801
Confirmation message	n/a

Table 57: APM_START_CFM description

3.1.5.2.2.1 Parameters

Name	Type	Description

status	u8	Status of the procedure: CO_OK: The AP has been successfully started. CO_BAD_PARAM: Some parameters of the request or of the device are not correct. Ex: the VIF on which the AP is started is not of type AP. CO_BUSY: The APM module is currently busy with a procedure and cannot handle the request. CO_OP_IN_PROGRESS: An AP is already started on this VIF. CO_FAIL: Used for all other failures.
vif_idx	u8	Index of the VIF on which the AP start was requested.
ch_idx	u8	Index of the channel context that has been associated to the VIF for this connection.
bcmc_idx	u8	Index of the STA used for the BC/MC traffic on this AP.

Table 58: APM_START_CFM parameters

3.1.5.2.3 APM_STOP_REQ

This message requests the FMAC SW to stop an AP that was previously started.

Msg_id	0x1802
Confirmation message	APM_STOP_CFM

Table 59: APM_STOP_REQ description

3.1.5.2.3.1 Parameters

Name	Type	Description
vif_idx	u8	Index of the VIF on which the AP has to be stopped.

Table 60: APM_STOP_REQ parameters

3.1.5.2.4 APM_STOP_CFM

This message confirms the AP stop procedure requested using the APM_STOP_REQ message.

Msg_id	0x1803
Confirmation message	n/a

Table 61: APM_STOP_CFM description

3.1.5.2.5 APM_START_CAC_REQ

This message requests the FMAC SW to start a Channel Availability Check procedure on the channel passed as parameter.

Msg_id	0x1804
Confirmation message	APM_START_CAC_CFM

Table 62: APM_START_CAC_REQ description

3.1.5.2.5.1 Parameters

Name	Type	Description
chan	struct scan_chan_tag	Channel on which the CAC shall be started.
center_freq1	u32	Center frequency of the first segment (in MHz). This parameter allows indicating to the FMAC SW whether the secondary channel is above or below the primary one.
center_freq2	u32	Center frequency of the second segment (in MHz). This parameter is applicable only in 80+80 configuration.
ch_width	u8	Channel width: 0x00: 20MHz 0x01: 40MHz 0x02: 80MHz 0x03: 160MHz or 80+80MHz
vif_idx	u8	Index of the VIF for which the CAC is started

Table 63: APM_START_CAC_REQ parameters

3.1.5.2.6 APM_START_CAC_CFM

This message confirms the start of CAC requested using the APM_START_CAC_REQ message. In case the CAC is not started, the status parameter indicates the failure of the procedure.

Msg_id	0x1805
Confirmation message	n/a

Table 64: APM_START_CAC_CFM description

3.1.5.2.6.1 Parameters

Name	Type	Description
status	u8	Status of the procedure: CO_OK: The CAC has been successfully started. Different from CO_OK if some failures occurred.

ch_idx	u8	Index of the channel context that has been associated to the VIF for this connection.

Table 65: APM_START_CAC_CFM parameters

3.1.5.2.7 APM_STOP_CAC_REQ

This message requests the FMAC SW to stop the CAC that was previously started.

Msg_id	0x1802
Confirmation message	APM_STOP_CAC_CFM

Table 66: APM_STOP_CAC_REQ description

3.1.5.2.7.1 Parameters

Name	Type	Description
vif_idx	u8	Index of the VIF on which the CAC was started.

Table 67: APM_STOP_CAC_REQ parameters

3.1.5.2.8 APM_STOP_CAC_CFM

This message confirms the CAC stop procedure requested using the APM_STOP_CAC_REQ message.

Msg_id	0x1803
Confirmation message	n/a

Table 68: APM_STOP_CAC_CFM description

3.1.6 Mesh Manager API description

3.1.6.1 Overview

Table 69 gives an overview of the requests/confirmations composing the MESH API:

Request	Confirmation	Description
MESH_START_REQ	MESH_START_CFM	Start a Mesh Point
MESH_STOP_REQ	MESH_STOP_CFM	Stop a started Mesh Point
MESH_UPDATE_REQ	MESH_UPDATE_CFM	Update parameters of a started Mesh Point
MESH_PEER_INFO_REQ	MESH_PEER_INFO_RSP	Get information about a given peer Mesh STA
MESH_PATH_CREATE_REQ	MESH_PATH_CREATE_CFM	Require creation of a mesh path with a given peer Mesh STA
MESH_PATH_UPDATE_REQ	MESH_PATH_UPDATE_CFM	Require update of an existing mesh path
MESH_PROXY_ADD_REQ	N/A	Inform UMAC about discovery of a proxied STA
MESH_PEER_UPDATE_NTF	N/A	Inform UMAC about a peering state update.

Table 69: Mesh Manager Control messages

3.1.6.2 Detailed description

3.1.6.2.1 MESH_START_REQ

This message requests the FMAC SW to start a Mesh Point on the VIF passed as parameter.

Msg_id	0x2000
Confirmation message	MESH_START_CFM

Table 70: MESH_START_REQ description

3.1.6.2.1.1 Parameters

Name	Type	Description
basic_rates	struct mac_rateset	Basic rates the MP will advertise.
chan	struct scan_chan_tag	Channel on which the MP shall be started.
center_freq1	u32	Center frequency of the first segment (in MHz). This parameter allows indicating to the FMAC SW whether the secondary channel is above or below the primary one. Ex: If the started MP has to be of 40MHz bandwidth, and the secondary channel is above the primary one, then the value of this field shall be 10MHz greater than the *freq* field of the *chan* parameter.
center_freq2	u32	Center frequency of the second segment (in MHz). This parameter is applicable only in 80+80 configuration.

ch_width	u8	Channel width: 0x00: 20MHz 0x01: 40MHz 0x02: 80MHz 0x03: 160MHz or 80+80MHz
dtim_period	u8	DTIM period to be used
bcn_int	u8	Beacon interval to be used (in TUs).
vif_idx	u8	Index of the VIF on which MP has to be created
mesh_id_len	u8	Length of provided Mesh ID
mesh_id	u8[MESH_MESHID_MAX_LEN] u8[32]	Mesh ID
ie_addr	u32	Address of additional Information Elements to be downloaded from host memory through a DMA transfer. These Information Elements are added at beacon's end.
ie_len	u16	Length of additional Information Elements
user_mpm	bool	Indicate if Mesh Peering Management protocol (MPM) is handled by host or directly by UMAC.
is_auth	bool	Indicate if security features have to be used on the created MP.
auth_id	u8	Authentication protocol identifier • 0 = No authentication method is required to establish mesh peerings within the MBSS • 1 = SAE • 2 = IEEE 802.11X authentication If auth_id is 1 or 2 then user_mpm value shall be true.

Table 71: MESH_START_REQ parameters

3.1.6.2.2 MESH_START_CFM

This message confirms the start of MP requested using the MESH_START_REQ message. In case the MP is not started, the status parameter indicates the failure of the procedure.

Msg_id	0x2001
Confirmation message	n/a

Table 72: MESH_START_CFM description

3.1.6.2.2.1 Parameters

Name	Type	Description

status	u8	Status of the procedure: CO_OK: The MP has been successfully started. CO_BAD_PARAM: Some parameters of the request or of the device are not correct. Ex: the VIF on which the MP is started is not of type MP. CO_BUSY: The MESH module is currently busy with a procedure and cannot handle the request. CO_OP_IN_PROGRESS: A MP is already started on this VIF. CO_FAIL: Used for all other failures.
vif_idx	u8	Index of the VIF on which the MP start was requested.
ch_idx	u8	Index of the channel context that has been associated to the VIF for this connection.
bcmc_idx	u8	Index of the STA used for the BC/MC traffic on this MP.

Table 73: MESH_START_CFM parameters

3.1.6.2.3 MESH_STOP_REQ

This message requests the FMAC SW to stop a MP that was previously started.

Msg_id	0x2002
Confirmation message	MESH_STOP_CFM

Table 74: MESH_STOP_REQ description

3.1.6.2.3.1 Parameters

Name	Type	Description
vif_idx	u8	Index of the VIF on which the MP has to be stopped.

Table 75: MESH_STOP_REQ parameters

3.1.6.2.4 MESH_STOP_CFM

This message confirms the MP stop procedure requested using the MESH_STOP_REQ message.

Msg_id	0x2003
Confirmation message	n/a

Table 76: MESH_STOP_CFM description

3.1.6.2.4.1 Parameters

Name	Type	Description

vif_idx	u8	Index of the VIF on which the MP stop was requested.
status	u8	Status of the procedure: CO_OK: The MP has been successfully stopped. CO_FAIL: Request handling has failed.

Table 77: MESH_STOP_CFM parameters

3.1.6.2.5 MESH_UPDATE_REQ

This message requests the FMAC SW to update one or several parameters of an existing MP interface.

Msg_id	0x2004
Confirmation message	MESH_UPDATE_CFM

Table 78: MESH_UPDATE_REQ description

3.1.6.2.5.1 Parameters

Name	Type	Description
flags	u8	Bit field indicating fields which have been updated • Bit 0: Update Root Mode • Bit 1: Update Gate Announcement • Bit 2: Update Mesh Forwarding support status • Bit 3: Update Local PS Mode
vif_idx	u8	VIF Index of the interface on which update has to be done
root_mode	u8	New Root Mode value • 0 = The Mesh STA is not a Root Mesh STA • 1 = The Mesh STA is a Root Mesh STA • 2 = The Mesh STA is a Root Mesh STA supporting Proactive PREQ with proactive PREP subfield set to 0 • 3 = The Mesh STA is a Root Mesh STA supporting Proactive PREQ with proactive PREP subfield set to 1 • 4 = The Mesh STA is a Root Mesh STA supporting Proactive RANN
gate_announ	bool	Enable or disable Gate role
mesh_forward	bool	Enable or disable support of Mesh Forwarding feature.
local_ps_mode	u8	Set Local PS Mode • 1 = MP PS Mode is Active meaning that the MP is always present • 2 = MP PS Mode is Light Sleep • 3 = MP PS Mode is Deep Sleep

Table 79: MESH_UPDATE_REQ parameters

3.1.6.2.6 MESH_UPDATE_CFM

This message confirms the MP update procedure requested using the MESH_UPDATE_REQ message.

