Freescale Semiconductor, Inc.
Reference Manual

Document Number: 802154MPAPIRM
Rev. 1, 07/2015

Freescale IEEE 802.15.4 MACPHY Software

Application Programming Interface

1 Introduction

The purpose of this document is to fully describe the API (Application Programming Interface) of the IEEE 802.15.4 Multiple PAN MAC and PHY Layers implementation for Real-Time Operating Systems, and the adjacent modules that contribute to the full functionality of the system.

1.1 Scope and objective

This document covers the user interface for the MACPHY library on RTOSes, presenting the set of ***function calls*** needed to drive the MACPHY library according to the IEEE Std. 802.15.4-2006 and IEEE Std. 802.15.4-2011 standard, for all supported hardware platforms.

Contents

1.2 Audience description

This document is intended for software development engineers, project managers, test engineers, and anyone else that has to understand how to use the MACPHY.

1.3 References

Table 1 References

Title	Reference	Version	Location
802.15.4-2006 - IEEE Standard for Information technology -- Local and metropolitan area networks -- Specific requirements -- Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (WPANs)	IEEE Std 802.15.4-2006	2006	http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=11161
IEEE Standard for Local and metropolitan area networks -- Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs)	IEEE Std 802.15.4™-2011(Revision of IEEE Std 802.15.4-2006)	2011	http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=6012485
IEEE Standard for Local and metropolitan area networks -- Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs) Amendment 1: MAC sublayer	IEEE Std 802.15.4e™-2012 (Amendment to IEEE Std 802.15.4™-2011)	2012	http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6185525
IEEE Standard for Local and metropolitan area networks -- Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs) Amendment 3: Physical Layer (PHY) Specifications for Low-Data-Rate, Wireless, Smart Metering Utility Networks	IEEE Std 802.15.4g™-2012 (Amendment to IEEE Std 802.15.4™-2011)	2012	http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6190698

1.4 Definitions, acronyms, and abbreviations

Table 2 Acronyms

Term / acronym	Definition
ARM	Advanced RISC Machine (computer processor)
BE	Backoff Exponent
CCA	Clear Channel Assessment
CSMA-CA	Carrier Sense Multiple Access with Collision Avoidance
DSN	Data Sequence Number
LQI	Link Quality Indication
MAC	Medium Access Control Layer
MCPS	MAC Common Part Sublayer
MCPS-SAP	MAC Common Part Sublayer Service Access Point
MFR	MAC Footer
MHR	MAC Header
MLME	MAC subLayer Management Entity
MLME-SAP	MAC subLayer Management Entity Service Access Point
MPDU	MAC protocol data unit
MSDU	MAC service data unit
PAN	Personal Area Network
PD	PHY data
PD-SAP	PHY data service access point
PHY	Physical Layer
PIB	PAN information base
PLME	Physical Layer Management Entity
PLME-SAP	Physical Layer Management Entity Service Access Point
PPDU	PHY Protocol Data Unit
PSDU	PHY Service Data Unit
RTOS	Real-Time Operating System
SAP	Service Access Point
SPDU	SSCS protocol data units
SSCS	Service-specific convergence sublayer

2 Functional description

The MACPHY will be available for MCR20 that integrates an IEEE 802.15.4-2006 standard-compliant 2.4 GHz transceiver, which can be connected to a 32-bit MCU via an SPI bus. A newly added component is the sequence manager. This is a hardware state machine, which controls the timing of all transmit, receive, and CCA operations for the MCR20 controller. The main features of the sequence manager are: hardware polling accelerator, continuous CCA, capability to conditionally perform CCA before every transmit operation, capability to conditionally transmit ACK frame after every receive operation, dual-PAN operation, active promiscuous mode.

The executed sequences can be simple transceiver operations, for example, transmit one frame, or perform a CCA on channel, or complex operations, for example, a "TR" sequence calls for a transmit frame operation followed by a receive frame operation.

2.1 Inter-layer communication

The system architecture is as follows:

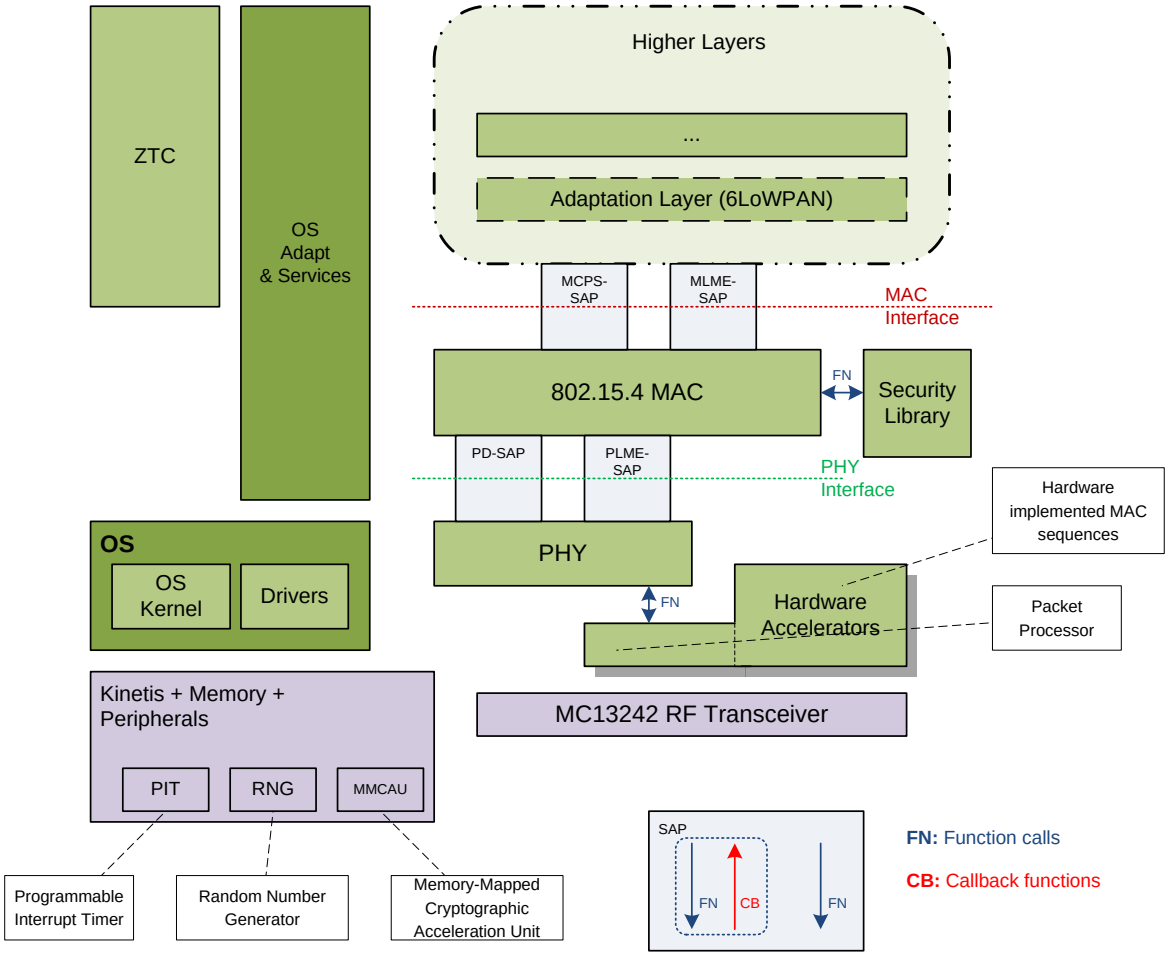

Figure 1 System software architecture

In this structure, the interactions between the layers of the stack are in terms of SAPs, implemented as primitives that represent the logical exchange of user data and control information. The scope of this document is to describe the interactions located conceptually.

These SAP primitives are implemented as asynchronous and synchronous services, as shown in the figure below.

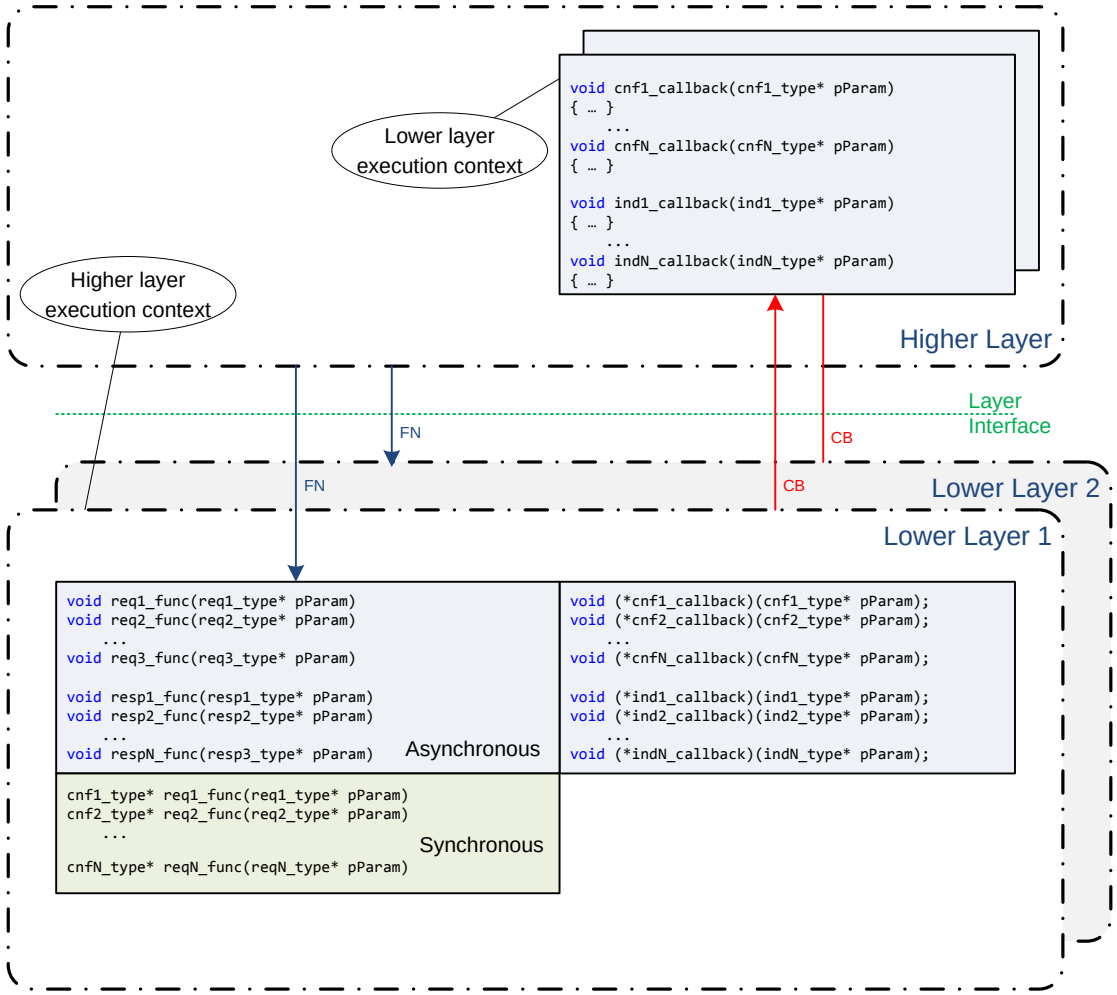

Figure 2 Layer interface and implementation for multiple PANs

2.1.1 Asynchronous services

Asynchronous MAC primitives are implemented as pairs of *direct function calls* and *function callbacks*.

