# DISTRIBUTED MEDIUM ACCESS CONTROL (MAC) FOR WIRELESS NETWORKS


*Making High-Speed Wireless A Reality ...*

RELEASE 1.01

DECEMBER 15, 2006

NOTICE

The WiMedia Alliance, Inc. (WiMedia) disclaims any and all warranties, whether expressed or implied, including (without limitation) any implied warranties of merchantability or fitness for a particular purpose. WiMedia reserves the right to make changes to the document without further notice.

# DISCLAIMER

This document and any translations are restricted to distribution and use within the WiMedia membership, and any commentary and other documents created that interpret or are used in the implementation of the Final Deliverable may be prepared, copied, published and distributed, in whole or in part, within the WiMedia membership, provided that the above copyright notice and this disclaimer are reprinted in full and conspicuously placed on all such documents. No Member shall alter, revise or otherwise change this document. WiMedia may, on behalf of the Alliance, make changes in order to adapt the content in connection with collaborating with other organizations. The limited permissions granted above are provided to Members only while they are in good standing with WiMedia, and shall cease upon the withdrawal or termination of membership.

This document and the information contained herein is provided on an “AS IS” and on an “ALL FAULTS” basis, and WiMedia DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION ANY WARRANTY THAT THE FINAL DELIVERABLE AND/OR THE USE OR IMPLEMENTATION THEREOF WILL NOT INFRINGE ANY INTELLECTUAL PROPERTY OR OTHER RIGHTS OF ANY THIRD PARTY, OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR TITLE.

Members of WiMedia are under an obligation to disclose any known Essential Patent Claims in the Final Deliverable. Several general and specific disclosures were made in connection with the Final Deliverable, and such general disclosures are available upon request. WiMedia makes no representations on the validity or scope of any Essential Patent Claims or other rights that might be claimed by a Member, nor the extent to which a Member is willing to license any such Essential Patent Claims. WiMedia has not made any separate investigation of any Essential Patent Claims that may be contained in this Final Deliverable.

Licensing obligations relating to Essential Patent Claims contained within this Final Deliverable are governed by the Intellectual Property Rights (IPR) Policy attached to Promoter, Contributor or Adopter Member Agreements.

WiMedia is, and shall at all times be, the sole entity that may authorize the use of certification marks, trademarks, or other special designations to indicate compliance with the Final Deliverable.

## WiMEDIA ALLIANCE MAC COMMITTEE

**Chair:** Kristoffer Fleming, Intel Corporation

**Technical Editors:** Ron Brown, FOCUS Enhancements, Inc.
Jay O'Conor, Philips Semiconductors

## MAC SPECIFICATION COMMITTEE

The following companies and members of the WiMedia MAC Specification committee contributed significant effort to the creation of this specification:

| | |
|---|---|
| Alereon | Martin Gravenstein |
| FOCUS Enhancements | Ron Brown |
| Intel | Kristoffer Fleming |
| NEC Electronics | Nobuyuki Mizukoshi |
| Nokia | Juha Salokannel |
| Philips Semiconductors | Chun-Ting Chou, Jay O'Conor, Javier del Prado Pavon |
| Samsung | Manoj Choudhary, Sunil Jogi |
| ST Microelectronics | Ljubica Blazevic |
| Staccato Communications | Larry Taylor |
| Texas Instruments | Jin-Meng Ho |
| TZero Technologies | Patrick Worfolk |
| WiQuest | Sid Schrum |
| Wisair | Haim Kupershmidt |

## CONTRIBUTORS

| | |
|---|---|
| Simcha Aronson | Infineon Technologies |
| Turi Aytur | Realtek Semiconductor |
| Alan Berkema | Hewlett-Packard |
| Ljubica Blazevic | ST Microelectronics |
| Chuck Brabenac | Intel |
| Ron Brown | FOCUS Enhancements |
| Richard Chen | Philips Semiconductors |
| Chun-Ting Chou | Philips Semiconductors |
| Manoj Choudhary | Samsung |
| Chong Chye Yaw | Philips Semiconductors |
| Joe Decuir | MCCI |
| Randy Erman | Stonestreet One |
| Dwayne Escola | Panasonic |
| Mark Fidler | Hewlett-Packard |
| Kristoffer Fleming | Intel |
| Dan Froelich | Intel |
| Stephan Gehring | TZero Technologies |
| Martin Gravenstein | Alereon |
| Jörg Habetha | Philips Semiconductors |

| | |
|---|---|
| Ghobad Heidari | Olympus |
| Guido Hiertz | Philips Semiconductors |
| Jin-Meng Ho | Texas Instruments |
| John Howard | Intel |
| Bob Huang | Sony |
| Sunil Jogi | Samsung |
| Peter Johansson | Congruent Software |
| John Keys | Intel |
| Paul Krakow | FOCUS Enhancements |
| Haim Kupershmidt | Wisair |
| Doug Lee | LeCroy |
| Bill Long | Staccato Communications |
| Suzuki Mitsuhiro | Sony |
| Nobuyuki Mizukoshi | NEC Electronics |
| Sam Mo | Panasonic |
| Alaa Muqattash | Olympus |
| Matthew Myers | Synopsys |
| Arun Naniyant | Samsung |
| Kaisa Nyberg | Nokia |
| Jay O'Conor | Philips Semiconductors |
| Nirmalendu Patra | Alereon |
| David Patton | Hewlett-Packard |
| Javier del Prado Pavon | Philips Semiconductors |
| Venkatesh Rajendran | Realtek Semiconductor |
| Mark Rich | CommStack |
| Tomoki Saito | NEC Electronics |
| Kazuyuki Sakoda | Sony |
| Juha Salokannel | Nokia |
| John Sarallo | WiQuest |
| Takashi Sato | Philips Semiconductors |
| Sid Schrum | WiQuest |
| Sai Shankar Nandagopalan | Philips Semiconductors |
| Israel Shapiro | Infineon Technologies |
| Etan Shirron | Infineon Technologies |
| Michael Sim | Panasonic |
| Fred Stivers | WiQuest |
| William Stoye | Artimi |
| Larry Taylor | Staccato Communications |
| Janne Tervonen | Nokia |
| Jean Tsao | HySignal |
| Alexander Weir | TES Electronic Solutions |
| Patrick Worfolk | TZero Technologies |
| Mehmet Zeybek | Philips Semiconductors |

# TABLE OF CONTENTS

**1 SCOPE ..... 1**
**2 REFERENCES ..... 2**
**3 DEFINITIONS ..... 3**
**4 ACRONYMS AND ABBREVIATIONS ..... 5**
**5 GENERAL DESCRIPTION ..... 7**
5.1 GENERAL DESCRIPTION OF THE ARCHITECTURE ..... 7
5.2 DEVICE ADDRESS ..... 7
5.3 FEATURES ASSUMED FROM THE PHY ..... 7
5.4 OVERVIEW OF MAC SERVICE FUNCTIONALITY ..... 8
5.5 MUX SUBLAYER ..... 12
5.6 MAC POLICIES ..... 12
5.7 TEST VECTORS ..... 12
**6 SERVICE ACCESS POINTS ..... 13**
6.1 GENERIC MANAGEMENT PRIMITIVES ..... 13
6.2 MLME SAP INTERFACE ..... 16
6.3 MAC SAP INTERFACE ..... 66
**7 MAC FRAME FORMATS ..... 70**
7.1 FRAME FORMAT CONVENTIONS ..... 70
7.2 GENERAL MAC FRAME FORMAT ..... 70
7.3 BEACON FRAMES ..... 75
7.4 CONTROL FRAMES ..... 77
7.5 COMMAND FRAMES ..... 80
7.6 DATA FRAMES ..... 85
7.7 AGGREGATED DATA FRAMES ..... 85
7.8 INFORMATION ELEMENTS ..... 86
**8 MAC SUBLAYER FUNCTIONAL DESCRIPTION ..... 101**
8.1 FRAME PROCESSING ..... 101
8.2 BEACON PERIOD ..... 110
8.3 PRIORITIZED CONTENTION ACCESS (PCA) ..... 116
8.4 DISTRIBUTED RESERVATION PROTOCOL (DRP) ..... 121
8.5 SYNCHRONIZATION OF DEVICES ..... 128
8.6 FRAGMENTATION AND REASSEMBLY ..... 130
8.7 AGGREGATION ..... 131
8.8 ACKNOWLEDGEMENT POLICIES ..... 131
8.9 PROBE ..... 134
8.10 DYNAMIC CHANNEL SELECTION ..... 134
8.11 MULTI-RATE SUPPORT ..... 134
8.12 TRANSMIT POWER CONTROL ..... 135
8.13 POWER MANAGEMENT MECHANISMS ..... 135

8.14 ASIE OPERATION ..... 138
8.15 RANGE MEASUREMENT OPERATION ..... 138
8.16 MAC SUBLAYER PARAMETERS ..... 141
**9 SECURITY ..... 143**
9.1 SECURITY MECHANISMS ..... 143
9.2 SECURITY MODES ..... 144
9.3 TEMPORAL KEYS ..... 147
9.4 FRAME RECEPTION STEPS AND REPLAY PREVENTION MEASURES ..... 152
9.5 AES-128 CCM INPUTS ..... 153
**ANNEX A (NORMATIVE) MUX SUBLAYER ..... 156**
A.1 MUX SERVICE ..... 156
A.2 MUX PROTOCOL DATA UNIT FORMAT ..... 156
**ANNEX B (NORMATIVE) MAC POLICIES ..... 158**
B.1 BEACON SLOT SELECTION ..... 158
B.2 RESERVATION LIMITS ..... 158
B.3 PCA RESERVATIONS ..... 160
B.4 RESERVATION LOCATIONS ..... 160
B.5 TRANSMIT POWER CONTROL ..... 160
B.6 MAC POLICIES PARAMETERS ..... 161
**ANNEX C (INFORMATIVE) RANGE MEASUREMENT CALCULATIONS ..... 162**
C.1 CALCULATE DISTANCE FOR A SINGLE MEASUREMENT ..... 162
C.2 CALCULATE DISTANCE UNCERTAINTY ..... 162
**ANNEX D (INFORMATIVE) TEST VECTORS ..... 166**
D.1 KCK/PTK GENERATION ..... 166
D.2 4-WAY HANDSHAKE MIC GENERATION ..... 167
D.3 NON-SECURE FRAME EXAMPLE ..... 168
D.4 SECURE FRAME EXAMPLE WITH EO = 0 ..... 169
D.5 SECURE FRAME EXAMPLE WITH EO = PAYLOAD LENGTH ..... 170
D.6 SECURE FRAME EXAMPLE WITH EO = 12 ..... 171
D.7 BEACON FRAME EXAMPLE ..... 172
D.8 FCS FIELD EXAMPLE ..... 175
**ANNEX E (INFORMATIVE) BIBLIOGRAPHY ..... 176**

# 1 Scope

This standard defines a distributed medium access control (MAC) sublayer for wireless networks. It specifies a wireless network structure that does not require any existing infrastructure for communication.

Categories of applications envisioned include electronic devices carried on a person, home electronics equipment, and personal computers and peripherals. Electronic devices carried on a person have specific requirements to support mobility and good power efficiency. Devices such as home electronics and computers are typically not as mobile, and not as sensitive to power efficiency. All of these devices benefit from a zero-infrastructure environment.

# 2 References

This standard shall be used in conjunction with the following publications. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this standard are encouraged to investigate the possibility of applying the most recent editions of the standards listed below.

IEEE Std 802®-2001, IEEE Standard for Local and Metropolitan Area Networks: Overview and Architecture.[1,2]

NIST FIPS Pub 197: Advanced Encryption Standard (AES), Federal Information Processing Standards Publication 197, US Department of Commerce/N.I.S.T., November 26, 2001.[3]

NIST Special Publication 800-38C, Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality.

*Multiband OFDM Physical Layer Specification*, WiMedia Alliance.

The Unicode Standard, Version 4.0, The Unicode Consortium, (Boston, MA, Addison-Wesley, 2003. ISBN 0-321-18578-1)

[1] IEEE publications are available from the Institute of Electrical and Electronics Engineers, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331, USA (http://standards.ieee.org/).

[2] The IEEE standards referenced are trademarks belonging to the Institute of Electrical and Electronics Engineers, Inc.

[3] NIST FIPS publications are available from the National Institute for Standards and Technology, 100 Bureau Drive, Stop 8900, Gaithersburg, MD 20899-8900 (http://www.nist.gov).

# 3 Definitions

For the purposes of this standard, the following terms and definitions apply. *The Authoritative Dictionary of IEEE Standards Terms*, Seventh Edition [B2][4], should be referenced for terms not defined in this clause.

**3.1 access category (AC):** A label for the common set of prioritized contention access (PCA) parameters that are used by a device to contend for the medium in order to transmit MAC protocol data units (MPDUs) with certain priorities.

**3.2 beacon group (BG):** The set of devices from which a device receives beacons that identify the same beacon period start time (BPST) as the device.

**3.3 beacon period (BP):** The period of time declared by a device during which it sends or listens for beacons.

**3.4 beacon period start time (BPST):** The start of the beacon period.

**3.5 channel:** The medium over which cooperating entities exchange information.

**3.6 data integrity:** The assurance that the data has not been modified from its original form.

**3.7 device:** An entity with a MAC sublayer that conforms to this standard.

**3.8 distributed reservation protocol (DRP):** A protocol implemented in each device to support negotiation and maintenance of channel time reservations binding on all neighbors of the reservation participants.

**3.9 extended beacon group:** The union of a device's beacon group and the beacon groups of all devices in the device's beacon group.

**3.10 frame:** A unit of data transmitted by a device.

**3.11 frame protection:** Security facilities provided for a frame, including (but not limited to) payload encryption, message authentication, and replay attack protection.

**3.12 MAC Client:** An entity above the MAC sublayer that generates MAC service data units for delivery to corresponding entities in other devices, and receives MAC service data units from such entities.

**3.13 MAC command data unit (MCDU):** A unit of data exchanged between peer medium access control entities in order to manage medium access control functions.

**3.14 MAC protocol data unit (MPDU):** The unit of data exchanged between two peer medium access control entities using the physical layer.

**3.15 MAC service data unit (MSDU):** Information that is delivered as a unit between medium access control service access points (SAPs).

**3.16 message integrity code (MIC):** A cryptographic checksum generated using a symmetric key that is typically appended to data in order to provide data integrity and source authentication similar to a digital signature.

**3.17 neighbor:** Any device in a device's beacon group.

**3.18 network allocation vector (NAV):** An indicator, maintained by each device capable of using PCA, of time periods when PCA-based transmission onto the wireless medium will not be initiated

[4] The numbers in brackets correspond to those of the bibliography in Annex E.

by the device, whether or not the device's clear-channel assessment function senses that the wireless medium is busy.

**3.19 prioritized contention access (PCA):** A prioritized CSMA/CA access mechanism used by devices for medium access.

**3.20 pseudo-random number generation:** The process of generating a deterministic sequence of bits from a given seed that has the statistical properties of a random sequence of bits when the seed is not known.

**3.21 random number generator:** A method or design that provides a sequence of bits that is unpredictable. A cryptographic random number generator is one specific type.

**3.22 reservation**: A named set of one or more medium access slots (MASs) within a superframe during which a device has preferential access to the medium.

**3.23 reservation block:** One or more temporally contiguous medium access slots (MASs) within a reservation not adjacent to other MASs in the reservation.

**3.24 secure frame:** A frame in which frame protection is applied.

**3.25 stream:** A logical flow of MSDUs from one device to one or more other devices.

**3.26 superframe:** The periodic time interval used in this standard to coordinate frame transmissions between devices, which contains a beacon period followed by a data period.

**3.27 symmetric key:** A secret key shared between two or more parties that may be used for both encryption and decryption as well as for message integrity code computation and verification.

**3.28 transmission opportunity (TXOP):** An interval of time obtained by a device using prioritized contention access (PCA) to initiate transmissions onto the medium.

**3.29 TXOP holder:** A device that has successfully contended for a TXOP.

**3.30 user priority:** A value assigned to an MSDU by the MAC client that determines the MSDU's transfer priority.

# 4 Acronyms and abbreviations

| | |
|---|---|
| AC | access category |
| ACK | acknowledgment |
| AES | advanced encryption standard |
| AIFS | arbitration inter-frame space |
| ASIE | application-specific information element |
| BPOIE | beacon period occupancy information element |
| BcstAddr | broadcast device address |
| BP | beacon period |
| BPST | beacon period start time |
| CBC-MAC | cipher block chaining-message authentication code |
| CCA | clear channel assessment |
| CCM | counter mode encryption and cipher block chaining message authentication code |
| CRC | cyclic redundancy check |
| CSMA/CA | carrier sense multiple access with collision avoidance |
| CTS | clear to send |
| DestAddr | destination device address |
| DevAddr | device address |
| B-ACK | block acknowledgment |
| DME | device management entity |
| DRP | distributed reservation protocol |
| EO | encryption offset |
| EUI | extended unique identifier |
| FCS | frame check sequence |
| FFI | fixed frequency interleaving |
| GTK | group temporal key |
| HCS | header check sequence |
| ID | identifier |
| IE | information element |
| IFS | inter-frame space |
| Imm-ACK | immediate acknowledgment |
| KCK | key confirmation key |
| MAC | medium access control |
| MAS | medium access slot |
| MCDU | MAC command data unit |
| McstAddr | multicast device address |
| MIB | management information base |
| MIC | message integrity code |
| MIFS | minimum inter-frame space |
| MKID | master key identifier |
| MLME | MAC sublayer management entity |
| MPDU | MAC protocol data unit |
| MSDU | MAC service data unit |
| NAV | network allocation vector |
| No-ACK | no acknowledgement |
| OUI | organizationally unique identifier |
| PCA | prioritized contention access |
| PHY | physical (layer) |
| PLCP | physical layer convergence protocol |
| PLME | physical layer management entity |
| PMK | pair-wise master key |
| PPDU | PHY protocol data unit |
| ppm | parts per million |
| PRF | pseudo-random function |

| | |
|---|---|
| PSDU | PHY service data unit |
| PTK | pair-wise temporal key |
| RSSI | received signal strength indication |
| RTS | request to send |
| SAP | service access point |
| SFC | secure frame counter |
| SFN | secure frame number |
| SIFS | short inter-frame space |
| SrcAddr | source device address |
| TFI | time frequency interleaving |
| TKID | temporal key identifier |
| TXOP | transmission opportunity |
| UDA | unused DRP reservation announcement |
| UDR | unused DRP reservation response |

# 5 General description

## 5.1 General description of the architecture

This standard defines a MAC service and protocol. As shown in Figure 1, this MAC sublayer corresponds to the MAC sublayer of the standard ISO/OSI-IEEE 802 reference model [B4]. The MAC service is provided by means of the MAC service access point (MAC SAP) to a single MAC service client, usually a higher layer protocol or adaptation layer. In this standard the MAC entity is represented by a device address.

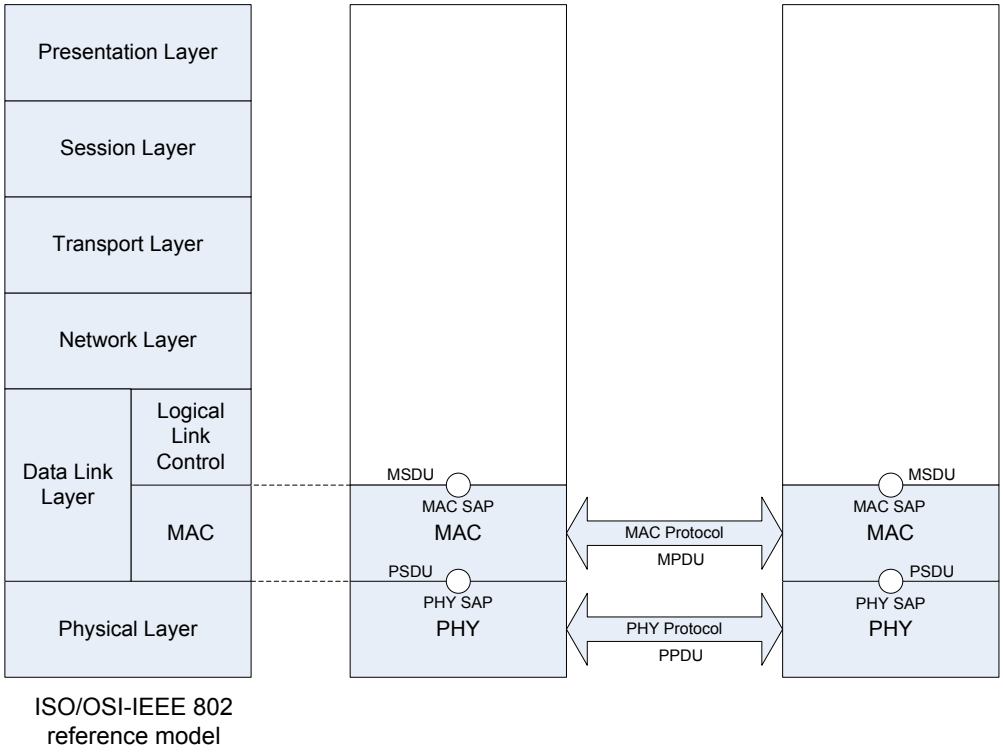



**Figure 1 — Architectural reference model**

The MAC sublayer in turn relies on the service provided by the PHY layer via the PHY service access point (PHY SAP). The MAC protocol applies between peer MAC entities.

## 5.2 Device address

Individual MAC entities are addressed via an EUI-48 [B1], and are associated with a volatile abbreviated address called a DevAddr. Unicast frames carry a destination DevAddr that identifies a single MAC entity.

DevAddrs are 16-bit values, generated locally, without central coordination. Consequently, it is possible for a single value to ambiguously identify two or more MAC entities. This standard provides mechanisms for resolving ambiguous DevAddrs.

The MAC addressing scheme includes multicast and broadcast address values. A multicast address identifies a group of MAC entities. The broadcast address identifies all MAC entities.

## 5.3 Features assumed from the PHY

A MAC entity is associated with a single PHY entity via that entity's PHY SAP. The PHY SAP is defined in the *Multiband OFDM Physical Layer Specification*.

The MAC sublayer requires the following features provided by the PHY:

— Frame transmission in both single frame and burst mode
— Frame reception for both single frame and burst mode transmission
— PLCP header error indication for both PHY and MAC header structures
— Clear channel assessment for estimation of medium activity
— Range measurement timestamps if MAC range measurement is supported

Figure 2 shows the structure of a PHY frame.

— There are two types of preamble: standard and burst.
— The PLCP header including MAC and PHY Headers is protected by a header check sequence (HCS).
— The Frame Payload is followed by its frame check sequence (FCS).

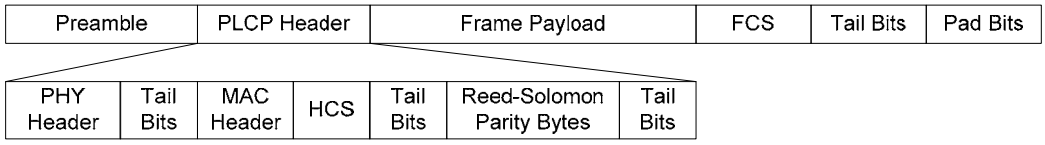



**Figure 2 — PHY Frame Structure**

Frames are transmitted by the PHY from the source device and delivered to the destination device in identical bit order. Throughout this specification reference to the start of a frame refers to the leading edge of the first symbol of the PHY frame at the local antenna and end of a frame refers to the trailing edge of the last symbol of the PHY frame.

Frame transmission and reception are supported by the exchange of parameters between the MAC sublayer and the PHY layer. These parameters allow the MAC entity to control, and be informed of, the frame transmission mode, the frame payload data rate and length, the frame preamble, the PHY channel and other PHY-related parameters.

In single frame transmission, the MAC entity has full control of frame timing. In burst mode transmission, the MAC entity has control of the first frame timing and the PHY provides accurate timing for the remaining frames in the burst.

## 5.4 Overview of MAC service functionality

The MAC service defined in this standard provides:

— Communication between cooperating devices within radio range on a single channel using the PHY;
— A distributed, reservation-based channel access mechanism;
— A prioritized, contention-based channel access mechanism;
— A synchronization facility for coordinated applications;
— Mechanisms for handling mobility and interference situations;
— Device power management by scheduling of frame transmission and reception;
— Secure communication with data authentication and encryption using cryptographic algorithms;
— A mechanism for measuring the distance between two devices.

The architecture of this MAC service is fully distributed. All devices provide all required MAC functions and optional functions as determined by the application. No device acts as a central coordinator.

Coordination of devices within radio range is achieved by the exchange of beacon frames. Periodic beacon transmission enables device discovery, supports dynamic network organization, and provides support for mobility. Beacons provide the basic timing for the network and carry reservation and scheduling information for accessing the medium.

### 5.4.1 Logical groups

The MAC protocol is specified with respect to an individual device, which has its own individual neighborhood. All MAC protocol facilities are expressed with respect to this individual neighborhood.

In a network formed with fully distributed medium access coordination, logical groups are formed around each device to facilitate contention-free frame exchanges while exploring medium reuse over different spatial regions. In this standard, these logical groups are a beacon group and an extended beacon group, both of which are determined with respect to an individual device.

### 5.4.2 Control algorithms

MAC protocol algorithms attempt to ensure that no member of the extended beacon group transmits a beacon frame at the same time as the device. Information included in beacon frames facilitates contention-free frame exchanges by ensuring that a device does not transmit frames while a neighbor is transmitting or receiving frames.

To permit correct frame reception, MAC protocol algorithms attempt to ensure that a device's DevAddr is unique within the device's extended beacon group.

### 5.4.3 Channel selection

When a device is enabled, it scans one or more channels for beacons and selects a channel. If no beacons are detected in the selected channel, the device creates its beacon period (BP) by sending a beacon.

If one or more beacons are detected in the selected channel, the device synchronizes its BP to existing beacons in the selected channel. The device exchanges data with members of its beacon group using the same channel the device selected for beacons.

Each device operates in a dynamic environment and under unlicensed operation rules. Thus, it is subject to interference from licensed users, other networks, and other unlicensed wireless entities in its channel. To enable the device to continue operation in this type of environment, each device has the capability to dynamically change the channel in which it operates without requiring disruption of links with its peers.

If at any time a device determines that the current channel is unsuitable, it uses the dynamic channel selection procedure, as described in 8.10, to move to a new channel.

### 5.4.4 Beacon period protection

Each device protects its and its neighbors' BPs for exclusive use of the beacon protocol. No transmissions other than beacons are attempted during the BP of any device. Protection of the device's BP is implicit.

A device may protect an alien BP, detected by reception of a beacon frame unaligned with the device's own BP, by announcing a reservation covering the alien BP in its beacon.

### 5.4.5 The superframe

The basic timing structure for frame exchange is a superframe. The superframe duration is specified as mSuperframeLength. The superframe is composed of 256 medium access slots (MASs), where each MAS duration is mMASLength.

Each superframe starts with a BP, which extends over one or more contiguous MASs. The start of the first MAS in the BP, and the superframe, is called the beacon period start time (BPST).

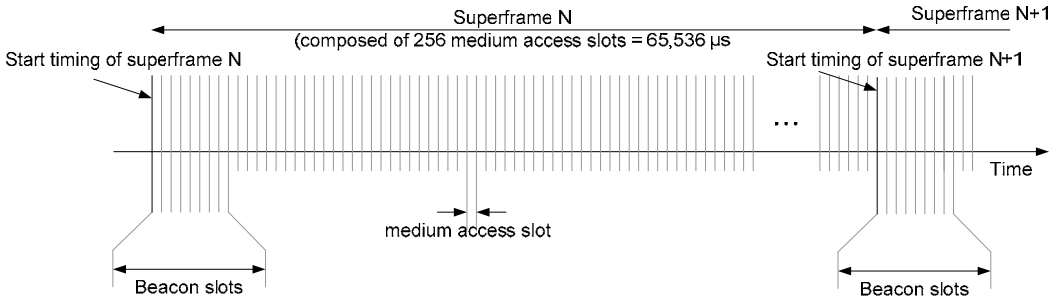



**Figure 3 —MAC superframe structure**

### 5.4.6 Medium access

The medium is accessed in one of three ways:

— During the BP, devices send only beacon frames, according to the rules specified in 8.2.
— During reservations, devices participating in the reservation send frames according to rules specified in 8.4.
— Outside the BP and reservations, devices may send frames using a prioritized contention-based access method, as described in 8.3.

### 5.4.7 Data communication between devices

Data is passed between the MAC entity and its client in MSDUs qualified by certain parameters. MSDUs are transported between devices in data frames. To reduce the frame error rate of a marginal link, data frames can be fragmented and reassembled, as described in 8.6. Fragments are numbered with an MSDU sequence number and a fragment number.

If the source device wishes to verify the delivery of a frame, then one of the acknowledgement policies is used, as described in 8.8. This standard provides for three types of acknowledgements to enable different applications. The No-ACK policy, described in 8.8.1, is appropriate for frames that do not require guaranteed delivery, or are delay sensitive and a retransmitted frame would arrive too late. The Imm-ACK policy, described in 8.8.2, provides an acknowledgement process in which each frame is individually acknowledged following the reception of the frame. The B-ACK policy, described in 8.8.3, lets the source send multiple frames without intervening ACK frames. Instead, the acknowledgements of the individual frames are grouped into a single response frame that is sent when requested by the source device. The B-ACK process decreases the overhead of the Imm-ACK process while allowing the source device to verify the delivery of frames to the destination.

If the source device does not receive the requested acknowledgement, then it may retransmit the frame, as described in 8.3.7 and 8.4.9, or it may discard the frame. The decision to retransmit or discard the frame depends on the type of data or command that is being sent, the number of times that the source device has attempted to send the frame, the length of time it has attempted to send the frame, and other implementation-dependent factors.

### 5.4.8 MAC frame data rates

MAC beacon frames are intended to be received and interpreted by all devices and hence their frame payloads are transmitted at pBeaconTransmitRate, which can be decoded by all recipients. Other frames are exchanged in a more restricted context and their frame payloads may be transmitted at higher data rates if possible. MAC headers are always transmitted at the lowest data rate supported by the PHY.

### 5.4.9 Security

Wireless networks present unique security challenges due to the loss of protection provided by wires and shielding. Distributed wireless networks present additional challenges due to the wide range of applications and use models that they must support. To name a few, eavesdroppers can overhear data exchanges not intended for them, whereas imposters can send forged data not using its own identity, can replay previously transmitted data, and can transmit modified data captured from a previous transmission.

This standard (clause 9) defines two levels of security: no security and strong security protection. Security protection includes data encryption, message integrity, and replay attack protection. Secure frames are used to provide security protection to data and aggregated data frames as well as selected control and command frames.

Three security modes are defined to control the level of security for devices in their communications. This standard allows for a device to use one of the two security levels or a combination of them in communicating with other devices by selecting the appropriate security mode (subclause 9.2).

This standard further specifies a 4-way handshake mechanism to enable two devices to derive their pair-wise temporal keys (PTKs) while authenticating their identity to each other. A secure relationship is established following a successful 4-way handshake between two devices (subclause 9.3.1). A 4-way handshake between two devices is conducted based on a shared master key. How two devices obtain their shared master keys is outside the scope of this standard.

In addition, this standard provides means for the solicitation and distribution of group temporal keys (GTKs). While PTKs are used for protecting unicast frames exchanged between two devices, GTKs are employed for protecting multicast and broadcast frames transmitted from a source device to a multicast or broadcast group of recipient devices (subclause 9.3.2).

A pseudo-random function is defined based on the MIC generation by CCM using AES-128 (subclause 9.3.3). It can be made available to entities outside the MAC sublayer for random number generation.

Secure frame counters and replay counters are set up on a per-temporal key basis to guarantee message freshness (subclause 9.4). No specific mechanisms are created in this standard to address denial of service attacks given the open nature of the wireless medium.

In this standard, 128-bit symmetric temporal keys are employed based on AES-128 with CCM to provide payload encryption and message integrity code (MIC) generation (subclause 9.5).

In general, this standard specifies security mechanisms, not security policies.

### 5.4.10 Information discovery

The protocols and facilities of this standard are supported by the exchange of information between devices. Information can be broadcast in beacon frames or requested in Probe commands. For each type of information, an Information Element (IE) is defined. IEs can be included by a device in its beacon at any time and may optionally be requested or provided using the Probe command.

A device uses the MAC Capabilities IE and PHY Capabilities IE to announce information about its support of variable or optional facilities. Declaration of capabilities is especially useful when a device detects changes in its immediate neighborhood.

### 5.4.11 Support for higher-layer timer synchronization

Some applications, for example, the transport and rendering of audio or video streams, require synchronization of timers located at different devices. Greater accuracy (in terms of jitter bounds) or finer timer granularity than that provided by the synchronization mechanism described in 8.5 may be an additional requirement. In support of such applications, this standard defines an optional MAC facility that enables layers above the MAC sublayer to accurately synchronize timers located in different devices. The facility is usable by more than one application at a time.

### 5.4.12 Rate adaptation

A mechanism for data rate adaptation is provided in 8.11. A receiver may use this mechanism to inform a transmitter of the optimal data rate to increase throughput and/or reduce the frame error rate (FER).

### 5.4.13 Power management

An important goal of this standard is to enable long operation time for battery powered devices. An effective method to extend battery life is to enable devices to turn off completely or reduce power for long periods of time, where a long period is relative to the superframe duration.

This standard provides two power management modes in which a device can operate: active and hibernation. Devices in active mode transmit and receive beacons in every superframe. Devices in hibernation mode hibernate for multiple superframes and do not transmit or receive in those superframes.

In addition, this standard provides facilities to support devices that sleep for portions of each superframe in order to save power.

To coordinate with neighbors, a device indicates its intention to hibernate by including a Hibernation Mode IE in its beacon. The Hibernation Mode IE specifies the number of superframes in which the device will sleep and will not send or receive beacons or any other frames.

Power management mechanisms are described in 8.13.

### 5.4.14 Range measurement

A device may contain provisions to support one-dimensional ranging measurements between devices using two-way time transfer techniques. This standard describes methods in the MAC sublayer to make range measurements in 8.15.

## 5.5 MUX sublayer

In order to enable the coexistence of concurrently active higher layer protocols within a single device, a multiplexing sublayer is defined. This sublayer routes outgoing and incoming MSDUs to and from their corresponding higher layers. The MUX sublayer is described in Annex A.

## 5.6 MAC policies

It is desirable to allow and facilitate equitable and efficient coexistence of devices with varying medium access requirements. For this purpose, Annex B specifies policies governing channel selection and sharing of bandwidth. These policies impose, among other things, certain restrictions on the number and configuration of MASs in DRP reservations, on the location of reserved MASs within a superframe, and on channel selection order.

## 5.7 Test vectors

To facilitate implementation and interoperability, Annex D provides examples of field encoding in MAC frames and the corresponding octet sequences passed to the PHY SAP. The examples include results from security operation and FCS calculation.

# 6 Service access points

Service access points (SAPs) are provided for both data transfer as well as management of the MAC sublayer. Data transfer for the MAC sublayer is through the MAC SAP.

Both the MAC sublayer and the PHY layer conceptually include management entities, called the MAC sublayer management entity (MLME) and physical layer management entity (PLME). These entities provide the layer management service interfaces for the layer management functions. In order to provide correct MAC protocol operation, a device management entity (DME) should be present within each device. The DME is a layer-independent entity that may be viewed as residing in a separate management plane or as residing "off to the side." The exact functionality of the DME is not specified in this standard, but in general this entity may be viewed as being responsible for such functions as the gathering of layer-dependent status from the various layer management entities, and similarly setting the value of layer-specific parameters. The DME typically performs such functions on behalf of the general system management entities and implements standard management protocols. Figure 4 depicts the relationship among the management entities.

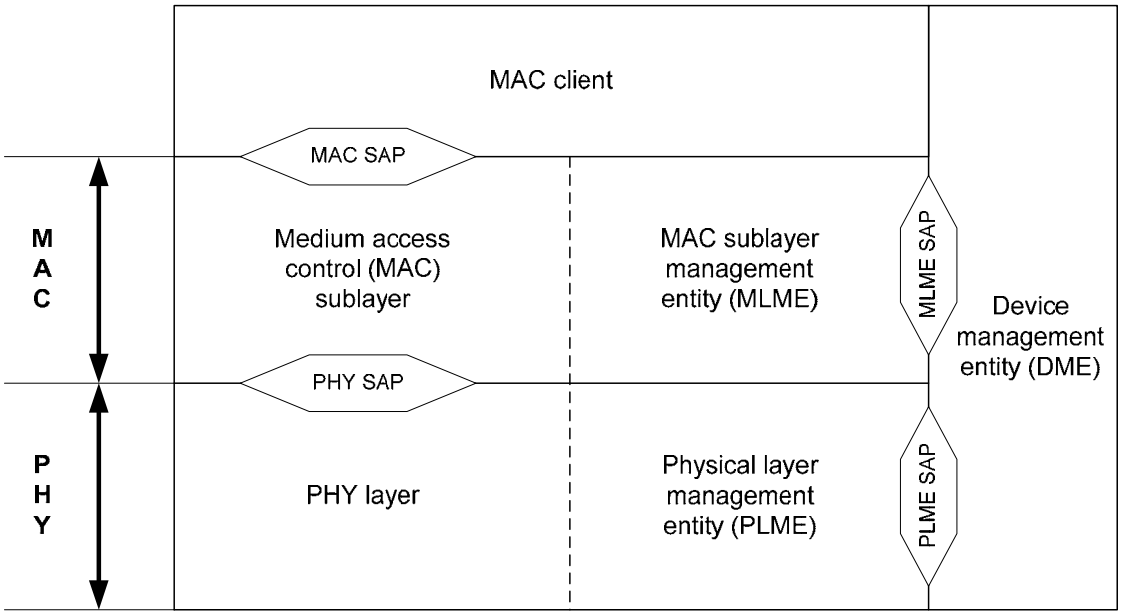



**Figure 4 — The SAP reference model used in this standard**

The various entities within this model interact in various ways. Certain of these interactions are defined explicitly within this standard, via a service access point (SAP) across which defined primitives are exchanged. Other interactions are not defined explicitly within this standard, such as the interface between the MAC entity and the MLME or the interface between the PHY and the PLME. The specific manner in which these MAC and PHY interfaces are integrated into the overall MAC sublayer and PHY layer is not specified within this standard.

This specification defines the MAC SAP and the MLME SAP.

## 6.1 Generic management primitives

The management information specific to the MAC sublayer is represented as a management information base (MIB). However, devices are not intended to be managed across a network but rather use the management information to ascertain the characteristics of the MAC sublayer.

The MLME is viewed as "containing" the MIB for the MAC sublayer. The generic model of MIB-related management primitives exchanged across the management SAP is to allow the SAP user

entity to either “GET” the value of a MIB attribute, or to “SET” the value of a MIB attribute. The invocation of a SET.request primitive may require the layer entity to perform certain defined actions.

The GET and SET primitives are represented as requests with associated confirm primitives. These primitives are prefixed by MLME for the MAC sublayer management SAP. The DME uses the service provided by the MLME through the MLME SAP. The primitives are summarized in Table 1.

**Table 1 — Summary of generic management primitives**

| Name | Request | Confirm |
|---|---|---|
| MLME-GET | 6.2.1 | 6.2.2 |
| MLME-SET | 6.2.3 | 6.2.4 |

The parameters used for these primitives are defined in Table 2.

**Table 2 — MLME generic management primitive parameters**

| Name | Type | Valid range | Description |
|---|---|---|---|
| MIBattribute | Octet string | Any MIB attribute as defined in 6.1.5 | The name of the MIB attribute |
| MIBvalue | Variable | As defined in 6.1.5 | The value of the MIB attribute |
| ResultCode | Enumeration | SUCCESS, INVALID_MIB_ATTRIBUTE_NAME, INVALID_MIB_ATTRIBUTE_VALUE, READ_ONLY_MIB_ATTRIBUTE, WRITE_ONLY_MIB_ATTRIBUTE | Indicates the result of the MLME or PLME request |

### 6.1.1 MLME-GET.request

This primitive requests information about a given MAC MIB attribute. The semantics of this primitive are:

```
MLME-GET.request(
     MIBattribute
     )
```

The primitive parameter is defined in Table 2.

#### 6.1.1.1 When generated

This primitive is generated by the DME to obtain information from the MAC MIB.

#### 6.1.1.2 Effect of receipt

The MLME attempts to retrieve the requested MIB attribute from its database and responds with MLME-GET.confirm that gives the result.

### 6.1.2 MLME-GET.confirm

This primitive reports the results of an information request about the MAC MIB. The semantics of this primitive are:

```
MLME-GET.confirm(
     ResultCode,
```

```
        MIBattribute,
        MIBvalue
        )
```

The primitive parameters are defined in Table 2.

#### 6.1.2.1 When generated

This primitive is generated in response to an MLME-GET.request by the DME.

#### 6.1.2.2 Effect of receipt

The primitive returns the appropriate MIB attribute value if the ResultCode is SUCCESS; otherwise it returns an error indication in the ResultCode. Possible values of the ResultCode that would indicate an error are INVALID_MIB_ATTRIBUTE_NAME and WRITE_ONLY_MIB_ATTRIBUTE.

### 6.1.3 MLME-SET.request

This primitive attempts to set the indicated MAC MIB attribute to the given value. The semantics of this primitive are:

```
MLME-SET.request(
     MIBattribute,
     MIBvalue
     )
```

The primitive parameters are defined in Table 2.

#### 6.1.3.1 When generated

This primitive is generated by the DME to set the indicated MAC MIB attribute.

#### 6.1.3.2 Effect of receipt

The MLME attempts to set the requested MIB attribute in its database. If this MIB attribute implies a specific action, then this requests that the action be performed. The MLME responds with MLME-SET.confirm that gives the result.

### 6.1.4 MLME-SET.confirm

This primitive reports the results of an attempt to set the value of an attribute in the MAC MIB. The semantics of this primitive are:

```
MLME-SET.confirm(
     ResultCode,
     MIBattribute
     )
```

The primitive parameters are defined in Table 2.

#### 6.1.4.1 When generated

This primitive is generated in response to an MLME-SET.request by the DME.

#### 6.1.4.2 Effect of receipt

If the ResultCode is SUCCESS, this confirms that the indicated MIB attribute was set to the requested value; otherwise it returns an error condition in the ResultCode. If this MIBattribute implies a specific action, then this confirms that the action was performed. Possible ResultCodes for an error are:

— INVALID_MIB_ATTRIBUTE_NAME
— INVALID_MIB_ATTRIBUTE_VALUE
— READ_ONLY_MIB_ATTRIBUTE

### 6.1.5 MAC management information base (MIB)

The MAC MIB comprises the managed objects, attributes, actions, and notifications required to manage the MAC sublayer of a device. Table 3 contains a list of managed items. All parameters are read/write, meaning that the DME is able to change it using the MLME-SET.request primitive.

**Table 3 —MAC MIB attributes**

| Managed Object | Range |
|---|---|
| mDevAddrType | {Private, Generated} |
| mSecurityModeSelected | {0,1,2} |

## 6.2 MLME SAP interface

### 6.2.1 Reset

The service primitives in this subclause are provided for the DME to reset the MAC entity. The primitives covered in this subclause are listed in Table 4.

**Table 4 — Reset primitives**

| Service Primitive | Request | Indication | Response | Confirm |
|---|---|---|---|---|
| MLME-RESET | 6.2.1.1 | | | 6.2.1.2 |

Table 5 lists the parameters that appear in the primitives of this subclause.

**Table 5 — Reset primitive parameters**

| Name | Type | Valid range | Description |
|---|---|---|---|
| ResultCode | Enumeration | SUCCESS, FAILURE | Appears in MLME-RESET.confirm only; indicates the result of the corresponding MLME-RESET.request. |

#### 6.2.1.1 MLME-RESET.request

This primitive requests to reset the MAC entity data path and its management entity and MIB.

The semantics of this primitive are:

```
MLME-RESET.request(
     )
```

##### 6.2.1.1.1 When generated

This primitive is generated by the DME when it needs to reset the MAC entity.

##### 6.2.1.1.2 Effect of receipt

The MAC entity resets both the transmit and receive state machines, its management entity and MIB to the default power on states or default values. The MAC entity discards all MSDUs and their fragments, if any, that are buffered for transmission to a peer MAC entity or delivery to the MAC client. Until the MLME receives other primitives, the MAC entity will not perform any transmit or receive operations. The MLME issues an MLME-RESET.confirm when the reset has completed, to reflect the results of the reset request.

#### 6.2.1.2 MLME-RESET.confirm

This primitive reports the results of a reset procedure.

The semantics of this primitive are:

```
MLME-RESET.confirm(
     ResultCode
     )
```

##### 6.2.1.2.1 When generated

This primitive is generated by the MLME as a result of an MLME-RESET.request at the completion of the reset operation.

##### 6.2.1.2.2 Effect of receipt

The DME is notified of the results of the reset procedure.

### 6.2.2 Scan

The service primitives in this subclause are provided for the DME to perform channel scan operations. The primitives covered in this subclause are listed in Table 6.

**Table 6 — Scan primitives**

| Service Primitive | Request | Indication | Response | Confirm |
|---|---|---|---|---|
| MLME-SCAN | 6.2.2.1 | | | 6.2.2.2 |
| MLME-SCAN-PLCP-HEADER-RECEIVED | | 6.2.2.3 | | |

Table 7 lists the parameters that appear in the primitives of this subclause.

**Table 7 — Scan primitive parameters**

| Name | Type | Valid range | Description |
|---|---|---|---|
| ChannelNumber | Enumeration | PHY dependent | The physical channel to be scanned |
| BPSTOffset | Integer | 0–65535 | The offset of the start of a received frame relative to the BPST of the device, measured in microseconds |
| LQI | Integer | 0–255 | Link quality indication |
| PLCPHeader | PLCP header | | The PLCP header of the received frame |
| ResultCode | Enumeration | SUCCESS, FAILURE | Indicates the result of the corresponding request |
| RSSI | Integer | 0–255 | Receive signal strength indication |
| ScanState | Enumeration | SCAN_ONLY, SCAN_OUTSIDE_BP, SCAN_WHILE_INACTIVE, SCAN_DISABLED | Sets the scan state in the MLME-SCAN primitive |

#### 6.2.2.1 MLME-SCAN.request

This primitive begins a scan operation.

The semantics of this primitive are:

```
MLME-SCAN.request(
     ChannelNumber,
     ScanState
     )
```

ScanState indicates the requested scanning state. Table 8 shows the possible scan types:

**Table 8 — ScanState parameter values**

| ScanState Value | Description |
|---|---|
| SCAN_ONLY | Scan only. No other transmit or receive operation is performed until the scan is completed. |
| SCAN_OUTSIDE_BP | Scan at all times except in the BP. |
| SCAN_WHILE_INACTIVE | Scan only when not scheduled to transmit or receive. |
| SCAN_DISABLED | Scanning is disabled. This is the default scanning state on power up or after an MLME-RESET request completes successfully. |

The ChannelNumber indicates the PHY channel to use during the scan.

#### 6.2.2.1.1 When generated

This primitive is generated by the DME to change the scanning state.

#### 6.2.2.1.2 Effect of receipt

The MLME initiates a change of the scanning state to the new state indicated by the request.

### 6.2.2.2 MLME-SCAN.confirm

This primitive confirms that a scanning state change initiated by the MLME-SCAN request has been successfully completed or that the state change attempt has failed.

The semantics of this primitive are:

```
MLME-SCAN.confirm(
     ResultCode
     )
```

ResultCode indicates whether the operation to change the scan state has successfully completed. If changing the scan state does not succeed, it is a vendor specific decision when to time out the operation and return FAILURE.

#### 6.2.2.2.1 When generated

This primitive is generated by the MLME as a result of an MLME-SCAN.request once the requested scan state has been entered or the operation has failed.

#### 6.2.2.2.2 Effect of receipt

The recipient is notified of the results of the procedure of changing the scan state.

### 6.2.2.3 MLME-SCAN-PLCP-HEADER-RECEIVED.indication

This indication informs the DME that a PLCP header was received. This indication only occurs when the scan state is not SCAN_DISABLED.

The semantics of this primitive are:

```
MLME-SCAN-PLCP-HEADER-RECEIVED.indication(
     PLCPHeader,
```

```
ChannelNumber,
BPSTOffset,
LQI,
RSSI
)
```

Channel Number is the PHY channel on which the PLCP header was received.

#### 6.2.2.3.1 When generated

This indication is generated by the MLME when the device receives a PLCP header and the MAC entity is not in the SCAN_DISABLED state.

#### 6.2.2.3.2 Effect of receipt

The recipient is notified that a PLCP header has been received.

### 6.2.3 Beacon transmission and reception

The service primitives in this subclause are provided for the DME to control beacon transmission and reception. The primitives covered in this subclause are listed in Table 9.

**Table 9 — Beacon primitives**

| Service Primitive | Request | Indication | Response | Confirm |
|---|---|---|---|---|
| MLME-BEACON-START | 6.2.3.1 | | | 6.2.3.2 |
| MLME-BEACON-STOP | 6.2.3.3 | | | 6.2.3.4 |
| MLME-BEACON-CHANGE-CHANNEL | 6.2.3.5 | | | 6.2.3.6 |
| MLME-BEACON | | 6.2.3.7 | | |
| MLME-BEACON-MERGE-BP | 6.2.3.8 | 6.2.3.9 | | 6.2.3.10 |

Table 10 lists the parameters that appear in the primitives of this subclause.

