Marvell Solutions

WIRELESS

WLAN Subsystem

Host Driver, Firmware, and Interface

IEEE 802.11a/g/b

Firmware Specification v5.1

Doc. No. MV-S103752-00, Rev. –
May 22, 2006
Document Classification: Proprietary

MOVING FORWARD
FASTER®

MARVELL®

Document Conventions

↘ **Note**

Provides related information or information of special importance.

! **Caution**

Indicates potential damage to hardware or software, or loss of data.

⚡ **Warning**

Indicates a risk of personal injury.

Document Status

Doc Status: 2.00	Technical Publication: 0.xx

For more information, visit our website at: www.marvell.com

No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying and recording, for any purpose, without the express written permission of Marvell. Marvell retains the right to make changes to this document at any time, without notice. Marvell makes no warranty of any kind, expressed or implied, with regard to any information contained in this document, including, but not limited to, the implied warranties of merchantability or fitness for any particular purpose. Further, Marvell does not warrant the accuracy or completeness of the information, text, graphics, or other items contained within this document.

Marvell products are not designed for use in life-support equipment or applications that would cause a life-threatening situation if any such products failed. Do not use Marvell products in these types of equipment or applications.

With respect to the products described herein, the user or recipient, in the absence of appropriate U.S. government authorization, agrees:

1) Not to re-export or release any such information consisting of technology, software or source code controlled for national security reasons by the U.S. Export Control Regulations ("EAR"), to a national of EAR Country Groups D:1 or E:2;

2) Not to export the direct product of such technology or such software, to EAR Country Groups D:1 or E:2, if such technology or software and direct products thereof are controlled for national security reasons by the EAR; and,

3) In the case of technology controlled for national security reasons under the EAR where the direct product of the technology is a complete plant or component of a plant, not to export to EAR Country Groups D:1 or E:2 the direct product of the plant or major component thereof, if such direct product is controlled for national security reasons by the EAR, or is subject to controls under the U.S. Munitions List ("USML").

At all times hereunder, the recipient of any such information agrees that they shall be deemed to have manually signed this document in connection with their receipt of any such information.

Table of Contents

THIS PAGE INTENTIONALLY LEFT BLANK

List of Tables

Section 1. Introduction

This document contains the specifications for the interface between the host driver and the firmware that powers Marvell® Wireless LAN system-on-chip (SoC) products for client (station) applications.

Note

This specification uses little-endian byte order.

1.1 Theory of Operation

1.1.1 Establishing a link with the AP

When the host driver wants to establish a link with a particular AP, it issues a command to the firmware to initiate the joining process.

When joining an AP, the station must pass two processes:

- Authentication process
- Association process

1.1.2 Terminating the Link with the AP

When the host driver wants to terminate the link with a particular AP, it issues a command to the firmware to initiate the de-authentication process.

1.1.3 Joining an Existing Ad-Hoc Network

When the host driver wants to join an existing Ad-Hoc network, it issues a command to the firmware to initiate the joining process. When joining an Ad-Hoc network, the flow would be the same as joining an Infrastructure except that it does not initiate the Authentication and Association processes.

1.1.4 Starting a Non-Existing Ad-Hoc Network

When the host driver wants to create a new Ad-Hoc network, it issues a command to the firmware to initiate the start process. This command should only be sent if no other networks exist that meet the driver criteria to join with.

1.1.5 Terminating Membership in an Ad-Hoc Network

When the host driver wants to terminate its membership in an Ad-Hoc network, it issues a command to the firmware to initiate the termination process.

1.1.6 Scanning for Existing Networks Within the Proximity

When the host driver wants to discover wireless networks within its proximity, it issues a command to the firmware to initiate the scanning process. The host driver can specify if the scan is that of an active or passive scan. The flow of the scan process is the same except that there are no probe requests sent out during a passive scan. Also the driver can specify specific filters based on SSID, BSSID, or Type.

1.2 Glossary

Table 1 lists the wireless acronyms and terms used in this document.

Table 1: Acronyms and Terms

Term	Definition
802.11	A family of specifications developed by the IEEE for wireless LAN technology
AC	Access Category
AES	Advanced Encryption Standard
AIFS	Arbitration Inter Frame Space
AP	Access Point
BCA	Bluetooth Coexistence Arbitration
BSS	Basic Service Set
CCMP	Counter mode with Cipher Block Chaining Message protocol
CF	CompactFlash
CW	Contention Window
DFS	Dynamic Frequency Selection
DTIM	Delivery Traffic Indication Message
EDCA	Enhanced Distributed Channel Access The prioritized CSMA/CA access mechanism used by QSTAs in a QBSS. This access mechanism is also used by the QAP and operates concurrently with a controlled channel access mechanism based on polling.
GTK	Group Transient Key
GWK	Groupwise Key
HAL	Hardware Abstraction Layer
IFS	Inter Frame Space
IP	Internet Protocol
LAN	Local Area Network
MAC	Media Access Controller
MIC	Message Integrity Code
MPDU	MAC Protocol Data Unit
MSDU	MAC Service Data Unit
PA	Power Amplifier
PCI	Peripheral Component Interconnect
PS	Power Save

Table 1: Acronyms and Terms (Continued)

Term	Definition
PSK	Pre-Shared Key
PTK	Pairwise Transient Key
PWK	Pairwise Key
QAP	QoS Enhanced Access Point
QBSS	QoS Enhanced Basic Service Set
QoS	Quality of Service
QSTA	QoS Enhanced Station A label for those MSDUs for which a common set of EDCA parameters are used for contending the channel. Each QSTA has four ACs.
RFI	Radio Frequency Interference
RSN	Robust Secure Network
SDIO	Secure Digital Input/Output
SNR	Signal to Noise Ratio
SoC	System-on-Chip
SQU	Internal SRAM Unit
SSID	Service Set ID A 32-character unique identifier attached to the header of packets sent over a WLAN that acts as a password when a mobile device tries to connect to the BSS.
STA	Station (Client)
TBTT	Target Beacon Transmission Time
TC	Traffic Class; For example, AC (Admission class)
TCP	Transmission Control Protocol
TLV	Tag Length Value
TPC	Transmit Power Control (802.11h)
TSC	TKIP Sequence Counter
UDP	User Datagram Protocol
UINT16	16-bit unsigned Integer
UINT32	32-bit unsigned Integer
UINT8	8-bit unsigned Integer
WCB	Wireless Control Block
WEP	Wired Equivalent Privacy
WLAN	Wireless Local Area Network

Table 1: Acronyms and Terms (Continued)

Term	Definition
WMM	Wi-Fi Multimedia
WPA	Wi-Fi Protected Access

Section 2. Driver and Firmware

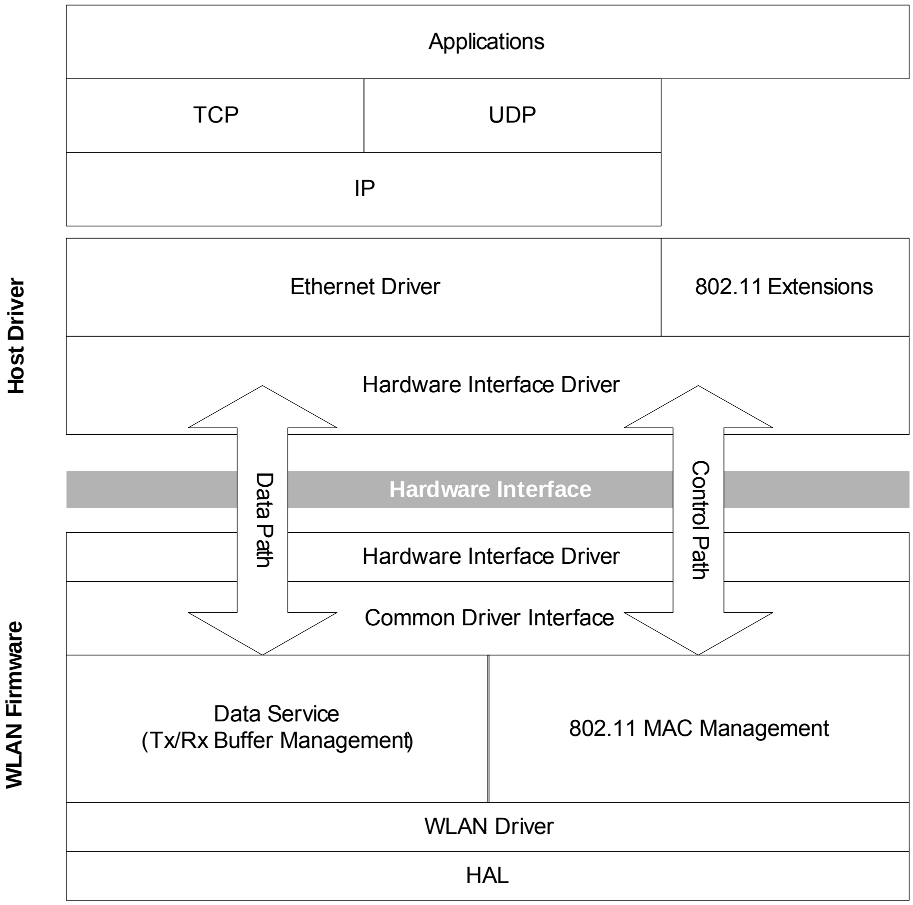
2.1 Driver and Firmware Architecture

A simplified view of the overall architecture of a typical host driver and the basic architecture of the device firmware is shown in Figure 1. The Wireless LAN (WLAN) firmware architecture is modular and allows the porting of some modules (i.e., 802.11 MLME) to reside in the host driver.

This basic architecture is typical of a thick firmware architecture, where the WLAN firmware handles all 802.11 MAC Management tasks.

- The host driver downloads standard 802.3 frames to the firmware to transmit over the wireless link as 802.11 frames.
- The WLAN firmware processes the received 802.11 frames and converts them into 802.3 frames before forwarding them to the host driver.

Figure 1: Standard Host Driver and WLAN Firmware Architecture



2.1.1 Host Driver Modules

The host driver modules are:

- **Ethernet Driver:** Standard Ethernet driver.
- **802.11 Extensions:** This module extends the standard Ethernet driver in order to view and control the state of the WLAN adapter.
- **Hardware Interface Driver:** This module controls the hardware interface on the host side.

2.1.2 Host Driver Control Pipes into Firmware

The host driver control pipes into the firmware are:

- **Command Requests:** The host driver can download command requests to the WLAN firmware using the control path, (see Section 4. "Control Path" on page 27).
- **Embed Control Directives:** The host driver can embed control directives in each data packet it downloads to the WLAN firmware for transmission over the air. Similarly, the WLAN firmware embeds WLAN receive information into each data packet uploaded to the host driver, (see Section 3. "Data Path" on page 21).

2.1.3 Firmware Modules

The available firmware modules are:

- **Hardware Interface Driver:** Controls the hardware interface on the WLAN system on a chip (SoC) side.
- **Common Driver Interface:** Supports the common driver interface.
- **Data Service:** Manages data buffers on transmit and receive paths and their flow. This module also handles any frame type conversion, such as 802.11 to 802.3.
- **802.11 MLME:** Handles 802.11 Medium Access Controller (MAC) layer management tasks.
- **WLAN Driver:** Manages the WLAN MAC interrupts and controls the hardware MAC.
- **HAL:** This module is the hardware abstraction layer.

2.2 Driver and Firmware After Card Insertion

This section describes the interaction between the host driver and the firmware after a WLAN card is inserted. After a card has been inserted, the host driver:

1. Downloads firmware into the WLAN module.
2. Waits until the firmware completes its boot sequence and indicates that it is ready.
   If the device does not contain EEPROM, then it needs to download EEPROM information at this time. This can be accomplished by calling **CMD_802_11_CAL_DATA_EXT**.
3. Sends the scan command to the firmware to get the list of available APs.
4. Sends Authenticate command to the firmware to prepare the firmware for association with the specified AP.
5. Sends the Associate command to the firmware to start the association process with the specified AP.

2.2.1 Firmware Download

2.2.1.1 SDIO Interface

The firmware download procedure using SDIO differs by the type of firmware used. If the firmware is single stage only, the single stage procedure needs to be completed. If the firmware type is two stage then the helper image procedure needs to be completed, along with the second stage firmware download procedure.

Note

See the SDIO section of the *Host Interface Register* document for the associated SoC.

To download the Helper download/single stage firmware:

1. Host checks if the firmware has already been downloaded. The host reads the Scratch Pad register for FN0 and FN1 to see if the values are 0xDC and 0xFE respectively. These values indicate that firmware has been successfully downloaded and is active.
   - If these values are returned the firmware has already been successfully download and is active. This procedure does not need to be done.
   - If these values are not returned the firmware has not been download, continue to the next step.
2. Host polls the Card Status register (0x20[3]; 0x20[2]).
3. Host checks if the current block is the last block:

   a) If the current block is the last block, the block length is adjusted to the exact length of the remaining bytes.

   b) If the current block is not the last block, then the length of data downloaded in this iteration is 60 (2*32 bytes - 4 byte header).
4. Host starts the transfer of firmware blocks.

   Each block is currently set at 32 bytes. The download length is set to 64 bytes (2 blocks x 32 bytes/block = 4 bytes header) in each iteration for CMD53 write.

   CMD53 is issued with the Block mode and the fixed address as the arguments and by writing into the I/O port register.
5. Repeat Steps 2–4 until the entire helper is downloaded.
6. Download the last packet with the length set to 0 to indicate end of data.
7. Wait 1 second for the helper to be ready.

To download the second stage firmware:

1. Host polls the Card Status register (0x20[3]; 0x20[2]).
2. Host reads the HOST_F1_RD_BASE_0 and HOST_F1_RD_BASE_1 registers (0x10 and 0x11) to get the data length set by helper for this iteration.

   If the length requested by helper is larger than 512 bytes, it is cut into multiple pieces for CMD53 write. The current download length is set to 512 bytes (16 blocks x 32 bytes per block) in each iteration of CMD53 write.
3. Host starts the download of 16 blocks of firmware (512 bytes).
4. Host copies the payload to the buffer.
5. Host writes 16 blocks of the firmware image data using CMD 53.
6. Repeat Steps 3 through 5 until the firmware image data specified by the helper (Step 2) for this iteration is downloaded completely.
7. Repeat Steps 2 through 6 until the entire firmware is downloaded completely.

To verify the firmware was properly downloaded:

1. The host reads the Scratch Pad register for FN0 and FN1 to see if the values are 0xDC and 0xFE, respectively. This indicates that firmware has been successfully downloaded and is active.

2.2.1.2 CF Interface

The downloading firmware procedures using the CF interface differs by the type of firmware used. If the firmware is single stage, only the single stage procedure needs to be completed. If the firmware type is second stage then the helper image procedure needs to be completed, along with the second stage firmware download procedure.

To download the Helper download/single stage firmware:

1. Host checks the Card Configuration and Scratch Port register (0x3E).
   - If this value is 0x5A, the firmware is already downloaded successfully.
   - If this value is 0x00, the boot loader is ready to begin firmware download, continue to Step 2.
   - If the value is none of the above, it is invalid value of this register.
2. Set the 5 bytes of the helper image to value of 0 (zero) for the CF interface.
3. The Helper download procedure starts:
   a) Write the number of bytes to be sent to I/O Command Write Length register (0x18) as part of a single block. The numbers of bytes to be sent is 256.
   b) Write this block to I/O Command Port register (0x1A) as 16-bit word write.
   c) Assert the Download Over Interrupt command in the Host Status register (0x00).
   d) Assert the Download Over Interrupt command in the Card Interrupt Cause register (0x02).
   e) The host polls the Card Status register (0x20[2]). The host polls this for 50 ms before declaring a failure on this particular block of firmware being downloaded.
4. Repeat Steps a to e till the entire firmware is downloaded.
5. The host sets the I/O Command Write Length register (0x18) to 0 (zero).
6. Assert the Download Over interrupt command in the Host Status register (0x00).
7. Assert the Download Over interrupt command in the Card Interrupt Cause register (offset 0x02).

To download the second stage firmware:

1. The host polls the Internal SRAM Unit (SQU) Read Base Address Low register (0x28) for a maximum of 10 ms.
   - If the value read from this register is 0x10, go to Step 2. This is the length of the firmware block that the helper is asking the host to download and needs to be downloaded to the card.
   - If the value is not 0x10, the host needs to timeout because the helper is not properly downloaded and is not ready in the prescribed time.
2. Check if the value is an odd integer:
   - If the value is an odd integer, retry downloading the same block of the firmware with the retry limit ≤ 20. If the retry limit > 20, firmware download cannot be completed. The firmware download is exited.
   - If the value is not odd, continue to Step 3.
3. Increment the length of the firmware block by 1.
4. Write the length of the firmware block in Step 3 into the I/O Command Write Length register (0x18).
5. Write firmware block in Step 3 to I/O Command Port register (0x1A) as 16-bit word writes.
6. Assert the Download Over interrupt command in the Host Status register (0x00).
7. Assert the Download Over interrupt command in the Card Interrupt Cause register (0x02).
8. Host polls the Card Status register (0x20[2]). There is no definite time period that the host polls this bit before declaring a failure on this particular block of firmware. This bit is set almost immediately by the already downloaded Helper.

9. Host polls Card Configuration and Scratch Port register (0x3E) till a value of 0x5A is set in it by the device. This indicates that firmware initialization is done. The host polls this for 50 ms before declaring a failure on this particular block of firmware being downloaded.

Note

See the CompactFlash section of the *Host Interface Register* document for the associated SoC.

2.2.1.3 G-SPI Interface

The downloading firmware using the G-SPI host interface differs by the type of firmware used. It the firmware is single stage only the single-stage procedure is required. If the firmware type is second stage then the helper image procedure is required, along with the second stage firmware download procedure.

To download the Helper download/single stage firmware:

1. Device (boot code) writes to Command Write Base Address register (0x14) to set the location for firmware download.
2. Device (boot code) writes 1 Host Interrupt Cause register (0x22[2]) to indicate that the device is ready for firmware download.

   Scratch pad 1 (0x0028) should contain the number of bytes downloaded to the firmware in the current iteration.
3. Wait for Host Interrupt Status register (0x5C[2]).
4. Write the data into the Command Read/Write Port register (0x18).
5. Clear the Host Interrupt Status register (0x5C[2]) to get the next interrupt.
6. Interrupt the device (boot code) by setting the Host Interrupt Status register (0x5C[2]).
7. Device (boot code) reads Host Interrupt Status register (0x5C) to determine the interrupt cause.
8. Device (boot code) moves data from SQU to I-TCM
9. Device (boot code) writes 0 to the Card Interrupt Cause register (0x02[2]) to re-enable interrupt.
10. Repeat Step 2 through 10 until all of the helper image is downloaded into the card.
11. Once the helper/single stage firmware download is complete, write zero to Scratch pad 1 and interrupt the bootloader. This completes the helper download.

Notes

- Up to 1024 bytes can be downloaded at a time in 1 iteration. 64 bytes at a time is recommended.
- If the firmware is a single stage firmware, verify the single-stage firmware download. The second stage firmware download is skipped.

To download the second stage firmware:

1. When the helper firmware has been completely downloaded the host writes 0 to Scratch Pad register 1 (0x28) to indicate that the helper-download is complete.
2. Wait until a non-zero number appears in Scratch Pad register 1 (0x28). The host must poll this register until it receives a non-zero number, say once in 100 µs.
3. The host verifies that the device has set the Host Interrupt Status register (0x5C[2]).
4. The host reads Scratch Pad register 1 (0x28) to determine how many bytes of firmware needs to be written (len).
   - If len = 0, then end (exit) of download.
   - If len = 1, then the previous downloaded data had CRC errors. In this case download the previous data again.

- If len is not equal to 0 and len is not equal to 1, write the len number of bytes of data into the Command Read/Write Port register (0x18).

5. Clear the Host Interrupt Status register (0x5C) to prepare for the next interrupt.
6. Interrupt the bootloader by setting the Host Interrupt Status register (0x5C[2]).
7. Continue Step 2 through 6 until len = 0 is achieved.

To verify the firmware was properly downloaded:

1. Read Scratch Pad 4 register (0x34). The value read should be 0x88888888. This value means the firmware was successfully downloaded and is running.

Note

This firmware should be downloaded only after the helper image has successfully being downloaded.

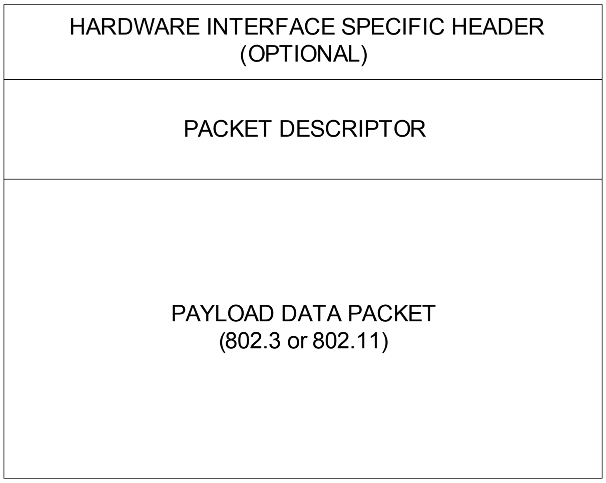
Section 3. Data Path

Each data packet (either 802.3 or 802.11) transferred across the hardware interface begins with a hardware specific header (optional) followed by the packet descriptor (required).

The hardware interface specific header is optional when the hardware interface supports data and command channels. Otherwise, it is mandatory. The header is needed to distinguish command packets from data packets on hardware interfaces that do not support channels with commands separate from the data channel (SDIO interface).

Figure 2 shows the general packet structure. The structure of the packet descriptor for packets sent from the firmware to the host is different from the structure of the packet descriptor sent from the host to the firmware.

Figure 2: General Packet Structure


HARDWARE INTERFACE SPECIFIC HEADER (OPTIONAL)
PACKET DESCRIPTOR
PAYLOAD DATA PACKET (802.3 or 802.11)

3.1 Receive Packet Descriptor

The structure of the Receive Packet Descriptor is:

Byte 3	Byte 2	Byte 1	Byte 0
RxControl	SNR	RxStatus	
RxRate	NF	RxPacketLength	
RxPacketLocation			
Reserved_1			
Reserved			Priority

The fields of the Receive Packet Descriptor are defined in Table 2.

Table 2: Fields in Receive Packet Descriptor

<table>
<tr><th>Field Name</th><th>Type</th><th>Description</th></tr>
<tr><td>RxStatus</td><td>UINT16</td><td>Reserved</td></tr>
<tr><td>SNR</td><td>UINT8</td><td>Signal to noise ratio for this packet
This should be a positive value (dB).</td></tr>
<tr><td>RxControl</td><td>UINT8</td><td>Reserved</td></tr>
<tr><td>RxPacketLength</td><td>UINT16</td><td>The number of bytes in the payload</td></tr>
<tr><td>NF</td><td>UINT8</td><td>Noise floor for this packet (dBm)
Noise floor is always negative. The absolute value is passed.</td></tr>
<tr><td>RxRate</td><td>UINT8</td><td>Rate at which this packet is received:
<table>
<tr><th>TxControl [3:0]</th><th>Data Rate (Mbps)</th></tr>
<tr><td>16:13</td><td>Reserved</td></tr>
<tr><td>12</td><td>54</td></tr>
<tr><td>11</td><td>48</td></tr>
<tr><td>10</td><td>36</td></tr>
<tr><td>9</td><td>24</td></tr>
<tr><td>8</td><td>18</td></tr>
<tr><td>7</td><td>12</td></tr>
<tr><td>6</td><td>9</td></tr>
<tr><td>5</td><td>6</td></tr>
<tr><td>4</td><td>Reserved</td></tr>
<tr><td>3</td><td>11</td></tr>
<tr><td>2</td><td>5.5</td></tr>
<tr><td>1</td><td>2</td></tr>
<tr><td>0</td><td>1</td></tr>
</table>
</td></tr>
<tr><td>RxPacketLocation</td><td>UINT32</td><td>Offset from the start of the packet to the beginning of the payload data packet</td></tr>
</table>

Table 2: Fields in Receive Packet Descriptor (Continued)

Field Name	Type	Description
Reserved_1	UINT32	Reserved
Priority	UINT8	Specifies the user priority of received packet
Reserved	UINT8[3]	Reserved

Note

RSSI should be calculated in the driver as a summation of the SNR and NF (see Table 2).

Notes

- Driver gets 802.3 frames only.
- Driver implementations must account for LLC SNAP header in the RX data buffer before passing on the frame to the higher layers (e.g. Ethernet). Failure to do so can result in unexpected application behavior.

3.2 Transmit Packet Descriptor

The structure of the Transmit Packet Descriptor (TXPD) is:

<table>
<tr><td>Byte 3</td><td>Byte 2</td><td>Byte 1</td><td>Byte 0 0</td></tr>
<tr><td colspan="4">TxStatus</td></tr>
<tr><td colspan="4">TxControl</td></tr>
<tr><td colspan="4">TxPacketLocation</td></tr>
<tr><td colspan="2">TxDestAddrHigh</td><td colspan="2">TxPacketLength</td></tr>
<tr><td colspan="4">TxDestAddrLow</td></tr>
<tr><td colspan="2">Reserved</td><td>Flags</td><td>Priority</td></tr>
</table>

The Transmit Packet Descriptor fields are defined in Table 3 through Table 4.

Table 3: Fields in Transmit Packet Descriptor

Field Name	Type	Description
TxStatus	UINT32	Reserved
TxControl	UINT32	The host driver uses this field to specify the per-packet transmit parameters TxControl bits are defined in Section 3.2.1 "Per-Packet Settings" on page 25.
TxPacketLocation	UINT32	Offset of the beginning of the payload data packet (802.3 or 802.11 frames) from the beginning of the packet (bytes)
TxPacketLength	UINT16	Number of bytes in the payload data frame
TxDestAddrHigh	UINT16	Destination MAC address bytes 4 to 5
TxDestAddrLow	UINT32	Destination MAC address bytes 0 to 3
Priority	UINT8	Specifies the user priority of transmit packet
Flags	UINT8	See Table 4, “Transmit Packet Flags,” on page 24
Reserved	UINT8[2]	Reserved

Table 4: Transmit Packet Flags

Bits	Descriptions
7:4	Reserved
3	Last packet indicator (used when sleep period is set)
2	Power management bit value (if Bit[1] is set)
1	Set to 1 to override firmware power management bit
0	Set to 1 to transmit a null data packet (firmware *manufactures* the packet)

3.2.1 Per-Packet Settings

Three parameters are used to set per-packet basis:

- Transmission rate
- ACK policy
- Retry limit

Table 5 displays the TxControl bits in the per-packet settings.