The *function calls* are implemented and exposed directly through their prototypes by the **lower layer (callee)**.

The *Request* or *Response* primitive is the argument in the function's prototype.

The *function callbacks* are exposed by the **lower layer** and implemented by the **higher layer (caller)**. In this case, the higher layer will register its callbacks by sending their pointers to the lower layer through a dedicated function, provided by the lower layer.

The *Confirm* or *Indication* primitive is the argument in the callback's prototype.

2.1.2 Synchronous services

Synchronous services are implemented as ***direct function calls***, for which the *Request* primitive is the argument in the function's prototype, and the *Confirm* primitive is the result of the function.

The lower layer exposes these services directly through their prototypes.

2.2 Concept of primitives

There are four types of primitives used in this document:

REQUEST (Req) – this type is used when a higher layer is requesting a service from a lower layer.

INDICATION (Ind) – this type is used by a lower layer, which is providing a service, to notify a higher layer of activities that are relevant for that higher layer. Indications may indicate continued operation over a period of time (regularly spaced in time), or they may indicate asynchronous events from the lower layer.

RESPONSE (Res) – this type is used by a higher layer to complete a procedure previously invoked by an indication primitive.

CONFIRM (Cnf) – this type is used by a lower layer, which is providing the requested service, to confirm to a higher layer that the activity has been completed.

3 IEEE 802.15.4 MAC interface

The MAC sublayer enables the transmission of MAC frames through the use of the physical channel. Besides the data service, it offers a management interface and manages access to the physical channel and network beaconing. It also controls frame validation, guarantees time slots, and handles node associations. It also provides means for application-appropriate security mechanisms.

The communication with the upper protocol layers is carried out through a standard IEEE 802.15.4-2006 MAC interface, described hereafter. Some features of the IEEE 802.15.4-2011 and its extensions are also described in this document.

The MAC sublayer provides two services: the MAC data service, and the MAC management service interfacing to the MAC sublayer management entity (MLME) service access point (SAP) (known as MLME-SAP). The MAC data service enables the transmission and reception of MAC protocol data units (MPDUs) across the PHY data service.

The MAC Layer interfaces with the Network Layer through a unified set of SAP **function calls** and **function callbacks**, specific to the MAC services.

If the interface primitives are implemented as *function calls*, the Network Layer calls the exposed functions (provided by the MAC Layer) to issue commands / requests.

If the interface primitives are implemented as *function callbacks*, these are implemented by the Network Layer and registered its callbacks by sending their pointers to the MAC layer through a dedicated function.

3.1 Constant macro definitions

The following defines refer to the entities used in the MAC API elements.

3.1.1 gMaxScanResults_c

This constant is proprietary. It describes the maximum number of PAN Descriptors that can be returned during the Scan procedure.

Synopsis:

```
#define gMaxScanResults_c (10)
```

3.1.2 gScanResultsPerBlock_c

This constant is proprietary. It describes the maximum number of PAN Descriptors that a PAN Descriptor list element can contain during a Scan procedure.

Synopsis:

```
#define gScanResultsPerBlock_c (5)
```

3.1.3 gMacFirstLogicalChannel_c

This constant defines the value of the first logical channel that the MAC layer accepts.

Synopsis:

```
#define gMacFirstLogicalChannel_c (gLogicalChannel11_c)
```

and

```
#define gMacFirstLogicalChannel_c (gLogicalChannel0_c) /* For SubGhz PHY */
```

3.1.4 gMacLastLogicalChannel_c

This constant defines the value of the last logical channel that the MAC layer accepts.

Synopsis:

```
#define gMacLastLogicalChannel_c (gLogicalChannel26_c)
```

and

```
#define gMacLastLogicalChannel_c (gLogicalChannel127_c) /* For SubGhz PHY */
```

3.1.5 gMsgMacFirst_c

This constant defines the value of the first MAC message identifier in the list of messages, from which all the subsequent messages get incremental values.

Synopsis:

```
#define gMsgMacFirst_c (gMcpsDataReq_c)
```

3.1.6 gMsgMacLast_c

This constant defines the value of the last MAC message identifier in the list of messages.

Synopsis:

```
#define gMsgMacLast_c (gMlmeBeaconCnf_c)
```

3.1.7 gMacCslTableSize_c

This constant defines the maximum number of entries in the CSL Synchronization table that specifies how many devices can the MAC layer store synchronization details for (CSL listening period and timestamp of next channel sample).

Synopsis:

```
#define gMacCslTableSize_c (10)
```

3.1.8 gMacCslMaxSequencesCnt_c

This constant defines the maximum number of CSL sequences that can be present in the MAC layer when CSL is enabled.

Synopsis:

```
#define gMacCslMaxSequencesCnt_c (12)
```

3.1.9 gMacRitTableSize_c

This constant defines the maximum number of entries in the RIT table that specifies how many devices can the MAC layer store listening schedule for (timestamp of last received RIT Data Request command and listening schedule).

Synopsis:

```
#define gMacRitTableSize_c (10)
```

3.1.10 gMacRitMaxSequencesCnt_c

This constant defines the maximum number of RIT sequences that can be present in the MAC layer when RIT is enabled.

Synopsis:

```
#define gMacRitMaxSequencesCnt_c (12)
```

3.1.11 gMacHoppingSequenceTableEntries_c

This constant defines the maximum number of channels that the MAC layer can store to hop on when TSCH is enabled.

Synopsis:

```
#define gMacHoppingSequenceTableEntries_c (128)
```

3.1.12 gMacSlotframeTableEntries_c

This constant defines the maximum number of TSCH slotframes that the MAC layer can store.

Synopsis:

```
#define gMacSlotframeTableEntries_c (2)
```

3.1.13 gMacLinkTableEntries_c

This constant defines the maximum number of TSCH links that the MAC layer can store.

Synopsis:

```
#define gMacLinkTableEntries_c (10)
```

3.1.14 gMacTschNeighborTableEntries_c

This constant defines the maximum number of TSCH time source neighbors that the MAC layer can store when TSCH is enabled. These are configured through the MLME-KEEP-ALIVE.Request primitive, and they must have the Rx Link with timekeeping option set for the same node address.

Synopsis:

```
#define gMacTschNeighborTableEntries_c (10)
```

3.1.15 gMacTschMaxPanCoordSync_c

This constant defines the maximum number of PAN Coordinators that the MAC layer can store during the TSCH Passive Scan procedure. This information is required for TSCH service enablement following the scan procedure.

Synopsis:

```
#define gMacTschMaxPanCoordSync_c (1)
```

3.1.16 gMacTschMaxTransactions_c

This constant defines the maximum number of Data Tx sequences that can be present in the MAC layer when TSCH is enabled.

Synopsis:

```
#define gMacTschMaxTransactions_c (10)
```

3.2 Data type definition

The following data types represent the parameters needed to implement the services exposed by the SAP primitives. The data types assume that the endianess of the processor architecture, with respect to the network-MAC interface, is not in the care of the upper layers. Instead, the MAC and/or lower layers are responsible for managing the endianess of multi-byte fields, such as the MAC extended address. As an example, the extended address is an ***uint64_t*** type instead of an ***uint8_t[8]*** one.

3.2.1 Enumerations definition

The following enumerations define the data types used in the MAC interface. Members of the structures that define the payload of the service and callback functions (described in detail later on) use these data types.

3.2.1.1 macMessageId_t

This type defines the unique identifiers of the messages used in the inter-layer communication. Some of the messages are defined for consistency and standard alignment only. Their functionality is superseded by functions implementing the standard primitives directly.