**Table 10 — Beacon primitive parameters**

| Name | Type | Valid range | Description |
|---|---|---|---|
| BeaconInfo | Array | Variable | A variable size array containing all information for a transmitted or received beacon |
| ChannelChangeIECount | Integer | 0–255 | The number of superframes in which to include a Channel Change IE prior to changing channels |
| ChannelNumber | Enumeration | PHY dependent | The physical channel on which the beacon will be transmitted or was received |
| BPMoveCountdown | Integer | 0–255 | The number of superframes before a BP merge commences, based on a neighbor's BP Switch IE |
| BPSTOffset | Integer | 0–65535 | The offset of the start of a received beacon relative to the BPST of the device, measured in microseconds |
| BeaconType | Enumeration | NEIGHBOR, ALIEN, OWN | The type of beacon reported, indicating whether a received beacon was classified as a neighbor or an alien, or if the beacon was transmitted by the device |
| LQI | Integer | 0–255 | The relative value of the PHY-dependent link quality indication associated with a received frame |
| ResultCode | Enumeration | SUCCESS, FAILURE, FAILURE_NO_SLOTS | Indicates the result of the corresponding MLME-BEACON-START or MLME-BEACON-STOP request |
| RSSI | Integer | 0–255 | The relative value of the PHY-dependent received signal strength indication associated with a received frame |

#### 6.2.3.1 MLME-BEACON-START.request

This primitive instructs the MAC entity to begin beacon transmission on the specified channel.

The semantics of this primitive are:

```
MLME-BEACON-START.request(
     ChannelNumber
     )
```

ChannelNumber is the physical layer channel on which beacon transmission will begin.

##### 6.2.3.1.1 When generated

This primitive is generated by the DME to instruct the MAC entity to begin beacon transmission.

##### 6.2.3.1.2 Effect of receipt

When the MLME receives the request it initiates the process of starting to transmit beacons on the indicated channel.

#### 6.2.3.2 MLME-BEACON-START.confirm

This primitive confirms the completion of an MLME-BEACON-START request. This confirmation indicates whether the first beacon has been transmitted.

The semantics of this primitive are:

```
MLME-BEACON-START.confirm(
     ResultCode
     )
```

ResultCode indicates whether beacons have begun to be successfully transmitted. If an operation to start beacon transmission is not succeeding, it is a vendor specific decision when to time out the operation and return FAILURE.

#### 6.2.3.2.1 When generated

This primitive is generated by the MLME as a result of an MLME-BEACON-START.request to indicate that the first beacon has been successfully transmitted or that the operation has failed.

#### 6.2.3.2.2 Effect of receipt

The recipient is notified of the results of the procedure of starting to transmit beacons.

### 6.2.3.3 MLME-BEACON-STOP.request

This primitive causes the MAC entity to stop transmitting beacons.

The semantics of this primitive are:

```
MLME-BEACON-STOP.request(
     )
```

#### 6.2.3.3.1 When generated

This primitive is generated by the DME to stop beacon transmission that was started with the MLME-BEACON-START request.

#### 6.2.3.3.2 Effect of receipt

The MLME immediately initiates ending the transmission of beacons.

### 6.2.3.4 MLME-BEACON-STOP.confirm

This primitive confirms that beacon transmission has been stopped by the MLME after the DME issued the MLME-BEACON-STOP request.

The semantics of this primitive are:

```
MLME-BEACON-STOP.confirm(
     ResultCode
     )
```

ResultCode indicates whether the beacon operation successfully stopped. If ending beacon transmission is not successful, it is a vendor specific decision when to time out the operation and return FAILURE.

#### 6.2.3.4.1 When generated

This primitive is generated by the MLME as a result of an MLME-BEACON-STOP.request once beacon transmission has been successfully stopped or the operation has failed.

#### 6.2.3.4.2 Effect of receipt

The recipient is notified of the results of the procedure of stopping beacon transmission. A successful confirmation indicates at least one BP has passed without a beacon being transmitted.

### 6.2.3.5 MLME-BEACON-CHANGE-CHANNEL.request

This primitive causes the MAC entity to change the channel on which it transmits beacons (and all other frames).

The semantics of this primitive are:

```
MLME-BEACON-CHANGE-CHANNEL.request(
     ChannelNumber,
     ChannelChangeIECount
     )
```

#### 6.2.3.5.1 When generated

This primitive is generated by the DME to select a new channel for transmission.

#### 6.2.3.5.2 Effect of receipt

The MLME includes Channel Change IEs in zero or more superframes as specified by the DME. It changes the PHY channel it uses to the channel indicated by the ChannelNumber parameter at the end of the superframe in which the Channel Change Countdown field was zero, or at the end of the current superframe if the ChannelChangeIECount parameter is zero.

### 6.2.3.6 MLME-BEACON-CHANGE-CHANNEL.confirm

This primitive confirms that the MLME has changed PHY channels as a result of a MLME-BEACON-CHANGE-CHANNEL request.

The semantics of this primitive are:

```
MLME-BEACON-CHANGE-CHANNEL.confirm(
     ResultCode
     )
```

ResultCode indicates whether the PHY channel was changed successfully. If changing the PHY channel is not successful, it is a vendor specific decision when to time out the operation and return FAILURE.

#### 6.2.3.6.1 When generated

This primitive is generated by the MLME as a result of an MLME-BEACON-CHANGE-CHANNEL.request once the PHY channel has been changed or the operation has failed.

#### 6.2.3.6.2 Effect of receipt

The recipient is notified of the results of the procedure of changing channels. A successful confirmation indicates at least one BP has passed with beacon transmission and reception on the new channel.

### 6.2.3.7 MLME-BEACON.indication

This indication informs the DME that a beacon was received or transmitted. The indication occurs any time the device receives a beacon or transmits its own beacon.

The semantics of this primitive are:

```
MLME-BEACON.indication(
     BeaconInfo,
     ChannelNumber,
     BPSTOffset,
     BeaconType,
     LQI,
     RSSI
     )
```

BeaconInfo is an array including the MAC Header, beacon parameters, and all information elements for the received or transmitted beacon. Channel Number is the PHY channel on which the beacon was received or transmitted.

If the device is active, the MAC entity always indicates at least one beacon, its own, during its BP.

#### 6.2.3.7.1 When generated

This indication is generated by the MLME when it determines it has transmitted or received a beacon frame. Beacon indications for received beacons can occur any time a device's receiver is enabled, whether scanning is enabled or not.

#### 6.2.3.7.2 Effect of receipt

The recipient is notified that a beacon has been received and provided with all information contained in the beacon.

### 6.2.3.8 MLME-BEACON-MERGE-BP.request

This primitive allows the DME to request initiation of the merging process of two previously separate BPs.

The semantics of this primitive are:

```
MLME-BEACON-MERGE-BP.request(
     BPSTOffset
     )
```

BPSTOffset identifies the BPST of the alien BP where the device will move.

#### 6.2.3.8.1 When generated

This primitive is generated by the DME to instruct the MAC entity to merge its BP with an alien BP.

#### 6.2.3.8.2 Effect of receipt

When the MLME receives this primitive, the device relocates its BP to the alien BP according to 8.2.6.3.

### 6.2.3.9 MLME-BEACON-MERGE-BP.indication

This indication informs the DME that a neighbor has announced it will change its BPST as a result of a BP merge operation.

The semantics of this primitive are:

```
MLME-BEACON-MERGE-BP.indication(
     BPSTOffset,
     BPMoveCountdown
     )
```

#### 6.2.3.9.1 When generated

This indication is generated by the MLME when the device receives a BP Switch IE in a beacon from a neighbor.

#### 6.2.3.9.2 Effect of receipt

The recipient is notified that a BP Switch IE was received and a BP merge may result.

### 6.2.3.10 MLME-BEACON-MERGE-BP.confirm

This primitive confirms that a BP merge has completed or terminated.

The semantics of this primitive are:

```
MLME-BEACON-MERGE-BP.confirm(
     ResultCode
     )
```

ResultCode indicates whether the merge operation successfully completed or was terminated.

#### 6.2.3.10.1 When generated

This primitive is generated by the MLME as a result of an MLME-BEACON-MERGE-BP.request once the BP merge operation has completed or terminated.

#### 6.2.3.10.2 Effect of receipt

The recipient is notified of the results of the BP merge procedure.

### 6.2.4 IE management

The service primitives in this subclause provide access to and control of information contained in certain IEs, for use by higher layers. The primitives covered in this subclause are listed in Table 11.

**Table 11 — IE Management primitives**

| Service Primitive | Request | Indication | Response | Confirm |
|---|---|---|---|---|
| MLME-SET-CAPABILITY-IE | 6.2.4.1 | | | 6.2.4.2 |
| MLME-IDENTIFICATION-IE | 6.2.4.3 | | | 6.2.4.4 |

Table 12 lists the parameters that appear in the primitives of this subclause.

**Table 12 — IE Management primitive parameters**

| Name | Type | Valid range | Description |
|---|---|---|---|
| CapIEData | Array | Variable | A variable size array containing Capabilities IE data |
| IdentificationIEData | Array | Variable | A variable size array containing the Identification IE data |
| ResultCode | Enumeration | SUCCESS, FAILURE | Indicates the result of the MLME-SET-CAPABILITIES-IE request |

#### 6.2.4.1 MLME-SET-CAPABILITIES-IE.request

This primitive changes the content of the Capabilities IE in the beacon.

The semantics of this primitive are:

```
MLME-SET-CAPABILITIES-IE.request(
     CapIEData
     )
```

CapIEData is the data to use in the Capabilities IE in subsequent beacons.

#### 6.2.4.1.1 When generated

This primitive is generated by the DME to instruct the MAC entity to change the Capabilities IE content in the beacon.

#### 6.2.4.1.2 Effect of receipt

When the MLME receives the request it initiates changing the Capabilities IE content in the beacon.

#### 6.2.4.2 MLME-SET-CAPABILITIES-IE.confirm

This primitive confirms the result of an operation to change the Capabilities IE content in the beacon in response to an MLME-SET-CAPABILITIES-IE request.

The semantics of this primitive are:

```
MLME-SET-CAPABILITIES-IE.confirm(
     ResultCode
     )
```

ResultCode indicates the result of the attempt to change the Capabilities IE content in the beacon. If the Capabilities IE content can not be changed the primitive is generated with a FAILURE status.

##### 6.2.4.2.1 When generated

This primitive is generated by the MLME as a result of an MLME-SET-CAPABILITIES-IE.request with a successful return status to indicate that the first beacon has been transmitted with the changed Capabilities IE content. If MLME-SET-CAPABILITIES-IE.request was called when beacons were not being transmitted, a successful status indicates that the new Capabilities IE content will be in the beacon when the next beacon is sent.

##### 6.2.4.2.2 Effect of receipt

The recipient is notified of the results of the procedure of changing the Capabilities IE content in the beacon.

#### 6.2.4.3 MLME-IDENTIFICATION-IE.request

This primitive controls the contents of the Identification IE in the beacon.

The semantics of this primitive are:

```
MLME-IDENTIFICATION-IE.request(
     IdentificationIEData
     )
```

##### 6.2.4.3.1 When generated

This primitive is generated by the DME to instruct the MAC entity to change the device's Identification IE content.

##### 6.2.4.3.2 Effect of receipt

When the MLME receives the request it changes the Identification IE content.

#### 6.2.4.4 MLME-IDENTIFICATION-IE.confirm

This primitive confirms the result of an operation to change the Identification IE content.

The semantics of this primitive are:

```
MLME-IDENTIFICATION-IE.confirm(
     ResultCode
     )
```

ResultCode indicates the result of the attempt to change the Identification IE content. If the Identification IE content cannot be changed the primitive is generated with a FAILURE status.

##### 6.2.4.4.1 When generated

This primitive is generated by the MLME as a result of an MLME-IDENTIFICATION-IE.request with a successful return status to indicate that the new content is stored and will be used in the next beacon to contain an Identification IE.

#### 6.2.4.4.2 Effect of receipt

The recipient is notified of the results of the procedure of changing the Identification IE content.

### 6.2.5 PTK establishment

The mechanism supports the procedure of establishing a new PTK with a peer MAC entity. The primitives covered in this subclause are listed in Table 13.

**Table 13 — PTK establishment primitives**

| Service Primitive | Request | Indication | Response | Confirm |
|---|---|---|---|---|
| MLME-PTK | 6.2.5.1 | 6.2.5.2 | 6.2.5.3 | 6.2.5.4 |

Table 14 lists the parameters that appear in the primitives of this subclause.

**Table 14 — PTK establishment primitive parameters**

| Name | Type | Valid range | Description |
|---|---|---|---|
| SrcEUI | EUI-48 | Any valid unicast EUI-48 | Specifies the EUI-48 of the device requesting to establish a new PTK by a 4-way handshake as described in clause 9 |
| DestEUI | EUI-48 | Any valid unicast EUI-48 | Specifies the EUI-48 of the device requested to establish a new PTK by a 4-way handshake as described in clause 9 |
| MessageNumber | Integer | Any valid positive number as specified in clause 9 | Specifies the number of the message being conveyed in the 4-way handshake |
| StatusCode | Integer | Any valid value as specified in clause 9 | Specifies the status of the 4-way handshake |
| PTKID | Integer | A non-zero number as specified in clause 9 | Specifies the TKID of the new PTK established by the 4-way handshake |
| MKID | Octet string | 16 octets as specified in clause 9 | Identifies the master key used to establish a new PTK by a 4-way handshake |
| PTK | Octet string | 16 octets as specified in clause 9 | Specifies the new PTK established by the 4-way handshake |
| PTKFailureTimeout | Integer | ≥1 | Specifies a time limit in microseconds after which the procedure of establishing a new PTK must be terminated |
| ResultCode | Enumeration | RESPONSE_RECEIVED, INVALID_PARAMETERS, TIMEOUT | Indicates the result of the corresponding MLME-PTK.request |

#### 6.2.5.1 MLME-PTK.request

This primitive requests establishment of a new PTK with a specified peer MAC entity as described in clause 9.

The semantics of this primitive are:

```
MLME-PTK.request(
     DestEUI,
     MessageNumber,
     StatusCode,
     PTKID,
```

```
        MKID,
        PTKFailureTimeout
        )
```

#### 6.2.5.1.1 When generated

This primitive is generated by the DME for the local MAC entity to start a 4-way handshake procedure to establish a new PTK with a specified peer MAC entity. This primitive is also generated by the DME for the local MAC entity to continue with an ongoing 4-way handshake procedure upon receiving a valid PTK command containing a Message Number of 2 and a StatusCode of zero.

#### 6.2.5.1.2 Effect of receipt

The MAC entity initiates or continues with a 4-way handshake using the master key specified by the MKID parameter to establish a new PTK with a specified peer MAC entity specified by the DestEUI. If the MessageNumber in the MLME-PTK.request is 1, the MAC entity generates a new random number as the I-Nonce and transmits a PTK command containing the first message of the 4-way handshake to the peer MAC entity. If the MessageNumber in the MLME-PTK.request is 3, the MAC entity uses its I-Nonce generated for the first message of the same 4-way handshake and transmits a PTK command containing the third message of the 4-way handshake to the peer MAC entity. The MLME subsequently issues an MLME-PTK.confirm to reflect the results.

### 6.2.5.2 MLME-PTK.indication

This primitive reports the request or establishment of a new PTK with a specific peer MAC entity.

The semantics of this primitive are:

```
MLME-PTK.indication(
     SrcEUI,
     MessageNumber,
     StatusCode,
     PTKID,
     PTK,
     MKID
     )
```

#### 6.2.5.2.1 When generated

This primitive is generated by the MLME as a result of the request or establishment of a new PTK with a specific peer MAC entity via a 4-way handshake procedure. Specifically, the primitive is generated by the MLME upon receiving a valid PTK command containing a Message Number of 1 or 3.

#### 6.2.5.2.2 Effect of receipt

The DME is notified of the request or establishment of a new PTK. The DME subsequently issues an MLME-PTK.response to reflect the actions taken.

If the MessageNumber in the MLME-PTK.indication is one, the DME verifies that the proposed PTKID is not being used by the local MAC entity for any temporal key and that establishing a new PTK using the MKID with the requesting DME is an acceptable action. The DME includes the results of the verification in the StatusCode parameter of MLME-PTK.response.

If the MessageNumber in the MLME-PTK.indication is 3 and the StatusCode in the MLME-PTK.response is zero, the DME follows to issue an MLME- KEY-UPDATE.request.

### 6.2.5.3 MLME-PTK.response

This primitive responds to the request or establishment of a new PTK with a specific peer MAC entity.

The semantics of this primitive are:

```
MLME-PTK.response(
     SrcEUI,
     MessageNumber,
     StatusCode,
     PTKID,
     MKID
     )
```

#### 6.2.5.3.1 When generated

This primitive is generated by the DME as a result of an MLME-PTK.indication reporting a valid request or establishment of a new PTK with a specific peer MAC entity via a 4-way handshake procedure.

#### 6.2.5.3.2 Effect of receipt

If the MessageNumber is 2, the MAC entity generates a new random number as the R-Nonce and transmits a PTK command containing the second message of the 4-way handshake to the specific peer MAC entity. If the MessageNumber is 4, the MAC entity uses its R-Nonce generated for the second message of the same 4-way handshake and transmits a PTK command containing the fourth message of the 4-way handshake to the specific peer MAC entity. The PTK command carries the StatusCode parameter in its Status Code field.

### 6.2.5.4 MLME-PTK.confirm

This primitive reports the results of the attempt to establish a new PTK with a specified peer MAC entity.

The semantics of this primitive are:

```
MLME-PTK.confirm(
     DestEUI,
     MessageNumber,
     StatusCode,
     PTKID,
     PTK,
     MKID,
     ResultCode
     )
```

#### 6.2.5.4.1 When generated

This primitive is generated by the MLME as a result of an MLME-PTK.request to establish a new PTK with a specified peer MAC entity. Specifically, the primitive is generated by the MLME upon receiving a valid PTK command containing a Message Number of 2 or 4; or generated in case of INVALID_PARAMETERS or PTKFailureTimeout.

#### 6.2.5.4.2 Effect of receipt

The DME is notified of the intermediate or final results of the procedure to establish a new PTK with a specified peer MAC entity.

If the MessageNumber is 4, the StatusCode is zero, and the ResultCode is RESPONSE_RECEIVED, the DME follows to issue an MLME-KEY-UPDATE.request.

## 6.2.6 GTK solicitation/distribution

The mechanism supports the procedure of soliciting a new GTK from, or distributing a new GTK to, a peer MAC entity. The primitives covered in this subclause are listed in Table 15.

**Table 15 — GTK solicitation/distribution primitives**

| Service Primitive | Request | Indication | Response | Confirm |
|---|---|---|---|---|
| MLME-GTK | 6.2.6.1 | 6.2.6.2 | 6.2.6.3 | 6.2.6.4 |

Table 16 lists the parameters that appear in the primitives of this subclause.

**Table 16 — GTK request/distribution primitive parameters**

| Name | Type | Valid range | Description |
|---|---|---|---|
| SrcEUI | EUI-48 | Any valid unicast EUI-48 | Specifies the EUI-48 of the device soliciting or distributing a new GTK |
| DestEUI | EUI-48 | Any valid unicast EUI-48 | Specifies the EUI-48 of the device requested to distribute or receive a new GTK |
| TKID | Integer | A non-zero number as specified in clause 9 | Identifies the PTK used to distribute the new GTK |
| MessageNumber | Integer | Any valid positive number as specified in clause 9 | Specifies the number of the message being conveyed in the GTK solicitation or distribution process |
| StatusCode | Integer | Any valid value as specified in clause 9 | Specifies the status of the GTK solicitation or distribution process |
| GTK | Octet string | 16 octets as specified in clause 9 | Specifies the new GTK being distributed |
| GTKID | Integer | A non-zero number as specified in clause 9 | Specifies the TKID of the new GTK being solicited or distributed |
| GroupEUI | EUI-48 | Any valid multicast or broadcast EUI-48 | Specifies the multicast group for which the GTK is being solicited or distributed, or specifies the GTK is for broadcast |
| Global | Boolean | FALSE, TRUE | If TRUE, indicates that the GTK update applies to all broadcast and multicast traffic from the device that distributed this GTK. If FALSE, indicates the update applies only to the specific GroupEUI identified. |
| GTKFailureTimeout | Integer | ≥1 | Specifies a time limit in microseconds after which the procedure of soliciting or distributing a new GTK must be terminated |
| ResultCode | Enumeration | RESPONSE_RECEIVED, INVALID_PARAMETERS, TIMEOUT | Indicates the result of the corresponding MLME-GTK.request |

### 6.2.6.1 MLME-GTK.request

This primitive requests that the MAC entity solicit a new GTK from, or distribute a new GTK to, a specified peer MAC entity as described in clause 9.

The semantics of this primitive are:

```
MLME-GTK.request(
     DestEUI,
     TKID,
     MessageNumber,
     StatusCode,
     Global,
```

```
      GTK,
      GTKID,
      GroupEUI,
      GTKFailureTimeout
      )
```

#### 6.2.6.1.1 When generated

This primitive is generated by the DME for the local MAC entity to solicit a new GTK from, or distribute a new GTK to, a specified peer MAC entity.

#### 6.2.6.1.2 Effect of receipt

The MAC entity initiates a new GTK solicitation or distribution procedure, based on the value of MessageNumber. If MessageNumber is zero, the MAC entity transmits a GTK command soliciting a new GTK from the specified peer MAC entity. If MessageNumber is one, the MAC entity distributes a new GTK to the specified peer MAC entity. The MLME subsequently issues an MLME-GTK.confirm to reflect the results.

### 6.2.6.2 MLME-GTK.indication

This primitive reports the solicitation or distribution of a new GTK by a specific peer MAC entity.

The semantics of this primitive are:

```
MLME-GTK.indication(
      SrcEUI,
      TKID,
      MessageNumber,
      StatusCode,
      Global,
      GTK,
      GTKID,
      GroupEUI
      )
```

#### 6.2.6.2.1 When generated

This primitive is generated by the MLME as a result of receiving a valid solicitation or distribution of a new GTK from a specific peer MAC entity.

#### 6.2.6.2.2 Effect of receipt

The DME is notified of the solicitation or distribution of a new GTK. If the received MessageNumber is zero, and if the DME accepts the solicitation for a new GTK, it subsequently issues an MLME-GTK.request to distribute a new GTK. If the received MessageNumber is one, the DME subsequently issues an MLME-GTK.response to reflect the actions taken with respect to the distribution of a new GTK.

If the StatusCode in the MLME-GTK.response is zero, the DME follows to issue an MLME-KEY-UPDATE.request.

### 6.2.6.3 MLME-GTK.response

This primitive responds to the solicitation or distribution of a new GTK by a specific peer MAC entity.

The semantics of this primitive are:

```
MLME-GTK.response(
      SrcEUI,
      TKID,
      MessageNumber,
```

```
        StatusCode,
        GTK,
        GTKID,
        GroupEUI
        )
```

#### 6.2.6.3.1 When generated

This primitive is generated by the DME as a result of an MLME-GTK.indication reporting a distribution of a new GTK from a specific peer MAC entity.

#### 6.2.6.3.2 Effect of receipt

The MAC entity transmits a GTK command responding to the distribution of a new GTK from the specific peer MAC entity. The GTK command carries the StatusCode parameter in its Status Code field.

### 6.2.6.4 MLME-GTK.confirm

This primitive reports the results of the attempt to solicit a new GTK from, or distribute a new GTK to, a specified peer MAC entity.

The semantics of this primitive are:

```
MLME-GTK.confirm(
     DestEUI,
     TKID,
     MessageNumber,
     StatusCode,
     GTK,
     GTKID,
     GroupEUI,
     ResultCode,
     )
```

#### 6.2.6.4.1 When generated

This primitive is generated by the MLME as a result of an MLME-GTK.request to solicit a new GTK from, or distribute a new GTK to, a specified peer MAC entity. Specifically, the primitive is generated by the MLME upon successfully transmitting a GTK command soliciting a new GTK or upon successfully receiving a GTK command responding to the distribution of a new GTK; the primitive is also generated in case of INVALID_PARAMETERS or GTKFailureTimeout.

#### 6.2.6.4.2 Effect of receipt

The DME is notified of the results of the procedure to solicit a new GTK from, or distribute a new GTK to, a specified peer MAC entity.

## 6.2.7 Temporal key update

The mechanism supports the procedure of installing or deleting either a PTK or a GTK. The primitives covered in this subclause are listed in Table 17.

**Table 17 — Temporal key update primitives**

| Service Primitive | Request | Indication | Response | Confirm |
|---|---|---|---|---|
| MLME-KEY-UPDATE | 6.2.7.1 | | | 6.2.7.2 |

Table 18 lists the parameters that appear in the primitives of this subclause.

**Table 18 — Temporal Key update primitive parameters**

| Name | Type | Valid range | Description |
|---|---|---|---|
| PeerEUI | EUI-48 | Any valid unicast EUI-48 | For a PTK, specifies the EUI-48 of the peer device sharing the PTK being updated. For a GTK, specifies the EUI-48 of the device which distributed the GTK being updated. |
| GroupEUI | EUI-48 | Any valid multicast or broadcast EUI-48, as defined in 8.1.1 | Specifies the multicast or broadcast EUI-48 to which the new GTK applies. |
| KeyType | Enumeration | PTK, GTK | Indicates whether the update is for a PTK or GTK |
| Global | Boolean | FALSE, TRUE | If TRUE, indicates that the GTK update applies to all broadcast and multicast traffic from the device that distributed this GTK. If FALSE, indicates the update applies only to the specific GroupEUI identified. |
| TKID_Old | Integer | A non-zero number as specified in clause 9, or zero | Specifies the TKID of the old PTK or GTK being replaced. If zero, indicates no old PTK or GTK is being replaced. |
| TKID | Integer | A non-zero number as specified in clause 9, or zero | Specifies the TKID of the new PTK or GTK being installed. If zero, indicates no new PTK or GTK is being installed. |
| KEY | Octet string | 16 octets as specified in clause 9 | Specifies the new PTK or GTK being installed |
| ResultCode | Enumeration | SUCCESS, INVALID_PARAMETERS | Indicates the result of the corresponding MLME-KEY-UPDATE.request |

#### 6.2.7.1 MLME-KEY-UPDATE.request

This primitive requests an update of a specified PTK or GTK.

The semantics of this primitive are:

```
MLME-KEY-UPDATE.request(
     PeerEUI,
     GroupEUI,
     KeyType,
     Global,
     TKID_Old,
     TKID,
     KEY
     )
```

##### 6.2.7.1.1 When generated

This primitive is generated by the DME for the local MAC entity to update a PTK or GTK.

##### 6.2.7.1.2 Effect of receipt

The MLME removes the keying credential identified by TKID_Old that was previously installed if TKID_Old is not zero. The MLME then installs the new PTK or GTK along with the TKID for the specified (PeerEUI, GroupEUI) if TKID is not zero. The MLME subsequently issues an MLME-KEY-UPDATE.confirm to reflect the results.

#### 6.2.7.2 MLME-KEY-UPDATE.confirm

This primitive reports the results of the attempt to update a PTK or GTK.

The semantics of this primitive are:

```
MLME-KEY-UPDATE.confirm(
     PeerEUI,
     GroupEUI,
     TKID_Old,
     TKID,
     KEY,
     ResultCode
     )
```

#### 6.2.7.2.1 When generated

This primitive is generated by the MLME as a result of an MLME-KEY-UPDATE.request to update a PTK or GTK.

#### 6.2.7.2.2 Effect of receipt

The DME is notified of the results of the procedure to update a specified PTK or GTK.

### 6.2.8 Security violation report

The mechanism supports the procedure of reporting a security violation. The primitives covered in this subclause are listed in Table 19.

**Table 19 — Security violation report primitives**

| Service Primitive | Request | Indication | Response | Confirm |
|---|---|---|---|---|
| MLME-SECURITY-VIOLATION | | 6.2.8.1 | | |

Table 20 lists the parameters that appear in the primitives of this subclause.

**Table 20 — Security violation report primitive parameters**

| Name | Type | Valid range | Description |
|---|---|---|---|
| ViolationCode | Enumeration | INVALID_MODE, INVALID_MIC, INVALID_TKID, REPLAYED_FRAME | Indicates the cause of a security violation |

#### 6.2.8.1 MLME-SECURITY-VIOLATION.indication

This primitive reports a security violation.

The semantics of this primitive are:

```
MLME-SECURITY-VIOLATION.indication(
     ViolationCode
     )
```

#### 6.2.8.1.1 When generated

This primitive is generated by the MLME as a result of encountering a security violation.

#### 6.2.8.1.2 Effect of receipt

The DME is notified of the cause of the security violation.

### 6.2.9 Pseudo-random function (PRF)

This mechanism provides higher layers through the DME with access to the MAC PRF. For simplicity, this primitive only provides 256-bit random numbers. Smaller quantities may be extracted from the result without compromising the randomness of the result. The primitives covered in this subclause are listed in Table 21.

**Table 21 — PRF primitives**

| Service Primitive | Request | Indication | Response | Confirm |
|---|---|---|---|---|
| MLME-PRF | 6.2.9.1 | | | 6.2.9.2 |

Table 22 lists the parameters that appear in the primitives of this subclause.

**Table 22 — PRF primitive parameters**

| Name | Type | Valid range | Description |
|---|---|---|---|
| Key | Octet String | A 16-octet symmetric key as specified in clause 9 | The key to be used for generating a random number |
| Nonce | Octet String | 13 octets as specified in clause 9 | The nonce to be used for generating a random number |
| DataBlocks | Octet String | 0–65535 octets | Arbitrary data to be used as input to the MAC PRF |
| DataLength | Integer | 0–65535 | Length in octets of DataBlocks |
| RandomNumber | Octet String | 32 octets | The generated 256-bit random number |
| ResultCode | Enumeration | SUCCESS, INVALID_PARAMETERS | The result of the MLME-PRF.request |

#### 6.2.9.1 MLME-PRF.request

This primitive requests generation of a 256-bit pseudo-random number using the MAC PRF.

The semantics of this primitive are:

```
MLME-PRF.request(
     Key,
     Nonce,
     DataBlocks,
     DataLength
     )
```

##### 6.2.9.1.1 When generated

This primitive is generated by the DME to request that the MAC entity generate a random number using the MAC PRF facility.

##### 6.2.9.1.2 Effect of receipt

The local MAC entity calls the PRF defined in clause 9 with the parameters contained in the primitive to generate a 256-bit random number. The semantics of the call to the PRF are:

PRF-256(Key, Nonce, “MLME_PRFrequest”, DataBlocks, DataLength)

#### 6.2.9.2 MLME-PRF.confirm

This primitive returns the result of the previously initiated MLME-PRF.request.

The semantics of this primitive are:

```
MLME-PRF.confirm(
     RandomNumber,
     ResultCode
     )
```

##### 6.2.9.2.1 When generated

This primitive is generated by the MLME as a result of an MLME-PRF.request to generate a 256-bit random number using the MAC PRF.

##### 6.2.9.2.2 Effect of receipt

The DME is notified of the results of the procedure of generating a 256-bit random number using the MAC PRF facility.

### 6.2.10 Application-specific IE (ASIE) management

The service primitives in this subclause allow the DME to add or remove an ASIE from the beacon content. The primitives covered in this subclause are listed in Table 23.

**Table 23 — ASIE management primitives**

| Service Primitive | Request | Indication | Response | Confirm |
|---|---|---|---|---|
| MLME-ASIE-ADD | 6.2.10.1 | | | 6.2.10.2 |
| MLME-ASIE-REMOVE | 6.2.10.3 | | | 6.2.10.4 |
| MLME-ASIE | | 6.2.10.5 | | |

Table 24 lists the parameters that appear in the primitives of this subclause.

**Table 24 — ASIE management primitive parameters**

| Name | Type | Valid range | Description |
|---|---|---|---|
| ASIEData | Array | Variable | A variable size array containing ASIE data |
| ASIEHandle | Integer | 0–255 | A handle associated with an ASIE that has been added to the beacon content |
| PositionAdvice | Integer | 0–255 | The recommended position of the ASIE in the beacon |
| ResultCode | Enumeration | SUCCESS, FAILURE | Indicates the result of the MLME-ASIE-ADD or MLME-ASIE-REMOVE requests |

#### 6.2.10.1 MLME-ASIE-ADD.request

This primitive changes the beacon content by adding an ASIE.

The semantics of this primitive are:

```
MLME-ASIE-ADD.request(
     ASIEHandle,
     ASIEData,
     PositionAdvice
     )
```

ASIEData is the data to be added to the beacon as an ASIE. PositionAdvice indicates the position where the new ASIE is added to the beacon. The ASIE is added after any IEs with Element ID less than or equal to PositionAdvice, and before any IEs with Element ID greater than PositionAdvice.

#### 6.2.10.1.1 When generated

This primitive is generated by the DME to instruct the MAC entity to add an ASIE to the beacon contents.

#### 6.2.10.1.2 Effect of receipt

When the MLME receives the request it initiates adding the ASIE to the beacon contents.

### 6.2.10.2 MLME-ASIE-ADD.confirm

This primitive confirms the result of an operation to add an ASIE to the beacon in response to an MLME-ASIE-ADD request.

The semantics of this primitive are:

```
MLME-ASIE-ADD.confirm(
     ASIEHandle,
     ResultCode
     )
```

ResultCode indicates the result of the attempt to add an ASIE to the beacon. If an operation to add an ASIE to the beacon is not succeeding, it is a vendor specific decision when to time out the operation and return FAILURE. If the ASIE has been successfully added to the beacon contents, ASIEHandle is returned with a handle to the ASIE. This handle can be used with the MLME-ASIE-REMOVE request to remove the ASIE from the beacon content.

#### 6.2.10.2.1 When generated

This primitive is generated by the MLME as a result of an MLME-ASIE-ADD.request with a successful return status to indicate that the first beacon has been successfully transmitted with the ASIE. If MLME-ASIE-ADD.request was called when beacons were not being transmitted, a successful status indicates that the ASIE will be in the beacon when the next beacon is sent. If the ASIE cannot be added to the beacon content the primitive is generated with a FAILURE status.

#### 6.2.10.2.2 Effect of receipt

The recipient is notified of the results of the procedure of adding an ASIE to the beacon content.

### 6.2.10.3 MLME-ASIE-REMOVE.request

This primitive instructs the MAC entity to remove an ASIE from the beacon content.

The semantics of this primitive are:

```
MLME-ASIE-REMOVE.request(
     ASIEHandle
     )
```

ASIEHandle is the handle of an ASIE that was successfully added to the beacon.

#### 6.2.10.3.1 When generated

This primitive is generated by the DME to remove an ASIE from the beacon.

#### 6.2.10.3.2 Effect of receipt

The MAC entity immediately initiates removing the ASIE indicated by ASIEHandle from the beacon content.

### 6.2.10.4 MLME-ASIE-REMOVE.confirm

This primitive confirms that the attempt to remove an ASIE from the beacon in response to an MLME-ASIE-REMOVE request has completed.

The semantics of this primitive are:

```
MLME-ASIE-REMOVE.confirm(
     ASIEHandle,
     ResultCode
     )
```

ResultCode indicates the status of the operation to remove an ASIE from the beacon content.

#### 6.2.10.4.1 When generated

This primitive is generated by the MLME as a result of an MLME-ASIE-REMOVE.request. If beacons are being transmitted a beacon must have been sent without the ASIE for a successful confirmation. If beacons are not being transmitted a successful confirmation indicates that the next beacon sent will not include the ASIE.

#### 6.2.10.4.2 Effect of receipt

The recipient is notified of the results of the procedure of removing an ASIE from the beacon.

### 6.2.10.5 MLME-ASIE.indication

This primitive indicates the reception of an ASIE from a neighbor.

The semantics of this primitive are:

```
MLME-ASIE.indication(
     ASIEData
     )
```

#### 6.2.10.5.1 When generated

This primitive is generated by the MLME when a beacon is received that contains an ASIE.

#### 6.2.10.5.2 Effect of receipt

The recipient is notified of reception of an ASIE.

## 6.2.11 Multicast address binding

The mechanism supports binding of multicast EUI-48s to McstAddrs for transmission and activating or deactivating specific multicast EUI-48s for reception. The primitives covered in this sub-clause are listed in Table 25.

**Table 25 — Multicast Address Binding primitives**

| Service Primitive | Request | Indication | Response | Confirm |
|---|---|---|---|---|
| MLME-MULTICAST-REGISTER | 6.2.11.1 | | | 6.2.11.2 |
| MLME-MULTICAST-DEREGISTER | 6.2.11.3 | | | 6.2.11.4 |
| MLME-MULTICAST-ACTIVATE | 6.2.11.5 | | | 6.2.11.6 |
| MLME-MULTICAST-DEACTIVATE | 6.2.11.7 | | | 6.2.11.8 |

Table 26 lists the parameters that appear in the primitives of this subclause.

**Table 26 — Multicast Address Binding primitive parameters**

| Name | Type | Valid range | Description |
|---|---|---|---|
| DestEUI | EUI-48 | Any valid multicast EUI-48 | Specifies the multicast group address to which the primitive applies |
| ResultCode | Enumeration | SUCCESS, FAILURE | Indicates the result of a multicast address binding request |

### 6.2.11.1 MLME-MULTICAST-REGISTER.request

This primitive binds the multicast EUI-48 DestEUI to a McstAddr.

The semantics of this primitive are:

```
MLME-MULTICAST-REGISTER.request(
     DestEUI
     )
```

#### 6.2.11.1.1 When generated

This primitive is generated by the DME for the device to bind the multicast EUI-48 specified by the parameters of the primitive to a McstAddr selected by the MAC entity. Multicast EUI-48s must be registered before being used in any MAC-DATA.request at the MAC SAP.

#### 6.2.11.1.2 Effect of receipt

The MAC entity generates a binding between DestEUI and a McstAddr and declares the binding in a Multicast Binding IE.

The MAC entity will use the bound McstAddr as the DestAddr for MSDUs passed to it via the MAC-DATA.request primitive with corresponding DestEUI. The MAC entity subsequently issues an MLME-MULTICAST-REGISTER.confirm to reflect the results.

### 6.2.11.2 MLME-MULTICAST-REGISTER.confirm

This primitive reports the result of a specified multicast address registration request.

The semantics of this primitive are:

```
MLME-MULTICAST-REGISTER.confirm(
     ResultCode
     )
```

#### 6.2.11.2.1 When generated

This primitive is generated by the MAC entity as a result of an MLME-MULTICAST-REGISTER.request to bind the multicast EUI-48 specified in that request to a McstAddr.

#### 6.2.11.2.2 Effect of receipt

The DME is notified of the results of the multicast address binding.

### 6.2.11.3 MLME-MULTICAST-DEREGISTER.request

This primitive invalidates the binding between the multicast EUI-48 DestEUI and a McstAddr.

The semantics of this primitive are:

```
MLME-MULTICAST-DEREGISTER.request(
     DestEUI
     )
```

#### 6.2.11.3.1 When generated

This primitive is generated by the DME for the device to release the binding of the multicast EUI-48 specified by the parameters of the primitive to a McstAddr.

#### 6.2.11.3.2 Effect of receipt

The MAC entity invalidates its binding between DestEUI and a McstAddr.

The MAC entity subsequently issues an MLME-MULTICAST-DEREGISTER.confirm to reflect the results.

### 6.2.11.4 MLME-MULTICAST-DEREGISTER.confirm

This primitive reports the result of a specified multicast address de-registration request.

The semantics of this primitive are:

```
MLME-MULTICAST-DEREGISTER.confirm(
     ResultCode
     )
```

#### 6.2.11.4.1 When generated

This primitive is generated by the MAC entity as a result of an MLME-MULTICAST-DEREGISTER.request to release any binding between the multicast EUI-48 specified in that request and a McstAddr.

#### 6.2.11.4.2 Effect of receipt

The DME is notified of the results of the multicast address de-registration request.

### 6.2.11.5 MLME-MULTICAST-ACTIVATE.request

This primitive activates the multicast EUI-48 for reception of multicast traffic.

The semantics of this primitive are:

```
MLME-MULTICAST-ACTIVATE.request(
     DestEUI
     )
```

#### 6.2.11.5.1 When generated

This primitive is generated by the DME for the device to activate reception of traffic on the multicast EUI-48 specified by the parameters of the primitive. A multicast address must be activated before the MAC entity will deliver any MSDUs with the multicast DestEUI to the MAC client via the MAC-DATA.indication at the MAC SAP.

#### 6.2.11.5.2 Effect of receipt

The MAC entity modifies its receive multicast filters to receive traffic addressed to a McstAddr that the source of the traffic has announced is bound to the multicast EUI-48 specified by the primitive.

The MAC entity subsequently issues an MLME-MULTICAST-ACTIVATE.confirm to reflect the results.

### 6.2.11.6 MLME-MULTICAST-ACTIVATE.confirm

This primitive reports the result of a multicast traffic activation request.

The semantics of this primitive are:

```
MLME-MULTICAST-ACTIVATE.confirm(
     ResultCode
     )
```

#### 6.2.11.6.1 When generated

This primitive is generated by the MAC entity as a result of an MLME-MULTICAST-ACTIVATE.request to activate the multicast EUI-48 specified in that request.

#### 6.2.11.6.2 Effect of receipt

The DME is notified of the results of the procedure of activating the specified multicast EUI-48.

### 6.2.11.7 MLME-MULTICAST-DEACTIVATE.request

This primitive deactivates the multicast EUI-48 for reception of multicast traffic. If the multicast EUI-48 is the NULL address (FF FF FF FF FF FF) all multicast EUI-48s are deactivated.

The semantics of this primitive are:

```
MLME-MULTICAST-DEACTIVATE.request(
     DestEUI
     )
```

#### 6.2.11.7.1 When generated

This primitive is generated by the DME for the device to deactivate reception of traffic on the multicast EUI-48 specified by the parameters of the primitive.

#### 6.2.11.7.2 Effect of receipt

The MAC entity modifies its multicast receive filters to discard MSDUs received with DestAddr set to a McstAddr that the source of the MSDU has announced is bound to the multicast EUI-48 specified by the primitive.

The MAC entity subsequently issues an MLME-MULTICAST-DEACTIVATE.confirm to reflect the results.

### 6.2.11.8 MLME-MULTICAST-DEACTIVATE.confirm

This primitive reports the result of a multicast traffic deactivation request.

The semantics of this primitive are:

```
MLME-MULTICAST-DEACTIVATE.confirm(
     ResultCode
     )
```

#### 6.2.11.8.1 When generated

This primitive is generated by the MAC entity as a result of an MLME-MULTICAST-DEACTIVATE.request to deactivate the multicast traffic specified in that request.

#### 6.2.11.8.2 Effect of receipt

The DME is notified of the results of the procedure of deactivating the specified multicast traffic.

### 6.2.12 Link events

The service primitives in this subclause are provided for the DME to monitor events related to the link status. The primitives covered in this subclause are listed in Table 27.

**Table 27 — Link event primitives**

| Service Primitive | Request | Indication | Response | Confirm |
|---|---|---|---|---|
| MLME-LINK-EVENT | 6.2.12.1 | 6.2.12.2 | | 6.2.12.3 |

Table 28 lists the parameters that appear in the primitives of this subclause.

**Table 28 — Link event primitive parameters**

| Name | Type | Valid range | Description |
|---|---|---|---|
| AccessMethod | Enumeration | PCA,<br>DRP | The access method used for transmission or receipt of a frame |
| Beacon | Enumeration | FALSE,<br>TRUE | Indicates whether a received frame was a beacon |
| BPSTOffset | Integer | 0–65535 | The offset of the start of a received frame relative to the BPST of the device, measured in microseconds |
| RemoteEUI | EUI-48 | Any valid unicast EUI-48 | Specifies the EUI-48 of the remote device in the link |
| LinkEventType | Enumeration | RECEIVE_SUCCESS,<br>RECEIVE_ERROR,<br>TRANSMIT_SUCCESS,<br>TRANSMIT_ERROR | The type of link event that occurred on a link being monitored |
| MonitorState | Enumeration | DISABLED,<br>ENABLED | Specifies whether link event observation is active for the specified link |
| PayloadSize | Integer | 0 to pMaxPayloadSize | Size of a transmitted or received frame payload |
| PHYRate | Enumeration | RATE_53_3,<br>RATE_80,<br>RATE_106_7,<br>RATE_160,<br>RATE_200,<br>RATE_320,<br>RATE_400,<br>RATE_480 | PHY data rate at which a frame was transmitted or received |
| ResultCode | Enumeration | SUCCESS,<br>FAILURE | Appears in MLME-LINK-EVENT.confirm and indicates the result of the corresponding MLME-LINK-EVENT request |
| RetryCount | Integer | Variable | Retry count for this transmission |
| ReceiveErrorInfo | Enumeration | PAYLOAD_ERROR,<br>UNSUPPORTED_RATE_ERROR,<br>GENERAL_ERROR | Provides additional information for an RECEIVE_ERROR |
| DeliveryID | Integer | As defined in 7.2.1.5 | The Delivery ID associated with a transmitted or received frame |

#### 6.2.12.1 MLME-LINK-EVENT.request

This primitive enables or disables the observation of link events for a specified link.

The semantics of this primitive are:

```
MLME-LINK-EVENT.request(
     RemoteEUI,
     MonitorState
     )
```

RemoteEUI is the EUI-48 for the remote device for which the link monitoring state is to be changed. MonitorState indicates whether monitoring link events is to be enabled or disabled for the specified link. Links are not monitored by default.

#### 6.2.12.1.1 When generated

This primitive is generated by the DME to enable or disable link monitoring for the specified link.

#### 6.2.12.1.2 Effect of receipt

The MLME initiates enabling or disabling observation of link events for the specified link.

### 6.2.12.2 MLME-LINK-EVENT.indication

This indication informs the DME that an event occurred for a link with monitoring enabled.

The semantics of this primitive are:

```
MLME-LINK-EVENT.indication(
     AccessMethod,
     Beacon,
     BPSTOffset,
     LinkEventType,
     PayloadSize,
     PHYRate,
     RemoteEUI,
     RetryCount,
     ReceiveErrorInfo,
     DeliveryID
     )
```

AccessMethod indicates if the frame associated with this event occurred during a DRP reservation or during PCA access time. Beacon indicates whether the event is associated with the receipt of a beacon. AccessMethod is undefined if Beacon is TRUE. RemoteEUI is the EUI-48 of the remote device associated with the link event. LinkEventType indicates the type of event that has occurred for the link that is being monitored. The possible link event types and their meaning for frames that are received in different ways are summarized and described in Table 29.

**Table 29 — Link event types**

| | Beacon | DRP / PCA No-ACK | DRP / PCA IMM-ACK, B-ACK |
|---|---|---|---|
| **RECEIVE_SUCCESS** | Beacon from RemoteEUI received correctly | Frame received correctly from RemoteEUI. | Frame received correctly from RemoteEUI |
| **RECEIVE_ERROR** | Beacon from RemoteEUI received with RECEIVE_ERROR | Frame received from RemoteEUI with RECEIVE_ERROR. | Frame received from RemoteEUI with RECEIVE_ERROR |
| **TRANSMIT_SUCCESS** | N/A | Frame transmitted to RemoteEUI. | Frame transmitted to RemoteEUI and acknowledgement received |
| **TRANSMIT_ERROR** | N/A | N/A | Frame transmitted to RemoteEUI and no acknowledgement received |

PayloadSize is the size of the frame associated with the link event (not including the FCS). PHYRate is the rate at which the frame associated with the link event was transmitted or received. ReceiveErrorInfo provides additional information if the even type is an RECEIVE_ERROR — it is undefined for other event types. RetryCount is number of times the transmission was attempted. DeliveryID is the Delivery ID associated with the frame that produced the link event if the event occurred during a reservation.

#### 6.2.12.2.1 When generated

This indication is generated by the MLME when it determines a link event has occurred for a link that is being monitored.

#### 6.2.12.2.2 Effect of receipt

The recipient is notified that an event has occurred for a link that is being monitored.

### 6.2.12.3 MLME-LINK-EVENT.confirm

This confirmation indicates that a request to enable or disable link monitoring initiated by the MLME-LINK-EVENT request has been completed.

The semantics of this primitive are:

```
MLME-LINK-EVENT.confirm(
     ResultCode
     )
```

ResultCode indicates whether the operation to change whether a link is monitored has successfully completed. If changing the state is not succeeding, it is a vendor specific decision when to time out the operation and return FAILURE.

#### 6.2.12.3.1 When generated

This primitive is generated by the MLME as a result of an MLME-LINK-EVENT.request once the requested state has been entered or the operation has failed.

#### 6.2.12.3.2 Effect of receipt

The recipient is notified of the results of the procedure of changing whether events are monitored for the specified link.

### 6.2.13 Probe

The service primitives in this subclause are provided for the DME to request Information Elements from another device. The primitives covered in this subclause are listed in Table 30.

**Table 30 — Probe primitives**

| Service Primitive | Request | Indication | Response | Confirm |
|---|---|---|---|---|
| MLME-PROBE | 6.2.13.1 | 6.2.13.2 | 6.2.13.3 | 6.2.13.4 |

Table 31 lists the parameters that appear in the primitives of this subclause.

**Table 31 — Probe primitive parameters**

| Name | Type | Valid range | Description |
|---|---|---|---|
| Explicit | Boolean | FALSE, TRUE | Controls whether a probe request is implicit or explicit |
| DestEUI | EUI-48 | Any valid EUI-48 | Specifies the EUI-48 of the target of the probe request |
| IEElementID | Integer | 0–255 | The element ID of an IE being requested |
| ASIESpecifierID | Integer | 0–65535 | a 16-bit value that identifies a company or organization, as listed in *WiMedia Assigned Numbers* [B5], that is responsible for the definition of the ASIE |
| ASIERequestInformation | Array | Variable | Specifies the contents of the Application-specific Request Information field to include in the Application-specific Probe IE. |
| IEInfo | Array | Variable | A variable size array containing all the data for an IE sent or received via the probe mechanism |
| ResultCode | Enumeration | SUCCESS, TIMEOUT | Indicates the result of the corresponding request |
| SrcEUI | EUI-48 | Any valid unicast EUI-48 | Specifies the EUI-48 of the source of a received probe request |
| RequestTimeout | Duration | 0–65535 | The time, in milliseconds, allotted for the completion of an MLME request. If this time elapses while the request is pending, it is terminated with a ResultCode of TIMEOUT. |

#### 6.2.13.1 MLME-PROBE.request

This primitive begins a request to another device for an IE.

The semantics of this primitive are:

```
MLME-PROBE.request(
     Explicit,
     DestEUI,
     IEElementID,
     ASIESpecifierID,
     RequestTimeout
     )
```

Explicit indicates whether the probe request is to be performed through beacons or using command frames. Explicit probe requests are performed using command frames, while implicit probe requests are made through beacons. DestEUI is the device ID of the target device for the probe request. The IEElementID is the element ID for the Information Element that is being requested through the probe request. If IEElementID specifies an ASIE, then ASIESpecifierID and ASIERequestInformation must be provided. ASIESpecifierID identifies the organizational definer of the ASIE, and ASIERequestInformation specifies the organization-specific information to include in the request. RequestTimeout is the maximum time a device should wait to complete the probe request, and should account for request and response delays up to mMaxLostBeacons superframes if the request is implicit.