Table 5: TxControl Bits

Bits	Usage
16:15	Reserved
14:13	Per-packet ACK policy
12:8	Per-packet retry limit
7:5	Reserved
4:0	Per-packet rate

3.2.1.1 Setting Per-Packet Transmission Rates

The host driver specifies the transmission rate per-packet by setting bit 4 in the TxControl field in the TxDescriptor. If bit 4 is set, the transmission rate is indicated by bits [3:0] on the Tx Control field. Table 6 shows the rate encoding of the Tx Control.

Table 6: Rate Encoding

TxControl [3:0]	Data Rate (Mbps)
16:13	Reserved
12	54
11	48
10	36
9	24
8	18
7	12
6	9
5	6
4	Reserved
3	11
2	5.5
1	2
0	1

Table 7 shows the rules for setting the per-packet data rate. The rules are needed to resolve conflicts between the global data rate setting and the per-packet data rate setting in the Tx Descriptor.

Table 7: Setting Per-Packet Transmission Rates Rules

TxControl [4]	Description
0	WLAN firmware ignores bits in TxControl [3:0] and sets the rate according to the global data rate settings
1	WLAN firmware sets the Tx rate per-packet to the rate specified in TxControl [3:0]

3.2.1.2 Setting Per-Packet ACK Policy

The host driver can set the ACK policy per-AC (access category) or per-packet. The host driver uses the API CMD_802_11_ACK_POLICY to set the ACK policy for an AC. Bits[14:13] of the TxControl field in the TX descriptor of each data packet, can be used to specify the ACK policy on a per-packet basis. This is only possible if WMM is enabled and QoS type frames are being generated by firmware. If WMM is not enabled, these bits have no effect.

Table 8 shows the functions of the per-packet ACK policy control bits in the TxControl field.

Table 8: ACK Policy Control Bits in TxControl

TxControl[14:13]	Description
10	Use Immediate ACK policy for this packet
11	Use No ACK policy for this packet
0x	Use the Per-AC ACK policy setting

3.2.1.3 Setting Per-Packet Retry Limit

The host driver can set the retry limit globally or per-packet. The host driver uses the command **CMD_802_11_SNMP_MIB** to set the global retry limit. Bits[12:8] of the TxControl field in the TX descriptor of each data packet, specifies the retry limit for that packet.

Table 9 shows the functions of the per-packet retry limit control bits in the TxControl field.

Table 9: Retry Limit Control Bits in TxControl

TxControl[12]	Description
0	Use global retry limit setting
1	Use retry limit specified in TxControl[11:8]

The retry limit specified in TxControl[11:8] is the sum of the number of retries and the first packet set out. Valid range is between 1 and 15.

Section 4. Control Path

This section specifies the format for command packets and the command exchange protocol between the host driver and the firmware.

The command packet format and the command exchange protocol between host driver and firmware are independent of the type of hardware interface. However, the mechanism for transporting the command packets across the hardware interface is hardware dependent.

4.1 Packet Format

The command packet (request or response) consists of a fixed-size header and variable-size body.

The packet header contains the command code, the size of the command packet, the sequence number, and the result. The result field is only meaningful in the response packet. The header is 8-byte in size. The most significant bit in the command code for a request is always set to 0. The most-significant bit in the command code for a response is always set to 1.

The command body contains data for the command. The maximum size of the body is 256 bytes. The structure of the body depends on the type of command.

Table 10 shows the common structure of the command packet.

Table 10: Common Command Packet Structure

Field Name	Type	Description
CmdCode	UINT16	Command code The corresponding code for the response packet is (CmdCode | 0x8000).
Size	UINT16	Size of the packet, including the header
SeqNum	UINT16	Sequence number, set by the host
Result	UINT16	Result code, set by the firmware This field is used only in the response packet. Set to 0 in the request packet.
Body	UINT8[]	Command body, containing data specific to the type of command Simple commands may omit this field.

For a detailed description of all the commands, see Section 5. "Host Commands" on page 29. For a detailed listing of Command and Result Codes, see Appendix A. "Command List" on page 135.

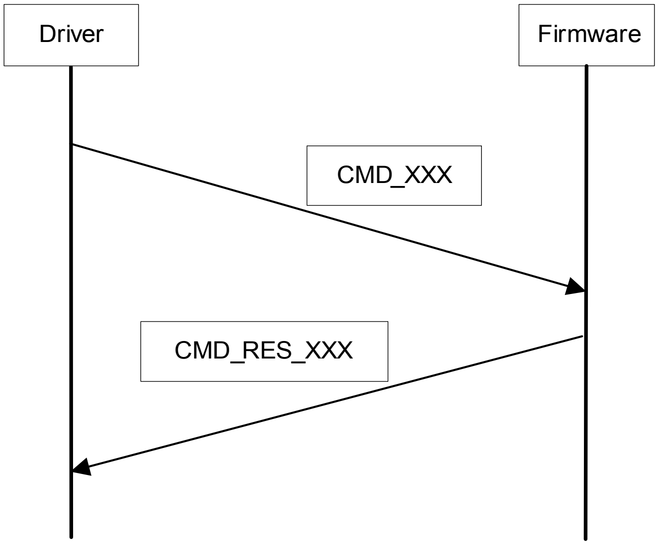
4.2 Protocol

The command exchange protocol follows the request-response model. The host driver issues a command request and the firmware sends back a command response. The command exchange protocol is serialized, the host driver must wait until it has received a command response for the current command request before it can send the next command request.

There are only a few exceptions to the request-response pair as for example the CMD_CODE_DNLD command to download firmware. This command is serviced by the boot loader in the ROM of the WLAN chip. The boot loader does not send back a command response. Each exception is specified at the corresponding host command description.

The mechanism for transporting the command request and command response packets across the hardware interface is hardware dependent.

Figure 3: Protocol Request Response Model



Section 5. Host Commands

This section describes the following:

- Interaction between the host driver and firmware after card insertion
- Commands supported

The firmware that conforms to this interface specification may elect to implement only a subset of the commands described here. The firmware shall return the response status code CMD_STATUS_UNSUPPORTED for any command request that it does not implement.

5.1 Reset and Initialization

Table 11 lists the supported commands for reset and initialization.

Table 11: Reset and Initialization Commands

Command	Description	Page
5.1.1 "CMD_802_11_RESET"	Resets the WLAN device	page 29
5.1.2 "CMD_802_11_SNMP_MIB"	Sets/gets the SNMP MIB	page 30
5.1.3 "CMD_802_11_MAC_ADDR"	WLAN MAC address	page 31
5.1.4 "CMD_MAC_MULTICAST_ADR"	Sets/gets MAC multicast filter table	page 32
5.1.5 "CMD_GSPI_BUS_CONFIG"	Sets/gets the G-SPI Bus mode and time delay between host address write and data read	page 33

5.1.1 CMD_802_11_RESET

The **CMD_802_11_RESET** command resets and power cycles the WLAN device. A firmware download is required to launch the WLAN device back to an operational state.

REQUEST

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11_RESET**
Size	UINT16	Number of bytes in command body
SeqNum	UINT16	Command sequence number
Result	UINT16	Not used (set to 0)

RESPONSE

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11_RESET** | 0x8000
Size	UINT16	Number of bytes in command body
SeqNum	UINT16	Command sequence number, sent by the host
Result	UINT16	Result code

5.1.2 CMD_802_11_SNMP_MIB

The **CMD_802_11_SNMP_MIB** command sets or gets the SNMP MIB values.

REQUEST

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11_SNMP_MIB**
Size	UINT16	Number of bytes in command body
SeqNum	UINT16	Command sequence number
Result	UINT16	Not used (set to 0)
Action	UINT16	Action: 0 = ACT_GET 1 = ACT_SET
OID	UINT16	Object identifier The following SNMP commands are defined: 0x00 = DesiredBSSType 0x01 = OpRateSet 0x02 = BcnPeriod 0x03 = DtimPeriod 0x04 = AssocRspTimeOut 0x05 = RtsThresh 0x06 = ShortRetryLim 0x07 = LongRetryLim **NOTE:** The short and long retry limits are the sum of the number of retries and the first packet set out. Valid range is between 1 and 15. 0x08 = FragThresh 0x09 = 802.11d Enable/Disable 0x0A = 802.11h Enable/Disable
Size	UINT16	Size of OID value
Value	UINT8[128]	Buffer to keep OID value

RESPONSE

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11_SNMP_MIB** | 0x8000
Size	UINT16	Number of bytes in command body
SeqNum	UINT16	Command sequence number, sent by the host
Result	UINT16	Result code
Action	UINT16	Action: 0 = ACT_GET 1 = ACT_SET
OID	UINT16	Object identifier This field is present only if action is ACT_GET.

Field Name	Type	Description
Size	UINT16	Size of OID value This field is present only if action is ACT_GET.
Value	UINT8[128]	Buffer to keep OID value This field is present only if action is ACT_GET.

5.1.3 CMD_802_11_MAC_ADDR

The host driver uses the **CMD_802_11_MAC_ADDR** command to set or get the MAC address stored in the RAM memory of the WLAN device.

REQUEST

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11_MAC_ADDR**
Size	UINT16	Number of bytes in command body
SeqNum	UINT16	Command sequence number
Result	UINT16	Not used (set to 0)
Action	UINT16	Action: 0 = ACT_GET 1 = ACT_SET
MACAddr	UINT8[6]	MAC address of WLAN device

RESPONSE

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11_MAC_ADDR** | 0x8000
Size	UINT16	Number of bytes in command body
SeqNum	UINT16	Command sequence number, sent by the host
Result	UINT16	Result code
Action	UINT16	Action: 0 = ACT_GET 1 = ACT_SET
MACAddr	UINT8[6]	MAC address of WLAN device.

5.1.4 CMD_MAC_MULTICAST_ADR

The **CMD_MAC_MULTICAST_ADR** command is used to program multicast MAC address into the hardware filter table in the WLAN SoC. The packets sent from these multicast MAC addresses are accepted.

REQUEST

Field Name	Type	Description
CmdCode	UINT16	**CMD_MAC_MULTICAST_ADR**
Size	UINT16	Number of bytes in command body
SeqNum	UINT16	Command sequence number
Result	UINT16	Not used
Action	UINT16	Action: 0 = ACT_GET 1 = ACT_SET
NumOfAdrs	UINT16	Number of multicast addresses This field is not used when Action is ACT_GET.
MACList	UINT8[32*6]	List of number of multicast addresses Maximum list size is 32. This field is not used when Action is ACT_GET.

RESPONSE

Field Name	Type	Description
CmdCode	UINT16	**CMD_MAC_MULTICAST_ADR** | 0x8000
Size	UINT16	Number of bytes in command body
SeqNum	UINT16	Command sequence number, sent by the host
Result	UINT16	Result code
Action	UINT16	Type of action set in the request message
NumOfAdrs	UINT16	Number of multicast addresses This field is not used when Action is ACT_SET.
MACList	UINT8[32*6]	List of number of multicast address Maximum list number is 32. This field is not used when Action is ACT_SET.

5.1.5 CMD_GSPI_BUS_CONFIG

The **CMD_GSPI_BUS_CONFIG** command sets or gets the G-SPI bus mode and time delay between host address write and data read. This command only applies to design platforms that support a G-SPI host interface (implemented in 88W8385 based designs).

REQUEST

<table>
<tr><th>Field Name</th><th>Type</th><th>Description</th></tr>
<tr><td>CmdCode</td><td>UINT16</td><td><b>CMD_GSPI_BUS_CONFIG</b></td></tr>
<tr><td>Size</td><td>UINT16</td><td>Number of bytes in the command body</td></tr>
<tr><td>SeqNum</td><td>UNIT16</td><td>Command sequence number</td></tr>
<tr><td>Result</td><td>UINT16</td><td>Not used (set to 0)</td></tr>
<tr><td>Action</td><td>UINT16</td><td>Action:
0 = ACT_GET
1 = ACT_SET</td></tr>
<tr><td>BusDelayMode</td><td>UINT16</td><td>Mode:
<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>15:4</td><td>Reserved</td></tr>
<tr><td>3</td><td>G-SPI clock:
0 = support G-SPI bus clock 25 MHz and under
1 = support G-SPI bus clock up to 50 MHz</td></tr>
<tr><td>2</td><td>Delay method between address phase and data phase for host read operation:
0 = HostTimeDelayToReadPort and HostTimeDelayToReadRegister specify the time delay between the rising edge of G-SPI clock (sclk) for the address LSb until the rising edge of sclk for the first valid data bit read from an G-SPI Port register. The time value is equal to the register value times 10 ns.
1 = HostTimeDelayToReadPort and HostTimeDelayToReadRegister values are in units of G-SPI clocks. They specify number of cycles of G-SPI clock between the address and data phases of a transaction. The clock signal should continue to be driven during this interval.</td></tr>
<tr><td>1:0</td><td>G-SPI data format:
00 = 8-bit address, 16-bit data
01 = 8-bit address, 32-bit data
10 = 16-bit address, 16-bit data
11 = 16-bit address, 32-bit data</td></tr>
</table></td></tr>
</table>

Field Name	Type	Description
HostTimeDelayToReadPort	UINT16	Host time delay to read G-SPI port register BusDelayMode[2] = 0: time unit = 10 ns BusDelayMode[2] = 1: time unit = G-SPI clock (sclk) typically it is a multiple of 16

HostTimeDelayToRead Register	UINT16	Host time delay to read G-SPI registers other than port register
		BusDelayMode[2] = 0: time unit = 10 ns
		BusDelayMode[2] = 1: time unit = G-SPI clock (sclk) typically multiple of 16

RESPONSE

<table>
<tr><th>Field Name</th><th>Type</th><th>Description</th></tr>
<tr><td>CmdCode</td><td>UINT16</td><td><b>CMD_GSPI_BUS_CONFIG</b> | 0x8000</td></tr>
<tr><td>Size</td><td>UINT16</td><td>Number of bytes in the command body</td></tr>
<tr><td>SeqNum</td><td>UNIT16</td><td>Command sequence number</td></tr>
<tr><td>Result</td><td>UINT16</td><td>Not used (set to 0)</td></tr>
<tr><td>Action</td><td>UINT16</td><td>Action:
0 = ACT_GET
1 = ACT_SET</td></tr>
<tr><td>BusDelayMode</td><td>UINT16</td><td>Mode:
<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>15:4</td><td>Reserved</td></tr>
<tr><td>3</td><td>G-SPI clock:
0 = support G-SPI bus clock 25 MHz and under
1 = support G-SPI bus clock up to 50 MHz</td></tr>
<tr><td>2</td><td>Delay method between address phase and data phase for host read operation:
0 = HostTimeDelayToReadPort and HostTimeDelayToReadRegister specify the time delay between the rising edge of G-SPI clock (sclk) for the address LSb until the rising edge of sclk for the first valid data bit read from an G-SPI Port register. The time value is equal to the register value times 10 ns.
1 = HostTimeDelayToReadPort and HostTimeDelayToReadRegister values are in units of G-SPI clocks. They specify number of cycles of G-SPI clock between the address and data phases of a transaction. The clock signal should continue to be driven during this interval.</td></tr>
<tr><td>1:0</td><td>G-SPI data format:
00 = 8-bit address, 16-bit data
01 = 8-bit address, 32-bit data
10 = 16-bit address, 16-bit data
11 = 16-bit address, 32-bit data</td></tr>
</table>
</td></tr>
</table>

Field Name	Type	Description

HostTimeDelayToReadPort	UINT16	Host time delay to read G-SPI port register
		BusDelayMode[2] = 0: time unit = 10 ns
		BusDelayMode[2] = 1: time unit = G-SPI clock (sclk) typically multiple of 16

HostTimeDelayToRead Register	UINT16	Host time delay to read G-SPI registers other than port register
		BusDelayMode[2] = 0: time unit = 10 ns
		BusDelayMode[2] = 1: time unit = G-SPI clock (sclk) typically multiple of 16

When the firmware gets this command from the host, the G-SPI interface configuration is changed immediately. The new configuration setting in the command takes effect right before the command response is issued to host. It is the responsibility of the host to send the command to G-SPI bus configuration and prepare for the command response using the G-SPI bus configuration.

Default values in firmware/hardware are:

- HostTimeDelayToReadPort = 0x00E0
- HostTimeDelayToReadRegister = 0x0050

5.2 MAC/PHY/RF Control

Table 12 lists the supported commands for MAC, PHY, and RF control.

Table 12: MAC/PHY/RF Control Commands

Command	Description	Page
5.2.1 "CMD_MAC_CONTROL"	Controls hardware MAC	page 36
5.2.2 "CMD_802_11_RADIO_CONTROL"	Controls the radio chip	page 38
5.2.3 "CMD_802_11_RF_ANTENNA"	Sets/gets the Tx and Rx antenna mode	page 39
5.2.4 "CMD_802_11_RF_TX_POWER"	Sets/gets Radio transmit power	page 40
5.2.5 "CMD_802_11_RF_CHANNEL"	Sets/gets RF channel	page 41

5.2.1 CMD_MAC_CONTROL

The **CMD_MAC_CONTROL** command is used to control the hardware MAC. This command is only used to set the fields described in the action field.

REQUEST

<table>
<tr><th>Field Name</th><th>Type</th><th>Description</th></tr>
<tr><td>CmdCode</td><td>UINT16</td><td><b>CMD_MAC_CONTROL</b></td></tr>
<tr><td>Size</td><td>UINT16</td><td>Number of bytes in command body</td></tr>
<tr><td>SeqNum</td><td>UINT16</td><td>Command sequence number</td></tr>
<tr><td>Result</td><td>UINT16</td><td>Not used (set to 0)</td></tr>
<tr><td>Action</td><td>UINT16</td><td>Action:
<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>15</td><td>WEP type:
0 = WEP40
1 = WEP104</td></tr>
<tr><td>14:12</td><td>Reserved</td></tr>
<tr><td>11</td><td>0 = disable WMM
1 = enable WMM</td></tr>
<tr><td>10</td><td>EnforceProtection
This bit use to enforce strict data protection of Rx and Tx.
0 = disable protection
1 = enable protection</td></tr>
<tr><td>9</td><td>Reserved</td></tr>
<tr><td>8</td><td>0 = all multicast off
1 = all multicast on</td></tr>
<tr><td>7</td><td>0 = promiscous off
1 = promiscous on</td></tr>
<tr><td>6</td><td>Reserved and set to 0</td></tr>
<tr><td>5</td><td>Reserved and set to 0</td></tr>
<tr><td>4</td><td>Reserved and set to 0</td></tr>
<tr><td>3</td><td>0 = WEP off
1 = WEP on</td></tr>
<tr><td>2</td><td>Reserved and set to 0</td></tr>
<tr><td>1</td><td>0 = Tx off
1 = Tx on</td></tr>
<tr><td>0</td><td>0 = Rx off
1 = Rx on</td></tr>
</table></td></tr>
<tr><td>Reserved</td><td>UINT16</td><td>Not used</td></tr>
</table>

RESPONSE

Field Name	Type	Description
CmdCode	UINT16	**CMD_MAC_CONTROL** | 0x8000
Size	UINT16	Number of bytes in command body
SeqNum	UINT16	Command sequence number, as set by the host
Result	UINT16	Result code
Action	UINT16	Action:

Bit	Description
15	WEP type: 0 = WEP40 1 = WEP104
14:12	Reserved
11	0 = disable WMM 1 = enable WMM
10	EnforceProtection This bit use to enforce strict data protection of Rx and Tx. 0 = disable protection 1 = enable protection
9	Reserved
8	0 = all multicast off 1 = all multicast on
7	0 = promiscous off 1 = promiscous on
6	Reserved and set to 0
5	Reserved and set to 0
4	Reserved and set to 0
3	0 = WEP off 1 = WEP on
2	Reserved and set to 0
1	0 = Tx off 1 = Tx on
0	0 = Rx off 1 = Rx on

5.2.2 CMD_802_11_RADIO_CONTROL

The **CMD_802_11_RADIO_CONTROL** command is used to control the PHY.

REQUEST

<table>
<tr><th>Field Name</th><th>Type</th><th>Description</th></tr>
<tr><td>CmdCode</td><td>UINT16</td><td><b>CMD_802_11_RADIO_CONTROL</b></td></tr>
<tr><td>Size</td><td>UINT16</td><td>Number of bytes in command body</td></tr>
<tr><td>SeqNum</td><td>UINT16</td><td>Command sequence number</td></tr>
<tr><td>Result</td><td>UINT16</td><td>Not used (set to 0)</td></tr>
<tr><td>Action</td><td>UINT16</td><td>Action:
0 = ACT_GET
1 = ACT_SET</td></tr>
<tr><td>Control</td><td>UINT16</td><td>Control:
<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>15:3</td><td>Reserved and set to 0</td></tr>
<tr><td>2:1</td><td>00 = long preamble
01 = short preamble
1x = auto preamble</td></tr>
<tr><td>0</td><td>0 = radio off
1 = radio on</td></tr>
</table></td></tr>
</table>

RESPONSE

<table>
<tr><th>Field Name</th><th>Type</th><th>Description</th></tr>
<tr><td>CmdCode</td><td>UINT16</td><td><b>CMD_802_11_RADIO_CONTROL</b> | 0x8000.</td></tr>
<tr><td>Size</td><td>UINT16</td><td>Number of bytes in command body</td></tr>
<tr><td>SeqNum</td><td>UINT16</td><td>Command sequence number, sent by the host</td></tr>
<tr><td>Result</td><td>UINT16</td><td>Result code</td></tr>
<tr><td>Action</td><td>UINT16</td><td>Action:
0 = ACT_GET
1 = ACT_SET</td></tr>
<tr><td>Control</td><td>UINT16</td><td>Control:
<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>15:3</td><td>Reserved and set to 0</td></tr>
<tr><td>2:1</td><td>00 = Long preamble
01 = Short preamble
1x = Auto preamble</td></tr>
<tr><td>0</td><td>0 = Radio off
1 = Radio on</td></tr>
</table></td></tr>
</table>

5.2.3 CMD_802_11_RF_ANTENNA

The **CMD_802_11_RF_ANTENNA** command sets or gets the Tx and Rx antenna mode. Antenna diversity is currently only implemented in 88W8385 based designs.

REQUEST

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11_RF_ANTENNA**
Size	UINT16	Number of bytes in command body
SeqNum	UINT16	Command sequence number
Result	UINT16	Not used (set to 0)
Action	UINT16	Action: 0x0001 = ACT_SET_RX: Set Rx antenna mode 0x0002 = ACT_SET_TX: Set Tx antenna mode 0x0004 = ACT_GET_RX: Get Rx antenna mode 0x0008 = ACT_GET_TX: Get Tx antenna mode
AntennaMode	UINT16	Antenna number: 1, 2, or 0xFFFF (diversity)

RESPONSE

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11_RF_ANTENNA** | 0x8000
Size	UINT16	Number of bytes in command body
SeqNum	UINT16	Command sequence number, sent by the host
Result	UINT16	Result code
Action	UINT16	Action: 0x0001 = ACT_SET_RX: Set Rx antenna mode 0x0002 = ACT_SET_TX: Set Tx antenna mode 0x0004 = ACT_GET_RX: Get Rx antenna mode 0x0008 = ACT_GET_TX: Get Tx antenna mode
AntennaMode	UINT16	Antenna number: 1, 2, or 0xFFFF (diversity)

5.2.4 CMD_802_11_RF_TX_POWER

The **CMD_802_11_RF_TX_POWER** command sets or gets the transmit power level.

REQUEST

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11_RF_TX_POWER**
Size	UINT16	Number of bytes in command body
SeqNum	UINT16	Command sequence number
Result	UINT16	Not used (set to 0)
Action	UINT16	Action: 0 = ACT_GET 1 = ACT_SET
CurrentLevel	SINT16	Current power level

RESPONSE

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11_RF_TX_POWER** | 0x8000
Size	UINT16	Number of bytes in command body
SeqNum	UINT16	Command sequence number, sent by the host
Result	UINT16	Result code (set to 0 for success)
Action	UINT16	Action: 0 = ACT_GET 1 = ACT_SET
CurrentLevel	SINT16	Current power levels
MaxPower	SINT8	Maximum valid power level
MinPower	SINT8	Minimum valid power level

The valid range of values for CurrentLevel are [MinPower,MaxPower]. The values of MinPower and MaxPower depend on the platform and are stored in the EEPROM during calibration for all manufacturing revisions of 0.5.12 and later. For older manufacturing revisions, MinPower is fixed to 0 dBm and MaxPower is fixed to 20 dBm when using the 88W801*x* (88W8010, 88W8015) RF chipsets and 21 dBm when using 88W803*x* (88W8030, 88W8031) RF chipsets.

The default value of CurrentLevel is the minimum (MaxPower, PA_P2) dBm. For minimum power level, the algorithm is not allowed to set the power to a value lower than the minimum value that the board is capable of supporting.

The CurrentLevel in the response always returns the current power level that is being used. It may vary with time if TPC or PA are enabled.

MaxLevel and MinLevel correspond to the maximum and minimum valid values for the power level.

For the power level set using this command to be sustained, the driver must ensure that both TPC and PA have been disabled.

5.2.5 CMD_802_11_RF_CHANNEL

The **CMD_802_11_RF_CHANNEL** command sets or gets the current channel value.

REQUEST

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11_RF_CHANNEL**
Size	UINT16	Number of bytes in command body
SeqNum	UINT16	Command sequence number
Result	UINT16	Not used (set to 0)
Action	UINT16	Action: 0 = ACT_GET 1 = ACT_SET
CurrentChannel	UINT16	Current channel number Only required if action is ACT_SET.
RFType	UINT16	Not used
Reserved	UINT16	Not used
ChannelList	UINT8[32]	Not used

RESPONSE

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11_RF_CHANNEL** | 0x8000
Size	UINT16	Number of bytes in command body
SeqNum	UINT16	Command sequence number, sent by the host
Result	UINT16	Result code
Action	UINT16	Action: 0 = ACT_GET 1 = ACT_SET
CurrentChannel	UINT16	Current channel number
RFType	UINT16	Not used
Reserved	UINT16	Not used
ChannelList	UINT8[32]	Not used

5.3 Register and Memory Access

Table 13 lists the supported commands for register and memory access.

Table 13: Register and Memory Access Commands

Command	Description	Page
5.3.1 "CMD_BBP_REG_ACCESS"	Peeks and pokes baseband processor hardware register	page 42
5.3.2 "CMD_RF_REG_ACCESS"	Peeks and pokes RF hardware register	page 43
5.3.3 "CMD_MAC_REG_ACCESS"	Peeks and pokes MAC hardware register	page 44
5.3.4 "CMD_EEPROM_ACCESS"	Retrieves the EEPROM data	page 45

5.3.1 CMD_BBP_REG_ACCESS

The **CMD_BBP_REG_ACCESS** command peeks or pokes a baseband processor (BBP) register.