Msg_id	0x2005
Confirmation message	n/a

Table 80: MESH_UPDATE_CFM description

3.1.6.2.6.1 Parameters

Name	Type	Description
status	u8	Status of the procedure: CO_OK: The MP has been successfully stopped. CO_FAIL: Request handling has failed.

Table 81: MESH_UPDATE_CFM parameters

3.1.6.2.7 MESH_PEER_INFO_REQ

This message requests the FMAC SW to stop a MP that was previously started.

Msg_id	0x2006
Confirmation message	MESH_PEER_INFO_RSP

Table 82: MESH_STOP_REQ description

3.1.6.2.7.1 Parameters

Name	Type	Description
sta_idx	u8	Index of the STA for which information are required

Table 83: MESH_PEER_INFO_REQ parameters

3.1.6.2.8 MESH_PEER_INFO_RSP

This message contains the information requested using the MESH_PEER_INFO_REQ message.

Msg_id	0x2007
Confirmation message	n/a

Table 84: MESH_PEER_INFO_RSP description

3.1.6.2.8.1 Parameters

Name	Type	Description
status	u8	Request status
sta_idx	u8	Index of the STA for which the information are provided
local_link_id	u16	Local Link ID
peer_link_id	u16	Peer Link ID
local_ps_mode	u8	Current Local PS Mode
peer_ps_mode	u8	Current Peer PS Mode
non_peer_ps_mode	u8	Current Non Peer PS Mode
link_state	u8	Link State

Table 85: MESH_PEER_INFO_RSP parameters

3.1.6.2.9 MESH_PATH_CREATE_REQ

This message requests the FMAC SW to create a mesh path to a given target mesh STA.

Msg_id	0x2008
Confirmation message	MESH_PATH_CREATE_CFM

Table 86: MESH_PATH_CREATE_REQ description

3.1.6.2.9.1 Parameters

Name	Type	Description
vif_idx	u8	Index of the VIF on which the mesh path has to be created
has_orig_addr	bool	Indicate if path creation is required on behalf of a locally proxied STA.
tgt_mac_addr	struct mac_addr	Path target MAC Address
orig_mac_addr	struct mac_addr	Path originator MAC Address, valid only if has_orig_addr is true.

Table 87: MESH_PATH_CREATE_REQ parameters

3.1.6.2.10 MESH_PATH_CREATE_CFM

This message confirms the Mesh Path creation procedure requested using the MESH_PATH_CREATE_REQ message.

Msg_id	0x2009
Confirmation message	n/a

Table 88: MESH_PATH_CREATE_CFM description

3.1.6.2.10.1 Parameters

Name	Type	Description
status	u8	Request status
vif_idx	u8	Index of the VIF on which path creation procedure has occurred

Table 89: MESH_PATH_CREATE_CFM parameters

3.1.6.2.11 MESH_PATH_UPDATE_REQ

This message requests the FMAC SW to update its information about an existing mesh path.

Msg_id	0x200A
Confirmation message	MESH_PATH_UPDATE_CFM

Table 90: MESH_PATH_UPDATE_REQ description

3.1.6.2.11.1 Parameters

Name	Type	Description
delete	bool	Indicate if mesh path has to be deleted
vif_idx	u8	Index of the VIF on which the mesh path has to be updated
tgt_mac_addr	struct mac_addr	Path target MAC Address
nhop_mac_addr	struct mac_addr	Next mesh STA hop MAC Address

Table 91: MESH_PATH_UPDATE_REQ parameters

3.1.6.2.12 MESH_PATH_UPDATE_CFM

This message confirms the Mesh Path update procedure requested using the MESH_PATH_UPDATE_REQ message.

Msg_id	0x200B
Confirmation message	n/a

Table 92: MESH_PATH_UPDATE_CFM description

3.1.6.2.12.1 Parameters

Name	Type	Description
status	u8	Request status

vif_idx	u8	Index of the VIF on which path update procedure has occurred

Table 93: MESH_PATH_UPDATE_CFM parameters

3.1.6.2.13 MESH_PROXY_ADD_REQ

This message informs the FMAC SW about discovery of a locally proxied interface. By knowing this information, the FMAC SW is able to answer to Path Request frame whose target address would be the address of this interface.

Msg_id	0x200C
Confirmation message	N/A

Table 94: MESH_PROXY_ADD_REQ description

3.1.6.2.13.1 Parameters

Name	Type	Description
vif_idx	u8	Index of the VIF on which the proxy information has to be kept.
ext_sta_addr	struct mac_addr	MAC Address of the proxied interface

Table 95: MESH_PROXY_ADD_REQ parameters

3.1.6.2.14 MESH_PROXY_UPDATE_IND

This message indicates that information about a proxied device have been updated.

msg_id	0x200D
Confirmation message	n/a

Table 96: MESH_PROXY_UPDATE_IND description

3.1.6.2.14.1 Parameters

Name	Type	Description
delete	bool	Indicate if information have been deleted from FMAC SW memory
local	bool	Indicate if device is locally proxied or if its proxy is a peer mesh STA.
vif_idx	u8	Index of the interface on which the proxy information have been updated.
ext_sta_addr	struct mac_addr	Address of the proxied device
proxy_mac_addr	struct mac_addr	Address of the mesh device acting as proxy for the indicated device.

Table 97: MESH_PROXY_UPDATE_IND parameters

3.1.6.2.15 MESH_PEER_UPDATE_IND

This message informs the host about establishment or deletion of a peering with a mesh STA.

msg_id	0x200E
Confirmation message	n/a

Table 98: MESH_PEER_UPDATE_IND description

3.1.6.2.15.1 Parameters

Name	Type	Description
estab	bool	Indicate if peering has been established or closed
vif_idx	u8	Index of the interface on which the peering status has been updated
sta_idx	u8	Index of the STA for which the peering status has been updated
peer_addr	struct mac_addr	MAC Address of the peer Mesh STA

Table 99: MESH_PEER_UPDATE_IND parameters

3.1.6.2.16 MESH_PEER_UPDATE_NTF

This message informs the FMAC SW of an update of a Mesh Peering state in case where the Mesh Peering Management protocol is handled in host and not locally in FMAC SW.

msg_id	0x200F
Confirmation message	n/a

Table 100: MESH_PEER_UPDATE_NTF description

3.1.6.2.16.1 Parameters

Name	Type	Description
vif_idx	u8	Index of the interface on which the peering status has been updated
sta_idx	u8	Index of the STA for which the peering status has been updated
state	u8	New peering state

Table 101: MESH_PEER_UPDATE_NTF parameters

3.1.6.2.17 MESH_PATH_UPDATE_IND

This message informs the host about addition or deletion of a mesh path.

msg_id	0x2010
Confirmation message	n/a

Table 102: MESH_PATH_UPDATE_IND description

3.1.6.2.17.1 Parameters

Name	Type	Description
delete	bool	Indicate if mesh path has been deleted or not
ext_sta	bool	Indicate if mesh path target is an external STA (meaning a STA that is not part of the MBSS but which is proxied by a mesh STA in the MBSS)
vif_idx	u8	Index of the interface on which the mesh path has been added or deleted
path_idx	u8	Index allocated for the path
tgt_mac_addr	struct mac_addr	MAC Address of the last mesh STA to reach
ext_sta_mac_addr	struct mac_addr	MAC Address of the target device if the device is outside the MBSS. This address is valid only if ext_sta value is true.
nhop_sta_idx	u8	STA index of the next hop Mesh STA.

Table 103: MESH_PATH_UPDATE_IND parameters

3.2 Data interface

3.2.1 Transmission

3.2.1.1 Transmission flow

The transmission of MSDUs between the driver and the FMAC involves the movement of 3 types of data structures:

1. The TX descriptors that include the information necessary for the FMAC to download the payloads into its buffer memory (e.g. TX buffer address in Host memory, Payload length, etc.)
2. The TX buffers that include the MSDU data in an Ethernet format
3. The TX confirmation descriptors that include some status fields that are uploaded by the FMAC once the transmission is complete.