Synopsis:

```
typedef enum
{
    gMcpsDataReq_c = gMsgMacFirst_c,
    gMcpsDataCnf_c,
    gMcpsDataInd_c,

    gMcpsPurgeReq_c,
    gMcpsPurgeCnf_c,

    gMcpsPromInd_c,

    gMlmeAssociateReq_c,
    gMlmeAssociateInd_c,
    gMlmeAssociateRes_c,
    gMlmeAssociateCnf_c,

    gMlmeDisassociateReq_c,
    gMlmeDisassociateInd_c,
    gMlmeDisassociateCnf_c,
    gMlmeBeaconNotifyInd_c,

    gMlmeGetReq_c,
    gMlmeGetCnf_c,

    gMlmeGtsReq_c,
    gMlmeGtsCnf_c,
    gMlmeGtsInd_c,

    gMlmeOrphanInd_c,
    gMlmeOrphanRes_c,
```

```
    gMlmeResetReq_c,
    gMlmeResetCnf_c,

    gMlmeRxEnableReq_c,
    gMlmeRxEnableCnf_c,

    gMlmeScanReq_c,
    gMlmeScanCnf_c,

    gMlmeCommStatusInd_c,

    gMlmeSetReq_c,
    gMlmeSetCnf_c,

    gMlmeStartReq_c,
    gMlmeStartCnf_c,

    gMlmeSyncReq_c,

    gMlmeSyncLossInd_c,

    gMlmePollReq_c,
    gMlmePollCnf_c,

    gMlmePollNotifyInd_c,

    gMlmeSetSlotframeReq_c,
    gMlmeSetSlotframeCnf_c,

    gMlmeSetLinkReq_c,
    gMlmeSetLinkCnf_c,

    gMlmeTschModeReq_c,
    gMlmeTschModeCnf_c,

    gMlmeKeepAliveReq_c,
    gMlmeKeepAliveCnf_c,

    gMlmeBeaconReq_c,
    gMlmeBeaconCnf_c,
} macMessageId_t;
```

Constant	Description	Value
gMcpsDataReq_c	Message associated with MCPS-DATA.request primitive.	0x00
gMcpsDataCnf_c	Message associated with MCPS-DATA.confirm primitive.	0x01
gMcpsDataInd_c	Message associated with MCPS-DATA.indication primitive.	0x02
gMcpsPurgeReq_c	Message associated with MCPS-PURGE.request primitive.	0x03
gMcpsPurgeCnf_c	Message associated with MCPS-PURGE.confirm primitive.	0x04
gMcpsPromInd_c	Message associated with MCPS-PROMISCUOS.indication primitive.	0x05
gMlmeAssociateReq_c	Message associated with MLME-ASSOCIATE.request primitive.	0x06
gMlmeAssociateInd_c	Message associated with MLME-ASSOCIATE.indication primitive.	0x07
gMlmeAssociateRes_c	Message associated with MLME-ASSOCIATE.response primitive.	0x08
gMlmeAssociateCnf_c	Message associated with MLME-ASSOCIATE.confirm primitive.	0x09
gMlmeDisassociateReq_c	Message associated with MLME-DISASSOCIATE.request primitive.	0x0A
gMlmeDisassociateInd_c	Message associated with MLME-DISASSOCIATE.indication primitive.	0x0B
gMlmeDisassociateCnf_c	Message associated with MLME-DISASSOCIATE.confirm primitive.	0x0C
gMlmeBeaconNotifyInd_c	Message associated with MLME-BEACON-NOTIFY.indication primitive.	0x0D

Constant	Description	Value
gMlmeGetReq_c	Message associated with MLME-GET.request primitive.	0x0E
gMlmeGetCnf_c	Message associated with MLME-GET.confirm primitive.	0x0F
gMlmeGtsReq_c	Message associated with MLME-GTS.request primitive.	0x10
gMlmeGtsCnf_c	Message associated with MLME-GTS.confirm primitive.	0x11
gMlmeGtsInd_c	Message associated with MLME-GTS.indication primitive.	0x12
gMlmeOrphanInd_c	Message associated with MLME-ORPHAN.indication primitive.	0x13
gMlmeOrphanRes_c	Message associated with MLME-ORPHAN.response primitive.	0x14
gMlmeResetReq_c	Message associated with MLME-RESET.request primitive.	0x15
gMlmeResetCnf_c	Message associated with MLME-RESET.confirm primitive.	0x16
gMlmeRxEnableReq_c	Message associated with MLME-RX-ENABLE.request primitive.	0x17
gMlmeRxEnableCnf_c	Message associated with MLME-RX-ENABLE.confirm primitive.	0x18
gMlmeScanReq_c	Message associated with MLME-SCAN.request primitive.	0x19
gMlmeScanCnf_c	Message associated with MLME-SCAN.confirm primitive.	0x1A
gMlmeCommStatusInd_c	Message associated with MLME-COMM-STATUS.indication primitive.	0x1B
gMlmeSetReq_c	Message associated with MLME-SET.request primitive.	0x1C
gMlmeSetCnf_c	Message associated with MLME-SET.confirm primitive.	0x1D
gMlmeStartReq_c	Message associated with MLME-START.request primitive.	0x1E
gMlmeStartCnf_c	Message associated with MLME-START.confirm primitive.	0x1F
gMlmeSyncReq_c	Message associated with MLME-SYNC.request primitive.	0x20
gMlmeSyncLossInd_c	Message associated with MLME-SYNC-LOSS.indication primitive.	0x21
gMlmePollReq_c	Message associated with MLME-POLL.request primitive.	0x22
gMlmePollCnf_c	Message associated with MLME-POLL.confirm primitive.	0x23
gMlmePollNotifyInd_c	Message associated with MLME-POLL-NOTIFY.indication primitive.	0x24
gMlmeSetSlotframeReq_c	Message associated with MLME-SET-SLOTFRAME.request primitive.	0x25
gMlmeSetSlotframeCnf_c	Message associated with MLME-SET-SLOTFRAME.confirm primitive.	0x26
gMlmeSetLinkReq_c	Message associated with MLME-SET-LINK.request primitive.	0x27
gMlmeSetLinkCnf_c	Message associated with MLME-SET-LINK.confirm primitive.	0x28
gMlmeTschModeReq_c	Message associated with MLME-TSCH-MODE.request primitive.	0x29
gMlmeTschModeCnf_c	Message associated with MLME-TSCH-MODE.confirm primitive.	0x2A
gMlmeKeepAliveReq_c	Message associated with MLME-KEEP-ALIVE.request primitive.	0x2B
gMlmeKeepAliveCnf_c	Message associated with MLME-KEEP-ALIVE.confirm primitive.	0x2C
gMlmeBeaconReq_c	Message associated with MLME-BEACON.request primitive.	0x2D
gMlmeBeaconCnf_c	Message associated with MLME-BEACON.confirm primitive.	0x2E

3.2.1.2 resultType_t

This type describes the enumerations used as result values by the primitives defined in the MAC layer.

Synopsis:

```
typedef enum
{
    gSuccess_c                = 0x00,
    gPanAtCapacity_c          = 0x01,
    gPanAccessDenied_c        = 0x02,
    gSlotframeNotFound_c      = 0x03,
    gMaxSlotframesExceeded_c  = 0x04,
    gUnkonwnLink_c            = 0x05,
    gMaxLinksExceeded_c       = 0x06,
    gNoSync_c                 = 0x07,
```

```
    gRxEnableDone_c           = 0xd0,
    gDpsNotSupported_d                  = 0xd1,
    gSoundingNotSupported_d             = 0xd2,
    gComputationNeeded_d                = 0xd3,
    gRangingNotSupported_d              = 0xd4,
    gRangingActive_d                    = 0xd5,
    gRangingRequestedButNotsupported_d  = 0xd6,
    gNoRangingRequested_d               = 0xd7,
    gUnavailableDevice_c                = 0xd8,
    gUnavailableSecurityLevel_c         = 0xd9,
    gCounterError_c           = 0xdb,
    gImproperKeyType_c        = 0xdc,
    gImproperSecurityLevel_c  = 0xdd,
    gUnsupportedLegacy_c      = 0xde,
    gUnsupportedSecurity_c    = 0xdf,
    gBeaconLost_c             = 0xe0,
    gChannelAccessFailure_c   = 0xe1,
    gDenied_c                 = 0xe2,
    gDisableTrxFailure_c      = 0xe3,
    gSecurityError_c          = 0xe4,
    gFrameTooLong_c           = 0xe5,
    gInvalidGts_c             = 0xe6,
    gInvalidHandle_c          = 0xe7,
    gInvalidParameter_c       = 0xe8,
    gNoAck_c                  = 0xe9,
    gNoBeacon_c               = 0xea,
    gNoData_c                 = 0xeb,
    gNoShortAddress_c         = 0xec,
    gOutOfCap_c               = 0xed,
    gPanIdConflict_c          = 0xee,
    gRealignment_c            = 0xef,
    gTransactionExpired_c     = 0xf0,
    gTransactionOverflow_c    = 0xf1,
    gTxActive_c               = 0xf2,
    gUnavailableKey_c         = 0xf3,
    gUnsupportedAttribute_c   = 0xf4,
    gInvalidAddress_c         = 0xf5,
    gOnTimeTooLong_c          = 0xf6,
    gPastTime_c               = 0xf7,
    gTrackingOff_c            = 0xf8,
    gInvalidIndex_c           = 0xf9,
    gLimitReached_c           = 0xfa,
    gReadOnly_c               = 0xfb,
    gScanInProgress_c         = 0xfc,
    gSuperframeOverlap_c      = 0xfd
} resultType_t;
```

Constant	Description	Value
gSuccess_c	The requested operation was completed successfully. For a transmission request, this value indicates a successful transmission.	0x00
gPanAtCapacity_c	Association Status – PAN at capacity.	0x01
gPanAccessDenied_c	Association Status – PAN access denied.	0x02
gSlotframeNotFound_c	MLME-SET-SLOTFRAME Status – requested slotframe not found	0x03
gMaxSlotframesExceeded_c	MLME-SET-SLOTFRAME Status – maximum number of slotframes reached	0x04
gUnkonwnLink_c	MLME-SET-LINK Status – requested link not found	0x05