REQUEST

Field Name	Type	Description
CmdCode	UINT16	**CMD_BBP_REG_ACCESS**
Size	UINT16	Number of bytes in command body
SeqNum	UINT16	Command sequence number
Result	UINT16	Not used (set to 0)
Action	UINT16	Action: 0 = ACT_GET 1 = ACT_SET
Offset	UINT16	Byte offset
Value	UINT8	Register value Only required when action is ACT_SET.
Reserved	UINT8[3]	Not used

RESPONSE

Field Name	Type	Description
CmdCode	UINT16	**CMD_BBP_REG_ACCESS** | 0x8000
Size	UINT16	Number of bytes in command body
SeqNum	UINT16	Command sequence number, sent by the host
Result	UINT16	Result code
Action	UINT16	Type of action set in the request message
Offset	UINT16	Byte offset
Value	UINT8	Register value Only required when action is ACT_GET.
Reserved	UINT8[3]	Not used

5.3.2 CMD_RF_REG_ACCESS

The **CMD_RF_REG_ACCESS** command peeks or pokes a PHY register.

REQUEST

Field Name	Type	Description
CmdCode	UINT16	**CMD_RF_REG_ACCESS**
Size	UINT16	Number of bytes in command body
SeqNum	UINT16	Command sequence number
Result	UINT16	Not used (set to 0)
Action	UINT16	Action: 0 = ACT_GET 1 = ACT_SET
Offset	UINT16	Byte offset
Value	UINT8	Register value Only required when action is ACT_SET.
Reserved	UINT8[3]	Not used

RESPONSE

Field Name	Type	Description
CmdCode	UINT16	**CMD_RF_REG_ACCESS** | 0x8000
Size	UINT16	Number of bytes in command body
SeqNum	UINT16	Command sequence number, sent by the host
Result	UINT16	Result code
Action	UINT16	Type of action set in the request message
Offset	UINT16	Byte offset
Value	UINT8	Register value Only required when action is ACT_GET.
Reserved	UINT8[3]	Not used

5.3.3 CMD_MAC_REG_ACCESS

The **CMD_MAC_REG_ACCESS** command peeks and pokes a MAC register.

REQUEST

Field Name	Type	Description
CmdCode	UINT16	**CMD_MAC_REG_ACCESS**
Size	UINT16	Number of bytes in command body
SeqNum	UINT16	Command sequence number
Result	UINT16	Not used (set to 0)
Action	UINT16	Action: 0 = ACT_GET 1 = ACT_SET
Offset	UINT16	Byte offset, aligned on 32-bit boundary
Value	UINT32	Register value Not used if action is ACT_GET.

RESPONSE

Field Name	Type	Description
CmdCode	UINT16	**CMD_MAC_REG_ACCESS** | 0x8000
Size	UINT16	Number of bytes in command body
SeqNum	UINT16	Command sequence number, sent by the host
Result	UINT16	Result code
Action	UINT16	Action: 0 = ACT_GET 1 = ACT_SET
Offset	UINT16	Byte offset, aligned on 32-bit boundary Present only if action is ACT_GET.
Value	UINT32	Register value Present only if action is ACT_GET.

5.3.4 CMD_EEPROM_ACCESS

The **CMD_EEPROM_ACCESS** command retrieves the EEPROM data.

REQUEST

Field Name	Type	Description
CmdCode	UINT16	**CMD_EEPROM_ACCESS** = 0x0059
Size	UINT16	Number of bytes in the command body
SeqNum	UINT16	Command sequence number
Result	UINT16	Not used (set to 0)
Action	UINT16	ACT_GET
Offset	UINT16	Multiples of 4 For example, 0, 4, 8, 12, 16, …
ByteCount	UINT16	Multiples of 4 For example, 4, 8, 12, 16, and 20. Maximum of 20 bytes.
Value	UINT8	User must provide a buffer of at least ByteCount length

RESPONSE

Field Name	Type	Description
CmdCode	UINT16	**CMD_EEPROM_ACCESS** | 0x8000
Size	UINT16	Number of bytes in the command body
SeqNum	UINT16	Command sequence number
Result	UINT16	Not used (set to 0)
Action	UINT16	ACT_GET
Offset	UINT16	Multiples of 4 For example, 0, 4, 8, 12, 16, …
ByteCount	UINT16	Multiples of 4 For example, 4, 8, 12, 16, and 20. Maximum of 20 bytes.
Value	UINT8	If result is Ok, it contains EEPROM value ordered from low to high offset

5.4 RF Calibration Data

Table 14 lists the RF calibration data commands.

Table 14: RF Calibration Commands

Command	Description	Page
5.4.1 "CMD_802_11_CAL_DATA_EXT"	Sets or gets the RF calibration data.	page 46

5.4.1 CMD_802_11_CAL_DATA_EXT

The **CMD_802_11_CAL_DATA_EXT** command sets or gets the RF calibration data.

A typical application of this command is for use in initializing custom stations without an EEPROM to store RF calibration data.

REQUEST

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11_CAL_DATA_EXT**
Size	UINT16	Number of bytes in command body
SeqNum	UNIT16	Command sequence number
Result	UINT16	Not used (set to 0)
Action	UINT16	Action: 0 = ACT_GET 1 = ACT_SET
Revision	UINT16	Revision to identify the contents of the following data
CalDataLen	UINT16	Length (bytes) of the following calibration data
CalData	UINT8[CalDataLen]	Calibration data contents

RESPONSE

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11_CAL_DATA_EXT** | 0x8000
Size	UINT16	Number of bytes in command body
SeqNum	UNIT16	Command sequence number
Result	UINT16	Result code
Action	UINT16	Action: 0 = ACT_GET 1 = ACT_SET
Revision	UINT16	Revision to identify the contents of the following data
CalDataLen	UINT16	Length of the following calibration data
CalData	UINT8[CalDataLen]	Calibration data contents

For the contents CalData, refer to the RD-88W MFG Manufacturing Test Suite documentation and/or contact a Marvell FAE.

5.5 Status Information

Table 15 lists the supported commands for status information.

Table 15: Status Information Commands

Command	Description	Page
5.5.1 "CMD_GET_HW_SPEC"	Gets hardware specifications	page 47
5.5.2 "CMD_802_11_GET_LOG"	Gets the WLAN log	page 49
5.5.3 "CMD_802_11_RSSI"	Gets the received radio signal strength	page 50

5.5.1 CMD_GET_HW_SPEC

The **CMD_GET_HW_SPEC** command queries hardware and other configuration data from the WLAN SoC, including the MAC address, multicast address list size, and hardware version number.

REQUEST

Field Name	Type	Description
CmdCode	UINT16	**CMD_GET_HW_SPEC**
Size	UINT16	Number of bytes in command body
SeqNum	UINT16	Command Sequence number
Result	UINT16	Not used
HwIfVersion	UINT16	Set to 0
HwVersion	UINT16	Set to 0
NumOfWCB	UINT16	Set to 0
NumOfMCastAdr	UINT16	Set to 0
PermanentAddr	UINT8[6]	Set to 0
RegionCode	UINT16	Set to 0
NumberOfAntenna	UINT16	Set to 0
FWReleaseNumber	UINT32	Set to 0
WcbBase	UINT32	Set to 0
RxPdRdPtr	UINT32	Set to 0
RxPdWrPtr	UINT32	Set to 0
FwCapInfo	UINT32	Set to 0

RESPONSE

Field Name	Type	Description
CmdCode	UINT16	**CMD_GET_HW_SPEC** | 0x8000
Size	UINT16	Number of bytes in command body
SeqNum	UINT16	Command sequence number, sent by the host
Result	UINT16	Result code

<table>
<tr><th>Field Name</th><th>Type</th><th>Description</th></tr>
<tr><td>HwIfVersion</td><td>UINT16</td><td>Hardware interface version number</td></tr>
<tr><td>Version</td><td>UINT16</td><td>Hardware version number</td></tr>
<tr><td>NumOfWCB</td><td>UINT16</td><td>Reserved</td></tr>
<tr><td>NumOfMCastAdr</td><td>UINT16</td><td>Maximum number of multicast addresses the firmware can handle.</td></tr>
<tr><td>PermanentAddr</td><td>UINT8[6]</td><td>MAC address
If set to 0xFFFFFFFFFFFF, the firmware MAC address is used as default, otherwise, the value in this field is used for the MAC address.</td></tr>
<tr><td>RegionCode</td><td>UINT16</td><td>Region code</td></tr>
<tr><td>NumberOfAntenna</td><td>UINT16</td><td>Number of antenna used</td></tr>
<tr><td>FWReleaseNumber</td><td>UINT32</td><td>Firmware release number
Example: 2.3.4.p1 = 0x01020304.</td></tr>
<tr><td>WcbBase</td><td>UINT32</td><td>Reserved</td></tr>
<tr><td>RxPdRdPtr</td><td>UINT32</td><td>Reserved</td></tr>
<tr><td>RxPdWrPtr</td><td>UINT32</td><td>Reserved</td></tr>
<tr><td>FwCapInfo</td><td>UINT32</td><td>Firmware Capability Information:
<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>10</td><td>Supports 802.11a
0 = do not support
1 = support</td></tr>
<tr><td>9</td><td>Supports 802.11g
0 = do not support
1 = support</td></tr>
<tr><td>8</td><td>Supports 802.11b
0 = do not support
1 = support</td></tr>
<tr><td>7:6</td><td>Rx Antenna Capability
00 = antenna 0 only
01 = antenna 1 only
1x = diversity</td></tr>
<tr><td>5:4</td><td>Tx Antenna Capability
00 = antenna 0 only
01 = antenna 1 only
1x = diversity</td></tr>
<tr><td>3</td><td>EEPROM Not Exit
0 = exit
1 = does not exit</td></tr>
<tr><td>2</td><td>Reserved</td></tr>
<tr><td>1</td><td>PS</td></tr>
<tr><td>0</td><td>WPA</td></tr>
</table>
</td></tr>
</table>

5.5.2 CMD_802_11_GET_LOG

The **CMD_802_11_GET_LOG** command gets the log.

REQUEST

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11_GET_LOG**
Size	UINT16	Number of bytes in command body
SeqNum	UINT16	Command sequence number
Result	UINT16	Not used (set to 0)

RESPONSE

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11_GET_LOG** | 0x8000
Size	UINT16	Number of bytes in response
SeqNum	UINT16	Same sequence number as in Get Log Request
Result	UINT16	Result code
dot11MulticastTransmitted FrameCount	UINT32	Increments when the multicast bit is set in the destination MAC address of a successfully transmitted MSDU
dot11FailedCount	UINT32	Increments when an MSDU is not transmitted successfully
dot11RetryCount	UINT32	Increments when an MSDU is successfully transmitted after one or more retransmissions
dot11MultipleRetryCount	UINT32	Increments when an MSDU is successfully transmitted after more than one retransmission
dot11FrameDuplicate Count	UINT32	Increments when a frame is received that the Sequence Control field is indicating a duplicate count
dot11RTSSuccessCount	UINT32	Increments when a CTS is received in response to an RTS
dot11RTSFailureCount	UINT32	Increments when a CTS is not received in response to an RTS
dot11ACKFailureCount	UINT32	Increments when an ACK is not received when expected
dot11ReceivedFragment Count	UINT32	Increments for each successfully received MPDU of type Data or Management
dot11MulticastReceived FrameCount	UINT32	Increments when a MSDU is received with the multicast bit set in the destination MAC address
dot11FCSErrorCount	UINT32	Increments when an FCS error is detected in a received MPDU
dot11TransmittedFrame Count	UINT32	Increments for each successfully transmitted MSDU
dot11WEP UndecryptableCount	UINT32	Increments when a frame is received with the WEP subfield of the Frame Control field set to one The WEPOn value for the key mapped to the TA's MAC address indicates that the frame is not encrypted or frame is discarded because the receiving station is not implementing the privacy option.

5.5.3 CMD_802_11_RSSI

The **CMD_802_11_RSSI** command gets the Rx signal strength.

REQUEST

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11_RSSI**
Size	UINT16	Number of bytes in command body
SeqNum	UINT16	Command sequence number
Result	UINT16	Not used (set to 0)
N	UINT16	Parameter used for exponential averaging

RESPONSE

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11_RSSI** | 0x8000
Size	UINT16	Number of bytes in command body
SeqNum	UINT16	Command sequence number, sent by the host
Result	UINT16	Result code (set to 0 for success)
SNR	UINT16	SNR (dB) in most recent beacon
NoiseFloor	UINT16	Absolute value of noise floor (dBm) in most recent beacon
AvgSNR	UINT16	Average SNR (dB) in the received beacons
AvgNoiseFloor	UINT16	Average absolute value of noise floor (dBm) in the received beacons

The field N in the request refers to the parameter used in the following formula for exponentially averaging the SNR in received beacons:

$$SNR_{avg} = ((N-1)*SNR_{avg} + SNR_{new})/N$$

Where SNR_{avg} is the average SNR, and SNR_{new} is the most recent value of SNR.

If N is set to 0, the current value in firmware is not changed. If N is set to 1, then this formula degenerates to always returning the SNR value in the most recent beacon. The higher the value of N, the more weight is given to the old average and hence the slower the convergence to any change in SNR. The default value of N is 8.

The SNR field in the response provides the SNR in the most recently received beacon.

The NoiseFloor field in the response provides the absolute value of the noise floor in the most recently received beacon.

The AvgSNR field in the response provides the exponentially averaged SNR in the received beacons. If N is set to 1, this provides the SNR in the most recently received beacon.

The AvgNoiseFloor field in the response provides the exponentially averaged absolute value of the noise floor in the received beacons. If N is set to 1, this provides the absolute value of the noise floor in the most recently received beacon.

5.6 LED Control

The **CMD_802_11_LED_CONTROL** API is used to control the operation of GPIO pins and LEDs in the firmware.

5.6.1 CMD_802_11_LED_CONTROL

The **CMD_802_11_LED_CONTROL** API is used to control the LED functionality.

REQUEST

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11_LED_CONTROL**
Size	UINT16	Number of bytes in command body
SeqNum	UINT16	Command sequence number
Result	UINT16	Not used (set to 0)
Action	UINT16	Action: 0 = ACT_GET 1 = ACT_SET
NumLed	UINT16	Reserved (set to 0)
NOTE: Above fixed fields are followed by a variable number of TLV fields.		

RESPONSE

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11_LED_CONTROL** | 0x8000
Size	UINT16	Number of bytes in command body
SeqNum	UINT16	Command sequence number, sent by the host
Result	UINT16	Result code (set to 0 for success)
Action	UINT16	Same as request
NumLed	UINT16	Number of LEDs implemented by firmware
NOTE: Above fixed fields are followed by a variable number of TLV fields.		

If action is ACT_SET, then the information in TLV fields in the request is applied to the firmware. If action is ACT_GET, then the TLV fields in the current firmware settings are returned in the response.

NumLed field is set in the response by the firmware to indicate the number of LEDs that are supported. This is set to 0 in the request and ignored by the firmware in the response. All the settings for LEDs beyond the LED indexed by this number are ignored by the firmware and is set to 0 by the driver.

The following TLVs can be included with this command depending on the chip/firmware version:

- **MrvlIETypes_LedGpio_t**
- **MrvlIETypes_LedBehavior_t**

Note

Please refer to the LED Configuration Application Note.

5.7 Scan

Table 16 lists the supported Scan commands.

Table 16: Scan Commands

Command	Description	Page
5.7.1 "CMD_802_11_SCAN"	Starts the scan process	page 52
5.7.2 "CMD_802_11_BG_SCAN_CONFIG"	Sets/gets background scan configuration	page 54
5.7.3 "CMD_802_11_BG_SCAN_QUERY"	Gets background scan results	page 57

5.7.1 CMD_802_11_SCAN

The **CMD_802_11_SCAN** command starts the scan process to discover existing wireless networks within proximity to the station.

REQUEST

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11_SCAN**
Size	UINT16	Number of bytes in command body
SeqNum	UINT16	Command sequence number
Result	UINT16	Not used (set to 0)
BssType	UINT8	BSS type: • BSS_INDEPENDENT • BSS_INFRASTRUCTURE • BSS_ANY
BssId	UINT8[6]	MAC address Filter on BSSID. Zeros out to disable filter.
SsIdParamSet	MrvlIEtypes_SsIdParamSet_t	SSID set parameter (optional)
ChanListParamSet	**MrvlIETypes_ChanListParamSet_t**	Channel to scan list set parameter
OpRateSet	MrvlIETypes_RatesParamSet_t	Supported data rates set parameter (optional)
NumProbes	**MrvlIETypes_NumProbes_t**	Number of probes to send (optional)

RESPONSE

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11_SCAN** | 0x8000
Size	UINT16	Number of bytes in command body
SeqNum	UINT16	Command sequence number, sent by the host
Result	UINT16	Result code
BufSize	UINT16	Scan response buffer size
NumOfSet	UINT8	Number of APs in the buffer

Field Name	Type	Description
BssDescSet	UINT8[]	Variable size buffer that keeps the information related the scanned APs
TSFTable	**MrvlIEtypes_TsfTimestamp_t**	UINT64 TSF timestamps table There are the same number of TSF values as there are NumOfSet BssDescSets in the command response.

The Size field indicates the entire command response size, including the TSFTable.

BufSize only stores the scan response buffer size (BssDescSet field), and does not include the size of the TSFTable.

The TSF Timestamp table has the firmware TSF value for each scan response that is received. The TSFTable is a TLV with a value of 0x0113.

BssDescSet Format

Field Name	Type	Description
IELength	UINT16	Total IE length
BSSID	UINT8[6]	BSSID
Rssi	UINT8	RSSI value as received from peer
Probe Response/Beacon Payload	UINT8[]	Fixed and variable length IE received on Probe Response or Beacon frames

Probe Response/Beacon Payload Format

Field Name	Type	Description
PktTimeStamp	UINT8[8]	Timestamp
BcnInterval	UINT16	Beacon interval
CapInfo	UINT16	Capabilities information
IEParameters	UINT8[]	Information element parameters

Note

Additional TLV's may optionally be returned at the end of the scan result buffer.

5.7.2 CMD_802_11_BG_SCAN_CONFIG

The **CMD_802_11_BG_SCAN_CONFIG** command gets/sets the current background scan configuration.

REQUEST

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11_BG_SCAN_CONFIG**
Size	UINT16	Number of bytes in the command body
SeqNum	UNIT16	Command sequence number
Result	UINT16	Not used (set to 0)
Action	UINT16	Action: 0 = ACT_GET 1 = ACT_SET
Enable	UINT8	0 = disable 1 = enable
BssType	UINT8	BSS type: 0x01 = BSS_INDEPENDENT 0x02 = BSS_INFRASTRUCTURE 0x03 = BSS_ANY
ChannelsPerScan	UINT8	Number of channels scanned during each scan
DiscardWhenFull	UINT8	Scan result to discard when scan list is full
Reserved	UINT16	Reserved
ScanInterval	UINT32	Interval between consecutive scans
StoreCondition	UINT32	Condition to store the result
ReportConditions	UINT32	Conditions to trigger report to host
MaxScanResults	UINT16	Maximum number of scan results to store

RESPONSE

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11_BG_SCAN_CONFIG** | 0x8000
Size	UINT16	Number of bytes in command body
SeqNum	UNIT16	Command sequence number, as sent by host
Result	UINT16	Result code (set to 0 for success)
Action	UINT16	Action: 0 = ACT_GET 1 = ACT_SET
Enable	UINT8	0 = disable 1 = enable
BssType	UINT8	BSS type: 0x01 = BSS_INDEPENDENT 0x02 = BSS_INFRASTRUCTURE 0x03 = BSS_ANY

Field Name	**Type**	**Description**
ChannelsPerScan	UINT8	Number of channels scanned during each scan
DiscardWhenFull	UINT8	Scan result to discard when scan list is full
Reserved	UINT16	Reserved
ScanInterval	UINT32	Interval between consecutive scans
StoreCondition	UINT32	Condition to store the result
ReportConditions	UINT32	Conditions to trigger report to host
MaxScanResults	UINT16	Maximum number of scan results to store

Action ACT_GET is used to query the current background scan configuration. ACT_SET is used to change the current background scan configuration.

Enable is used to enable or disable the background scanning. If the background scan is disabled, it aborts the current scan in progress.

BssType determines whether only infrastructure BSS, IBSS, or all BSS descriptions are stored as a part of the background scan results.

DiscardWhenFull is reserved.

ChannelsPerScan indicates the number of channels (from the ChannelList) to be scanned at each scan instance. This determines the duration the station stays away from its operational channel, so this must be set judiciously. Specifically, the total time spent in one scan instance should be less than the listen interval of the station.

ScanInterval is the time (in ms) between two consecutive background scan instances.

StoreCondition is used to decide whether to store a BSS description as a part of the background scan results. The available conditions are:

- Bit 0 = SSID match
- Bit 1 = SSID match AND SNR above SNR threshold

ReportConditions determine when the firmware generates a background scan report event to the host. It is a bitmap with one bit corresponding to each condition below:

- Bit 0 = SSID match
- Bit 1 = SSID match AND SNR above SNR threshold

Any condition set in the bitmap triggers a report to the host.

MaxScanResults is reserved.

To allow this command to be extensible, there may be zero or more TLV fields included in the request command at the end. In order to ensure backward compatibility, any TLV field not recognized must be ignored and skipped.

The following TLV fields are currently defined:

- **MrvlIETypes_ChanListParamSet_t** (required)
- MrvlIETypes_SsIdParamSet_t (optional—Absent TLV matches any SSID)
- **MrvlIETypes_NumProbes_t** (optional)
- **MrvlIETypes_BcastProbe_t** (optional)
- **MrvlIETypes_NumSSIDProbe_t** (optional)

MrvlIETypes_NumProbes_t, **MrvlIETypes_BcastProbe_t**, and **MrvlIETypes_NumSSIDProbe_t** TLVs control the number and type of probe requests that are generated on each channel configured for active scan.

Table 17 enumerates various possible combinations of the probe related parameters and the expected result in each case.

Table 17: Probe Combinations

ScanType	NumProbes	BcastProbe	NumSSIDProbe	Expected Results
Passive	X	X	X	**No** probe requests generated
Active	X	0	0	**No** probe requests generated
Active	1	1	0	Only **one** probe request with broadcast SSID generated
Active	1	0	1	Only **one** probe request with the first SSID in the SSID list generated (default configuration)
Active	1	1	1	**Two** probe requests generated, one with broadcast SSID and one with the first SSID in the SSID list
Active	1	0	2	**Two** probe requests generated, one with the first SSID in the SSID list, and one with the second SSID in the SSID list
Active	1	1	2	**Three** probe requests generated, one with broadcast SSID, one with the first SSID in the SSID list, and one with the second SSID in the SSID list
Active	2	0	1	**Two** probe requests with the first SSID in the SSID list generated
Active	2	1	0	**Two** probe requests with broadcast SSID generated
Active	2	1	1	**Four** probe requests generated, two with broadcast SSID and two with the first SSID in the SSID list

The current firmware implementation has a limitation of not being able to transmit more than 2 probe requests in any given channel. Any configuration which results in more than 2 probe request frames should not be used. The above examples are for illustrating the usage of these TLVs only.

5.7.3 CMD_802_11_BG_SCAN_QUERY

The **CMD_802_11_BG_SCAN_QUERY** command gets the current background scan results. It can be issued at any time by the driver (even if the firmware is currently scanning some channel as part of the ongoing background scanning). The firmware ensures that it generated a coherent response by suitably synchronizing access to the scan results buffer.

REQUEST

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11_BG_SCAN_QUERY**
Size	UINT16	Number of bytes in command body
SeqNum	UNIT16	Command sequence number
Result	UINT16	Not used (set to 0)
Flush	UINT8	Flush the results or not

RESPONSE

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11_BG_SCAN_QUERY** | 0x8000
Size	UINT16	Number of bytes in command body
SeqNum	UNIT16	Command sequence number, as sent by host
Result	UINT16	Result code (set to 0 for success)
ReportCondition	UINT32	Report condition(s) matched since last query
BssDescSize	UINT16	Total size of the scan result to follow
NumSets	UINT8	Number of descriptions to follow
BssDesc	UINT8[]	Current scan results

Flush determines whether the firmware should flush the current background scan results after responding to this command or not.

The response contains all the background scan results currently stored in the firmware and also the conditions that were met since the last report.

5.7.4 Background Scan Report Event

This event is generated from the firmware to the host whenever any of the report conditions are matched. The driver should issue the background scan query command when it receives this event.

Note

Background scan is only supported in Deep Sleep mode.

5.7.5 Background Scan Operation

5.7.5.1 Not Connected and Deep Sleep

If the station is put in Deep Sleep mode with the background scan enabled, it wakes up periodically to perform the background scan and then goes back to sleep. Each time the station goes to sleep, it sets a wakeup timeout equal to ScanInterval. It could be woken up from sleep by either the timer or the host driver. If it is woken up by the timer, it performs a scan according to the configured background scan parameters, and goes back to sleep. If it is woken by the host driver, then it stays awake and continues to perform the background scan until the host driver puts it back to sleep.

If, when the station is woken up by the timer, it scans and one of the reporting conditions is matched, then the firmware wakes up the host using the host wakeup procedure. The host driver must be prepared to handle this scenario when it enables background scanning in Deep Sleep mode.

Note that in case a condition matches during background scan, the firmware generates a Background Scan Report event. In this case, the firmware does not generate the Deep Sleep Awake event that it usually generates when the host brings it out of Deep Sleep mode. See Section 6. "MAC Events" on page 113, for Background Scan event and Deep Sleep Awake event values.

Note

Refer to the Background Scan Application Note.

5.8 Network Start/Stop/Join

Table 18 lists the supported network start, stop, and join commands.

Table 18: Network Start/Stop/Join Commands

Command	Description	Page
5.8.1 "CMD_802_11_ASSOCIATE"	Initiate an association with the AP	page 59
5.8.2 "CMD_802_11_AD_HOC_START"	Starts an Ad-Hoc network	page 61
5.8.3 "CMD_802_11_AD_HOC_JOIN"	Join an Ad-Hoc network	page 62
5.8.4 "CMD_802_11_AD_HOC_STOP"	Stops Ad-Hoc Network	page 64

5.8.1 CMD_802_11_ASSOCIATE

The **CMD_802_11_ASSOCIATE** command passes all needed information from driver to firmware to build the association request packet.