The transmission flow is summarized in the figure below:

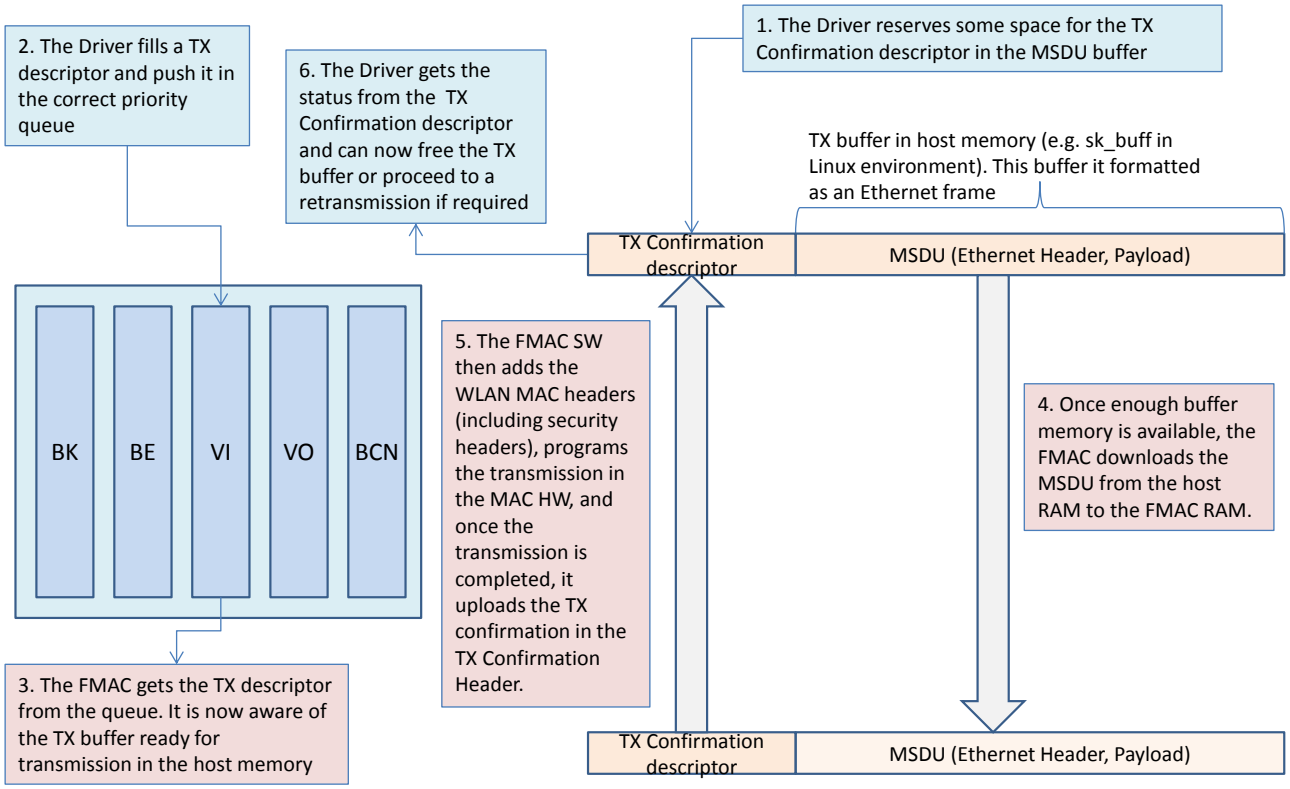

Figure 3: Transmission flow

A credit based flow control mechanism is also implemented. This mechanism, which is independent from the HW interface between the host and the FMAC, allows controlling how many packets can be pushed by the driver to the FMAC at a given time, on a RA/TID basis. It is used to avoid sending packets outside the transmission window in case a BlockAck agreement has been established.

It is recommended that the driver above the FMAC implements a TX packet queuing mechanism based on RA/TID (i.e. destination/priority), to get the best performance in transmission. The sample driver provided with the FMAC SW IP implements such a queuing.

3.2.1.2 Transmission Confirmation Descriptor

The TX Confirmation Descriptor is generally placed in the MSDU payload buffer, at a place reserved by the driver. The physical address of this descriptor is passed to the FMAC SW (using a specific field of the TX descriptor), and the status is then uploaded to this descriptor by the FMAC SW upon the packet transmission.

The fields of this descriptor are the following:

Name	Type	Description
pn	u16[4]	TKIP, CCMP or WAPI packet counter that was assigned by the FMAC SW on the first transmission of this MSDU. This field is null in other security modes.
sn	u16	Sequence number that was assigned by the FMAC SW on the first transmission of this MSDU.
timestamp	u16	Timestamp that was assigned by the FMAC SW on the first transmission of this MSDU.
credits	s8	Number of credits allocated to the RA/TID pair attached to this packet. This number has to be added to the credit count maintained by the driver.
status	u32	This field is formatted according to [3], chapter 3.1.3, field Status Information. It has to be initialized to 0 by the driver before being pushed to the FMAC. This field will then be used by the driver in the context of the TX confirmation checking to know if the buffer has been transmitted (if different from 0) or if it is still pending for transmission in the FMAC (if equal to 0).

Table 104: TX Confirmation Descriptor fields

3.2.1.3 Transmission Descriptor

The transmission descriptors are used to indicate to the FMAC the packets pending for transmission in the host memory. Once a transmission descriptor has been pushed to the FMAC, the Driver has to wait for the transmission confirmation before freeing the TX buffer.

In order to fulfill QoS requirements, the transmission descriptors are pushed in different queues according to their priority.

The format of the transmission descriptor is described in the table below:

Name	Type	Description
packet_addr	u32	Physical address of the TX buffer in host memory. It has to point to the first byte of the MSDU.
packet_len	u16	Length of the MSDU.
status_desc_addr	u32	Physical address of the TX confirmation descriptor associated to this buffer in host memory.
eth_dest_addr	u8[6]	Destination MAC address, as present in the MSDU Ethernet header.
eth_src_addr	u8[6]	Source MAC address, as present in the MSDU Ethernet header.

ethertype	u16	Ethernet type, as present in the MSDU Ethernet header.
pn	u16[4]	TKIP, CCMP or WAPI packet counter that was assigned by the FMAC SW on the first transmission of this MSDU (used only in case of a retry requested by the Driver). The value put here comes from the TX confirmation descriptor uploaded at the end of the first transmission attempt of this packet.
sn	u16	Sequence number that was assigned by the FMAC SW on the first transmission of this MSDU (used only in case of a retry requested by the Driver). The value put here comes from the TX confirmation descriptor uploaded at the end of the first transmission attempt of this packet.
timestamp	u16	Timestamp that was assigned by the FMAC SW on the first transmission of this MSDU (used only in case of a retry requested by the Driver). The value put here comes from the TX confirmation descriptor uploaded at the end of the first transmission attempt of this packet.
tid	u8	For the MSDUs that go to QoS STAs, this field contains the TID to be put in the QoS Control field. It is taken from the DSCP field of the IP header. For packets not going to QoS STAs, this field shall be set to 0xFF.
vif_idx	u8	Index of the virtual interface that will perform the transmission.
staid	u8	Identifier of the station this packet is for.
flags	u8	Bit field containing some information flags about the transmission: bit0: When set, it indicates that the current frame is a retry that was requested by the FMAC SW after a previous attempt. bit1: Reserved bit2: When set, it indicates that more data is available for this destination STAs in the driver queues. Bit3-7: Reserved

Table 105: TX Descriptor fields

3.2.2 Reception

3.2.2.1 Reception flow

The reception of MPDUs involves the movement of 2 types of data structures between the FMAC and the driver:

1. The RX buffer descriptors. This structure is provided by the driver to give information to the FMAC about the buffers it has allocated for the receptions.
2. The RX buffers. This structure is contiguous memory space allowing to receive the biggest packets.
3. The RX descriptors. This structure contains information such as a status indicating the action the driver has to perform and an identifier of the RX buffer on which the action shall be performed.

At startup a pool of both types of data structure are allocated by the driver. The physical addresses of these structures are then shared between the host and FMAC in two lists that are written by the host (in order to indicate that a free structure (either a buffer or a descriptor) is available, and read cleared by the FMAC when it needs to get a fresh structure.

The movement of the RX buffers upon a reception is summarized in the figure below:

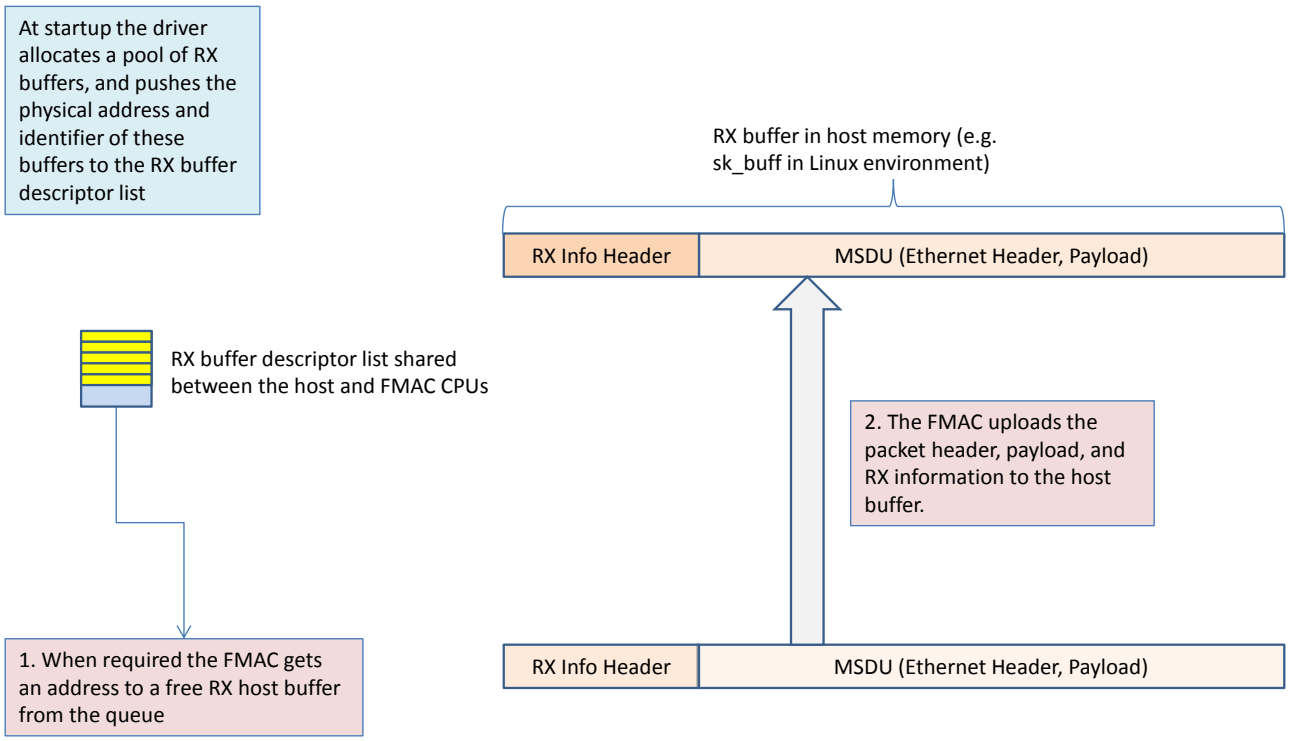

Figure 4: RX Buffer movements

The movement of the RX descriptors is summarized in the figure below:

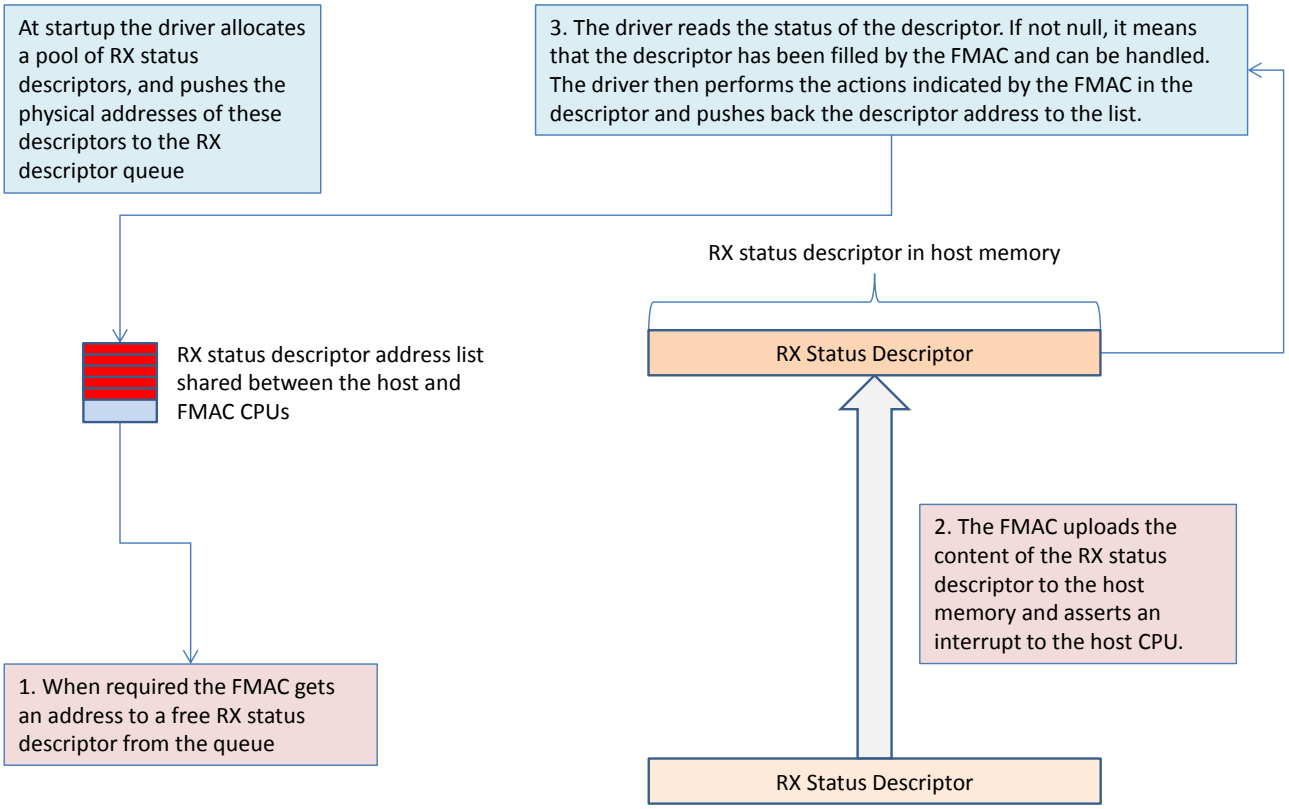

Figure 5: RX status descriptor movements

These two mechanisms are then used according to the type of reception. A few examples of receptions are explained below:

Reception of an unfragmented MSDU, not under BlockAck agreement:

1. Upon the frame reception by the HW, the FMAC first gets a RX host buffer address from the list and uploads the MSDU to this buffer.
2. The FMAC then uploads a RX descriptor, indicating in the descriptor status that the driver shall forward the packet immediately to the networking layers. This status also indicates to the driver that it needs to reallocate a new buffer and make it available for the FMAC SW (address pushed in the RX buffer list).

Reception of a fragmented MSDU:

First fragment:

1. Upon the frame reception by the HW, the FMAC first gets a RX host buffer address from the list and uploads the first fragment of the MSDU to this buffer.
2. The FMAC then uploads a RX descriptor, indicating in the descriptor status that the driver shall keep the packet (because it is not complete). This status also indicates to the driver that it needs to reallocate a new buffer and make it available for the FMAC SW (address pushed in the RX buffer list).

Intermediate fragment:

1. Upon the frame reception by the HW, the FMAC uploads the fragment of the MSDU to the host buffer that was previously allocated upon the reception of the first fragment.

Last fragment:

1. Upon the frame reception by the HW, the FMAC uploads the fragment of the MSDU to the host buffer that was previously allocated upon the reception of the first fragment.
2. The FMAC then uploads a RX descriptor, indicating in the descriptor status that the driver shall now forward the packet.

Reception of an unordered MSDU, under BlockAck agreement:

1. Upon the frame reception by the HW, the FMAC first gets a RX host buffer address from the list and uploads the MSDU to this buffer.
2. The FMAC then uploads a RX descriptor, indicating in the descriptor status that the driver shall keep the packet. This status also indicates to the driver that it needs to reallocate a new buffer and make it available for the FMAC SW (address pushed in the RX buffer list).

Reception of an ordered MSDU, under BlockAck agreement:

1. Upon the frame reception by the HW, the FMAC first gets a RX host buffer address from the list and uploads the MSDU to this buffer.
2. The FMAC then uploads a RX descriptor, indicating in the descriptor status that the driver shall forward the packet immediately to the networking layers. This status also indicates to the driver that it needs to reallocate a new buffer and make it available for the FMAC SW (address pushed in the RX buffer list).
3. In case the reception of this packet allows moving the RX window (e.g. packet n received after packets n+1, n+2, etc.), then RX descriptors are uploaded in order to request to the driver to forward the previously received unordered packets).

3.2.2.2 Receive Buffer Descriptor

The RX buffer descriptor is provided by the driver to the FMAC to inform about the RX buffer location in host memory, and the identifier of the buffer.

The fields of this descriptor are the following:

Name	Type	Description
hostid	u32	Identifier of the Host Buffer. This field is not used directly by the FMAC SW. Nevertheless it is set in the RX status descriptors forwarded by the FMAC in order to allow the driver recovering information about the buffer. In a Linux like environment, this value should typically be the pointer to the sk_buff structure that has been allocated for the RX buffer.
dma_addr	u32	Physical address of the RX buffer space in host memory. This value is used by the FMAC SW to program the DMA transfer of the received payloads to the host buffer.

Table 106: RX buffer descriptor fields

3.2.2.3 Receive Information Header

The Receive Information Header includes all the properties of the received packet. These properties include the length of the packet, its decryption status, its PHY parameters (rate/MCS, bandwidth, guard interval, channel, RSSI, etc.).

The fields of this header are the following:

Name	Type	Description
length	u32	This field contains the length of the received MPDU, including the different header length (MAC Header, IV/EIV). In case of TKIP frame reception, it also includes the TKIP MIC length. FCS, ICV or CCMP MIC lengths are not included in this field.
tsflo	u32	This field contains the lower 4 bytes of the timestamp (TSF) at which the frame ended on air.
tsfhi	u32	This field contains the higher 4 bytes of the timestamp (TSF) at which the frame ended on air.
recvec1a	u32	This field is formatted according to [3], chapter 3.2.3, field Receive Vector 1a.
recvec1b	u32	This field is formatted according to [3], chapter 3.2.3, field Receive Vector 1b.
recvec1c	u32	This field is formatted according to [3], chapter 3.2.3, field Receive Vector 1c.
recvec1d	u32	This field is formatted according to [3], chapter 3.2.3, field Receive Vector 1d.
recvec2a	u32	This field is formatted according to [3], chapter 3.2.3, field Receive Vector 2a.
recvec2b	u32	This field is formatted according to [3], chapter 3.2.3, field Receive Vector 2b.
status	u32	This field is formatted according to [3], chapter 3.2.3, field MPDU Status Information.
phychannelinfo1	u32	This field contains some information about the channel on which the frame was received. It is composed of the following elements: - Bits 31-16: prim20_freq (in MHz) - Bits 15-8: channel_type (0: 20MHz, 1: 40 MHz, 2: 80MHz, 3: 160MHz or 80+80MHz) - Bits 7-0: band (0: 2.4GHz, 1: 5GHz)
phychannelinfo2	u32	This field contains some information about the channel on which the frame was received. It is composed of the following elements: - Bits 31-16: center2_freq (in MHz) - Bits 15-0: center1_freq (in MHz)

flags	u32	Bit field providing additional information about the packet: bit0: When set, this bit indicates that the packet in memory is an A-MSDU that shall be deaggregated by the driver. bit1: When set, this bit indicates that the packet in memory was not converted to an Ethernet format, and has a WLAN format. bit8-15: Index of the interface (VIF) that received the packet (0xFF if unknown). bit16-23: Index of the station that transmitted the packet (0xFF if unknown). All other bits are reserved.