The definition of the request primitive to request a single IE does not prohibit an implementation from requesting multiple IEs in a single request.

#### 6.2.13.1.1 When generated

This primitive is generated by the DME to initiate a probe request. A probe request is only sent to a device that indicates it supports probe requests in its Capabilities IE.

#### 6.2.13.1.2 Effect of receipt

The MLME initiates the probe request through the specified mechanism. If implicit, the request must be attempted in the next beacon transmitted. If the mechanism is explicit, the MAC entity issues the probe request using command frames, using an immediate acknowledgement policy.

### 6.2.13.2 MLME-PROBE.indication

This indication informs the DME that a probe request was received from another device.

The semantics of this primitive are:

```
MLME-PROBE.indication(
     IEElementID,
     ASIESpecifierID
     Explicit
     SrcEUI
     )
```

SrcEUI is the device ID of the device that sent the probe request. The IEElementID is the element ID for the Information Element that is being requested through the probe request.

#### 6.2.13.2.1 When generated

This indication is generated by the MLME when it receives a probe request.

#### 6.2.13.2.2 Effect of receipt

The recipient is notified that a probe request was received.

### 6.2.13.3 MLME-PROBE.response

In response to a probe request, this primitive causes the requested Information Element to be transmitted to the requestor.

The semantics of this primitive are:

```
MLME-PROBE.response(
     Explicit,
     DestEUI,
     IEInfo,
     TransmissionTimeout
     )
```

Explicit indicates whether the probe request is to be performed through beacons or using command frames. The use of Explicit must be set to the same value as what was delivered in the corresponding probe request. DestEUI is the device ID of the target device for the probe request. The IEInfo array contains the full IE to be transmitted. Transmission timeout is the timeout for the probe response if explicit access is used. Transmission timeout is not used if the request is implicit.

#### 6.2.13.3.1 When generated

This primitive is generated by the DME in response to receipt of a Probe Request IE, whether received in the beacon or signaled by MLME-PROBE.indication.

#### 6.2.13.3.2 Effect of receipt

The MLME initiates the probe response through the specified mechanism. If mechanism is through the beacon, the response is attempted in the next beacon transmitted. If the mechanism is explicit, the MLME issues the probe response using an immediate acknowledgement policy.

### 6.2.13.4 MLME-PROBE.confirm

This indication informs the DME that a probe response was received from another device.

The semantics of this primitive are:

```
MLME-PROBE.confirm(
     IEInfo,
     SrcEUI,
     ResultCode
     )
```

SrcEUI is the device ID of the device that sent the probe response. The IEInfo is the information element data provided in the probe response. ResultCode indicates SUCCESS if the confirmation is received within the RequestTimeout period, or TIMEOUT if RequestTimeout expired before receiving the confirmation.

#### 6.2.13.4.1 When generated

This indication is generated by the MLME when it receives a probe response through PCA traffic. A probe response through a beacon is received by the DME through a MLME-BEACON.indication.

#### 6.2.13.4.2 Effect of receipt

The recipient is notified that a probe response was received.

### 6.2.14 Hibernation and sleep cycle

The service primitives in this subclause are provided for the DME to control hibernation, hibernation anchor support, and sleep cycles during a superframe. The primitives covered in this subclause are listed in Table 32.

**Table 32 — Hibernation primitives**

| Service Primitive | Request | Indication | Response | Confirm |
|---|---|---|---|---|
| MLME-HIBERNATE | 6.2.14.1 | | | 6.2.14.2 |
| MLME-WAKE | 6.2.14.3 | 6.2.14.4 | | 6.2.14.5 |
| MLME-HIBERNATION-ANCHOR | 6.2.14.6 | 6.2.14.7 | | 6.2.14.8 |
| MLME-SLEEP-SCHEDULE | 6.2.14.9 | | | 6.2.14.10 |

Table 33 lists the parameters that appear in the primitives of this subclause.

**Table 33 — Hibernation primitive parameters**

| Name | Type | Valid range | Description |
|---|---|---|---|
| Start | Integer | 0-N | The number of superframes before the device will enter hibernation mode. N is implementation-specific. |
| InitialCountdown | Integer | 1–255 | The number of superframes a device will indicate its intent to hibernate before hibernation begins |
| HibernationDuration | Integer | 1–255 | The number of superframes the device intends to hibernate |
| Repetitive | Boolean | FALSE, TRUE | TRUE if the device will enter hibernate mode periodically based on the HibernationDuration and WakeDuration parameters |
| WakeDuration | Integer | 1–N | The number of superframes that a device will be in active mode after leaving hibernation mode, if Repetitive is used. N is implementation-specific. |
| AnchorOperation | Enumeration | ACTIVATE, DEACTIVATE | Specifies whether device will start or stop operating as a hibernation anchor |
| OnInNeighborHardReservations | Boolean | FALSE, TRUE | Indicates if a device in ON_PER_LOAD sleep mode should be on during neighbors' hard reservations |
| OnInNeighborSoftReservations | Boolean | FALSE, TRUE | Indicates if a device in ON_PER_LOAD sleep mode should be on during neighbors' soft reservations |
| OnForPCA | Boolean | FALSE, TRUE | Indicates if a device in ON_PER_LOAD sleep mode should be on during MASs available for PCA |
| PCAMASCount | Integer | 0–255 | Indicates the maximum number of MASs per superframe that the device should be on for PCA |
| ResultCode | Enumeration | SUCCESS, FAILURE, FAILURE_NO_SLOTS | Indicates the result of the corresponding MLME-HIBERNATE or MLME-WAKE request |
| SleepMode | Enumeration | ALWAYS_ON, ON_PER_SCHEDULE, ON_PER_LOAD | Indicates the type of sleep operation a device should perform |
| SleepSchedule | Bitmap | one bit per MAS | For a device in ON_PER_SCHEDULE sleep mode, each bit indicates if the device should sleep or receive traffic during the corresponding MAS |

#### 6.2.14.1 MLME-HIBERNATE.request

This primitive instructs the MAC entity to begin the hibernation process. During hibernation the MAC entity ceases all operations on the medium including transmission of beacons.

The semantics of this primitive are:

```
MLME-HIBERNATE.request(
     Start,
     InitialCountdown,
     HibernationDuration
     Repetitive,
     WakeDuration
     )
```

Before hibernation begins the MAC entity adds the Hibernation Mode IE to its beacon. InitialCountdown indicates how many superframes the Hibernation Mode IE will be included in the beacon before hibernation begins. HibernationDuration indicates the number of superframes the hibernation is intended to last. HibernationDuration is used in the Hibernation Mode IE which includes the number of superframes the device intends to hibernate.

##### 6.2.14.1.1 When generated

This primitive is generated by the DME to instruct the MAC entity to begin the hibernation process. This request may be made at any time after the MLME-WAKE.indication is received if another hibernation request was previously active.

##### 6.2.14.1.2 Effect of receipt

When the MLME receives the request it initiates the hibernation process.

#### 6.2.14.2 MLME-HIBERNATE.confirm

This primitive confirms the result of the hibernation operation started by the MLME-HIBERNATE.request.

The semantics of this primitive are:

```
MLME-HIBERNATE.confirm(
     ResultCode
     )
```

ResultCode indicates the result of the attempt to begin the hibernation process. If an operation to start the hibernation process is not succeeding, it is a vendor specific decision when to time out the operation and return FAILURE. A return code of SUCCESS indicates that a beacon has been sent containing the Hibernation Mode IE.

##### 6.2.14.2.1 When generated

This primitive is generated by the MLME as a result of an MLME-HIBERNATE.request to indicate that the first beacon has been transmitted with the Hibernation Mode IE or that the operation has failed.

##### 6.2.14.2.2 Effect of receipt

The recipient is notified of the results of the procedure of starting the hibernation process.

#### 6.2.14.3 MLME-WAKE.request

This primitive instructs the MAC entity to immediately exit hibernation mode and resume transmission of beacons as soon as possible, regardless of a previously-scheduled hibernation period length. This primitive also cancels any previous hibernation mode request and instructs the MAC entity to remain in active mode until further notice.

The semantics of this primitive are:

```
MLME-WAKE.request(
     )
```

#### 6.2.14.3.1 When generated

This primitive is generated by the DME to end the hibernation process and resume transmission of beacons.

#### 6.2.14.3.2 Effect of receipt

The MAC entity immediately ends hibernation and resumes transmission of beacons. If this primitive is used before the MAC entity has begun hibernation (while it is still transmitting beacons with the Hibernation Mode IE) the MAC entity attempts to remove the Hibernation Mode IE from the beacon and does not continue the hibernation process.

### 6.2.14.4 MLME-WAKE.indication

This indication informs the DME that the MAC entity has begun the process of exiting hibernation.

The semantics of this primitive are:

```
MLME-WAKE.indication(
     )
```

#### 6.2.14.4.1 When generated

This indication is generated by the MLME when it begins to scan the channel in preparation for resuming transmission of beacons after a hibernation period. This indication is generated even if an MLME-WAKE request was not issued.

#### 6.2.14.4.2 Effect of receipt

The recipient is notified that the MAC entity is scanning the channel and is preparing to resume transmission of beacons. The DME may make another MLME-HIBERNATE request at any time after receiving a MLME-WAKE indication.

### 6.2.14.5 MLME-WAKE.confirm

This primitive confirms that the attempt to end hibernation in response to an MLME-WAKE request has completed. Upon successful completion of this request the device will have resumed transmission of beacons.

The semantics of this primitive are:

```
MLME-WAKE.confirm(
     ResultCode
     )
```

ResultCode indicates whether transmission of beacons has successfully resumed. A BP must have passed with a beacon being transmitted for the confirmation to complete successfully. If ending hibernation is not succeeding, it is a vendor specific decision when to time out the operation and return FAILURE.

#### 6.2.14.5.1 When generated

This primitive is generated by the MLME as a result of an MLME-WAKE.request. The operation is successfully completed once a beacon has been transmitted. If the wake request occurred before the last beacon with the Hibernation Mode IE was transmitted, the operation is completed successfully once a beacon is transmitted without the Hibernation Mode IE. If transmission of beacons cannot resume, a confirmation is generated with a status indicating FAILURE.

#### 6.2.14.5.2 Effect of receipt

The recipient is notified of the results of the procedure of exiting hibernation.

### 6.2.14.6 MLME-HIBERNATION-ANCHOR.request

This enables the transmission of the Hibernation Anchor IE in the beacon.

The semantics of this primitive are:

```
MLME-HIBERNATION-ANCHOR.request(
     AnchorOperation
     )
```

#### 6.2.14.6.1 When generated

This primitive is generated by the DME to enable support for the Hibernation Anchor IE.

#### 6.2.14.6.2 Effect of receipt

The MAC entity indicates support for acting as a hibernation anchor in its Capabilities IE and includes a Hibernation Anchor IE in its beacon after receiving a Hibernation IE from any neighbor.

### 6.2.14.7 MLME-HIBERNATION-ANCHOR.indication

This indication informs the DME of reception of a Hibernation Anchor IE.

The semantics of this primitive are:

```
MLME-HIBERNATION-ANCHOR.indication(
     )
```

#### 6.2.14.7.1 When generated

This primitive is generated by the MLME when a Hibernation Anchor IE is received in the beacon.

#### 6.2.14.7.2 Effect of receipt

The recipient is notified of the reception of a Hibernation Anchor IE.

### 6.2.14.8 MLME-HIBERNATION-ANCHOR.confirm

This primitive indicates the result of an MLME-HIBERNATION-ANCHOR.request.

The semantics of this primitive are:

```
MLME-HIBERNATION-ANCHOR.confirm(
     ResultCode
     )
```

ResultCode indicates the result of the request to act as a hibernation anchor. Possible codes are SUCCESS and FAILURE.

#### 6.2.14.8.1 When generated

This primitive is generated by the MLME in response to a MLME-HIBERNATION-ANCHOR.request.

#### 6.2.14.8.2 Effect of receipt

The recipient is notified of the completion of the request to act or cease acting as a hibernation anchor.

#### 6.2.14.9 MLME-SLEEP-SCHEDULE.request

This request informs the device in what MASs it should be available to receive traffic and when it can turn off its receiver.

The semantics of this primitive are:

```
MLME-SLEEP-SCHEDULE.request(
     SleepMode,
     OnInHardReservations,
     OnInSoftReservations,
     OnForPCA,
     PCAMASCount,
     SleepSchedule
     )
```

SleepSchedule specifies the MASs during which the device should be capable of receiving a transmitted frame.

##### 6.2.14.9.1 When generated

This primitive is generated by the DME to control reception in the MAC sublayer.

##### 6.2.14.9.2 Effect of receipt

When the MLME receives this request, it updates its sleep schedule and PCA Availability IE.

#### 6.2.14.10 MLME-SLEEP-SCHEDULE.confirm

This primitive indicates the result of an MLME-SLEEP-SCHEDULE.request.

The semantics of this primitive are:

```
MLME-SLEEP-SCHEDULE.confirm(
     ResultCode
     )
```

ResultCode indicates the result of the request to change the sleep schedule.

##### 6.2.14.10.1 When generated

This primitive is generated by the MLME as a result of an MLME-SLEEP-SCHEDULE.request.

##### 6.2.14.10.2 Effect of receipt

The recipient is notified of the results of the request.

### 6.2.15 Higher layer synchronization support

This mechanism supports the process of synchronization among higher-layer protocol entities residing within different devices. The actual synchronization mechanism in the higher layer is out of the scope of this standard. In principle, the MLME indicates the transmission/reception of frames with a specific multicast address in the DestAddr field of an MSDU of type data. The primitives covered in this subclause are listed in Table 34.

**Table 34 — Higher layer synchronization support primitives**

| Service Primitive | Request | Indication | Response | Confirm |
|---|---|---|---|---|
| MLME-HL-SYNC | 6.2.15.1 | 6.2.15.2 | | 6.2.15.3 |

Table 35 lists the parameters that appear in the primitives of this subclause.

**Table 35 — Higher layer synchronization primitive parameters**

| Name | Type | Valid range | Description |
|---|---|---|---|
| GroupEUI | EUI-48 | Any valid multicast EUI-48 | Specifies the multicast group to which the synchronization frames are addressed. A synchronization frame is a data type frame with higher layer synchronization information. |
| ResultCode | Enumeration | SUCCESS, NOT_SUPPORTED | Indicates the result of the MLME-HL-SYNC.request |
| SrcEUI | EUI-48 | Any valid unicast EUI-48 | Specifies the EUI-48 of the device that transmitted the higher layer synchronization frame. |
| SequenceNumber | Integer | As defined in the frame format | Specifies the Sequence Number of the higher layer synchronization frame received or transmitted |

#### 6.2.15.1 MLME-HL-SYNC.request

This primitive requests activation of the synchronization support mechanism.

The semantics of this primitive are:

```
MLME-HL-SYNC.request(
     GroupEUI
     )
```

##### 6.2.15.1.1 When generated

This primitive is generated by the DME when a higher layer protocol initiates a synchronization process.

##### 6.2.15.1.2 Effect of Receipt

This request activates the synchronization support mechanism at the device. The MLME subsequently issues an MLME-HL-SYNC.confirm that reflects the results of the higher layer synchronization support request. If the request has been successful, and the higher layer synchronization support mechanism has been activated, the MLME issues an MMLE-HL-SYNC.indication whenever a higher layer synchronization frame, which is a data type frame with the specified McstAddr in the DestAddr field, is received or transmitted.

#### 6.2.15.2 MLME-HL-SYNC.indication

This primitive indicates the transmission or reception of a higher layer synchronization frame. The indication is delivered with respect to the start of the frame on the medium, whether transmitted or received by the MAC entity.

The semantics of this primitive are:

```
MLME-HL-SYNC.indication(
     GroupEUI,
     SrcEUI,
     SequenceNumber
     )
```

##### 6.2.15.2.1 When generated

This primitive is generated by the MLME when the successful reception or transmission of a higher layer synchronization frame is detected, as indicated by the PHY. The higher layer synchronization

frame is identified by the McstAddr registered by an earlier MLME-HL-SYNC.request primitive, in the DestAddr field of a data type frame.

#### 6.2.15.2.2 Effect of Receipt

The DME is notified of the reception or transmission of a higher layer synchronization frame.

### 6.2.15.3 MLME-HL-SYNC.confirm

This primitive confirms the activation of the higher layer synchronization support mechanism.

The semantics of this primitive are:

```
MLME-HL-SYNC.confirm(
     ResultCode
     )
```

#### 6.2.15.3.1 When generated

This primitive is generated by the MLME as a result of an MLME-HL-SYNC.request to activate the higher layer synchronization support mechanism for a particular multicast address.

#### 6.2.15.3.2 Effect of Receipt

The DME is notified of the activation of the higher layer synchronization support mechanism. The result code of NOT_SUPPORTED is issued if the MLME does not support the higher layer synchronization support mechanism or if the address provided by the MLME-HL-SYNC.request is not a multicast address.

## 6.2.16 Reservation management

The DRP provides methods for devices to establish, modify and release reservations. Reservation negotiations are requested by the DME and confirmed by the MLME via the service primitives defined in this subclause. There are two conceptual interface options between the DME and MLME defined in this subclause. The MLME-RESOURCE primitives provide a high-level interface where resource allocation is handled within the MLME. The MLME-DRP primitives provide a low-level interface where resource allocation is handled in the DME using information obtained from the MLME using the MLME-LINK-EVENT primitives. Subclause 6.2.16.1 illustrates a reference model and indicates the interface locations. Table 36 summarizes the reservation management service primitives.

**Table 36 — DRP service primitives**

| Service primitive | Request | Indication | Response | Confirm |
|---|---|---|---|---|
| MLME-RESOURCE | 6.2.16.2 | 6.2.16.3 | 6.2.16.4 | 6.2.16.5 |
| MLME-DRP | 6.2.16.6 | 6.2.16.7 | 6.2.16.8 | 6.2.16.9 |

Table 37 defines the parameters used by the DRP service primitives.

**Table 37 — DRP service primitive parameters**

| Name | Type | Valid range | Description |
|---|---|---|---|
| AvailabilityBitmap | Implementation-dependent | | Specifies the MASs available for DRP reservation |
| DestEUI | EUI-48 | Any valid EUI-48, or NULL for PCA reservations | Identifies the respondent (either a single device or a multicast group) in the DRP negotiation initiated by the device identified by SrcEUI |
| Explicit | Boolean | FALSE, TRUE | Controls whether DRP negotiation is implicit or explicit |
| FinalReservation | Boolean | FALSE, TRUE | Valid for multicast reservations, only. When TRUE, the final reservation is signaled to the multicast recipients |
| MinBW | Integer | 0–480000 | Minimum required bandwidth for the reservation, in Kbps. |
| DesiredBW | Integer | 0–480000 | Desired bandwidth for the reservation, in Kbps. Shall not be lower than the MinBW parameter. |
| AvailableBW | Integer | 0–480000 | Bandwidth estimated to be available for the reservation, in Kbps. For the initiator of the reservation shall not exceed the DesiredBW parameter and shall not be below MinBW.[5] |
| ReasonCode | Enumeration | ACCEPTED, CONFLICT, PENDING, DENIED | Additional completion status information for the MLME request |
| MaxServiceInterval | Integer | 1–255 | Maximum service interval acceptable for the reservation, in units of MASs. |
| QoSGoal | Enumeration | PREMIUM, HIGH, BEST_EFFORT | The Quality of Service goal of the connection to be served by the reservation. May be mapped to target Packet Error Rate (PER) and margins (SNR, reservation time) for the connection. |
| ReservationBitmap | Implementation-dependent | | Specifies the MASs desired or obtained for the reservation |
| ReservationType | Enumeration | HARD, SOFT, PRIVATE, or PCA as specified by Table 59 | Reservation type |
| ResultCode | Enumeration | FAILURE, SUCCESS, TIMEOUT, MODIFIED | Completion status of the MLME request |
| SrcEUI | EUI-48 | Any valid unicast EUI-48 | Identifies the DRP negotiation initiator |
| StreamIndex | Integer | 0–7, as specified in 7.2.1.5 | Identifies a stream from the device identified by SrcEUI to the device(s) identified by DestEUI |

[5] Note that for the recipient of the reservation request the estimation of the available bandwidth may differ from the one at the initiator side due to asymmetric link conditions.

Although the actual format of AvailabilityBitmap and ReservationBitmap is implementation-dependent, conceptually it is an array of 256 entries, each of which corresponds to one of the 256 MASs within a superframe. The zero entries identify MASs that are to be excluded from the reservation while nonzero entries identify MASs that are to be included.

Within a device's beacon group, the reservation parameters DestEUI, SrcEUI, StreamIndex, and ReservationType uniquely identify a reservation.

#### 6.2.16.1 Resource allocation and rate adaptation reference model

Figure 5 shows a reference model for the resource allocation and rate adaptation architecture. Requirements for resources coming from an application are expressed in terms of required bandwidth and quality of service parameters. Depending on the specific implementation of higher layer functions, specific functionality may be allocated either in the MAC sublayer or in the DME.

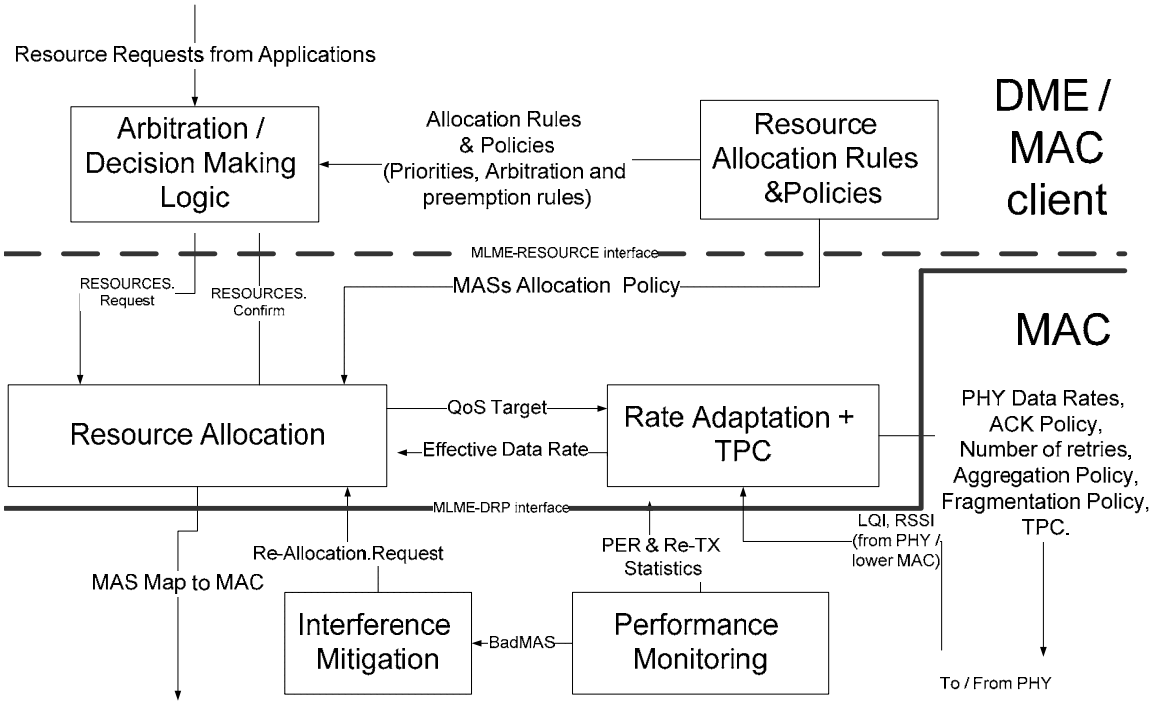



**Figure 5 — Resource allocation and rate adaptation reference model**

#### 6.2.16.2 MLME-RESOURCE.request

This primitive requests the creation of a new reservation or the modification or release of an existing reservation. The primitive's semantics are as follows:

```
MLME-RESOURCE.request(
     DestEUI,
     StreamIndex,
     ReservationType,
     MinBW,
     DesiredBW,
     MaxServiceInterval,
     QOSGoal,
     Explicit
     )
```

The MinBW and DesiredBW parameters define the BW requirements for the reservation. If these parameters are set to zero, this indicates the release of the reservation.

#### 6.2.16.2.1 When generated

The DME signals this primitive to the MLME in order to create a new reservation, modify an existing reservation or release an existing reservation.

#### 6.2.16.2.2 Effect of receipt

Based on the parameters of the resources request primitive and the condition of the link with the respondent of the reservation, the MLME constructs one or more DRP IEs that provide the desired new reservation or changes to the reservation and either a) includes the DRP IEs in a subsequent beacon transmission or b) transmits the DRP IEs in a command frame to the device identified by DestEUI. Once DRP negotiation, either implicit or explicit, respectively, is initiated, the MLME completes it as specified in 8.4.

If the source is not able to support the requested resource reservation with the parameters included in the MLME-DRP.request, the MLME responds with an MLME-RESOURCE.confirm with ResultCode set to FAILURE and does not initiate the DRP negotiation process.

### 6.2.16.3 MLME-RESOURCE.indication

This primitive indicates a request from a peer DME to create a new reservation or to modify or release an existing reservation. The primitive's semantics are as follows:

```
MLME-RESOURCE.indication(
     DestEUI,
     SrcEUI,
     StreamIndex,
     ReservationType,
     AvailableBW,
     ReservationBitmap,
     Explicit
     )
```

The ReservationBitmap identifies the MASs proposed by the reservation owner to be included in the new or modified reservation. If no MASs are identified in the ReservationBitmap, the reservation is released.

#### 6.2.16.3.1 When generated

The MLME generates this primitive either when a DRP request to create a new reservation or modify an existing reservation is received, or when changes in PHY channel conditions cause a decrease in AvailableBW. Note that the MLME also generates this primitive when a reservation is released.

#### 6.2.16.3.2 Effect of receipt

The DME evaluates the reservation request in terms of the device’s availability and generates an MLME-RESOURCE.response.

### 6.2.16.4 MLME-RESOURCE.response

The DME uses this primitive to respond to a request for the creation of a new reservation or the modification or release of an existing reservation. The primitive's semantics are as follows:

```
MLME-RESOURCE.response(
     DestEUI,
     SrcEUI,
     StreamIndex,
     ReasonCode,
     ResultCode
     )
```

#### 6.2.16.4.1 When generated

The DME generates this primitive in order to trigger a response to a reservation request.

#### 6.2.16.4.2 Effect of receipt

The MLME constructs one or more DRP IEs that describe the DRP reservation response and either a) includes the DRP IEs in a subsequent beacon transmission or b) transmits the DRP IEs in a command frame to the device identified by DestEUI.

### 6.2.16.5 MLME-RESOURCE.confirm

This primitive signals the completion, successfully or in error, of DRP negotiations initiated by a corresponding MLME-RESOURCE.request. The primitive's semantics are as follows:

```
MLME-RESOURCE.confirm(
     AvailableBW,
     ResultCode,
     ReasonCode
     )
```

#### 6.2.16.5.1 When generated

The MLME signals this primitive to the DME in order to confirm the success or failure of DRP negotiations initiated by a corresponding MLME-RESOURCE.request, or to indicate to the DME that there is a change in available bandwidth (AvailableBW).

For explicit unicast negotiations, if a DRP reservation response command frame is not received within an application-specific timeout after the DRP reservation request is sent, the MLME informs the DME by issuing a MLME-RESOURCE.confirm with ResultCode equal to TIMEOUT. Once a DRP reservation response command frame is received, the source informs the DME by generating an MLME-RESOURCE.confirm with the appropriate ResultCode.

For implicit unicast negotiations, if a DRP reservation response is not received within a time interval equal to mMaxLostBeacons after the final DRP reservation request is sent, the MLME informs the DME by issuing an MLME-RESOURCE.confirm with ResultCode equal to TIMEOUT.

Once a DRP reservation response is received, the source informs the DME by generating an MLME-RESOURCE.confirm with the appropriate ResultCode.

If the PHY channel conditions change in a way that modifies the effective bandwidth available for this reservation (AvailableBW), the MLME will also issue the MLME-RESOURCE.confirm with ResultCode equal to MODIFIED, and the new estimated AvailableBW parameter.

#### 6.2.16.5.2 Effect of receipt

Depending on the value of ResultCode and ReasonCode, the DME should take appropriate action. For example, if the reservation was established successfully, the DME may signal the MAC client to transfer data to the MAC for that stream. If the DRP negotiation failed, or the available bandwidth has changed, the DME should signal the lack of resources to the MAC client.

### 6.2.16.6 MLME-DRP.request

This primitive requests the creation of a new reservation or the modification or release of an existing reservation. The primitive's semantics are as follows:

```
MLME-DRP.request(
     DestEUI,
     StreamIndex,
     ReservationType,
     ReservationBitmap,
     FinalReservation,
```

```
        Explicit
        )
```

The ReservationBitmap parameter defines the MASs to include in a new or modified reservation. If no MASs are identified in the ReservationBitmap, that indicates the reservation is released.

#### 6.2.16.6.1 When generated

This DME signals this primitive to the MLME in order to create a new reservation, modify an existing reservation or release an existing reservation.

#### 6.2.16.6.2 Effect of receipt

The MLME constructs one or more DRP IEs that describe the desired new reservation or changes to the reservation and either a) includes the DRP IEs in a subsequent beacon transmission or b) transmits the DRP IEs in a command frame to the device identified by DestEUI. Once DRP negotiation, either implicit or explicit, respectively, is initiated, the MLME completes it as specified in 8.4.

In order to negotiate a reservation, the MAC entity constructs DRP IEs according to the parameters of the MLME-DRP.request provided by the DME. If the source is not able to support the requested DRP reservation with the parameters included in the MLME-DRP.request, the MLME responds with an MLME-DRP.confirm with ResultCode set to FAILURE and does not initiate the DRP negotiation process.

### 6.2.16.7 MLME-DRP.indication

This primitive indicates a request from a peer DME to create a new reservation or to modify or release an existing reservation. The primitive's semantics are as follows:

```
MLME-DRP.indication(
     DestEUI,
     SrcEUI,
     StreamIndex,
     ReservationType,
     ReservationBitmap,
     Explicit
     )
```

The ReservationBitmap identifies the MASs proposed by the reservation owner to be included in the new or modified reservation. If no MASs are identified in the ReservationBitmap, the reservation is released.

#### 6.2.16.7.1 When generated

The MLME generates this primitive when a DRP request to create a new reservation or modify an existing reservation is received. It also generates this primitive when a reservation is released.

#### 6.2.16.7.2 Effect of receipt

The DME evaluates the reservation request in terms of the device’s availability and generates an MLME-DRP.response.

### 6.2.16.8 MLME-DRP.response

The DME uses this primitive to respond to a request for the creation of a new reservation or the modification or release of an existing reservation. The primitive's semantics are as follows:

```
MLME-DRP.response(
     DestEUI,
     SrcEUI,
     StreamIndex,
     ReservationType,
```

```
        ReservationBitmap,
        Explicit,
        ReasonCode,
        ResultCode
        )
```

#### 6.2.16.8.1 When generated

The DME generates this primitive in order to respond to a reservation request.

#### 6.2.16.8.2 Effect of receipt

The MLME constructs one or more DRP IEs that describe the DRP reservation response and either a) includes the DRP IEs in a subsequent beacon transmission or b) transmits the DRP IEs in a command frame to the device identified by DestEUI.

### 6.2.16.9 MLME-DRP.confirm

This primitive signals the completion, successfully or in error, of DRP negotiations initiated by a corresponding MLME-DRP.request. The primitive's semantics are as follows:

```
MLME-DRP.confirm(
     AvailabilityBitmap,
     ResultCode,
     ReasonCode
     )
```

#### 6.2.16.9.1 When generated

The MLME signals this primitive to the DME in order to confirm the success or failure of DRP negotiations initiated by a corresponding MLME-DRP.request.

For explicit unicast negotiations, if a DRP reservation response command frame is not received within an application-specific timeout after the DRP reservation request is sent, the MAC entity informs the DME by issuing a MLME-DRP.confirm with ResultCode equal to TIMEOUT. Once a DRP reservation response command frame is received, the source informs the DME by generating an MLME-DRP.confirm with the appropriate ResultCode.

For implicit unicast negotiations, if a DRP reservation response is not received within a time interval equal to mMaxLostBeacons superframes after the final DRP reservation request is sent, the MAC entity informs the DME by issuing an MLME-DRP.confirm with ResultCode equal to TIMEOUT.

Once a DRP reservation response is received, the source informs the DME by generating an MLME-DRP.confirm with the appropriate ResultCode.

#### 6.2.16.9.2 Effect of receipt

Dependent upon the value of ResultCode and ReasonCode, the DME should take appropriate action. For example, if the reservation is established successfully, the DME may signal the MAC client to transfer data to the MAC entity for that stream. If the DRP negotiation failed, the DME should signal the lack of resources to the responsible MAC client (higher-layer entity).

### 6.2.17 Connection configuration primitives

This mechanism provides control over the rate adaptation algorithm and metrics. The primitives covered in this subclause are listed in Table 38.

**Table 38 — Connection configuration primitives**

| Service primitive | Request | Indication | Response | Confirm |
|---|---|---|---|---|
| MLME-CONNECTION-CONFIG | 6.2.17.1 | | | 6.2.17.2 |

Table 39 defines the parameters that appear in the primitives of this subclause.

**Table 39 — Connection configuration primitive parameters**

| Name | Type | Valid range | Description |
|---|---|---|---|
| DestEUI | Integer | Any valid EUI-48 | Specifies the EUI-48 of remote responding device |
| DeliveryID | Integer | 0–15 | Specifies the user priority of the MSDU for a value in range 0 through 7, or the stream index of the MSDU for a value in range 8 through 15 |
| PHYRate | Enumeration | RATE_53_3, RATE_80, RATE_106_7, RATE_160, RATE_200, RATE_320, RATE_400, RATE_480 | PHY data rate at which packets are to be transmitted for the given connection |
| ACKPolicy | Enumeration | No_ACK / Imm-ACK / B-ACK | ACK policy to be used on the given connection |
| NOfReTX | Integer | 1–MaxNOfReTX | Number of re-transmissions allowed for a given connection |
| OptimalMPDUSize | Integer | PHY dependent | Optimal MPDU size for a given connection |
| AggregationPolicy | Enumeration | ENABLED, DISABLED | Aggregation policy enabled / disabled for a given connection |
| FragmentationPolicy | Enumeration | ENABLED, DISABLED | Fragmentation policy enabled / disabled for a given connection |
| ResultCode | Enumeration | FAILURE, SUCCESS | Completion status of the MLME request |

### 6.2.17.1 MLME-CONNECTION-CONFIG.request

This primitive is used to configure the rate adaptation parameters of a particular connection. The semantics of this primitive are:

```
MLME-CONNECTION-CONFIG.request(
     DestEUI,
     DeliveryID,
     PHYRate,
     ACKPolicy,
     NOfReTX,
```

```
        OptimalMPDUSize,
        AggregationPolicy,
        FragmentationPolicy
        )
```

#### 6.2.17.1.1 When generated

The primitive is generated by the DME whenever there is a need to update the parameters of the connection related to PHY rate adaptation.

#### 6.2.17.1.2 Effect of receipt

Upon reception of the primitive, the MLME will update the configuration of the specific connection with the parameters of the primitive, and will generate the MLME-CONNECTION-CONFIG.confirm response.

### 6.2.17.2 MLME-CONNECTION-CONFIG.confirm

The semantics of this primitive are:

```
MLME-CONNECTION-CONFIG.confirm(
     DestEUI,
     DeliveryID,
     ResultCode)
```

#### 6.2.17.2.1 When generated

This primitive is generated by the MLME in response to the MLME-CONNECTION-CONFIG.request.

#### 6.2.17.2.2 Effect of receipt

Upon reception of this primitive, the DME is notified of the success or failure of the updated connection configuration.

## 6.2.18 Range measurement

This mechanism supports range measurement between a device and a neighbor. The primitives covered in this subclause are listed in Table 40.

**Table 40 — Range measurement service primitives**

| Service primitive | Request | Indication | Response | Confirm |
|---|---|---|---|---|
| MLME-RANGE-MEASUREMENT | 6.2.18.1 | 6.2.18.2 | | 6.2.18.3 |

Table 41 defines the parameters used by the range measurement service primitives.

**Table 41 — Range measurement parameters**

| Name | Type | Valid Range | Description |
|---|---|---|---|
| Result | Enumeration | FAILURE, SUCCESS | Indicates the result of the range measurement operation |
| LREnable | Boolean | FALSE, TRUE | If TRUE, enable local range measurement, otherwise disable |
| DestEUI | EUI-48 | Any valid unicast EUI-48 | Specifies the EUI-48 of remote responding device |
| SrcEUI | EUI-48 | Any valid unicast EUI-48 | Specifies the EUI-48 of the remote requesting device |
| RMN | Integer | 0–255 | Number of measurements requested |
| MeasurementResultSet | Array | As described in Figure 29 | Range measurement results |
| MeasurementResultSetCount | Integer | 0–255 | Number of measurement results in MeasurementResultSet |

#### 6.2.18.1 MLME-RANGE-MEASUREMENT.request

This primitive is used to initiate one or more consecutive ranging measurements.

```
MLME-RANGE-MEASUREMENT.request(
     DestEUI,
     RMN
     )
```

##### 6.2.18.1.1 When generated

The DME signals this primitive to the MLME to initiate range measurement with a neighbor of the device. Parameter RMN may be one (for a simple estimate) or greater than one, so that the results of repeated measurements can be used to improve accuracy.

##### 6.2.18.1.2 Effect of receipt

The MLME generates frames to deliver over the medium to carry out the requested range measurements, and then delivers the result data to the DME.

#### 6.2.18.2 MLME-RANGE-MEASUREMENT.indication

This primitive is used to inform the DME that a range measurement request was received.

```
MLME-RANGE-MEASUREMENT.indication(
     SrcEUI,
     RMN
     )
```

##### 6.2.18.2.1 When generated

The MLME signals this primitive to the DME when it receives a range Measurement command frame with Range Type set to Range Measurement Request.

##### 6.2.18.2.2 Effect of receipt

The DME is advised that a range measurement operation has started.

#### 6.2.18.3 MLME-RANGE-MEASUREMENT.confirm

This primitive reports the results of a range measurement operation.

```
MLME-RANGE-MEASUREMENT.confirm(
     Result,
     MeasurementResultSet,
     MeasurementResultSetCount
     )
```

##### 6.2.18.3.1 When generated

The MLME signals this primitive to the DME when it has completed a requested range measurement operation.

##### 6.2.18.3.2 Effect of receipt

As specified in Annex C, the DME may use a single measurement result to calculate a single range estimate. Local and remote ranging clock options may be used to calculate confidence levels or error probabilities. Multiple measurements may be used to detect and correct for frequency errors between the local and remote clock frequencies.

### 6.2.19 Application-specific command management

The MAC sublayer provides application-specific command frames for sending control information for the MAC client. At the sending device, command data is passed via the primitives in this subclause, along with the parameters describing the attributes of the command, to the MAC entity for transfer to a peer MAC entity for unicast traffic or a group of MAC entities for multicast or broadcast traffic. At the recipient device, the MAC entity delivers received application-specific command data to the MAC client.

Application-specific commands may be fragmented for transfer between peer MAC entities.

Application-specific commands may be transmitted using PCA or within DRP reservations.

**Table 42 — Application-specific command management primitives**

| Service primitive | Request | Indication | Response | Confirm |
|---|---|---|---|---|
| MLME-AS-COMMAND | 6.2.19.1 | 6.2.19.2 | | 6.2.19.3 |

Table 43 lists the parameters that appear in the MLME-AS-COMMAND primitives defined in this subclause.

**Table 43 — Application-specific command management parameters**

| Name | Type | Valid range | Description |
|---|---|---|---|
| DestEUI | EUI-48 | Any valid EUI-48 | Specifies the EUI-48 of the recipient device of the MCDU |
| SrcEUI | EUI-48 | Any valid unicast EUI-48 | Specifies the EUI-48 of the sending device of the MCDU |
| TKID | Integer | Any valid TKID as defined in 7.2.6.1, or zero | Specifies a PTK or GTK used for protecting the MCDU. If zero, indicates no security protection for this MCDU. |
| ACKPolicy | Enumeration | ACK,<br>NO_ACK | Specifies whether or not the MCDU requires acknowledgement |
| TxTimeout | Duration | 0–65535 | Specifies the amount of time in milliseconds in which the MCDU needs to be successfully sent |
| ASCommandData | Array | Variable | Specifies the contents of the application-specific command frame |
| ResultCode | Enumeration | SUCCESS,<br>TX_TIMEOUT,<br>OTHER_REASONS | Indicates the result of the MCDU transfer request |

#### 6.2.19.1 MLME-AS-COMMAND.request

This primitive requests the transfer of an application-specific command to a peer MAC entity or a group of peer MAC entities.

The semantics of this primitive are:

```
MLME-AS-COMMAND.request(
     DestEUI,
     TKID,
     ACKPolicy,
     TxTimeout,
     ASCommandData
     )
```

##### 6.2.19.1.1 When generated

This primitive is generated by the MAC client when an application-specific command is to be transferred to a specified recipient.

##### 6.2.19.1.2 Effect of receipt

The MAC entity attempts to transmit the application-specific command frame using PCA or in a DRP reservation based on the other parameters provided in the primitive. The MAC entity subsequently issues a MLME-AS-COMMAND.confirm to reflect the results.

#### 6.2.19.2 MLME-AS-COMMAND.indication

This primitive reports the reception of an application-specific command from a specified sender.

The semantics of this primitive are:

```
MLME-AS-COMMAND.indication(
     SrcEUI,
     DestEUI,
     TKID,
     ACKPolicy,
```

```
        ASCommandData
        )
```

#### 6.2.19.2.1 When generated

This primitive is generated by the MAC entity to deliver to the MAC client a correctly received valid application-specific command frame addressed to this device.

#### 6.2.19.2.2 Effect of receipt

The MAC client is provided with a valid application-specific command frame addressed to this device from a specified sender.

### 6.2.19.3 MLME-AS-COMMAND.confirm

This primitive reports the result of an application-specific command frame transfer attempt to a specified recipient.

The semantics of this primitive are:

```
MLME-AS-COMMAND.confirm(
     DestEUI,
     ResultCode
     )
```

#### 6.2.19.3.1 When generated

This primitive is generated by the MAC entity as a result of a MLME-AS-COMMAND.request to transfer an application-specific command frame to a specified recipient.

#### 6.2.19.3.2 Effect of receipt

The MAC client is notified of the results of the attempt by the MAC entity to transfer an application-specific command frame based on the parameters specified in an earlier MLME-AS-COMMAND.request.

## 6.3 MAC SAP interface

The MAC sublayer provides a data transfer service to the MAC client via the MAC SAP, the logical data interface between the MAC client and the MAC sublayer. At the sending device, data is passed via the MAC SAP in MSDUs, along with the parameters describing the attributes of the MSDU, to the MAC entity for transfer to a peer MAC entity for unicast traffic or a group of MAC entities for multicast or broadcast traffic. At the recipient device, the MAC entity delivers received data also in MSDUs to the MAC client.

Each MSDU is tagged with a user priority or stream index as indicated by the Delivery ID parameter passed along with the MSDU at the MAC SAP. There are eight levels for user priority, and eight values for Stream Index designating up to eight possible streams between the sender and recipient. Legal values for the Delivery ID parameter are in the range of 0 to 15. A number in the range between 0 and 7, inclusive, denotes a user priority as defined by the IEEE 802.1D priority tag. A number in the range between 8 and 15, inclusive, identifies a stream from the sender to the recipient. The Delivery ID parameter is propagated across the wireless medium in the Delivery ID field of the Frame Control field of the MAC header.

Each device has an EUI-48 that, if non-NULL, uniquely identifies the device. This identifier is included in beacon frames, along with a 16-bit DevAddr selected by the device.

When an MSDU is delivered to the MAC sublayer by the MAC client, the device determines the DevAddr of the intended recipient based on the EUI-48 provided with the MSDU and information it receives in neighbors' beacons.

Before passing an MSDU to the MAC client, a device determines the EUI-48 of the sender of the MSDU using information it receives in neighbors' beacons.

All MSDUs labeled with the same Delivery ID and addressed to the same destination EUI-48 are transmitted in the order in which they arrived at the local MAC SAP when they are subject to the Imm-ACK or No-ACK acknowledgement policy. They may be transmitted out of order due to retries when the B-ACK acknowledgement policy is applied. However, MSDUs labeled with different Delivery ID values and/or addressed to different destination EUI-48s are not necessarily transmitted in the order in which they arrived at the local MAC SAP, since the MAC entity may reorder them for transmission based on their priorities and other considerations or constraints.

MSDUs of the same Delivery ID received by the recipient MAC entity are released to the MAC client via the local MAC SAP in the same order as they arrived at the MAC SAP of the sending device. The recipient MAC entity delivers MSDUs bearing different Delivery ID values to the MAC client in a fashion that preserves the respective orders of the MSDUs carrying the same Delivery ID values.

MSDUs may be fragmented or aggregated for transfer between peer MAC entities, but are always passed through the MAC SAP in whole MSDUs.

The MAC sublayer provides contention and reservation-based frame transfers. Contention based transfers use the prioritized contention access (PCA) method as specified in 8.2.3, while reservation-based transfers use the distributed reservation protocol (DRP) method as specified in 8.2.4. MSDUs tagged with a user priority are transmitted using PCA. MSDUs tagged with a Stream Index are transmitted primarily in a reservation and optionally by PCA.

The primitives covered in this subclause are listed in Table 44.

**Table 44 — MAC SAP primitives**

| Service primitive | Request | Indication | Response | Confirm |
|---|---|---|---|---|
| MAC-DATA | 6.3.1 | 6.3.2 | | 6.3.3 |

Table 45 lists the parameters that appear in the MAC SAP primitives defined in this subclause.

**Table 45 — MAC-SAP primitive parameters**

| Name | Type | Valid range | Description |
|---|---|---|---|
| DestEUI | EUI-48 | Any valid EUI-48 | Specifies the EUI-48 of the recipient device of the MSDU |
| SrcEUI | EUI-48 | Any valid unicast EUI-48 | Specifies the EUI-48 of the sending device of the MSDU |
| TKID | Integer | Any valid TKID as defined in 7.2.6.1, or zero | Specifies a PTK or GTK used for protecting the MSDU. If zero, indicates no security protection for this MSDU. |
| DeliveryID | Integer | 0–15 | Specifies the user priority of the MSDU for a value in range 0 through 7, or the stream index of the MSDU for a value in range 8 through 15 |
| TransmitTimeout | Duration | 0–65535 | Specifies the amount of time in milliseconds in which the MSDU needs to be successfully sent |
| MSDU | Octet string | | Specifies the data passing the MAC SAP before transmission or after reception |
| ResultCode | Enumeration | SUCCESS, TRANSMIT_TIMEOUT, OTHER_REASONS | Indicates the result of the MSDU transfer attempt |

### 6.3.1 MAC-DATA.request

This primitive requests the transfer of an MSDU to a peer MAC entity or a group of peer MAC entities.

The semantics of this primitive are:

```
MAC-DATA.request(
     DestEUI,
     TKID,
     DeliveryID,
     TransmitTimeout,
     MSDU
     )
```

#### 6.3.1.1 When generated

This primitive is generated by the MAC client when an MSDU is to be transferred to a specified recipient.

#### 6.3.1.2 Effect of receipt

The MAC entity attempts to transmit the MSDU based on the other parameters provided in the primitive. The MAC entity subsequently issues a MAC-DATA.confirm to reflect the results.

### 6.3.2 MAC-DATA.indication

This primitive reports the reception of an MSDU from a specified sender.

The semantics of this primitive are:

```
MAC-DATA.indication(
     SrcEUI,
     DestEUI,
     TKID,
     DeliveryID,
```

```
     MSDU
     )
```

#### 6.3.2.1 When generated

This primitive is generated by the MAC entity to deliver to the MAC client a received MSDU addressed to this device.

#### 6.3.2.2 Effect of receipt

The MAC client is provided with an MSDU addressed to this device from a specified sender.

### 6.3.3 MAC-DATA.confirm

This primitive reports the result of an MSDU transfer attempt to a specified recipient.

The semantics of this primitive are:

```
MAC-DATA.confirm(
     DestEUI,
     DeliveryID,
     ResultCode
     )
```

#### 6.3.3.1 When generated

This primitive is generated by the MAC entity as a result of a MAC-DATA.request to transfer an MSDU to a specified recipient.

#### 6.3.3.2 Effect of receipt

The MAC client is notified of the results of the attempt by the MAC entity to transfer an MSDU based on the parameters specified in an earlier MAC-DATA.request.

# 7 MAC frame formats

This clause specifies the format of MAC frames. An overview of the MAC frame with descriptions of common fields is followed by subclauses for each frame type and subtype. The final subclause contains a list of information elements that may appear in beacon frames and some command frames.

## 7.1 Frame format conventions

The following conventions and definitions apply throughout this clause.

### 7.1.1 Figures

MAC frames are described as a sequence of fields in a specific order. Figures in clause 7 depict fields in the order they are delivered to the PHY SAP, from left to right, where the left-most field is transmitted first in time. In field figures, bits within the field are numbered from the least-significant bit on the right to the most-significant bit on the left.

An example sequence of fields is illustrated in Figure 6.

| octets: 2 | 1 | ... | 4 |
|---|---|---|---|
| First field transmitted (2 octets) | Second field transmitted (1 octet) | … | Last field transmitted (4 octets) |

**Figure 6 — Example sequence of fields**

### 7.1.2 Octet order

Unless otherwise noted, fields longer than a single octet are delivered to the PHY SAP in order from the octet containing the least-significant bits to the octet containing the most-significant bits.