REQUEST

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11_ASSOCIATE**
Size	UINT16	Number of bytes in command body
SeqNum	UINT16	Command sequence number, sent by the host
Result	UINT16	Not used (set to 0)
PeerStaAddr	UINT8[6]	Peer MAC address
CapInfo	UINT16	Capability information
ListenInterval	UINT16	Listen interval
BcnPeriod	UINT16	Beacon period
DtimPeriod	UINT8	DTIM period
SsIdParamSet	MrvlIEtypes_SsIdParamSet_t	SSID set parameter
PhyParamDSSet	MrvlIETypes_PhyParamDSSet_t	Specifies DS parameters
CfParamSet	MrvlIETypes_CfParamSet_t	Specifies CF parameters
OpRateSet	MrvlIETypes_RatesParamSet_t	Supported data rates set parameter
RsnParamSet	MrvlIETypes_RsnParamSet_t	RSN type/length extended IEEE IE (See Section 8. "TLV Usage" on page 203 only when RSN is required.)
VendorParamSet	MrvlIETypes_VendorParamSet_t	Vendor specific type/length extended IEEE IE for WMM or WPA (See Section 8. "TLV Usage" on page 203 only when WMM or WPA is required.)
PowerCapability	**MrvlIETypes_LocalPowerCapabiity_t**	Set minimum/maximum power capability (optional)

Field Name	Type	Description
SupportedChannelsSet	MrvlIETypes_SupportedChannels_t	802.11h supported channel element Only when 802.11h is required.
PassThroughIEs	**MrvlIETypes_Passthrough_t**	Add IEEE IEs to the association request directly (optional)

RESPONSE

<table>
<tr><th>Field Name</th><th>Type</th><th>Description</th></tr>
<tr><td>CmdCode</td><td>UINT16</td><td><b>CMD_802_11_ASSOCIATE</b> | 0x8000</td></tr>
<tr><td>Size</td><td>UINT16</td><td>Number of bytes in command body</td></tr>
<tr><td>SeqNum</td><td>UINT16</td><td>Command sequence number, sent by the host</td></tr>
<tr><td>Result</td><td>UINT16</td><td>Result code</td></tr>
<tr><td>AssocRsp</td><td>IEEEtypes_AssocRsp_t</td><td>Entire IEEE (re)association response
Format is:
<table>
<tr><th>Field Name</th><th>Type</th><th>Description</th></tr>
<tr><td>Capability</td><td>UINT16</td><td>IEEEtypes_CapInfo_t</td></tr>
<tr><td>StatusCode</td><td>UINT16</td><td>IEEEtypes_StatusCode_t</td></tr>
<tr><td>AId</td><td>UINT16</td><td>IEEEtypes_AId_t</td></tr>
<tr><td>IEBuffer</td><td>u8[]</td><td>Rest of the (re)association response</td></tr>
</table>
</td></tr>
</table>

The Association response contains the exact same fields, IEs, and values as the response received from the AP. The firmware does not modify the data before passing it on to the driver. Refer to the IEEE specification for more detailed information regarding the potential response values.

5.8.2 CMD_802_11_AD_HOC_START

The **CMD_802_11_AD_HOC_START** command is used to start an Ad-Hoc network.

REQUEST

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11_AD_HOC_START**
Size	UINT16	Number of bytes in command body
SeqNum	UINT16	Command sequence number
Result	UINT16	Not used (set to 0)
SSID	UINT8[32]	SSID
BssType	UINT8	BSS type to use: BSS_INDEPENDENT
BcnPeriod	UINT16	Specify beacon period (ms)
ATIMWindow	UINT8	Specify ATIM window (TU) 1 TU = 1.024 ms
DsParamSet	IEEEtypes_DsParamSet_t	IEEE DS parameter set element
Reserved	UINT8[4]	Initialize as zero
IbssParamSet	IEEEtypes_IbssParamSet_t	IEEE IBSS parameter set
Reserved	UINT8[4]	Initialize as zero
ProbeDelay	UINT16	Specify probe delay (µs)
CapInfo	IEEEtypes_CapInfo_t	Capability information
Data Rate	UINT8[14]	Supported data rates Data rate are in multiples of 500 Kbps. Basic data rates have MSb set to 1, for example: 0x82 = 1 Mbps 0x84 = 2 Mbps 0x8B = 5.5 Mbps 0x96 = 11 Mbps
IbssDfs	MrvlIETypes_Ibss_Dfs_t	Station originated IBSS DFS element with current station as DFS owner (802.11h only)
QuietPeriod	MrvlIETypes_Quiet_t	Quiet period for the new IBSS (802.11h only).

RESPONSE

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11_AD_HOC_START** | 0x8000
Size	UINT16	Number of bytes in command body
SeqNum	UINT16	Command sequence number, sent by the host
Result	UINT16	Result code

5.8.3 CMD_802_11_AD_HOC_JOIN

The **CMD_802_11_AD_HOC_JOIN** command is used to join an Ad-Hoc network.

REQUEST

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11_AD_HOC_JOIN**
Size	UINT16	Number of bytes in command body
SeqNum	UINT16	Command sequence number
Result	UINT16	Not used (set to 0)
BssDesc	**IEEEtypes_BssDesc_t**	BSS descriptor
FailTimeout	UINT16	Join failure time out (TU)
ProbeDelay	UINT16	Probe delay (μs)
IbssDfs	MrvlIETypes_Ibss_Dfs_t	Station originated IBSS DFS element with the DFS owner and recovery time from the existing IBSS (802.11h only)
QuietPeriod	MrvlIETypes_Quiet_t	Quiet period for the new IBSS (802.11h only)

REPONSE

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11_AD_HOC_JOIN** | 0x8000
Size	UINT16	Number of bytes in command body
SeqNum	UINT16	Command sequence number, sent by the host
Result	UINT16	Result code

To connect to an Ad-Hoc network:

1. Send a SNMP MIB command with subcommand ACT_SET to set the firmware to Ad-Hoc mode, and set the OID field to 0000 for DesiredBSSType.
2. Send SCAN command with the SSID of the creator.
3. Appropriately set the RF channel.
4. Send a Join Ad-Hoc network command.

5.8.3.1 IEEEtypes_BssDesc_t

Table 19: BSS Description Set

Field Name	Data Type	Description
BssId	UINT8[6]	MAC address
SsId	UINT8[32]	SSID
BssType	IEEEtypes_Bss_e	See Table 20
BcnPeriod	UINT16	Beacon period
DtimPeriod	UINT8	Specify DTIM period (TBTTs)
Timestamp	UINT8[8]	Timestamp
StartTs	UINT8[8]	Starting timestamp
DsParamSet	IEEEtypes_DsParamSet_t	IEEE DS parameter set element
Reserved	UINT8[4]	Initialize as zero
IbssParamSet	IEEEtypes_IbssParamSet_t	IEEE IBSS parameter set
Reserved	UINT8[4]	Initialize as zero
CapInfo	UINT16	Firmware capability information
DataRates	UINT8[8] for 802.11b UINT8[14] for 802.11g	Data rates

Table 20: IEEEtypes_Bss_e Set

Field Name	Description
BSS_INFRASTRUCTURE	1
BSS_INDEPENDENT	2
BSS_ANY	3 (Recommended for use when joining Ad-Hoc networks)

5.8.4 CMD_802_11_AD_HOC_STOP

The **CMD_802_11_AD_HOC_STOP** command is used to stop an Ad-Hoc network. The firmware exits Ad-Hoc mode and changes to idle mode.

REQUEST

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11_AD_HOC_STOP**
Size	UINT16	Number of bytes in command body
SeqNum	UNIT16	Command sequence number
Result	UINT16	Not used (set to 0)

RESPONSE

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11_AD_HOC_STOP**
Size	UINT16	Number of bytes in command body
SeqNum	UNIT16	Command sequence number, sent by the host
Result	UINT16	Result code

5.9 Security

Table 21 lists the supported security commands.

Table 21: Security Commands

Command	Description	Page
5.9.1 "CMD_802_11_AUTHENTICATE"	Starts authentication process with the AP	page 65
5.9.2 "CMD_802_11_DEAUTHENTICATE"	Starts de-authentication process with the AP	page 66
5.9.3 "CMD_802_11_SET_WEP"	Configures the WEP keys	page 67
5.9.4 "CMD_802_11_ENABLE_RSN"	Sets/gets RSN enable state	page 68
5.9.5 "CMD_802_11_KEY_MATERIAL"	Sets/gets key material used to do Tx encryption or Rx decryption	page 69

5.9.1 CMD_802_11_AUTHENTICATE

Authentication happens internally in the firmware during an Associate command. The **CMD_802_11_AUTHENTICATE** command only sets the authentication suite in the firmware. It does not trigger the firmware to start the authentication process with the AP. This API should be called anytime a change in the authentication type is required for future association requests.

REQUEST

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11_AUTHENTICATE**
Size	UINT16	Number of bytes in command body
SeqNum	UINT16	Command sequence number
Result	UINT16	Not used (set to 0)
MacAddr	UINT8[6]	Peer MAC address
AuthType	UINT8	Auth type to use: 0x00 = AUTH_MODE_OPEN 0x01 = AUTH__MODE_SHARED 0x80 = AUTH__MODE_NETWORK_EAP

RESPONSE

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11_AUTHENTICATE** | 0x8000.
Size	UINT16	Number of bytes in command body
SeqNum	UINT16	Command sequence number, sent by the host
Result	UINT16	Result code

5.9.2 CMD_802_11_DEAUTHENTICATE

The **CMD_802_11_DEAUTHENTICATE** command starts the de-authentication process with the AP.

REQUEST

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11_DEAUTHENTICATE**
Size	UINT16	Number of bytes in command body
SeqNum	UINT16	Command sequence number
Result	UINT16	Not used (set to 0)
PeerStaAddr	UINT8[6]	Peer MAC address
ReasonCode	UINT16	Reason code defined in IEEE 802.11 specification section 7.3.1.7 to indicate de-authentication reason

RESPONSE

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11_DEAUTHENTICATE** | 0x8000
Size	UINT16	Number of bytes in command body
SeqNum	UINT16	Command sequence number, sent by the host
Result	UINT16	Result code
PeerStaAddr	UINT8[6]	Peer MAC address

5.9.3 CMD_802_11_SET_WEP

The **CMD_802_11_SET_WEP** command configures WEP keys. When a key is no longer needed, the driver should remove it from firmware. This prevents it from being used by firmware accidentally.

REQUEST

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11_SET_WEP**
Size	UINT16	Number of bytes in command body
SeqNum	UINT16	Command sequence number
Result	UINT16	Not used (set to 0)
Action	UINT16	Action: 0x02 = ACT_ADD 0x04 = ACT_REMOVE 0x08 = ACT_USE_DEFAULT
TxKeyIndx	UINT16	Key set being used for transmit Range is 0–3
K1WEPType	UINT8	WEP key: 0x1 = TYPE_WEP_40_BIT 0x2 = TYPE_WEP_104_BIT
K2WEPType	UINT8	WEP key: 0x1 = TYPE_WEP_40_BIT 0x2 = TYPE_WEP_104_BIT
K3WEPType	UINT8	WEP key: 0x1 = TYPE_WEP_40_BIT 0x2 = TYPE_WEP_104_BIT
K4WEPType	UINT8	WEP key: 0x1 = TYPE_WEP_40_BIT 0x2 = TYPE_WEP_104_BIT
WEP1	UINT8[16]	WEP key1 value
WEP2	UINT8[16]	WEP key2 value
WEP3	UINT8[16]	WEP key3 value
WEP4	UINT8[16]	WEP key4 value

RESPONSE

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11_SET_WEP** | 0x8000
Size	UINT16	Number of bytes in command body
SeqNum	UINT16	Command sequence number, sent by the host
Result	UINT16	Result code

5.9.4 CMD_802_11_ENABLE_RSN

The host driver uses the **CMD_802_11_ENABLE_RSN** command to set or get the RSN enable state.

REQUEST

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11_ENABLE_RSN**
Size	UINT16	Number of bytes in command body
SeqNum	UINT16	Command sequence number
Result	UINT16	Not used (set to 0)
Action	UINT16	Action: 0 = ACT_GET 1 = ACT_SET
Enable	UINT16	Enable or disable RSN: 0 = disable RSN 1 = enable RSN

RESPONSE

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11_ENABLE_RSN** | 0x8000
Size	UINT16	Number of bytes in command body
SeqNum	UINT16	Command sequence number, sent by the host
Result	UINT16	Result code
Action	UINT16	Action: 0 = ACT_GET 1 = ACT_SET
Enable	UINT16	Enable or disable RSN: 0 = disable RSN 1 = enable RSN

5.9.5 CMD_802_11_KEY_MATERIAL

CMD_802_11_KEY_MATERIAL is a generic command used to set/get the key material used to do Tx encryption or Rx decryption of unicast, multicast, and broadcast data packets for WPA2.

This command can be used to replace WPA Set/Get Pairwise Key and WPA Set/Get Group Key commands.

Since this command makes use of variable length IE, it is possible to include multiple **MrvlIETypes_KeyParamSet_t** in a single command. It is assumed the driver validates the API command before passing it down to firmware. If there are conflicts in two or more **MrvlIETypes_KeyParamSet_t**, the latter of parameter set in the chain takes precedence. For example, if the command contains a **MrvlIETypes_KeyParamSet_t** of type TKIP (unicast) and a **MrvlIETypes_KeyParamSet_t** of type AES (unicast), the firmware encrypts/decrypts data packets using the AES key.

REQUEST

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11_KEY_MATERIAL** (0x005E)
Size	UINT16	Number of bytes in command body
SeqNum	UNIT16	Command sequence number
Result	UINT16	Not used (set to 0)
Action	UINT16	Action: 0 = ACT_GET: returns all the available keys 1 = ACT_SET
KeyParamSet	**MrvlIETypes_KeyParamSet_t**[]	Key parameter set(s) Specify key parameters with ACT_SET.

RESPONSE

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11_KEY_MATERIAL** | 0x8000.
Size	UINT16	Number of bytes in command body
SeqNum	UINT16	Command sequence number as sent by the host
Result	UINT16	Result code
Action	UINT16	Action: 0 = ACT_GET 1 = ACT_SET
KeyParamSet	**MrvlIETypes_KeyParamSet_t**[]	Key parameter set Return valid only for ACT_GET. Return as many as needed.

5.10 Rate Adaptation

Table 22 lists the rate adaptation commands.

Table 22: Rate Adaptation Commands

Command	Description	Page
5.10.1 "CMD_802_11_RATE_ADAPT_RATESET"	Sets/gets transmit data rate	page 70
5.10.2 "CMD_TX_RATE_QUERY"	Reports the current Tx rate of the first packet associated with rate adaptation	page 72

5.10.1 CMD_802_11_RATE_ADAPT_RATESET

The **CMD_802_11_RATE_ADAPT_RATESET** command sets or gets the transmit data rate.

REQUEST

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11_RATE_ADAPT_RATESET**
Size	UINT16	Number of bytes in command body
SeqNum	UINT16	Command sequence number
Result	UINT16	Not used (set to 0)
Action	UINT16	Action: 0 = ACT_GET 1 = ACT_SET
EnableHwAuto	UINT16	Hardware auto adapt enabled or not
Bitmap	UINT16	Bitmap of rates allowed for rate adaptation

RESPONSE

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11_RATE_ADAPT_RATESET** | 0x8000
Size	UINT16	Number of bytes in command body
SeqNum	UINT16	Command sequence number, as sent by host
Result	UINT16	Result code (set to 0 for success)
Action	UINT16	Same as in request
EnableHwAuto	UINT16	Hardware auto adapt enabled or not
Bitmap	UINT16	Bitmap of rates allowed for rate adaptation

ACT_SET is used to change the set of allowed rates and/or to enable/disable the hardware auto rate drop. ACT_GET is used to query the current settings.

The EnableHwAuto field indicates whether the hardware auto rate drop is enabled or not. If the hardware auto rate drop is enabled, then the hardware automatically transmits packet retries at a lower rate based on a pre-defined look-up table. This field is set to 1 to enable hardware auto rate drop, and is set to 0 to disable it.

Bitmap in the request is the bitmap of allowed rates for rate adaptation. The rate adaptation algorithm (both in firmware and hardware) only switches between rates that are included in the bitmap by setting the corresponding bit in the bitmap.

Bitmap in the response is the current bitmap of allowed rates. This is obtained by taking an intersection of the rate set configured using this command and the current set of supported rates. If this intersection turns out to be an empty set, then the lowest supported rate is selected.

Table 23 shows the bitmap correlation to data rates. Bit 0 is the least significant bit (LSB).

Table 23: Data Rate

Bit	Data Rate
15:13	Reserved
12	54 Mbps
11	48 Mbps
10	36 Mbps
9	24 Mbps
8	18 Mbps
7	12 Mbps
6	9 Mbps
5	6 Mbps
4	Reserved
3	11 Mbps
2	5.5 Mbps
1	2 Mbps
0	1 Mbps

If only one bit is set in the bitmap, then the rate adaptation algorithm is completely disabled and a single fixed rate is always used.

5.10.2 CMD_TX_RATE_QUERY

The **CMD_TX_RATE_QUERY** command reports the current Tx rate of the first packet associated with rate adaptation. The host needs to exit Power Save mode before issuing this command. To reduce the impact on throughput, do not use this command too frequently.

REQUEST

Field Name	Type	Description
CmdCode	UINT16	**CMD_TX_RATE_QUERY** (0x007F)
Size	UINT16	Number of bytes in the command body
SeqNum	UNIT16	Command sequence number
Result	UINT16	Not used (set to 0)
TxRate	UINT16	Not used (set to 0)

RESPONSE

Field Name	Type	Description
CmdCode	UINT16	**CMD_TX_RATE_QUERY** | 0x8000 (0x807F)
Size	UINT16	Number of bytes in the command body
SeqNum	UNIT16	Command sequence number
Result	UINT16	Result code
TxRate	UINT16	Return Tx data rate value

Table 24: Tx Data Rate Value

Value	Data Rate
13 and up	Reserved
12	54 Mbps
11	48 Mbps
10	36 Mbps
9	24 Mbps
8	18 Mbps
7	12 Mbps
6	9 Mbps
5	6 Mbps
4	Reserved
3	11 Mbps
2	5.5 Mbps
1	2 Mbps
0	1 Mbps

Note

The **CMD_TX_RATE_QUERY** command is only supported in certain specific versions of firmware v5.0 for 88W8385 and 88W8399 devices.

5.11 Transmit Power Control

The Marvell enhanced transmit power control (TPC) algorithm minimizes power consumption while maintaining the maximum attainable throughput.

There are three distinct power levels:

- P0
- P1
- P2

The TPC algorithm dynamically adapts between these three power levels only.

The power is dynamically adjusted only at the highest transmit rate (11 Mbps for 802.11b and 54 Mbps for 802.11a/g networks). When the transmit rate is less than the highest rate, the highest transmit power (P2) is always used.

The key metrics used in the algorithm are the packet error rate (PER) and SNR indicator. The firmware maintains an exponentially averaged value of SNR based on the SNR of all received beacons.

The following thresholds are defined:

- PER_Th—PER threshold, the value is different for reducing and increasing power
- SNR_Th—SNR threshold, the value is different for each power transition

If at any point, the PER exceeds PER_Th and the average SNR is below SNR_Th, then the power is increased.

Also, after every second or after a certain large number of packets are transmitted (whichever happens earlier), if the PER is less than PER_Th, the average SNR is above SNR_Th, and the transmit rate is the maximum, then the power is reduced.

The actual thresholds for each power transition at the highest rate of 11 Mbps and the three TPC power levels of P0=0 dBm, P1=5 dBm and P2=10 dBm are as follows:

P0->P1 transition: PER_Th=20%, SNR_Th_P0->P1=21dB

P1->P2 transition: PER_Th=20%, SNR_Th_P1->P2=15dB

P2->P1 transition: PER_Th=15%, SNR_Th_P2->P1=16dB

P1->P0 transition: PER_Th=15%, SNR_Th_P1->P0=22dB

If the values of P0, P1, and P2 are changed, the corresponding SNR thresholds are changed accordingly. Thus if P1 is changed from 7 dBm to (7+Delta) dBm, then SNR_Th_P2->P1 changes to (SNR_Th_P2->P1 - Delta) and SNR_Th_P1->P2 changes to (SNR_Th_P1->P2 - Delta).

It is recommended that the user not change the three default TPC power levels. If the three TPC power levels are changed by the user then the algorithm is not optimal. It is also recommended that the user use the TPC option without the UseSNR flag turned on. The received SNR thresholds are optimized for the three default power levels only.

The transmit power is set according to Table 25.

Table 25: Transmit Power Settings

TPC Enabled	PA enabled	Transmit power setting
No	No	Always set to CurrentLevel
Yes	No	Set to P2 at all rates except the highest rate Set to P0, P1, or P2 (according to TPC algorithm) at the highest rate.
No	Yes	Set to PA_P0, PA_P1, or PA_P2 (according to PA algorithm)
Yes	Yes	Set to P0, P1, or P2 (according to TPC algorithm) at the highest rate Set to PA_P0, PA_P1, or PA_P2 (according to PA algorithm) at all other rates.

When power adaptation is enabled, Table 26 is used to adjust the power based on the current rate and manual control of the transmit power through the **CMD_802_11_RF_TX_POWER** API is disabled.

Table 26: Power Adaptation Settings

Rate (Mbps)	Power (dBm)
54, 48	PA_P0
12, 18, 24, 36	PA_P1
1, 2, 5.5, 11, 6, 9	PA_P2

The PA bias settings correspond to three different current levels depending on the transmit power as follows:

- Low current for 0 dBm to 5 dBm
- Medium current for 6 dBm to 10 dBm
- High current for 11 dBm to MaxPower dBm

Table 27 lists the supported transmit power control commands.

Table 27: Transmit Power Control Commands

Command	Description	Page
5.11.1 "CMD_802_11_TPC_CFG"	Controls TPC functionality	page 75
5.11.2 "CMD_802_11_PA_CFG"	Configures power adaptation related settings	page 76

5.11.1 CMD_802_11_TPC_CFG

The **CMD_802_11_TPC_CFG** command is used to control TPC functionality.

REQUEST

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11_TPC_CFG**
Size	UINT16	Number of bytes in command body
SeqNum	UINT16	Command sequence number
Result	UINT16	Not used (set to 0)
Action	UINT16	Action: 0 = ACT_GET 1 = ACT_SET
EnableTPC	UINT8	Enable automatic TPC at the highest rate
P0	SINT8	P0 power level for TPC
P1	SINT8	P1 power level for TPC
P2	SINT8	P2 power level for TPC
UseSNR	UINT8	Use SNR (in addition to PER) for TPC algorithm

RESPONSE

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11_TPC_CFG** | 0x8000
Size	UINT16	Number of bytes in command body
SeqNum	UINT16	Command sequence number, as sent by host
Result	UINT16	Result code (set to 0 for success)
Action	UINT16	Action: 0 = ACT_GET 1 = ACT_SET
EnableTPC	UINT8	Enable automatic TPC at the highest rate
P0	SINT8	P0 power level for TPC
P1	SINT8	P1 power level for TPC
P2	SINT8	P2 power level for TPC
UseSNR	UINT8	Use SNR (in addition to PER) for TPC algorithm

If EnableTPC is set, automatic TPC is enabled. If EnableTPC is cleared, then automatic TPC is disabled. The automatic TPC is disabled by default.

P0, P1, and P2 are the power levels used at the highest rate when TPC is enabled. The default values of P0, P1, and P2 are 5 dB, 10 dB, and 13 dB if the client is a 802.11g/802.11a client and 5 dBm,10 dBm, and 18 dBm if the client is a 802.11b client respectively.

If UseSNR is set, then averaged SNR is used in the automatic TPC algorithm, otherwise not.

5.11.2 CMD_802_11_PA_CFG

The **CMD_802_11_PA_CFG** command is used to configure power adaptation related settings.

REQUEST

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11_PA_CFG**
Size	UINT16	Number of bytes in command body
SeqNum	UINT16	Command sequence number
Result	UINT16	Not used (set to 0)
Action	UINT16	Action: 0 = ACT_GET 1 = ACT_SET
EnablePA	UINT8	Disable power adaptation based on current rate
PA_P0	SINT8	P0 power level for power adaptation
PA_P1	SINT8	P1 power level for power adaptation
PA_P2	SINT8	P2 power level for power adaptation

RESPONSE

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11_PA_CFG** | 0x8000
Size	UINT16	Number of bytes in command body
SeqNum	UINT16	Command sequence number, as sent by host
Result	UINT16	Result code (set to 0 for success)
Action	UINT16	Action: 0 = ACT_GET 1 = ACT_SET
EnablePA	UINT8	Disable power adaptation based on current rate
PA_P0	SINT8	P0 power level for power adaptation
PA_P1	SINT8	P1 power level for power adaptation
PA_P2	SINT8	P2 power level for power adaptation

If EnablePA is set, the rate based power adaptation is enabled in the firmware. Power adaptation is enabled by default.

PA_P0, PA_P1, and PA_P2 are the power levels to be used for different rates when power adaptation is enabled. The default values of these are 13, 15, and 18 dBm respectively.

Note

Please refer to the TPC and Rate Adaptation Application Note.

5.12 Event Subscription

Table 28 lists the supported event subscription commands.

Table 28: Event Subscription Commands

Command	Description	Page
5.12.1 "CMD_802_11_SUBSCRIBE_EVENT"	Subscribe to events and set thresholds	page 77

5.12.1 CMD_802_11_SUBSCRIBE_EVENT

The **CMD_802_11_SUBSCRIBE_EVENT** command allows the host driver to subscribe to a variety of different events and dynamically set thresholds specific to each event.

REQUEST

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11_SUBSCRIBE_EVENT**
Size	UINT16	Number of bytes in the command body
SeqNum	UINT16	Command sequence number
Result	UINT16	Not used (set to 0)
Action	UINT16	Action: 0 = ACT_GET 1 = ACT_SET
Events	UINT16	Bitmap of subscribed events

RESPONSE

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11_SUBSCRIBE_EVENT** | 0x8000
Size	UINT16	Number of bytes in the command body
SeqNum	UINT16	Command sequence number, as sent by host
Result	UINT16	Result code (set to 0 for success)
Action	UINT16	Action: 0 = ACT_GET 1 = ACT_SET
Events	UINT16	Bitmap of subscribed events

ACT_SET is used to change the set of subscribed events. ACT_GET is used to query the current set of subscribed events.

The Events field in the request is the bitmap of subscribed events. Events field in the response is the current bitmap of subscribed events. The default value of the Events bitmap is 0, which means no events are subscribed to.

The bits in the bitmap correspond to the events according to Table 29. Bit 0 is the least significant bit (LSB).