Constant	Description	Value
gMaxLinksExceeded_c	MLME-SET-LINK Status – maximum number of links reached	0x06
gNoSync_c	MLME-TSCH-MODE Status – synchronization could not be performed to the PAN Coordinator	0x07
gRxEnableDone_c	Not part of IEEE Std. 804.15.4	0xd0
gDpsNotSupported_d	MLME-DPS Status – a parameter in the request is not supported	0xd1
gSoundingNotSupported_d	MLME-SOUNDING Status – channel sounding capability is not supported	0xd2
gComputationNeeded_d	MLME-CALIBRATE Status – PHY does not support autonomous self-calibration	0xd3
gRangingNotSupported_d	MLME-RX-ENABLE Status – Rx with ranging not supported	0xd4
gRangingActive_d	MCPS-DATA – Ranging is active	0xd5
gRangingRequestedButNotsupported_d	MCPS-DATA – Ranging requested but not supported	0xd6
gNoRangingRequested_d	MCPS-DATA – PHY does not supporting ranging	0xd7
gUnavailableDevice_c	The DeviceDescriptor lookup procedure has failed for the received frame. No device which matches the input addressing mode, address, and PAN Id was found	0xd8
gUnavailableSecurityLevel_c	The SecurityLevelDescriptor lookup procedure has failed for the received frame. No SecurityLevelDescriptor was found which matches the input frmae type and command frame identifier	0xd9
gCounterError_c	The frame counter purportedly applied by the originator of the received frame is invalid.	0xdb
gImproperKeyType_c	The key purportedly applied by the originator of the received frame is not allowed to be used with that frame type according to the key usage policy of the recipient.	0xdc
gImproperSecurityLevel_c	The security level purportedly applied by the originator of the received frame does not meet the minimum security level required / expected by the recipient for that frame type.	0xdd
gUnsupportedLegacy_c	The received frame was purportedly secured using security based on IEEE Std 802.15.4-2003, and such security is not supported by this standard.	0xde
gUnsupportedSecurity_c	The security purportedly applied by the originator of the received frame is not supported.	0xdf
gBeaconLost_c	The beacon was lost following a synchronization request.	0xe0
gChannelAccessFailure_c	A transmission could not take place due to activity on the channel, for example, the CSMA-CA mechanism has failed.	0xe1
gDenied_c	The GTS request has been denied by the PAN coordinator.	0xe2
gDisableTrxFailure_c	The attempt to disable the transceiver has failed.	0xe3
gSecurityError_c	Cryptographic processing of the received secured frame failed.	0xe4
gFrameTooLong_c	Either a frame resulting from processing has a length that is greater than aMaxPHYPacketSize or the requested transaction is too large to fit in the CAP or GTS.	0xe5
gInvalidGts_c	The requested GTS transmission failed because the specified GTS either did not have a transmit GTS direction or was not defined.	0xe6

Constant	Description	Value
gInvalidHandle_c	A request to purge an MSDU from the transaction queue was made using an MSDU handle that was not found in the transaction table.	0xe7
gInvalidParameter_c	A parameter in the primitive is either not supported or is out of the valid range.	0xe8
gNoAck_c	No acknowledgment was received after *macMaxFrameRetries*.	0xe9
gNoBeacon_c	A scan operation failed to find any network beacons.	0xea
gNoData_c	No response data were available following a request.	0xeb
gNoShortAddress_c	The operation failed because a 16-bit short address was not allocated.	0xec
gOutOfCap_c	A receiver enable request was unsuccessful because it could not be completed within the CAP. It is used only to meet the backwards compatibility requirements for IEEE Std 802.15.4-2003.	0xed
gPanIdConflict_c	A PAN identifier conflict has been detected and communicated to the PAN coordinator.	0xee
gRealignment_c	A coordinator realignment command has been received.	0xef
gTransactionExpired_c	The transaction has expired and its information was discarded.	0xf0
gTransactionOverflow_c	There is no capacity to store the transaction.	0xf1
gTxActive_c	The transceiver was in the transmitter enabled state when the receiver was requested to be enabled. It is used only to meet the backwards compatibility requirements for IEEE Std 802.15.4-2003.	0xf2
gUnavailableKey_c	The key purportedly used by the originator of the received frame is not available, or, if available, the originating device is not known or is blacklisted with that particular key.	0xf3
gUnsupportedAttribute_c	A SET/GET request was issued with the identifier of a PIB attribute that is not supported.	0xf4
gInvalidAddress_c	A request to send data was unsuccessful because neither the source address parameters nor the destination address parameters were present.	0xf5
gOnTimeTooLong_c	A receiver enable request was unsuccessful because it specified a number of symbols that was longer than the beacon interval.	0xf6
gPastTime_c	A receiver enable request was unsuccessful because it could not be completed within the current superframe and was not permitted to be deferred until the next superframe.	0xf7
gTrackingOff_c	The device was instructed to start sending beacons based on the timing of the beacon transmissions of its coordinator, but the device is not currently tracking the beacon of its coordinator.	0xf8
gInvalidIndex_c	An attempt to write to a MAC PIB attribute that is in a table failed, because the specified table index was out of range.	0xf9
gLimitReached_c	A scan operation terminated prematurely because the number of PAN descriptors stored reached an implementation-specified maximum.	0xfa
gReadOnly_c	A SET/GET request was issued with the identifier of an attribute that is read only.	0xfb
gScanInProgress_c	A request to perform a scan operation failed because the MLME was in the process of performing a previously initiated scan operation.	0xfc

Constant	Description	Value
gSuperframeOverlap_c	The device was instructed to start sending beacons based on the timing of the beacon transmissions of its coordinator, but the instructed start time overlapped the transmission time of the beacon of its coordinator.	0xfd

3.2.1.3 addrModeType_t

Addressing mode type.

Synopsis:

```
typedef enum
{
    gAddrModeNoAddress_c            = 0x00,
    gAddrModeReserved_c             = 0x01,
    gAddrModeShortAddress_c         = 0x02,
    gAddrModeExtendedAddress_c      = 0x03
} addrModeType_t;
```

Constant	Description	Value
gAddrModeNoAddress_c	No address (addressing fields omitted)	0x00
gAddrModeReserved_c	Reserved value	0x01
gAddrModeShortAddress_c	16-bit short address	0x02
gAddrModeExtendedAddress_c	64-bit extended address	0x03

3.2.1.4 macTxOptions_t

Bit type used to indicate the transmission options for the MSDU.

Synopsis:

```
typedef enum
{
    gMacTxOptionsAck_c              = 0x01,
    gMacTxOptionGts_c               = 0x02,
    gMacTxOptionIndirect_c          = 0x04
} macTxOptions_t;
```

Constant	Description	Value
gMacTxOptionsAck_c	Acknowledged transmission is required	0x01
gMacTxOptionGts_c	GTS transmission is required	0x02
gMacTxOptionIndirect_c	Indirect transmission is required	0x04

3.2.1.5 macSecurityLevel_t

Security levels available to the MAC sublayer.

Synopsis:

```
typedef enum
{
    gMacSecurityNone_c              = 0x00,
    gMacSecurityMic32_c             = 0x01,
    gMacSecurityMic64_c             = 0x02,
    gMacSecurityMic128_c            = 0x03,
    gMacSecurityEnc_c               = 0x04,
    gMacSecurityEncMic32_c          = 0x05,
    gMacSecurityEncMic64_c          = 0x06,
    gMacSecurityEncMic128_c         = 0x07
} macSecurityLevel_t;
```

Constant	Description			Value
	Security attributes	Data confidentiality	Data authenticity	
gMacSecurityNone_c	None	OFF	NO (*M* = 0)	0x00
gMacSecurityMic32_c	MIC-32	OFF	YES (M = 4)	0x01
gMacSecurityMic64_c	MIC-64	OFF	YES (M = 8)	0x02
gMacSecurityMic128_c	MIC-128	OFF	YES (M = 16)	0x03
gMacSecurityEnc_c	ENC	ON	NO (M = 0)	0x04
gMacSecurityEncMic32_c	ENC-MIC-32	ON	YES (M = 4)	0x05
gMacSecurityEncMic64_c	ENC-MIC-64	ON	YES (M = 8)	0x06
gMacSecurityEncMic128_c	ENC-MIC-128	ON	YES (M = 16)	0x07

3.2.1.6 keyIdModeType_t

Values of the key identifier mode.

Synopsis:

```
typedef enum
{
    gKeyIdMode0_c                   = 0x00,
    gKeyIdMode1_c                   = 0x01,
    gKeyIdMode2_c                   = 0x02,
    gKeyIdMode3_c                   = 0x03
} keyIdModeType_t;
```

Constant	Description		Value
	Description	Key Identifier field length (octets)	
gKeyIdMode0_c	Key is determined implicitly from the originator and recipient(s) of the frame, as indicated in the frame header.	0	0x00
gKeyIdMode1_c	Key is determined from the 1-octet Key Index subfield of the Key Identifier field of the auxiliary security header in conjunction with *macDefaultKeySource*.	1	0x01
gKeyIdMode2_c	Key is determined explicitly from the 4-octet Key Source subfield and the 1-octet Key Index subfield of the Key Identifier field of the auxiliary security header.	5	0x02
gKeyIdMode3_c	Key is determined explicitly from the 8-octet Key Source subfield and the 1-octet Key Index subfield of the Key Identifier field of the auxiliary security header.	9	0x03

3.2.1.7 logicalChannelId_t

Type to identify the logical channel number.