Table 107: RX Information Header fields

3.2.2.4 Receive Status Descriptor

The RX status descriptor controls the behavior of the driver when a new packet has been received.

The fields of this descriptor are the following:

Name	Type	Description
hostid	u32	Identifier of the Host Buffer. This value is passed by the driver in the RX buffer list along with the physical address of the buffer. It is not used by the FMAC SW, but it allows indicating to the host which RX buffer is affected by the present RX descriptor.
length	u32	Length of the host buffer. This field is used only in case the status indicates that the length of the host buffer needs to be updated.
status	u8	This field indicates the actions that the host needs to perform upon the reception of the present RX descriptor: 0x00: RX Descriptor not yet written by the FMAC FW 0x01: Forward the buffer to the upper layers and provide a fresh RX buffer to the FMAC SW. 0x02: Temporarily keep the buffer and provide a fresh RX buffer to the FMAC SW. 0x03: Free the buffer. 0x04: Update the length of the buffer 0x05: Update the length of the buffer and copy the length in the Ethernet Header 0x06: Do nothing 0x07-0xFF: reserved

Table 108: RX status descriptor fields

4 API usage

4.1 Overview

This section describes the interaction between the WLAN driver and the FMAC SW for the classical WLAN procedures. The procedures described in the following chapters are supposed to be performed starting from a system in down state (i.e. just after initialization or after applying the initialization procedure as described in **Error! Reference source ot found.**), unless stated differently.

The message sequence charts below are indicative and allow getting a good understanding of the FMAC procedures. They are not intended to provide an exhaustive list of all the allowed message combinations.

Some procedures involve the transmission and reception of messages part of the LMAC SW API (the prefix of those messages is MM_). These messages are described in [4].

The conventions used in the message sequence charts are described in Figure 6.

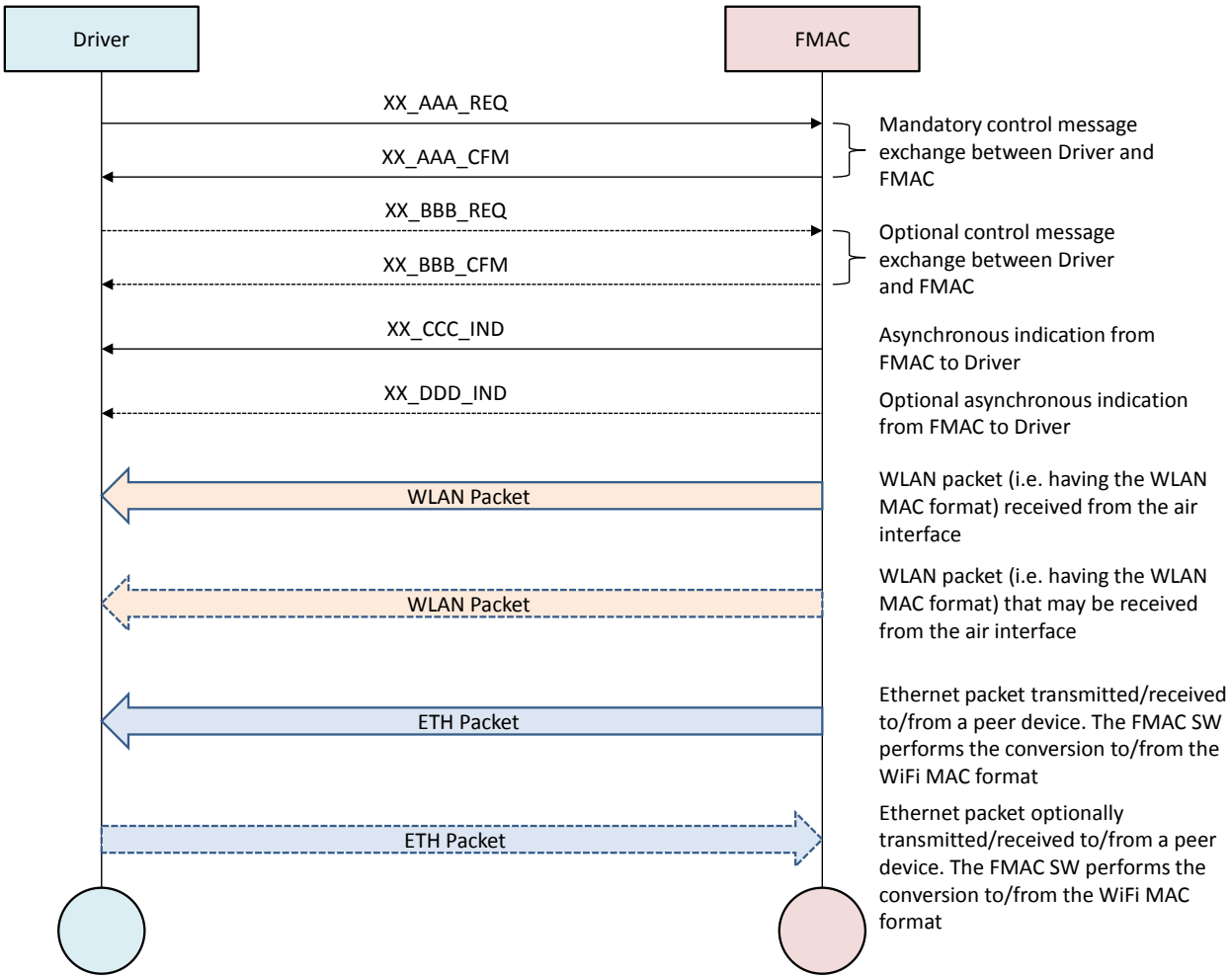

Figure 6: Message sequence chart conventions

4.2 Procedures

4.2.1 System initialization

This procedure performs the initialization of the system. It is supposed to be executed at the after the FMAC FW boot.

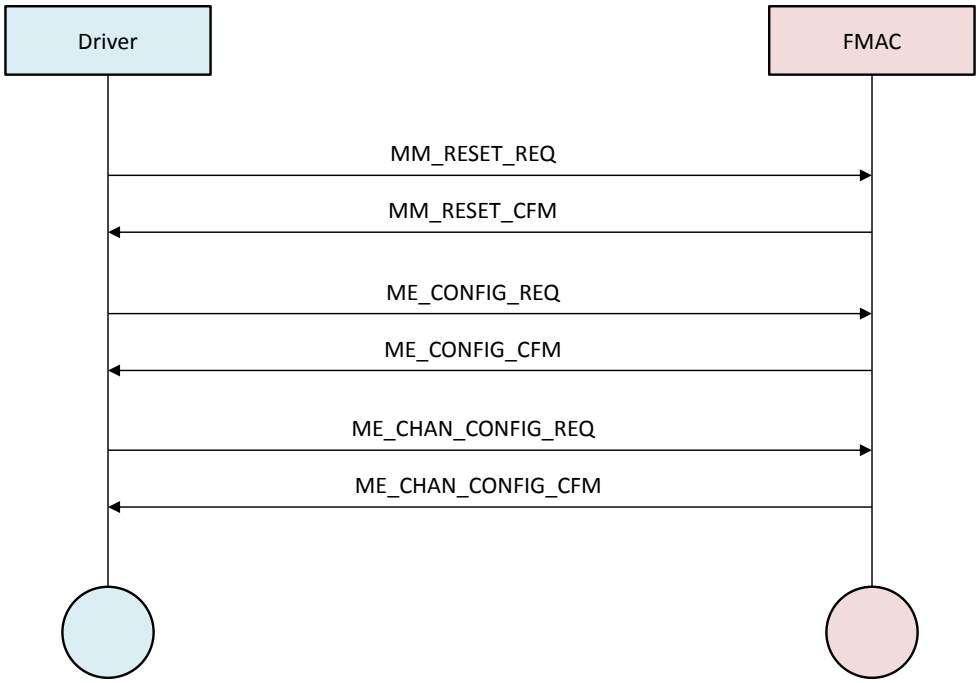

Figure 7: System initialization

4.2.2 Managing the wireless interfaces

All operational modes, except the monitor mode, require the addition of a wireless interface. This action will set a MAC address to the HW, allowing it to send the acknowledgments. It also configures the HW in the correct operational mode, i.e. STA, AP or IBSS.

Once an interface has been added, the procedures listed in the following chapters can be performed.