Table 29: Subscribed Events Bitmap Bits

Bit	Event
6-15	Reserved
5	SNR_HIGH—This event is generated when the average received SNR in beacons goes above a threshold. The threshold is specified by a TLV in the request.
4	RSSI_HIGH—This event is generated when the average received RSSI in beacons goes above a threshold. The threshold is specified by a TLV in the request.
3	LINK_LOSS—This event is generated when the number of consecutive beacons missed exceeds a threshold. The threshold is specified by a TLV in the request.
2	MAX_FAIL—This event is generated when the number of consecutive transmit failures exceeds a threshold. The threshold is specified by a TLV in the request.
1	SNR_LOW—This event is generated when the average received SNR in beacons goes below a threshold. The threshold is specified by a TLV in the request.
0	RSSI_LOW—This event is generated when the average received RSSI in beacons goes below a threshold. The threshold is specified by a TLV in the request.

To allow this command to be extensible, there can be variable length TLV fields after the aforementioned fixed fields in the command request. The allowed TLVs are:

- **MrvlIETypes_LowRssiThreshold_t** (see Section 8.4.4 "MrvlIETypes_LowRssiThreshold_t" on page 126)
- **MrvlIETypes_LowSnrThreshold_t** (see Section 8.4.5 "MrvlIETypes_LowSnrThreshold_t" on page 126)
- **MrvlIETypes_FailureCount_t** (see Section 8.4.6 "MrvlIETypes_FailureCount_t" on page 127)
- **MrvlIETypes_BeaconsMissed_t** (see Section 8.4.7 "MrvlIETypes_BeaconsMissed_t" on page 127)
- **MrvlIETypes_HighRssiThreshold_t** (see Section 8.4.16 "MrvlIETypes_HighRssiThreshold_t" on page 132)
- **MrvlIETypes_HighSnrThreshold_t** (see Section 8.4.17 "MrvlIETypes_HighSnrThreshold_t" on page 132)

5.13 Power Management

This section describes the various power management modes supported by Marvell devices.

Table 30 lists the supported power management commands.

Table 30: Power Management Commands

Command	Description	Page
Section 5.13.3 "Power Management Commands"		
5.13.3.1 "CMD_802_11_PS_MODE"	Sets/gets PS mode	page 81
5.13.3.2 "CMD_802_11_SLEEP_PARAMS"	Sets/gets sleep parameters	page 83
5.13.3.3 "CMD_802_11_HOST_SLEEP_CFG"	Used by the host to configure the host wakeup semantics before going to sleep	page 85
5.13.3.4 "CMD_802_11_WAKEUP_CONFIRM"	Sends a Host Awake event	page 86
5.13.3.5 "CMD_802_11_FW_WAKE_METHOD"	Firmware wake method	page 87
Section 5.13.7 "Deep Sleep Mode"		
5.13.7.1 "CMD_802_11_DEEP_SLEEP"	Initiates Deep Sleep mode	page 91

A station (client) can be in one of the two power management (PM) modes:

- Active mode—stays in Awake state
- Power Save (PS) mode—stays in either Awake or Doze state)

The AP always needs to be aware of the current power management mode that the station is in. For this reason, the station notifies the AP every time it switches modes. It sends a Null data frame with the PM bit set to 0 when entering Active mode or 1 when entering PS mode. It retries the Null frame until it is successfully acknowledged by the AP. If the station needs to switch to Active mode and there is some data to transmit, it uses the PM bit in the data frame to indicate the mode switch to the AP and avoids generating an extra Null frame.

The AP buffers all unicast traffic for a station in PS mode and indicates the presence of buffered traffic through Traffic Indication Map (TIM) information element (IE) in the beacon frame. Also, if any associated station is in PS mode, the AP delivers multicast traffic immediately following a Delivery Traffic Indication Message (DTIM) beacon only.

By default, the Marvell station always stays in Active mode. If the host enables PS mode, it activates an algorithm which switches between Active and PS mode as required. One of the parameters provided by the host driver is the number of DTIM periods the station is allowed to sleep before waking up to hear a beacon.

The algorithm operates as follows:

- When the station is in Active mode to start with, and if there is no transmit or receive activity for one beacon interval, the station decides to switch to PS mode and sends a Null packet to the AP to signal that activity.
- When the station is in PS mode, the station computes the total sleep time so that it wakes up a *wakeup offset* before the TBTT corresponding to the desired DTIM beacon. The total sleep time is based on the number of DTIM periods set by the host driver.
- The wakeup offset depends on the total time required to power up the device and the accuracy of the low power oscillator which is used during power down mode.
- When the station wakes up, it waits to receive a beacon. If the incoming beacon indicates unicast or multicast traffic or the host generates transmit packets, the station transitions to Active mode and sends a Null packet to the AP to signal that activity. If none of these conditions occur, the station again computes the sleep time and goes back to sleep.

- This process repeats as long as PS mode stays enabled. If the host disables PS mode, then the algorithm is deactivated and the station always stays in the Active mode.

5.13.1 Entering Power Save Mode

The host driver sends the command **CMD_802_11_PS_MODE** with the Action field set to PS_CMD_ENTER. After sending the command, the host driver must not send any data or commands to the firmware.

When firmware enters PS mode, a number of hardware blocks are turned off (hardware MAC, baseband processor, and RF chip). A wakeup timer starts that wakes up the WLAN device at the next expected beacon transmit time from the AP.

When in PS mode, the hardware bus interface to the host is turned off, as well. It is important that the host driver does not try to send commands or data to the firmware when it is sleeping.

When the listening interval equals DTIM or multiples of DTIM, the device achieves the lowest current consumption. The listening interval does not need to equal multiples of DTIM. Listening interval is implemented as follows:

- Listening interval equals DTIM or multiples of DTIM:
  - WLAN device wakes up on every DTIM.
- Listening interval is less than one DTIM interval (for example, DTIM = 10, LI = 3):
  - DTIM count is numbered as 0,9,8,7,6,5,4,3,2,1,0,9,8,7,6,5,4,3,2,1,0.
  - WLAN device wakes up at DTIM count = 0.
  - WLAN device also wakes up at DTIM count = 9, 6, 3, 9, 6, 3.
- Listening interval is larger than one DTIM interval
  - WLAN device wakes up at DTIM count = 0.
  - Listening interval is not used.
- WLAN device waking up on multiple DTIM interval
  - Listening interval is not used. WLAN device only wakes up on the specified DTIM interval.

5.13.2 Exiting Power Save Mode

To exit PS mode, the host driver sends the command **CMD_802_11_PS_MODE** with subcommand PS_CMD_EXIT to the firmware. After exiting PS mode, the WLAN device returns to normal operation.

5.13.3 Power Management Commands

This section describes the APIs between the driver and the firmware which enable the different power management modes.

5.13.3.1 CMD_802_11_PS_MODE

The **CMD_802_11_PS_MODE** command sets or gets the PS mode.

REQUEST

<table>
<tr><th>Field Name</th><th>Type</th><th>Description</th></tr>
<tr><td>CmdCode</td><td>UINT16</td><td><b>CMD_802_11_PS_MODE</b></td></tr>
<tr><td>Size</td><td>UINT16</td><td>Number of bytes in command body</td></tr>
<tr><td>SeqNum</td><td>UINT16</td><td>Command sequence number</td></tr>
<tr><td>Result</td><td>UINT16</td><td>Not used (set to 0)</td></tr>
<tr><td>Action</td><td>UINT16</td><td>Power save subcommand:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>PS_CMD_ENTER (0x0030)</td><td>Enter PS mode</td></tr>
<tr><td>PS_CMD_EXIT (0x0031)</td><td>Exit PS mode</td></tr>
<tr><td>PS_CMD_SLEEP_CFM (0x0034)</td><td>Host driver sends this subcommand to acknowledge receipt of the <i>sleep</i> event
By confirming the sleep event, the host driver then stops transmitting data to the firmware.</td></tr>
</table></td></tr>
<tr><td>NullPktInterval</td><td>UINT16</td><td>Null packet interval(s) for keep alive in Infra-Power Save mode
For other modes, this field is ignored.
0 = unchanged
0xFFFF = disable Null packet keep alive others
Null packet keep alive interval default is 0x001E (30 s).</td></tr>
<tr><td>NumDtims</td><td>UINT16</td><td>Number of DTIM intervals
Default = 0</td></tr>
<tr><td>Reserved</td><td>UINT16</td><td>Reserved (set to 0)</td></tr>
<tr><td>LocalListenInterval</td><td>UINT16</td><td>Local Listen Interval</td></tr>
</table>

RESPONSE

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11_PS_MODE** | 0x8000
Size	UINT16	Number of bytes in command body
SeqNum	UINT16	Command sequence number, sent by the host
Result	UINT16	Result code
Action	UINT16	Action specified in the request message

Field Name	Type	Description
NullPktInterval	UINT16	Null packet interval (s) for keep alive in Infra-Power Save mode For other modes, this field is ignored. 0 = unchanged 0xFFFF = disable Null packet keep alive others Null packet keep alive interval default is 0x001E (30 s).
NumDtims	UINT16	Number of DTIM intervals Default = 0
Reserved	UINT16	Reserved (set to 0)
LocalListenInterval	UINT16	Local Listen Interval

The action PS_CMD_ENTER is used to enter power save both in Infrastructure and Ad Hoc modes.

NumDtims indicates the number of DTIM intervals the firmware sleeps for. The valid value range for NumDtims is [0,5] and 0xFFFE. Setting to 0, the current value of numDtims in the firmware stays unchanged. If this is set to 0xFFFE, then the firmware does not wake up for DTIMs at all (the wakeup interval is completely governed by the listen interval).

LocalListenInterval indicates the value of listen interval used to determine wakeup instances. This value can be different from the listen interval that is sent to the AP in the association command. The valid value range of this is [0,20]. Setting to 0, the current value of listen interval in the firmware stays unchanged. If NumDtims is greater than 1 (and not 0xFFFE) or if the Listen Interval is larger than the DTIM period, the Listen Interval is ignored.

The action PS_CMD_EXIT is enabled to exit PS mode.

The action PS_CMD_SLEEP_CFM is used to send the sleep event and allows the firmware to put the device to sleep.

NullPktInterval indicates the time interval between two consecutive Null packets. Accordingly, Null packets are sent out at every pre-configured time interval whenever the firmware wakes up next time. The firmware does not wake up exclusively to transmit the Null packet. It transmits a Null packet next time it wakes-up due to the DTIM or Listen interval expiring.

5.13.3.2 CMD_802_11_SLEEP_PARAMS

The **CMD_802_11_SLEEP_PARAMS** command is used to set or get the various sleep parameters used for the correct power save operation. Implemented in 88W8381and 88W8385 based designs.

REQUEST

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11_SLEEP_PARAMS**
Size	UINT16	Number of bytes in command body
SeqNum	UINT16	Command sequence number
Result	UINT16	Not used (set to 0)
Action	UINT16	Action: 0 = ACT_GET 1 = ACT_SET
Error	UINT16	Sleep clock error (ppm)
Offset	UINT16	Wakeup offset (µs)
StableTime	UINT16	Clock stabilization time (µs)
CalControl	UINT8	Control periodic calibration
ExternalSleepClk	UINT8	Control the use of external sleep clock
Reserved	UINT16	Reserved (set to 0)

RESPONSE

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11_SLEEP_PARAMS** | 0x8000
Size	UINT16	Number of bytes in command body
SeqNum	UINT16	Command sequence number
Result	UINT16	Not used (set to 0)
Action	UINT16	Action: 0 = ACT_GET 1 = ACT_SET
Error	UINT16	Sleep clock error (ppm)
Offset	UINT16	Wakeup offset (µs)
StableTime	UINT16	Clock stabilization time (µs)
CalControl	UINT8	Control periodic calibration
ExternalSleepClk	UINT8	Control the use of external sleep clock
Reserved	UINT16	Reserved

Error is the sleep clock error in parts per million (ppm). The valid range is 0 to 65535 ppm. The default value is 5000 ppm.

Offset is the wakeup offset from the TBTT in microseconds (µs). This denotes the total time from when the stable clock is available to when the wireless system is ready to receive packets (RF fully settles down). The valid range is 0 to 65535 µs. The default value is 500 µs.

StableTime is the time taken by the oscillator to stabilize (µs). This determines the amount of time that the PMU state machine inserts between enabling the oscillator and enabling the PLL. This should be set to the maximum time elapsed between the clock enable (GPIO[6]) being asserted and a stable oscillator output being available. The valid range is 0 to 65535 µs. The default value is 4000 µs.

CalControl determines whether the periodic calibration is turned on or not. The valid range is 0 to 2. If this is set to 1, the periodic calibration is enabled. If this is set to 2, the periodic calibration is disabled. Disable periodic calibration when the high accuracy external sleep clock is being used. The default value is 1, which means that the periodic calibration is enabled by default.

ExternalSleepClk determines whether the external sleep clock is used or not. The valid range of values is 0 to 2. If this is set to 1, the internal sleep clock is used. If this is set to 2, the external sleep clock is used. The default value is 1, which means that the internal sleep clock is used by default. The host driver must be careful to set this to 2 only when an external sleep clock is connected.

Reserved is reserved for allowing this command to be easily extended in the future without changing its length.

Setting any of the above parameters to 0 keeps the value of that parameter in the firmware unchanged.

This command can be invoked by the driver at any time and the change in any value(s) takes effect immediately.

5.13.3.3 CMD_802_11_HOST_SLEEP_CFG

The **CMD_802_11_HOST_SLEEP_CFG** command is used by the host to configure the wakeup semantics of the host before it goes to sleep.

REQUEST

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11_HOST_SLEEP_CFG**
Size	UINT16	Number of bytes in the command body
SeqNum	UINT16	Command sequence number
Result	UINT16	Not used (set to 0)
Criteria	UINT32	Criteria to wake up the host
GPIO	UINT8	0xFF = if wakeup through GPIO is not required All others = GPIO pin number to use for wakeup
Gap	UINT8	Time (ms) to wait between triggering the GPIO pin and sending the awake event
Optional	**MrvlIETypes_HostSleepFilterType1**	EthType packets that can wake up the host See Section 8.4.15 "MrvlIETypes_HostSleepFilterType1" on page 131 for more information.

RESPONSE

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11_HOST_SLEEP_CFG** | 0x8000
Size	UINT16	Number of bytes in the command body
SeqNum	UINT16	Command sequence number, as sent by host
Result	UINT16	Result code (set to 0 for success)
Criteria	UINT32	Same as in request
GPIO	UINT8	Same as in request
Gap	UINT8	Same as in request
Optional	**MrvlIETypes_HostSleepFilterType1**	Same as in request

Criteria is the bitmap describing the conditions which should cause the firmware to wake up the host. The various conditions are:

- Bit 0 = reception of any broadcast data
- Bit 1 = reception of any unicast data
- Bit 2 = generation of any MAC event
- Bit 3 = generation of any multicast data

Note that the host is always woken up if any of the events corresponding to the link loss happens. Further notes on host wakeup conditions are:

- Host wakeup condition is *created* in the firmware using Criteria value other than 0xFFFFFFFF.
- Host wakeup condition is *removed* from the firmware using Criteria value 0xFFFFFFFF.
- Once the host wakeup condition is *created*, it is retained across link losses and re-associations until it is *removed*.
- Host wakeup condition can be *created* as many times as the host wishes. Only the most recent condition is retained by the firmware.
- At initialization, the host wakeup condition is set to 0xFFFFFFFF in the firmware. In other words, no host wakeup condition is *created* by default.
- Host can go to sleep only when the host wakeup condition is both *created* and *activated*.
- Host wakeup condition is *activated* whenever the firmware receives a **CMD_802_11_WAKEUP_CONFIRM** or **CMD_802_11_DEEP_SLEEP** command provided the condition has been *created*.
- Any command from the host driver *deactivates* the host wakeup condition.

GPIO indicates the GPIO pin number that is triggered to wake up the host. If this is set to 0xFF, then no GPIO pin is triggered.

Gap indicates the time in milliseconds that the firmware must wait after triggering the GPIO pin before it generates the PS Awake event. This field is ignored if GPIO is set to 0xFF.

The optional **MrvlIETypes_HostSleepFilterType1** TLV is used to further specify the EthType packets that can wake up the host. Its value is 277 (0x115).

5.13.3.4 CMD_802_11_WAKEUP_CONFIRM

The **CMD_802_11_WAKEUP_CONFIRM** command is used during host-initiated wake method. The firmware sends a Host Awake (MAC) event to the host and the host responds to this event by sending this command.

REQUEST

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11_WAKEUP_CONFIRM**
Size	UINT16	Number of bytes in command body
SeqNum	UNIT16	Command sequence number
Result	UINT16	Not used (set to 0)

RESPONSE

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11_WAKEUP_CONFIRM** | 0x8000
Size	UINT16	Number of bytes in command body
SeqNum	UNIT16	Command sequence number, as sent by host
Result	UINT16	Result code (0 on success)

5.13.3.5 CMD_802_11_FW_WAKE_METHOD

The **CMD_802_11_FW_WAKE_METHOD** command is used by the host to indicate to the firmware how it wishes to wakeup when the firmware is in Power Save mode or Deep Sleep mode. This command is issued by the host prior to enabling the firmware into any Power Save mode. This command is enabled once, the firmware remembers this setting until it resets.

REQUEST

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11_FW_WAKE_METHOD**
Size	UINT16	Number of bytes in command body
SeqNum	UNIT16	Command sequence number
Result	UINT16	Not used (set to 0)
Action	UINT16	Action: 0 = ACT_GET 1 = ACT_SET
Method	UINT16	Method to wakeup firmware

RESPONSE

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11_FW_WAKE_METHOD** | 0x8000
Size	UINT16	Number of bytes in command body
SeqNum	UNIT16	Command sequence number
Result	UINT16	Not used (set to 0)
Action	UINT16	Same as request
Method	UINT16	Method to wakeup firmware

Method indicates the method by which the host intends to wakeup the firmware. This can be one of the following:

- 0 = Leave the current method to wakeup firmware unchanged
- 1 = Firmware wakeup through the command interrupt
- 2 = Firmware wakeup through the GPIO pin

The default method for firmware wakeup is through the command interrupt for SDIO and GSPI interfaces.

5.13.4 Power Save Events

The firmware sends the following events to the host driver when in PS mode:

- Awake event
- Host Wakeup event
- Sleep event
- Deep Sleep Awake event

5.13.4.1 Awake Event

This event tells the host driver that the firmware is awake and ready. The method in which the host uses to wakeup the device depends on the host interface.

5.13.4.1.1 SDIO Interface

The firmware generates the Awake event to the host driver as follows:

1. Write 0x32 into Scratch_Register_1.
2. Write (0x0A << 3) into Scratch_Register_2.
3. Generate a Card Upload Ready interrupt to the host.

5.13.4.1.2 CF Interface

The firmware generates the Awake event to the host driver as follows:

1. Write ((0x0A << 8) | 0x10) into the CF Card Status register.
2. Generate a card event interrupt to the host.

5.13.4.1.3 G-SPI Interface

The firmware generates the Awake event to the host driver as follows:

1. Writes to the scratch pad register 3 (SP3) the value 0x0A or 0x0B.
2. Generates a CardEvent by writing to the CICR (Card Intr Cause Reg).

5.13.4.2 Host Wakeup Event

The firmware generates a host wakeup event when the host tries to wakeup the firmware or by indicating that the firmware has woken up.

5.13.4.3 Sleep Event

This event tells the host driver that the firmware is about to go into Sleep mode.

5.13.4.3.1 SDIO Interface

The firmware generates the Sleep event to the host driver as follows:

1. Write 0x32 into Scratch_Register_1.
2. Write (0x0B << 3) into Scratch_Register_2.
3. Generate a Card Upload Ready interrupt to the host.

5.13.4.3.2 CF Interface

The firmware generates the Sleep event to the host driver as follows:

1. Write ((0x0B << 8) | 0x10) into the CF Card Status register.
2. Generate a card event interrupt to the host.

5.13.4.3.3 G-SPI Interface

The firmware generates the Sleep event to the host driver as follows:

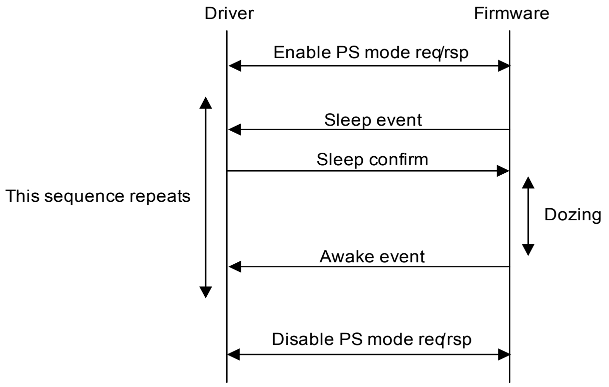
1. Write ((0x40 << 0)) into SPI Host Interface register 2.

5.13.5 Driver/Firmware Interaction

Every time the station wakes up, the firmware immediately sends an event notification to the driver to indicate that it is now awake.

Every time the firmware decides to put the system to sleep, it sends an event notification to the driver to indicate an action to put the system to sleep. The firmware then waits for a confirmation message from the host driver before it actually puts the system to sleep. Figure 4 shows this interaction.

Figure 4: Driver/Firmware Interaction

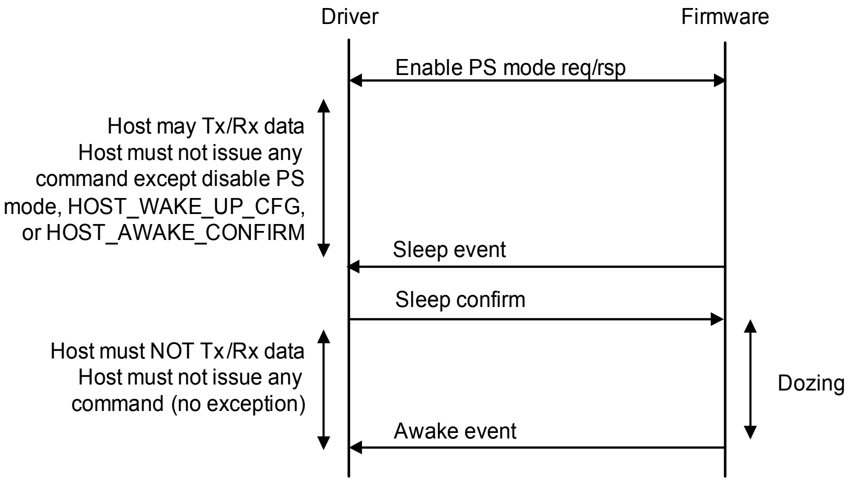


5.13.6 Assumptions

The host must not issue any command (except the disable PS mode command or **CMD_802_11_WAKEUP_CONFIRM** command) after it enables PS mode. It may, however, continue to process data transmit and receive.

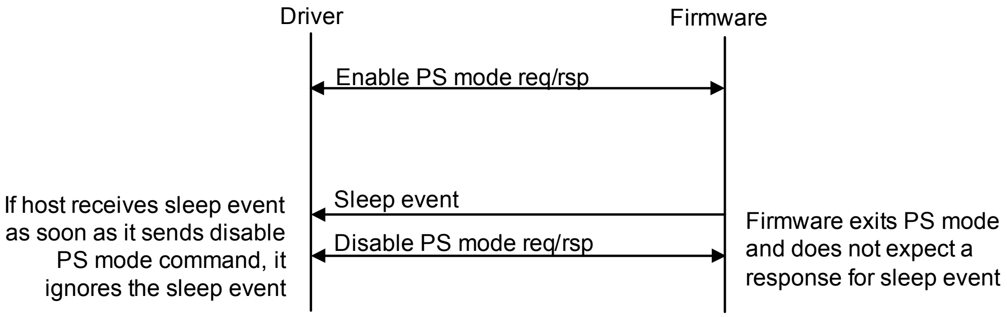
After sending the sleep confirm message, the host must not issue the disable PS mode command until it receives an awake indication. It should also not attempt to transmit or receive any data during this window (see Figure 5).

Figure 5: PS Mode—Assumption 1



It is possible that the sleep event from firmware and disable PS command from the host are generated simultaneously. In this case, the firmware honors the disable PS command and exits PS mode. The host ignores the sleep event in this case. This is depicted in Figure 6.

Figure 6: PS Mode—Assumption 2



5.13.7 Deep Sleep Mode

The host may put the WLAN subsystem in Deep Sleep mode, during which the WLAN subsystem does not wakeup at all. The host sends a signal through the GPIO to wakeup the WLAN subsystem. The wakeup mechanism is interface-dependent.

5.13.7.1 CMD_802_11_DEEP_SLEEP

In order to initiate Deep Sleep mode, the driver issues the **CMD_802_11_DEEP_SLEEP** command. The firmware immediately puts the system into Deep Sleep mode when receiving this command.

REQUEST

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11_DEEP_SLEEP**
Size	UINT16	Number of bytes in command body
SeqNum	UINT16	Command sequence number
Result	UINT16	Not used (set to 0)

There is no response to this command.

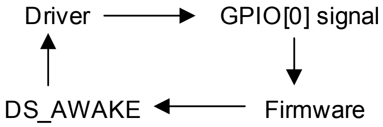
Notes

- When the driver issues the **CMD_802_11_DEEP_SLEEP** command to the device (firmware) it should wait for at least one second before it wakes up the device (firmware) from Deep Sleep mode.
- Once the driver wakes up the device (firmware) from Deep Sleep mode, it should wait for one second before sending traffic or putting the device into Deep Sleep mode again.

The last two limitations are in place to guard instability time when the device is waking up or going to sleep.

5.13.7.2 Deep Sleep Awake Event

To exit Deep Sleep mode, the driver triggers the GPIO pin when the user requests it exit Deep Sleep mode. After awakened by the GPIO signal, the firmware sends a DS_AWAKE event to the driver.



For the CF interface only, the host driver can read or write any register to trigger the GPIO[0] signal.

After awaking from Deep Sleep mode, the device wakes up and waits for future commands.

Note

The GPIO[0] signal is level triggered for at least 30 µs. For the 88W8381 based designs, a 500 ns level triggered signal should be applied.

5.13.7.3 Deep Sleep Mode Assumptions

Deep Sleep mode of operation is supported only when the system is in Idle (not associated) mode.

It is the responsibility of the host driver to ensure that Deep Sleep mode never gets invoked when the system is not in Idle mode.

Also, it is recommended that after issuing the **CMD_802_11_DEEP_SLEEP** command, the driver should wait for at least one second and allow the device to settle down before exiting Deep Sleep mode by triggering the GPIO pin.

5.13.8 APSD

Automatic Power Save Delivery (APSD) refers to the enhanced mechanisms introduced in the 802.11e specification to allow power save operation in the presence of periodic traffic.

5.13.8.1 APSD Overview

These additional mechanisms can only be used if APSD is supported by both the client (stations) and the AP. There are two modes of operations for APSD:

- Scheduled
- Unscheduled

Scheduled APSD requires negotiation of a delivery schedule between the AP and the client. The schedule specification includes service start time and service interval. A scheduled service period (SP) starts at the service start time and every service interval later thereafter. The client must be ready to receive traffic from the AP at the start of each SP.