Synopsis:

```
typedef uint8_t logicalChannelId_t;
typedef enum {
  gLogicalChannel0_c = 0,
  gLogicalChannel1_c = 1,
  gLogicalChannel2_c = 2,
  gLogicalChannel3_c = 3,
  gLogicalChannel4_c = 4,
  gLogicalChannel5_c = 5,
  gLogicalChannel6_c = 6,
  gLogicalChannel7_c = 7,
  gLogicalChannel8_c = 8,
  gLogicalChannel9_c = 9,
  gLogicalChannel10_c = 10,
  gLogicalChannel11_c = 11,
  gLogicalChannel12_c = 12,
  gLogicalChannel13_c = 13,
  gLogicalChannel14_c = 14,
  gLogicalChannel15_c = 15,
  gLogicalChannel16_c = 16,
  gLogicalChannel17_c = 17,
  gLogicalChannel18_c = 18,
  gLogicalChannel19_c = 19,
  gLogicalChannel20_c = 20,
  gLogicalChannel21_c = 21,
  gLogicalChannel22_c = 22,
  gLogicalChannel23_c = 23,
  gLogicalChannel24_c = 24,
  gLogicalChannel25_c = 25,
  gLogicalChannel26_c = 26,
  gLogicalChannel27_c = 27,
  gLogicalChannel28_c = 28,
  gLogicalChannel29_c = 29,
  gLogicalChannel30_c = 30,
  gLogicalChannel31_c = 31,
```

```
  gLogicalChannel32_c = 32,
  gLogicalChannel33_c = 33,
  gLogicalChannel34_c = 34,
  gLogicalChannel35_c = 35,
  gLogicalChannel36_c = 36,
  gLogicalChannel37_c = 37,
  gLogicalChannel38_c = 38,
  gLogicalChannel39_c = 39,
  gLogicalChannel40_c = 40,
  gLogicalChannel41_c = 41,
  gLogicalChannel42_c = 42,
  gLogicalChannel43_c = 43,
  gLogicalChannel44_c = 44,
  gLogicalChannel45_c = 45,
  gLogicalChannel46_c = 46,
  gLogicalChannel47_c = 47,
  gLogicalChannel48_c = 48,
  gLogicalChannel49_c = 49,
  gLogicalChannel50_c = 50,
  gLogicalChannel51_c = 51,
  gLogicalChannel52_c = 52,
  gLogicalChannel53_c = 53,
  gLogicalChannel54_c = 54,
  gLogicalChannel55_c = 55,
  gLogicalChannel56_c = 56,
  gLogicalChannel57_c = 57,
  gLogicalChannel58_c = 58,
  gLogicalChannel59_c = 59,
  gLogicalChannel60_c = 60,
  gLogicalChannel61_c = 61,
  gLogicalChannel62_c = 62,
  gLogicalChannel63_c = 63,
  gLogicalChannel64_c = 64,
  gLogicalChannel65_c = 65,
  gLogicalChannel66_c = 66,
  gLogicalChannel67_c = 67,
  gLogicalChannel68_c = 68,
  gLogicalChannel69_c = 69,
  gLogicalChannel70_c = 70,
  gLogicalChannel71_c = 71,
  gLogicalChannel72_c = 72,
  gLogicalChannel73_c = 73,
  gLogicalChannel74_c = 74,
  gLogicalChannel75_c = 75,
  gLogicalChannel76_c = 76,
  gLogicalChannel77_c = 77,
  gLogicalChannel78_c = 78,
  gLogicalChannel79_c = 79,
  gLogicalChannel80_c = 80,
  gLogicalChannel81_c = 81,
  gLogicalChannel82_c = 82,
  gLogicalChannel83_c = 83,
  gLogicalChannel84_c = 84,
  gLogicalChannel85_c = 85,
  gLogicalChannel86_c = 86,
  gLogicalChannel87_c = 87,
  gLogicalChannel88_c = 88,
  gLogicalChannel89_c = 89,
  gLogicalChannel90_c = 90,
```

```
  gLogicalChannel91_c = 91,
  gLogicalChannel92_c = 92,
  gLogicalChannel93_c = 93,
  gLogicalChannel94_c = 94,
  gLogicalChannel95_c = 95,
  gLogicalChannel96_c = 96,
  gLogicalChannel97_c = 97,
  gLogicalChannel98_c = 98,
  gLogicalChannel99_c = 99,
  gLogicalChannel100_c = 100,
  gLogicalChannel101_c = 101,
  gLogicalChannel102_c = 102,
  gLogicalChannel103_c = 103,
  gLogicalChannel104_c = 104,
  gLogicalChannel105_c = 105,
  gLogicalChannel106_c = 106,
  gLogicalChannel107_c = 107,
  gLogicalChannel108_c = 108,
  gLogicalChannel109_c = 109,
  gLogicalChannel110_c = 110,
  gLogicalChannel111_c = 111,
  gLogicalChannel112_c = 112,
  gLogicalChannel113_c = 113,
  gLogicalChannel114_c = 114,
  gLogicalChannel115_c = 115,
  gLogicalChannel116_c = 116,
  gLogicalChannel117_c = 117,
  gLogicalChannel118_c = 118,
  gLogicalChannel119_c = 119,
  gLogicalChannel120_c = 120,
  gLogicalChannel121_c = 121,
  gLogicalChannel122_c = 122,
  gLogicalChannel123_c = 123,
  gLogicalChannel124_c = 124,
  gLogicalChannel125_c = 125,
  gLogicalChannel126_c = 126,
  gLogicalChannel127_c = 127,
  gLogicalChannel128_c = 128,
} logicalChannelId_tag;
```

3.2.1.8 channelMask_t

Bit type to identify the logical channel number.

Synopsis:

```
typedef uint32_t channelMask_t;
```

and

```
typedef uint32_t channelMask_t[5]; /* For SubGhz SUN Phy Modes */


typedef enum
{
    gChannelMask00_c             = 0x00000001,
    gChannelMask01_c             = 0x00000002,
    gChannelMask02_c             = 0x00000004,
    gChannelMask03_c             = 0x00000008,
    gChannelMask04_c             = 0x00000010,
```

```
    gChannelMask05_c            = 0x00000020,
    gChannelMask06_c            = 0x00000040,
    gChannelMask07_c            = 0x00000080,
    gChannelMask08_c            = 0x00000100,
    gChannelMask09_c            = 0x00000200,
    gChannelMask10_c            = 0x00000400,
    gChannelMask11_c            = 0x00000800,
    gChannelMask12_c            = 0x00001000,
    gChannelMask13_c            = 0x00002000,
    gChannelMask14_c            = 0x00004000,
    gChannelMask15_c            = 0x00008000,
    gChannelMask16_c            = 0x00010000,
    gChannelMask17_c            = 0x00020000,
    gChannelMask18_c            = 0x00040000,
    gChannelMask19_c            = 0x00080000,
    gChannelMask20_c            = 0x00100000,
    gChannelMask21_c            = 0x00200000,
    gChannelMask22_c            = 0x00400000,
    gChannelMask23_c            = 0x00800000,
    gChannelMask24_c            = 0x01000000,
    gChannelMask25_c            = 0x02000000,
    gChannelMask26_c            = 0x04000000,
    gChannelMask27_c            = 0x08000000,
    gChannelMask28_c            = 0x10000000,
    gChannelMask29_c            = 0x20000000,
    gChannelMask30_c            = 0x40000000,
    gChannelMask31_c            = 0x80000000,
} channelMask_tag;
```

3.2.1.9 channelPageId_t

Bit type to identify the channel page number.

Synopsis:

```
typedef enum
{
    gChannelPageId0_c               = 0x00,
    gChannelPageId1_c               = 0x01,
    gChannelPageId2_c               = 0x02,
    gChannelPageId9_c               = 0x09,
} channelPageId_t;
```

Constant	Description		Value
	Channel number(s) (decimal)	Channel number description	
gChannelPageId0_c	0	Channel 0 is in the 868 MHz band using BPSK	0x00
	1–10	Channels 1 to 10 are in the 915 MHz band using BPSK	
	11–26	Channels 11 to 26 are in the 2.4 GHz band using O-QPSK	
gChannelPageId1_c	0	Channel 0 is in the 868 MHz band using ASK	0x01
	1–10	Channels 1 to 10 are in the 915 MHz band using ASK	
	11–26	Reserved	
gChannelPageId2_c	0	Channel 0 is in the 868 MHz band using O-QPSK	0x02
	1–10	Channels 1 to 10 are in the 915 MHz band using O-QPSK	
	11–26	Reserved	
gChannelPageId9_c	0-37	802.15.4g standard-defined SUN Phy Modes	

3.2.1.10 macCapabilityInfo_t

Synopsis:

```
typedef uint8_t macCapabilityInfo_t;
typedef enum
{
    gCapInfoAltPanCoord_c           = 0x01,
    gCapInfoDeviceFfd_c             = 0x02,
    gCapInfoPowerMains_c            = 0x04,
    gCapInfoRxWhenIdle_c            = 0x08,
    gCapInfoSecurity_c              = 0x40,
    gCapInfoAllocAddr_c             = 0x80
} macCapabilityInfo_tag;
```

Constant	Description	Value
gCapInfoAltPanCoord_c	The device is capable of becoming the PAN coordinator.	0x01
gCapInfoDeviceFfd_c	The device is an FFD.	0x02
gCapInfoPowerMains_c	The device is mains-powered, and not battery-powered.	0x04
gCapInfoRxWhenIdle_c	Receiver is on when the unit is idle.	0x08
gCapInfoSecurity_c	The device can send / receive secured MAC frames.	0x40
gCapInfoAllocAddr_c	The device asks the coordinator to allocate a 16-bit short address as a result of the association procedure.	0x80

3.2.1.11 macDisassociateReason_t

Synopsis:

```
typedef enum
{
    gDisassociateCoord_t            = 0x01,
    gDisassociateDevice_t           = 0x02
} macDisassociateReason_t;
```

Constant	Description	Value
gDisassociateCoord_t	The coordinator wants the device to leave the PAN.	0x01
gDisassociateDevice_t	The device wants to leave the PAN.	0x02

3.2.1.12 macScanType_t

This type describes the Scan procedure supported by the MAC layer.

Synopsis:

```
typedef enum
{
    gScanModeED_c                   = 0x00,
    gScanModeActive_c               = 0x01,
    gScanModePassive_c              = 0x02,
    gScanModeOrphan_c               = 0x03,
```

```
    gScanModeFastED_c               = 0x04
} macScanType_t;
```

Constant	Description	Value
gScanModeED_c	Energy Detection scan.	0x00
gScanModeActive_c	Active scan.	0x01
gScanModePassive_c	Passive scan.	0x02
gScanModeOrphan_c	Orphan scan.	0x03
gScanModeFastED_c	Fast Energy Detect	0x04

3.2.1.13 **pibId_t**

Identifier of the MAC PIB attributes.