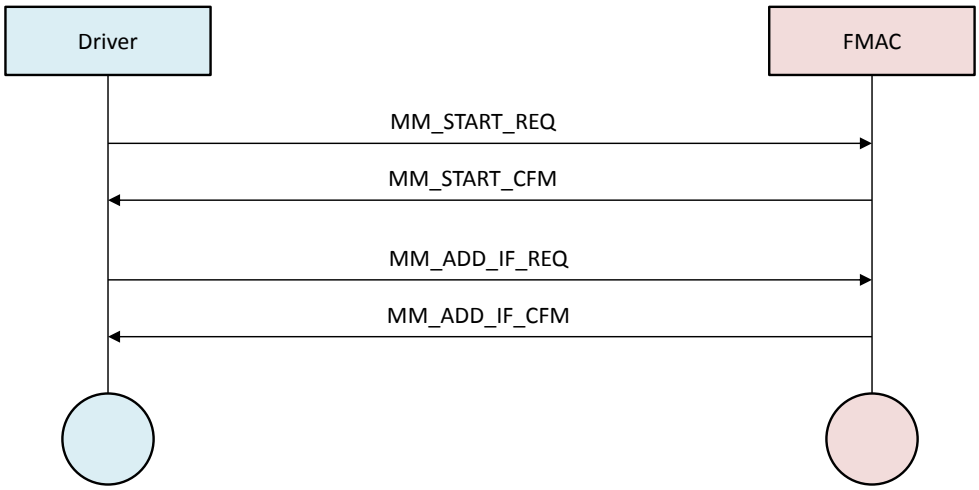

Figure 8: Adding a wireless interface

When an interface is not useful anymore, it is possible to remove it by applying the procedure below. Once all the interfaces have been removed, the system is automatically put back in monitor mode.

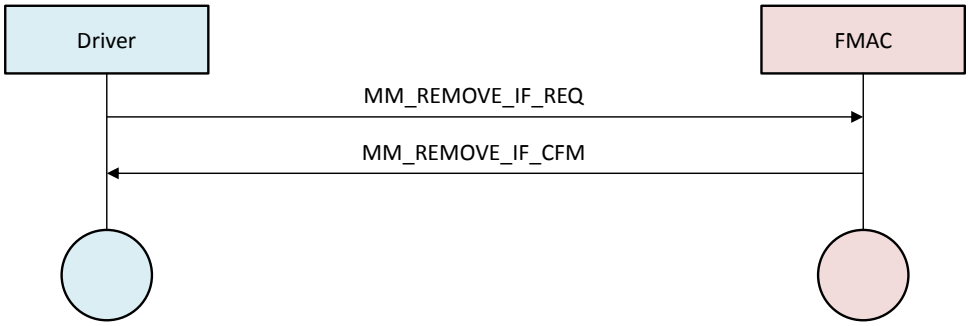

Figure 9: Removing a wireless interface

4.2.3 Scanning the wireless networks

In order to discover the WLAN networks in the neighborhood, the scanning procedure has to be performed. The scanning procedure is fully managed by the FMAC.

It can be executed when at least one interface has been added to the FMAC (see 4.2.2).

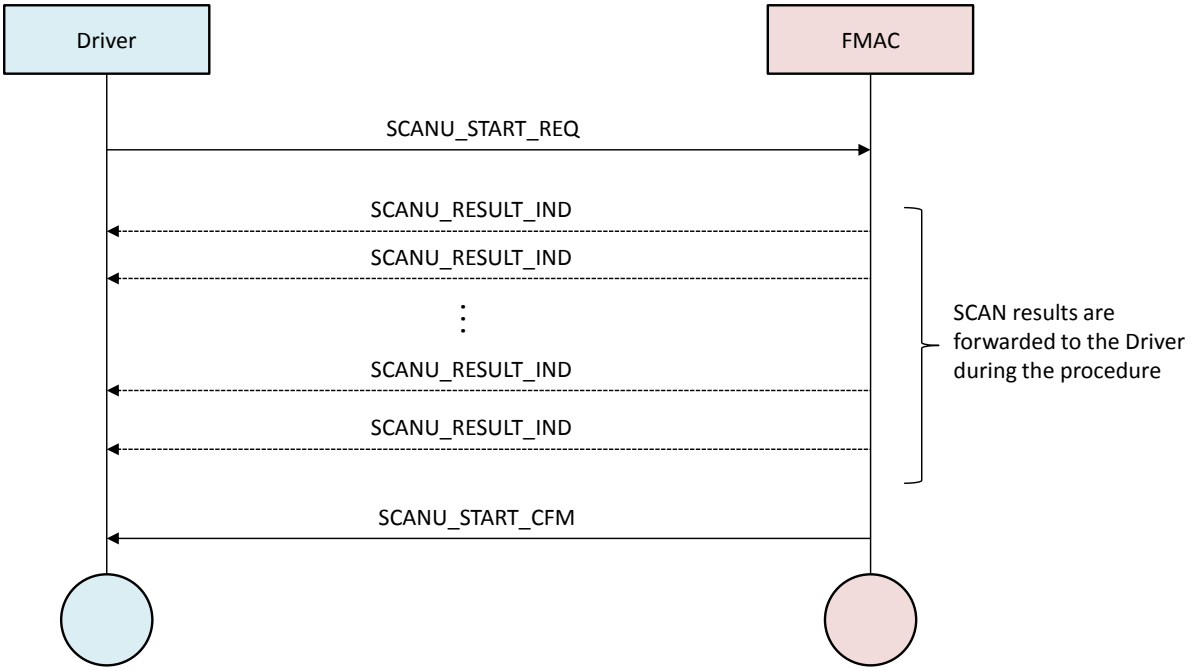

Figure 10: Scanning procedure

4.2.4 Managing the keys

When creating or associating to a BSS that is using protection, security keys have to be set to the MAC HW to allow the encryption and decryption of the transmitted and received packets.

Depending on the type of BSS security (WEP, WPA, WPA2 or WAPI), the type of key (group/default or pairwise), and the role of the device (AP or STA), the key setting procedure has to be performed at a different step:

- If WEP security is used, the WEP keys (which are default keys) are set prior to the STA association or AP starting.
- If WPA or WPA2 is used, two cases apply depending on the key type:
    - Pairwise Key: The key is set after the 4-way handshake, following the association procedure (see 4.2.5.1 for STA and for AP).
    - Group Key: In STA mode, the group key is set after the group key handshake, following the association procedure. In AP mode, the group key prior to the AP starting.

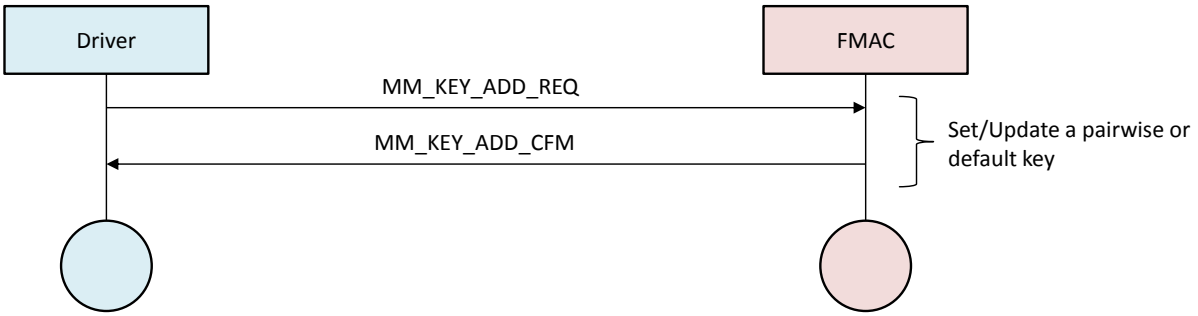

Figure 11: Key setting/update

If required, pairwise or group keys can be removed from the MAC HW:

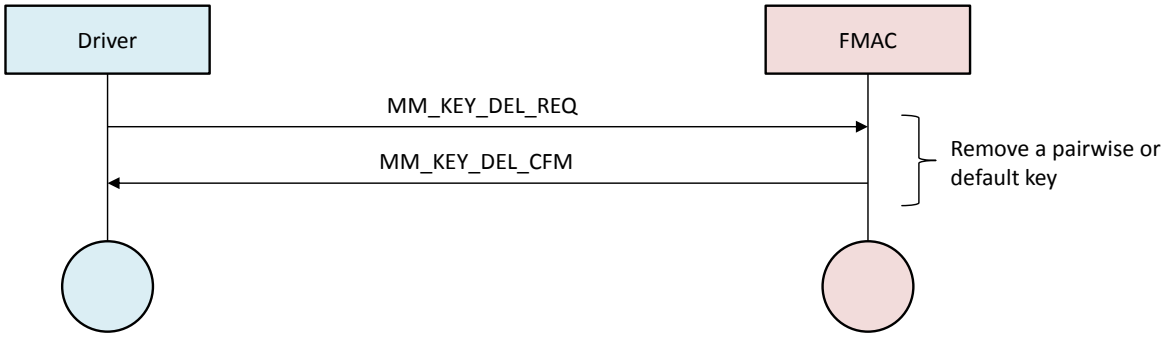

Figure 12: Key deletion

4.2.5 STA Mode

4.2.5.1 Associating to an unsecured Access Point

The association procedure to an unsecured AP is fully managed by the FMAC.

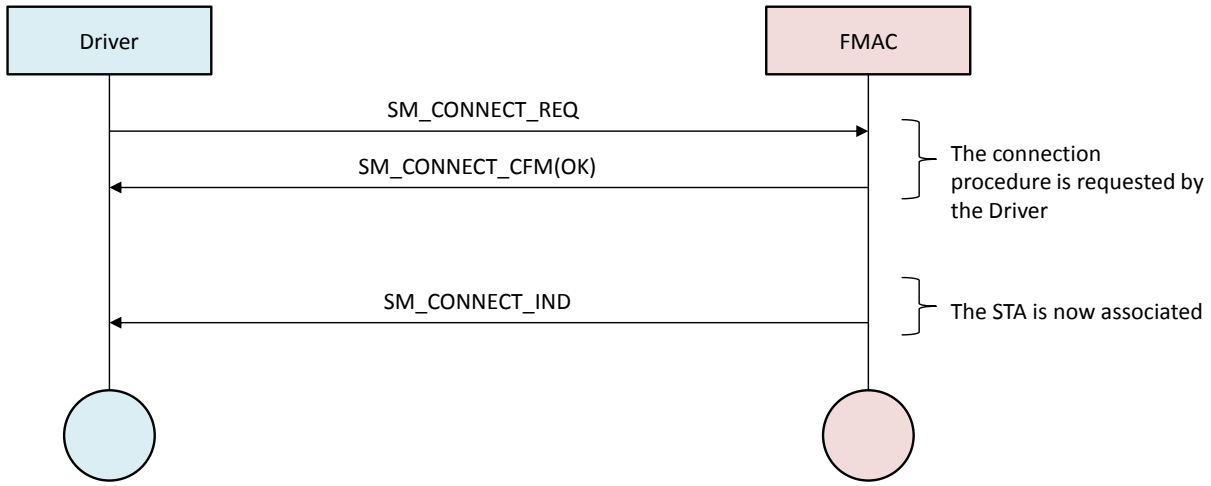

Figure 13: Association procedure (unsecured AP)

4.2.5.2 Associating to a WEP Access Point

The association procedure to a WEP AP requires to first setting the default key that will be used.