Unscheduled APSD (also referred to as UPSD) does not require setting up an explicit schedule. However, the client may choose to enable only certain access categories (AC) to be serviced using this mechanism. Specifically, the client may choose to trigger-enable certain ACs and delivery-enable certain other ACs. If an AC is trigger-enabled, then any uplink QoS Data/Null frame belonging to that AC triggers an unscheduled SP. If an AC is delivery-enabled, then it is serviced by the AP during an unscheduled SP (in the order of priority).

The end of service period (EOSP) bit in the MAC header of the transmitted packets indicates the end of a service period (for both scheduled and unscheduled SP). When the SP ends, the client may power down until the start of the next SP.

The number of frames the AP is allowed to send in a single SP is controlled by the client. It chooses between 2, 4, 6, or all of the frames buffered at the AP to be delivered during a single SP. This information is conveyed to the AP during the association.

This section describes the design for the UPSD mode of operation. The scheduled version of APSD is not being implemented currently.

5.13.8.1.1 CMD_802_11_SLEEP_PERIOD

The default infrastructure power save algorithm allows the device to go to sleep only after at least one beacon interval of inactivity. Also, once it puts the device to sleep, the device is only woken up before the designated TBTT (depending on listen interval and DTIM period).

This document describes some enhancements to the basic power save algorithm to allow power savings in the presence of frequent periodic traffic.

The formats of the **CMD_802_11_SLEEP_PERIOD** command and the corresponding response are as follows.

REQUEST

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11_SLEEP_PERIOD**
Size	UINT16	Number of bytes in command body
SeqNum	UNIT16	Command sequence number
Result	UINT16	Not used (set to 0)
Action	UINT16	Action: 0x0000 = ACT_GET 0x0001 = ACT_SET
SleepPeriod	UINT16	Sleep period (ms)

RESPONSE

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11_SLEEP_PERIOD** | 0x8000
Size	UINT16	Number of bytes in command body
SeqNum	UNIT16	Command sequence number, as sent by host
Result	UINT16	Result code (set to 0 for success)
Action	UINT16	Same as request
SleepPeriod	UINT16	Sleep period (ms)

The SleepPeriod field specifies the interval between consecutive wakeup instances of the device (ms). Typically, this is much less than the beacon interval. The valid range for this is [10, 60]. A value of 0 indicates that the sleep period is disabled. In other words, the device sleeps and wakes up for DTIM beacons as per the legacy power save procedure. The default value is 0.

5.13.8.2 APSD Initialization

For APSD initialization, WMM must be enabled in a BSS before the UPSD is enabled. The AP advertises its support for the UPSD by setting the UPSD bit in the QoS Info field of the WMM IE in the beacons and probe responses transmitted by it.

If the AP does not support the UPSD, the client cannot use the mechanisms provided by the UPSD. If the AP supports the UPSD, then the client may choose to utilize the mechanisms provided by the UPSD.

The client includes the WMM IE in the (re)association request frame to indicate to the AP that it wishes to use the WMM. Within the WMM IE, it uses the QoS Info field to request the use of the UPSD.

The format of QoS Info field is as listed in Table 31.

Table 31: QoS Information

Bit	Description
7	Reserved
6:5	Maximum SP Length
4	Reserved
3	AC_BEU_APSD Flag
2	AC_BKU_APSD Flag
1	AC_VIU_APSD Flag
0	AC_VOU_APSD Flag

The WMM IE may be included in the association command sent by the driver to the firmware.

The driver sets this field by:

- Setting the reserved fields to 0
- Setting the Max SP Length field to 0 (recommended)
- To enable UPSD, set all four UPSD flags
- UPSD is not enabled when no flags are set

The current design does not support scenarios where one, two, or three UPSD flags are set, because the implementation is significantly more complex. These configurations do not enable any compelling use cases that are not covered by what is supported.

Therefore, when UPSD is enabled, all four UPSD flags must be set.

In this case, all the downlink traffic from the AP is delivery-enabled, and all the uplink traffic from the client is trigger-enabled. Any uplink transmission triggers the delivery of downlink packets (in the order of priority).

The TIM bit as well as the More Data bit indicates the presence of more buffered traffic at the AP belonging to any AC.

The behavior of the firmware in this case depends on the value of the sleep period that has been set (using the sleep period API) by the driver. The two cases are described in the following sections.

5.13.8.2.1 Zero Sleep Period

If the sleep period is set to zero, there is no periodic traffic (such as a voice call) in progress. This operates very similar to the legacy PS behavior.

Each time the device wakes up to receive a beacon, it inspects the TIM bit in the beacon to determine whether there is any traffic buffered at the AP or not. If the TIM bit is set, it generates a QoS Null frame, which triggers the delivery of all the frames buffered at the AP. It also monitors the EOSP bit in the received frames and goes back to sleep after either the More Data or the EOSP bit goes to 0.

If the TIM indicates buffered multicast/broadcast frames, it waits to receive all multicast/broadcast frames as well.

During this entire exchange the device stays in PS state, which is indicated by the PM bit being set in all frames transmitted by the device (including the QoS Null).

5.13.8.2.2 Non-Zero Sleep Period

If the sleep period is set to a non-zero value, there is some sort of periodic traffic (such as a voice call) in progress. In this case, the operation is as follows.

The firmware wakes up with the period specified by the host. In addition, it may wakeup at the DTIM beacons (if required by the host) in order to receive any multicast traffic. It may also periodically wakeup to receive beacons in order to synchronize the value of local TSF counter.

Each time it wakes up, it sends an Awake event to the host driver.

Then, it waits to receive a packet from the host. If the host does not have any packets ready at that time, it is recommended that the host generate a Null frame to minimize the time the firmware stays up and consequently maximize the power savings. Section 3.2 "Transmit Packet Descriptor" on page 24 describes how the driver generates a Null frame.

When receiving, firmware transmits a packet (or Null Frame) as it’s received. The firmware continues as long as it keeps receiving data with the EOSP bit cleared. When it receives the EOSP indication, the firmware prepares to go back to sleep.

Before going back to sleep, firmware does the following:

- Sends sleep request to host and waits for sleep confirm response
- Computes time until next wakeup instant, and programs sleep timer accordingly

5.13.8.3 APSD Last Packet Indication

The firmware implementation depends on the driver supporting the last packet indication in the transmit packets.

When the driver has no more buffered packets to transfer to the firmware, it sets the last packet bit in the transmit info header of the packet. The firmware always waits to receive the last packet indication (except when time-out occurs) before it initiates the sequence for going back to sleep.

The driver must guarantee that it does not transfer any more packets to the firmware (after it sends this indication) until it receives another awake event. This allows the firmware to determine the last trigger sent and determine the last EOSP indication that received from the AP. See Section 3.2 "Transmit Packet Descriptor" on page 24 for the API setting of the last packet indicator.

5.13.8.4 APSD Driver Requirements

The optimal operation of the UPSD depends on co-operation between driver and the firmware. Steps the driver takes to ensure maximum power savings is obtained include:

- Respond to sleep request with sleep confirm in a timely manner:
  - If the driver is in the middle of a packet download when it receives a sleep request, it waits until that transaction is complete, and then issues a sleep confirm (without initiating another transaction).
  - If the driver is waiting for a command response when it receives a sleep request, it waits until it receives the response, and then issues a sleep confirm (without initiating another transaction).
  - If the driver is in the middle of receiving a packet when it receives a sleep request, it waits until that transaction is complete, and then issues a sleep confirm (without initiating another transaction).
- When receiving an Awake event, it promptly transfers all the queued packets to the firmware.
- Sets the last packet indicator in the last packet it transfers to firmware.
- After setting the last packet indicator, it should not transfer packets to the firmware until the next awake event (even if received before the sleep request from firmware).
- If there are no queued packets when the awake event is received, sends a Null frame with the last packet indicator set.

Note

Please refer to the APSD Application Note.

5.14 Bluetooth Coexistence

The driver/application configures the WLAN MAC block via firmware for Bluetooth Coexistence Arbitration (BCA) by using the existing command. See Section 5.3.3 "CMD_MAC_REG_ACCESS" (Hardware MAC Register Access) to peek and poke WLAN MAC hardware registers directly. While using this method, the driver/application level must have enough knowledge to program the low level WLAN MAC registers for BCA functionality. In addition, the **CMD_802_11_BCA_CONFIG_TIMESHARE** is used to configure the BCA timeshare interval and percentage of time in this timeshare interval.

5.14.1 CMD_802_11_BCA_CONFIG_TIMESHARE

The **CMD_802_11_BCA_CONFIG_TIMESHARE** command configures the BCA timeshare interval and percentage of time in the timeshare interval.

For bluetooth and WLAN coexistence, WLAN traffic always wins arbitration when low priority bluetooth and low priority WLAN traffic are both subjected during an arbitration window. In addition, WLAN traffic always wins arbitration when high priority bluetooth and high priority WLAN traffic is subjected to arbitration. The high priority WLAN traffic is limited to frame types of very short duration and so should cause minimal interference to bluetooth traffic. However, low priority WLAN traffic includes data frames of a long duration and high throughput. This may cause low priority bluetooth traffic to stall as WLAN is always preferred by the arbiter by default.

The host driver controls access to allow bluetooth traffic. Using the command, the host driver can specify the total time for the timeshare interval and percentage of time (in the timeshare interval) to be given to bluetooth traffic in cases where equal priority bluetooth and WLAN traffic is seen by the arbiter.

This API is implemented in 88W8385 based designs.

REQUEST

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11_BCA_CONFIG_TIMESHARE**
Size	UINT16	Number of bytes in command body
SeqNum	UNIT16	Command sequence number
Result	UINT16	Not used (set to 0)
Action	UINT16	Action: 0 = ACT_GET 1 = ACT_SET
TrafficType	UINT16	Type of traffic for which to configure this timeshare interval: 0 = WLAN and bluetooth are low priority 1 = WLAN and bluetooth are high priority
TimeshareInterval	UINT32	Total timeshare interval (ms) Valid range from 20 ms to 60000 ms in multiples of 10 ms. If value specified is not in multiples of 10 then a floor value (multiple of 10) is used. ACT_SET = configures the timeshare interval ACT_GET = value ignored

Field Name	Type	Description
BTTime	UINT32	Bluetooth time (ms) Time interval within TimeshareInterval where the PTA arbiter selects bluetooth traffic over WLAN traffic, in the case where both bluetooth and WLAN traffic have equal priority. This value should be in multiples of 10 with a valid range from 0 to TimeshareInterval. If the specified value is not multiples of 10 then a floor value (multiple of 10) is used. ACT_SET = configures the bluetooth time interval ACT_GET = value ignored

RESPONSE

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11_BCA_CONFIG_TIMESHARE** | 0x8000
Size	UINT16	Number of bytes in command body
SeqNum	UNIT16	Command sequence number
Result	UINT16	Result code (set to 0 on success)
Action	UINT16	Action: 0 = ACT_GET 1 = ACT_SET
TrafficType	UINT16	Type of traffic for which to configure this timeshare interval: 0 = WLAN and bluetooth are low priority 1 = WLAN and bluetooth are high priority
TimeshareInterval	UINT32	Total timeshare interval (ms) ACT_SET = configures the timeshare interval ACT_GET = returns the timeshare value currently used by firmware
BTTime	UINT32	Bluetooth time (ms) ACT_SET = configures the bluetooth time interval ACT_GET = returns the bluetooth time value currently used by the firmware

5.15 WMM

Table 32 lists the supported WMM commands.

Table 32: WMM Commands

Command	Description	Page
5.15.1 "CMD_WMM_GET_STATUS"	Retrieves the current WMM state	page 99
5.15.2 "CMD_WMM_ACK_POLICY"	Specifies and retrieves the data packet acknowledgement scheme used for WMM traffic	page 100

WMM can be enabled or disabled in the firmware through the **CMD_MAC_CONTROL** command. See Section 5.2.1 "CMD_MAC_CONTROL" on page 36 for details.

5.15.1 CMD_WMM_GET_STATUS

The **CMD_WMM_GET_STATUS** command is issued when the driver/host wants to retrieve the current WMM state in the firmware. The driver layer is required to issue this command in response to a WMM Status Change Event (Event code 23, see Section 6. "MAC Events" on page 113).

The **CMD_WMM_GET_STATUS** command does not take any input arguments, but must provide sufficient buffer space for the expected response. The command response is TLV based and returns a Queue Status TLV for each queue as well as a TLV for the WMM Parameter IE if it has been received by the firmware while associated to a BSS.

REQUEST

Field Name	Type	Description
CmdCode	UINT16	**CMD_WMM_GET_STATUS**
Size	UINT16	Number of bytes in command body
SeqNum	UINT16	Command sequence number
Result	UINT16	Not used (set to 0)
ResponseBuffer	UINT8[]	Not used (set to 0) Provide sufficient buffer space for the WMM IE Parameter TLV and the four Queue Status TLVs as shown in the response.

RESPONSE

Field Name	Type	Description
CmdCode	UINT16	**CMD_WMM_GET_STATUS** | 0x8000
Size	UINT16	Number of bytes in command body
SeqNum	UINT16	Command sequence number; as sent by host
Result	UINT16	Result code (set to 0 for success)
QueueStatus	**MrvlIETypes_WmmQStatus_t**[4]	Queue Status TLV for each AC queue See 8.4.13 "MrvlIETypes_WmmQStatus_t" on page 130 for details.
WMM Parameter IE	MrvlIETypes_VendorParamSet_t	Vendor specific IEEE IE with extended 16-bit type and length fields This vendor specific IE contains the WMM Parameter IE as described by the WMM Specification.

The result from the **CMD_WMM_GET_STATUS** command when issued in response to a WMM Status Change event is four Queue Status TLVs (one for each AC), followed by the WMM Parameter IE without any intermediate pad bytes or reserved fields.

The WMM Parameter IE is a Marvell extended version of the standard IEEE IE. The fields and layout of the TLV match the IEEE IE exactly except for the 16-bit element ID (0xDD; 221) and the 16-bit element length (currently 24 bytes by the standard).

5.15.2 CMD_WMM_ACK_POLICY

The **CMD_WMM_ACK_POLICY** API is used to specify and retrieve the data packet acknowledgement scheme. This API is used by firmware for WMM traffic. Firmware supports two acknowledgement policies:

- Immediate acknowledgement
- No acknowledgement

The acknowledgement policies are supported per Access Category. The firmware specifies the acknowledgment policy in the QoS control field of the MAC header.

REQUEST

Field Name	Type	Description
CmdCode	UINT16	**CMD_WMM_ACK_POLICY**
Size	UINT16	Number of bytes in command body
SeqNum	UNIT16	Command sequence number
Result	UINT16	Not used (set to 0)
Action	UINT16	Action: 0 = ACT_GET 1 = ACT_SET
User Priority	UINT8	User priority for which the ACK policy is to be set
AckPolicy	UINT8	WMM_ACK_POLICY_IMM_ACK WMM_ACK_POLICY_NO_ACK

RESPONSE

Field Name	Type	Description
CmdCode	UINT16	**CMD_WMM_ACK_POLICY** | 0x8000
Size	UINT16	Number of bytes in command response
SeqNum	UNIT16	Command sequence number, as sent by host
Result	UINT16	Result code (set to 0 for success)
Action	UINT16	Same as in command
User Priority	UINT8	Same as in command
AckPolicy	UINT8	Same as in command

It is not necessary for the driver to enable WMM before this command can be used. If this command is issued before enabling WMM, the command simply stores (in case of ACT_SET) the ACK policy for future use when WMM is enabled. If this command is issued during an active WMM association, then ACK policy change takes effect immediately.

The ACK policy specified through this command has no effect on non-WMM associations.

Using action field value of ACT_GET, the host driver retrieves the ACK policy currently used by the firmware.

Note

Please refer to the WMM Application Note.

5.16 802.11a

The **CMD_802_11_BAND_CONFIG** API is used to support 802.11a operations and is only applicable to Marvell 802.11a WLAN devices.

5.16.1 CMD_802_11_BAND_CONFIG

The **CMD_802_11_BAND_CONFIG** API is used to set or get the RF band settings.

REQUEST

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11_BAND_CONFIG**
Size	UINT16	Number of bytes in command body
SeqNum	UNIT16	Command sequence number
Result	UINT16	Not used (set to 0)
Action	UINT16	Action: 0 = ACT_GET 1 = ACT_SET
BandSelection	UINT16	Band selection: 0x0 = 802.11b (2.4 GHz) 0x1 = 802.11g (2.4 GHz) 0x2 = 802.11a (5 GHz) 0x3 = 802.11j (4 GHz)
Channel	UINT16	Channel number
Channel Width	UINT16	Channel width: 0 = CHAN_WIDTH_20MHZ 1 = CHAN_WIDTH_10MHZ

RESPONSE

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11_BAND_CONFIG** | 0x8000
Size	UINT16	Number of bytes in command body
SeqNum	UNIT16	Command sequence number
Result	UINT16	Result code
Action	UINT16	Action: 0 = ACT_GET 1 = ACT_SET
BandSelection	UINT16	0x0 = 802.11b (2.4 GHz) 0x1 = 802.11g (2.4 GHz) 0x2 = 802.11a (5 GHz) 0x3 = 802.11j (4 GHz)

Field Name	Type	Description
Channel	UINT16	Channel number
Channel Width	UINT16	Channel width: 0 = CHAN_WIDTH_20MHZ 1 = CHAN_WIDTH_10MHZ

Whenever the driver needs to send **CMD_802_11_SCAN**, **CMD_802_11_AD_HOC_START**, or **CMD_802_11_AD_HOC_JOIN** commands to firmware, it must first to send **CMD_802_11_BAND_CONFIG** in advance to indicate which band it is going to use. Firmware uses the band to enable 802.11a, 802.11g, or 802.11b operation. If 802.11g is selected, 802.11b operation is included automatically.

After scan is completed, the driver sends a **CMD_802_11_BAND_CONFIG** command to put the station in the correct operating band and channel.

5.17 802.11d

Table 33 lists the supported 802.11d commands.

Table 33: 802.11d Commands

Command	Description	Page
5.17.1 "CMD_802_11D_DOMAIN_INFO"	Sets/gets 802.11d domain information	page 103
5.17.2 "CMD_802_11_RGN_CODE"	Sets/gets region code stored in the EEPROM	page 104

5.17.1 CMD_802_11D_DOMAIN_INFO

Enabling 802.11d is controlled by the existing firmware SNMP command with OID 9. Writing a value of 0x01 (unsigned byte) enables 802.11d and writing 0x00 (unsigned byte) disables 802.11d.

The **CMD_802_11D_DOMAIN_INFO** command is used to get or set 802.11d domain information.

REQUEST

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11D_DOMAIN_INFO** (0x005b) (0x00xx)
Size	UINT16	Number of bytes in command body
SeqNum	UINT16	Command sequence number
Result	UINT16	Not used (set to 0)
Action	UINT16	Action: 0 = ACT_GET 1 = ACT_SET
Domain	**MrvlIETypes_DomainParam _t**	802.11 domain parameters TLV

RESPONSE

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11D_DOMAIN_INFO** | 0x8000
Size	UINT16	Number of bytes in command body
SeqNum	UINT16	Command sequence number sent by host
Result	UINT16	Result Code
Action	UINT16	Action: 0 = ACT_GET 1 = ACT_SET
Domain	**MrvlIETypes_DomainParam _t**	802.11 domain parameters TLV

For Get Domain Information command, the driver should send 10 bytes (8 bytes host header plus 2 bytes of ACT_GET) to firmware and the firmware appends the Domain Information in the response. For Set Domain Information command, the driver should send 10 bytes (8 bytes host header plus 2 bytes of ACT_GET) plus Domain Information to firmware, firmware parses the domain information, and returns the command request buffer with result fields set appropriately.

5.17.2 CMD_802_11_RGN_CODE

The host driver uses the **CMD_802_11_RGN_CODE** command to set or get the region code stored in the EEPROM memory of the WLAN device. The EEPROM is write-protected during normal operation of the WLAN device. Write-protection is a hardware feature of the EEPROM.

One GPIO line is designated for the EEPROM write protection feature. The WLAN firmware, in concert with the power up sequence of the WLAN device, ensures that the EEPROM write protect pin remains asserted at all times. The pin designated for write protection is de-asserted during manufacturing only.

REQUEST

<table>
<tr><th>Field Name</th><th>Type</th><th>Description</th></tr>
<tr><td>CmdCode</td><td>UINT16</td><td><b>CMD_802_11_RGN_CODE</b></td></tr>
<tr><td>Size</td><td>UINT16</td><td>Number of bytes in command body</td></tr>
<tr><td>SeqNum</td><td>UINT16</td><td>Command sequence number</td></tr>
<tr><td>Result</td><td>UINT16</td><td>Not used (set to 0)</td></tr>
<tr><td>Action</td><td>UINT16</td><td>Action:
0 = ACT_GET
1 = ACT_SET</td></tr>
<tr><td>RgnCode</td><td>UINT16</td><td>Region code
The value for the region code must conform to Table 50 in the document <i>ANSI/IEEE Std 802.11, 1999 Edition</i>.
<table>
<tr><th>Value</th><th>Region</th></tr>
<tr><td>0x10</td><td>United States</td></tr>
<tr><td>0x20</td><td>Canada</td></tr>
<tr><td>0x30</td><td>Most of Europe</td></tr>
<tr><td>0x31</td><td>Spain</td></tr>
<tr><td>0x32</td><td>France</td></tr>
<tr><td>0x40</td><td>Japan</td></tr>
</table>
</td></tr>
</table>

RESPONSE

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11_RGN_CODE** | 0x8000
Size	UINT16	Number of bytes in command body
SeqNum	UINT16	Command sequence number, sent by the host
Result	UINT16	Result code

<table>
<tr><th>Field Name</th><th>Type</th><th>Description</th></tr>
<tr><td>Action</td><td>UINT16</td><td>Action:
0 = ACT_GET
1 = ACT_SET</td></tr>
<tr><td>RgnCode</td><td>UINT16</td><td>Region code
The value for the region code must conform to Table 50 in the document <i>ANSI/IEEE Std 802.11, 1999 Edition</i>.
<table>
<tr><th>Value</th><th>Region</th></tr>
<tr><td>0x10</td><td>United States</td></tr>
<tr><td>0x20</td><td>Canada</td></tr>
<tr><td>0x30</td><td>Most of Europe</td></tr>
<tr><td>0x31</td><td>Spain</td></tr>
<tr><td>0x32</td><td>France</td></tr>
<tr><td>0x40</td><td>Japan</td></tr>
</table>
</td></tr>
</table>

5.18 802.11h

The APIs in this section support the 802.11h protocol. The 802.11h protocol is used for radar detection for 802.11a channels in Europe. 802.11h support in the firmware is controlled by the SNMP command with OID 10. Writing a value of 0x00 disables 802.11h and writing 0x01 enables 802.11h.

Table 34 lists the supported 802.11h commands.

Table 34: 802.11h Commands

Command	Description	Page
Section 5.18.1 "DFS"		
5.18.1.1 "CMD_802_11H_MEASUREMENT_REQUEST"	Sends measurement request	page 106
5.18.1.2 "CMD_802_11H_GET_MEASUREMENT_REPORT"	Gets measurement response report frame	page 107
5.18.1.3 "CMD_802_11H_CHAN_SW_ANN"	Broadcasts a channel switch announcement	page 108
Section 5.18.2 "TPC"		
5.18.2.1 "CMD_802_11H_TPC_INFO"	Gets TPC information	page 109
5.18.2.2 "CMD_802_11H_TPC_ADAPT_REQ"	Requests TPC report	page 110

5.18.1 DFS

The APIs in this section support the 802.11h Dynamic Frequency Selection (DFS) protocol extensions.

5.18.1.1 CMD_802_11H_MEASUREMENT_REQUEST

Using **CMD_802_11H_MEASUREMENT_REQUEST** the driver can send a measurement request to firmware. This measurement request also indicates the MAC address of the station that needs to make the measurement. If MAC address is that of the current station, the firmware makes the measurement and replies with a measurement response. If it is intended for another station, the firmware sends the measurement request to the intended station and sends an event to the driver when it gets a response.

REQUEST

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11H_MEASUREMENT_REQUEST**
Size	UINT16	Number of bytes in command body
SeqNum	UNIT16	Command sequence number
Result	UINT16	Set to CMD_STATUS_ERROR if measurement request cannot be completed
Peer Station Address	UINT8[6]	MAC address of the peer entity to which the request is sent
Dialog Token	UINT8	Same as the IEEE measurement request action frame dialog token
Measurement Request Set	Variable	Measurement request set

The measurement request set comprises the following data structure:

Field Name	Type	Description
Request Mode	UINT8	Request mode
Type	UINT8	Measurement type
Measurement Request	Variable	Measurement request: • Basic • CCA • RPI

5.18.1.2 CMD_802_11H_GET_MEASUREMENT_REPORT

RESPONSE

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11H_GET_MEASUREMENT_REPORT**
Size	UINT16	Number of bytes in command body
SeqNum	UINT16	Command sequence number
Result	UINT16	Not used (set to 0)
Peer Station Address	UINT8[6]	MAC Address of the Peer Entity

The sequence of messages between driver and firmware is:

1. The firmware sends a CBP_EV_MEASUREMENT_RESPONSE_RDY event to the driver.
2. The driver should reply with a **CMD_802_11H_GET_MEASUREMENT_REPORT** command with the Peer Station Address set to zero.
3. The firmware responds by filling this command with the measurement response report frame. The firmware also fills the peer station address with the MAC address of the station that generated the report.

RESPONSE

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11H_GET_MEASUREMENT_REPORT** 0x8000
Size	UINT16	Number of bytes in command body
SeqNum	UINT16	Command sequence number
Result	UINT16	Not used (set to 0)
Peer Station Address	UINT8[6]	MAC Address of the Peer Entity reporting peer station
Dialog Token	UINT8	Same as the IEEE Measurement Report Action Frame Dialog Token
Measurement Report Set	Variable	Measurement Report Set

The measurement report set comprises the following data structure:

Field Name	Type	Description
Report Mode	UINT8	Report Mode
Type	UINT8	Measurement Type
Measurement	Variable	IEEE 802.11h specification report: • Basic report • CCA report • RPI reports

There are four cases when the measurement request and measurement responses are exchanged.

- The driver may send a measurement request to the firmware for a measurement to be done by the firmware. In this case, the firmware responds with a measurement response after the measurement is done.
- The driver may send a measurement request to the firmware for a measurement to be done by the peer station. In this case, the firmware sends the same measurement request to the peer station. When the peer station responds with a measurement report, the firmware sends the same to the driver.
- On detecting radar, if the peer station accepts autonomous reports and if the firmware is not the DFS owner of an IBSS, the firmware sends an autonomous measurement report to the peer station. This report is not acknowledged by the driver. There is no measurement request in this transaction.
- The driver may choose to send a measurement report to a peer station. It does so by sending a measurement report command to the firmware with the peer station address equal to the MAC address of the peer station. There is no measurement request in this transaction.