Synopsis:

```
typedef uint8_t pibId_t;
typedef enum
{
    // Freescale specific MAC PIBs
    gMPibRole_c                                 = 0x20,
    gMPibLogicalChannel_c                       = 0x21,
    gMPibTreemodeStartTime_c                    = 0x22,
    gMPibPanIdConflictDetection_c               = 0x23,
    gMPibBeaconResponseDenied_c                 = 0x24,
    gMPibNBSuperFrameInterval_c                 = 0x25,
    gMPibBeaconPayloadLengthVendor              = 0x26,
    gMPibBeaconResponseLQIThreshold_c           = 0x27,
    gMPibUseExtendedAddressForBeacon_c          = 0x28,
    gMPibMac2003Compatibility_c                 = 0x2F,
    // In MAC2011 this is a standard PIB, in MAC2006 it is a costant. It must be
modifiable for testing. */
    gMacPibExtendedAddress_c                    = 0x30,

    // TSCH
    gMPibDisconnectTime_c                       = 0x31,
    gMPibJoinPriority_c                         = 0x32,
    gMPibASN_c                                  = 0x33,
    gMPibNoHLBuffers_c                          = 0x34,
    gMPibEBSN_c                                 = 0x35,
    gMPibTschEnabled_c                          = 0x36,
    gMPibEBIEList_c                             = 0x37,
    gMPibTimeslotTemplate_c                     = 0x38,
    gMPibHoppingSequenceList_c                  = 0x39,
    gMPibHoppingSequenceLength_c                = 0x3A,
    gMPibTschRole_c                             = 0x3B,

    // Standard MAC PIBs
    gMPibAckWaitDuration_c                      = 0x40,
    gMPibAssociationPermit_c                    = 0x41,
    gMPibAutoRequest_c                          = 0x42,
    gMPibBattLifeExt_c                          = 0x43,
    gMPibBattLifeExtPeriods_c                   = 0x44,
    gMPibBeaconPayload_c                        = 0x45,
    gMPibBeaconPayloadLength_c                  = 0x46,
    gMPibBeaconOrder_c                          = 0x47,
```

```
    gMPibBeaconTxTime_c                         = 0x48,
    gMPibBSN_c                                  = 0x49,
    gMPibCoordExtendedAddress_c                 = 0x4a,
    gMPibCoordShortAddress_c                    = 0x4b,
    gMPibDSN_c                                  = 0x4c,
    gMPibGtsPermit_c                            = 0x4d,
    gMPibMaxCSMABackoffs_c                      = 0x4e,
    gMPibMinBE_c                                = 0x4f,
    gMPibPanId_c                                = 0x50,
    gMPibPromiscuousMode_c                      = 0x51,
    gMPibRxOnWhenIdle_c                         = 0x52,
    gMPibShortAddress_c                         = 0x53,
    gMPibSuperframeOrder_c                      = 0x54,
    gMPibTransactionPersistenceTime_c           = 0x55,
    gMPibAssociatedPANCoord_c                   = 0x56,
    gMPibMaxBE_c                                = 0x57,
    gMPibMaxFrameTotalWaitTime_c                = 0x58,
    gMPibMaxFrameRetries_c                      = 0x59,
    gMPibResponseWaitTime_c                     = 0x5a,
    gMPibSyncSymbolOffset_c                     = 0x5b,
    gMPibTimestampSupported_c                   = 0x5c,
    gMPibSecurityEnabled_c                      = 0x5d,
    gMPibMinLIFSPeriod_c                        = 0x5e,
    gMPibMinSIFSPeriod_c                        = 0x5f,

    gMPibTxControlActiveDuration_c              = 0x60,
    gMPibTxControlPauseDuration_c               = 0x61,
    gMPibTxTotalDuration_c                      = 0x62,

    // CSL
    gMPibCslPeriod_c                            = 0x63,
    gMPibCslMaxPeriod_c                         = 0x64,
    gMPibCslFramePendingWait_c                  = 0x65,
    gMPibEnhAckWaitDuration_c                   = 0x66,

    // RIT
    gMPibRitPeriod_c                            = 0x67,
    gMPibRitDataWaitDuration_c                  = 0x68,
    gMPibRitTxWaitDuration_c                    = 0x69,
    gMPibRitIe_c                                = 0x6A,

    // SubGHz
    gMacPibPhyMode_c                            = 0x6B,
    gMacPibPhyCCADuration_c                     = 0x6C,
    gMacPibPhyFSKScramblePSDU_c                 = 0x6D,
    gMacPibPhyFrequencyBand_c                   = 0x6E,

    // MAC2006 Security Related PIB Attributes
    gMPibKeyTable_c                             = 0x71,
    gMPibKeyTableEntries_c                      = 0x72
    gMPibDeviceTable_c                          = 0x73,
    gMPibDeviceTableEntries_c                   = 0x74
    gMPibSecurityLevelTable_c                   = 0x75,
    gMPibSecurityLevelTableEntries_c            = 0x76
    gMPibFrameCounter_c                         = 0x77,
    gMPibAutoRequestSecurityLevel_c             = 0x78,
    gMPibAutoRequestKeyIdMode_c                 = 0x79,
    gMPibAutoRequestKeySource_c                 = 0x7a,
    gMPibAutoRequestKeyIndex_c                  = 0x7b,
```

```
    gMPibDefaultKeySource_c                     = 0x7c,
    gMPibPANCoordExtendedAddress_c              = 0x7d,
    gMPibPANCoordShortAddress_c                 = 0x7e,
    // Freescale specific MAC 2006 and MAC2011 security PIBs
    // KeyDescriptor
    gMPibKeyIdLookupList_c                      = 0x7F,
    gMPibKeyIdLookupListEntries_c               = 0x80,
    gMPibKeyDeviceList_c                        = 0x81,
    gMPibKeyDeviceListEntries_c                 = 0x82,
    gMPibKeyUsageList_c                         = 0x83,
    gMPibKeyUsageListEntries_c                  = 0x84,
    gMPibKey_c                                  = 0x85,
    // KeyUsageDescriptor
    gMPibKeyUsageFrameType_c                    = 0x86,
    gMPibKeyUsageCommnadFrameIdentifier_c       = 0x87,
    // KeyDeviceDescriptor
    gMPibKeyDeviceDescriptorHandle_c            = 0x88,
    gMPibUniqueDevice_c                         = 0x89,
    gMPibBlackListed_c                          = 0x8A,
    // SecurityLevelDescriptor
    gMPibSecLevFrameType_c                      = 0x8B,
    gMPibSecLevCommnadFrameIdentifier_c         = 0x8C,
    gMPibSecLevSecurityMinimum_c                = 0x8D,
    gMPibSecLevDeviceOverrideSecurityMinimum_c  = 0x8E,
    // DeviceDescriptor
    gMPibDeviceDescriptorPanId_c                = 0x8F,
    gMPibDeviceDescriptorShortAddress_c         = 0x90,
    gMPibDeviceDescriptorExtAddress_c           = 0x91,
    gMPibDeviceDescriptorFrameCounter_c         = 0x92,
    gMPibDeviceDescriptorExempt                 = 0x93,
    // KeyIdLookupDescriptor
    gMPibKeyIdLookupData_c                      = 0x94,
    gMPibKeyIdLookupDataSize_c                  = 0x95,
    gMPibiKeyTableCrtEntry_c                    = 0x96,
    gMPibiDeviceTableCrtEntry_c                 = 0x97,
    gMPibiSecurityLevelTableCrtEntry_c          = 0x98,
    gMPibiKeyIdLookuplistCrtEntry_c             = 0x99,
    gMPibiKeyDeviceListCrtEntry_c               = 0x9B,
    gMPibiKeyUsageListCrtEntry_c                = 0x9A,
} macPibId_t;
```

Member	Description	Value
gMPibRole_c	–	0x20
gMPibLogicalChannel_c	–	0x21
gMPibTreemodeStartTime_c	–	0x22
gMPibPanIdConflictDetection_c	–	0x23
gMPibBeaconResponseDenied_c	–	0x24
gMPibNBSuperFrameInterval_c	–	0x25
gMPibBeaconPayloadLengthVendor	–	0x26
gMPibBeaconResponseLQIThreshold_c	–	0x27
gMPibUseExtendedAddressForBeacon_c	–	0x28
gMPibMac2003Compatibility_c	–	0x2F

Member	Description	Value
gMacPibExtendedAddress_c	–	0x30
gMPibDisconnectTime_c	–	0x31
gMPibJoinPriority_c	–	0x32
gMPibASN_c	–	0x33
gMPibNoHLBuffers_c	–	0x34
gMPibEBSN_c	–	0x35
gMPibTschEnabled_c	–	0x36
gMPibEBIEList_c	–	0x37
gMPibTimeslotTemplate_c	–	0x38
gMPibHoppingSequenceList_c	–	0x39
gMPibHoppingSequenceLength_c	–	0x3A
gMPibTschRole_c	–	0x3B
gMPibAckWaitDuration_c	–	0x40
gMPibAssociationPermit_c	–	0x41
gMPibAutoRequest_c	–	0x42
gMPibBattLifeExt_c	–	0x43
gMPibBattLifeExtPeriods_c	–	0x44
gMPibBeaconPayload_c	–	0x45
gMPibBeaconPayloadLength_c	–	0x46
gMPibBeaconOrder_c	–	0x47
gMPibBeaconTxTime_c	–	0x48
gMPibBSN_c	–	0x49
gMPibCoordExtendedAddress_c	–	0x4a
gMPibCoordShortAddress_c	–	0x4b
gMPibDSN_c	–	0x4c
gMPibGtsPermit_c	–	0x4d
gMPibMaxCSMABackoffs_c	–	0x4e
gMPibMinBE_c	–	0x4f
gMPibPanId_c	–	0x50
gMPibPromiscuousMode_c	–	0x51
gMPibRxOnWhenIdle_c	–	0x52
gMPibShortAddress_c	–	0x53
gMPibSuperframeOrder_c	–	0x54
gMPibTransactionPersistenceTime_c	–	0x55
gMPibAssociatedPANCoord_c	–	0x56

Member	Description	Value
gMPibMaxBE_c	–	0x57
gMPibMaxFrameTotalWaitTime_c	–	0x58
gMPibMaxFrameRetries_c	–	0x59
gMPibResponseWaitTime_c	–	0x5a
gMPibSyncSymbolOffset_c	–	0x5b
gMPibTimestampSupported_c	–	0x5c
gMPibSecurityEnabled_c	–	0x5d
gMPibMinLIFSPeriod_c	–	0x5e
gMPibMinSIFSPeriod_c	–	0x5f
gMPibCslPeriod_c	–	0x63
gMPibCslMaxPeriod_c	–	0x64
gMPibCslFramePendingWait_c	–	0x65
gMPibEnhAckWaitDuration_c	–	0x66
gMPibRitPeriod_c	–	0x67
gMPibRitDataWaitDuration_c	–	0x68
gMPibRitTxWaitDuration_c	–	0x69
gMPibRitIe_c	–	0x6a
gMacPibPhyMode_c	–	0x6b
gMacPibPhyCCADuration_c	–	0x6c
gMacPibPhyFSKScramblePSDU_c	–	0x6d
gMacPibPhyFrequencyBand_c	–	0x6e
gMPibKeyTable_c	–	0x71
gMPibKeyTableEntries_c	–	0x72
gMPibDeviceTable_c	–	0x73
gMPibDeviceTableEntries_c	–	0x74
gMPibSecurityLevelTable_c	–	0x75
gMPibSecurityLevelTableEntries_c	–	0x76
gMPibFrameCounter_c	–	0x77
gMPibAutoRequestSecurityLevel_c	–	0x78
gMPibAutoRequestKeyIdMode_c	–	0x79
gMPibAutoRequestKeySource_c	–	0x7a
gMPibAutoRequestKeyIndex_c	–	0x7b
gMPibDefaultKeySource_c	–	0x7c
gMPibPANCoordExtendedAddress_c	–	0x7d
gMPibPANCoordShortAddress_c	–	0x7e