An example of a bitmap specification for a two-octet field is illustrated in Figure 7.

| bits: b15-b13 | b12-b8 | b7-b0 |
|---|---|---|
| Most-significant bits of second octet transmitted | Least-significant bits of second octet transmitted | First octet transmitted |

**Figure 7 — Example bitmap specification for a field**

### 7.1.3 Encoding

Values specified in decimal are encoded in unsigned binary unless otherwise stated.

A bitmap is a sequence of bits, labeled as bit[0] through bit[N-1]. A bitmap is encoded in a field such that bit[0] corresponds to the least-significant bit of the field and subsequent bitmap elements correspond to subsequent significant bits of the field. Octets of the field are presented to the PHY SAP in order from least-significant octet to most-significant octet.

Reserved fields and subfields are set to zero on transmission and ignored on reception. Fields and subfields are not set to reserved values on transmission. Unless otherwise noted, fields or subfields that are set to reserved values or are defined based on other fields or subfields that are set to reserved values are ignored on reception.

## 7.2 General MAC frame format

A MAC frame consists of a fixed-length MAC Header and an optional variable-length MAC Frame Body. The MAC Header is illustrated in Figure 8.

| octets: 2 | 2 | 2 | 2 | 2 |
|---|---|---|---|---|
| Frame Control | DestAddr | SrcAddr | Sequence Control | Access Information |

**Figure 8 — MAC Header format**

The MAC Frame Body, when present, contains a Frame Payload and Frame Check Sequence (FCS) as shown in Figure 9.

| octets: $L_n$ | 4 |
|---|---|
| Frame Payload | FCS |

**Figure 9 — MAC Frame Body format**

In secure frames the Frame Payload includes security fields as shown in Figure 10. The left-most four fields in Figure 10 are collectively referred to as the Security Header.

| octets: 3 | 1 | 2 | 6 | P | 8 |
|---|---|---|---|---|---|
| Temporal Key Identifier (TKID) | Security Reserved | Encryption Offset (EO) | Secure Frame Number (SFN) | Secure Payload | Message Integrity Code (MIC) |

**Figure 10 — Frame Payload field format for secure frames**

The Frame Payload length ranges from zero to mMaxFramePayloadSize. If the Frame Payload length is zero, the FCS field is not included, and there is no MAC Frame Body. The Frame Payload length includes the length of the security fields for a secure frame.

In this clause, a reference to the payload of a frame indicates the Frame Payload field of a non-secure frame, or the Secure Payload field of a secure frame. The payload is a sequence of octets labeled as payload[0] through payload[P-1]. Octets are passed to the PHY SAP in ascending index-value order.

### 7.2.1 Frame Control

The Frame Control field is illustrated in Figure 11.

| bits: b15-b14 | b13 | b12-b9 | b8-b6 | b5-b4 | b3 | b2-b0 |
|---|---|---|---|---|---|---|
| Reserved | Retry | Frame Subtype / Delivery ID | Frame Type | ACK Policy | Secure | Protocol Version |

**Figure 11 — Frame Control field format**

#### 7.2.1.1 Protocol Version

The Protocol Version field is invariant in size and placement across all revisions of this standard. For this revision of the standard, Protocol Version is set to zero. All other values are reserved.

#### 7.2.1.2 Secure

The Secure bit is set to one in a secure frame, which is protected using the temporal key specified by the Temporal Key Identifier (TKID). The Secure bit is set to zero otherwise. Frames with the Secure bit set to one use the Frame Payload format for secure frames as shown in Figure 10. Valid settings for the Secure bit in each frame type are listed in Table 72 in subclause 9.2.

#### 7.2.1.3 ACK Policy

The ACK Policy field is set to the type of acknowledgement requested by the transmitter. Acknowledgement procedures are described in 8.8. The allowed values for the ACK Policy field are defined in Table 46.

**Table 46 — ACK Policy field encoding**

| Value | ACK policy type | Description |
|---|---|---|
| 0 | No-ACK | The recipient(s) do not acknowledge the transmission, and the sender treats the transmission as successful without regard for the actual result. The use of this policy is defined in 8.8.1. |
| 1 | Imm-ACK | The addressed recipient returns an Imm-ACK frame after correct reception, according to the procedures defined in 8.8.2. |
| 2 | B-ACK | The addressed recipient keeps track of the frames received with this policy until requested to respond with a B-ACK frame, according to the procedures defined in 8.8.3. |
| 3 | B-ACK Request | The addressed recipient returns a B-ACK frame after reception, according to the procedures defined in 8.8.3. |

#### 7.2.1.4 Frame Type

The Frame Type field is set to the type of frame that is being sent. Table 47 lists the valid frame type values, descriptions, and the subclauses that describe the format and use of each of the individual frame types.

**Table 47 — Frame Type field encoding**

| Value | Frame type | Subclause |
|---|---|---|
| 0 | Beacon frame | 7.3 |
| 1 | Control frame | 7.4 |
| 2 | Command frame | 7.5 |
| 3 | Data frame | 7.6 |
| 4 | Aggregated data frame | 7.7 |
| 5–7 | Reserved | |

#### 7.2.1.5 Frame Subtype / Delivery ID

The Frame Subtype / Delivery ID field is used to assist a receiver in the proper processing of received frames. In control or command frames, this field is used as Frame Subtype, as defined in Table 51 (subclause 7.4) and Table 53 (subclause 7.5). In data frames and aggregated data frames, this field is used as Delivery ID as defined in Table 48.

**Table 48 — Delivery ID encoding in Frame Control**

| b12 | b11-b9 |
|---|---|
| 0 | User Priority |
| 1 | Stream Index |

This field is reserved in all other frame types.

#### 7.2.1.6 Retry

The Retry bit is set to one in any data, aggregated data, or command frame that is a retransmission of an earlier frame. It is reserved in all other frame types.

### 7.2.2 DestAddr

The DestAddr field is set to the DevAddr of the intended recipient(s) of the frame. The DevAddr specifies a single device for a unicast frame, a group of devices for a multicast frame, or all devices for a broadcast frame. DevAddr values are described in 8.1.1.

### 7.2.3 SrcAddr

The SrcAddr field is set to the DevAddr of the transmitter of the frame.

### 7.2.4 Sequence Control

The Sequence Control field identifies the order of MSDUs/MCDUs and their fragments. The Sequence Control field is illustrated in Figure 12. The Sequence Control field is reserved in control frames.

| bits: b15 | b14 | b13-b3 | b2-b0 |
|---|---|---|---|
| Reserved | More Fragments | Sequence Number | Fragment Number |

**Figure 12 — Sequence Control field format**

#### 7.2.4.1 Fragment Number

The Fragment Number field is set to the number of the fragment within the MSDU or MCDU. The fragment number is zero in the first or only fragment of an MSDU or MCDU and is incremented by one for each successive fragment of that MSDU or MCDU.

#### 7.2.4.2 Sequence Number

The Sequence Number field is set to the sequence number of the MSDU or MCDU, as defined in 8.1.9.3.

The Sequence Number field is used for duplicate frame detection, as described in 8.1.7, and to preserve frame order when using the B-ACK mechanism, as described in 8.8.3.

The Sequence Number field is reserved in control frames.

#### 7.2.4.3 More Fragments

The More Fragments field is set to zero to indicate that the current fragment is the sole or final fragment of the current MSDU or MCDU; otherwise the field is set to one.

### 7.2.5 Access Information

The Access Information field is illustrated in Figure 13.

| bits: b15 | b14 | b13-b0 |
|---|---|---|
| Access Method | More Frames | Duration |

**Figure 13 — Access Information field format**

#### 7.2.5.1 Duration

The Duration field is 14 bits in length and is set to an expected medium busy interval after the end of the PLCP header of the current frame in units of microseconds. The duration value is set as defined in 8.1.9.1 and used to update the network allocation vector (NAV) according to the procedures defined in 8.3.2.

#### 7.2.5.2 More Frames

In frames sent with the Access Method bit set to one, the More Frames bit is set to zero if the transmitter will not send further frames to the same recipient during the current reservation block; otherwise it is set to one.

In frames sent with the Access Method bit set to zero, the More Frames bit is set to zero if the transmitter will not send further frames to the same recipient during the current superframe; otherwise it is set to one.

The More Frames bit is reserved in beacon and control frames. Additional rules regarding the More Frames field are specified in 8.1.9.2.

#### 7.2.5.3 Access Method

The Access Method bit is set to one in all frames transmitted within a hard or private DRP reservation block by the reservation owner or target prior to the release of the reservation block, including the UDA and UDR control frames that release the reservation block.

The Access Method bit may be set to one in frames transmitted within a Soft DRP reservation block without backoff by the reservation owner.

The Access Method bit in an Imm-ACK, B-ACK or CTS control frame is set to the same value as the Access Method bit in the corresponding received frame.

The Access Method bit is set to zero in frames transmitted at all other times, other than in beacon frames.

The Access Method bit is reserved in beacon frames.

### 7.2.6 Frame Payload

The Frame Payload field is a variable length field that carries the information that is to be transferred to a device or group of devices. In a secure frame, it includes the required security fields as shown in Figure 10 and defined below.

#### 7.2.6.1 Temporal Key Identifier (TKID)

The TKID field is an identifier for the temporal key used to protect the frame. The TKID uniquely identifies this key from any other temporal keys held by the sender and the recipient(s) of the frame. It does not need to uniquely identify the key for devices not holding the key.

#### 7.2.6.2 Security Reserved

The Security Reserved field is reserved, but included in authentication of the frame.

#### 7.2.6.3 Encryption Offset (EO)

The Encryption Offset field indicates where encryption starts, in octets, relative to the beginning of the Secure Payload, as shown in Figure 10. A value of zero indicates that the entire Secure Payload is encrypted. A non-zero value in this field indicates that the first EO octets of the Security Payload are not encrypted. Regardless of the value of this field, the entire Secure Payload, along with other appropriate fields, is authenticated by the MIC.

#### 7.2.6.4 Secure Frame Number (SFN)

The SFN field provides message freshness as a defense against replay attacks. The SFN field in a secure frame is set to the next value of the sender's secure frame counter (SFC) for the temporal key used by this frame. SFC setting and replay protection are described in 9.4.2.

#### 7.2.6.5 Secure Payload

The Secure Payload field in secure frames is the counterpart of the Frame Payload field in non-secure frames. It contains the information specific to individual frame types and protected by the symmetric key identified in the TKID field of the same frame.

#### 7.2.6.6 Message Integrity Code (MIC)

The MIC field contains an 8-octet cryptographic checksum used to protect the integrity of the MAC Header and Frame Payload.

### 7.2.7 FCS

The FCS field contains a 32-bit value that represents a CRC polynomial of degree 31.

The CRC is calculated over a calculation field, which is the entire Frame Payload field for this specification. The calculation field is mapped to a message polynomial M(x) of degree k-1, where k is the number of bits in the calculation field. The least-significant bit of the first octet presented to the PHY SAP is the coefficient of the $x^{k-1}$ term, and the most-significant bit of the last octet transmitted is the coefficient of the $x^0$ term.

The CRC is calculated using the following standard generator polynomial of degree 32:

$$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

The CRC polynomial is the one's complement of the modulo 2 sum of the following remainders:

— The remainder resulting from $x^k \times (x^{31} + x^{30} + ... + x + 1)$ divided (modulo 2) by G(x).
— The remainder resulting from $x^{32} \times M(x)$, divided (modulo 2) by G(x).

The FCS field value is derived from the CRC polynomial such that the least-significant bit is the coefficient of the $x^{31}$ term and the most-significant bit is the coefficient of the $x^0$ term. Figure 14 illustrates the encoding of the FCS field for the CRC polynomial:

$$a_{31}x^{31} + a_{30}x^{30} + a_{29}x^{29} + \ldots + a_2x^2 + a_1x + a_0$$

| bits: b31 | b30 | b29 | ... | b2 | b1 | b0 |
|---|---|---|---|---|---|---|
| $a_0$ | $a_1$ | $a_2$ | … | $a_{29}$ | $a_{30}$ | $a_{31}$ |

**Figure 14 — FCS field encoding**

In a common implementation, at the transmitter, the initial remainder of the division is preset to all ones and is then modified via division of the calculation field by the generator polynomial G(x). The one's complement of this remainder is the FCS field. At the receiver, the initial remainder is preset to all ones. The serial incoming bits of the calculation field and FCS, when divided by G(x) in the absence of transmission errors, results in a unique non-zero remainder value. The unique remainder value is the polynomial:

$$x^{31} + x^{30} + x^{26} + x^{25} + x^{24} + x^{18} + x^{15} + x^{14} + x^{12} + x^{11} + x^{10} + x^8 + x^6 + x^5 + x^4 + x^3 + x + 1$$

## 7.3 Beacon frames

MAC Header field settings for beacon frames are described in Table 49. Beacon frames are also referred to as beacons throughout this specification.

**Table 49 — MAC Header field values for beacon frames**

| Header field | Value |
|---|---|
| Protocol Version | 0 |
| Secure | 0 |
| ACK Policy | 0 (No-ACK) |
| Frame Type | 0 (beacon frame) |
| Frame Subtype / Delivery ID | Reserved |
| Retry | Reserved |
| DestAddr | BcstAddr |
| SrcAddr | DevAddr of the transmitter |
| Sequence Control | As defined in 7.2.4 and 8.1.9.3 |
| Duration | As defined in 7.2.5.1 and 8.1.9.1 |
| More Frames | Reserved |
| Access Method | Reserved |

The beacon frame payload is illustrated in Figure 15.

| octets: 8 | $L_1$ | … | $L_N$ |
|---|---|---|---|
| Beacon Parameters | Information Element 1 | … | Information Element N |

**Figure 15 — Payload format for Beacon frames**

The information elements (IEs) that may be included in a beacon frame are listed in Table 57 in 7.8. IEs are included in order of increasing Element ID, except for ASIEs. ASIEs do not appear prior to any IE with Element ID zero through seven, but may appear anywhere after those IEs. DRP IEs that have the same Target DevAddr and Stream Index are adjacent to each other in the beacon.

The Beacon Parameters field is illustrated in Figure 16.

| octets: 6 | 1 | 1 |
|---|---|---|
| Device Identifier | Beacon Slot Number | Device Control |

**Figure 16 — Beacon Parameters field format**

The Device Identifier field is set to the EUI-48 [B1] of the device sending the beacon. A device may use a NULL EUI-48 value (all bits set to one) to indicate it does not have a unique EUI-48 value. The EUI is a sequence of 6 octets, labeled as eui[0] through eui[5]. The first three octets (eui[0] through eui[2]) are the manufacturer's OUI, and the last three octets (eui[3] through eui[5]) are the manufacturer-selected extension identifier. Octets of the EUI are passed to the PHY SAP in ascending index-value order.

The Beacon Slot Number field is set to the number of the beacon slot where the beacon is sent within the beacon period (BP), in the range of [0, mMaxBPLength-1], except in beacons sent in signaling slots. In signaling slots it is set to the number of the device's non-signaling beacon slot.

The Device Control field is illustrated in Figure 17.

| bits: b7-b6 | b5-b2 | b1 | b0 |
|---|---|---|---|
| Security Mode | Reserved | Signaling Slot | Movable |

**Figure 17 — Device Control field format**

The Movable bit is set to one if the beacon is movable according to 8.2.5, and is set to zero otherwise.

The Signaling Slot bit is set to one if the beacon is sent in a signaling beacon slot according to 8.2.3, and is set to zero otherwise.

The Security Mode field is set to the security mode at which the device is currently operating.

## 7.4 Control frames

Default MAC Header field settings for control frames are listed in Table 50. Specific MAC Header field settings and payload descriptions for each of the control frames are shown in the following subclauses.

**Table 50 —MAC Header field values for control frames**

| Header field | Value |
|---|---|
| Protocol Version | 0 |
| Secure | As defined in 7.2.1.2 |
| ACK Policy | 0 (No-ACK) |
| Frame Type | 1 (control frame) |
| Frame Subtype | Value from Table 51 |
| Retry | Reserved |
| DestAddr | DevAddr of the recipient |
| SrcAddr | DevAddr of the transmitter |
| Sequence Control | Reserved |
| Duration | As described in 7.2.5.1 and 8.1.9.1 |
| More Frames | Reserved |
| Access Method | As described in 7.2.5.3 |

Table 51 lists valid values for the Frame Subtype field for control frames.

**Table 51 — Frame Subtype field encoding for control frames**

| Value | Control frame subtype | Description |
|---|---|---|
| 0 | Imm-ACK | Acknowledges correct receipt of the previously-received frame |
| 1 | B-ACK | Acknowledges correct or incorrect receipt of one or more preceding frames |
| 2 | RTS | Announces to a recipient device that a frame is ready for transmission and requests confirmation of ability to receive |
| 3 | CTS | Responds to an RTS control frame that the recipient is able to receive |
| 4 | UDA | Announces to neighbors of the transmitting device that the remainder of a reservation block it owns is available for use by other devices via PCA |
| 5 | UDR | Announces to neighbors of the transmitting device that the remainder of a reservation block of which it is the target is available for use by other devices via PCA |
| 6–13 | Reserved | Reserved |
| 14 | Application-specific | At discretion of application owner |
| 15 | Reserved | Reserved |

### 7.4.1 Immediate Acknowledgement (Imm-ACK)

In Imm-ACK frames, the DestAddr field is set to the SrcAddr of the received frame that is acknowledged. Imm-ACK frames have no frame payload.

### 7.4.2 Block Acknowledgement (B-ACK)

In B-ACK frames, the DestAddr field is set to the SrcAddr of the frame that requested the B-ACK.

The B-ACK frame acknowledges correct or incorrect receipt of the previous sequence of frames and provides information for the transmission of the next sequence of frames as described in 8.8.3. The B-ACK frame payload is illustrated in Figure 18.

| octets: 2 | 1 | 1 | 2 | 0-n |
|---|---|---|---|---|
| Buffer Size | Frame Count | Reserved | Sequence Control | Frame Bitmap |

**Figure 18 — Payload format for B-ACK frames**

The Buffer Size field specifies the maximum number of octets in the sum of the frame payloads of all frames in the next B-ACK sequence.

The Frame Count field specifies the maximum number of frames in the next B-ACK sequence.

The Sequence Control and Frame Bitmap fields together specify an acknowledgement window of MSDU fragments and their reception status. The Sequence Control field specifies the Sequence Number and Fragment Number that start the acknowledgement window.

| bits: b15-14 | b13-b3 | b2-b0 |
|---|---|---|
| Reserved | Sequence Number | Fragment Number |

**Figure 19 — Sequence Control field format**

The Frame Bitmap field varies in length. A zero-length Frame Bitmap field indicates an acknowledgement window of length zero. Otherwise, the least-significant octet of the Frame Bitmap field corresponds to the MSDU indicated by the Sequence Control field, and each bit of the octet corresponds to a fragment of that MSDU. The least-significant bit in each octet corresponds to the first fragment and successive bits correspond to successive fragments. Successive octets present in the Frame Bitmap field correspond to successive MSDUs, and each bit corresponds to a fragment of the MSDU. The acknowledgement window ends at fragment seven of the MSDU that corresponds to the most-significant octet in the Frame Bitmap.

For all bits within the Frame Bitmap, a value of one indicates that the corresponding fragment was received in either the current sequence or an earlier one. A value of zero indicates that the corresponding fragment was not received in the current sequence (although it may have been received in an earlier one). Bits of the least-significant octet of the Frame Bitmap field corresponding to fragments prior to the start of the acknowledgement window are undefined. Frames with a Sequence Number earlier than the Sequence Number indicated in the Sequence Control field were not received in the last B-ACK sequence. Such frames were previously received or are no longer expected.

### 7.4.3 Request To Send (RTS)

In RTS frames, the DestAddr field is set to the DevAddr of the device to receive the following frame from the transmitter. RTS frames have no frame payload.

### 7.4.4 Clear To Send (CTS)

In CTS frames, the DestAddr field is set to the SrcAddr of the received RTS frame. CTS frames have no frame payload.

### 7.4.5 Unused DRP Reservation Announcement (UDA)

The UDA frame is used to explicitly release the remaining time of the current Hard or Private DRP reservation block. The DestAddr field is set to BcstAddr.

The UDA frame payload includes a list of DevAddrs of the devices that will respond with a UDR frame, as shown in Figure 20.

| octets: 2 | … | 2 |
|---|---|---|
| DevAddr 1 | … | DevAddr N |

**Figure 20 — Payload format for UDA frames**

### 7.4.6 Unused DRP Reservation Response (UDR)

The UDR frame is used to respond to UDA frames to explicitly release the remaining time of the current Hard or Private DRP reservation block. The DestAddr field is set to the SrcAddr of the received UDA frame. UDR frames have no frame payload.

### 7.4.7 Application-specific

The payload format for Application-specific control frames is illustrated in Figure 21.

| octets: 2 | … |
|---|---|
| Specifier ID | Data |

**Figure 21 — Payload format for Application-specific control frames**

The Specifier ID field is set to a 16-bit value that identifies a company or organization, as listed in *WiMedia Assigned Numbers* [B5]. The owner of the Specifier ID defines the format and use of the Data field.

## 7.5 Command frames

Default MAC Header settings for command frames are shown in Table 52.

**Table 52 — Default MAC Header field values for command frames**

| Header field | Value |
|---|---|
| Protocol Version | 0 |
| Secure | As defined in 7.2.1.2 |
| ACK Policy | 0 (No-ACK) or 1 (Imm-ACK) |
| Frame Type | 2 (command frame) |
| Frame Subtype | Value from Table 53 |
| Retry | As defined in 7.2.1.6 |
| DestAddr | DevAddr of the recipient |
| SrcAddr | DevAddr of the transmitter |
| Sequence Control | As defined in 7.2.4 |
| Duration | As described in 7.2.5.1 and 8.1.9.1 |
| More Frames | As defined in 7.2.5.2 |
| Access Method | As defined in 7.2.5.3 |

Table 53 contains a list of valid values for the Frame Subtype field for command frames.

**Table 53 — Frame Subtype field encoding for Command frames**

| Value | Command frame subtype | Description |
|---|---|---|
| 0 | DRP Reservation Request | Used to request creation or modification of a DRP reservation |
| 1 | DRP Reservation Response | Used to respond to a DRP reservation request command |
| 2 | Probe | Used to request for, or respond with, information elements |
| 3 | Pair-wise Temporal Key (PTK) | Used to derive a PTK via a 4-way handshake between two devices |
| 4 | Group Temporal Key (GTK) | Used to solicit or distribute a GTK within a secure relationship |
| 5 | Range Measurement | Used to exchange timing information for range measurement |
| 6–13 | Reserved | Reserved |
| 14 | Application-specific | At discretion of application owner |
| 15 | Reserved | Reserved |

### 7.5.1 DRP Reservation Request

The DRP Reservation Request command frame is used to create or modify a DRP reservation. The DRP Reservation Request command frame payload is illustrated in Figure 22.

| octets: $M_1$ | $M_2$ | … | $M_N$ |
|---|---|---|---|
| DRP IE-1 | DRP IE-2 | … | DRP IE-N |

**Figure 22 — Payload format for DRP Reservation Request command frames**

Each DRP IE field included in the command frame corresponds to a reservation request identified by the Target/Owner DevAddr, Stream Index, and Reservation Type in the IE. The DRP IE is defined in 7.8.6.

### 7.5.2 DRP Reservation Response

The DRP Reservation Response command frame is used to respond to a DRP Reservation Request command frame. The DRP Reservation Response command frame payload is illustrated in Figure 23.

| octets: $M_1$ | $M_2$ | … | $M_N$ | 2 to 34 |
|---|---|---|---|---|
| DRP IE-1 | DRP IE-2 | … | DRP IE-N | DRP Availability IE |

**Figure 23 — Payload format for DRP Reservation Response command frames**

The DRP Reservation Response command frame includes all the DRP IEs from the reservation request. The DRP Availability IE is included according to the rules defined in 8.4.

### 7.5.3 Probe

The Probe command frame is used to request information from a device or respond to a Probe request. The payload format is illustrated in Figure 24.

| octets: $M_1$ | $M_2$ | … | $M_N$ |
|---|---|---|---|
| Information Element 1 | Information Element 2 | … | Information Element N |

**Figure 24 — Payload format for Probe command frames**

If the payload includes a Probe IE, the command requests information from the recipient. Each Information Element field contains one information element.

### 7.5.4 Pairwise Temporal Key (PTK)

The PTK command frame is used in a 4-way handshake by a pair of devices, as described in 9.3.1, to authenticate each other and to derive a shared symmetric PTK for securing certain unicast traffic between the two devices. The PTK command frame is illustrated in Figure 25.

| octets: 1 | 1 | 3 | 11 | 16 | 16 | 8 |
|---|---|---|---|---|---|---|
| Message Number | Status Code | PTKID | Reserved | MKID | I-Nonce / R-Nonce | PTK MIC |

**Figure 25 — Payload format for PTK command frames**

The Message Number is set to 1, 2, 3, or 4, respectively, in the PTK command containing the first, second, third, or fourth message of the 4-way handshake. The other values of this field are reserved.

The Status Code in a PTK command indicates the current status of the 4-way handshake at the device sending this command. It is encoded as shown in Table 54.

**Table 54 — Status Code field encoding in PTK commands**

| Value | Meaning |
|---|---|
| 0 | Normal—the 4-way handshake proceeds. |
| 1 | Aborted—the 4-way handshake is aborted per security policy. |
| 2 | Aborted—the 4-way handshake is aborted in order to yield to a concurrent 4-way handshake using the same master key. |
| 3 | PTKID not accepted—it is the TKID of a PTK or GTK being possessed by this device. |
| 4–255 | Reserved |

The PTKID is set to a non-zero number as the TKID of the PTK to be derived from this 4-way handshake procedure. The initiator of the 4-way handshake chooses this value after determining that this value is different from the TKID of the PTK, if any, that is to be replaced by the new PTK, and the TKID of any PTK or GTK it currently possesses.

The MKID identifies the master key used in this 4-way handshake as described in 9.3.1.

The I-Nonce/R-Nonce is a random number generated by the initiator or responder for this 4-way handshake. This field is set to I-Nonce, the random number generated by the initiator in the

command containing a Message Number of 1 or 3, and is set to R-Nonce, the random number generated by the responder in the command containing a Message Number of 2 or 4.

The PTK MIC in the PTK command containing a Message Number of 1 is set to zero on transmission and is ignored on reception.

The PTK MIC in the PTK command containing a Message Number of 2, 3, or 4 is set to the MIC that protects the fields in the payload of this command using the KCK generated from the first two messages of the 4-way handshake as specified in 9.3.1.

The MAC Header for the PTK command frame is set as indicated in Table 52, with the ACK Policy set to Imm-ACK.

### 7.5.5 Group Temporal Key (GTK)

The GTK command frame is used to solicit or distribute a GTK following a PTK update. The GTK is used to secure certain multicast traffic from a sending device to a group of recipient devices, and is chosen by the sending device. The GTK command frame is always in secure form, and the Secure Payload field is illustrated in Figure 26.

| octets: 1 | 1 | 3 | 3 | 2 | 6 | 16 |
|---|---|---|---|---|---|---|
| Message Number | Status Code | GTKID | Reserved | GroupAddr | GTK SFC | GTK |

**Figure 26 — Payload format for GTK command frames**

The Message Number is set to 0 in the GTK command transmitted by a multicast recipient device to solicit a new GTK from a multicast sender. The Message Number is set to 1 in the GTK command transmitted by a multicast sender to distribute a new GTK to a multicast recipient. The Message Number is set to 2 in the GTK command transmitted by a multicast recipient device to respond to the distribution of a new GTK command.

The Status Code in a GTK command indicates the current status of the GTK solicitation or distribution at the device sending this command. It is encoded as shown in Table 55.

**Table 55 — Status Code field encoding in GTK commands**

| Value | Meaning |
|---|---|
| 0 | Normal—GTK solicitation or distribution proceeds. |
| 1 | Rejected—GTK solicitation or distribution is rejected per security policy. |
| 2 | GTKID not accepted—it is the TKID of a PTK or GTK being possessed by this device. |
| 3–255 | Reserved |

The GTKID in the GTK command containing a Message Number of 0 is set to the TKID of the GTK being solicited. It is set to zero if the soliciting device does not know the TKID of the GTK it is soliciting.

The GTKID in the GTK command containing a Message Number of 1 is set to a non-zero number as the TKID of the GTK being distributed. The distributor chooses this value after determining that this value is different from the TKID of the GTK, if any, that is to be replaced by the new GTK, and the TKID of any PTK or GTK the distributor or recipient currently possesses.

The GTKID in the GTK command containing a Message Number of 2 is set to the GTKID in the last received GTK command containing a Message Number of 1.

The GroupAddr is set to the McstAddr or BcstAddr for which the GTK is being solicited or distributed. It is set to 0x0001 if the GTK is applied to all broadcast and multicast traffic from the device distributing this GTK.

The GTK SFC in the GTK command containing a Message Number of 0 is set to zero on transmission and ignored on reception.

The GTK SFC in the GTK command containing a Message Number of 1 is set to the current value of the secure frame counter set up for the GTK being distributed.

The GTK SFC in the GTK command containing a Message Number of 2 is set to the GTK SFC in the last received GTK command containing a Message Number of 1.

The GTK is the GTK distributed by the multicast sender for the McstAddr. In a GTK command soliciting a GTK, the GTK is set to zero prior to encryption.

The MAC Header for the GTK command frame is set as indicated in Table 52, with the ACK Policy set to Imm-ACK.

### 7.5.6 Range Measurement

The Range measurement command frame payload is illustrated in Figure 27.

| octets: 1 | N |
|---|---|
| Range Type | Range Payload |

**Figure 27 — Payload format for Range Measurement command frames**

The Range Payload field format definition depends on the Range Type as shown in Table 56.

**Table 56— Range Type field encoding**

| Value | Range Payload |
|---|---|
| 0 | Range Measurement Request |
| 1 | Range Measurement |
| 2 | Range Measurement Report |
| 3–255 | Reserved |

The Range Payload field for Range Measurement Request type is illustrated in Figure 28.

| octets: 1 |
|---|
| Requested Measurement Number |

**Figure 28 — Range Payload field format for Range Measurement Request type**

The Requested Measurement Number field contains the number of consecutive two-way time transfer measurements.

The Range Payload field for Range Measurement type contains zero octets.

The Range Payload field for Range Measurement Report type is illustrated in Figure 29.

| octets: 1 | 1 | 1 | 4 | 4 | … | 4 | 4 |
|---|---|---|---|---|---|---|---|
| Measurement Count | Range Supported | PHYClockAccuracy | $R1C_1$ | $T2C_1$ | … | $R1C_N$ | $T2C_N$ |

**Figure 29 — Range Payload field format for Range Measurement Report type**

The Measurement Count field indicates the number of measurements reported.

The Range Supported field is illustrated in Figure 30, and indicates range measurement support and range measurement precision. Bits are set to one to indicate support or to zero to indicate the feature is not supported.

| bits: b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| Reserved | 32-bit counter supported | 24-bit counter supported | 4224 MHz sample precision | 2112 MHz sample precision | 1056 MHz sample precision | 528 MHz sample precision | Range measurements supported |

**Figure 30 — Range Supported field format**

PHYClockAccuracy indicates the accuracy of the PHY clock in units of ppm.

Each pair of R1C and T2C fields contains the range measurement reception timer and range measurement transmission timer value respectively.

### 7.5.7 Application-specific

The payload format for Application-specific command frames is illustrated in Figure 31.

| octets: 2 | … |
|---|---|
| Specifier ID | Data |

**Figure 31 — Payload format for Application-specific command frame**

The Specifier ID field is set to a 16-bit value that identifies a company or organization, as listed in *WiMedia Assigned Numbers* [B5]. The owner of the Specifier ID defines the format and use of the Data field.

## 7.6 Data frames

MAC Header and Frame Payload fields in data frames are set as described in 7.2.

## 7.7 Aggregated data frames

In aggregated data frames, the payload contains an Aggregation Header and multiple MSDUs, each aligned to a 4-octet boundary. The aggregated data frame payload is illustrated in Figure 32.

| octets: 2+(2×N) | 0 or 2 | $M_1$ | 0-3 | $M_2$ | 0-3 | … | $M_N$ |
|---|---|---|---|---|---|---|---|
| Aggregation Header | Pad to 4-octet boundary | MSDU 1 | Pad to 4-octet boundary | MSDU 2 | Pad to 4-octet boundary | … | MSDU N |

**Figure 32 — Payload format for aggregated data frames**

The Frame Payload size for aggregated data frames is subject to the same maximum size as any Frame Payload.

The Aggregation Header field is illustrated in Figure 33.

| octets: 1 | 1 | 2 | ... | 2 |
|---|---|---|---|---|
| MSDU Count | Reserved | Length of MSDU 1 | … | Length of MSDU N |

**Figure 33 — Aggregation Header field format**

The MSDU Count field contains the number of MSDUs included in the aggregated frame.

The Length fields in the Aggregation Header field indicate the length in octets of the corresponding MSDUs. The lengths do not include the Pad octets.

## 7.8 Information elements

This subclause defines the information elements (IEs) that can appear in beacons and certain command frames.

The general format of all IEs is illustrated in Figure 34.

| octets: 1 | 1 | N |
|---|---|---|
| Element ID | Length (=N) | IE-specific fields |

**Figure 34 — General IE format**

The Element ID field is set to the value as listed in Table 57 that identifies the information element.

The Length field is set to the length, in octets, of the IE-specific fields that follow.

The IE-specific fields contain information specific to the IE.

Table 57 contains a list of IEs defined in this standard.

**Table 57 —Information elements**

| Element ID | Information element | Description |
|---|---|---|
| 0 | Traffic Indication Map (TIM) IE | Indicates that a device has data buffered for transmission via PCA |
| 1 | Beacon Period Occupancy IE (BPOIE) | Provides information on neighbors' BP occupancy in the previous superframe |
| 2 | PCA Availability IE | Indicates the MASs that a device is available to receive PCA frames and transmit the required response |
| 3–7 | Reserved | Reserved |
| 8 | DRP Availability IE | Indicates a device's availability for new DRP reservations |
| 9 | Distributed Reservation Protocol (DRP) IE | Indicates a reservation with another device |
| 10 | Hibernation Mode IE | Indicates the device will go to hibernation mode for one or more superframes but intends to wake at a specified time in the future |
| 11 | BP Switch IE | Indicates the device will change its BPST at a specified future time |
| 12 | MAC Capabilities IE | Indicates which MAC capabilities a device supports |
| 13 | PHY Capabilities IE | Indicates which PHY capabilities a device supports |
| 14 | Probe IE | Indicates a device is requesting one or more IEs from another device or/and responding with requested IEs |
| 15 | Application-specific Probe IE | Indicates a device is requesting an Application-specific IE from another device |
| 16 | Link Feedback IE | Provides data rate and power control feedback |
| 17 | Hibernation Anchor IE | Provides information on devices in hibernation mode |
| 18 | Channel Change IE | Indicates a device will change to another channel |
| 19 | Identification IE | Provides identifying information about the device, including a name string |
| 20 | Master Key Identifier (MKID) IE | Identifies some or all of the master keys held by the transmitting device |
| 21 | Relinquish Request IE | Indicates that a neighbor requests that a device release one or more MASs from its reservations. |
| 22 | Multicast Address Binding (MAB) IE | Indicates an address binding between a multicast EUI-48 and a McstAddr |
| 23–249 | Reserved | Reserved |
| 250–254 | Reserved | Reserved for allocation in *WiMedia Assigned Numbers* [B5] |
| 255 | Application-Specific IE (ASIE) | Use varies depending on the application |

### 7.8.1 Application-specific IE (ASIE)

The ASIE is illustrated in Figure 35.

| octets: 1 | 1 | 2 | N |
|---|---|---|---|
| Element ID | Length (=2+N) | ASIE Specifier ID | Application-specific Data |

**Figure 35 — ASIE format**

The ASIE Specifier ID field is set to a 16-bit value that identifies a company or organization, as listed in *WiMedia Assigned Numbers* [B5].

The owner of the ASIE Specifier ID defines the format and use of the Application-specific Data field.

### 7.8.2 Application-specific Probe IE

The Application-specific Probe IE is used to request an application-specific IE from a device. It is illustrated in Figure 36.

| octets: 1 | 1 | 2 | 2 | N |
|---|---|---|---|---|
| Element ID | Length (=4+N) | Target DevAddr | ASIE Specifier ID | Application-specific Request Information |

**Figure 36 — Application-specific Probe IE format**

The Target DevAddr field is set to the DevAddr of the device from which an ASIE is requested.

The ASIE Specifier ID is set to a 16-bit value that identifies a company or organization, as listed in *WiMedia Assigned Numbers* [B5].

The owner of the ASIE Specifier ID defines the format and use of the Application-specific Request Information field.

### 7.8.3 Beacon Period Occupancy IE (BPOIE)

The BPOIE provides information on the BP observed by the device sending the IE. The BPOIE is illustrated in Figure 37.

| octets: 1 | 1 | 1 | K | 2 | … | 2 |
|---|---|---|---|---|---|---|
| Element ID | Length (=1+K+2×N) | BP Length | Beacon Slot Info Bitmap | DevAddr 1 | … | DevAddr N |

**Figure 37 —BPOIE format**

The BP Length field is set to the length of the BP, measured in beacon slots, as defined in 8.2.2.

The Beacon Slot Info Bitmap field consists of K octets of 2-bit elements to indicate the beacon slot occupancy and movability in the BP, where K = Ceiling (BP_Length/4). Each element n, numbered from 0 to 4×K-1, corresponds to beacon slot n and is encoded as shown in Table 58. Element zero is the least-significant two bits of the field. Unused elements, if any, are set to zero.

**Table 58 — Beacon Slot Info Bitmap element encoding**

| Element value | Beacon slot status |
|---|---|
| 0 | Unoccupied (non-movable)<br><br>No PHY indication of medium activity was received in the corresponding beacon slot in the last superframe, or any frame header received with a valid HCS was not a beacon frame. |
| 1 | Occupied & non-movable<br><br>A beacon frame was received with a valid HCS and FCS in the corresponding beacon slot in the last superframe, and the Movable bit in that beacon was set to zero, or a beacon frame was received in the corresponding beacon slot in a previous superframe that indicated a hibernation period that has not expired, as described in 8.13.4. |
| 2 | Occupied & movable<br><br>A PHY indication of medium activity was received in the corresponding beacon slot in the last superframe, and resulted in reception of either a frame header with invalid HCS or a beacon frame with invalid FCS. |
| 3 | Occupied & movable<br><br>A beacon frame was received with a valid HCS and FCS in the corresponding beacon slot in the last superframe, and the Movable bit in that beacon was set to one. |

The DevAddr fields correspond to beacon slots encoded as occupied in the Beacon Slot Info Bitmap. They are included in ascending beacon slot order. If a beacon was received with a valid HCS at a beacon slot in the last superframe, the corresponding DevAddr field is set to the SrcAddr in the MAC header of that received beacon. If a frame was received with an invalid HCS from a beacon slot in the last superframe, the corresponding DevAddr field is set to BcstAddr. If a neighbor of the device is in hibernation mode, the DevAddr field that corresponds to the hibernating neighbor's beacon slot is set to the DevAddr of that neighbor.

### 7.8.4 BP Switch IE

The BP Switch IE indicates a device will change its BPST to align with an alien BP. It is illustrated in Figure 38.

| octets: 1 | 1 | 1 | 1 | 2 |
|---|---|---|---|---|
| Element ID | Length (=4) | BP Move Countdown | Beacon Slot Offset | BPST Offset |

**Figure 38 —BP Switch IE format**

The BP Move Countdown field is set to the number of superframes after which the device will adjust its BPST. If BP Move Countdown is zero, the next beacon frame transmitted will be at the time specified by this IE.

The Beacon Slot Offset field is set to a positive number by which the device will adjust its beacon slot number when changing its BPST or is set to zero to indicate the device will join the alien BP using normal BP join rules.

The BPST Offset field is set to the positive amount of time the device will delay its BPST, in microseconds.

### 7.8.5 Channel Change IE

A Channel Change IE announces that a device is preparing to change to another channel.

The Channel Change IE is illustrated in Figure 39.

| octets: 1 | 1 | 1 | 1 |
|---|---|---|---|
| Element ID | Length (=2) | Channel Change Countdown | New Channel Number |

**Figure 39 — Channel Change IE format**

The Channel Change Countdown field is set to the number of superframes remaining until the device changes to the new channel. If this field is zero, the device will change to the new channel at the end of the current superframe.

The New Channel Number field is set to the channel number of the new channel to which the device will change.

### 7.8.6 Distributed Reservation Protocol (DRP) IE

A DRP IE is used to negotiate a reservation or part of a reservation for certain MASs and to announce the reserved MASs. The DRP IE is illustrated in Figure 40.

| octets: 1 | 1 | 2 | 2 | 4 | | 4 |
|---|---|---|---|---|---|---|
| Element ID | Length (=4+4×N) | DRP Control | Target/Owner DevAddr | DRP Allocation 1 | … | DRP Allocation N |

**Figure 40 — DRP IE format**

The DRP Control field is illustrated in Figure 41.

| bits: b15-b13 | b12 | b11 | b10 | b9 | b8-b6 | b5-b3 | b2-b0 |
|---|---|---|---|---|---|---|---|
| Reserved | Unsafe | Conflict Tie-breaker | Owner | Reservation Status | Reason Code | Stream Index | Reservation Type |

**Figure 41 — DRP Control field format**

The Reservation Type field is set to the type of the reservation and is encoded as shown in Table 59.

**Table 59 — Reservation Type field encoding**

| Value | Reservation Type |
|---|---|
| 0 | Alien BP |
| 1 | Hard |
| 2 | Soft |
| 3 | Private |
| 4 | PCA |
| 5–7 | Reserved |

The Stream Index field identifies the stream of data to be sent in the reservation. This field is reserved if the Reservation Type is Alien BP or PCA.

The Reason Code is used by a reservation target to indicate whether a DRP reservation request was successful and is encoded as shown in Table 60. The Reason Code is set to zero in a DRP IE sent during negotiation by a reservation owner and by a device maintaining an established

reservation. The Reason Code is set to Modified by a device if some of the MASs claimed in the reservation have been removed or if DRP IEs have been combined, split, or both. The field is reserved if the Reservation Type is Alien BP or PCA.

**Table 60 — Reason Code field encoding**

| Value | Code | Meaning |
|---|---|---|
| 0 | Accepted | The DRP reservation request is granted |
| 1 | Conflict | The DRP reservation request or existing reservation is in conflict with one or more existing DRP reservations |
| 2 | Pending | The DRP reservation request is being processed |
| 3 | Denied | The DRP reservation request is rejected or existing DRP reservation can no longer be accepted |
| 4 | Modified | The DRP reservation is still maintained but has been reduced in size or multiple DRP IEs for the same reservation have been combined |
| 5–7 | Reserved | Reserved |

The Reservation Status bit indicates the status of the DRP negotiation process. The Reservation Status bit is set to zero in a DRP IE for a reservation that is under negotiation or in conflict. It is set to one by a device granting or maintaining a reservation, which is then referred to as an established reservation.

The Owner bit is set to one if the device transmitting the DRP IE is the reservation owner, or to zero if the device transmitting the DRP IE is a reservation target. The bit is reserved if the Reservation Type is Alien BP.

The Conflict Tie-breaker bit is set to a random value of zero or one when a reservation request is made. The same value selected is used as long as the reservation is in effect. For all DRP IEs that represent the same reservation, the Conflict Tie-breaker bit is set to the same value.

The Target/Owner DevAddr field is set to the DevAddr of the reservation target if the device transmitting this DRP IE is the reservation owner. The reservation target may be a unicast or multicast DevAddr. The field is set to the DevAddr of the reservation owner if the device transmitting the DRP IE is a reservation target. The field is reserved if the Reservation Type is Alien BP or PCA.

The Unsafe bit is set to one if any of the MASs identified in the DRP Allocation fields is considered in excess of reservation limits.

A DRP IE contains one or more DRP Allocation fields. Each DRP Allocation field is encoded using a zone structure. The superframe is split into 16 zones numbered from 0 to 15 starting from the BPST. Each zone contains 16 consecutive MASs, which are numbered from 0 to 15 within the zone.

The format of a DRP Allocation field is illustrated in Figure 42.

| octets: 2 | 2 |
|---|---|
| Zone Bitmap | MAS Bitmap |

**Figure 42 — DRP Allocation field format**

The Zone Bitmap field identifies the zones that contain reserved MASs. If a bit in the field is set to one, the corresponding zone contains reserved MASs, where bit zero corresponds to zone zero.

The MAS Bitmap specifies which MASs in the zones identified by the Zone Bitmap field are part of the reservation. If a bit in the field is set to one, the corresponding MAS within each zone identified by the Zone Bitmap is included in the reservation, where bit zero corresponds to MAS zero within the zone.

### 7.8.7 DRP Availability IE

The DRP Availability IE is used by a device to indicate its view of the current utilization of MASs in the current superframe. The DRP Availability IE is illustrated in Figure 43.

| octets: 1 | 1 | N (0 to 32) |
|---|---|---|
| Element ID | Length (=N) | DRP Availability Bitmap |

**Figure 43 — DRP Availability IE format**

The DRP Availability Bitmap field is up to 256 bits long, one bit for each MAS in the superframe, where the least-significant bit of the field corresponds to the first MAS in the superframe and successive bits correspond to successive MASs. Each bit is set to one if the device is available for a DRP reservation in the corresponding MAS, or is set to zero otherwise. If the DRP Availability Bitmap field is smaller than 32 octets, the bits in octets not included at the end of the bitmap are treated as zero.

### 7.8.8 Hibernation Anchor IE

The Hibernation Anchor IE is illustrated in Figure 44.

| octets: 1 | 1 | 3 | … | 3 |
|---|---|---|---|---|
| Element ID | Length (=3×N) | Hibernation Mode Device Information 1 | … | Hibernation Mode Device Information N |

**Figure 44 — Hibernation Anchor IE format**

The Hibernation Mode Device Information field is illustrated in Figure 45.

| octets: 2 | 1 |
|---|---|
| Hibernation Mode Neighbor DevAddr | Wakeup Countdown |

**Figure 45 — Hibernation Mode Device Information field format**

The Hibernation Mode Neighbor DevAddr field is set to the DevAddr of the neighbor in hibernation mode.

The Wakeup Countdown field is set to the number of remaining superframes before the device in hibernation mode is expected to wake up. A value of zero indicates that the device is scheduled to be in active mode in the next superframe.

### 7.8.9 Hibernation Mode IE

The Hibernation Mode IE is illustrated in Figure 46.

| octets: 1 | 1 | 1 | 1 |
|---|---|---|---|
| Element ID | Length (=2) | Hibernation Countdown | Hibernation Duration |

**Figure 46 — Hibernation Mode IE format**

The Hibernation Countdown field is set to the number of superframes remaining until the device begins hibernation. A value of zero indicates that the device will enter hibernation mode at the end of the current superframe.

The Hibernation Duration field is set to the number of superframes for which the device intends to hibernate.

### 7.8.10 Identification IE

The Identification IE provides identifying information about the device, including a name string. The Identification IE is illustrated in Figure 47.

| octets:1 | 1 | $M_1$ | … | $M_N$ |
|---|---|---|---|---|
| Element ID | Length (=$M_1$+…+ $M_N$) | Device Information 1 | … | Device Information N |

**Figure 47 — Identification IE format**

The general format of the Device Information field is illustrated in Figure 48.

| octets: 1 | 1 | N |
|---|---|---|
| Device Information Type | Device Information Length (=N) | Device Information Data |

**Figure 48 — Device Information field format**

The encoding for the Device Information Type field is shown in Table 61.

**Table 61 — Device Information Type field encoding**

| Value | Device Information Data field contents |
|---|---|
| 0 | Vendor ID |
| 1 | Vendor Type |
| 2 | Name String |
| 3–255 | Reserved |

The Device Information Length field indicates the length, in octets, of the Device Information Data Field that follows.

The Device Information Data field, if Device Information Type is Vendor ID, is illustrated in Figure 49.

| octets: 3 |
|---|
| Vendor ID |

**Figure 49 — Device Information Data field format for Vendor ID**

The Vendor ID is set to an OUI that indicates the vendor of the device. The OUI is a sequence of 3 octets, labeled as oui[0] through oui[2]. Octets of the OUI are passed to the PHY SAP in ascending index-value order.

The Device Information Data field, if Device Information Type is Vendor Type, is illustrated in Figure 50.

| octets: 3 | 3 |
|---|---|
| Vendor ID | Device Type ID |

**Figure 50 — Device Information Data field format for Vendor Type**

The Vendor ID field is set to an OUI that indicates the entity that assigns the values used in the Device Type ID field. The Device Type ID field indicates the type of device.

The Device Information Data field, if Device Information Type is Name String, contains the name of the device encoded in Unicode UTF-16LE format, and is illustrated in Figure 51.

| octets: 2 | … | 2 |
|---|---|---|
| Name String Unicode Char 1 | … | Name String Unicode Char N |

**Figure 51 — Device Information Data field format for Name String**

### 7.8.11 Link Feedback IE

The Link Feedback IE contains information on the recommended change to the data rate and transmit power level by a recipient device for one or more source devices. The Link Feedback IE is illustrated in Figure 52.

| octets: 1 | 1 | 3 | | 3 |
|---|---|---|---|---|
| Element ID | Length (=3xN) | Link 1 | … | Link N |

**Figure 52 — Link Feedback IE format**

The Link field is illustrated in Figure 53.

| bits: b23-b20 | b19-b16 | b15-b0 |
|---|---|---|
| Data Rate | Transmit Power Level Change | DevAddr |

**Figure 53 —Link field format**

The DevAddr field is set to the DevAddr of the source device for which the feedback is provided.

The Transmit Power Level Change field is set to the change in transmit power level that the recipient recommends to the source device. The Transmit Power Level Change field encoding is shown in Table 62.

**Table 62— Transmit Power Level Change field encoding**

| Value | Power level change |
|---|---|
| 1000–1101 | Reserved |
| 1110 | -2 |
| 1111 | -1 |
| 0000 | no change |
| 0001 | +1 |
| 0010 | +2 |
| 0011–0111 | Reserved |

The Data Rate field is set to the data rate that the recipient device recommends that the source device use. The Data Rate field is encoded as shown in Table 63.