5.18.1.3 CMD_802_11H_CHAN_SW_ANN

The **CMD_802_11H_CHAN_SW_ANN** command is used to send a channel switch announcement and is for test purposes only. In an IBSS, the firmware manages any necessary channel switch announcements. In infrastructure mode, only the AP can send a channel switch announcement.

REQUEST

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11H_CHAN_SW_ANN** (0x0061)
Size	UINT16	Number of bytes in command body
SeqNum	UINT16	Command sequence number
Result	UINT16	Not used (set to 0)
Ch_Sw_Mode	UINT8	Set to 1 if transmissions are to cease pending the channel switch
ChNum	UINT8	New channel
ChSwCnt	UINT8	Number of TBTTs until the channel switch (as defined by IEEE)

5.18.2 TPC

The APIs in this section support the 802.11h TPC protocol extensions.

5.18.2.1 CMD_802_11H_TPC_INFO

REQUEST

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11H_TPC_INFO** (0x005C)
Size	UINT16	Number of bytes in command body
SeqNum	UNIT16	Command sequence number
Result	UINT16	Not used (set to 0)
Local Power Constraint	**MrvlIETypes_LocalPowerConstraint_t**	802.11h local power constraint information included in Ad-Hoc Beacon/Probe responses and used to set the Tx power used by firmware
Power Capability Element	**MrvlIETypes_LocalPowerCapabiity_t**	802.11h power capability element included in (re)association frames

RESPONSE

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11H_TPC_INFO** | 0x8000
Size	UINT16	Number of bytes in command body
SeqNum	UNIT16	Command sequence number
Result	UINT16	Result code
Local Power Constraint	**MrvlIETypes_LocalPowerConstraint_t**	802.11h local power constraint information included in Ad-Hoc beacon/probe responses
Power Capability Element	**MrvlIETypes_LocalPowerCapabiity_t**	802.11h power capability element to be included in (re)association frames

5.18.2.2 CMD_802_11H_TPC_ADAPT_REQ

REQUEST

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11H_TPC_ADAPT_REQ**
Size	UINT16	Number of bytes in command body
SeqNum	UINT16	Command sequence number
Result	UINT16	Not used (set to 0)
Peer Station Address	UINT8[6]	MAC address of the peer entity to which the request is sent
Timeout	UINT16	Time to wait for a TPC report (ms)
IEEE Rate Index	UINT8	Rate at which the TPC request frame should be sent to peer station:

Rate (Mbps)	IEEE RateIndex
1	2
2	4
5.5	11
11	22
Reserved	Reserved
6	12
9	18
12	24
18	36
24	48
36	72
48	96
54	108

RESPONSE

Field Name	Type	Description
CmdCode	UINT16	**CMD_802_11H_TPC_ADAPT_REQ** | 0x8000
Size	UINT16	Number of bytes in command body
SeqNum	UINT16	Command sequence number
Result	UINT16	Result code
TPC Result Code	UINT8	TPC result code: 0x0 = TPC_SUCCESS 0x1 = TPC_INVALID_PARAMETERS 0x2 = TPC_UNSPECIFIED_FAILURE
Tx Power	INT8	Signed value used to indicate Tx power used by remote station to transmit the Tx report frame (dBm)

Field Name	Type	Description
Link Margin	INT8	Signed value used to indicate link margin reported by remote station (dBm)
RSSI	INT8	Signed value used to indicate RSSI of the received TPC report frame (dBm)

If a valid TPC report frame is received before timeout occurs:

- TPC result code is set to TPC_SUCCESS, and Tx power
- Link margin and RSSI are reported for the TPC report frame

If TPC result code is not set to TPC_SUCCESS:

- Indicates a failure sending the TPC request frame or a timeout waiting for TPC report
- Contents of Tx power, Link Margin, and RSSI are unpredictable

THIS PAGE INTENTIONALLY LEFT BLANK

Section 6. MAC Events

The WLAN SoC device firmware generates events to notify the host driver of certain MAC events that occurred during normal operation. MAC events do not carry data. Only the event type is communicated to the host driver. The mechanism for generating events is dependent upon the hardware interface.

6.1 Events Supported

Table 35 lists the supported events:

Table 35: Event Support

Events	Value	Description
Beacon Lost No Scan	3	Beacon lost with no scan.
Link Sense	4	In the IBSS network, if the total number of joined stations (including self) changes from one station to more than one station, Link Sense is issued.
Deauthenticate	8	Generated when the firmware receives a 802.11 deauthenticate management frame from the AP.
Disassociate	9	Generated when the firmware receives a 802.11 disassociate management frame from the AP.
PS Awake	10	Generated when the firmware first takes the WLAN SoC device out of Sleep mode.
Enter Sleep Mode	11	Generated when the firmware is about to put the WLAN SoC device in Sleep mode.
Group MIC Error	13	MIC error generated for Broadcast packets.
UNICAST MIC Error	14	MIC error generated for Unicast packets.
Deep Sleep Awake	16	Generated to wakeup the card from Deep Sleep mode.
Ad-Hoc Beacon Lost	17	In the IBSS network, if the total number of joined stations (including self) changed from more than one station to only one station, Ad-Hoc Beacon Lost is issued Only implemented in 88W8385 based designs.
Host Awake	18	Generated when the WLAN SoC device would like to wakeup the host.
Stop Transmit	19	Generated to indicate the firmware is not ready to receive data packets. Generation of this event can occur during measurements, quiet periods, or channel switches (802.11h only).
Start Transmit	20	Generated to indicate the firmware is ready to receive data packets (802.11h only).
Channel Switch	21	Generated when the firmware changes the operating channel (802.11h only).

Table 35: Event Support (Continued)

Events	Value	Description
Measurement Report Ready	22	Generated when the firmware has a measurement report waiting for driver retrieval using the Section 5.18.1.2 "CMD_802_11H_GET_MEASUREMENT_REPORT" on page 107 (802.11h only).
WMM Status Change	23	Generated when the firmware monitors a change in the WMM state. This is a result of an AP change in the WMM Parameter IE or a change in the operational state of one of the AC queues (see Section 5.15.1 "CMD_WMM_GET_STATUS" on page 99).
Background Scan Report	24	Generated as a result of a completed background scan report (see Section 5.7.2 "CMD_802_11_BG_SCAN_CONFIG" on page 54).
RSSI Low	25	Subscribed event (see Section 5.12 "Event Subscription" on page 77).
SNR Low	26	Subscribed event (see Section 5.12 "Event Subscription" on page 77).
MAX Failures	27	Subscribed event (see Section 5.12 "Event Subscription" on page 77).
RSSI High	28	Subscribed event (see Section 5.12 "Event Subscription" on page 77).
SNR High	29	Subscribed event (see Section 5.12 "Event Subscription" on page 77).

Section 7. WPA, QoS, and RSSI Support

7.1 WPA Support

WPA is supported by the following:

- WPA related commands
- Tx descriptor for each packet contains bit fields to specify the per packet encryption method

7.1.1 WPA-PSK Setup Procedure

The procedure for setting up WPA-PSK is:

1. Set firmware to Infrastructure mode.
2. Set firmware to open authentication.
3. Enable RSN in the firmware.
4. Send Association command to firmware.
5. Supplicant sends PWK to driver and the driver forwards the PWK to the firmware.
6. Supplicant sends GWK to driver and the driver forwards the GWK to firmware.

See Section 5.9 "Security" on page 65 for additional information.

7.2 QoS Support

Quality of Service (QoS) support is based on the WMM subset of the 802.11e specification. WMM features are supported by the following:

- WMM related commands (see Section 5.15 "WMM" on page 98)
- Per packet control provided by the Tx Descriptors (see Section 3. "Data Path" on page 21)

Not implemented in 88W8381 based designs.

7.3 RSSI and Noise Floor Support

The firmware supports the reporting of RSSI and Noise Floor values:

- RSSI can be derived by adding Noise Floor to SNR value
- Noise Floor is returned in each received packet through the NF field in the Rx Descriptor (dBm).
- Signal Quality is determined by RSSI and SNR.

See Section 5.5 "Status Information" on page 47 for additional information.

THIS PAGE INTENTIONALLY LEFT BLANK

Section 8. TLV Usage

This section describes the functions of the TLV in API commands.

8.1 TLV Format

All arrays (with exception of MAC Address) in the APIs follow the Marvell Information Element Parameter Set as defined in Table 36 and Table 37.

Table 36 shows the structure of a standard IEEE IE type. Marvell IE types (see Table 37) that are translations of IEEE types are exact duplicates of the IEEE type (except for the 16-bit extension of the Type and Length fields).

Table 36: IEEE IE Type Format

Field Name	Type	Description
Type	UINT8	Type ID
Length	UINT8	Length of payload
Payload	UINT8[]	Data

Table 37: MrvlIEType Format

Field Name	Type	Description
Type	UINT16	Type ID
Length	UINT16	Length of payload
Payload	UINT8[]	Data

Marvell IE Parameter Set is defined with a Type member of UINT16 because the lower byte value is reserved for IEEE 802.11 standard IE type definitions. The reserved value greater than 0xFF is for Marvell Specific IE types.

Marvell IE Parameter Set is defined with a Length member of UINT16 (no payload limit at 255 bytes of data) because there is no need for a large payload.

8.2 TLV Defined Types

Some TLVs in commands are required only for special extensions to the API. Cases where the functionality of the TLV is not required, the TLV can be omitted from the command. TLVs are not required to be in a specific order in the command structure. TLVs are always contiguous in memory and do not include pad bytes unless expressly indicated in the TLV structure itself. In APIs where non-TLV arguments are specified, TLVs are appended directly after the fixed field arguments without any pad bytes.

Table 38: IEEE 802.11 Standard IE Translated to Marvell IE

TLV Type (UINT16)	Description
0x0000	MrvlIETypes_SsIdParamSet_t
0x0001	MrvlIETypes_RatesParamSet_t
0x0002	MrvlIETypes_PhyParamFHSet_t (Frequency Hopping not supported)
0x0003	MrvlIETypes_PhyParamDSSet_t
0x0004	MrvlIETypes_CfParamSet_t
0x0006	MrvlIETypes_IbssParamSet_t
0x0007	**MrvlIETypes_DomainParam _t**
0x0020	**MrvlIETypes_LocalPowerConstraint_t**
0x0021	**MrvlIETypes_LocalPowerCapabiity_t**
0x0024	MrvlIETypes_SupportedChannels_t
0x0028	MrvlIETypes_Quiet_t
0x0029	MrvlIETypes_Ibss_Dfs_t
0x0030	MrvlIETypes_RsnParamSet_t
0x00DD	MrvlIETypes_VendorParamSet_t

Table 39: Marvell Proprietary IE

TLV Type (UINT16)	Description
0x0100	**MrvlIETypes_KeyParamSet_t**
0x0101	**MrvlIETypes_ChanListParamSet_t**
0x0102	**MrvlIETypes_NumProbes_t**
0x0103	Reserved
0x0104	**MrvlIETypes_LowRssiThreshold_t**
0x0105	**MrvlIETypes_LowSnrThreshold_t**
0x0106	**MrvlIETypes_FailureCount_t**
0x0107	**MrvlIETypes_BeaconsMissed_t**
0x0108	**MrvlIETypes_LedGpio_t**
0x0109	**MrvlIETypes_LedBehavior_t**
0x010A	**MrvlIETypes_Passthrough_t**
0x010B to 0x010D	Reserved

Table 39: Marvell Proprietary IE (Continued)

TLV Type (UINT16)	Description
0x010E	**MrvlIETypes_BcastProbe_t**
0x010F	**MrvlIETypes_NumSSIDProbe_t**
0x0110	**MrvlIETypes_WmmQStatus_t**
0x0112	Reserved
0x0113	**MrvlIEtypes_TsfTimestamp_t**
0x0114	Reserved
0x0115	**MrvlIETypes_HostSleepFilterType1**
0x0116 to 0x0117	Reserved
0x0118	**MrvlIETypes_HighRssiThreshold_t**
0x0119	**MrvlIETypes_HighSnrThreshold_t**

8.3 Marvell Extended IEEE IE Formats

Marvell Extended IEEE IEs maintain the same data and structure as the standard IEEE IEs, except for the 16-bit Type and Length fields. The additional upper byte in these fields is always set to 0x00 for IEEE replacement IEs.

This section displays examples of the IEEE IEs in cases where non-intuitive functionality in the firmware may be triggered from the IE parameters. Refer to the IEEE documentation for field layout of types not explicitly defined here.

Table 40: Marvell Extended IEEE IE Formats

IE Format	Description	Page
8.3.1 "MrvlIETypes_DomainParam _t"	802.11 domain parameters TLV	page 120
8.3.2 "MrvlIETypes_LocalPowerConstraint_t"	802.11h local power constraint information	page 120
8.3.3 "MrvlIETypes_LocalPowerCapabiity_t"	802.11h power capability element	page 121

8.3.1 MrvlIETypes_DomainParam _t

The **MrvlIETypes_DomainParam _t** TLV has the following format:

Field Name	Type	Description
Type	UINT16	Type ID = 0x0007
Length	UINT16	Length of payload
CountryCode	UINT8[3]	Country string
FirstChannel	UINT8	First channel in a sub-band
NumChannels	UINT8	Number of channels in sub-band
MaxTxPower	UINT8	Power for each channel (dBm)
...	...	...
FirstChannel	UINT8	First channel in a sub-band
NumChannels	UINT8	Number of channels in sub-band
MaxTxPower	UINT8	Power for each channel (dBm)

Firmware does not enable channel validity checks on the 802.11d domain information provided in Set Domain command. The driver is responsible for ensuring the channels provided in the Domain Information command are valid.

Firmware always assumes the channels specified in a sub-band triplet are adjacent channels (5 MHz apart). If the AP specifies a different channel spacing in the Country IE, the driver must convert the channel representation in the Country IE to a default 5 MHz spacing, expected by firmware, before issuing the Set Domain command to firmware.

For instance, if the AP’s Country IE contains a channel triplet (36, 4, 20) in 5 GHz band, firmware interprets this as channels 36, 37, 38 and 39 with 20 dBm max power even though channels 37, 38, and 39 does not exist. To correct this problem, the driver must convert this to four triplets:

- 36, 1, 20
- 40, 1, 20
- 44, 1, 20
- 48, 1, 20

8.3.2 MrvlIETypes_LocalPowerConstraint_t

The **MrvlIETypes_LocalPowerConstraint_t** TLV has the following format:

Field Name	Type	Description
Type	UINT16	Type ID = 0x0020
Length	UINT16	Length of TLV excluding Type and Length fields
Channel	UINT8	Channel number for which the local constraint is imposed
Power Constraint	UINT8	Local power constraint (dBm) Power constraint value received in beacon/probe response frames or a user specified/hard coded value when starting a BSS in Ad-Hoc mode.

8.3.3 MrvlIETypes_LocalPowerCapabiity_t

The **MrvlIETypes_LocalPowerCapabiity_t** TLV has the following format:

Field Name	**Type**	**Description**
Type	UINT16	Type ID = 0x0021
Length	UINT16	Length of TLV excluding Type and Length fields
Minimum Power	INT8	Signed value indicating minimum power the station is capable of transmitting (dBm)
Maximum Power	INT8	Signed value indicating maximum power of the station (dBm) If the driver specifies a value that exceeds the regulatory domain maximum minus any local channel constraint, then the firmware limits the power. If the value specified is less than the regulatory maximum minus any local channel constraint, the value specified by driver is used as is.

A zero length local power constraint (or power capability) element removes any previously set local power constraint (or power capability) in firmware.

8.4 Marvell Proprietary IE Formats

This section lists the format for various Marvell proprietary IE used in firmware API commands.

Table 41: Marvell Proprietary IE Formats

IE Format	Description	Page
8.4.1 "MrvlIETypes_KeyParamSet_t"	Sets key parameter	page 123
8.4.2 "MrvlIETypes_ChanListParamSet_t"	Used in scan and background scan requests	page 125
8.4.3 "MrvlIETypes_NumProbes_t"	Number of times each transmitted probe request frame is replicated	page 125
8.4.4 "MrvlIETypes_LowRssiThreshold_t"	Value of low RSSI threshold value	page 126
8.4.5 "MrvlIETypes_LowSnrThreshold_t"	Low SNR threshold value	page 126
8.4.6 "MrvlIETypes_FailureCount_t"	Consecutive failure count threshold	page 127
8.4.7 "MrvlIETypes_BeaconsMissed_t"	Number of consecutive missing beacons	page 127
8.4.8 "MrvlIETypes_LedGpio_t"	Maps the GPIO pin used for each LED	page 128
8.4.9 "MrvlIETypes_LedBehavior_t"	Status of LED behavior	page 128
8.4.10 "MrvlIETypes_Passthrough_t"	Adds IEEE IEs to the command	page 129
8.4.11 "MrvlIETypes_BcastProbe_t"	Determines if a probe request frame should be generated	page 129
8.4.12 "MrvlIETypes_NumSSIDProbe_t"	Number of SSIDs for which directed probes need to be generated	page 129
8.4.13 "MrvlIETypes_WmmQStatus_t"	Queue status TLV	page 130
8.4.14 "MrvlIEtypes_TsfTimestamp_t"	TSF timestamps	page 130
8.4.15 "MrvlIETypes_HostSleepFilterType1"	Specifies EthType packets that can wake up the host	page 131
8.4.16 "MrvlIETypes_HighRssiThreshold_t"	Value of high RSSI threshold value	page 132
8.4.17 "MrvlIETypes_HighSnrThreshold_t"	High SNR threshold value	page 132

8.4.1 MrvlIETypes_KeyParamSet_t

The **MrvlIETypes_KeyParamSet_t** TLV has the following format:

Field Name	Type	Description
Type	UINT16	Type ID = 0x0100
Length	UINT16	Length of payload
KeyTypeId	UINT16	Type of key: 0x0 = WEP 0x1 = TKIP 0x2 = AES
KeyInfo	UINT16	Key control info specific to a KeyTypeId
KeyLen	UINT16	Length of key
Key	UINT8[]	Key material of size KeyLen

8.4.1.1 TKIP Key Type

Bit definition of KeyInfo for the TKIP key type material is:

Field Name	Type	Description
Reserved	Bit3 - Bit15	Reserved
isKeyEnabled	Bit2	Key enabled and valid for use
isUnicastKey	Bit1	Key used as the unicast key
isMulticastKey	Bit0	Key used as the multicast key

If it is required that the KeyInfo be updated for options such as enabling/disabling the key, the command can be sent down with only KeyInfo and the key length set to zero. This way only the KeyInfo is updated and not the key value.

If the connection with a peer that made use of this key is lost during the course of operation, the key is automatically disabled by the firmware. Firmware notifies the driver of connection lost via a link lost event. Upon receiving the link lost event, the driver assumes that all key(s) currently being used are disabled.

The key materials in TKIP consist of 32 octets in length. It contains one TKIP key of 16 octets and two MIC keys of 8 octets each.

Table 42: TKIP Type Key Material Definition

Field Name	Type	Description
TkipKey	UNIT8[16]	TKIP encryption/decryption key
TkipTxMicKey	UNIT8[8]	TKIP transmission MIC key
TkipRxMicKey	UNIT8[8]	TKIP receive MIC key

8.4.1.2 AES Key Type

Bit definition of KeyInfo for the AES key type material is:

Field Name	Type	Description
Reserved	Bit3–Bit15	Reserved
IsKeyEnabled	Bit2	Key enable and valid for use
IsUnicastKey	Bit1	Key used as the unicast key
IsMulticastKey	Bit0	Key used as the multicast key

If it is required that the KeyInfo be updated for options such as enabling/disabling the key, the command can be sent down with only KeyInfo and the key length set to zero. This way only the KeyInfo is updated and not the key value.

If the connection with a peer that made use of this key is lost during the course of operation, the key is automatically disabled by firmware. Firmware notifies the driver of connection lost via a link lost event. Upon receiving the link lost event, the driver assumes that all key(s) currently being used are disabled.

The key materials in AES consisted of 16 octets in length.

Table 43: AES Type Key Material Definition

Field Name	Type	Description
AesKey	UNIT8[16]	AES encryption/decryption key

8.4.2 MrvlIETypes_ChanListParamSet_t

The **MrvlIETypes_ChanListParamSet_t** TLV is used in scan and background scan requests and has the following format:

<table>
<tr><th>Field Name</th><th>Type</th><th>Description</th></tr>
<tr><td>Type</td><td>UINT16</td><td>Type ID = 0x0101</td></tr>
<tr><td>Length</td><td>UINT16</td><td>Length of payload</td></tr>
<tr><td colspan="3">Multiple instances of the following fields, one set for each channel to be scanned:</td></tr>
<tr><td>RadioType</td><td>UINT8</td><td>Type of radio:
0x00 = RADIO_BG
0x01 = RADIO_A</td></tr>
<tr><td>ChanNumber</td><td>UINT8</td><td>Channel number for the specified band</td></tr>
<tr><td>ScanType</td><td>ScanType_t (1 octet)</td><td>Scan Type bit map octet
Where ScanType_t is defined as:
<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>7</td><td>Reserved (set to 0)</td></tr>
<tr><td>6</td><td>Reserved (set to 0)</td></tr>
<tr><td>5</td><td>Reserved (set to 0)</td></tr>
<tr><td>4</td><td>Reserved (set to 0)</td></tr>
<tr><td>3</td><td>Reserved (set to 0)</td></tr>
<tr><td>2</td><td>Reserved (set to 0)</td></tr>
<tr><td>1</td><td>Disable channel filter
Turns off the filtering of scan responses from adjacent channels.</td></tr>
<tr><td>0</td><td>Passive scan
Disables the use of probe requests.</td></tr>
</table></td></tr>
<tr><td>MinScanTime</td><td>UINT16</td><td>Minimum scan time (ms)</td></tr>
<tr><td>MaxScanTime</td><td>UINT16</td><td>Maximum scan time (ms)</td></tr>
</table>

8.4.3 MrvlIETypes_NumProbes_t

The **MrvlIETypes_NumProbes_t** TLV has the following format:

Field Name	Type	Description
Type	UINT16	Type ID = 0x0102
Length	UINT16	Length of payload (always 2 bytes)
NumProbes	UINT16	Number of probes to be sent

This indicates the number of times each transmitted probe request frame is replicated. The valid range is [1, 2]. The default value is 1.

8.4.4 MrvlIETypes_LowRssiThreshold_t

The **MrvlIETypes_LowRssiThreshold_t** TLV has the following format.

Field Name	Type	Description
Type	UINT16	Type Id = 0x0104
Length	UINT16	Length of payload (always 2 bytes)
Value	UINT8	Absolute value of RSSI threshold value (in dBm)
Frequency	UINT8	Reporting frequency

Value field specifies the absolute value of the RSSI threshold, the actual value of RSSI threshold is always negative. The RSSI_LOW event is triggered when the average RSSI in received beacons falls below the actual value.

Frequency field specifies the reporting frequency for this event. If it is set to 0, then the event is reported only once, and then automatically unsubscribed. If it is set to 1, then the event is reported every time it occurs. If it is set to N (N>1), an event is generated only when the condition happens N consecutive times.

8.4.5 MrvlIETypes_LowSnrThreshold_t

The **MrvlIETypes_LowSnrThreshold_t** TLV has the following format.

Field Name	Type	Description
Type	UINT16	Type Id = 0x0105
Length	UINT16	Length of payload (always 2 bytes)
Value	UINT8	SNR threshold value (in dB)
Frequency	UINT8	Reporting frequency

Value field specifies the SNR threshold. The SNR_LOW event is triggered when the average SNR in received beacons falls below this value.

Frequency field specifies the reporting frequency for this event. If it is set to 0, then the event is reported only once, and then automatically unsubscribed. If it is set to 1, then the event is reported every time it occurs. If it is set to N (N>1), an event is generated only when the condition happens N consecutive times.

8.4.6 MrvlIETypes_FailureCount_t

The **MrvlIETypes_FailureCount_t** TLV has the following format:

Field Name	Type	Description
Type	UINT16	Type ID = 0x0106
Length	UINT16	Length of payload (always 2 bytes)
Value	UINT8	Failure count threshold
Frequency	UINT8	Reporting frequency

Value field specifies the consecutive failure count threshold that triggers the generation of the MAX_FAIL event. Once this event is generated, the consecutive failure count is reset to 0.

Frequency field specifies the reporting frequency for this event. If it is set to 0, then the event is reported only once, and then automatically unsubscribed. If it is set to 1, then the event is reported every time it occurs. If it is set to N, then the event is reported every Nth time it occurs. At initialization, the MAX_FAIL event is not subscribed by default.

8.4.7 MrvlIETypes_BeaconsMissed_t

The **MrvlIETypes_BeaconsMissed_t** TLV has the following format:

Field Name	Type	Description
Type	UINT16	Type ID = 0x0107
Length	UINT16	Length of payload (always 2 bytes)
Value	UINT8	Number of missed beacons
Reserved	UINT8	Reserved

Value field specifies the number of consecutive missing beacons which triggers the LINK_LOSS event. This event is generated only once after which the firmware resets its state.

At initialization, the LINK_LOSS event is subscribed by default. The default value of MissedBeacons is 60.

At initialization, the RSSI_LOW, SNR_LOW and MAX_FAIL events are NOT subscribed by default.

8.4.8 MrvlIETypes_LedGpio_t

The fixed fields are followed by an optional TLV field to map the GPIO pin used for each LED. The **MrvlIETypes_LedGpio_t** TLV has the following format:

Field Name	Type	Description
Type	UINT16	Type ID = 0x0108
Length	UINT16	Length of payload (equals twice the number of LEDs configured)
LedNumber	UINT8	LED number to be mapped to GPIO number below
GpioNumber	UINT8	GPIO pin number used to control LED number above
…		
LedNumber	UINT8	LED number to be mapped to GPIO number below
LedNGpio	UINT8	GPIO pin number used to control LED number above

The payload is variable length (depending on the number of LEDs the host wishes to configure). The configuration of LEDs that are not included in this TLV is left unchanged.

GpioNumber field indicates the GPIO pin number (used to control LED indicated by the LedNumber field). The valid range for this field is [0,16]. If it is set to 16, the LED is disabled completely. By default, only LED 1 is enabled in the firmware, and GPIO 1 is used to control it.

The host must ensure that any GPIO pin it wants to use in the firmware to control LEDs is available.

Note

GPIO[6] is used for reference clock control. GPIO[0] is configured as an input to the SoC and is used to wakeup the SoC.