Member	Description	Value
gMPibKeyIdLookupList_c	–	0x7F
gMPibKeyIdLookupListEntries_c	–	0x80
gMPibKeyDeviceList_c	–	0x81
gMPibKeyDeviceListEntries_c	–	0x82
gMPibKeyUsageList_c	–	0x83
gMPibKeyUsageListEntries_c	–	0x84
gMPibKey_c	–	0x85
gMPibKeyUsageFrameType_c	–	0x86
gMPibKeyUsageCommnadFrameIdentifier_c	–	0x87
gMPibKeyDeviceDescriptorHandle_c	–	0x88
gMPibUniqueDevice_c	–	0x89
gMPibBlackListed_c	–	0x8A
gMPibSecLevFrameType_c	–	0x8B
gMPibSecLevCommnadFrameIdentifier_c	–	0x8C
gMPibSecLevSecurityMinimum_c	–	0x8D
gMPibSecLevDeviceOverrideSecurityMinimum_c	–	0x8E
gMPibDeviceDescriptorPanId_c	–	0x8F
gMPibDeviceDescriptorShortAddress_c	–	0x90
gMPibDeviceDescriptorExtAddress_c	–	0x91
gMPibDeviceDescriptorFrameCounter_c	–	0x92
gMPibDeviceDescriptorExempt	–	0x93
gMPibKeyIdLookupData_c	–	0x94
gMPibKeyIdLookupDataSize_c	–	0x95
gMPibiKeyTableCrtEntry_c	–	0x96
gMPibiDeviceTableCrtEntry_c	–	0x97
gMPibiSecurityLevelTableCrtEntry_c	–	0x98
gMPibiKeyIdLookuplistCrtEntry_c	–	0x99
gMPibiKeyDeviceListCrtEntry_c	–	0x9B
gMPibiKeyUsageListCrtEntry_c	–	0x9A

3.2.1.14 macState_t

This type describes the operational states that the MAC layer can be in.

Synopsis:

```
typedef enum
```

```
{
    gMacStateIdle_c     = 0x00,
    gMacStateBusy_c     = 0x01,
    gMacStateNotEmpty_c = 0x02
} macState_t;
```

Constant	Description	Value
gMacStateIdle_c	All the MAC instances are Idle	0x00
gMacStateBusy_c	At least one of the MAC instances has an Active sequence	0x01
gMacStateNotEmpty_c	At least one of the MAC instances has pending messages	0x02

3.2.1.15 beaconType_t

This type describes the Beacon frame types.

Synopsis:

```
typedef enum
{
    gMacBeacon_c         = 0x00,
    gMacEnhancedBeacon_c = 0x01,
}beaconType_t;
```

Constant	Description	Value
gMacBeacon_c	Beacon frame type	0x00
gMacEnhancedBeacon_c	Enhanced Beacon frame type	0x01

3.2.1.16 macPayloadIeId_t

This type describes the Payload Information Element IDs that can be set in the EB IE List PIB to be included in the Enhanced Beacon frame.

Synopsis:

```
typedef enum
{
    gMacPayloadIeIdChannelHoppingSequence_c = 0x09,
    gMacPayloadIeIdTschSynchronization_c    = 0x1a,
    gMacPayloadIeIdTschSlotframeAndLink_c   = 0x1b,
    gMacPayloadIeIdTschTimeslot_c           = 0x1c,
}macPayloadIeId_t;
```

Constant	Description	Value
gMacPayloadIeIdChannelHoppingSequence_c	Channel Hopping Sequence	0x09
gMacPayloadIeIdTschSynchronization_c	TSCH Synchronization	0x1a
gMacPayloadIeIdTschSlotframeAndLink_c	TSCH Slotframe and Link	0x1b
gMacPayloadIeIdTschTimeslot_c	TSCH Timeslot	0x1c

3.2.1.17 macLinkType_t

This type describes the TSCH Link types.

Synopsis:

```
typedef enum
```

```
{
    gMacLinkTypeNormal_c       = 0x00,
    gMacLinkTypeAdvertising_c  = 0x01,
}macLinkType_t;
```

Constant	Description	Value
gMacLinkTypeNormal_c	Normal Link type	0x00
gMacLinkTypeAdvertising_c	Advertising Link type that can be used for sending the Enhanced Beacon frames	0x01

3.2.1.18 macSetSlotframeOp_t

This type describes the TSCH Set Slotframe operation types.

Synopsis:

```
typedef enum
{
    gMacSetSlotframeOpAdd_c    = 0x00,
    gMacSetSlotframeOpDelete_c = 0x02,
    gMacSetSlotframeOpModify_c = 0x03,
}macSetSlotframeOp_t;
```

Constant	Description	Value
gMacSetSlotframeOpAdd_c	Add slotframe	0x00
gMacSetSlotframeOpDelete_c	Delete slotframe	0x02
gMacSetSlotframeOpModify_c	Modify slotframe	0x03

3.2.1.19 macSetLinkOp_t

This type describes the TSCH Set Link operation types.

Synopsis:

```
typedef enum
{
    gMacSetLinkOpAdd_c         = 0x00,
    gMacSetLinkOpDelete_c      = 0x01,
    gMacSetLinkOpModify_c      = 0x02,
}macSetLinkOp_t;
```

Constant	Description	Value
gMacSetLinkOpAdd_c	Add link	0x00
gMacSetLinkOpDelete_c	Delete link	0x01
gMacSetLinkOpModify_c	Modify link	0x02

3.2.1.20 macTschMode_t

This type describes the TSCH Mode operation types.

Synopsis:

```
typedef enum
{
    gMacTschModeOff_c          = 0x00,
```

```
    gMacTschModeOn_c            = 0x01,
}macTschMode_t;
```

Constant	Description	Value
gMacTschModeOff _c	TSCH disable	0x00
gMacTschModeOn _c	TSCH enable	0x01

3.2.1.21 macTschRole_t

This type describes the TSCH Role.

Synopsis:

```
typedef enum
{
    gMacTschRolePANCoordinator_c  = 0x00,
    gMacTschRoleDevice_c  = 0x01,
}macTschRole_t;
```

Constant	Description	Value
gMacTschRolePANCoordinator	Node is the Coordinator of the TSCH PAN	0x00
gMacTschRoleDevice_c	Node joins a TSCH PAN	0x01

3.2.1.22 phyMode_t

This type describes the supported Sub-1 GHz PHY modes.

Synopsis:

```
typedef enum
{
    gPhyMode1_c   = 0x00,
    gPhyMode2_c   = 0x01,
    gPhyMode3_c   = 0x02, /* currently not supported */
    gPhyMode3_c   = 0x03, /* currently not supported */
    gPhyMode1ARIB_c = 0x04,
    gPhyMode2ARIB_c = 0x05,
    gPhyMode3ARIB_c = 0x06,
} phyMode_t;
```

Constant	Description	Value
gPhyMode1_c	MR-FSK Operating Mode 1	0x00
gPhyMode2_c	MR-FSK Operating Mode 2	0x01
gPhyMode3_c	N/A	0x02
gPhyMode4_c	N/A	0x03
gPhyMode1ARIB_c	MR-FSK 920Mhz ARIB Operating Mode 1	0x04
gPhyMode2ARIB_c	MR-FSK 920Mhz ARIB Operating Mode 2	0x05
gPhyMode3ARIB_c	MR-FSK 920Mhz ARIB Operating Mode 3	0x06

NOTE

For ARIB PHY modes make sure the right CCA Duration values are set in macPhyCCADuration mac PIB, depending on MAC logical channel value set.

Phy Mode	CCA Duration	Channel Range
gPhyMode1ARIB _c	12 symbols 12 symbols or 500 symbols 500 symbols	0 to 9 9 to 14 14 to 37
gPhyMode2ARIB _c	500 symbols 12 symbols or 500 symbols 12 symbols	0 to 3 3 to 6 6 to 17
gPhyMode3ARIB _c	500 symbols 12 symbols or 500 symbols 12 symbols	0 to 2 2 to 4 4 to 11

3.2.1.23 phyFreqBand_t

This type describes the supported Sub-1 GHz PHY frequency band IDs.

Synopsis:

```
typedef enum
{
  gFreq169_400__169_475MHz_c = 0x00, // 169.400-169.475   (Europe)
  gFreq470__510MHz_c = 0x02,     // 470-510   (China)
  gFreq779__787MHz_c = 0x03,     // 779-787   (China)
  gFreq863__870MHz_c = 0x04,     // 863-870   (Europe)
  gFreq902__928MHz_c = 0x07,     // 902-928   (U.S.)
  gFreq920__928MHz_c = 0x09,     // 920-928   (Japan) - Includes ARIB modes
  gFreq865__867MHz_c = 0x0E,     // 865-867   (India)
} phyFreqBand_t;
```

Constant	Description	Value
gFreq169_400__169_475MHz_c	Europe – 169.400 – 169.475 MHz	0x00
gFreq470__510MHz_c	China – 470 – 510 MHz	0x02
gFreq779__787MHz_c	China – 779 – 787 MHz	0x03
gFreq863__870MHz_c	Europe – 863 – 870 MHz	0x04
gFreq902__928MHz_c	U.S. – 902 – 928 MHz	0x07
gFreq920__928MHz_c	Japan – 920 – 928 MHz	0x09
gFreq865__867MHz_c	India (Non-4G standard) – 865 – 867 MHz	0x0E

3.2.2 Structure definition

The following structures define the data types used in the MAC interface. Members of the structures that define the payload of the service and callback functions (described later in detail) use these data types.

3.2.2.1 macMsgHeader_t

This structure describes the generic MAC message header.