**Table 63 — Data Rate field encoding**

| Value | Data Rate (Mbit/s) |
|---|---|
| 0 | 53.3 |
| 1 | 80 |
| 2 | 106.7 |
| 3 | 160 |
| 4 | 200 |
| 5 | 320 |
| 6 | 400 |
| 7 | 480 |
| 8–15 | Reserved |

### 7.8.12 MAC Capabilities IE

The MAC Capabilities IE is illustrated in Figure 54.

| octets: 1 | 1 | 2 | X |
|---|---|---|---|
| Element ID | Length (=2+X) | MAC Capability Bitmap | Reserved |

**Figure 54 — MAC Capabilities IE format**

The MAC Capability Bitmap field indicates capabilities supported by the MAC entity. A bit is set to one if the corresponding attribute is supported, or is set to zero otherwise. This field is encoded as described in Table 64. Subsequent octets are reserved and may or may not be present.

**Table 64 — MAC Capability Bitmap**

| Octet | Bit | Attribute | Description |
|---|---|---|---|
| 0 | 0 | PCA | Capable of transmitting and receiving frames using the PCA mechanism |
| | 1 | Hard DRP | Capable of being the owner and target of Hard DRP reservations |
| | 2 | Soft DRP | Capable of being the owner and target of Soft DRP reservations |
| | 3 | Block ACK | Capable of transmitting and acknowledging frames using the B-ACK mechanism |
| | 4 | Explicit DRP negotiation | Capable of negotiating a DRP reservation using command frames |
| | 5 | Hibernation anchor | Capable of acting as a hibernation anchor |
| | 6 | Probe | Capable of responding to Probe IEs received in command frames |
| | 7 | Link feedback | Capable of generating and interpreting a Link Feedback IE |
| 1 | 0 | Range measurement | Capable of initiating and participating in range measurement calculations |
| | 1–7 | Reserved | Reserved |

### 7.8.13 Master Key Identifier (MKID) IE

The MKID IE is used to identify some or all of the master keys possessed by the device. The MKID IE is illustrated in Figure 55.

| octets: 1 | 1 | 16 | … | 16 |
|---|---|---|---|---|
| Element ID | Length (=16×N) | MKID 1 | … | MKID N |

**Figure 55 - MKID IE format**

Each MKID field is set to the identifier of a master key possessed by the device.

### 7.8.14 Multicast Address Binding (MAB) IE

Each device maps multicast EUI-48s to McstAddrs in the 16-bit DevAddr address range. The MAB IE declares the binding between a multicast EUI-48 and the McstAddr that the device will use when transmitting frames destined for that multicast EUI-48.

The format of the MAB IE is shown in Figure 56.

| octets: 1 | 1 | 8 | … | 8 |
|---|---|---|---|---|
| Element ID | Length (=8×N) | Multicast Address Binding Block 1 | … | Multicast Address Binding Block N |

**Figure 56 - MAB IE format**

The format of the Multicast Address Binding Block field is shown in Figure 57.

| octets: 6 | 2 |
|---|---|
| MEUI | MDevAddr |

**Figure 57 — Multicast Address Binding Block format**

The MEUI field is set to the multicast EUI-48 supplied by the MAC client at the MAC SAP.

The MDevAddr field is set to the multicast DevAddr bound to the MEUI field by the MAC entity from the McstAddr address range.

### 7.8.15 PCA Availability IE

The PCA Availability IE identifies the MASs in which a device will be available to receive PCA traffic and transmit the required response.

The PCA Availability IE is illustrated in Figure 58.

| octets: 1 | 1 | 1 | N (0 to 32) |
|---|---|---|---|
| Element ID | Length (=N+1) | Interpretation | PCA Availability Bitmap |

**Figure 58 — PCA Availability IE format**

The Interpretation field contains information that specifies the meaning of each bit in the PCA Availability Bitmap field. The Interpretation field is illustrated in Figure 59.

| bits: b7-b1 | b0 |
|---|---|
| Reserved | TIM IE Required |

**Figure 59 — Interpretation field format**

The TIM IE Required bit is set to one if the device will only be available to receive PCA traffic in the specified MASs after receiving a TIM IE that addresses it. The bit is set to zero if the device will be available to receive PCA traffic in the specified MASs regardless of TIM IE reception.

The PCA Availability Bitmap field is up to 256 bits long, one bit for each MAS in the superframe, where the least-significant bit of the field corresponds to the first MAS in the superframe and successive bits correspond to successive MASs. Each bit is set to one if the device is available to receive PCA traffic and transmit the required response in the corresponding MAS, or is set to zero otherwise. If the PCA Availability Bitmap field is smaller than 32 octets, the bits in octets not included at the end of the bitmap are treated as zero.

### 7.8.16 PHY Capabilities IE

The PHY Capabilities IE pertaining to the Multiband OFDM PHY is illustrated in Figure 60.

| octets: 1 | 1 | 3 | X |
|---|---|---|---|
| Element ID | Length (=3+X) | PHY Capability Bitmap | Reserved |

**Figure 60 — PHY Capabilities IE format**

The PHY Capability Bitmap field indicates capabilities supported by the PHY, as defined in the Multiband OFDM Physical Layer specification. A bit is set to one if the corresponding attribute is

supported, or is set to zero otherwise. This field is encoded as described in Table 65. Subsequent octets are reserved and may or may not be present.

**Table 65 — PHY Capability Bitmap**

| Octet | Bit | Attribute | Description |
|---|---|---|---|
| 0 | 0 | Band group 1 TFI | Capable of transmitting and receiving frames using TFI channels in band group 1 |
| | 1 | Band group 1 FFI | Capable of transmitting and receiving frames using FFI channels in band group 1 |
| | 2 | Band group 2 TFI | Capable of transmitting and receiving frames using TFI channels in band group 2 |
| | 3 | Band group 2 FFI | Capable of transmitting and receiving frames using FFI channels in band group 2 |
| | 4 | Band group 3 TFI | Capable of transmitting and receiving frames using TFI channels in band group 3 |
| | 5 | Band group 3 FFI | Capable of transmitting and receiving frames using FFI channels in band group 3 |
| | 6 | Band group 4 TFI | Capable of transmitting and receiving frames using TFI channels in band group 4 |
| | 7 | Band group 4 FFI | Capable of transmitting and receiving frames using FFI channels in band group 4 |
| 1 | 0 | Reserved | Reserved |
| | 1 | Band group 5 FFI | Capable of transmitting and receiving frames using FFI channels in band group 5 |
| | 2–7 | Reserved | Reserved |
| 2 | 0 | 53.3 Mb/s | Capable of receiving frames using the 53.3 Mb/s PHY data rate |
| | 1 | 80 Mb/s | Capable of receiving frames using the 80 Mb/s PHY data rate |
| | 2 | 106.7 Mb/s | Capable of receiving frames using the 106.7 Mb/s PHY data rate |
| | 3 | 160 Mb/s | Capable of receiving frames using the 160 Mb/s PHY data rate |
| | 4 | 200 Mb/s | Capable of receiving frames using the 200 Mb/s PHY data rate |
| | 5 | 320 Mb/s | Capable of receiving frames using the 320 Mb/s PHY data rate |
| | 6 | 400 Mb/s | Capable of receiving frames using the 400 Mb/s PHY data rate |
| | 7 | 480 Mb/s | Capable of receiving frames using the 480 Mb/s PHY data rate |

### 7.8.17 Probe IE

The Probe IE is used to request information from a device. It is illustrated in Figure 61.

| octets: 1 | 1 | 2 | 1 | … | 1 |
|---|---|---|---|---|---|
| Element ID | Length (=2+N) | Target DevAddr | Requested Element ID 1 | … | Requested Element ID N |

**Figure 61 — Probe IE format for standard IEs**

The Target DevAddr field is set to the DevAddr of the device from which IEs are requested or the device that requests IEs.

Each Requested Element ID field is set to the element ID of a requested IE.

### 7.8.18 Relinquish Request IE

The Relinquish Request IE is used to request that a device release one or more MASs from one or more existing reservations. It identifies the target device and the desired MASs, and is illustrated in Figure 62.

| octets: 1 | 1 | 2 | 2 | 4 | | 4 |
|---|---|---|---|---|---|---|
| Element ID | Length (=4+4×N) | Relinquish Request Control | Target DevAddr | Allocation 1 | … | Allocation N |

**Figure 62 — Relinquish Request IE format**

The Relinquish Request Control field is illustrated in Figure 63.

| bits: b15-b4 | b3-b0 |
|---|---|
| Reserved | Reason Code |

**Figure 63 — Relinquish Request Control field format**

The Reason Code field indicates the reason for the request, and is encoded as shown in Table 66.

**Table 66 — Reason Code field encoding**

| Value | Code | Meaning |
|---|---|---|
| 0 | Non-specific | No reason specified. |
| 1 | Over-allocation | The target device holds more MASs than permitted by policy. |
| 2–15 | Reserved | Reserved |

The Target DevAddr field is set to the DevAddr of the device that is requested to release MASs.

A Relinquish Request IE contains one or more Allocation fields. Each Allocation field is encoded using a zone structure. The superframe is split into 16 zones numbered from 0 to 15 starting from the BPST. Each zone contains 16 consecutive MASs, which are numbered from 0 to 15 within the zone.

The general format of an Allocation field is illustrated in Figure 64.

| octets: 2 | 2 |
|---|---|
| Zone Bitmap | MAS Bitmap |

**Figure 64 — Allocation field format**

The Zone Bitmap field identifies the zones that contain requested MASs. If a bit in the field is set to one, the corresponding zone contains requested MASs, where bit zero corresponds to zone zero.

The MAS Bitmap specifies which MASs in the zones identified by the Zone Bitmap field are part of the request. If a bit in the field is set to one, the corresponding MAS within each zone identified by the Zone Bitmap is included in the request, where bit zero corresponds to MAS zero within the zone.

### 7.8.19 Traffic Indication Map (TIM) IE

The TIM IE is used to indicate that an active mode device has data buffered for transmission via PCA. The TIM IE is illustrated in Figure 65.

| octets:1 | 1 | 2 | … | 2 |
|---|---|---|---|---|
| Element ID | Length (=2×N) | DevAddr 1 | … | DevAddr N |

**Figure 65 — TIM IE format**

Each DevAddr field is set to a valid target DevAddr for which PCA traffic is buffered.

# 8 MAC sublayer functional description

This clause specifies MAC sublayer functionality. The rules for transmission and reception of MAC frames, including setting and processing MAC header fields and information elements, are specified in 8.1.

Channel time is divided into superframes, with each superframe composed of two major parts, the beacon period (BP) and the data period. Beacon transmission and reception in the BP and merging of BPs are specified in 8.2.

During the data period devices send and receive data using prioritized contention access (PCA) or in reservations established using the distributed reservation protocol (DRP). PCA permits multiple devices to contend for access to the medium based on traffic priority, and is specified in 8.3. The DRP enables a device to gain scheduled access to the medium within a negotiated reservation, and is specified in 8.4.

Device synchronization is specified in 8.5. The fragmentation and reassembly of MSDUs is specified in 8.6. Aggregation of multiple MSDUs in a single frame is specified in 8.7. Acknowledgement mechanisms are specified in 8.8. Subclauses 8.9 through 8.15 specify probe commands, dynamic channel selection, multi-rate support, transmit power control, power management mechanisms, use of ASIEs, and range measurement. Subclause 8.16 specifies values for all MAC sublayer parameters.

## 8.1 Frame processing

This subclause provides rules on preparing MAC frames for transmission and processing them on reception. The rules cover MAC header fields and information elements.

### 8.1.1 Frame addresses

Frames are addressed using DevAddrs. There are four types of DevAddrs; Private, Generated, Multicast, and Broadcast. Table 67 shows the range for each type of DevAddr.

**Table 67 — DevAddr types and ranges**

| Type | Range |
|---|---|
| Private | 0x0000–0x00FF |
| Generated | 0x0100–0xFEFF |
| Multicast (McstAddr) | 0xFF00–0xFFFE |
| Broadcast (BcstAddr) | 0xFFFF |

A device shall associate a DevAddr of either type Private or type Generated with its local MAC entity. A device that uses a NULL EUI-48 shall use a Private DevAddr. If a device uses a Generated DevAddr, it shall select the DevAddr from the Generated DevAddr range at random with equal probability and should ensure that the generated value is unique among all devices in its extended beacon group. Selection and conflict resolution for Private DevAddrs is out of scope of this standard.

In all frames transmitted, a device shall set the SrcAddr field to its own DevAddr. In unicast frames, the DestAddr field shall be set to the DevAddr of the recipient. In multicast frames, the DestAddr field shall be set to an address from the Multicast DevAddr range, as specified in 8.1.10.13. In broadcast frames, the DestAddr field shall be set to the Broadcast DevAddr.

A device shall not transmit frames addressed to a recipient with a Private DevAddr at any time outside a Private reservation. A device with a Private DevAddr shall not transmit non-beacon frames outside a Private reservation.

#### 8.1.1.1 DevAddr Conflicts

A device with a Generated DevAddr shall recognize that its DevAddr is in conflict if any of the following conditions occurs:

— It receives a MAC header in which the SrcAddr is the same as its own DevAddr; or
— It receives a beacon frame in which the BPOIE contains a DevAddr that is the same as its own but corresponds to a beacon slot in which the device did not transmit a beacon and was not in hibernation mode.

A device that recognizes that its DevAddr is in conflict shall generate a new DevAddr to resolve the DevAddr conflict.

### 8.1.2 Frame reception

Unless otherwise indicated, a frame is considered to be received by the device if it has a valid header check sequence (HCS) and frame check sequence (FCS) as defined in 7.2.7 and indicates a protocol version that is supported by the device. The HCS is validated by the PHY, which indicates whether or not a header error occurred.[6]

A MAC header is considered to be received by the device if it has a valid HCS and indicates a protocol version supported by the device, regardless of the FCS validation.

### 8.1.3 Frame transaction

A frame transaction consists of an optional RTS/CTS frame exchange, a single frame, and the associated acknowledgement frame if requested by the ACK policy.

Figure 66 shows some frame transaction examples.

[6] For the WiMedia PHY, header errors are reported in the RXVECTOR parameter, as defined in Table 10-5 of the Multiband OFDM Physical Layer specification r1.0.

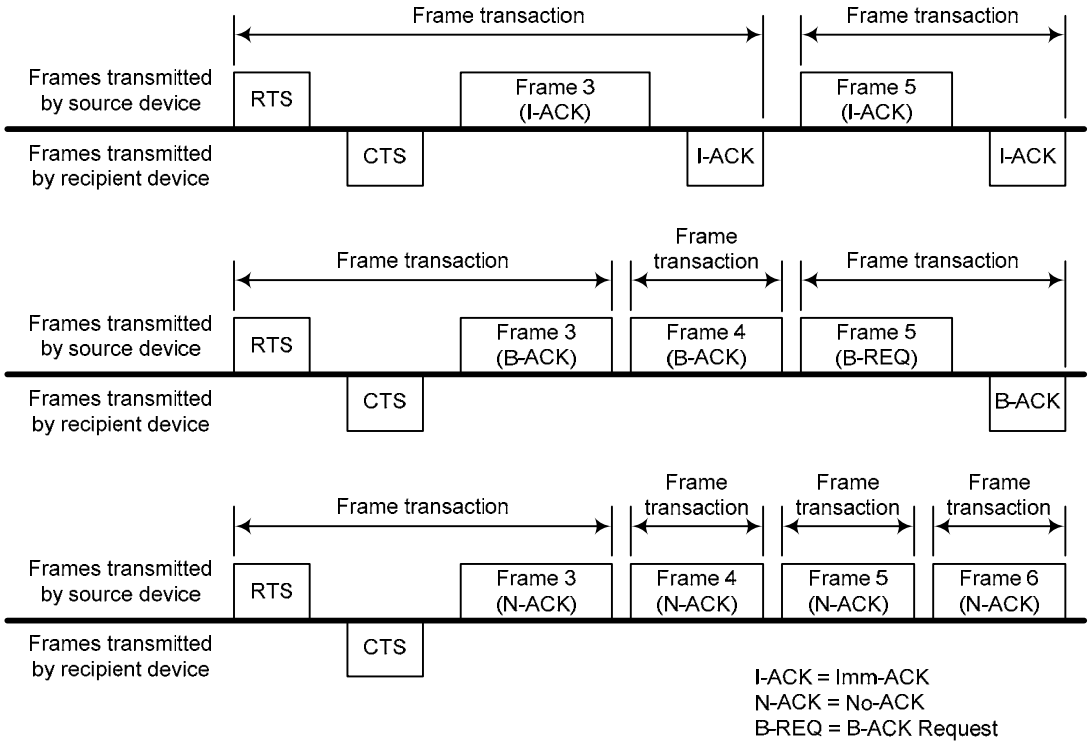



**Figure 66 — Frame transaction examples**

### 8.1.4 Frame transfer

A source device shall transmit MSDUs associated with the same Delivery ID and addressed to the same destination EUI-48 in the order in which they arrived at the local MAC SAP. The device shall treat each MSDU of length n as a sequence of octets, labelled MSDU[0] to MSDU[n-1], and shall place these octets in the payload field in ascending index-value order. The device shall transmit fragments of an MSDU or MCDU in order of increasing fragment number.

When using the B-ACK mechanism, a source device may retransmit some previously transmitted frames, causing the sequence numbers and fragment numbers of the retransmitted frames to be out of order with respect to previously transmitted frames.

A source device may reorder MSDUs for transmission if their associated Delivery IDs or destination EUI-48s are different.

A recipient device shall release MSDUs to the MAC client that were transmitted by the same source device with the same Delivery ID in order of increasing sequence number values.

A source device may fragment or aggregate MSDUs for transfer between peer MAC entities, but the recipient device shall deliver whole individual MSDUs through the MAC SAP to the MAC client.

### 8.1.5 Frame retry

A frame retry is a retransmission of a previously transmitted frame from the same source device to the same recipient device. In a frame that is retransmitted, the source device shall set the Retry bit to one.

Unless otherwise stated, in this specification “transmission” means transmission of a new frame or retransmission of a previously transmitted frame.

A device may retransmit a frame as needed, taking into consideration such factors as delay requirements, fairness policies, channel conditions, and medium availability. A device shall apply the medium access rules for new frame transmissions when retransmitting frames, unless stated otherwise.

### 8.1.6 Inter-frame space (IFS)

Three types of IFS are used in this standard: the minimum inter-frame space (MIFS), the short inter-frame space (SIFS), and the arbitration inter-frame space (AIFS[i]). There are four values of AIFS depending on the access category of the traffic. The actual values of the MIFS, SIFS, and AIFS are PHY-dependent. The derivation of the values of AIFS[i] is described in 8.3.4.1.

A device shall not start transmission of a frame on the medium with non-zero length payload earlier than MIFS, or with zero length payload earlier than SIFS, after the end of a frame it transmitted previously on the medium. A device shall not start transmission of a frame on the medium earlier than SIFS duration after the end of a previously received frame on the medium.

#### 8.1.6.1 MIFS

Burst frame transmissions are those frames transmitted from the same device where the timing of each frame in the burst after the first is related to the preceding frame through use of the PHY burst mode. In this case a MIFS duration will occur between frames in the burst, as shown in Figure 67. All frames in a burst except the last frame shall be sent with the ACK Policy field set to No-ACK or B-ACK. The last frame in a burst may be sent with any ACK Policy.

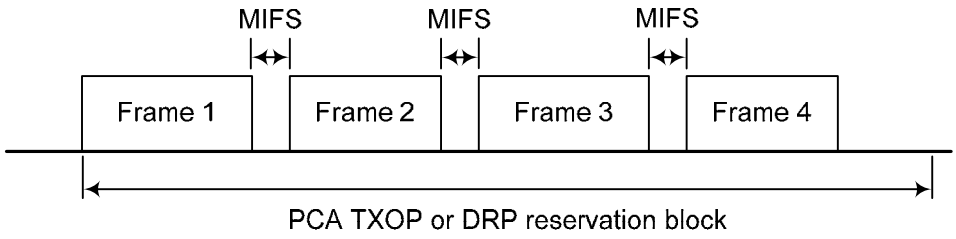



**Figure 67 — Use of MIFS**

Within a burst, the Duration field shall cover only consecutive frames addressed to the same destination. If the burst continues after the Duration is exhausted, the next frame shall use a standard preamble. The length of MIFS is given by the pMIFS parameter defined in Table 71.

#### 8.1.6.2 SIFS

Within a frame transaction, all frames shall be separated by a SIFS interval.

The length of SIFS is given by the pSIFS parameter defined in Table 71.

#### 8.1.6.3 AIFS

The AIFS is the minimum time that a device using PCA defers access to the medium after it determines the medium to have become idle.

### 8.1.7 Duplicate detection

Because a device may not receive an Imm-ACK or B-ACK response for a frame it transmitted, it may send duplicate frames even though the intended recipient has already received and acknowledged the frame. A recipient device shall consider a received frame to be a duplicate if the Retry bit is set and the Sequence Control field has the same value as the previous frame received

with the same SrcAddr, DestAddr, and Delivery ID field values. A recipient device shall not release a duplicate frame to the MAC client.

### 8.1.8 RTS/CTS use

An RTS/CTS exchange, when used, precedes data, aggregated data, or command frames to be transferred from a source device to a recipient device. Without a frame body, the RTS frame allows the source device to regain medium access relatively quickly in case of an unsuccessful transmission. With an appropriately set Duration field as specified in 8.1.9.1, the RTS and CTS frames prevent the neighbors of the source and recipient devices from accessing the medium while the source and recipient are exchanging the following frames.

A source device may transmit an RTS frame as part of one or more frame transactions with another device in an obtained PCA TXOP or an established reservation block. In a PCA TXOP, a device should transmit an RTS frame prior to transmitting a sequence of frames using the No-ACK acknowledgment policy or the B-ACK mechanism if those frame transmissions would otherwise not be covered by the Duration field contained in a frame transmitted previously between the same source and recipient devices.

If a reservation target receives an RTS frame addressed to it in the reservation block, from the reservation owner, it shall transmit a CTS frame pSIFS after the end of the received frame, regardless of its NAV setting. If a device receives an RTS frame addressed to it outside a reservation block, it shall transmit a CTS frame pSIFS after the end of the received frame if and only if its NAV is zero and the CTS frame transmission will be completed pSIFS before the start of the next BP or before the start of its own or a neighbor's established reservation block.

On receiving an expected CTS response, the source device shall transmit the frame, or the first of the frames, for which it transmitted the preceding RTS frame pSIFS after the end of the received CTS frame. If the source device does not receive the expected CTS frame pSIFS plus the CTS frame transmission time after the end of the RTS frame transmission, and it transmitted the RTS frame in a PCA TXOP, it shall invoke a backoff as specified in 8.3. If it transmitted the RTS frame in one of its reservation blocks, it shall not retransmit the RTS frame or transmit another frame earlier than pSIFS after the end of the expected CTS frame.

### 8.1.9 MAC header fields

#### 8.1.9.1 Duration

A device shall set the Duration field in beacon frames to one of the following:

— The time remaining in the BP measured from the end of the PLCP header of the beacon frame, as determined by the largest BP length announced by neighbors of the device in the previous superframe;
— The transmission time of the frame body of the beacon frame; or
— Zero.

A device shall set the Duration field in RTS, command, data, or aggregated data frames to the sum of:

— The transmission time of the frame body of the current frame;
— The transmission time of the expected response frame for the current frame (CTS, Imm-ACK, or B-ACK frame), if any;
— The transmission time of subsequent frames, if any, to be sent to the same recipient up to and including (a) the next RTS frame or frame with ACK Policy set to Imm-ACK or B-ACK Request or (b) the last frame in the PCA TXOP or reservation block, whichever is earlier; or, alternatively, the transmission time of the next frame in the PCA TXOP or reservation block to be sent to the same recipient, if any; and
— All the IFSs separating the frames included in the Duration calculation.

A device shall round a fractional calculated value for Duration in microseconds up to the next integer.

A device may estimate the transmission time of a B-ACK frame body based on the expected length and data rate, or may assume a zero-length frame body.

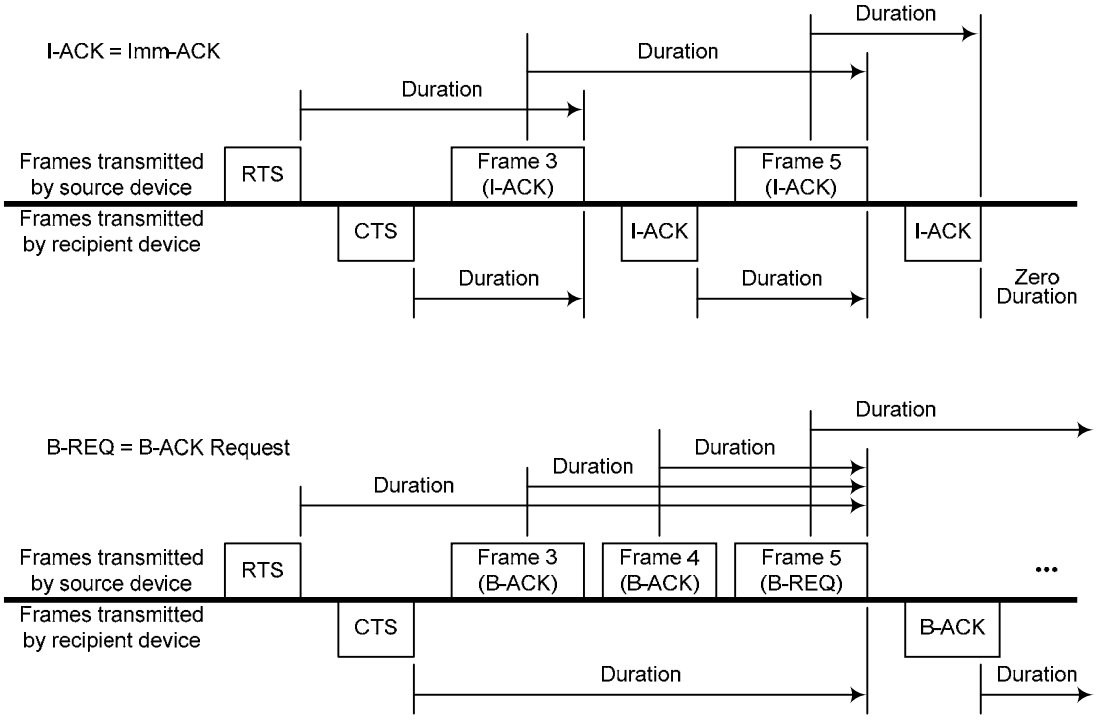
A device shall set the Duration field in CTS, Imm-ACK and B-ACK frames to the larger of zero or a value equal to the duration value contained in the previous frame minus pSIFS, minus the transmission time of the frame body of the received frame to which the CTS, Imm-ACK or B-ACK is responding, minus the transmission time up to the end of the PLCP header of this CTS, Imm-ACK or B-ACK frame.

The following exceptions to previous rules are allowed:

— For frames with ACK Policy set to B-ACK Request, a device may set the Duration to the sum of the transmission time of the frame body of the B-ACK Request frame plus a SIFS plus the estimated transmission time of the expected B-ACK response frame.
— A device may set the Duration for any frame sent in a Hard or Private reservation block other than UDA or UDR frames to zero.

A device shall set the Duration field in UDA and UDR frames to a time interval extending from the end of the PLCP header of the current frame to the time when the remaining DRP reservation block is to be released.

Examples of Duration field values are illustrated in Figure 68.



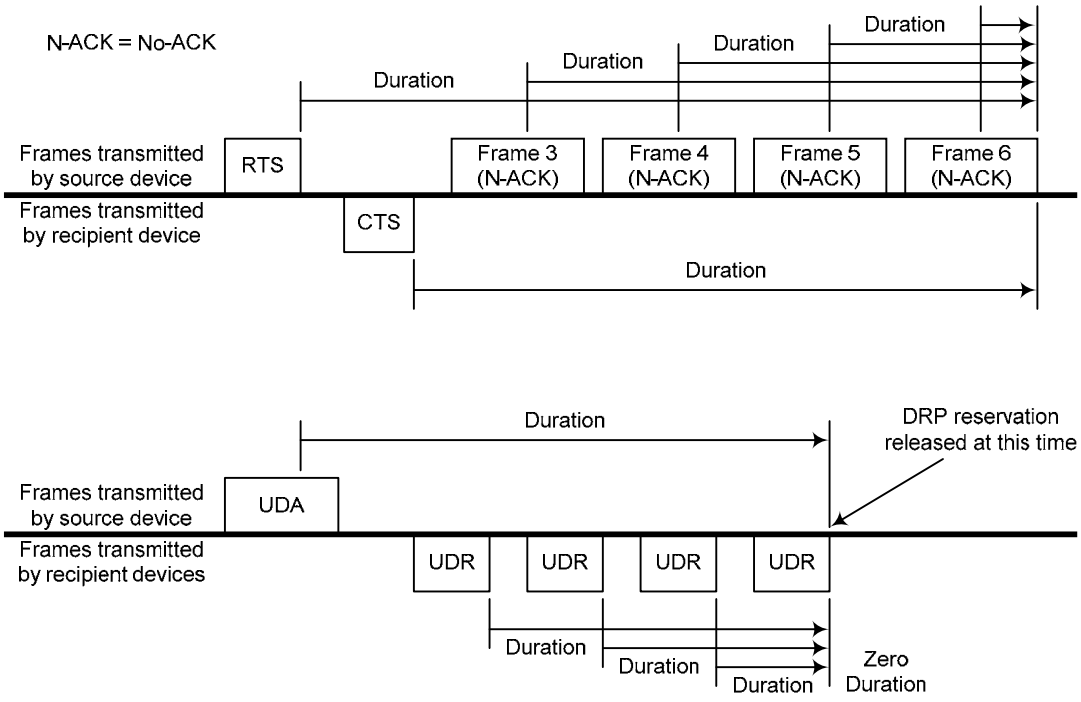



**Figure 68 — Duration examples**

#### 8.1.9.2 More Frames

If a device sets the More Frames bit to zero in a frame sent with Access Method set to one, it shall not transmit additional frames to the same recipient(s) within the reservation block.

If a device sets the More Frames bit to zero in a frame sent with Access Method set to zero, it shall not transmit additional frames using PCA to the same recipient(s) within the current superframe unless the recipient did not include a PCA Availability IE in its beacon or included a PCA Availability IE in its beacon with the TIM IE Required bit set to zero.

#### 8.1.9.3 Sequence Number

The Sequence Number field value is used for duplicate detection for frames sent using the Imm-ACK acknowledgement policy. It is used for both duplicate detection and reordering for frames sent using the B-ACK mechanism.

A device shall assign each MSDU or MCDU transmitted a sequence number from a modulo 2048 counter.

A device shall assign the same sequence number to each fragment of an MSDU or MCDU.

A single sequence number applies to all MSDUs contained in an aggregated data frame. A device shall increment the sequence number counter by one for each transmitted aggregated data frame.

A device shall use a dedicated counter for MCDUs.

A device shall use a dedicated counter for each sequence of MSDUs addressed to the same DestAddr with the same Delivery ID using B-ACK acknowledgement policy.

A device may use one counter for all other MSDUs, or may use a dedicated counter for MSDUs with the same Delivery ID field value addressed to the same DestAddr.

In each beacon frame transmitted in a superframe, a device shall set the Sequence Number field from a dedicated counter that increments once per superframe, modulo 2048, or shall set it to zero.

### 8.1.10 Information elements

IEs are contained in beacon and command frames. They convey certain control and management information. IEs may be explicitly requested using Probe command frames.

A device shall include IEs in its beacon frame such that they apply to the superframe in which the beacon is transmitted. A device shall interpret IEs contained in beacons received in the current superframe to apply to that superframe.

The remainder of this subclause describes when each IE is generated.

#### 8.1.10.1 Application-Specific IE (ASIE)

A device may include an ASIE in its beacon for each of its applications which have made the request, as described in 8.14. The scope of the ASIE is dependent on the application that requested the inclusion of the ASIE.

#### 8.1.10.2 Beacon Period Occupancy IE (BPOIE)

A device shall always include a BPOIE in its beacon. In the BPOIE the device shall reflect beacons received from neighbors in the previous superframe, as well as information retained based on hibernation mode rules.

#### 8.1.10.3 BP Switch IE

A device should include a BP Switch IE in its beacon prior to changing its BPST, as specified in 8.2.6.

#### 8.1.10.4 Channel Change IE

A device should include a Channel Change IE in its beacon prior to changing to a different channel. A device that includes a Channel Change IE should change channels as indicated in the IE.

#### 8.1.10.5 Distributed Reservation Protocol (DRP) IE

A device shall include DRP IEs in its beacon for all reservations in which it participates as a reservation owner or target, as described in 8.4.

If a device receives a frame containing a DRP IE with the Reservation Type field set to a reserved value, it shall not respond to the DRP IE, but shall treat the DRP IE as if the Reservation Type field were set to Private.

#### 8.1.10.6 DRP Availability IE

A device shall include a DRP Availability IE in its beacon as required to support DRP reservation negotiation, as described in 8.4.

#### 8.1.10.7 Hibernation Anchor IE

A device that indicates it is capable of acting as a hibernation anchor should include a Hibernation Anchor IE in its beacon to provide information on neighbors that are currently in hibernation mode as described in 8.13.5.

#### 8.1.10.8 Hibernation Mode IE

A device shall include a Hibernation Mode IE in its beacon before entering hibernation mode, as specified in 8.13.4. A device that receives a Hibernation Mode IE shall report the beacon slot of the

transmitter as occupied and non-movable in the BPOIE included in its beacons during the reported hibernation duration.

#### 8.1.10.9 Identification IE

A device may include an Identification IE in its beacon to provide its own identifying information to neighbors.

#### 8.1.10.10 Link Feedback IE

A device may include a Link Feedback IE in its beacon to provide feedback on a link with a specific neighbor.

#### 8.1.10.11 MAC Capabilities IE

A device may include a MAC Capabilities IE in its beacon.

#### 8.1.10.12 Master Key Identifier (MKID) IE

A device may include a MKID IE in its beacon to identify some or all of the master keys it possesses.

#### 8.1.10.13 Multicast Address Binding (MAB) IE

A device may include a MAB IE for any active multicast bindings between multicast EUI-48s and McstAddrs. A device should include a MAB IE in its beacon for at least mMaxLostBeacons+1 superframes on registering a multicast address binding for transmission and upon detection of a change in the beacon group.

The MAC entity shall translate the multicast EUI-48 provided by the MAC client along with an MSDU to the bound multicast DevAddr for use in the transmission of the MSDU over the medium.

A device shall not transmit frames with a McstAddr destination address unless a binding to a multicast EUI-48 has been declared by inclusion of a corresponding MAB IE in its beacon.

On receipt of a MAB IE the MAC entity shall establish an association between the source of the MAB IE and the multicast DevAddr and multicast EUI-48 in each Multicast Address Binding Block, to be used in address translations for the bound multicast addresses.

The MAC entity shall deliver received MSDUs addressed to an activated multicast DevAddr to the MAC client on the multicast EUI-48 bound to that multicast DevAddr by the source device of the MSDU.

#### 8.1.10.14 PCA Availability IE

A device may include a PCA Availability IE in its beacon as needed to facilitate PCA in the presence of reservations or power constraints.

#### 8.1.10.15 PHY Capabilities IE

A device may include a PHY Capabilities IE in its beacon.

#### 8.1.10.16 Probe IE

A device may include a Probe IE in its beacon to request certain IEs from another device.

#### 8.1.10.17 Relinquish Request IE

A device may include a Relinquish Request IE in its beacon to request that a neighbor release one or more MASs from reservations.

If a reservation target receives a request to relinquish MASs included in the reservation, it shall include in its beacon a DRP Availability IE and a Relinquish Request IE identifying those MASs with the Target DevAddr field set to the DevAddr of the reservation owner.

#### 8.1.10.18 Traffic Indication Map (TIM) IE

A device shall include a TIM IE in its beacon in any superframe when it has frames queued for transmission to one or more recipients that have the TIM IE Required bit set in a PCA Availability IE in the previous superframe. The TIM IE shall include the DevAddr of all such recipients.

## 8.2 Beacon period

Each superframe starts with a BP, which has a maximum length of mMaxBPLength beacon slots. The length of each beacon slot is mBeaconSlotLength. Beacon slots in the BP are numbered in sequence, starting at zero. The first mSignalSlotCount beacon slots of a BP are referred to as signaling slots and are used to extend the BP length of neighbors.

An active mode device shall transmit and receive beacons as described in 8.2. When transmitting in a beacon slot, a device shall start transmission of the frame on the medium at the beginning of that beacon slot.

A device shall transmit beacons at pBeaconTransmitRate. The transmission time of beacon frames shall not exceed mMaxBeaconLength. This allows for a guard time of at least mGuardTime and pSIFS between the end of a beacon and the start of the next beacon slot.

Figure 69 illustrates an example of a BP observed by a device in a given superframe.

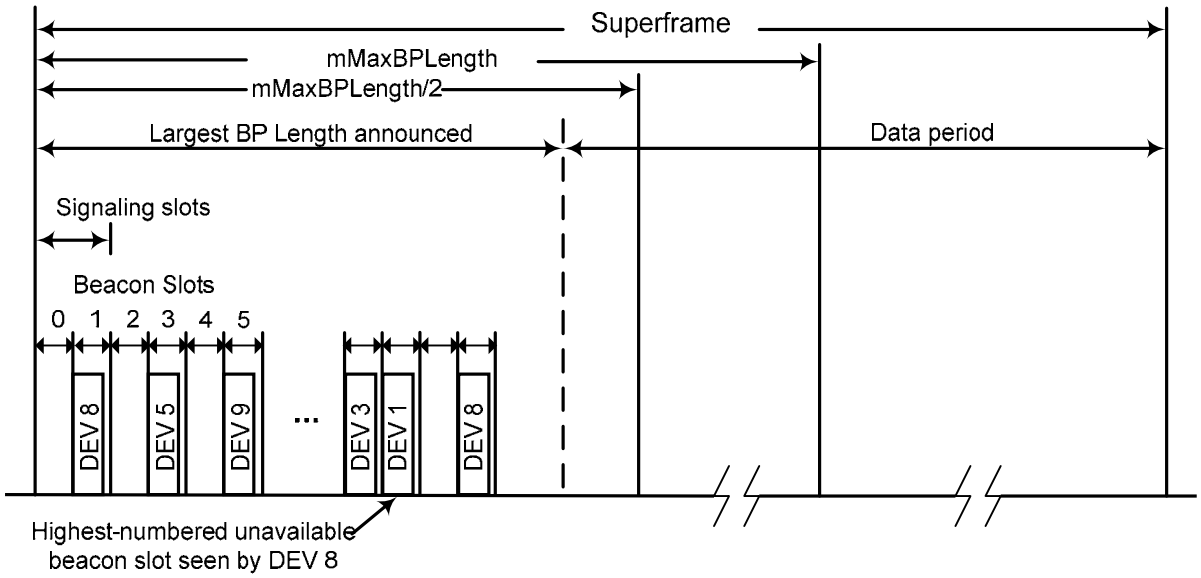



**Figure 69 — Example BP structure**

### 8.2.1 Beacon slot state

For beacon transmission and BP contraction purposes, a device shall consider a beacon slot unavailable if in any of the latest mMaxLostBeacons+1 superframes:

— The beacon slot was considered to be occupied (according to Table 58); or
— The beacon slot was encoded as occupied (according to Table 58) in the BPOIE of any beacon received by the device.

A device shall consider a beacon slot available in all other cases.

### 8.2.2 BP length

A device shall announce its BP length in its beacon as a count of beacon slots starting from the BPST. The announced BP length shall include a) the device's own beacon slot in the current superframe, b) all unavailable beacon slots in the BP of the prior superframe, and c) the beacon slot indicated in any beacon received in a signaling slot in the prior superframe.

The announced BP length shall not include more than mBPExtension beacon slots after the latest of a, b, and c, above, unless otherwise indicated in 8.2.6. The announced BP length shall not exceed mMaxBPLength. Power-sensitive devices generally should not include any beacon slots after the last unavailable beacon slot in their announced BP length.

The BP length reported by a device varies, as new devices become members of its extended beacon group, and as the device or other devices in its extended beacon group choose a new beacon slot for beacon collision resolution or BP contraction.

### 8.2.3 Beacon transmission and reception

Before a device transmits any frames, it shall scan for beacons for at least one superframe. If the device receives no beacon frame headers during the scan, it shall create a new BP and send a beacon in the first beacon slot after the signaling slots. If the device receives one or more beacon headers, but no beacon frames with a valid FCS during the scan, the device should scan for an additional superframe.

If the device receives one or more beacons during the scan, it shall not create a new BP. Instead, prior to communicating with another device, the device shall transmit a beacon in a beacon slot chosen from up to mBPExtension beacon slots located after the highest-numbered unavailable beacon slot in the last superframe and within mMaxBPLength after the BPST.

If a device detects a beacon collision as described in 8.2.4, it shall choose a different beacon slot for its subsequent beacon transmissions from up to mBPExtension beacon slots located after the highest-numbered unavailable beacon slot in the last superframe and within mMaxBPLength after the BPST.

If the beacon slot chosen for its beacon transmission is located beyond the BP length of any of its neighbors, the device shall also transmit the same beacon, except with the Signaling Slot bit set to one, in a randomly chosen signaling beacon slot in the BP, except as described in 8.2.6.1. The signaling slot is used under the above conditions regardless of whether a device is sending a beacon for the first time in an existing BP or changing the beacon slot after detecting a beacon collision. A device shall send a beacon in the signaling slot until its neighbors extend their BP lengths to include its beacon slot but only up to mMaxLostBeacons+1 superframes. After transmitting a beacon in a signaling slot for mMaxLostBeacons+1 superframes, a device shall wait for at least mMaxLostBeacons+1 superframes before sending a beacon in a signaling slot again.

If two BPs overlap as described in 8.2.6, a device may wait for a random number of superframes before sending a beacon in a signaling slot to reduce potential collisions.

An active mode device shall listen for neighbors' beacons in the first N beacon slots in each superframe, where N is the greater of its BP Length values for the current and previous superframes, as defined in 8.2.2. At a minimum, the device shall listen for intervals such that it would receive a frame with a reception time within mGuardTime of the start of any of the N beacon slots. If a device received a beacon in a signaling beacon slot in the previous superframe, it shall set its BP Length to include the beacon slot indicated in the beacon received in the signaling slot. If a device received a beacon with invalid FCS, or detected a medium activity that did not result in reception of a frame with valid HCS, in a signaling slot in the previous superframe, it shall listen for beacons for an additional mBPExtension beacon slots after its last announced BP length, but not more than mMaxBPLength beacon slots.

An active mode device shall transmit a beacon in each superframe, except as follows: In order to detect beacon collisions with neighbors, a device shall skip beacon transmission aperiodically, and listen for a potential neighbor in its beacon slot. A device shall skip beacon transmission at least every mMaxNeighborDetectionInterval. When a device skips beacon transmission, it shall act as if the skipped beacon were transmitted.

With the exception of transmitting its own beacon as described in this subclause, a device shall not transmit frames during the announced BP length of any of its neighbors.

If a device does not receive a beacon from a neighbor in the current BP, it shall use information contained in the most-recently received beacon from the neighbor as if the beacon were received in the current superframe, except when determining the contents of the Beacon Slot Info Bitmap and DevAddr fields in its BPOIE. If a device does not receive a beacon from another device for more than mMaxLostBeacons consecutive superframes, it shall not consider the device a neighbor for purposes of this specification.

### 8.2.4 Beacon collision detection

A device shall consider itself involved in a beacon collision with another device in its extended beacon group if one of the following events occurs:

— Its beacon slot is reported as occupied in the BPOIE in any beacon it receives in the current superframe, but the corresponding DevAddr is neither BcstAddr nor its own DevAddr used in the previous superframe.
— Its beacon slot has been reported as occupied and the corresponding DevAddr has been BcstAddr in the BPOIE of a beacon it received in the same beacon slot in each of the latest mMaxLostBeacons superframes.
— After skipping beacon transmission in the previous superframe, its beacon slot is reported as occupied in the BPOIE of any beacon it receives in the current superframe.
— When skipping beacon transmission in the current superframe, it receives in its beacon slot in the current superframe: a MAC header of type beacon frame, or a PHY indication of medium activity that does not result in correct reception of a MAC header.

### 8.2.5 BP contraction

A device shall consider its beacon to be movable if in the previous superframe it found at least one available beacon slot between the signaling slots and the beacon slot it indicates in its beacon in the current superframe. A device that includes a Hibernation Mode IE in its beacon shall consider its beacon to be non-movable during the announced hibernation period.

A device not involved in a beacon collision or a BP merge shall shift its beacon into the earliest available beacon slot following the signaling beacon slots in the BP of the next superframe, if in each of the latest mMaxLostBeacons+1 superframes:

— The device’s beacon was movable; and
— All the beacon slots after the device's own and within the device's BP length from the BPST were considered to be non-movable (per Table 58) or encoded as non-movable in the BPOIE of any beacon received by the device (per Table 58).

Figure 70 shows some examples of BP contraction.

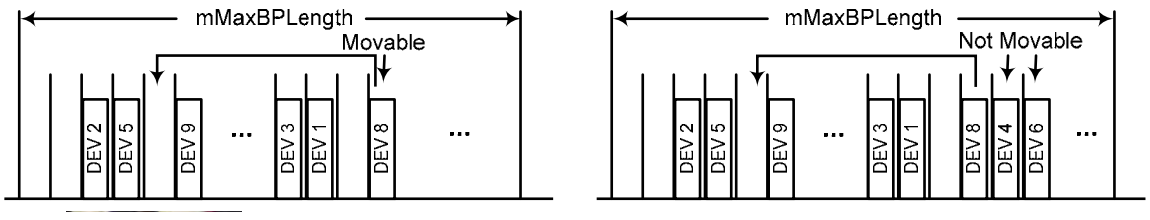



**Figure 70 — Illustration for BP contraction by example devices**

### 8.2.6 Merger of multiple BPs

Due to changes in the propagation environment, mobility, or other effects, devices using two or more unaligned BPSTs may come into range. This causes overlapping superframes. A received beacon, with valid HCS and FCS, that indicates a BPST that is not aligned with a device's own BPST is referred to as an alien beacon. The BP defined by the BPST and BP length in an alien beacon is referred to as an alien BP.

Synchronization problems could cause the beacon of a fast device to appear to be an alien beacon. A device shall consider a BPST to be aligned with its own if that BPST differs from its own by less than 2×mGuardTime. A device shall consider an alien BP to overlap its own if its BPST falls within the alien BP or if the alien BPST falls within its own BP. A device shall not consider a beacon that has the Signaling Slot bit set to one to be an alien beacon.

If a device does not receive an alien beacon for up to mMaxLostBeacons superframes after receiving one in a previous superframe, it shall use information contained in the most-recently received beacon as if the alien beacon were received at the same offset within the current superframe.

#### 8.2.6.1 Overlapping BPs

If the BPST of a device falls within an alien BP, the device shall relocate its beacon to the alien BP according to the following rules:

1. The device shall change its BPST to the BPST of the alien BP.
2. The device shall adjust its beacon slot number such that its new beacon slot number is its old beacon slot number plus one, plus the number of the highest occupied beacon slot indicated in any beacon received in the alien BP, minus mSignalSlotCount. Alternatively, it shall follow normal BP join rules as specified in 8.2.3 to relocate its beacon to the alien BP.
3. The device shall not send further beacons in its previous BP.
4. After changing its BPST, if the device is required to send a beacon in a signaling slot according to 8.2.3, it should wait for a random number of superframes before sending the beacon in the signaling slot. The device should choose the random number with equal probability in the range zero to the BP Length declared in its last beacon before relocating to the alien BP.

#### 8.2.6.2 Non-overlapping BPs

If a device detects an alien BP that does not overlap in time with its own BP, it shall merge BPs according to the following rules.

1. The device shall include in its beacon a DRP IE with Reservation Type set to Alien BP for the alien BP. Since the MAS boundaries may not be aligned, the device may need to include an additional MAS in the reservation to completely cover the alien BP. If the device received multiple beacons from the alien BP, it shall include all MASs used by the largest reported BP length in the reservation. If the MASs occupied by the alien BP change over time, the device shall update the DRP IE accordingly.
2. The device shall relocate its beacon to the alien BP, according to 8.2.6.3, within mBPMergeWaitTime if the alien BPST falls within the first half of the superframe, or within 1.5×mBPMergeWaitTime if the alien BPST falls within the second half of the superframe, but shall not relocate to the alien BP if a beacon received in that alien BP includes a BP Switch IE.

A device that transmits or receives a beacon in its own BP that contains a DRP IE with Reservation Type set to Alien BP shall observe the following rules:

1. The device should not change beacon slots except as required by merge rules in 8.2.6, unless a collision is detected.

2. If the device transmits the beacon that contains the Alien BP reservation, it shall listen for beacons during the MASs indicated in the reservation. If the device receives the beacon that contains the Alien BP reservation, it should listen for beacons during the MASs indicated in the reservation.

#### 8.2.6.3 Beacon relocation

If a device starts or has started the beacon relocation process and receives an alien beacon, it shall follow these rules:

A. If the device did not include a BP Switch IE in its last beacon, it shall include a BP Switch IE in its beacon in the following superframe with the fields set as follows:

   A1. The device shall set the BP Move Countdown field to mInitialMoveCountdown.

   A2. The device shall set the BPST Offset field to the positive difference in microseconds between the alien BPST and the device's BPST. That is, the field contains the number of microseconds that the device must delay its own BPST to align with the alien BPST. If multiple alien beacons are received, the device shall set the BPST Offset field to the largest calculated value.

   A3. The device shall set the Beacon Slot Offset field to:

      a. One plus the number of the highest occupied beacon slot indicated by any beacon received in the alien BP, based on the Beacon Slot Number field and BPOIE, minus mSignalSlotCount; or

      b. Zero to indicate the device will join the alien BP using normal join rules as specified in 8.2.3.

B. If the device included a BP Switch IE in its last beacon, it shall modify the BP Switch IE in the following superframe as follows:

   B1. If the elapsed time between the device's BPST and the following alien BPST is larger than the device's BPST Offset field + 2×mGuardTime, the device shall set the BP Move Countdown field, the BPST Offset field, and the Beacon Slot Offset field as described in A1, A2 and A3 above respectively.