8.4.9 MrvlIETypes_LedBehavior_t

The **MrvlIETypes_LedBehavior_t** TLV has the following format:

Field Name	Type	Description
Type	UINT16	Type ID = 0x0109
Length	UINT16	Length of payload (4 bytes)
FirmwareState	UINT8	Firmware state
LedNumber	UINT8	LED number
LedState	UINT8	LED state corresponding to the firmware state
LedArgs	UINT8	Arguments for LED state

Use of the LedBehavior TLV is not supported on 88W8381 or 88W8385 based designs.

Note

This command is implemented in firmware v6.0 or later.

8.4.10 MrvlIETypes_Passthrough_t

The **MrvlIETypes_Passthrough_t** TLV has the following format:

Field Name	Type	Description
Type	UINT16	Type ID = 0x010A
Length	UINT16	Length of payload
Data	UINT8[n]	IEEE IE(s) to be added to the management frame generated by the command

The data field of the pass through TLV is added to the management frame generated. If multiple IEEE IEs are in the Data field, they are parsed and inserted in the correct order. Multiple instances of the pass through TLV are accepted in an API.

8.4.11 MrvlIETypes_BcastProbe_t

The **MrvlIETypes_BcastProbe_t** TLV has the following format:

Field Name	Type	Description
Type	UINT16	Type ID = 0x010E
Length	UINT16	Length of payload (always 2 bytes)
Value	UINT16	Whether a broadcast SSID probe should be generated or not

This indicates whether a probe request frame containing a broadcast SSID should be generated on each channel for which active scan is requested.

Valid range for this field is [0,1]. If the value is 1, then a broadcast SSID probe is generated. If the value is 0, then the broadcast SSID probe is not generated. The default value is 0.

8.4.12 MrvlIETypes_NumSSIDProbe_t

The **MrvlIETypes_NumSSIDProbe_t** TLV has the following format:

Field Name	Type	Description
Type	UINT16	Type ID = 0x010F
Length	UINT16	Length of payload (always 2 bytes)
Value	UINT16	Number of SSIDs for which directed probes need to be generated

This indicates the number of SSIDs for which the directed probes request frames need to be transmitted on each channel for which active scan is requested. If the value is N, then probe requests are generated corresponding to the first N SSIDs in the list of SSIDs provided by the driver. This value must not be greater than the number of SSIDs provided in the command.

Valid range in this field is [0, 2]. The default value is 1.

8.4.13 MrvlIETypes_WmmQStatus_t

The **MrvlIETypes_WmmQStatus_t** has the following format:

Field Name	Type	Description
Type	UINT16	Type ID = 0x0110
Length	UINT16	Length of payload
QueueIndex	UINT8	AC queue by priority: 0 = AC_BK 1 = AC_BE 2 = AC_VI 3 = AC_VO
Disabled	UINT8	Set if data traffic is allowed on this AC Queue Flows are disabled when admission control (FlowRequired) is set by the AP and TSPEC negotiation (ADDTS messaging) has not yet been completed.
TriggeredPS	UINT8	Not currently supported
FlowDirection	UINT8	Direction of the flow setup on this AC Only bi-directional is currently supported: 3 = Bi-Directional
FlowRequired	UINT8	Set if admission control is required for this AC
FlowCreated	UINT8	Set if a TSPEC has been admitted by the AP for this AC via Admission Control messaging
MediumTime	UINT32	Not currently supported

8.4.14 MrvlIEtypes_TsfTimestamp_t

The **MrvlIEtypes_TsfTimestamp_t** has the following format:

Field Name	Type	Description
Type	UINT16	Type ID = 0x0113
Len	UINT16	Length of payload
tsfTable	UINT64[]	TSF timestamps

8.4.15 MrvlIETypes_HostSleepFilterType1

MrvlIETypes_HostSleepFilterType1 TLV has the following format:

Field Name	Type	Description
Type	UINT16	Type Id = 0x0115
Length	UINT16	Length of Payload
Payload	UINT8[]	Payload

Payload contains 0 or more fixed size entries of EthTypeEntry. Each EthTypeEntry has the following format:

Field Name	Type	Description
AddrType	UINT16	Type of MAC address: 1 = Bcast 2 = Unicast 3 = Multicast
EthType	UINT16	Ethernet type field in 802.2 header Example: 0x0806 = ARP
Ipv4Addr	UINT32	IP v4 address specific to EthType

AddrType specifies the type of MAC address. Each value of AddrType requires the corresponding Bcast/Unicast/Multicast bit in Criteria to be set. This field is not used if set to 0xFFFF.

EthType specifies the IEEE Ethernet type. For example, an EthType entry for matching ARP request has EthType 0x0806. For faster matching, EthType is in network byte order, i.e. big endian. For the example of ARP, the lowest address byte of EthType is the most significant byte 0x08, and the higher address byte is 0x06.

Ipv4Addr is the IP v4 address specific to EthType. For example, ARP request matching specifies the IP v4 address of the host. This field is not used if set to 0xFFFFFFFF. For faster matching, Ipv4Addr is in network byte order. For the example of 192.168.0.2 (0xC0A80002), the lowest address byte of Ipv4Addr is 0xC0, followed by 0x80, 0x00, and 0x02 (the highest address byte).

Each EthTypeEntry specifies an *allow* filter, (matching packet wakes up the host). The AddrType (if not 0xFFFF), EthType, and Ipv4Addr (if not 0xFFFFFFFF) fields are AND'd together to form the filter. Filters are processed in the storage order and in an OR relationship.

As an example, if the host wishes to wake up on any multicast packet or an ARP packet (a broadcast packet) intended for the host, it should set bit 3 (multicast) and bit 0 (broadcast) in Criteria and include the following **MrvlIETypes_HostSleepFilterType1** TLV:

Field Name	Type	Description
Type	UINT16	Type Id (**MrvlIETypes_HostSleepFilterType1**)
Length	UINT16	16
AddrType	UINT16	3 (multicast)
EthType	UINT16	0x ????
Ipv4Addr	UINT32	0xFFFFFFFF
AddrType	UINT16	1 (broadcast)

Field Name	Type	Description
EthType	UINT16	0x0806
Ipv4Addr	UINT32	192.168.0.88

Note

The **MrvlIETypes_HostSleepFilterType1** TLV is only supported in specific versions of firmware v5.0 for 88W8385 and 88W8399 devices.

8.4.16 MrvlIETypes_HighRssiThreshold_t

The **MrvlIETypes_HighRssiThreshold_t** TLV has the following format.

Field Name	Type	Description
Type	UINT16	Type Id = 0x0118
Length	UINT16	Length of payload (always 2 bytes)
Value	UINT8	Absolute value of RSSI threshold value (in dBm)
Frequency	UINT8	Reporting frequency

Value field specifies the absolute value of the RSSI threshold, the actual value of RSSI threshold is always negative. The RSSI_HIGH event is triggered when the average RSSI in received beacons goes above the actual value.

Frequency field specifies the reporting frequency for this event. If it is set to 0, then the event is reported only once, and then automatically unsubscribed. If it is set to 1, then the event is reported every time it occurs. If it is set to N (N>1), an event is generated only when the condition happens N consecutive times.

8.4.17 MrvlIETypes_HighSnrThreshold_t

The **MrvlIETypes_HighSnrThreshold_t** TLV has the following format.

Field Name	Type	Description
Type	UINT16	Type Id = 0x0119
Length	UINT16	Length of payload (always 2 bytes)
Value	UINT8	SNR threshold value (in dB)
Frequency	UINT8	Reporting frequency

Value field specifies the SNR threshold. The SNR_HIGH event is triggered when the average SNR in received beacons goes above this value.

Frequency field specifies the reporting frequency for this event. If it is set to 0, then the event is reported only once, and then automatically unsubscribed. If it is set to 1, then the event is reported every time it occurs. If it is set to N (N>1), an event is generated only when the condition happens N consecutive times.

Section 9. Delta Between Versions

Major differences between the v5.1 specification and the v5.0 specification include:

- Changed functionality of **CMD_802_11_SCAN** command
- Changed **CMD_802_11_ASSOCIATE** to be TLV based
- Added get beacon RSSI feature in Ad-Hoc mode
- Added **MrvlIEtypes_TsfTimestamp_t** information element
- New helper image used
  The new helper image maps to a higher memory address.
- Implemented new parameter to control PS mode NULL packet generation in **CMD_802_11_PS_MODE**
- Removed unsupported APIs:
  - Non-TLV based SCAN
  - Non-TLV based CMD_802_11_ASSOCIATE_EXT
  - HostCmd_CMD_802_11_CAL_DATA—**CMD_802_11_CAL_DATA_EXT** is used instead
  - HostCmd_CMD_802_11_QUERY_STATUS
- Additional TLVs:
  - **MrvlIETypes_HostSleepFilterType1**
  - **MrvlIETypes_WmmQStatus_t**
  - **MrvlIETypes_LowRssiThreshold_t**
  - **MrvlIETypes_LowSnrThreshold_t**
  - **MrvlIETypes_HighRssiThreshold_t**
  - **MrvlIETypes_HighSnrThreshold_t**

Major differences between the v5.0 specification and the v4.0 specification include:

- CMD_802_11_RATE_ADAPT_RATESET command
- CMD_802_11_TPC_CFG command
- CMD_802_11_PA_CFG command
- CMD_802_11_SUBSCRIBE_EVENT command
- CMD_802_11_LED_CONTROL command
- CMD_802_11_PS_MODE command
- CMD_802_11_HWM_CONFIG_DONE command
- CMD_802_11_FW_WAKE_METHOD command
- 802.11h support
- TLV Usage section

Major differences between the v4.0 specification and the v3.3 specification include:

- Background Scan support
- 802.11d support
- WMM support
- WMM status change related commands and indications
- USB interface APIs
- CMD_802_11_KEY_MATERIAL command
- TKIP Key Type

- AES Key Type
- CMD_802_11_INACTIVITY_TO command
- CMD_802_11_SLEEP_PERIOD command

Major differences between the v3.3 specification and the v3.1 specification include:

- 802.11a enhancements
- EAP support
- G-SPI host interface support
- CMD_802_11_ASSOCIATE_EXT command
- CMD_EEPROM_ACCESS command
- CMD_GSPI_BUS_CONFIG command
- CMD_802_11_CAL_DATA_EXT command

Major differences between the v3.1 specification and the v3.0 specification include:

- Power Management operation
- Driver/Firmware interaction
- Deep Sleep
- Calibration Data
- RFI Tx and Rx
- Associate Extended
- Stop Ad-Hoc network
- WPA Multicast Cipher
- Pre-TBTT
- Bluetooth Coexistence
- MLME SAP Interface Data Structure

Major differences between the v3.0 specification and the v2.0 specification include:

- Updated command and response for Scan and Association commands.
- Added multiple DTIM for power save.
- Added Link Loss event indication to host.

Major differences between the v2.0 specification and the v1.0 specification include:

- The Receive Packet Descriptor has been changed. The size stays the same, but the fields in the first 4 bytes have been re-organized.
- The Transmit Packet Descriptor has been changed. The TxControl field has been changed. It contain more fields to support WPA and QoS features.
- WPA feature support
- QoS/WME feature support
- The command to get the hardware specification was changed: add a field to return the hardware interface version number.

Appendix A. Command List

Table 44: Command List

Symbolic Name	Value	Description
Reset and Initialization		
CMD_802_11_RESET	0x0005	Resets the WLAN device
CMD_802_11_SNMP_MIB	0x0016	Sets/gets the SNMP MIB
CMD_802_11_MAC_ADDR	0x004D	WLAN MAC address
CMD_MAC_MULTICAST_ADR	0x0010	Sets/gets MAC multicast filter table
CMD_GSPI_BUS_CONFIG	0x005A	Sets/gets the G-SPI Bus mode and time delay between host address write and data read
MAC/PHY/RF Control		
CMD_MAC_CONTROL	0x0028	Controls hardware MAC
CMD_802_11_RADIO_CONTROL	0x001C	Controls the radio chip
CMD_802_11_RF_ANTENNA	0x0020	Sets/gets the Tx and Rx antenna mode
CMD_802_11_RF_TX_POWER	0x001E	Sets/gets radio transmit power
CMD_802_11_RF_CHANNEL	0x001D	Sets/gets RF channel
Register and Memory Access		
CMD_BBP_REG_ACCESS	0x001A	Peeks and pokes baseband processor hardware register
CMD_RF_REG_ACCESS	0x001B	Peeks and pokes RF hardware register
CMD_MAC_REG_ACCESS	0x0019	Peeks and pokes MAC hardware register
CMD_EEPROM_ACCESS	0x0059	Retrieves the EEPROM data
RF Calibration Data		
CMD_802_11_CAL_DATA_EXT	0x006D	Sets/gets the RF calibration data
Status Information		
CMD_GET_HW_SPEC	0x0003	Gets hardware specifications
CMD_802_11_GET_LOG	0x000B	Gets the WLAN log
CMD_802_11_RSSI	0x001F	Gets the received radio signal strength
LED Control		
CMD_802_11_LED_CONTROL	0x004E	LED Control
Scan		
CMD_802_11_SCAN	0x0006	Starts the scan process
CMD_802_11_BG_SCAN_CONFIG	0x006B	Sets/gets background scan configuration
CMD_802_11_BG_SCAN_QUERY	0x006C	Gets background scan results

Table 44: Command List (Continued)

Symbolic Name	Value	Description
Network Start/Stop/Join		
CMD_802_11_ASSOCIATE	0x0050	Initiate an association with the AP
CMD_802_11_AD_HOC_START	0x002B	Starts an Ad-Hoc network
CMD_802_11_AD_HOC_JOIN	0x002C	Join an Ad-Hoc network
CMD_802_11_AD_HOC_STOP	0x0040	Stops Ad-Hoc Network
Security		
CMD_802_11_AUTHENTICATE	0x0011	Starts authentication process with the AP
CMD_802_11_DEAUTHENTICATE	0x0024	Starts de-authentication process with the AP
CMD_802_11_SET_WEP	0x0013	Configures the WEP keys
CMD_802_11_ENABLE_RSN	0x002F	Sets/gets RSN enable state
CMD_802_11_KEY_MATERIAL	0x005E	Sets/gets key material used to do Tx encryption or Rx decryption
Rate Adaptation		
CMD_802_11_RATE_ADAPT_RATESET	0x0076	Sets/gets transmit data rate
CMD_TX_RATE_QUERY	0x007F	Reports the current Tx rate
Transmit Power Control		
CMD_802_11_TPC_CFG	0x0072	Controls TPC functionality
CMD_802_11_PA_CFG	0x0073	Configures power adaptation related settings
Event Subscription		
CMD_802_11_SUBSCRIBE_EVENT	0x0075	Subscribe to events and set thresholds
Power Management Commands		
CMD_802_11_PS_MODE	0x0021	Sets/gets PS mode
CMD_802_11_SLEEP_PARAMS	0x0066	Sets/gets sleep parameters
CMD_802_11_HOST_SLEEP_CFG	0x0043	Configures the wakeup semantics of the host driver
CMD_802_11_WAKEUP_CONFIRM	0x0044	Sends a Host Awake event
CMD_802_11_FW_WAKE_METHOD	0x0074	Firmware wake method
CMD_802_11_DEEP_SLEEP	0x003E	Initiates Deep Sleep mode
CMD_802_11_SLEEP_PERIOD	0x0068	Sleep period command
Bluetooth Coexistence		
CMD_802_11_BCA_CONFIG_TIMESHARE	0x0069	Configures the BCA timeshare interval and percentage of time in this timeshare interval
WMM		
CMD_WMM_GET_STATUS	0x0071	Retrieves the current WMM state
CMD_WMM_ACK_POLICY	0x005C	Specifies and retrieves the data packet acknowledgement scheme used for WMM traffic

Table 44: Command List (Continued)

Symbolic Name	Value	Description
802.11a		
CMD_802_11_BAND_CONFIG	0x0058	Sets/gets the RF band settings
802.11d		
CMD_802_11D_DOMAIN_INFO	0x005B	Sets/gets 802.11d domain information
CMD_802_11_RGN_CODE	0x004C	Sets/gets region code stored in the EEPROM
802.11h		
CMD_802_11H_MEASUREMENT_REQUEST	0x0062	Sends measurement request
CMD_802_11H_GET_MEASUREMENT_REPORT	0x0063	Gets measurement response report frame
CMD_802_11H_CHAN_SW_ANN	0x0061	Broadcasts a channel switch announcement
CMD_802_11H_TPC_INFO	0X005F	Gets TPC information
CMD_802_11H_TPC_ADAPT_REQ	0X0060	Requests TPC report

Table 45: Command Result Code

Symbolic Name	Value	Description
CMD_STATUS_SUCCESS	0x0000	No error
CMD_STATUS_ERROR	0x0001	Command failed
CMD_STATUS_UNSUPPORTED	0x0002	Command is not supported
CMD_STATUS_PENDING	0x0003	Command pending
CMD_STATUS_BUSY	0x0004	Previous command is being processed
CMD_STATUS_PARTIAL_DATA	0x0005	

THIS PAGE INTENTIONALLY LEFT BLANK

Appendix B. Revision History

Table 46: Revision History

Document Type	Document Revision
Release	Rev. –
First release.	
Changes from v5.0 document (MV-S800432-00 Rev. –): ***Data Path*** • Section 3.1 "Receive Packet Descriptor" on page 22: removed NextRxpd field, field now reserved • Table 2, “Fields in Receive Packet Descriptor,” on page 22: changed RxStatus, and RxControl field descriptions to reserved; removed NextRxpd field, field now reserved; updated RxRate field description to match Table 6, “Rate Encoding,” on page 25; removed PCI value from RxPacketLocation field description • Table, “Bits in RxControl,”: removed • Section 3.2 "Transmit Packet Descriptor" on page 24: removed PCI from description • Table 3, “Fields in Transmit Packet Descriptor,” on page 24: changed TxStatus, and RxControl field descriptions to reserved; removed PCI value from TxPacketLocation field description • Table, “Bits in TxStatus—PCI Only,”: removed ***Host Commands*** • Table 11, “Reset and Initialization Commands,” on page 29: added • Section 5.1.3 "CMD_802_11_MAC_ADDR" on page 31: updated both request and response Action parameter • Section 5.2.1 "CMD_MAC_CONTROL" on page 36: removed bit[15] WEP Type note • Table 12, “MAC/PHY/RF Control Commands,” on page 35: added • Table 13, “Register and Memory Access Commands,” on page 42: added • Table 14, “RF Calibration Commands,” on page 46: added • Table 15, “Status Information Commands,” on page 47: added • Section 5.5.1 "CMD_GET_HW_SPEC" on page 47: changed NumOfWCB, WcbBase, RxPdRdPtr, and RxPdWrPtr field descriptions to reserved in the response • Table 16, “Scan Commands,” on page 52: added • Section 5.7.1 "CMD_802_11_SCAN" on page 52: changed SsIdParamSet and OpRateSet fields to optional; updated response table to include TSFTable field • Section "Connected and Deep Sleep": removed • Table 18, “Network Start/Stop/Join Commands,” on page 59: added • Section 5.8.1 "CMD_802_11_ASSOCIATE" on page 59: removed Capinfo, ReassociationAp, Status, Aid, InfoLength, and Info fields from the response table • Table 21, “Security Commands,” on page 65: added • Section 5.10.2 "CMD_TX_RATE_QUERY" on page 72: added • Table 27, “Transmit Power Control Commands,” on page 74: added • Table 28, “Event Subscription Commands,” on page 77: added • Section 5.12.1 "CMD_802_11_SUBSCRIBE_EVENT" on page 77: changed default value from 0x0008 to 0x0; added MrvlIETypes_LowRssiThreshold _t TLV, MrvlIETypes_LowSnrThreshold _t TLV, MrvlIETypes_HighRssiThreshold_t TLV, and MrvlIETypes_HighSnrThreshold_t TLV to supported TLV list • Table 29, “Subscribed Events Bitmap Bits,” on page 78: added bits 4 and 5 • Table 30, “Power Management Commands,” on page 79: added • Section 5.13.3.1 "CMD_802_11_PS_MODE" on page 81: added NullPktInterval parameter and description • Section 5.13.3.2 "CMD_802_11_SLEEP_PARAMS" on page 83: update description to include information on reserved parameter and invoking command •	

Table 46: Revision History (Continued)

Document Type	Document Revision
• Section "CMD_802_11_HOST_WAKE_UP_CFG": removed • Section 5.13.3.3 "CMD_802_11_HOST_SLEEP_CFG" on page 85: added • Section 5.13.3.5 "CMD_802_11_FW_WAKE_METHOD" on page 87: clarified description on when command is issued; removed CF interface information • Section 5.13.6 "Assumptions" on page 90: removed CMD_802_11_HOST_SLEEP_CFG command reference • Section 5.13.7 "Deep Sleep Mode" on page 90: removed description on how to initiate signal • Section 5.13.7.3 "Deep Sleep Mode Assumptions" on page 91: clarified operation supported in Idle mode • Section 5.15 "WMM" on page 98: added enable and disable information • Table 32, “WMM Commands,” on page 98: added • Section 5.15.1 "CMD_WMM_GET_STATUS" on page 99: updated command request and response to reflect TLV functionality • Section 5.15.2 "CMD_WMM_ACK_POLICY" on page 100: moved section so it was no longer a sub section of Section 5.15.1 "CMD_WMM_GET_STATUS" on page 99 • Table 33, “802.11d Commands,” on page 103: added • Table 34, “802.11h Commands,” on page 105: added ***MAC Events*** • Table 35, “Event Support,” on page 113: value 23 changed to WMM Status Change; added RSSI High and SNR High events ***TLV Usage*** • Table 40, “Marvell Extended IEEE IE Formats,” on page 119: added • Table 41, “Marvell Proprietary IE Formats,” on page 122: added • Section 8.4.2 "MrvlIETypes_ChanListParamSet_t" on page 125: updated format table • Section 8.4.3 "MrvlIETypes_NumProbes_t" on page 125: removed value field name • Section 8.4.4 "MrvlIETypes_LowRssiThreshold_t" on page 126: added • Section 8.4.5 "MrvlIETypes_LowSnrThreshold_t" on page 126: added • Section "MrvlIETypes_ReassociationAp_t": removed • Section 8.4.7 "MrvlIETypes_BeaconsMissed_t" on page 127: updated MissedBeacon description • Section 8.4.8 "MrvlIETypes_LedGpio_t" on page 128: added note about function of GPIO[0] and GPIO[6] • Section 8.4.13 "MrvlIETypes_WmmQStatus_t" on page 130: added • Section 8.4.14 "MrvlIEtypes_TsfTimestamp_t" on page 130: added • Section 8.4.15 "MrvlIETypes_HostSleepFilterType1" on page 131: added • Section 8.4.16 "MrvlIETypes_HighRssiThreshold_t" on page 132: added • Section 8.4.17 "MrvlIETypes_HighSnrThreshold_t" on page 132: added • Section 8.4.15 "MrvlIETypes_HostSleepFilterType1" on page 131: added	

THIS PAGE INTENTIONALLY LEFT BLANK

MARVELL®

MOVING FORWARD
FASTER®

Marvell Semiconductor, Inc.

5488 Marvell Lane
Santa Clara, CA 95054, USA

Tel: 1.408.222.2500
Fax: 1.408.752.9028

www.marvell.com

Worldwide Corporate Offices

Marvell Semiconductor, Inc.
5488 Marvell Lane
Santa Clara, CA 95054, USA
Tel: 1.408.222.2500
Fax: 1.408.752.9028

Marvell Asia Pte, Ltd.
151 Lorong Chuan, #02-05
New Tech Park, Singapore 556741
Tel: 65.6756.1600
Fax: 65.6756.7600

Marvell Japan K.K.
Shinjuku Center Bldg. 44F
1-25-1, Nishi-Shinjuku, Shinjuku-ku
Tokyo 163-0644, Japan
Tel: 81.3.5324.0355
Fax: 81.3.5324.0354

Marvell Semiconductor Israel, Ltd.
6 Hamada Street
Mordot HaCarmel Industrial Park
Yokneam 20692, Israel
Tel: 972.4.909.1500
Fax: 972.4.909.1501

Marvell Semiconductor Korea, Ltd.
Rm. 1603, Korea Trade Center 159-1
Samsung-Dong, Kangnam-Ku
Seoul, Korea 135-729
Tel: 82.2.551-6070-6079
Fax: 82.2.551.6080

Radlan Computer Communications, Ltd.
Atidim Technological Park, Bldg. #4
P O Box 58179
Tel Aviv 61580, Israel
Tel: 972.3.645.8555
Fax: 972.3.645.8544

Worldwide Sales Offices

Western US

Marvell
5488 Marvell Lane
Santa Clara, CA 95054, USA
Tel: 1.408.222.2500
Fax: 1.408.752.9028
Sales Fax: 1.408.752.9029

Central US

Marvell
9600 North MoPac Drive, Suite #215
Austin, TX 78759, USA
Tel: 1.512.343.0593
Fax: 1.512.340.9970

Eastern US/Canada

Marvell
Parlee Office Park
1 Meeting House Road, Suite 1
Chelmsford, MA 01824 , USA
Tel: 1.978.250.0588
Fax: 1.978.250.0589

Europe

Marvell
c/o Harts CA
3 Churchgates
Church Lane
Berkhamsted
Hertfordshire, HP4 2UB
United Kingdom
Tel: 44.1442.263341
Fax: 44.1442.211543

Israel

Marvell
6 Hamada Street
Mordot HaCarmel Industrial Park
Yokneam 20692, Israel
Tel: 972.4.909.1500
Fax: 972.4.909.1501

China

Marvell
10J, No. 1800, Zhong Shan West Road
Shanghai, PRC 200235
Tel: 86.21.6440.1350
Fax: 86.21.6440.1705

Marvell
Rm. 1102/1103, Jintai Fudi Mansion
#9 An Ning Zhuang West Rd.
Qing He, Haidian District
Beijing, PRC 100085
Tel: 86.10.8274.3831
Fax: 86.10.8274.3830

Japan

Marvell
Shinjuku Center Bldg. 44F
1-25-1, Nishi-Shinjuku, Shinjuku-ku
Tokyo 163-0644, Japan
Tel: 81.3.5324.0355
Fax: 81.3.5324.0354

Taiwan

Marvell
2Fl., No.1, Alley 20, Lane 407, Sec. 2
Ti-Ding Blvd., Nei Hu District
Taipei, Taiwan, 114, R.O.C
Tel: 886.2.8177.7071
Fax: 886.2.8752.5707

Korea

Marvell
Rm. 1603, Korea Trade Center 159-1
Samsung-Dong, Kangnam-Ku
Seoul, Korea 135-729
Tel: 82.2.551-6070-6079
Fax: 82.2.551.6080

**For more information, visit our website at:
www.marvell.com**