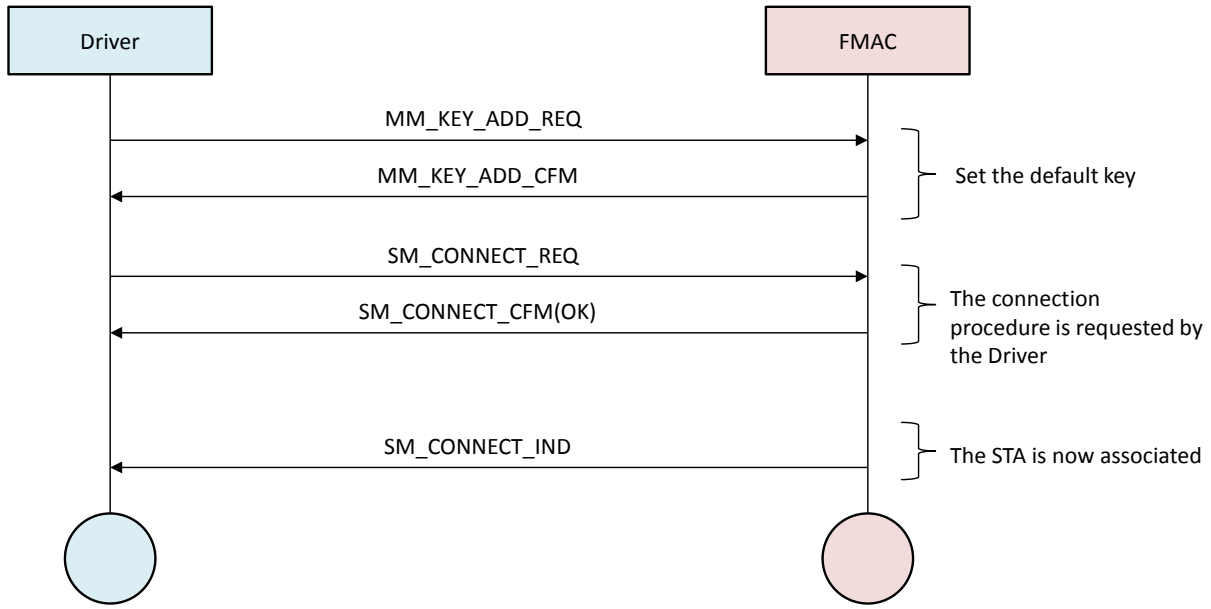

Figure 14: Association procedure (WEP AP)

4.2.5.3 Associating to a WPA/WPA2/WAPI Access Point

The association procedure to a WPA/WPA2 or WAPI AP involves the supplicant program running on the host CPU to proceed to the authentication and generate the keys.

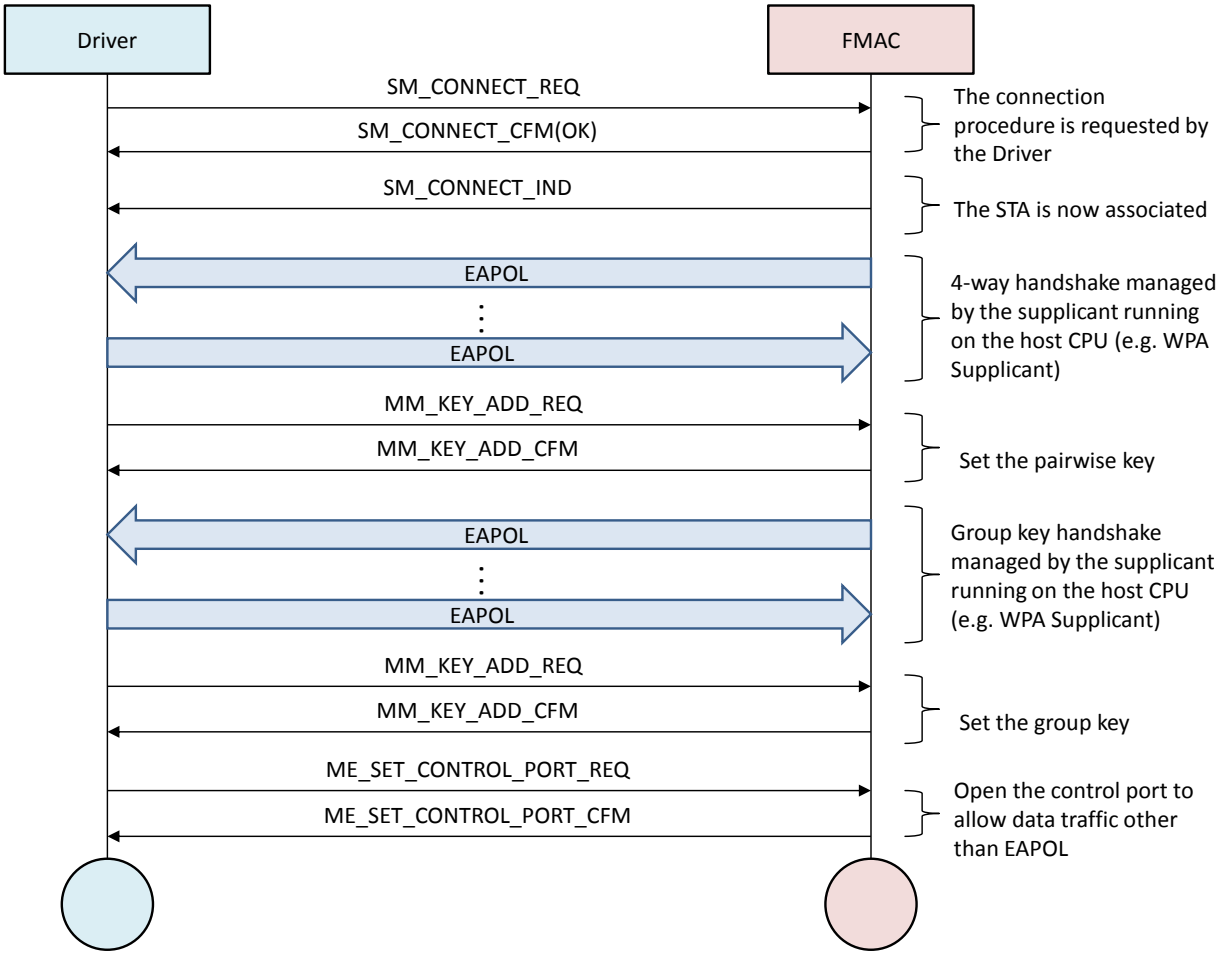

Figure 15: Association procedure (WPA/WPA2/WAPI AP)

4.2.6 AP Mode

4.2.6.1 Starting the AP

The AP mode is started by a single message sent to the FMAC. The FMAC will then manage the beacon transmissions.

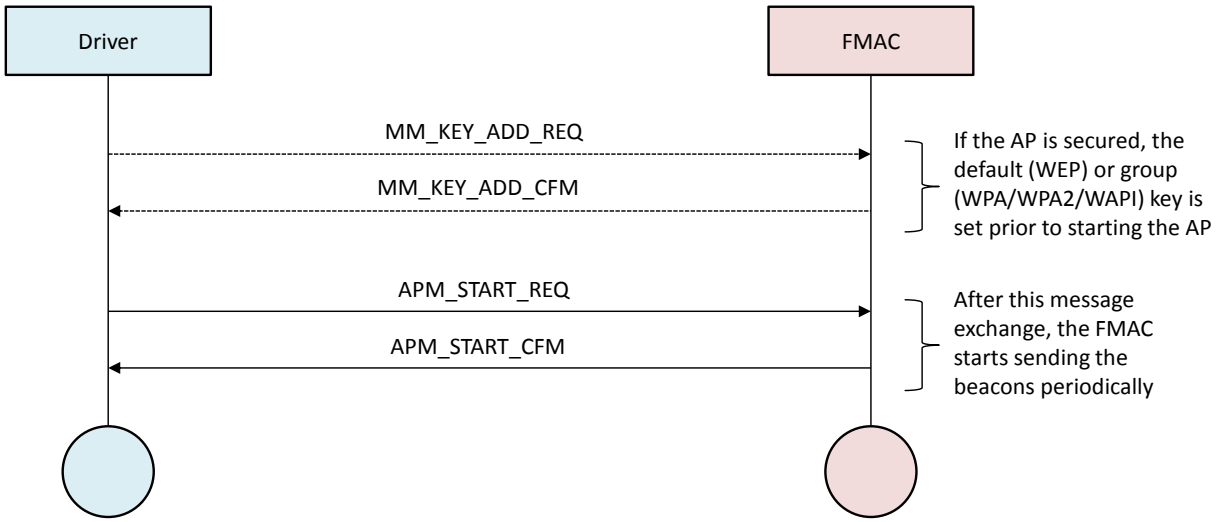

Figure 16: Starting the AP

4.2.6.2 Updating the beacon

In case the host application managing the AP requires modifying the beacon, the following message is sent to the FMAC.