   B2. If the elapsed time between the device's BPST and the following alien BPST is larger than the device's BPST Offset field - 2×mGuardTime and smaller than the device's BPST Offset field + 2×mGuardTime, the device shall set the BPST Offset field as described in A2. It shall set the Beacon Slot Offset field as described in A3 if the value in the field would be increased, or leave it unchanged otherwise. It shall set the BP Move Countdown field to one less than the value used in its last beacon if the Beacon Slot Offset field is unchanged, or set it as described in A1 if the Beacon Slot Offset field is changed.

If a device receives a neighbor's beacon that contains a BP Switch IE, it shall follow these rules:

C. If the device did not include a BP Switch IE in its last beacon, it shall include a BP Switch IE in its beacon in the following superframe with the fields set as follows:

   C1. The device shall set the BP Move Countdown field to the BP Move Countdown field of the neighbor's BP Switch IE.

   C2. The device shall set the BPST Offset field to the value of the same field contained in the neighbor's beacon.

   C3. The device shall set the Beacon Slot Offset field to:

      a. The larger of: one plus the number of the highest occupied beacon slot indicated by any alien beacon received in the alien BP identified by the neighbor's BP Switch IE, based on the Beacon Slot Number field and BPOIE,

minus mSignalSlotCount; or the Beacon Slot Offset field contained in the neighbor's beacon; or

b. Zero, to indicate the device will join the alien BP using normal join rules.

D. If the device included a BP Switch IE in its last beacon, it shall modify the BP Switch IE as follows:

D1. If the BPST Offset field contained in the neighbor's beacon is larger than the device's BPST Offset field + 2×mGuardTime, the device shall set the BP Move Countdown field, the BPST Offset field, and the Beacon Slot Offset field as described in C1, C2 and C3 above respectively.

D2. If the difference between the BPST Offset field contained in the neighbor's beacon and the device's BPST Offset field is smaller than 2×mGuardTime, the device shall modify its BP Switch IE as follows:

a. If the Beacon Slot Offset field contained in the neighbor's beacon is larger than the device's Beacon Slot Offset field, the device shall set the BP Move Countdown field, the BPST Offset field, and the Beacon Slot Offset field as described in C1, C2 and C3 above respectively.

b. If the Beacon Slot Offset field contained in the neighbor's beacon is equal to or smaller than the device's Beacon Slot Offset field, the device does not receive alien beacons from the alien BP indicated by its current BPST Offset field, and the BPMoveCountdown field contained in the neighbor's beacon is less than the device's BPMoveCountdown field, then the device shall set the BPST Offset field as described in C2 above. It shall not change the Beacon Slot Offset field. It shall set the BP Move Countdown field to one less than the value used in its last beacon.

If a device included a BP Switch IE in its beacon of the previous superframe and none of the conditions within B or D apply, the device shall not change the BPST Offset field or the Beacon Slot Offset field, and shall set the BP Move Countdown field to one less than the value used in its beacon of the previous superframe.

If a device includes a BP Switch IE in its beacon, it shall continue to do so until it completes or halts the relocation process.

— If a device receives an alien beacon that indicates relocation earlier than its planned relocation, the device shall halt the relocation process.
— If a neighbor halts the relocation process, the device shall halt the relocation process.

To halt the relocation process, a device shall include a BP Switch IE in its beacon with BPST Offset field set to 65535, Beacon Slot Offset field set to zero, and BP Move Countdown field set to mInitialMoveCountdown. In following superframes, it shall follow the rules above.

At the end of the superframe in which a device includes a BP Switch IE with a BP Move Countdown field equal to zero, the device shall adjust its BPST based on its BPST Offset field. It may transmit a beacon in that superframe, or delay one superframe to begin beacon transmission in its new BP. After relocating its beacon to the alien BP, the device shall include neither the BP Switch IE nor the alien BP DRP IE in its beacon. If the Beacon Slot Offset field was non-zero, the device shall transmit a beacon in the beacon slot with number equal to its prior beacon slot number plus the value from the Beacon Slot Offset field. If this beacon slot number is greater than or equal to mMaxBPLength, the device shall follow the normal BP join rules as described in 8.2.3 to relocate its beacon to the alien BP.

#### 8.2.6.4 BP extension

A device that receives an alien beacon with a BP Switch IE with Beacon Slot Offset field greater than zero shall set its BP length to at least the sum of the Beacon Slot Offset field and the BP length reported in the alien beacon, but not greater than mMaxBPLength.

## 8.3 Prioritized contention access (PCA)

The PCA mechanism provides differentiated, distributed contention access to the medium for four access categories (ACs) of frames buffered in a device for transmission. A device employs a prioritized contention procedure for each AC to obtain a TXOP for the frames belonging to that AC using the PCA parameters associated with that AC.

For data and aggregated data frames, the four ACs are mapped from eight user priorities as defined in Table 68.

**Table 68 — User Priority to access category mappings**

| Priority | User Priority ( Same as 802.1D User Priority) | 802.1D Designation | AC | Designation (Informative) |
|---|---|---|---|---|
| Lowest | 1 | BK | AC_BK | Background |
| | 2 | - | AC_BK | Background |
| | 0 | BE | AC_BE | Best effort |
| | 3 | EE | AC_BE | Best effort |
| | 4 | CL | AC_VI | Video |
| | 5 | VI | AC_VI | Video |
| | 6 | VO | AC_VO | Voice |
| Highest | 7 | NC | AC_VO | Voice |

For command frames, any appropriate AC may be selected.

### 8.3.1 PCA medium availability

A device shall consider the medium to be unavailable for PCA at all of the following times:

— Within the device's BP or neighbors' BPs;
— Within alien BP reservation blocks announced by itself or its neighbors;
— Within hard and private reservation blocks with Reservation Status set to one announced by itself or its neighbors, unless the reservation block has been released;
— Within soft reservation blocks with Reservation Status set to one if a neighbor is the reservation target and the reservation owner is not a neighbor, unless the device is the reservation owner; and
— For a zero-length interval at the start of soft or PCA reservation blocks with Reservation Status set to one if a neighbor is the reservation owner, for purposes of determining TXOP limits.

At all other times, a device shall consider the medium available for PCA.

### 8.3.2 NAV

A device that transmits or receives frames using PCA shall maintain a network allocation vector (NAV) that contains the remaining time that a neighbor device has indicated it will access the medium. A device that receives a MAC header not addressed to it shall update its NAV with the received Duration field if the new NAV value is greater than the current NAV value. A device shall consider the updated NAV value to start at the end of the PLCP header on the medium.

A device that receives a MAC header with invalid HCS outside its unreleased reservation blocks shall update its NAV as if the frame were correctly received with Duration equal to mAccessDelay.

A device shall reduce its NAV as time elapses until it reaches zero. The NAV shall be maintained to at least mClockResolution.

### 8.3.3 Medium status

For PCA purposes, a device shall consider the medium to be busy for any of the following conditions:

— When its CCA mechanism indicates that the medium is busy;
— When the device's NAV is greater than zero;
— When the device is transmitting or receiving a frame on the medium;
— When the Duration announced in a previously transmitted frame has not yet expired; and
— When the medium is unavailable for PCA.

At all other times a device shall consider the medium to be idle.

### 8.3.4 PCA parameters

A device shall use the set of PCA parameters defined for an AC to obtain a TXOP or perform backoff for this AC. These parameters are summarized below. The parameter values are specified in Table 70 in 8.16.

#### 8.3.4.1 AIFS[AC]

A device shall wait for the medium to become idle for AIFS[AC] before obtaining a TXOP or starting/resuming decrementing the backoff counter for the AC. AIFS[AC] is defined below:

AIFS[AC] = pSIFS + mAIFSN[AC] × pSlotTime

AIFS[AC] is related to other timings as illustrated in Figure 71 and Figure 72.

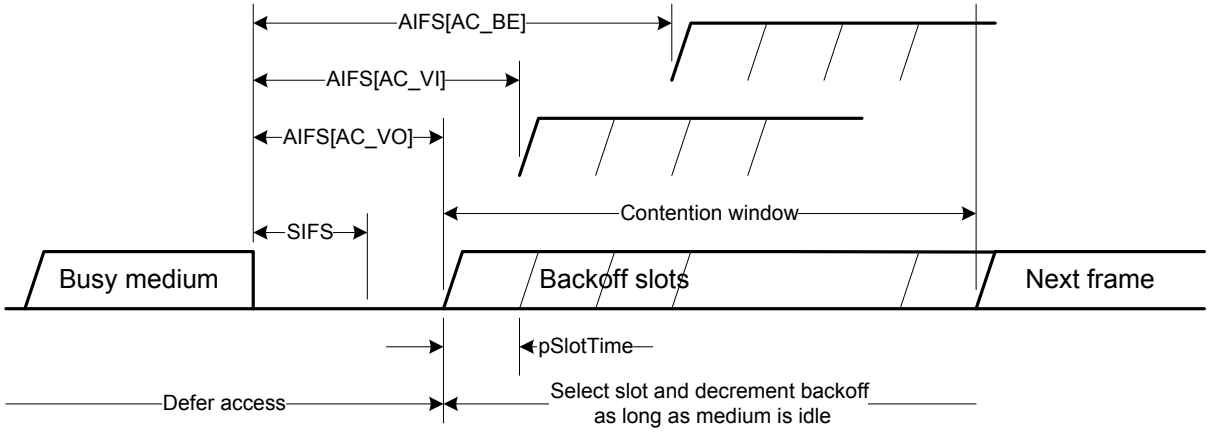



**Figure 71 — IFS relationships for PCA**

#### 8.3.4.2 mCWmin[AC] and mCWmax[AC]

A device shall set CW[AC] to an appropriate integer in the range [mCWmin, mCWmax] after invoking a backoff for the AC, and shall set the backoff counter for the AC to an integer sampled from a random variable uniformly distributed over the interval [0, CW[AC]].

#### 8.3.4.3 mTXOPLimit[AC]

A device shall not initiate a frame transaction in a TXOP it obtained for an AC unless the frame transaction can be completed within mTXOPLimit[AC] of the start of the TXOP and pSIFS plus mGuardTime before the medium becomes unavailable for PCA.

### 8.3.5 Obtaining a TXOP

A device shall consider itself to have obtained a TXOP for an AC if it meets the following conditions:

— The device has one or more newly arrived data frames or newly generated command frames belonging to this AC;
— The device had a backoff counter of zero value for this AC and had no frames belonging to this AC prior to the arrival or generation of the new frames;
— The device determines that the medium has been idle for AIFS[AC] or longer; and
— The device has no backoff counters of zero value for other ACs, or has backoff counters of zero value for some other ACs, but such ACs have a lower priority than this AC or the device has no frames belonging to those ACs that are ready for transmission.

The device shall start transmitting a frame belonging to this AC, which may be an RTS frame, as soon as the above conditions are satisfied, subject to the criteria stated in 8.3.6. The device shall treat the start of the frame transmission on the wireless medium as the start of the TXOP.

A device shall also consider itself to have obtained a TXOP for an AC if it meets the following conditions:

— The device has one or more frames belonging to this AC buffered for transmission, including retry;
— The device set the backoff counter for this AC to zero in the last backoff for this AC and determines that the medium has been idle for AIFS[AC] since that backoff at the end of the current backoff slot, or the device decrements its backoff counter for this AC from one to zero in the current backoff slot; and
— The device has no backoff counters of zero value for other ACs, or has backoff counters of zero value for some other ACs, but such ACs have a lower priority than this AC or the device has no frames belonging to those ACs that are ready for transmission.

The TXOP shall start at the end of the current backoff slot, i.e., the start of the next backoff slot.

Figure 72 illustrates the timing relationships in obtaining a TXOP.

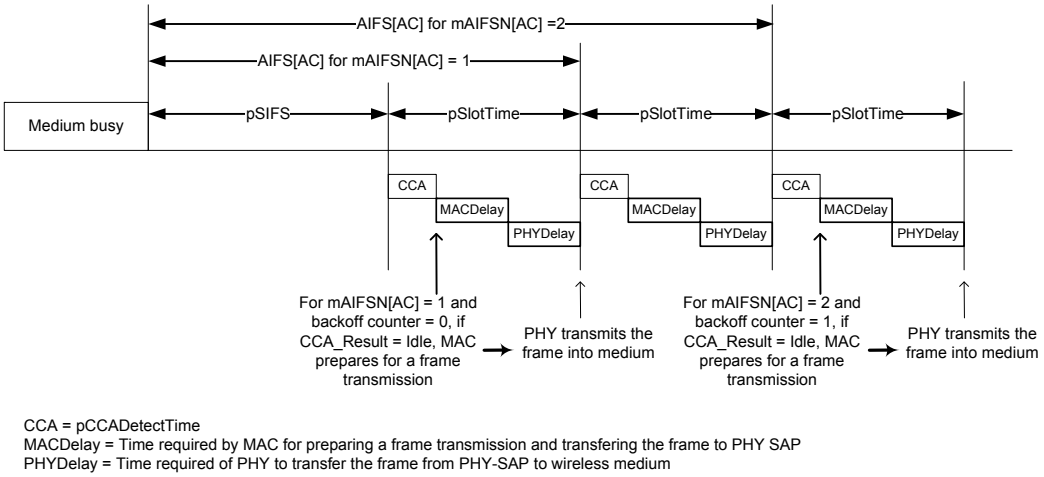



**Figure 72 — PCA timing relationships**

A device shall ensure that the TXOP it has obtained for an AC shall not be longer than mTXOPLimit[AC] and shall end pSIFS plus mGuardTime before the medium becomes unavailable for PCA.

### 8.3.6 Using a TXOP

A device that has obtained a TXOP is referred to as a TXOP owner. A TXOP owner shall initiate a frame transaction, and continue with one or more frame transactions that belong to the same AC without backoff, in the TXOP it has obtained for this AC, subject to the following criteria:

— Each transaction in the TXOP will be completed within the obtained TXOP; and
— The recipient device will be available to receive and respond during that frame transmission.

A device may retry a frame in a new TXOP that will result in the frame transaction that exceeds the mTXOPLimit[AC] restriction under the following circumstances:

— The frame is the sole frame transmitted by the device in the current TXOP; and
— The frame transaction will be completed pSIFS and mGuardTime before the medium becomes unavailable for PCA.

A recipient device shall not transmit a CTS frame in response to a received RTS frame if its NAV is greater than zero. A recipient device shall not transmit a CTS, Imm-ACK or B-ACK response to a received frame requiring such a response if the response will not be completed pSIFS before the medium becomes unavailable for its PCA.

Under the rules stated above, the following timings apply to transmissions, including responses, in a TXOP (these timings are referenced with respect to transmission to or reception from the wireless medium):

— The TXOP owner shall transmit the first frame of the first or sole frame transaction in the TXOP at the start of the TXOP.
— After transmitting a frame with the ACK Policy set to No-ACK or B-ACK, the TXOP owner shall transmit a subsequent frame pMIFS or pSIFS after the end of that transmitted frame.
— After receiving an RTS frame or a non-RTS frame with the ACK Policy set to Imm-ACK or B-ACK Request, the recipient device shall transmit a CTS frame or an Imm-ACK or B-ACK frame pSIFS after the end of the received frame.
— After receiving an expected CTS, Imm-ACK or B-ACK response to the preceding frame it transmitted, the TXOP owner shall transmit the next frame, or retransmit a frame it

transmitted earlier in the case of receiving a B-ACK, pSIFS after the end of the received frame.

— After receiving a requested B-ACK frame with a valid HCS but an invalid FCS, the TXOP owner shall retransmit the last frame it transmitted, or transmit the next frame, pSIFS after the end of the B-ACK frame.

A device shall not transmit frames using PCA to a recipient device within a MAS if the recipient has indicated in a PCA Availability IE in the current superframe that it is unavailable in that MAS. A device that indicates available MASs in a PCA Availability IE in the current superframe and does not set the TIM IE Required bit shall be available to receive frames during those MASs in that superframe if the medium is available for PCA. If the device sets the TIM IE required bit, it shall listen during indicated MASs where the medium is available for PCA in the superframe in which it received a TIM IE containing its DevAddr. If the device does not include a PCA Availability IE in its beacon, it shall be available to receive frames during all MASs available for PCA.

### 8.3.7 Invoking a backoff procedure

A device shall maintain a backoff counter for each AC to transmit frames belonging to the AC using PCA.

A device shall set the backoff counter for an AC to an integer sampled from a random variable uniformly distributed over the range [0, CW[AC]], inclusive, when it invokes a backoff for this AC. The device shall initialize CW[AC] to mCWmin[AC] before invoking any backoff for the AC, adjusting CW[AC] in the range [mCWmin[AC], mCWmax[AC]], inclusive, in the course of performing PCA for the AC as described below.

The device shall set CW[AC] back to mCWmin[AC] after receiving a CTS or Imm-ACK frame or the MAC header of a B-ACK frame expected in response to the last transmitted frame that belonged to the AC, or upon transmitting a frame with ACK Policy set to No-ACK or B-ACK that belongs to the AC. A device shall also set CW[AC] back to mCWMin[AC], but shall not select a new backoff counter value, after discarding a buffered frame belonging to the AC.

A device shall invoke a backoff procedure and draw a new backoff counter value as specified below.

1. A device shall invoke a backoff for an AC, with CW[AC] set to mCWmin[AC], when it has an MSDU arriving at its MAC SAP, or a command generated at the MAC entity, that belongs to this AC under the following conditions:
   — The device had a backoff counter of zero value for this AC but is not in the middle of a frame transaction belonging to the same AC; and
   — The device determines that the medium is busy, or the device has a backoff counter of zero value for another AC, and such an AC has a higher priority than this AC and the device has frames belonging to that AC that are ready for transmission.
2. A device shall invoke a backoff for an AC, with CW[AC] set to mCWmin[AC], at the end of transmitting a frame with the ACK policy set to No-ACK or B-ACK, or at the end of receiving an expected Imm-ACK or B-ACK response to its last transmitted frame, under the following condition:
   — The device has no other frames belonging to this AC for transmission in the current TXOP obtained for this AC.
3. A device shall invoke a backoff for an AC, with CW[AC] set to mCWmin[AC], at the end of transmitting a frame with the ACK policy set to No-ACK or B-ACK, or at the end of correctly receiving the MAC header of an expected Imm-ACK or B-ACK response frame to its last transmitted frame, under the following conditions:
   — The device has one or more frames belonging to this AC that need to be transferred over the wireless medium; and

— The device finds that there is not enough time remaining in the current TXOP obtained for this AC to complete the pending frame transaction(s) belonging to this AC.

4. A device shall invoke a backoff for an AC, with CW[AC] (but not the backoff counter in general) kept to the same value for this AC, at the start of a TXOP obtained for the AC under the following condition:

   — The device finds that there is not enough time to complete a pending frame transaction belonging to this AC in the obtained TXOP.

5. A device shall invoke a backoff for an AC, with CW[AC] set to the smaller of mCWmax[AC] or 2×CW[AC]+1 (the latter CW[AC] being the last CW value for this AC), at the end of the current backoff slot under the following conditions:

   — The device has one or more frames belonging to this AC buffered for transmission, including retry;
   — The device set the backoff counter for this AC to zero in the last backoff for this AC and determines that the medium has been idle for AIFS[AC] since that backoff at the end of the current backoff slot, or the device decrements its backoff counter for this AC from one to zero in the current backoff slot; and
   — The device has a backoff counter of zero value for another AC, and such an AC has a higher priority than this AC and the device has frames belonging to that AC that are ready for transmission.

6. A device shall invoke a backoff for an AC, with CW[AC] set to the smaller of mCWmax[AC] or 2×CW[AC]+1 (the latter CW[AC] being the last CW value for this AC), at pSIFS plus the Imm-ACK frame transmission time after the end of the last frame it transmitted, under the following condition:

   — The device does not receive an expected CTS or Imm-ACK frame, or does not correctly receive the MAC header of a requested B-ACK frame by this time.

### 8.3.8 Decrementing a backoff counter

Upon invoking a backoff for an AC, a device shall ensure that the medium is idle for AIFS[AC] before starting to decrement the backoff counter for the AC. To this end, a device shall define the first backoff slot to start at the time when the medium has been idle for pSIFS after the backoff invocation, as illustrated in Figure 72, with subsequent backoff slots following successively until the medium becomes busy. All backoff slots have a length of pSlotTime.

A device shall treat the CCA result pCCADetectTime after the start of a backoff slot to be the CCA result for this backoff slot. After the medium has been idle for AIFS[AC] since it invokes the backoff for the AC, where AIFS[AC] ends at the end of the current backoff slot, i.e., at the start of the next backoff slot, the device shall decrement the backoff counter for the AC (and its backoff counters for other ACs if appropriate) by one pCCADetectTime after the start of the backoff slot if it finds the CCA result to be idle at this time and determines the medium to be idle for the backoff slot, unless the backoff counter for the AC is already at zero value. This procedure is also illustrated in Figure 72.

The device shall freeze the backoff counter for any AC once the medium becomes busy. The device shall treat the residual backoff counter value as if the value were set due to the invocation of a backoff for the AC, following the above procedure to resume decrementing the backoff counter.

## 8.4 Distributed reservation protocol (DRP)

The DRP enables devices to reserve one or more MASs that the device can use to communicate with one or more neighbors. All devices that use the DRP for transmission or reception shall announce their reservations by including DRP IEs in their beacons (see 7.8.6). A reservation is the set of MASs identified by DRP IEs with the same values in the Target/Owner DevAddr, Owner, Reservation Type, and Stream Index fields.

Reservation negotiation is always initiated by the device that will initiate frame transactions in the reservation, referred to as the reservation owner. The device that will receive information is referred to as the reservation target.

### 8.4.1 Reservation type

Each DRP IE, whether included in a beacon or separately transmitted during explicit DRP negotiation, specifies a reservation type. A device shall decode all DRP IEs in all beacons received from neighbors and shall not transmit frames except as permitted by the reservation type. For all reservation types, a device shall not initiate a frame transaction in a reservation block if that transaction would not complete pSIFS plus mGuardTime before the end of the reservation block.

Reservation types are defined and summarized in Table 69.

**Table 69 — Reservation Types**

| Reservation Type | Description | Reference |
|---|---|---|
| Alien BP | Prevents transmission during MASs occupied by an alien BP. | 8.4.1.1 |
| Hard | Provides exclusive access to the medium for the reservation owner and target; unused time should be released for PCA. | 8.4.1.2 |
| Soft | Permits PCA, but the reservation owner has preferential access. | 8.4.1.3 |
| Private | Provides exclusive access to the medium for the reservation owner and target. Channel access methods and frame exchange sequences are out of scope of this specification; unused time should be released for PCA. | 8.4.1.4 |
| PCA | Reserves time for PCA. No device has preferential access. | 8.4.1.5 |

#### 8.4.1.1 Alien BP reservations

A device shall announce an alien BP reservation to protect alien BPs as described in 8.2.6. A device shall not transmit frames during an alien BP reservation except possibly to send a beacon in the alien BP.

#### 8.4.1.2 Hard reservations

In a hard reservation, devices other than the reservation owner and target(s) shall not transmit frames. Devices other than the reservation owner shall not initiate frame transactions. If there is remaining time in a reservation block that will not be used, the reservation owner and target(s) should release the reservation block by transmitting UDA and UDR frames as described in 8.4.8. A device may consider the remainder of a reservation block available, subject to other medium access rules, after it has received a UDA or UDR frame that releases the reservation block and the duration indicated in that received frame has expired.

A device shall not transmit a data or aggregated data frame in a hard reservation unless the Delivery ID field is set to a Stream Index that is the same as the Stream Index for the reservation and the DestAddr of the frame is the same as the Target DevAddr for the reservation or the DestAddr of the frame matches the DevAddr of any target of an established multicast reservation. A device may transmit any command or control frame in a hard reservation.

#### 8.4.1.3 Soft reservations

In a soft reservation, devices access the medium following PCA rules. The reservation owner may access the medium with the highest priority AIFS and without performing backoff. It may begin transmission at the beginning of each reservation block. It may initiate an additional frame transaction after any transaction it initiated but shall not initiate such a transaction later than pSIFS

after the end of the previous frame transaction. The reservation owner shall not transmit a data or aggregated data frame without backoff unless the Delivery ID field is set to a Stream Index that is the same as the Stream Index for the reservation and the DestAddr of the frame is the same as the Target DevAddr for the reservation or the DestAddr of the frame matches the DevAddr of any target of an established multicast reservation. The reservation owner may transmit any command or control frame without backoff. Neighbors of a reservation owner shall follow PCA rules to access the medium. Neighbors of a reservation target that are not neighbors of the reservation owner shall not access the medium.

#### 8.4.1.4 Private reservations

The channel access method and frame exchange sequences used during a private reservation are out of the scope of this standard. Standard frame formats and frame types shall be used during a private reservation. In a private reservation, neighbors of the reservation owner and target(s) shall not transmit frames. If there is remaining time unused during a reservation block, the reservation owner and target(s) should release the reservation block by transmitting UDA and UDR frames. Devices may consider the remainder of the reservation block available, subject to other medium access rules, after they have received a UDA or UDR frame that releases the reservation block and the duration indicated in that received frame has expired, as described in 8.4.8.

#### 8.4.1.5 PCA reservations

During a PCA reservation, any device may access the medium using PCA rules.

### 8.4.2 DRP Availability IE

The DRP Availability IE identifies the MASs where a device is able to establish a new DRP reservation.

The combination of information from DRP Availability IEs and DRP IEs allows an owner to determine an appropriate time for a new DRP reservation. In order to facilitate the DRP negotiation process, devices that are aware of existing neighbors' DRP reservations should mark the reserved MASs as unavailable.

A device should mark a MAS unavailable if the device includes it in a DRP IE with the Reservation Status bit set to one. It should mark a MAS unavailable if a neighbor includes it in a DRP IE with a target other than the device, whether the Reservation Status bit is zero or one. It shall mark a MAS unavailable if any BP occupies any portion of that MAS, based on information in any beacon received in the latest mMaxLostBeacons+1 superframes.

### 8.4.3 DRP reservation negotiation

There are two mechanisms used to negotiate a reservation: explicit and implicit. For explicit negotiation, the reservation owner and target use DRP Reservation Request and DRP Reservation Response command frames to negotiate the desired reservation. For implicit negotiation, the reservation owner and target use DRP IEs transmitted in their beacons. For either negotiation mechanism, the reservation owner completes the negotiation by including an appropriate DRP IE in its beacon.

A device shall not negotiate for MASs that are included in a DRP IE received from a neighbor or any other DRP IE included in the device's beacon, unless the MASs are referenced only in a DRP IE with Reason Code set to Denied.

A device shall announce in the MAC Capabilities IE in its beacon whether it is capable of explicit DRP negotiation. A device shall not initiate an explicit DRP negotiation with devices that do not support it.

A device shall only initiate negotiation for a reservation as the reservation owner.

For reservations of type Alien BP, there is no negotiation with neighbors. A device shall include the appropriate DRP IE with Reservation Status set to one on detection of an alien BP, as specified in 8.2.6.

For reservations of type PCA, there is no negotiation with neighbors. A device may select any available MAS to include in a reservation of type PCA. The device may also select MASs included in a neighbor's reservation of type PCA. The device shall not set the Reservation Status bit to one in a PCA reservation unless it included a DRP IE in its beacon in the previous superframe that identified the same MASs, with Reservation Type set to PCA and Reservation Status set to zero or one.

#### 8.4.3.1 Negotiation

When negotiating a reservation, the reservation owner shall set the Target/Owner DevAddr field of the DRP IE to the DevAddr of the reservation target. It shall set the Reservation Status bit to zero and the Reason Code to Accepted in the DRP IE. For new streams, the Stream Index shall be set to a value that is currently not used with this Target DevAddr and has not been used as such for mMaxLostBeacons+1 superframes. To negotiate additional MASs for an existing stream, the Stream Index shall be set to the value used for the existing stream.

A reservation owner shall not transmit unicast frames within reserved MASs in a hard, soft, or private reservation unless it and the recipient included DRP IEs with the Reservation Status bit set to one in their most-recently transmitted beacons.

When negotiating a reservation, a reservation target shall set the Target/Owner DevAddr field of the DRP IE to the DevAddr of the reservation owner. If a unicast reservation is granted, it shall set the Reservation Status bit to one and the Reason Code to Accepted. If a multicast reservation is granted, it shall set the Reservation Status bit to the same value included in the DRP IE by the reservation owner, and shall set the Reason Code to Accepted. If the reservation is not granted, it shall set the Reservation Status bit to zero. If the reservation cannot be granted due to a conflict with its own or its neighbors' reservations, the reservation target shall set the Reason Code to Conflict. If the reservation is not granted, it shall set the Reason Code to Denied. If the reservation target cannot grant the reservation immediately, it may set the Reason Code to Pending, and deliver a final response later. For a unicast reservation, the reservation target shall set the DRP Allocation fields to match those in the request. For a multicast reservation, it shall set the DRP Allocation fields to match the request, or to include a subset of the MASs included in the request.

#### 8.4.3.2 Explicit negotiation

To start explicit DRP negotiation, the reservation owner shall send a DRP Reservation Request command frame to the target device, as defined in 7.5.1.

On reception of a DRP Reservation Request command the reservation target shall send a DRP Reservation Response command, as defined in 7.5.2, to the reservation owner. The fields in the DRP IE shall be set according to 8.4.3.1. If the reservation cannot be granted due to a conflict with its own or its neighbors' reservations, the reservation target shall include a DRP Availability IE in the DRP Reservation Response command frame.

In a DRP Reservation Response command frame for a multicast reservation, the reservation target shall include a DRP Availability IE for a Reason Code other than Denied. Final multicast reservations are established implicitly, as described in 8.4.3.3.

#### 8.4.3.3 Implicit negotiation

Implicit negotiation is carried out by transmitting DRP IE(s) in beacon frames. A device that supports the DRP shall parse all beacons received from neighbors for DRP IE(s) whose Target/Owner DevAddr field matches either the device's DevAddr or a multicast DevAddr for which the device has activated multicast reception. From this initial selection, the device shall process the DRP IE(s) that are new with respect to DRP IE(s) included in the most recently received beacon from the same device as a DRP reservation request or a DRP reservation response.

To start implicit negotiation, a reservation owner shall include a DRP IE that describes the proposed reservation in its beacon. The device should continue to include the DRP IE for at least mMaxLostBeacons+1 consecutive superframes or until a response is received.

On reception of a unicast DRP reservation request in a beacon, the reservation target shall include a DRP reservation response in its beacon no later than the next superframe, with fields set as described in 8.4.3.1. If the Reason Code indicates Conflict, the reservation target shall include a DRP Availability IE in its beacon.

As long as the reservation owner includes a unicast DRP reservation request in its beacon, the reservation target shall continue to include the DRP reservation response in its beacon. The reservation target shall not change the Reservation Status bit to one if there is a reservation conflict with its neighbors.

On reception of a multicast DRP reservation request, a reservation target shall include a reservation response DRP IE in its beacon no later than the next superframe if it is a member of the targeted multicast group. The fields in the DRP IE shall be set according to 8.4.3.1. If the Reservation Status bit in the response is zero, the reservation target shall include a DRP Availability IE in its beacon unless the Reason Code is set to Denied.

A device that elects to receive traffic in an already established multicast reservation does not negotiate the reservation. To join an established multicast reservation that does not conflict with other existing reservations, a device shall include corresponding DRP IE(s) in its beacon with Reservation Status bit set to one and Reason Code set to Accepted.

A device that cannot join an established multicast reservation because of an availability conflict may inform the source by including the corresponding DRP IE(s) in its beacon with Reservation Status bit set to zero, and the Reason Code set to Conflict. The device shall also include the DRP Availability IE in the beacon.

#### 8.4.3.4 Negotiation conclusion

To conclude negotiation for a unicast reservation, the reservation owner shall set Reservation Status to one in the DRP IE in its beacon after receiving a beacon from the reservation target that contains a corresponding DRP IE with Reservation Status set to one. To conclude negotiation for a multicast reservation, the reservation owner may set Reservation Status to one in a DRP IE in its beacon in the next superframe after transmitting the same DRP IE with Reservation Status set to zero, regardless of responses from potential multicast recipients. If a reservation conflict exists, the reservation owner shall not set the Reservation Status bit to one except as specified in 8.4.5.

### 8.4.4 DRP reservation announcements

Once negotiation for a reservation successfully completes, the reservation owner and target shall include DRP IE(s) in their beacons that describe the reservation. Within each DRP IE, the Reason Code shall be set to Accepted and the Reservation Status bit shall be set to one. The devices shall include the DRP IEs in each beacon transmitted until the reservation is modified or terminated.

### 8.4.5 Resolution of DRP reservation conflicts

Devices engaged in independent DRP negotiation could attempt to reserve the same MAS, or due to mobility, devices could have reserved the same MAS. A conflict exists between DRP reservations if a MAS is included in both reservations. A device might detect a conflict during a DRP negotiation or after a reservation has been established. Reservations of type Alien BP never conflict with other reservations of type Alien BP. Reservations of type PCA never conflict with other reservations of type PCA.

A device shall apply the following rules to a conflict between a DRP IE included in its beacon and another DRP IE included by a neighbor:

1. If the device’s reservation is of type Alien BP, the device shall maintain the reservation.

2. If the neighbor's reservation is of type Alien BP, the device shall not transmit frames in conflicting MASs. If the device is the reservation target, it shall also set the Reason Code in its DRP IE to Conflict.
3. If the device's DRP IE has the Reservation Status bit set to zero and the neighbor's DRP IE has the Reservation Status bit set to one, the device shall not set the Reservation Status bit to one and shall not transmit frames in conflicting MASs. If the device is the reservation target, it shall also set the Reason Code in its DRP IE to Conflict.
4. If the device's DRP IE has the Reservation Status bit set to one and the neighbor's DRP IE has the Reservation Status bit set to zero, the device may maintain the reservation.
5. If the device's DRP IE and neighbor's DRP IE have the Reservation Status bit set to the same value and one of the following conditions is true, the device may maintain the reservation.
    a. The device's DRP IE and neighbor's DRP IE have the Conflict Tie-breaker bit set to the same value and the device's occupied beacon slot number is lower than the beacon slot number of the neighbor; or
    b. The device's DRP IE and neighbor's DRP IE have the Conflict Tie-breaker bit set to different values and the device's occupied beacon slot number is higher than the beacon slot number of the neighbor.
6. If the device's DRP IE and neighbor's DRP IE have the Reservation Status bit set to zero and one of the following conditions is true, the device shall not set the Reservation Status bit to one. If the device is the reservation target, it shall set the Reason Code in its DRP IE to Conflict.
    a. The device's DRP IE and neighbor's DRP IE have the Conflict Tie-breaker bit set to the same value and the device's occupied beacon slot number is higher than the beacon slot number of the neighbor; or
    b. The device's DRP IE and neighbor's DRP IE have the Conflict Tie-breaker bit set to different values and the device's occupied beacon slot number is lower than the beacon slot number of the neighbor.
7. If the device's DRP IE and neighbor's DRP IE have the Reservation Status bit set to one and one of the following conditions is true, the device shall not transmit frames in conflicting MASs. It shall remove the conflicting MASs from the reservation or set the Reservation Status to zero. If the device is the reservation target, it shall set the Reason Code in its DRP IE to Conflict.
    a. The device's DRP IE and neighbor's DRP IE have the Conflict Tie-breaker bit set to the same value and the device's occupied beacon slot number is higher than the beacon slot number of the neighbor; or
    b. The device's DRP IE and neighbor's DRP IE have the Conflict Tie-breaker bit set to different values and the device's occupied beacon slot number is lower than the beacon slot number of the neighbor.

When a reservation owner withdraws a reservation or part of a reservation due to a conflict, it shall invoke a backoff procedure prior to requesting additional MASs in any reservation. The device shall initialize the backoff window *BackoffWin* to mDRPBackoffWinMin. When the backoff algorithm is invoked, the device shall select a random number N uniformly from [0, *BackoffWin*-1]. The device shall not request additional MASs for N superframes. If a further negotiation fails due to a conflict, the device shall double *BackoffWin*, up to a maximum of mDRPBackoffWinMax. After a negotiation completes, the device shall generate a new backoff N. If a device does not request any MASs for 4×*BackoffWin* superframes, the device may terminate this backoff procedure and request MASs at any time unless another conflict occurs.

If a reservation target sets Reason Code to Conflict in any DRP IE in its beacon, it shall include a DRP Availability IE in the same beacon.

### 8.4.6 BPST realignment and existing DRP reservations

A device that realigns its BPST as described in 8.2.6 may assert new DRP reservations with Reservation Status bits set to one in the new beacon so long as they are equivalent to its old DRP reservations with the Reservation Status bit set to one in the prior BP. For this purpose, two DRP reservations are equivalent if their corresponding Target/Owner DevAddr, Stream Index, and Reservation Type fields are the same and the number of MASs claimed by the new reservation is less than or equal to the number claimed by the old reservation.

A device that realigns its BPST shall not assert DRP reservations with MASs that conflict with any BP it announced or detected. The device shall not assert DRP reservations with MASs that conflict with reservations with Reservation Status equal to one announced in the new BP unless no other MASs are available. Any conflict with existing reservations shall be resolved according to the procedures specified in 8.4.5.

### 8.4.7 Modification and termination of existing DRP reservations

A reservation owner may reserve additional MASs for a stream by negotiating an addition to the reservation using a DRP IE with the same Target/Owner DevAddr, Stream Index, and Reservation Type. Once negotiation has completed successfully, the reservation owner should combine the DRP IEs. When combining DRP IEs, the reservation owner shall set the Reason Code to Modified until a DRP IE is received from the reservation target that describes the combined reservation.

A reservation owner may remove MASs from an established reservation without changing the Reservation Status bit in the DRP IE. If a reservation owner removes some MASs from an established reservation, it shall set the Reason Code in its DRP IE to Modified until the reservation target has changed its DRP IE to match.

A reservation target may remove MASs from an established reservation without changing the Reservation Status bit in the DRP IE due to a conflict, as described in 8.4.5, or due to reception of a Relinquish Request IE. If the reservation target is unicast, the reservation owner shall remove the same MASs from the reservation or terminate the reservation in the current or following superframe.

To terminate a reservation, the reservation owner shall remove the DRP IE from its beacon.

If a reservation owner changes or removes a DRP IE, the reservation targets shall update or remove the corresponding DRP IE from their beacons in the current or following superframe.

To terminate a reservation, a reservation target shall set the Reservation Status bit to zero and the Reason Code to an appropriate value, as if responding to an initial reservation request. The reservation owner shall terminate the corresponding reservation or set the corresponding Reservation Status bit to zero in the current or following superframe.

If a reservation owner or target does not receive a beacon or any other frame from the other participant in the reservation for more than mMaxLostBeacons superframes, it shall consider the reservation terminated, and shall remove the corresponding DRP IE(s) from its beacon.

### 8.4.8 Release of hard or private DRP reservation blocks

If time remains in a hard or private DRP reservation block after a reservation owner completes transmission of associated buffered traffic, it should release the reservation block by sending a UDA frame. If the remaining time in the reservation block is not sufficient for the exchange of UDA and UDR control frames, no action should be taken. The transmitter of the UDA control frame shall include a list of device(s) that shall respond to this announcement. The list should consist of those devices that have previously included the corresponding DRP IE(s) in their beacons. The order in which the DevAddrs of the devices are mentioned in the list is the order in which they shall respond with a UDR control frame. This allows devices around the transmitter as well as the devices in the list to be informed about the early end of the reservation block.

On reception of a UDA control frame, a device shall check whether its DevAddr is included in the device list of the UDA control frame. If its DevAddr is included in the list it shall respond to the UDA control frame with a UDR control frame after a delay given by:

*Time_to_send_Response = pSIFS + pSlotTime + (Position_in_list_in_UDA) × (UDR_control_frame_duration + pSIFS)*

Time_to_send_Response is calculated from the end of reception of the UDA control frame. Possible values of Position_in_list_in_UDA are in the range [0, N-1], inclusive.

UDA and UDR control frames release the time between the end of PLCP header of the last UDR control frame, as indicated by the Duration value in the MAC header of UDA and UDR control frames, and the end of the reservation block. Other MASs described by the reservation that do not belong to the current reservation block, either in the same superframe or following superframes, are not released.

The Duration value in the UDA control frame shall cover the UDA control as well as all expected UDR control frames. The Duration value in the UDR control frames shall be set to the Duration value in the UDA control frame minus the time between the end of the PLCP header of the UDA control frame and the end of the respective UDR control frame. This value is given by the following equation:

*Duration_value_in_UDR = Duration_value_in_UDA - (UDA_frame_body_transmission_time + pSIFS + pSlotTime + (Position_in_list_in_UDA) × (UDR_control_frame_transmission_time + pSIFS)) - UDR_control_frame_transmission_time.*

### 8.4.9 Retransmit procedures in DRP reservations

In a hard DRP reservation block, if the reservation owner transmits a frame with ACK Policy set to Imm-ACK or B-ACK, but does not receive the expected acknowledgement frame, it may retransmit the frame within the same reservation block if the reservation block has not been released.

Devices other than the reservation owner that retransmit frames in a soft DRP reservation block shall follow the PCA rules defined in 8.3.

A device shall not retransmit a frame earlier than pSIFS after the end of an expected acknowledgement or CTS frame, whether or not it receives the expected frame. A device shall not retransmit a frame in the current reservation block if there is not enough time remaining in the reservation block for the entire frame transaction.

## 8.5 Synchronization of devices

Each device shall maintain a beacon period start time (BPST). The device shall derive all times for communication with its neighbors based on the current BPST. The device shall adjust its BPST in order to maintain superframe synchronization with its slowest neighbor. A device shall synchronize with such a device before it sends its first beacon.

When a device receives a beacon from a neighbor, the device determines the difference between the beacon's actual reception time and the expected reception time. The beacon's actual reception time is an estimate of the time that the start of the beacon preamble arrived at the receiving device's antenna. The expected reception time is determined from the Beacon Slot Number field of the received beacon and the receiving device's BPST. If the difference is positive, then the neighbor is slower. In order to maintain superframe synchronization with a slower neighbor, the device shall delay its BPST by the difference, but shall limit the adjustment to a maximum of mMaxSynchronizationAdjustment per superframe. The adjustment to BPST may occur at any time following the detection of a slower device, but shall be done before the end of the superframe. Earlier adjustment of the BPST allows a tighter synchronization with the slower neighbor.

A device shall not use a beacon with the Signaling Slot bit set for synchronization.

If a device does not receive a beacon from a neighbor, the device may use historical measurements to estimate the impact on superframe synchronization and increment its BPST accordingly. This estimate may be applied for up to mMaxLostBeacons consecutive superframes.

Beacon transmit time and measured beacon receive time shall be accurate to at least mClockResolution.

### 8.5.1 Clock accuracy

A device shall maintain a clock at least as accurate as mClockAccuracy. All time measurements, such as MAS boundary and frame reception time measurements, shall be measured with a minimum resolution of mClockResolution.

### 8.5.2 Synchronization for devices in hibernation mode

Devices in hibernation mode may become unsynchronized beyond the mGuardTime value during hibernation. A device in hibernation mode shall wake up at least one superframe before it will send a beacon and shall synchronize to the slowest clock in the beacon group during this superframe.

### 8.5.3 Guard times

Due to inaccuracies in superframe synchronization and drift between synchronization events, MAS start times for different devices are not synchronized perfectly. To ensure a full SIFS interval between transmissions in adjacent MASs, the devices shall maintain a SIFS interval and guard interval at the end of a reservation block. Guard times apply to all boundaries of DRP reservation blocks and BPs.

Figure 73 is an illustration of how a device uses the guard interval to maintain a SIFS interval between transmissions in adjacent reservation blocks.

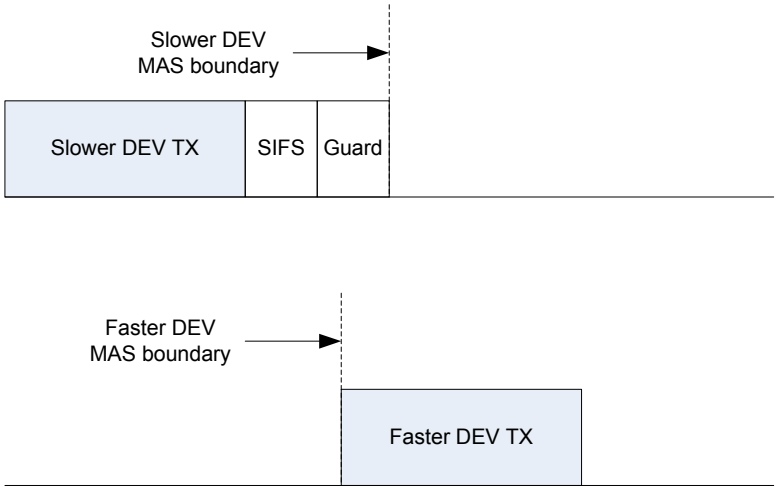



**Figure 73 — Guard Time**

The length of the guard interval, mGuardTime, depends on the maximum difference between devices' MAS boundary times. The difference arises from synchronization error and drift. The guard time is determined as follows:

*mGuardTime = MaxSynchronizationError + MaxDrift,*

where MaxSynchronizationError is the worst case error in superframe synchronization and MaxDrift is the worst case drift.

Synchronization is achieved during the BP as described in 8.5. For purposes of determining guard time, MaxSynchronizationError is calculated as twice mClockResolution.

Drift is a function of the clock accuracy and the time elapsed (SynchronizationInterval) since a synchronization event. The maximum drift, MaxDrift, is calculated using the worst case value for clock accuracy, mClockAccuracy, and the longest SynchronizationInterval:

*MaxDrift = 2 × mClockAccuracy (ppm) × 1E-6 × SynchronizationInterval,*

where

*SynchronizationInterval = (mMaxLostBeacons+1) × mSuperframeLength.*

Propagation delay will also affect timing uncertainty, but in a short-range network propagation delays are small. At 10 m range, the propagation delay is around 33 ns. This is much smaller than mClockResolution and it is ignored in calculating the length of the guard interval.

A device transmitting in a reservation block may start transmission of the preamble for the first frame at the point where it calculates the start of the reservation block to be based on its local clock. For frames that use No-ACK or B-ACK acknowledgement policy, the transmitting device shall ensure that there is enough time remaining in the reservation block to transmit the frame and allow for a SIFS plus mGuardTime before the end of the reservation block as calculated by that device.

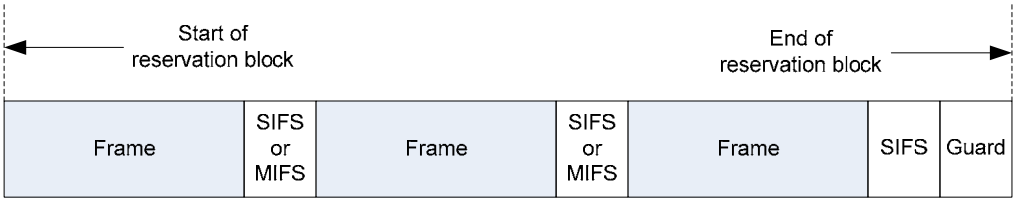



**Figure 74 — SIFS and guard time in a DRP reservation block – No-ACK**

If Imm-ACK is used, or a B-ACK is requested by the last frame, the transmitting device shall also ensure there is enough time for a SIFS interval, the ACK, another SIFS interval, and the guard time, as shown in Figure 75.

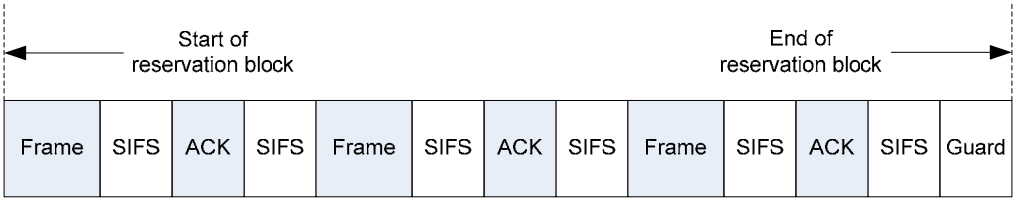



**Figure 75 — SIFS and Guard Time in a DRP reservation block – Imm-ACK**

A device shall be able to receive a frame that is transmitted within the bounds of allowable transmission, accounting for the worst case drift. A device shall begin listening mGuardTime prior to the start of a DRP reservation block, the start of a BP, or the start of a MAS in which the device announced it would be available.

## 8.6 Fragmentation and reassembly

A source device may fragment each MSDU/MCDU.

A device shall not fragment any MSDU/MCDU to more than mMaxFragmentCount fragments. Fragments may be of varying sizes. Once a device transmits a frame containing a whole MSDU/MCDU or a fragment thereof, the device shall not fragment or refragment the frame. The device shall not create frame fragments smaller than mMinFragmentSize.

The device shall set the Fragment Number field in the first fragment to zero. It shall set each subsequent fragment to the Fragment Number field in the previous fragment plus one. The device shall not increment the Fragment Number field when a fragment is retransmitted.

A device shall assign the same Sequence Number to all fragments of an MSDU/MCDU.

The device shall completely reassemble an MSDU/MCDU in the correct order before delivery to the MAC client. The device shall discard any MSDU/MCDU with missing fragments. If the No-ACK policy is used, the recipient device shall discard an MSDU/MCDU immediately if a fragment is missing. Otherwise, a recipient device shall discard the fragments of an MSDU if the MSDU is not completely received within an implementation-dependent timeout.

If B-ACK is used, unacknowledged fragments from multiple MSDUs belonging to the same stream may be retransmitted in the same sequence. In this case it is the responsibility of the recipient device to deliver the MSDUs in the correct order to the MAC client.

If a source device discards a fragment of an MSDU/MCDU, the device shall discard all fragments of the MSDU/MCDU.

## 8.7 Aggregation

A transmitter may aggregate multiple MSDUs with identical Delivery ID into a single frame payload. A device shall aggregate no more than mAggregationLimit MSDUs into an aggregated data frame.

A single MAC header is associated with an aggregated payload and it shall apply equally to each MSDU in the aggregated payload. In terms of acknowledgement and retransmission, both the transmitter and receiver shall treat the aggregated payload as a single indivisible entity. On receiving an aggregated data frame, the recipient shall deliver MSDUs from the aggregated payload to its MAC client maintaining the ordering from the payload.

The aggregated MSDUs shall be aligned on 4-octet boundaries. Prior to each MSDU, 0 to 3 pad octets shall be inserted such that the MSDU starts on a 4-octet boundary.

## 8.8 Acknowledgement policies

This clause defines three acknowledgement policies: no acknowledgement (No-ACK), immediate acknowledgement (Imm-ACK) and block acknowledgement (B-ACK).

A device shall acknowledge all received unicast frames with the ACK Policy field set to either Imm-ACK or B-ACK Request and DestAddr set to the DevAddr of this device. The device shall acknowledge the reception without regard to security validation. A device that receives a frame with a higher Protocol Version than it supports shall discard the frame without acknowledgement.

### 8.8.1 No-ACK

A frame with ACK policy set to No-ACK, as defined in 7.2.1.3, shall not be acknowledged by the recipient. The transmitting device MAC entity assumes the frame has been successfully transmitted and proceeds to the next frame upon completion of current frame. All broadcast and multicast frames shall have ACK Policy set to No-ACK.

### 8.8.2 Immediate ACK

On reception of a frame with ACK Policy set to Imm-ACK, a device shall respond with an Imm-ACK frame, as defined in 7.4.1, transmitted pSIFS after the end of the received frame.

### 8.8.3 Block ACK

The B-ACK mechanism allows a source device to transmit multiple frames and to receive a single acknowledgement frame from the recipient indicating which frames were received and which need to be retransmitted.

A source device initiates the use of the B-ACK mechanism with a recipient device for frames either from the same stream or of the same user priority. If the recipient device accepts use of the B-ACK mechanism, it indicates the maximum number and size of the frames it can buffer. The source device transmits a sequence of frames to the recipient, each from the same stream or of the same user priority, limited by the announced buffer size and maximum number of frames. The initial frames in the sequence are all transmitted with ACK Policy set to B-ACK. The final frame in the sequence is transmitted with ACK Policy set to B-ACK Request. On receipt of such a frame, the recipient device returns a B-ACK frame giving feedback on the frames received and indicating the buffer space available for the next B-ACK sequence.

A source device may invoke multiple instances of the B-ACK mechanism with the same recipient device, each for a different stream or user priority. A source device may also invoke the B-ACK mechanism with multiple recipient devices.

#### 8.8.3.1 Initiation

A source device may activate the B-ACK mechanism independently for any stream or user priority traffic to any potential recipient device advertising B-ACK capability in its MAC Capabilities IE. A source device shall initiate use of the B-ACK mechanism by transmitting a frame with ACK Policy set to B-ACK Request to the recipient device. A source device shall use a dedicated Sequence Number counter for each stream or user priority traffic using the B-ACK mechanism with a recipient. After transmitting the frame, the source device shall follow the rules of operation as described in 8.8.3.2.

When receiving a frame with ACK Policy set to B-ACK Request from a source device for a stream or user priority traffic not currently using the B-ACK mechanism, the recipient device shall respond as follows:

— To acknowledge receipt of the frame but reject the request for starting a new instance of B-ACK mechanism, the recipient device shall respond with a B-ACK frame with no frame payload.
— To accept the request for starting a new instance of B-ACK mechanism, the recipient device shall respond with a B-ACK frame with a frame payload indicating the allowed maximum size (in frames and octets) for the next B-ACK sequence. The recipient shall acknowledge the received frame by indicating its reception in the acknowledgement window.

A recipient device may also accept a request to use the B-ACK mechanism even if the request frame has an invalid FCS. To accomplish this, the recipient device shall respond with a B-ACK frame with a frame payload that indicates the allowed maximum size for the next B-ACK sequence, but without acknowledgement of the frame with the invalid FCS.

A recipient device, even though it advertises B-ACK capability in its MAC Capabilities IE, may reject a request to use the B-ACK mechanism for any reason, including a temporary unavailability of resources or a lengthy setup process requiring a delayed start time. Thus, after being rejected, a source device may keep trying to initiate use of the B-ACK mechanism by sending the next frame with ACK Policy set to B-ACK Request.

#### 8.8.3.2 Operation

After transmitting a frame with ACK Policy set to B-ACK Request, the source device expects to receive a B-ACK frame in response and takes one of the following actions:

— If the source device does not receive a B-ACK frame, it shall assume that the recipient device did not receive the request frame. To continue B-ACK operation, the source device shall retransmit the request frame with the same ACK Policy using applicable medium access rules as described in 8.3 and 8.4.
— If the source device receives a B-ACK frame with no frame payload, it shall treat the transmitted frame as received and consider this use of the B-ACK mechanism to be terminated. The B-ACK provides no acknowledgement of any other frames.

— If the source device receives a B-ACK frame with a frame payload and with either Frame Count or Buffer Size set to zero, it shall process the acknowledgement as described below. To continue the B-ACK operation, the source device shall retransmit the requesting frame with the same ACK Policy, independently of whether the frame was indicated as received or not. If the requesting frame was indicated as received, the source device alternatively may transmit a zero-length payload frame with the same Sequence Control and Delivery ID to the recipient device.
— If the source device receives a B-ACK frame with a frame payload containing non-zero values for both Frame Count and Buffer Size, then it shall process the acknowledgement as described below. To continue the B-ACK operation, the source device shall send frames with ACK Policy set to B-ACK or B-ACK Request as described below.

The source device processes the B-ACK frame acknowledgement as follows:

— Frames being held for retransmission with a sequence number earlier than the one indicated by the Sequence Control field were not received in the last B-ACK sequence, but shall not be retransmitted.
— Frames being held for retransmission with sequence and fragment number within the acknowledgement window (specified by the Sequence Control field and the Frame Bitmap field) with corresponding bit set to one were received and shall not be retransmitted.
— Other frames being held for retransmission should be retransmitted in the next sequence, ordered by increasing sequence and fragment numbers.

After receiving a B-ACK frame with non-zero values for Frame Count and Buffer Size, the source device may transmit a sequence of frames. Each sequence of frames shall consist of zero or more frames with ACK Policy set to B-ACK followed by a single frame with ACK Policy set to B-ACK Request. The total number of frames must not exceed the Frame Count value specified in the B-ACK frame and the sum of the lengths of the frame payloads shall not exceed the Buffer Size value specified in the B-ACK frame. The sequence of frames may be transmitted in multiple PCA TXOPs or DRP reservation blocks and may be interleaved with frames to other recipients or of other streams or user priorities, subject to all the medium access rules. Within a sequence, the frames shall be ordered by increasing sequence and fragment numbers. Due to retransmissions, this ordering might not hold from one sequence to the next and frames transmitted within a sequence might not have consecutive sequence and fragment numbers.

When the recipient device receives a frame with ACK Policy set to B-ACK Request, it shall respond using SIFS with a B-ACK frame. To continue operation, the B-ACK frame shall contain a frame payload. If the recipient device receives a frame with a valid HCS but an invalid FCS and with ACK Policy set to B-ACK Request, the device shall also respond with a B-ACK frame with a frame payload. Within the B-ACK frame payload, the recipient device shall set the Frame Count and Buffer Size fields to limit the size of the next sequence of frames. It shall also set the Sequence Control and Frame Bitmap fields to indicate to the source device which frames should be retransmitted.

A recipient device may implement a timeout that indicates when to stop waiting for missing frames, allowing some MSDUs to be released to the MAC client and B-ACK buffer resources to be freed. A recipient device may also implement a timeout to expire an instance of the B-ACK mechanism that appears to be inactive.

#### 8.8.3.3 Termination

To terminate use of the B-ACK mechanism, the source device shall transmit a frame from the appropriate stream or of the appropriate user priority to the recipient device with ACK Policy set to anything other than B-ACK or B-ACK Request.

The recipient device may terminate use of the B-ACK mechanism by responding to a frame with ACK Policy set to B-ACK Request with a B-ACK frame with no frame payload.

## 8.9 Probe

The Probe IE and Application-specific Probe IE may be used in beacons and probe commands to request one or more IEs from the target device identified in the probe IE. Target devices are not required to respond with all requested IEs. If a target device supports the Probe command frame for one or more IEs, it shall set the Probe bit in its MAC Capabilities IE to one, or otherwise it shall set the bit to zero.

A device shall include a MAC Capabilities IE or a PHY Capabilities IE in its beacon if it is the target of a Probe IE received in a beacon that includes the MAC Capabilities IE Element ID or the PHY Capabilities IE Element ID, respectively.

On reception of either probe IE in a beacon, a target device shall include a response in its beacon for the next mMaxLostBeacons superframes.

On reception of either probe IE in a Probe command frame, a target device should respond with a Probe command frame addressed to the sender within one superframe or include a response in its beacon for the next mMaxLostBeacons superframes.

In the Probe command frame or beacon, the target device shall include:

— A Probe IE, with Target DevAddr set to the DevAddr of the requestor, that includes no Requested Element IEs to reject the probe; or
— One or more requested IEs.

## 8.10 Dynamic channel selection

Dynamic channel selection is a process that allows a group of devices to change channels in a coordinated manner.

A device may initiate dynamic channel selection after it has performed a channel scan, as described in 8.2.3. If a device initiates dynamic channel selection, it shall include a Channel Change IE in its beacon sent in the current channel, as described in 7.8.5.

In a Channel Change IE, the device shall set the New Channel Number field to the number of the new channel. It shall set the Change Channel Count field to the remaining number of superframes before the device will move to another channel. In successive superframes, the Change Channel Count field should be decremented.

If the value set in the Change Channel Count field is zero, the device shall move to the new channel at the end of the current superframe.

On reception of the Channel Change IE, a device that also intends to change channels in a coordinated manner should include a Channel Change IE with the same field values in its beacon.

## 8.11 Multi-rate support

A device shall transmit beacons at pBeaconTransmitRate.

Devices shall transmit non-beacon frames only at data rates supported by the intended recipient, based on information from the recipient's PHY Capabilities IE.

A recipient device may use the Link Feedback IE to suggest the optimal data rate to be used by a source device, for example, to increase throughput and/or to reduce the frame error rate. The data rate in the Link Feedback IE should be interpreted as the maximum data rate that the source device should use for this particular link, for an acceptable frame error rate. The source device is not required to follow the recommendation. The method to determine the optimal data rate in the recipient is beyond the scope of this standard.

## 8.12 Transmit power control

A device shall not transmit frames at a higher transmit power level than that used for its most-recently transmitted beacon.

A recipient device may recommend a transmit power level change to be used by a source device by including a Link Feedback IE in its beacon. A device that receives a Link Feedback IE is not required to change its transmit power level. The method to determine transmit power recommendations is out of the scope of this standard, but the recipient device might use the signal to noise ratio, received signal strength, frame error ratio or other parameters to determine the transmit power change to recommend to the source device.

## 8.13 Power management mechanisms

### 8.13.1 Power management modes

A device may be in one of two power management modes in each superframe:

— Active mode: the device will send and receive beacon(s) in the current superframe.
— Hibernation mode: the device will not send a beacon or other frames in the current superframe.

Before entering hibernation mode, a device shall announce in previous superframe(s) that it is entering hibernation mode.

### 8.13.2 Device power states

Active mode devices may be in one of two power states within a superframe:

— Awake: device is able to transmit and receive.
— Sleep: device does not transmit or receive.

A device that is changing from Sleep to Awake state and has a frame in queue to transmit using PCA shall perform CCA until a MAC header is received or up to mAccessDelay time, before determining the medium state.

The value of mAccessDelay is equal to the time required to transmit one maximum length frame, transmitted at the lowest mandatory data rate, plus the time to transmit an Imm-ACK plus pSIFS.

### 8.13.3 Power state transitions

Active mode devices shall transition between Awake and Sleep states according to the following rules:

1. A device shall be in the Awake state mGuardTime prior to its BPST in every superframe to participate in the transmission and reception of beacons.
2. If a device has data traffic pending to be transmitted in DRP reservations in the current superframe, it shall be in Awake state prior to the start of each relevant DRP reservation block to start its transmission. The device may go into Sleep state for the rest of the reservation block if all the pending transmissions completed successfully. The device should release the DRP reservation block before entering Sleep state. A device shall set the More Frames bit to zero in the last frame it transmits during a reservation block.
3. If a device has data traffic pending to be transmitted via PCA in the current superframe, it shall signal its intent with a relevant TIM IE in its beacon. Once all of its transmissions are completed, the device may go into Sleep state for the rest of the superframe, subject to other rules in this subclause. A device shall set the More Frames bit to zero in the last frame it transmits to a particular recipient using PCA during a superframe.
4. If a device is expecting to receive transmissions from other devices in a DRP reservation block, as indicated in the beacons of those devices, it shall be in Awake state mGuardTime

prior to the start of the reservation block for the reception of the planned transmission. It may go into Sleep state either at the end of the reservation block or when all the pending data has been received, as indicated by the More Frames bit. If the device receives a UDA frame, it shall not go into Sleep state until after transmitting a corresponding UDR frame.

5. If a device is expecting to receive transmissions from other devices via PCA, it may include a PCA Availability IE in its beacon and shall be in Awake state mGuardTime prior to the announced MASs. A device that does not include a PCA Availability IE in its beacon shall be in Awake state in all MASs available for PCA in the current superframe. Once all pending data has been received, as indicated by the More Frames bit, the device may go into Sleep state for the rest of the superframe.

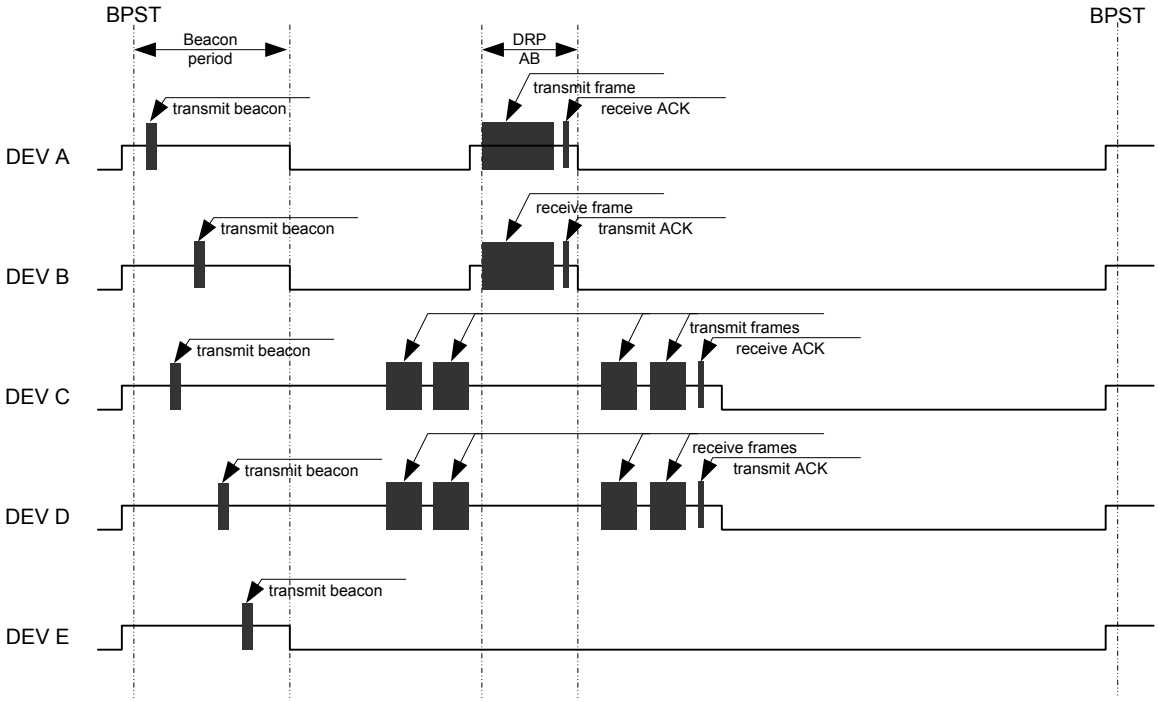



**Figure 76 — Power state transition for devices in active mode**

Figure 76 illustrates the power state transition for devices in active mode.

— DEV A is a device that has data traffic pending to be transmitted in a DRP reservation block in the current superframe.
— DEV B is a device that is expecting to receive a planned transmission from DEV A in a DRP reservation block in the current superframe.
— DEV C is a device that has data traffic pending to be transmitted via PCA in the current superframe.
— DEV D is a device that is expecting to receive a planned transmission from DEV C via PCA in the current superframe.
— DEV E is a device that does not have any traffic pending in its transmission queues, and is not expecting any planned transmission from other devices.

### 8.13.4 Hibernation mode operation

A device using hibernation mode shall operate according to the following rules:

— A device shall signal its intent to go into hibernation mode by including a Hibernation Mode IE in its beacon, as defined in 7.8.9. The Hibernation Duration field in the Hibernation Mode IE shall contain a non-zero value that specifies the duration of the hibernation period.

  A device may signal its intent to go into hibernation mode in several superframes. The value of the Hibernation Countdown field in the Hibernation Mode IE shall be set to indicate the number of remaining superframes before the device enters hibernation mode. In each successive superframe, the device shall reduce the value of the Hibernation Countdown field by one. If this field is set to zero, the device enters hibernation mode at the start of the next superframe.

— When in hibernation mode, the device shall not send a beacon or other traffic. The device should terminate all established DRP reservations before entering hibernation.

— A device may leave hibernation mode prior to the end of its announced hibernation period by sending its beacon.

— A device in hibernation mode shall scan for beacons during the BP for one or more superframes immediately prior to the end of its hibernation period, in order to re-establish synchronization.

— If a device exiting hibernation mode finds that its former beacon slot is neither occupied (per Table 58) nor encoded as occupied (per Table 58) with a DevAddr not its own in the BPOIE of any beacon received by the device in the last superframe, the device may transmit a beacon in that beacon slot. Otherwise, the device shall transmit a beacon as if it were doing so for the first time.

Active mode devices in the presence of hibernation mode devices shall operate as follows:

— If an active mode device receives a neighbor’s beacon that includes a Hibernation Mode IE, the device shall consider all DRP reservations with that neighbor to be terminated at the start of its hibernation period. An active mode device shall not commence any communication with a hibernation mode device until that device leaves hibernation mode. After receiving a beacon that includes a Hibernation Mode IE with Hibernation Countdown less than or equal to mMaxLostBeacons, an active mode device that misses the remaining expected beacons shall consider the device to be in hibernation mode as indicated in the Hibernation Mode IE.

— If a device does not receive a beacon from a neighbor at the end of the hibernation duration indicated in the neighbor’s Hibernation Mode IE, it shall treat the neighbor’s beacon slot as occupied, but shall not indicate it as occupied in its BPOIE, for up to mMaxHibernationProtection after the neighbor entered hibernation or until any beacon is received in the neighbor’s beacon slot.

  During a neighbor's hibernation period an active mode device shall continue to mark the hibernation mode device’s beacon slot as occupied and non-movable in its BPOIE. If the active mode device receives another neighbor's beacon in the hibernation mode device’s beacon slot, the device shall still advertise the hibernation mode device’s DevAddr in its BPOIE.

— If an active mode device has unicast traffic for a hibernation mode device, it should buffer its traffic until the hibernation mode device enters active mode.

— If an active mode device has multicast or broadcast traffic it should not delay transmission of the traffic, even if it is aware that some intended recipients are in hibernation mode. It may buffer its multicast traffic for a hibernation mode device until the intended recipient enters active mode, and then deliver the buffered multicast data.

### 8.13.5 Hibernation anchor operation

Active mode devices that are capable of acting as a hibernation anchor should indicate hibernation anchor capability in its MAC Capabilities IE. A device that indicates such capability should include a

Hibernation Anchor IE in its beacon to convey information about neighbors in hibernation mode. A device may terminate its role as a hibernation anchor at any time, but at that time it should remove indication of the capability from its MAC Capabilities IE.

Devices, such as those that were recently off or in hibernation mode, may not have information about the hibernation state of their neighbors. These devices may use the information provided by Hibernation Anchor IEs for scheduling communication with neighbors in hibernation mode.

Upon reception of a beacon containing a Hibernation Mode IE in which the Hibernation Countdown is set to zero, a hibernation anchor should include a Hibernation Anchor IE. It shall set the Wakeup Countdown field in the Hibernation Anchor IE based on the Hibernation Duration field in the received Hibernation Mode IE. It shall decrement the Wakeup Countdown field in each successive superframe until the field reaches zero. After it transmits a beacon with a Hibernation Anchor IE that contains a Hibernation Mode Device Information field with Wakeup Countdown set to zero, it shall remove the corresponding Hibernation Mode Device Information field from the Hibernation Anchor IE. It shall not include a Hibernation Anchor IE if there are no Hibernation Mode Device Information fields in the IE.

If the hibernation anchor receives a beacon from a hibernation mode device prior to the end of the announced hibernation duration, the hibernation anchor shall remove the corresponding Hibernation Mode Device Information field from the Hibernation Anchor IE in the next beacon.

After receiving a neighbor's beacon that includes a Hibernation Mode IE with Hibernation Countdown less than or equal to mMaxLostBeacons, a hibernation anchor device that misses the remaining beacons from the neighbor shall consider the device to be in hibernation mode as indicated in the Hibernation Mode IE and should include that device in the Hibernation Anchor IE.

## 8.14 ASIE operation

Zero or more ASIEs may be included in each beacon. ASIEs may appear within the IE area in a beacon as defined in 8.1.10. Unrecognized ASIEs shall be ignored. The format of the ASIE payload is defined by the owner of the value in the ASIE Specifier ID field and is outside the scope of this document.

## 8.15 Range measurement operation

This subclause describes the support for cooperative two-way time transfer measurements.

The support of range measurement is optional. A device that is capable of initiating and participating in range measurements shall declare Range Measurement capability in its MAC Capabilities IE. A device shall not initiate a range measurement with a neighbor that does not declare it is capable of range measurement.

If the MAC entity is instructed to initiate one or more consecutive ranging measurements with a specified remote device, it shall use Range Measurement command frames as described in 7.5.6.

In order to perform a range measurement, the initiator device shall turn on the PHY range timer and send a Range command frame of type Range Measurement Request to the recipient device. The ACK Policy field shall be set to Imm-ACK in the Range command frames of type Range Measurement Request. The request contains the number of range measurements to be performed in the Requested Measurement Number field. Upon acknowledgement of the request, the initiator shall send the Range command frame of type Range Measurement to the recipient device with the ACK Policy field set to Imm-ACK.

For each Range command frame of type Range Measurement the initiator device shall:

1. Capture the PHY range timer value at the time of transmission of the Range command frame of type Range Measurement, T1, and add the calibration constant: T1c = T1 + pRangingTransmitDelay.
2. Capture the PHY range timer value, R2, at the time of reception of the Imm-ACK and subtract the calibration constant: R2c = R2 - pRangingReceiveDelay.

Upon reception of the Range command frame of type Range Measurement Report, the initiator passes the values in the report to the DME. The PHY range timer may be turned off.

If a device supports range measurement and receives a Range command frame of type Range Measurement Request, it shall turn on the PHY range timer. For each Range command frame of type Range Measurement received, the recipient shall:

1. Capture the PHY range timer value = R1 at the time of reception of the Range command frame of type Range Measurement and subtract the calibration constant: R1c = R1 - pRangingReceiveDelay.
2. Capture the PHY range timer value at the time of transmission of the Imm-ACK, T2 and add the calibration constant: T2c = T2 + pRangingTransmitDelay.

The recipient shall further send a Range command frame of type Range Measurement Report to the initiator after reception of all Range command frames of type Range Measurement. The number of Range command frames of type Range Measurement is specified in the Requested Measurement Number field in the Range command frame of type Range Measurement Request. After the report command frame is sent, the PHY range timer may be turned off.

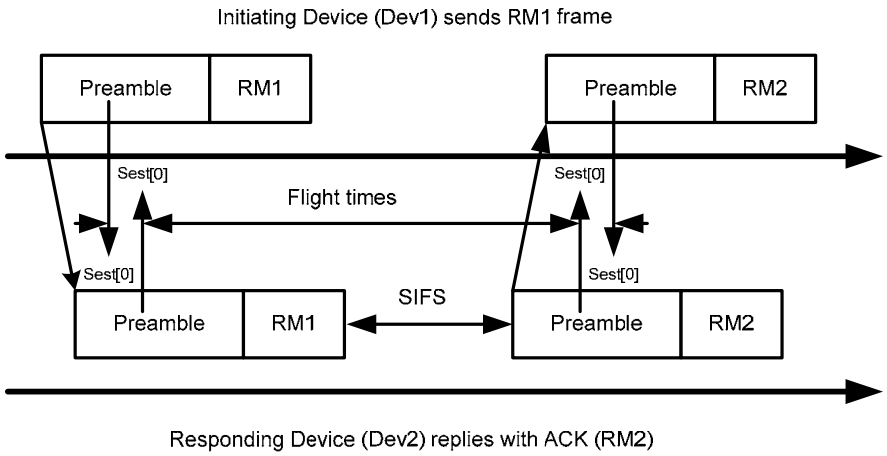



**Figure 77 — Example of range measurement frame pair**

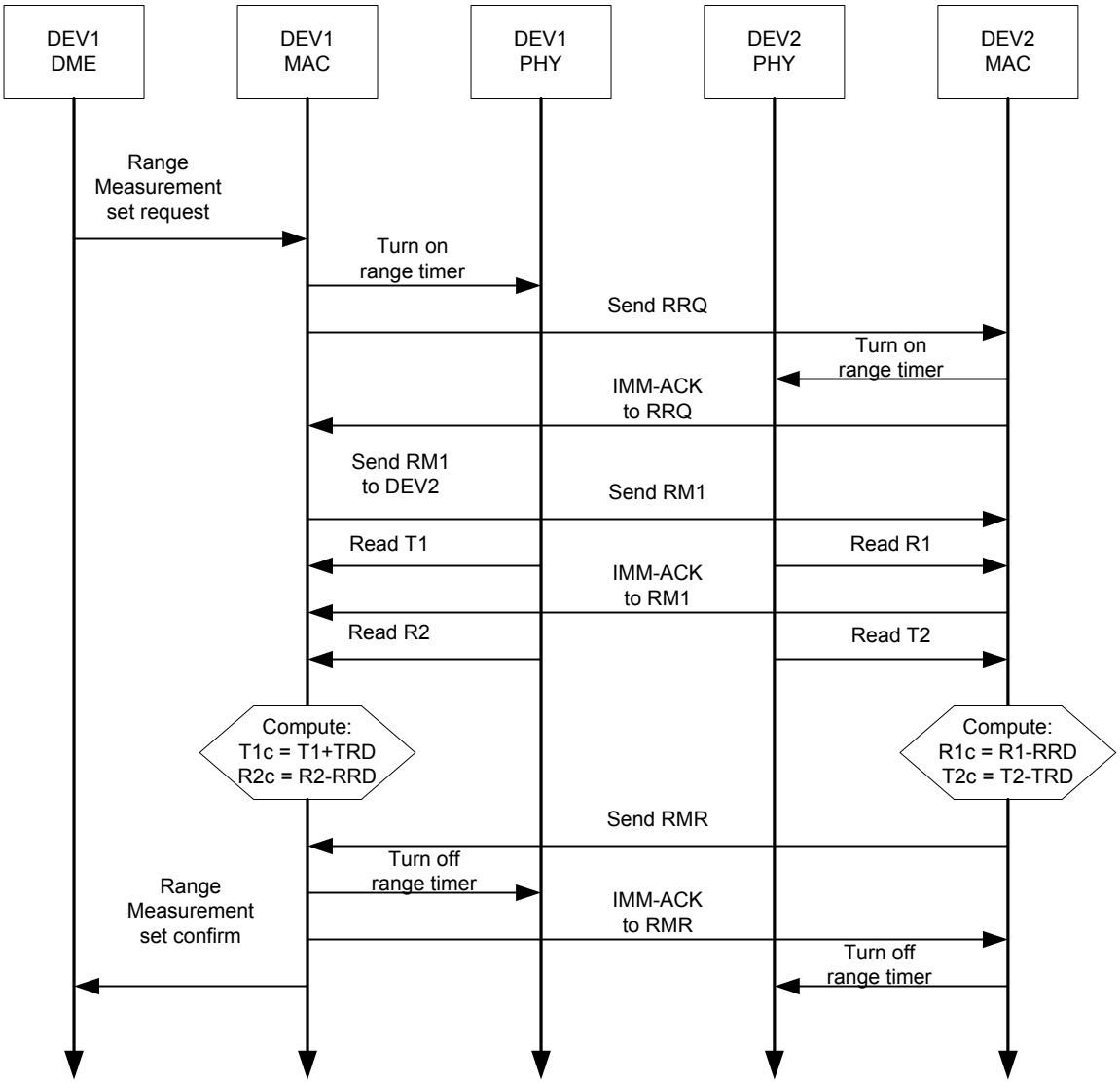



**Figure 78 — Single pair range measurement transaction**

## 8.16 MAC sublayer parameters

Table 70 contains the values for the MAC sublayer parameters.

**Table 70 — MAC sublayer parameters**

| Parameter | Value |
|---|---|
| mAccessDelay | 652 μs |
| mAggregationLimit | 63 |
| mBeaconSlotLength | 85 μs |
| mBPExtension | 8 beacon slots |
| mBPMergeWaitTime | 128 superframes |
| mClockAccuracy | 20 ppm |
| mClockResolution | 1 μs |
| mDRPBackoffWinMax | 16 superframes |
| mDRPBackoffWinMin | 2 superframes |
| mGuardTime | 12 μs |
| mInitialMoveCountdown | 3 × mMaxLostBeacons |
| mMASLength | 256 μs |
| mMaxBeaconLength | mBeaconSlotLength - pSIFS - mGuardTime |
| mMaxBPLength | 96 beacon slots |
| mMaxFragmentCount | 8 |
| mMaxFramePayloadSize | pMaxFramePayloadSize |
| mMaxHibernationProtection | 128 superframes |
| mMaxLostBeacons | 3 |
| mMaxNeighborDetectionInterval | 128 superframes |
| mMaxSynchronizationAdjustment | 4 us |
| mMinFragmentSize | 1 |
| mSignalSlotCount | 2 beacon slots |
| mSuperframeLength | 256 × mMASLength |
| mCWMin[AC_BK] | 15 |
| mCWMin[AC_BE] | 15 |
| mCWMin[AC_VI] | 7 |
| mCWMin[AC_VO] | 3 |

| | |
|---|---|
| mCWMax[AC_BK] | 1023 |
| mCWMax[AC_BE] | 1023 |
| mCWMax[AC_VI] | 511 |
| mCWMax[AC_VO] | 255 |
| mAIFSN[AC_BK] | 7 |
| mAIFSN[AC_BE] | 4 |
| mAIFSN[AC_VI] | 2 |
| mAIFSN[AC_VO] | 1 |
| mTXOPLimit[AC_BK] | 512 μs |
| mTXOPLimit[AC_BE] | 512 μs |
| mTXOPLimit[AC_VI] | 1024 μs |
| mTXOPLimit[AC_VO] | 256 μs |

Table 71 contains the values of the PHY dependent parameters used by the MAC sublayer for the WiMedia PHY.

**Table 71 — PHY-dependent MAC sublayer parameters for the WiMedia PHY**

| Parameter | Value |
|---|---|
| pBeaconTransmitRate | 53.3 Mbps |
| pCCADetectTime | 5.625 μs |
| pClockAccuracy | 20 ppm |
| pMaxFramePayloadSize | 4095 octets |
| pMIFS | 1.875 μs |
| pRangingReceiveDelay | Implementation-dependent |
| pRangingTransmitDelay | Implementation-dependent |
| pSIFS | 10 μs |
| pSlotTime | 9 μs |

# 9 Security

This clause specifies the security mechanisms needed to provide the security service introduced in 5.4.9. Subclause 9.1 reviews these security mechanisms. Subclause 9.2 defines security modes that govern the security operation of devices. Subclause 9.3 specifies the 4-way handshake procedure for two devices to establish pair-wise temporal keys (PTKs) and a secure relationship. This subclause also describes how a device may solicit or distribute group temporal keys (GTKs) within a secure relationship. Subclause 9.4 describes the procedures for frame reception and replay prevention. Subclause 9.5 provides the parameters needed in applying the AES-128 CCM cryptography to compute the message integrity code (MIC) and encrypt the secure payload for secure frames.

## 9.1 Security mechanisms

The security mechanisms specified in this standard control the security operation of devices by setting appropriate security modes. They allow devices to authenticate each other, to derive PTKs, and to establish secure relationships. They also enable devices to solicit or distribute GTKs within established secure relationships. In addition, the security mechanisms provide replay attack prevention measures through the use of secure frame counters (SFCs) and replay counters. The security mechanisms specify the parameters needed in applying the AES-128 CCM to protect the privacy and integrity of unicast and broadcast/multicast traffic using PTKs and GTKs, respectively. Privacy is protected by encrypting the secure payload, while integrity is protected by including a MIC.

Two devices use a shared master key to establish a secure relationship. The establishment and management of master keys are additional security facilities that need to be provided outside the MAC sublayer.

### 9.1.1 Security operation

Security modes are defined to control the level of security required of a device in its communications with other devices. Three security modes are provided. Mode 0 allows a device to communicate without security protection. Mode 1 allows a device to use both secure and non-secure frames for data exchanges. Mode 2 restricts a device to use security facilities in transmitting and receiving certain frames.

A device announces its selected security mode in the Beacon Parameters field in its beacons.

### 9.1.2 4-way handshake

The 4-way handshake mechanism enables two devices to use a shared master key to authenticate the identity of each other and to establish a new PTK for protecting certain frames exchanged between the two devices. By way of a successful 4-way handshake, the two devices establish a secure relationship with each other.

A device initiates a 4-way handshake with another device only if it has determined that it shares a master key with that device. The master key is not exposed in the 4-way handshake; it is specified by a master key identifier (MKID). A 4-way handshake is affected by use of the MLME-PTK primitives and PTK commands. Following a successful 4-way handshake, each device installs the new PTK into the MAC entity via the MLME-KEY-UPDATE primitives.

### 9.1.3 Key transport

Two devices establish a new PTK via a 4-way handshake. The PTK is derived from a shared master key and two new random numbers generated by the two devices. A PTK is never transmitted directly in any frame, encrypted or not.

Two devices, after establishing a secure relationship via a successful 4-way handshake, distribute their respective GTKs for protecting their broadcast traffic to each other, if applicable. Additionally,

a device may distribute GTKs for protecting certain multicast traffic addressed to those devices with which the device has a valid secure relationship. A device may also request, or solicit, GTKs used to protect multicast traffic from the multicast source devices.

A GTK is solicited or distributed by use of the MLME-GTK primitives and GTK commands. It is sent in encrypted form.

### 9.1.4 Freshness protection

Freshness protection insures that no parties can successfully replay previously captured messages as an attack. This standard defines secure frame counters and replay counters on a per-temporal key basis to provide freshness protection.

### 9.1.5 Data encryption

Data encryption uses a symmetric cipher to protect data from access by parties not possessing the encryption key. This key is a PTK for unicast traffic transmitted between two devices and a GTK for broadcast/multicast traffic transmitted from a sender to a group of recipients.

Secure frames using a TKID not recognized by the recipient device are reported to the DME using MLME-SECURITY-VIOLATION.indication with ViolationCode set to INVALID_TKID.

AES-128 counter mode is used for data encryption in this standard.

### 9.1.6 Frame integrity protection

Frames are protected from modification by other parties by message authentication using a MIC. The MIC also provides assurance that the sender of the frame possesses the correct temporal key. This key is shared among a group of devices or only between two devices. The MIC is a cryptographic checksum of the message to be protected.

All secure frames that fail MIC checks are reported to the DME using MLME-SECURITY-VIOLATION.indication with ViolationCode set to INVALID_MIC.

AES-128 cipher block chaining-message authentication code (CBC-MAC) is used for MIC calculation in this standard.

## 9.2 Security modes

The security mode indicates whether a device is permitted or required to establish a secure relationship with another device for data communications.

Two devices establish a secure relationship by a 4-way handshake based on a shared master key as described in 9.3.

Once two devices establish a secure relationship, they shall use secure frames for frame transfers between them as specified in Table 72 and Table 73. Either device shall discard a received frame from the other device if the frame is required to be a secure frame but was transmitted as a non-secure frame.

Data and aggregated data frames shall be transmitted using the temporal key specified by the TKID passed through the MAC SAP along with the corresponding MSDU. Command and control frames, when transmitted as secure frames in a secure relationship, shall employ a temporal key currently possessed in that secure relationship.

In Table 72, “N” indicates a non-secure frame, and “S” indicates a secure frame.

**Table 72 — Frame protection in a secure relationship**

| Frame type or subtype | Frame protection | Meaning |
|---|---|---|
| Beacon frame | N | Beacon frames shall be sent as non-secure frames. |
| Imm-ACK control frame | N | Imm-ACK frames shall be sent as non-secure frames. |
| B-ACK control frame | N | B-ACK frames shall be sent as non-secure frames. |
| RTS control frame | N | RTS frames shall be sent as non-secure frames. |
| CTS control frame | N | CTS frames shall be sent as non-secure frames. |
| UDA control frame | N | UDA frames shall be sent as non-secure frames. |
| UDR control frame | N | UDR frames shall be sent as non-secure frames. |
| Application-specific control frame | N, S | Application-specific control frames may be sent as secure or non-secure frames. |
| DRP Reservation Request command frame | N, S | DRP Reservation Request frames may be sent as secure or non-secure frames. |
| DRP Reservation Response command frame | N, S | DRP Reservation Response frames may be sent as secure or non-secure frames. |
| Probe command frame | N, S | Probe frames may be sent as secure or non-secure frames. |
| PTK command frame | N, S | PTK frames may be sent as secure or non-secure frames. |
| GTK command frame | S | GTK frames shall be sent as secure frames. |
| Range Measurement command frame | N, S | Range Measurement frames may be sent as secure or non-secure frames. |
| Probe command frame | N, S | Probe frames may be sent as secure or non-secure frames. |
| Application-specific command frame | N, S | Application-specific command frames may be sent as secure or non-secure frames. |
| Data frame | S | Data frames shall be sent as secure frames. |
| Aggregated data frame | S | Aggregated data frames shall be sent as secure frames. |

Table 73 specifies the values of the Encryption Offset (EO) field in secure frames.

**Table 73 — EO values in secure frames**

| Frame type or subtype | EO value |
|---|---|
| Application-specific control frame | Application defined |
| DRP Reservation Request command frame | Length of Secure Payload |
| DRP Reservation Response command frame | Length of Secure Payload |
| PTK command frame | 0 |
| GTK command frame | 0 |
| Range Measurement command frame | Length of Secure Payload |
| Probe command frame | Variable |
| Application-specific command frame | Application defined |
| Data frame | Variable |
| Aggregated data frame | Length of (Aggregation Header + Aggregation Header Pad octets) |

### 9.2.1 Security mode 0

A device operating in security mode 0 shall use non-secure frames to communicate with other devices. Such a device shall not establish a secure relationship with any other device.

If a device operating in this mode receives a secure frame, the MAC entity shall discard the frame. The MLME then issues an MLME-SECURITY-VIOLATION.indication with ViolationCode set to INVALID_MODE.

### 9.2.2 Security mode 1

A device operating in security mode 1 shall use non-secure frames to communicate with devices operating in security mode 0. The device shall also use non-secure frames to communicate with devices operating in security mode 1 with which it does not have secure relationships. The device shall use secure frames according to Table 72 and Table 73 to communicate with another device operating in security mode 1 with which it has a secure relationship. It shall not establish secure relationships with other devices unless those devices are also operating in security mode 1.

A device operating in security mode 1 may or may not respond to command frames received from other devices with which it does not have a secure relationship.

If a device operating in security mode 1 receives a secure frame from a device with which it does not have a secure relationship, the MAC entity shall discard the frame. The MLME then issues an MLME-SECURITY-VIOLATION.indication with ViolationCode set to INVALID_TKID.

If a device operating in mode 1 receives a non-secure frame from a device with which it has a secure relationship, but the frame is required to be a secure frame per Table 72, the MAC entity shall discard the frame. The MLME then issues an MLME-SECURITY-VIOLATION.indication with ViolationCode set to INVALID_MODE.

A DME that chooses to enable security mode 1 must understand and accept the responsibility that comes with receiving non-secure frames. The DME shall instruct the higher layers to handle the received non-secure frames in a safe and secure manner.

A compliant MAC entity shall never use security mode 1 by default. Security mode 1 shall be entered from either mode 0 or mode 2. Requiring that a DME explicitly select this mode serves as

an indication that the DME is aware of the security responsibilities it accepts when enabling security mode 1.

### 9.2.3 Security mode 2

A device operating in security mode 2 shall not establish a secure relationship with devices operating in either security mode 0 or security mode 1. The device shall use secure frames based on Table 72 and Table 73 to communicate with another device operating in security mode 2 and having a secure relationship with it. A device operating in security mode 2 shall establish a secure relationship with another device operating in the same security mode by a 4-way handshake prior to data exchanges.

If a device operating in mode 2 receives a secure frame from a device with which it does not have a secure relationship, the MAC entity shall discard the frame. The MLME then issues an MLME-SECURITY-VIOLATION.indication with ViolationCode set to INVALID_TKID.

If a device operating in mode 2 receives a non-secure frame that is required to have frame protection per Table 72, regardless of whether the device has a secure relationship with the device transmitting the frame, the MAC entity shall discard the frame. The MLME then issues an MLME-SECURITY-VIOLATION.indication with ViolationCode set to INVALID_MODE.

## 9.3 Temporal keys

Two devices establish a secure relationship based on a shared master key by employing a 4-way handshake to derive a PTK as described in this subclause. They may establish a PTK for each master key they share. Two devices have a secure relationship as long as they possess a currently installed PTK. A device's DevAddr is part of the information used in deriving a PTK. Once a PTK is established, it shall not be changed due to a change in the device's DevAddr.

A device solicits a GTK from, or distributes a GTK to, another device sharing a PTK as also described in this subclause.

Master keys are identified by MKIDs. A device is not required to include an MKID IE in its beacon, nor is it required to advertise every MKID it possesses in the MKID IE included in its beacon. They may advertise some or all of the MKIDs they possess in an MKID IE in their beacons. A device may probe another device for the MKIDs possessed by that device by addressing an appropriate Probe IE in a beacon or Probe command to that device. A device shall list all the MKIDs it possesses in the MKID IE in response to a probe request for its MKIDs.

### 9.3.1 Mutual authentication and PTK derivation

This standard uses a 4-way handshake to provide mutual authentication and PTK generation for two devices sharing a master key. To perform a 4-way handshake, the two devices assume the roles of "initiator" and "responder", respectively. A 4-way handshake consists of four messages, called message 1, message 2, message 3, and message 4 in this standard, that are sent back and forth between the two devices. The device sending message 1 becomes the initiator. The other device becomes the responder.

#### 9.3.1.1 4-way handshake message 1

The initiator shall begin a 4-way handshake by composing and sending message 1 in a PTK command to the responder. In this command, the initiator shall specify the MKID for use in the 4-way handshake, propose a TKID for the PTK to be derived, and include a unique 128-bit cryptographic random number, I-Nonce. The proposed TKID shall be different from any TKID currently installed in the initiator's local MAC entity or being used in an in-progress 4-way handshake involving this initiator device. The I-Nonce shall be generated anew each time the initiator starts a new 4-way handshake.

On reception of message 1, the responder shall verify that the requested TKID is unique (i.e., not currently installed for an active temporal key or requested by an in-process 4-way handshake exchange). The responder shall perform the following steps:

1. Generate a new 128-bit cryptographic random number, R-Nonce.
2. Derive the PTK and KCK as specified in 9.3.4.
3. Construct and send message 2 in a PTK command.

#### 9.3.1.2 4-way handshake message 2

The responder shall send message 2 to the initiator as specified in 9.3.1.1. In this command, the responder shall include an appropriate Status Code, the newly generated R-Nonce, and the PTK MIC value computed for the message using the newly derived KCK according to 9.3.5. If the proposed TKID in message 1 is not unique, the responder shall so indicate in the Status Code.

On reception of message 2, the initiator shall perform the following steps:

1. Derive the PTK and KCK as specified in 9.3.4.
2. Recalculate the PTK MIC for the received message using the KCK according to 9.3.5. If the recalculated PTK MIC does not match the PTK MIC field from this message, discard and disregard message 2 and abort the 4-way handshake. Otherwise, consider this message a proof that the responder holds the correct master key, and proceed to the next step.
3. Check the Status Code returned in the received message. If the Status Code indicates an abortion of the 4-way handshake by the responder, stop the 4-way handshake as well. If the Status Code indicates a conflict of the proposed TKID at the responder, restart the 4-way handshake with a different TKID. If the Status Code indicates a normal status, proceed to the next step.
4. Construct and send message 3 in a PTK command.

#### 9.3.1.3 4-way handshake message 3

The initiator shall send message 3 to the responder as specified in 9.3.1.2. In this command, the initiator shall include the same I-Nonce as contained in message 1 and a PTK MIC computed for this message using the newly derived KCK according to 9.3.5.

On reception of message 3, the responder shall perform the following steps:

1. Verify the PTK MIC for this message using the KCK according to 9.3.5. If the calculated PTK MIC does not match the PTK MIC field from this message, discard and disregard message 3 and abort the 4-way handshake. Otherwise, consider this message a proof that the initiator holds the correct master key, and proceed to the next two steps.
2. Construct and send message 4 in a PTK command.
3. Install the PTK using the MLME-KEY-UPDATE primitives.

#### 9.3.1.4 4-way handshake message 4

The responder shall send message 4 to the initiator as specified in 9.3.1.3. In this command, the responder shall include the same R-Nonce as contained in message 2 and a PTK MIC computed for this message using the KCK according to 9.3.5.

On reception of message 4, the initiator shall perform the following step:

1. Verify the PTK MIC for this message using the KCK according to 9.3.5. If the calculated PTK MIC does not match the PTK MIC field from this message, discard and disregard message 4 and abort the 4-way handshake. Otherwise, install the PTK using the MLME-KEY-UPDATE primitives.

### 9.3.2 GTK exchange

Upon successful completion of a 4-way handshake and installation of the resulting PTK, the initiator and responder each shall use GTK command frames (with Message Number set to 1) to distribute their respective GTKs for broadcast traffic to each other. Each may also use a GTK command to distribute a GTK for protecting certain multicast traffic to an intended recipient with which it holds a valid PTK.

On reception of a valid GTK command frame marked as Message Number 1, a device shall verify that the GTKID is a unique TKID. The device shall then respond with a GTK command frame with Message Number set to 2 and Status Code set to the appropriate value.

A recipient may request a GTK for certain multicast traffic in the form of a GTK command (with Message Number set to 0) from the source device if it holds a valid PTK with the source.

On reception of a valid GTK command marked as Message Number 0, the multicast source device shall respond with a GTK command marked as Message Number 1, which may or may not contain the requested GTK. The requesting device, upon receiving this GTK command and verifying the uniqueness of the proposed TKID, shall further return a GTK command with Message Number set to 2 and Status Code set to the appropriate value.

A source device distributing a GTK shall check the Status Code indicated in the returned GTK command (Message Number set to 2). If the Status Code indicates a conflict of the proposed TKID at the recipient device, the source device shall propose a new TKID and re-distribute the GTK to the recipient. After receiving a returned GTK command from the recipient with the Status Code indicating a normal status, the source device shall use the new TKID to re-distribute the GTK to each of the devices to which it has previously distributed the GTK and with which it maintains a secure relationship.

A device installs a newly distributed or received GTK using the MLME-KEY-UPDATE primitives.

A GTK shall be a 128-bit cryptographic-grade random number. A fresh GTK shall be generated when the distributing device establishes a new group relationship. Subclause 9.3.6 provides an example means of generating a fresh GTK.

### 9.3.3 Pseudo-random function (PRF) definition

A PRF is used in several places in the security specification. Depending on the use, the PRF may need to output values of 64 bits, 128 bits, and 256 bits. This subclause defines three PRF variants:

— PRF-64, which outputs 64 bits,
— PRF-128, which outputs 128 bits, and
— PRF-256, which outputs 256 bits.

In the following, *K* denotes a 128-bit symmetric key, *N* denotes a 13-octet nonce value, *A* denotes a unique 14-octet ASCII text label for each different use of the PRF, *B* denotes the input data stream, *Blen* specifies the length of this data stream, and || denotes concatenation. Blocks are each 16 octets long, and are defined as inputs to the AES-128 CCM for the MIC generation as specified in 9.5.

CCM-MAC-FUNCTION(*K, N, A, B, Blen*)

**begin**

Form authentication block B_0 from flags = 0x59, *N*, and l(m) = 0

Form authentication block B_1 from l(a) = 14 + *Blen* and *A*

Form additional authentication blocks from *B*

*(with last block zero padded as needed)*

Form encryption block A_0 from flags = 0x01, *N*, and Counter_*0* = 0

*R* ← MIC (*K, B_0, B_1, ..., A_0*)

**return** *R*

PRF(*K, N, A, B, Blen, Len*)

**for** *i* ← 1 **to** ( *Len* + 63)/64 **do**

*R* ← *R* || CCM-MAC-FUNCTION(*K, N, A, B, Blen*)

*N* ← *N* + 1

**return** L(*R*, 0, *Len*) = *Len* most-significant bits of *R*

PRF-64(*K, N, A, B, Blen*) = PRF(*K, N, A, B, Blen,* 64)

PRF-128(*K, N, A, B, Blen*) = PRF(*K, N, A, B, Blen,* 128)

PRF-256(*K, N, A, B, Blen*) = PRF(*K, N, A, B, Blen,* 256)

### 9.3.4 PTK and KCK derivation

PRF-256 shall be employed to generate the PTK and KCK associated with a 4-way handshake as used in 9.3.1 based on the following parameters as defined in Table 74.