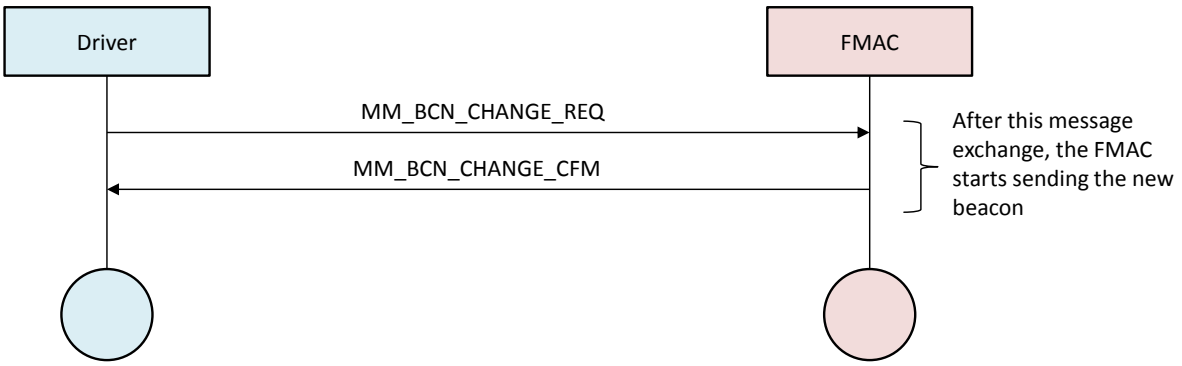

Figure 17: Updating the beacon

4.2.6.3 Stopping the AP

An AP running can be stopped via the following message.

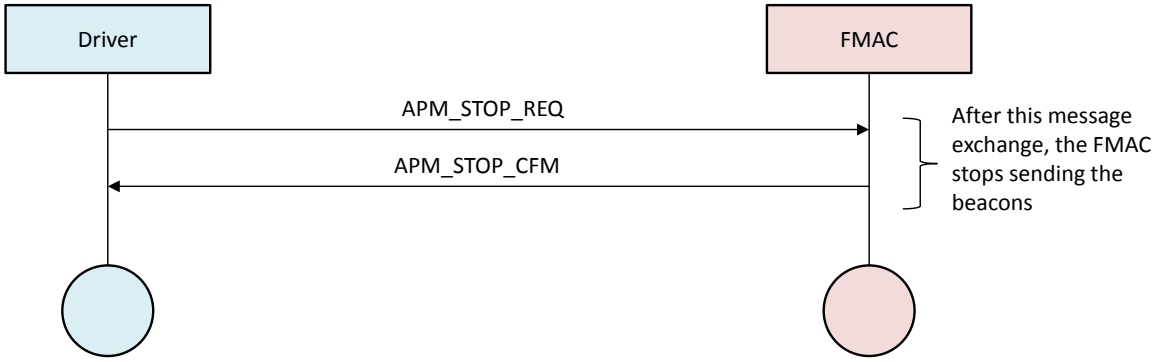

Figure 18: Stopping the AP

4.2.6.4 Associating a station

When a STA desires to associate to the AP, the following procedure applies:

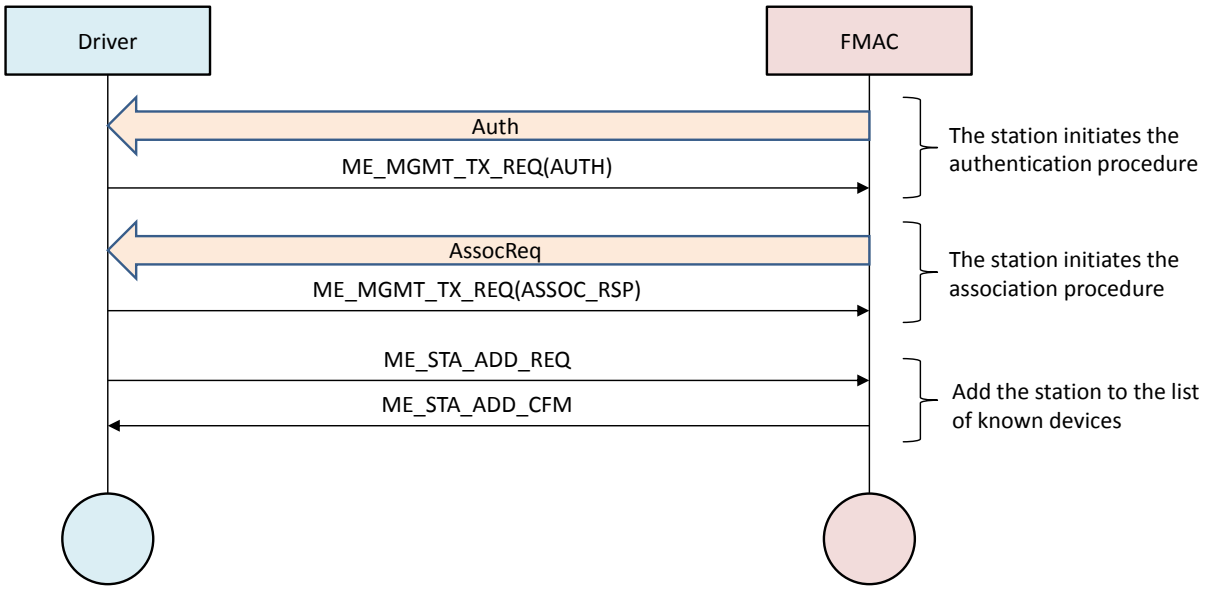

Figure 19: Associating a STA to the AP

If the AP is secured using WPA/WPA2 or WAPI the 4-way handshake is initiated immediately after the association procedure in order to generate the pairwise key and distribute the group key to the new station.

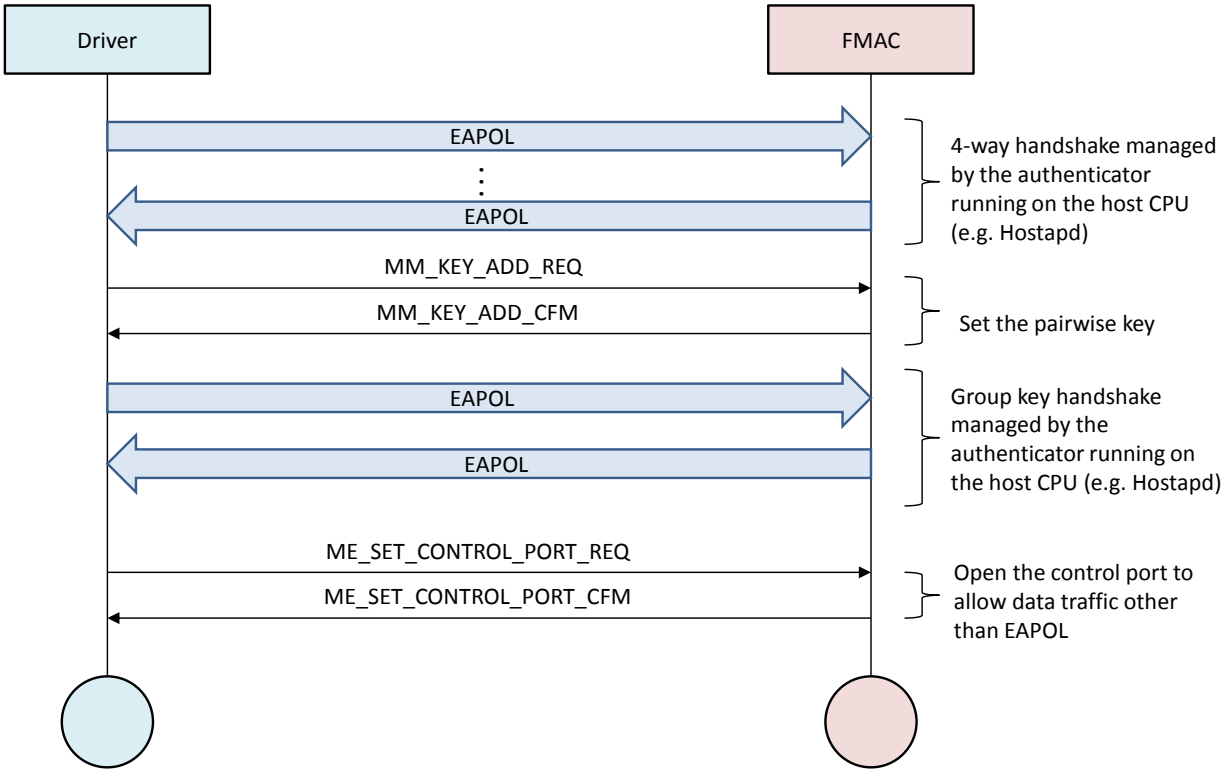

Figure 20: 4-way handshake in AP mode

References

[1] Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications, IEEE Std 802.11™-2012

[2] Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: Enhancements for Higher Throughput, IEEE Std 802.11n™-2009

[3] RW-WLAN-nX-MAC-HW User Manual Document (RW-WLAN-nX-MAC-HW-UM/1.03)

[4] RW-WLAN-nX-LMAC-SW User Manual Document (RW-WLAN-nX-LMAC-SW-UM/1.04)