*K* – The PMK
*N* – B12-11= InitiatorDevAddr, B10-9= ResponderDevAddr, B8-6 = PTKID, B5-0 = zero
*A* – “Pair-wise keys”
*B* – I-Nonce || R-Nonce
*Blen* –32

**Table 74 — PTK and KCK generation parameters**

| Name | Size (octets) | Description |
|---|---|---|
| InitiatorDevAddr | 2 | DevAddr of device with role of initiator |
| ResponderDevAddr | 2 | DevAddr of device with role of responder |
| I-Nonce | 16 | Random number selected by initiator (in message 1) |
| R- Nonce | 16 | Random number selected by responder (in message 2) |
| PTKID | 3 | Negotiated TKID value for the PTK to be derived (in message 1) |
| PMK | 16 | A pre-shared pair-wise master key identified by the MKID (in message 1) |

The PRF-256 is called with these parameters to compute a 256-bit key stream:

KeyStream ← PRF-256(*K, N, A, B, Blen*)

This key stream is then split to form the desired PTK and KCK. The least-significant 16 octets of KeyStream become the KCK while the most-significant 16 octets become the PTK, as illustrated in Table 75.

**Table 75 — KCK and PTK in KeyStream**

| Key | Source |
|---|---|
| KCK | KeyStream octets 0 through 15 |
| PTK | KeyStream octets 16 through 31 |

### 9.3.5 PTK MIC generation

The 4-way handshake uses an “out-of-band MIC” calculation for the PTK MIC field in handshake messages 2-4. PRF-64 shall be used to provide the PTK MIC calculation. The PRF-64 parameters shall be defined as follows based on Table 74:

*K* – The KCK
*N* – B12-11 = InitiatorDevAddr, B10-9 = ResponderDevAddr, B8-6 = PTKID, B5-0 = zero
*A* – “out-of-bandMIC”
*B* – Fields from Message Number to I-Nonce/R-Nonce contained in the PTK command
*Blen* – Length in octets of *B* = 48

PTK MIC ← PRF-64(*K*, *N*, *A*, *B*, *Blen*)

### 9.3.6 Random number generation

To implement the cryptographic mechanisms outlined in this standard, every platform needs to be able to generate cryptographic grade random numbers. NIST Special Publication 800-38C gives a detailed explanation of the notion of cryptographic grade random numbers and provides guidance for collecting suitable randomness. It recommends collecting random samples from multiple sources followed by conditioning with PRF. This method can provide a means for an implementation to create an unpredictable seed for a pseudo-random generation function. The example below shows how to distill such a seed using random samples and PRF-128.

LoopCounter = 0

Nonce = 0

**while** LoopCounter < 32 **begin**

result = PRF-128(0, Nonce, “InitRandomSeed”, DevAddr || Time || result || LoopCounter, dataLen)

Nonce ← Nonce + 1

result ← result || <randomness samples>

**end**

GlobalSeed = PRF-128(0, Nonce, “InitRandomSeed”, DevAddr || Time || result || LoopCounter, dataLen)

Once the seed has been distilled, it can be used as a key for further random number generation. The 4-way handshake requires each party to supply a 128-bit random number. This number can be generated using the seed and PRF-128.

GenerateRandomNonce

**begin**

N = DevAddr || DevAddr || zero

Collect randomness samples

result = PRF-128(Global Seed, N, “Random Numbers”, <randomness samples>, length of samples)

**return result**

## 9.4 Frame reception steps and replay prevention measures

A recipient device shall carry out the reception steps and replay prevention measures as specified in this subclause.

### 9.4.1 Frame reception

The MAC entity shall perform the following validation steps when receiving frames:

1. Validate the FCS. If this validation fails, discard the frame. Otherwise, acknowledge the received frame using the appropriate acknowledgment rules, and proceed to the next step.
2. Validate the Secure bit setting in the MAC Header and take the appropriate actions according to its security mode as specified in 9.2. If the frame is not discarded and the Secure bit is set to one, proceed to the next step.
3. Validate the TKID. If the TKID does not identify a currently installed PTK or GTK, discard the frame. The MLME issues an MLME-SECURITY-VIOLATION.indication with ViolationCode set to INVALID_TKID. Otherwise, proceed to the next step.
4. Validate the MIC using the identified PTK or GTK as specified in 9.5. If this validation fails, discard the frame. The MLME issues an MLME-SECURITY-VIOLATION.indication with ViolationCode set to INVALID_MIC. Otherwise, proceed to the next step.
5. Detect frame replay as specified in 9.4.2. If replay is detected, discard the frame. The MLME issues an MLME-SECURITY-VIOLATION.indication with ViolationCode set to REPLAYED_FRAME. Otherwise, update the replay counter that was set up for the PTK or GTK used for this frame as also specified in 9.4.2, and proceed to the next step.
6. Process the frame as specified in clause 8, including duplicate frame filtering. If the frame was already received, discard it. Otherwise, proceed to the next step.
7. Decrypt the frame. This step may be taken in parallel with the MIC validation step.

### 9.4.2 Replay prevention

Each transmitting MAC entity shall set up a 48-bit SFC and initialize it to zero when a temporal key, PTK or GTK, is installed to it. The MAC entity shall increment the SFC by one before transmitting a secure frame—whether a new frame or a retry—that uses the temporal key, and shall set the SFN in that secure frame to the value of the SFC after the increment.

Each recipient MAC entity shall set up a 48-bit replay counter when a temporal key, PTK or GTK, is installed to it. The MAC entity shall initialize the replay counter to zero for an installed PTK, and to the GTK SFC for an installed GTK which was contained in the GTK command distributing the GTK.

Upon receipt of a secure frame with valid FCS and MIC, the recipient shall perform replay attack detection and protection as follows:

The recipient shall compare the SFN extracted from the received frame with the reading of the replay counter for the temporal key used by the frame. If the extracted SFN is smaller than or equal to the replay counter reading, the recipient MAC entity shall discard the frame and the MLME issues an MLME-SECURITY-VIOLATION.indication with ViolationCode set to REPLAYED_FRAME. Otherwise, the recipient shall set the corresponding replay counter to the received SFN.

The recipient shall insure that the frame passes FCS validation, replay prevention, and MIC verification before using the SFN to update its replay counter.

### 9.4.3 Implications on GTKs

Because a recipient maintains only one replay counter per installed temporal key, that recipient can receive traffic from only one source using a given temporal key. A scheme that allows multiple source devices to use the same GTK will result in frames sent from some of those sources being seen as replay attacks. To avoid this problem, each source device in a group is required to distribute a unique GTK to the recipients in the group.

## 9.5 AES-128 CCM Inputs

AES-128 CCM provides confidentiality, authentication, and integrity for secure frames defined in this standard. This subclause specifies the various fields required for AES-128 CCM operation.

### 9.5.1 Overview

AES, the Advanced Encryption Standard, is specified in FIPS PUB 197. AES-128 defines a symmetric block cipher that processes 128-bit data blocks using 128-bit cipher keys. CCM, counter with CBC-MAC, is specified in RFC 3610. CCM employs counter mode for encryption and cipher block chaining for authentication. AES-128 CCM combines AES-128 with CCM to encrypt and authenticate messages.

Encryption is done on part or all of the Secure Payload, while authentication is provided by a message integrity code (MIC) that is included in each secure frame. MIC also protects the integrity of the MAC Header and Frame Payload in a secure frame.

CCM has two input parameters — M (number of octets in authentication field) and L (number of octets in length field). For this standard, M = 8 and L = 2.

CCM requires the use of a temporal key and a unique Nonce for each transmitted frame to be protected. The SFN is combined with frame addressing and temporal key identification information to provide a unique Nonce for every secure frame. Since every frame protection with a key requires a unique Nonce, temporal keys have a known lifetime. Each temporal key can be used to protect up to *n* frames, where *n* is the maximum value of the SFN. All security guarantees are void if a nonce value is used more than once with the same temporal key.

In the following figures in this subclause showing the format of Nonce and CCM blocks, the most-significant octet is represented to the left of the other octets.

### 9.5.2 Nonce

The CCM Nonce is a 13-octet field, consisting of the 2-octet SrcAddr, 2-octet DestAddr, 3-octet TKID, and 6-octet SFN for the current frame. The Nonce is used as a component of authentication block B_0, an input to CBC-MAC. It is also used as a component of input block A_i for CCM encryption. It provides the uniqueness that CCM requires for each instance of authentication/encryption. The CCM Nonce shall be formatted as shown in Figure 79. In this figure, each component of the Nonce is represented with the most-significant octet on the left and the least-significant octet on the right.

| octets: 2 | 2 | 3 | 6 |
|---|---|---|---|
| SrcAddr | DestAddr | TKID | SFN |

**Figure 79 — Nonce input to the CCM algorithm**

### 9.5.3 CCM blocks

The CCM authentication blocks shall be formatted as shown in Figure 80 and further described below.

| octets: 1 | 13 | 2 | 2 | 10 | 2 | 1 | 1 | EO | 0-15 | P – EO | 0-15 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Flags ( = 0x59) | Nonce | Encrypted data length l(m) = P – EO | Additional authenticated data length l(a) = 14 + EO | MAC Header | Encryption Offset (EO) | Security Reserved | 0 | Secure Payload portion not to be encrypted | Zero padding | Secure Payload portion to be encrypted | Zero padding |
| B_0 | | | B_1 | | | | | B_2, …, B_(M-1) | | B_M, …, B_N | |

**Figure 80 — Input to CCM authentication blocks**

#### 9.5.3.1 Authentication block B_0

Authentication block B_0 is the first input block to the CBC-MAC algorithm. It shall be formatted as shown in Figure 81. The component l(m) is represented with the most-significant octet on the left and the least-significant octet on the right. The Nonce component is represented with the least-significant octet on the left and the most-significant octet on the right.

| octets: 1 | 13 | 2 |
|---|---|---|
| Flags = 0x59 | Nonce | l(m) |

**Figure 81 — Format of authentication block B_0**

#### 9.5.3.2 Authentication block B_1

Authentication block B_1 is the second input block to the CBC-MAC algorithm. It shall be formatted as shown in Figure 82. In this block, the l(a) component is represented with the most-significant octet on the left and the least-significant octet on the right. The EO and MAC Header components are represented with the first octet transmitted into the wireless medium on the left and the last transmitted octet on the right.

| octets: 2 | 10 | 2 | 1 | 1 |
|---|---|---|---|---|
| l(a) | MAC Header | EO | Security Reserved | 0 |

**Figure 82 — Format of authentication block B_1**

#### 9.5.3.3 Authentication blocks B_2, …, B_N

Authentication blocks B_2, …, B_(M-1) and B_M, …, B_N, if any, are additional input blocks to the CBC-MAC algorithm. They shall be formatted as shown in Figure 83. They are formed by breaking the Secure Payload portion not to be encrypted into 16-octet blocks and the Secure Payload

portion to be encrypted into 16-octet blocks. The last block constructed from the Secure Payload portion not to be encrypted is padded with zero values as needed to insure 16-octet block length. Likewise, the last block constructed from the Secure Payload portion to be encrypted is padded with zero values as needed to insure 16-octet block length. The padding octets are not transmitted onto the wireless medium.

| octets: EO | 0-15 | P – EO | 0-15 |
|---|---|---|---|
| Secure Payload portion not to be encrypted | Zero padding | Secure Payload portion to be encrypted | Zero padding |
| B_2, …, B_(M-1) | | B_M, …, B_N | |

**Figure 83 — Format of authentication blocks beginning from B_2**

In each of the blocks B_2, …, B_(M-1) or B_M, …, B_N, the Secure Payload portion not to be, or to be, encrypted shall be represented with the earliest octet transmitted into the wireless medium on the left and the latest transmitted octet on the right. When needed, B_(M-1) and B_N are padded with zeros to the right.

#### 9.5.3.4 Encryption blocks A_0, A_1, …, A_m

CCM uses encryption blocks A_0, A_1, …, A_m to generate key stream blocks that are used to encrypt the CBC-MAC and the Secure Payload portion to be encrypted. These blocks shall be formed as shown in Figure 84. In this figure, Counter *i* is a 2-octet monotonically incrementing counter that shall be initialized to 0 for each secure frame. It shall be incremented by one for each successive encryption block. The Counter *i* component of A_i shall be represented with the most-significant octet on the left and the least-significant octet on the right. The Nonce component shall be represented with the least-significant octet on the left and the most-significant octet on the right.

| octets: 1 | 13 | 2 |
|---|---|---|
| Flags = 0x01 | Nonce | Counter *i* |

**Figure 84 — Format of A_i blocks**

# Annex A (normative) MUX sublayer

The MUX sublayer is the MAC client as depicted in Figure 4 and routes data between the MAC sublayer and MUX clients.

## A.1 MUX service

The MUX sublayer is expressed in terms of the MUX SAP, the MUX service, and the MUX client. Each MUX client is associated with a unique protocol. Service data units presented at the MUX SAP by the MUX client are therefore associated with that protocol.

The protocol is encoded in a MUX header as either:

— A protocol identifier and an OUI; or
— An IEEE Ethernet type value [B3].

The MUX service adds a MUX header to the MUX service data unit to construct a MUX protocol data unit. The MUX sublayer makes use of the service provided by the MAC sublayer for the transfer of its protocol data units.

On receipt of a MUX protocol data unit from the MAC sublayer, the MUX service removes the MUX header and delivers the transported service data unit to the appropriate MUX client based on the identified protocol.

## A.2 MUX protocol data unit format

A MUX protocol data unit consists of a MUX Header and a MUX Payload and is illustrated in Figure 85.

| octets: 2 or 5 | N |
|---|---|
| MUX Header | MUX Payload |

**Figure 85 — MUX protocol data unit format**

The MUX Payload field contains the MUX service data unit that is a payload data unit of the protocol identified in the MUX Header.

The first two octets of the MUX Header are encoded as unsigned binary values, and are delivered to the MAC sublayer in order from the octet containing the most-significant bits to the octet containing the least-significant bits. The octet order for this field is the reverse of that for most fields in this specification.

The MUX Payload is a sequence of octets labeled as MUX Payload[0] through MUX Payload[M-1]. Octets are passed to the MAC sublayer in ascending index-value order.

The MUX Header and MUX Payload together form the payload of the MAC sublayer, which appears to the MAC as a sequence of octets labeled as payload[0] through payload[P-1], as specified in 7.2.

There are three versions of the MUX Header, which are distinguished based on the value of the first two octets of the header.

### A.2.1 MUX Header—OUI version

The first version has a length of five octets and is illustrated in Figure 86.

| octets: 2 | 3 |
|---|---|
| Protocol ID<br>(0x0000–0x00FF) | OUI |

**Figure 86 — Format of first version of MUX Header**

The Protocol ID field is restricted to values from 0 through 255 and is set to a value that identifies a protocol defined by the owner of the OUI specified in the OUI field. The OUI is a sequence of 3 octets, labeled as oui[0] through oui[2]. Octets of the OUI are passed to the MAC sublayer in ascending index-value order.

### A.2.2 MUX Header—reserved version

The second version of the MUX Header has a length of 2 octets and is illustrated in Figure 87.

| octets: 2 |
|---|
| Protocol ID<br>(0x0100–0x05FF) |

**Figure 87 — Format of second version of MUX Header**

The Protocol ID field is restricted to values from 256 through 1535. These values identify the protocol that defines the MUX Payload format, and are defined by WiMedia and listed in *WiMedia Assigned Numbers* [B5].

### A.2.3 MUX Header—Ethernet type version

The third version of the MUX Header has a length of two octets and is illustrated in Figure 88.

| octets: 2 |
|---|
| Ethernet Type<br>(0x0600–0xFFFF) |

**Figure 88 — Format of third version of MUX Header**

The Ethernet Type field is restricted to values from 1536 through 65535 and is set to the value of an Ethernet type [B3] identifying a protocol.

# Annex B (normative) MAC policies

## B.1 Beacon slot selection

When a device selects an initial beacon slot after scanning for beacons as described in 8.2.3, the device shall transmit a beacon only if it selected a slot within mMaxBPLength/2 after the BPST.

## B.2 Reservation limits

A reservation consists of a row component and a column component.

Row component: A portion of a reservation that includes an equal number of MASs at the same offset(s) within every zone, optionally excluding zone zero, as indicated in the DRP IE(s).

Column component: The portion of the reservation that is not a row component.

Rules stated in this subclause apply independently to a device whether it is a reservation owner or a reservation target. They do not apply to DRP IEs with Reservation Type set to Alien BP.

A device may consider contiguous reservation blocks from multiple column components in the same zone as if they were a single reservation block in a single column component.

A device shall not allocate more channel time than necessary for its optimal operation.

A device shall set the Unsafe bit of the DRP Control field of a DRP IE according to the following rules:

1. A device shall not identify more than mTotalMASLimit MASs in DRP IEs with the Unsafe bit set to zero.
2. A device shall not identify more than Y consecutive MAS in the same zone within a column component in DRP IEs with the Unsafe bit set to zero, where Y is a function of the MAS number within the zone (counting from zero) of the earliest reserved MAS within the set of consecutive MASs, as shown in Table 76.

**Table 76 — Reservation block size limits**

| First MAS number | Y |
|---|---|
| 0 | 8 |
| 1 | 7 |
| 2 | 6 |
| 3 | 5 |
| 4 | 4 |
| 5 | 4 |
| 6 | 4 |
| 7 | 4 |
| 8 | 4 |
| 9 | 4 |
| 10 | 4 |
| 11 | 4 |
| 12 | 4 |
| 13 | 3 |
| 14 | 2 |
| 15 | 1 |

3. A device shall not set the Unsafe bit to one in DRP IEs except to comply with 1 or 2.

A device shall not include MASs in zone zero in the column component of a reservation.

A device may at any time send a Relinquish Request IE in its beacon where the Target DevAddr identifies a device transmitting its beacon with one or more DRP IEs with the Unsafe bit of the DRP IE Control field set to one (unsafe DRP IEs). The device shall not set the Target DevAddr field to identify a device if that device does not include any unsafe DRP IEs in its beacon, unless forwarding a received Relinquish Request IE to its reservation owner, as specified in 8.1.10.17.

The Allocation fields of the Relinquish Request IE should identify MASs in one or more unsafe DRP IEs.

The Reason Code of the Relinquish Request Control field should be set to a valid Reason Code indicating the reason for requesting the identified MASs.

If a device receives a beacon that contains a Relinquish Request IE with Target DevAddr set to its own DevAddr that identifies MASs it includes in an unsafe DRP IE, it shall:

— Modify its DRP IEs to remove the identified MASs; or
— Modify its DRP IEs such that the Unsafe bit in any DRP IE that includes one or more identified MASs is set to zero per the previous rules in this subclause.

The device shall make this adjustment within mUnsafeReleaseLimit superframes after first receiving the Relinquish Request IE.

If a device requests a neighbor to release MASs in an unsafe DRP IE, the device shall not include a new unsafe DRP IE in its beacon or change a DRP IE to set the Unsafe bit to one until mOwnerUnsafeHoldoff has passed.

If a device includes an unsafe DRP IE in its beacon and it receives a Relinquish Request IE that identifies MASs included in the DRP IE, it shall not include a new unsafe DRP IE in its beacon or change a DRP IE to set the Unsafe bit to one until the neighbor requesting release establishes a new DRP IE or mTargetUnsafeHoldoff has passed.

## B.3 PCA reservations

If a device initiates frame transactions in PCA reservations established by its neighbors, it should also establish PCA reservations that include the MASs it uses. A device shall not initiate frame transactions in more than mTotalMASLimit MASs that are included in DRP IEs with the Unsafe bit set to zero and Reservation Type set to PCA included in beacons by the device or any neighbor.

## B.4 Reservation locations

As referenced in this subclause, a reservation owner shall select MASs based on a minimum latency requirement of not less than 4 milliseconds, or on a medium utilization efficiency or power consumption requirement for a minimum reservation block length.

In the row component of a reservation, the reservation owner shall select reservation blocks such that the lowest MAS number selected within a zone is maximized, except that the reservation owner is not required to use more than one reservation block per zone.

In the column component of a reservation, the reservation owner shall select reservation blocks that meet its requirements such that each block is located within the first eight MASs of its zone, if possible. If not possible, the reservation owner shall select reservation blocks that meet its requirements and that minimize the highest MAS number selected in any zone.

If multiple potential zone locations meet the previous requirements, the reservation owner shall select reservation blocks in zones such that the latest used set in the following ordered list of sets is as early as possible:

[{8}, {4 or 12}, {2, 6, 10, or 14}, {1, 3, 5, 7, 9, 11, 13, or 15}].

If there are multiple possible zone locations that use the same latest set, the reservation owner should minimize the highest MAS number selected within the zones. The reservation owner shall place each reservation block at the earliest available location within its zone.

If the MASs available for reservation by the reservation owner change, it shall determine if its reservations would still meet the reservation location rules listed in B.4. If not, the reservation owner shall change its reservations within mCompactionLimit superframes such that they meet the reservation location rules.

A reservation owner may disregard the reservation location rules for MASs identified in DRP IEs with the Unsafe bit set to one according to the Y limit stated in B.2.

## B.5 Transmit power control

A device shall support transmit power control and use the lowest possible transmit power with which it can maintain its links.

## B.6 MAC policies parameters

Table 77 contains the values for the MAC policies parameters.

**Table 77 — MAC policies parameters**

| Parameter | Value |
|---|---|
| mCompactionLimit | 32 superframes |
| mOwnerUnsafeHoldoff | 32 superframes |
| mTargetUnsafeHoldoff | 2×mMaxLostBeacons superframes |
| mTotalMASLimit | 112 MASs |
| mUnsafeReleaseLimit | 4 superframes |

# Annex C (informative) Range measurement calculations

The MAC sublayer and the PHY layer can do two-way time transfer range measurement with remote devices. These measurements result in timing data, which needs further processing. This Annex describes two calculations that may be performed:

— Distance calculation
— Distance uncertainty

These calculations are shown assuming a 4224 MHz clock.

## C.1 Calculate distance for a single measurement

The MAC entity reports four values per measurement:

— T1c = corrected RM1 send time
— R1c = corrected RM1 receive time
— T2c = corrected RM2 send time
— R2c = corrected RM2 receive time

These are 32-bit numbers, in units of 4224 MHz clock periods.

T1c and R2c are measured from the same timer in the local DEV1.

R1c and T2c are measured from the same timer in the remote DEV2.

The round trip delay is T1c-R2c. The delay through DEV2 is R1c-T2c.

Subtracting the DEV2 delay from the round trip delay yields two flight times.

Therefore: Flight Time = ½((R2c-T1c)-(T2c-R1c)) in units of 4224MHz cycles.

The distance = Flight Time x (c/4224MHz), where c = speed of light.

c/4224MHz = 70.9754 mm, ~71mm.

## C.2 Calculate distance uncertainty

There are several sources of uncertainty or error in range time measurements:

— Noise
— Multipath effects
— Calibration delay constant accuracy (pRangingTtransmitDelay & pRangingReceiveDelay)
— Calibration delay constant precision
— Timer clock resolution
— Timer clock accuracy
— Timer clock differences between the two devices
— Relative motion

### C.2.1 Noise effects

The most likely effects of noise are twofold:

— Cause bit errors that prevent the intact deliver of ranging measurement frames RM1 or RM2 (the ACK for RM1).
— Cause variance in the accuracy of the receiving PHY's detection of the ranging reference signal.

If frames are not delivered, there will be incomplete sets of range measurement data.

If there are multiple measurements, the DME can use intact measurements to interpolate which measurements are missing, and discard the corresponding data. For example, if an RM2 frame was not received intact, there will be a missing R2c value.

Noise can cause jitter in the detection of timing references. This jitter will be reported in multiples of the receiving PHY's timing clock. For example, if noise causes a timing reference to be detected one clock early, and the timer is running at 1056 MHz (an optional value), that will result in a 14.2 cm error; 1056 MHz = ¼ of 4224 MHz, and the error is halved in the C.1 distance calculation.

The DME can attempt to improve distance measurement accuracy in the presence of noise by making repeated measurements and averaging the results.

### C.2.2 Multipath effects

The line-of-sight signal will be the first to arrive, yielding the correct range. However, signals will bounce, and non-line-of-sight signals may exceed the amplitude of the line-of-sight signal.

To avoid detecting multiple peaks, the PHY should employ some means to respond to the first peak, and ignore others shortly after.

If the PHY responds first to a non-line-of-sight signal, that signal will have taken a longer flight path, and will therefore take longer to arrive. The perceived flight time will be longer, and the computed distance will track. Therefore, in the presence of multipath, the calculated range distance measurements will be upper bounds on the true distance.

### C.2.3 Calibration delay constant accuracy & precision

The delays in the PHY are defined as:

pRangingTransmitDelay = the time from the generation of the reference signal, which triggers the RTIMER capture (e.g. T1 or T2), to the time this signal reaches the device antenna. T1c = T1 + pRangingTransmitDelay.

pRangingReceiveDelay = the time from the arrival of the reference signal at the antenna to the time this signal is first detected in the PHY, triggering the RTIMER clock capture (e.g. R1 or R2). R1c = R1 - pRangingReceiveDelay.

These constants are 16-bit unsigned integer values, in units of 4224 MHz clock periods.

There are pairs of these constants for each PHY. Therefore, each of these four constants can contribute two errors:

— Inaccuracy, as specified by the manufacturer
— Quantization error (in the order of one 4224 MHz cycle, 237 psec)

The manufacturer may only be able to accurately measure and characterize the delays from inside the PHY DSP to the chip leads. The delays from the chip pins to the antenna may vary a small amount, depending on trace or cable lengths. Manufacturers my have to estimate these external transmission delays.

Since each delay is uncorrelated with the others, the expected quantization error is the RMS sum of four +/- half bit errors.

### C.2.4 Timer clock resolution

The distance measure is limited by the resolution of the timers used for measuring time of flight. The implemented clock resolution generates simple quantization errors.

The WiMedia PHY specification allows options in the PHY ranging timer resolution:

— 528 MHz, 56.8 cm, minimum if ranging is implemented
— 1056 MHz, 28.4 cm, optional
— 2112 MHz, 14.2 cm, optional
— 4224 MHz, 7.1 cm, maximum option

The local PHY reports these options to the MAC via a PLME static parameter. The DME can query these via thePLME-GET.request primitive.

The remote PHY reports this same RangingSupported static parameter to the remote MAC entity, which delivers it to the local MAC entity by the RMR frame. The local MAC entity in turn reports this value to the DME via the MLME-RANGE-MEASUREMENT.confirm primitive.

Therefore, either PHY can contribute quantization error. The expected value for either contribution is half of the quantization distance (e.g. 3.6–28.4 cm). The expected value for the combination of these errors is the RMS sum of the two contributions (e.g. 5–40 cm), since the clocks are not synchronized and therefore uncorrelated.

### C.2.5 Timer clock accuracy

The WiMedia PHY specification requires that the clock frequency must be within 20 ppm. Without correction, deviation by either clock generates time measurement errors. These errors are proportionate to two factors:

— The extent of inaccuracy (for each PHY clock)
— The time taken to perform each two-way time transfer measurement

The local PHY reports the manufacturer-specified timer clock tolerance to the MAC via a PLME static parameter. The units are in ppm (parts per million). The DME can query this via the PLME-GET.request primitive.

The remote PHY reports this same pClockAccuracy static parameter to the remote MAC entity, which delivers it to the local MAC entity by the RMR frame. The local MAC entity in turn reports this value to the DME via the MLME-RANGE-MEASUREMENT.confirm primitive. These values do not report the actual clock frequency errors; they report tolerances.

Each timer clock inaccuracy contributes error accumulated over the time it runs. The DEV1 clock should run approximately 23.4 μs + one flight time; the DEV2 clock should run approximately 23.4 μs — one flight time. 23.4 μs = 1 preamble (10.0) + frame (~3.4) + SIFS (10.0).

23.4 μs = 98,841 cycles of 4224 MHz. 20 ppm of 98,841 is approximately two clock cycles. Therefore, either PHY clock can generate errors of +/- 2 clocks; the maximum error could be just under 4 clock cycles. Because these effects are halved in the distance calculation, the maximum possible error would be 14 cm (28/2) if the clocks both err in the same direction.

The expected value of this error depends on the actual statistics of the crystals commonly used in these devices. If the clock errors are randomly distributed, it will be lower (Gaussian around zero) because positive and negative errors offset.

### C.2.6 Clock frequency differences

If the two PHY clocks are offset in the same direction, the errors will add, as described above. If they are in the opposite directions, they will cancel; the calculated result will be the same as that resulting from both clocks having the same offset as the average of their offsets.

It is possible using repeated measurements to detect and correct for differences between the local and the remote clock frequencies. However, the local DEV1 has no way of knowing whether its own clock or the remote clock is actually more accurate. (It only has the tolerances.) This operation would be useful only if the local clock is very tight (e.g. 5ppm) and the remote clock is very loose (e.g. 40 ppm).

### C.2.7 Clock frequency detection

If the local device wants to normalize measurements to its own clock, it can detect and correct for frequency offset differences between its local clock and the remote clock.

Detection of the clock frequency offset between the local clock and remote clock can be simple if there is little noise induced jitter. Consider the case of executing a set of N consecutive ranging measurements (e.g. Ranging_Measurement-set.request(DestAddr, N)). The MAC entity will return

N sets of timer values using Ranging_Measurement-set.confirm. Assuming the clock error is constant, that clock error (in 4224 MHz cycles) can be approximated by:

[(Nth T1c)-(1st T1c)] – [(Nth R1c)-(1st R1c)]

or

[(Nth T2c)-(1st T2c)] – [(Nth R2c)-(1st R2c)]

Working backwards, the relative error in any one measurement would be:

([(Nth T1c)-(1st T1c)] – [(Nth R1c)-(1st R1c)])/(2(N-1)).

For example, if N = 4, [(4th T1c)-(1st T1c)] should be approximately 2(4-1) x 23.4 μs = 140.4 μs, or 593,050 4224 MHz clock periods. If the remote clock differs by 20ppm, the cumulative error over 4 consecutive measurements would be approximately +/- 24 clock periods or 4 clocks per measurement, +/-28.4 cm.

If noise induced jitter is large, the repeated measurements might need to be processed to fit straight lines through the timing data, then process as shown in the previous paragraph.

### C.2.8 Motion effects

For motion to effect a single two-way time transfer measurement as described in this standard, the relative motion of the two devices would need to meet or exceed a clock wavelength in the measurement time of 23.4 μs. Using the most precise clock, 4224 MHz, the relative motion would have to be (71 mm / 23.4 μs) = (71 km / 23.4 s) ~ 3.0 km/s ~ Mach 10! Therefore, for practical purposes, relative motion will have no measurable effect.

# Annex D (informative) Test vectors

## D.1 KCK/PTK generation

**Table 78 — PTK and KCK generation input parameters**

| Parameter | Value |
|---|---|
| InitiatorDevAddr | 0xDEAD |
| ResponderDevAddr | 0xBEEF |
| PTKID | 0xDEAD32 |
| I-Nonce | (Sequence of octets in hexadecimal in transmit order)<br>10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F |
| R-Nonce | (Sequence of octets in hexadecimal in transmit order)<br>20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F |
| PMK | (Sequence of octets in hexadecimal in operation order)<br>C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF |

Table 79 contains the KCK and PTK octet sequences generated from the parameters in Table 78.

**Table 79 — KCK and PTK**

| Parameter | Value |
|---|---|
| KCK | (Sequence of octets in hexadecimal in operation order)<br>50 C9 32 81 90 3A 6E CB 3F 91 DC A8 57 05 59 DB |
| PTK | (Sequence of octets in hexadecimal in operation order)<br>D2 B6 FA 70 FD D1 00 84 B5 AB 1A F9 04 E7 5D CA |

## D.2 4-way handshake MIC generation

**Table 80 — 4-way handshake MIC generation parameters**

| Parameter | Value |
|---|---|
| InitiatorDevAddr | 0xDEAD |
| ResponderDevAddr | 0xBEEF |
| PTKID | 0xDEAD32 |
| Message Number | 2 (decimal) |
| Status Code | 0 (decimal) |
| MKID | (Sequence of octets in hexadecimal in transmit order)<br>F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 FA FB FC FD FE FF |
| I-Nonce | (Sequence of octets in hexadecimal in transmit order)<br>10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F |
| R-Nonce | (Sequence of octets in hexadecimal in transmit order)<br>20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F |
| KCK | (Sequence of octets in hexadecimal in operation order)<br>50 C9 32 81 90 3A 6E CB 3F 91 DC A8 57 05 59 DB |

The octets comprising the frame payload of the PTK command frame, in the order passed to the PHY SAP, are:

| | | | | |
|---|---|---|---|---|
| 02 | | | | (Message Number) |
| 00 | | | | (Status Code) |
| 32 AD DE | | | | (PTKID) |
| 00 00 00 00 | 00 00 00 00 | 00 00 00 | | (Reserved) |
| F0 F1 F2 F3 | F4 F5 F6 F7 | F8 F9 FA FB | FC FD FE FF | (MKID) |
| 20 21 22 23 | 24 25 26 27 | 28 29 2A 2B | 2C 2D 2E 2F | (R-Nonce) |
| 74 5E 5C 73 | F8 86 26 DE | | | (MIC) |

## D.3 Non-secure frame example

**Table 81 — Field values for a non-secure frame**

| MAC Fields | | Value | |
|---|---|---|---|
| Frame Control | Protocol Version (b2-b0) | 0 | 0x00E0 |
| | Secure (b3) | 0 | |
| | ACK Policy (b5-b4) | 2 (B-ACK) | |
| | Frame Type (b8-b6) | 3 (Data) | |
| | Frame Subtype / Delivery ID (b12-b9) | 0 | |
| | Retry (b13) | 0 | |
| | Reserved (b15-b14) | 0 | |
| Dest Addr | | 0xBEEF | |
| SrcAddr | | 0xDEAD | |
| Sequence Control | Fragment Number (b2-b0) | 0 | 0x0178 |
| | Sequence Number (b13-b3) | 47 (decimal) | |
| | More Fragments (b14) | 0 | |
| | Reserved (b15) | 0 | |
| Access Information | Duration (b13-b0) | 52 (decimal) | 0x8034 |
| | More Frames (b14) | 0 | |
| | Access Method (b15) | 1 | |
| Payload | | (Sequence of octets in hexadecimal in transmit order)<br>00 01 02 03 04 05 06 07 08 09 0A 0B<br>0C 0D 0E 0F 10 11 12 13 | |

The octets comprising the MAC Header, the Frame Payload and the FCS are passed to the PHY SAP in the following order:

| | | | |
|---|---|---|---|
| E0 00 EF BE | AD DE 78 01 | 34 80 | (MAC Header) |
| 00 01 02 03 | 04 05 06 07 | 08 09 0A 0B | (Payload) |
| 0C 0D 0E 0F | 10 11 12 13 | | (Payload) |
| A4 FF DD 3B | | | (FCS) |

## D.4 Secure frame example with EO = 0

**Table 82 — Field values for a secure frame with EO = 0**

| MAC Fields | | Value | |
|---|---|---|---|
| Frame Control | Protocol Version (b2-b0) | 0 | 0x00E8 |
| | Secure (b3) | 1 | |
| | ACK Policy (b5-b4) | 2 (B-ACK) | |
| | Frame Type (b8-b6) | 3 (Data) | |
| | Frame Subtype / Delivery ID (b12-b9) | 0 | |
| | Retry (b13) | 0 | |
| | Reserved (b15-b14) | 0 | |
| Dest Addr | | 0xBEEF | |
| SrcAddr | | 0xDEAD | |
| Sequence Control | Fragment Number (b2-b0) | 0 | 0x0178 |
| | Sequence Number (b13-b3) | 47 (decimal) | |
| | More Fragments (b14) | 0 | |
| | Reserved (b15) | 0 | |
| Access Information | Duration (b13-b0) | 52 (decimal) | 0x8034 |
| | More Frames (b14) | 0 | |
| | Access Method (b15) | 1 | |
| Temporal Key Identifier | | 0xDEAD32 | |
| Encryption Offset | | 0 | 0x0000 |
| Secure Frame Number | | 0x001122334455 | |
| Payload | | (Sequence of octets in hexadecimal in transmit order)<br>00 01 02 03 04 05 06 07 08 09 0A 0B<br>0C 0D 0E 0F 10 11 12 13 | |

Using the PTK of D.1, the octets comprising the MAC Header, the Security Header, the Secure Payload, the MIC and the FCS are passed to the PHY SAP in the following order:

| | | | |
|---|---|---|---|
| E8 00 EF BE | AD DE 78 01 | 34 80 | (MAC Header) |
| 32 AD DE 00 | 00 00 55 44 | 33 22 11 00 | (Security Header) |
| BA 68 93 02 | EE 86 0E 58 | A3 70 74 71 | (Secure Payload) |
| 60 E7 B5 95 | 51 8F F7 B5 | | (Secure Payload) |
| 2C 89 02 11 | F3 B1 37 0B | | (MIC) |
| E9 CB AB 31 | | | (FCS) |

## D.5 Secure frame example with EO = payload length

**Table 83 — Field values for a secure frame with EO = payload length**

| MAC Fields | | Value | |
|---|---|---|---|
| Frame Control | Protocol Version (b2-b0) | 0 | 0x00E8 |
| | Secure (b3) | 1 | |
| | ACK Policy (b5-b4) | 2 (B-ACK) | |
| | Frame Type (b8-b6) | 3 (Data) | |
| | Frame Subtype / Delivery ID (b12-b9) | 0 | |
| | Retry (b13) | 0 | |
| | Reserved (b15-b14) | 0 | |
| Dest Addr | | 0xBEEF | |
| SrcAddr | | 0xDEAD | |
| Sequence Control | Fragment Number (b2-b0) | 4 | 0x017C |
| | Sequence Number (b13-b3) | 47 (decimal) | |
| | More Fragments (b14) | 0 | |
| | Reserved (b15) | 0 | |
| Access Information | Duration (b13-b0) | 52 (decimal) | 0x8034 |
| | More Frames (b14) | 0 | |
| | Access Method (b15) | 1 | |
| Temporal Key Identifier | | 0xDEAD32 | |
| Encryption Offset | | 20 (decimal) | 0x0014 |
| Secure Frame Number | | 0x001122334456 | |
| Payload | | (Sequence of octets in hexadecimal in transmit order)<br>00 01 02 03 04 05 06 07 08 09 0A 0B<br>0C 0D 0E 0F 10 11 12 13 | |

Using the PTK of D.1, the octets comprising the MAC Header, the Security Header, the Secure Payload, the MIC and the FCS are passed to the PHY SAP in the following order:

| | | | |
|---|---|---|---|
| E8 00 EF BE | AD DE 7C 01 | 34 80 | (MAC Header) |
| 32 AD DE 00 | 14 00 56 44 | 33 22 11 00 | (Security Header) |
| 00 01 02 03 | 04 05 06 07 | 08 09 0A 0B | (Secure Payload) |
| 0C 0D 0E 0F | 10 11 12 13 | | (Secure Payload) |
| EE C3 7E 15 | 3C AD 20 0F | | (MIC) |
| EE BF E7 0C | | | (FCS) |

## D.6 Secure frame example with EO = 12

**Table 84 — Field values for a secure frame with EO = 12**

| MAC Fields | | Value | |
|---|---|---|---|
| Frame Control | Protocol Version (b2-b0) | 0 | 0x00E8 |
| | Secure (b3) | 1 | |
| | ACK Policy (b5-b4) | 2 (B-ACK) | |
| | Frame Type (b8-b6) | 3 (Data) | |
| | Frame Subtype / Delivery ID (b12-b9) | 0 | |
| | Retry (b13) | 0 | |
| | Reserved (b15-b14) | 0 | |
| Dest Addr | | 0xBEEF | |
| SrcAddr | | 0xDEAD | |
| Sequence Control | Fragment Number (b2-b0) | 0 | 0x0180 |
| | Sequence Number (b13-b3) | 48 (decimal) | |
| | More Fragments (b14) | 0 | |
| | Reserved (b15) | 0 | |
| Access Information | Duration (b13-b0) | 52 (decimal) | 0x8034 |
| | More Frames (b14) | 0 | |
| | Access Method (b15) | 1 | |
| Temporal Key Identifier | | 0xDEAD32 | |
| Encryption Offset | | 12 (decimal) | 0x000C |
| Secure Frame Number | | 0x001122334457 | |
| Payload | | (Sequence of octets in hexadecimal in transmit order)<br>00 01 02 03 04 05 06 07 08 09 0A 0B<br>0C 0D 0E 0F 10 11 12 13 | |

Using the PTK of D.1, the octets comprising the MAC Header, the Security Header, the Secure Payload, the MIC and the FCS are passed to the PHY SAP in the following order:

| | | | |
|---|---|---|---|
| E8 00 EF BE | AD DE 80 01 | 34 80 | (MAC Header) |
| 32 AD DE 00 | 0C 00 57 44 | 33 22 11 00 | (Security Header) |
| 00 01 02 03 | 04 05 06 07 | 08 09 0A 0B | (Secure Payload) |
| 79 AF AC F2 | 3F 94 9A FB | | (Secure Payload) |
| 03 5D 76 0A | 32 8F 04 E6 | | (MIC) |
| 11 10 72 C2 | | | (FCS) |

## D.7 Beacon frame example

**Table 85 — Field values for a beacon frame**

| MAC Fields | | Value | |
|---|---|---|---|
| Frame Control | Protocol Version (b2-b0) | 0 | 0x0000 |
| | Secure (b3) | 0 | |
| | ACK Policy (b5-b4) | 0 (No-ACK) | |
| | Frame Type (b8-b6) | 0 (Beacon) | |
| | Frame Subtype / Delivery ID (b12-b9) | 0 | |
| | Retry (b13) | 0 | |
| | Reserved (b15-b14) | 0 | |
| Dest Addr | | 0xFFFF | |
| SrcAddr | | 0xDEAD | |
| Sequence Control | Fragment Number (b2-b0) | 0 | 0x0DF0 |
| | Sequence Number (b13-b3) | 446 (decimal) | |
| | More Fragments (b14) | 0 | |
| | Reserved (b15) | 0 | |
| Access Information | Duration (b13-b0) | 0 | 0x0000 |
| | More Frames (b14) | 0 | |
| | Access Method (b15) | 0 | |

The payload of a beacon frame consists of a Beacon Parameters field followed by a sequence of information elements.

**Table 86 — Field values for the Beacon Parameters field**

| | | | | |
|---|---|---|---|---|
| Beacon Parameters | Device Identifier | | 00-14-EF-01-23-45 | (Sequence of octets in hexadecimal in transmit order) |
| | Beacon Slot Number | | 3 | |
| | Device Control | Movable (b0) | 0 | 00 14 EF 01 23 45 03 80 |
| | | Signalling Slot (b1) | 0 | |
| | | Reserved (b5-b2) | 0 | |
| | | Security (b7-b6) | 2 | |

**Table 87 — Field values for a BPOIE**

| | | | |
|---|---|---|---|
| BPOIE | Element ID | 1 | (Sequence of octets in hexadecimal in transmit order)<br><br>01 0B 0E 10 09 00 00 CE 0A 01 C0 FF FF |
| | Length | 11 (decimal) | |
| | BP Length | 14 (decimal) | |
| | Beacon Slot Info Bitmap | Slot 0: 0<br>Slot 1: 0<br>Slot 2: 1<br>Slot 3: 0<br>Slot 4: 1<br>Slot 5: 2<br>Slots 6-13: 0 | |
| | Dev Addrs | Slot 2: 0x0ACE<br>Slot 4: 0xC001<br>Slot 5: 0xFFFF | |

**Table 88 — Field values for a PCA Availability IE**

| | | | |
|---|---|---|---|
| PCA Availability IE | Element ID | | 2 | (Sequence of octets in hexadecimal in transmit order)<br><br>02 05 01 C0 FF FF 3F |
| | Length | | 5 | |
| | Interpretation | TIM IE Required (b0) | 1 | |
| | | Reserved (b7-b1) | 0 | |
| | PCA Availability Bitmap | | Slots 0-5: 0<br>Slots 6-29: 1<br>Slots 30-255: 0 | |

**Table 89 — Field values for a DRP IE**

| | | | | |
|---|---|---|---|---|
| DRP IE | Element ID | | 9 | (Sequence of octets in hexadecimal in transmit order)<br><br>09 08 19 0E CE 0A FE FF 00 C0 |
| | Length | | 8 | |
| | DRP Control | Reservation Type (b2-b0) | 1 (Hard) | |
| | | Stream Index (b5-b3) | 3 | |
| | | Reason Code (b8-b6) | 0 (Accepted) | |
| | | Reservation Status (b9) | 1 | |
| | | Owner (b10) | 1 | |
| | | Conflict Tiebreaker (b11) | 1 | |
| | | Unsafe (b12) | 0 | |
| | | Reserved (b15-b13) | 0 | |
| | Target/Owner DevAddr | | 0x0ACE | |
| | DRP Allocation 1 | Zone Bitmap | Zone 0: 0<br>Zones 1-15: 1 | |
| | | MAS Bitmap | MASs 0-13: 0<br>MASs 14-15: 1 | |

**Table 90 — Field values for a MAC Capabilities IE**

| | | | | |
|---|---|---|---|---|
| MAC Capabilities IE | Element ID | | 12 (decimal) | (Sequence of octets in hexadecimal in transmit order)<br><br>0C 02 8B 01 |
| | Length | | 2 | |
| | MAC Capability Bitmap (octet 0) | PCA (b0) | 1 | |
| | | Hard DRP (b1) | 1 | |
| | | Soft DRP (b2) | 0 | |
| | | Block ACK (b3) | 1 | |
| | | Explicit DRP negotiation (b4) | 0 | |
| | | Hibernation anchor (b5) | 0 | |
| | | Probe (b6) | 0 | |
| | | Link feedback (b7) | 1 | |
| | MAC Capability Bitmap (octet 1) | Range measurement (b0) | 1 | |
| | | Reserved (b7-b1) | 0 | |

**Table 91 — Field values for an Identification IE**

| | | | | |
|---|---|---|---|---|
| Identification IE | Element ID | | 19 (decimal) | (Sequence of octets in hexadecimal in transmit order)<br><br>13 13 00 03<br>00 14 EF 02<br>0C 4D 00 61<br>00 63 00 44<br>00 65 00 76<br>00 |
| | Length | | 19 (decimal) | |
| | Device Information 1 | Device Information Type | 0 (Vendor ID) | |
| | | Device Information Length | 3 | |
| | | Device Information Data | 00-14-EF | |
| | Device Information 2 | Device Information Type | 2 (Name String) | |
| | | Device Information Length | 12 (decimal) | |
| | | Device Information Data | “MacDev” | |

The octets comprising the MAC Header, the Frame Payload and the FCS are passed to the PHY SAP in the following order:

| | | | |
|---|---|---|---|
| 00 00 FF FF | AD DE F0 0D | 00 00 | (MAC Header) |
| 00 14 EF 01 | 23 45 03 80 | | (Payload – Beacon Parameters) |
| 01 0B 0E 10 | 09 00 00 CE | 0A 01 C0 FF | ( – BPOIE) |
| FF | | | |
| 02 05 01 C0 | FF FF 3F | | ( – PCA Availability IE) |
| 09 08 19 0E | CE 0A FE FF | 00 C0 | ( – DRP IE) |
| 0C 02 8B 01 | | | ( – MAC Capabilities IE) |
| 13 13 00 03 | 00 14 EF 02 | 0C 4D 00 61 | ( – Identification IE) |
| 00 63 00 44 | 00 65 00 76 | 00 | |
| 4B B5 CA 2F | | | (FCS) |

## D.8 FCS field example

This subclause provides details of the CRC polynomials in the FCS field calculation described in 7.2.7 for the secure frame example in D.4.

The frame payload expressed in hexadecimal form, in the order the octets are delivered to the PHY SAP, is:

32 AD DE 00 00 00 55 44 33 22 11 00 BA 68 93 02

EE 86 0E 58 A3 70 74 71 60 E7 B5 95 51 8F F7 B5

2C 89 02 11 F3 B1 37 0B

The coefficients of the terms of the corresponding message polynomial, from the highest-order to the lowest-order term, are:

0100 1100 1011 0101 0111 1011 0000 0000 … 1100 1111 1000 1101 1110 1100 1101 0000

The message polynomial is:

$$x^{318} + x^{315} + x^{314} + x^{311} + x^{309} + x^{308} + x^{306} + x^{304} + x^{302} + x^{301} + x^{300} + x^{299} + x^{297} + x^{296} + \ldots + x^{31} + x^{30} + x^{27} + x^{26} + x^{25} + x^{24} + x^{23} + x^{19} + x^{18} + x^{16} + x^{15} + x^{14} + x^{13} + x^{11} + x^{10} + x^{7} + x^{6} + x^{4}$$

The CRC calculation results in the following polynomial:

$$x^{31} + x^{28} + x^{26} + x^{25} + x^{24} + x^{23} + x^{22} + x^{20} + x^{17} + x^{16} + x^{15} + x^{14} + x^{12} + x^{10} + x^{8} + x^{7} + x^{3} + x^{2}$$

The corresponding coefficients are:

1001 0111 1101 0011 1101 0101 1000 1100

The FCS field expressed in hexadecimal form, in the order delivered to the PHY SAP, is:

E9 CB AB 31

# Annex E (informative) Bibliography

[B1] "Guidelines for use of a 48-bit Extended Unique Identifier (EUI-48™)", http://standards.ieee.org/regauth/oui/tutorials/EUI48.html

[B2] IEEE 100, *The Authoritative Dictionary of IEEE Standards Terms*, Seventh Edition, New York, Institute of Electrical and Electronics Engineers, Inc.

[B3] IEEE STD 802.3, 1998 *Local and Metropolitan Area Networks EtherType Field Tutorial*, Rev 0.7, New York, Institute of Electrical and Electronics Engineers, Inc.

[B4] ISO/IEC 7498-1:1994, Information Technology—Open Systems Interconnection—Basic Reference Model: The Basic Model.[7]

[B5] WiMedia Assigned Numbers, WiMedia Alliance.

[7] ISO/IEC publications are available from the ISO Central Secretariat, Case Postale 56, 1 rue de Varembé, CH-1211, Genève 20, Switzerland/Suisse. ISO/IEC publications are also available in the United States from the Sales Department, American National Standards Institute, 11 West 42nd Street, 13th Floor, New York, NY 10036, USA.