Synopsis:

```
typedef STRUCT macMsgHeader _tag
{
    macMessageId_t      msgType;
} macMsgHeader_t;
```

Member	Description	Value
msgType	Identifier of the message.	macMessageId_t

3.2.2.2 nwkToMcpsMessage_t

This structure describes the generic MAC message that enables communication from the NWK layer to the MCPS Service Access Point.

Synopsis:

```
typedef STRUCT nwkToMcpsMessage_tag
{
    macMessageId_t      msgType;
    UNION
    {
        mcpsDataReq_t       dataReq;
        mcpsPurgeReq_t      purgeReq;
    } msgData;
} nwkToMcpsMessage_t;
```

Member	Description	Value
msgType	Discriminator of the contained message.	macMessageId_t
dataReq	Descriptor of the MCPS-DATA.Request primitive.	mcpsDataReq_t
purgeReq	Descriptor of the MCPS-PURGE.Request primitive.	mcpsPurgeReq_t

3.2.2.3 mcpsToNwkMessage_t

This structure describes the generic MAC message that enables communication from the MCPS Service Access Point to the NWK layer.

Synopsis:

```
typedef STRUCT mcpsToNwkMessage_tag
{
    macMessageId_t      msgType;
    UNION
    {
        mcpsDataCnf_t       dataCnf;
        mcpsDataInd_t       dataInd;
        mcpsPurgeCnf_t      purgeCnf;
        mcpsPromInd_t       promInd;
    } msgData;
} mcpsToNwkMessage_t;
```

Member	Description	Value
msgType	Discriminator of the contained message.	macMessageId_t
dataCnf	Descriptor of the MCPS-DATA.Confirm primitive.	mcpsDataCnf_t
dataInd	Descriptor of the MCPS-DATA.Indication primitive.	mcpsDataInd_t
purgeCnf	Descriptor of the MCPS-PURGE.Request primitive.	mcpsPurgeCnf_t
promInd	Descriptor of the MCPS-PROMISCUOUS.Indication proprietary primitive.	mcpsPromInd_t

3.2.2.4 mlmeMessage_t

This structure describes the generic MAC message that enables communication from the NWK layer to the MLME Service Access Point.

Synopsis:

```
typedef STRUCT mlmeMessage_tag
{
    macMessageId_t      msgType;
    UNION
    {
        mlmeAssociateReq_t      associateReq;
        mlmeAssociateRes_t      associateRes;
        mlmeDisassociateReq_t   disassociateReq;
        mlmeGetReq_t            getReq;
        mlmeGtsReq_t            gtsReq;
        mlmeOrphanRes_t         orphanRes;
        mlmeResetReq_t          resetReq;
        mlmeRxEnableReq_t       rxEnableReq;
        mlmeScanReq_t           scanReq;
        mlmeSetReq_t            setReq;
        mlmeStartReq_t          startReq;
        mlmeSyncReq_t           syncReq;
        mlmePollReq_t           pollReq;
        mlmeSetSlotframeReq_t   setSlotframeReq;
        mlmeSetLinkReq_t        setLinkReq;
        mlmeTschModeReq_t       tschModeReq;
        mlmeKeepAliveReq_t      keepAliveReq;
        mlmeBeaconReq_t         beaconReq;
    } msgData;
} mlmeMessage_t;
```

Member	Description	Value
msgType	Discriminator of the contained message.	macMessageId_t
associateReq	Descriptor of the MLME-ASSOCIATE.Request primitive.	mlmeAssociateReq_t
associateRes	Descriptor of the MLME-ASSOCIATE.Response primitive.	mlmeAssociateRes_t
disassociateReq	Descriptor of the MLME-DISASSOCIATE.Request primitive.	mlmeDisassociateReq_t
getReq	Descriptor of the MLME-GET.Request primitive.	mlmeGetReq_t
gtsReq	Descriptor of the MLME-GTS.Request primitive.	mlmeGtsReq_t
orphanRes	Descriptor of the MLME-ORPHAN.Response primitive.	mlmeOrphanRes_t
resetReq	Descriptor of the MLME-RESET.Request primitive.	mlmeResetReq_t
rxEnableReq	Descriptor of the MLME-RX-ENABLE.Request primitive.	mlmeRxEnableReq_t
scanReq	Descriptor of the MLME-SCAN.Request primitive.	mlmeScanReq_t
setReq	Descriptor of the MLME-SET.Request primitive.	mlmeSetReq_t
startReq	Descriptor of the MLME-START.Request primitive.	mlmeStartReq_t
syncReq	Descriptor of the MLME-SYNC.Request primitive.	mlmeSyncReq_t
pollReq	Descriptor of the MLME-POLL.Request primitive.	mlmePollReq_t
setSlotframeReq	Descriptor of the MLME-SET-SLOTFRAME.Request primitive.	mlmeSetSlotframeReq_t
setLinkReq	Descriptor of the MLME-SET-LINK.Request primitive.	mlmeSetLinkReq_t
tschModeReq	Descriptor of the MLME-TSCH-MODE.Request primitive.	mlmeTschModeReq_t
keepAliveReq	Descriptor of the MLME-KEEP-ALIVE.Request primitive.	mlmeKeepAliveReq_t
beaconReq	Descriptor of the MLME-BEACON.Request primitive.	mlmeBeaconReq_t

3.2.2.5 nwkMessage_t

This structure describes the generic MAC message that enables communication from the MLME Service Access Point to the NWK layer.

Synopsis:

```
typedef STRUCT nwkMessage_tag
{
    macMessageId_t      msgType;
    UNION
    {
        mlmeAssociateInd_t      associateInd;
        mlmeAssociateCnf_t      associateCnf;
        mlmeDisassociateInd_t   disassociateInd;
        mlmeDisassociateCnf_t   disassociateCnf;
        mlmeBeaconNotifyInd_t   beaconNotifyInd;
        mlmeGetCnf_t            getCnf;
        mlmeGtsInd_t            gtsInd;
        mlmeGtsCnf_t            gtsCnf;
        mlmeOrphanInd_t         orphanInd;
        mlmeResetCnf_t          resetCnf;
        mlmeRxEnableCnf_t       rxEnableCnf;
        mlmeScanCnf_t           scanCnf;
        mlmeCommStatusInd_t     commStatusInd;
        mlmeSetCnf_t            setCnf;
        mlmeStartCnf_t          startCnf;
        mlmeSyncLossInd_t       syncLossInd;
        mlmePollCnf_t           pollCnf;
        mlmePollNotifyInd_t     pollNotifyInd;
        mlmeSetSlotframeCnf_t   setSlotframeCnf;
        mlmeSetLinkCnf_t        setLinkCnf;
        mlmeTschModeCnf_t       tschModeCnf;
        mlmeKeepAliveCnf_t      keepAliveCnf;
        mlmeBeaconCnf_t         beaconCnf;
    } msgData;
} nwkMessage_t;
```

Member	Description	Value
msgType	Discriminator of the contained message.	macMessageId_t
associateInd	Descriptor of the MLME-ASSOCIATE.Indication primitive.	mlmeAssociateInd_t
associateCnf	Descriptor of the MLME-ASSOCIATE.Confirm primitive.	mlmeAssociateCnf_t
disassociateInd	Descriptor of the MLME-DISASSOCIATE.Indication primitive.	mlmeDisassociateInd_t
disassociateCnf	Descriptor of the MLME-DISASSOCIATE.Confirm primitive.	mlmeDisassociateCnf_t
beaconNotifyInd	Descriptor of the MLME-BEACON-NOTIFY.Indication primitive.	mlmeBeaconNotifyInd_t
getCnf	Descriptor of the MLME-GET.Confirm primitive.	mlmeGetCnf_t
gtsInd	Descriptor of the MLME-GTS.Indication primitive.	mlmeGtsInd_t
gtsCnf	Descriptor of the MLME-GTS.Confirm primitive.	mlmeGtsCnf_t
orphanInd	Descriptor of the MLME-ORPHAN.Indication primitive.	mlmeOrphanInd_t
resetCnf	Descriptor of the MLME-RESET.Confirm primitive.	mlmeResetCnf_t
rxEnableCnf	Descriptor of the MLME-RX-ENABLE.Confirm primitive.	mlmeRxEnableCnf_t
scanCnf	Descriptor of the MLME-SCAN.Confirm primitive.	mlmeScanCnf_t
commStatusInd	Descriptor of the MLME-COMM-STATUS.Indication primitive.	mlmeCommStatusInd_t
setCnf	Descriptor of the MLME-SET.Confirm primitive.	mlmeSetCnf_t
startCnf	Descriptor of the MLME-START.Confirm primitive.	mlmeStartCnf_t
syncLossInd	Descriptor of the MLME-SYNC-LOSS.Indication primitive.	mlmeSyncLossInd_t
pollCnf	Descriptor of the MLME-POLL.Confirm primitive.	mlmePollCnf_t

Member	Description	Value
pollNotifyInd	Descriptor of the MLME-POLL-NOTIFY.Indication primitive.	mlmePollNotifyInd_t
setSlotframeCnf	Descriptor of the MLME-SET-SLOTFRAME.Confirm primitive.	mlmeSetSlotframeCnf_t
setLinkCnf	Descriptor of the MLME-SET-LINK.Confirm primitive.	mlmeSetLinkCnf_t
tschModeCnf	Descriptor of the MLME-TSCH-MODE.Confirm primitive.	mlmeTschModeCnf_t
keepAliveCnf	Descriptor of the MLME-KEEP-ALIVE.Confirm primitive.	mlmeKeepAliveCnf_t
beaconCnf	Descriptor of the MLME-BEACON.Confirm primitive.	mlmeBeaconCnf_t

3.2.2.6 macSuperframeSpec_t

The superframe specification, as specified in the received beacon frame.

Synopsis:

```
typedef STRUCT macSuperframeSpec_tag;
{
    uint16_t        beaconOrder         :4;
    uint16_t        superframeOrder     :4;
    uint16_t        finalCapSlot        :4;
    uint16_t        ble                 :1;
    uint16_t        reserved            :1;
    uint16_t        panCoordinator      :1;
    uint16_t        associationPermit   :1;
} macSuperframeSpec_t;
```

Member	Description	Value
beaconOrder	Specifies the transmission interval of the beacon.	0-15
superframeOrder	Specifies the length of time during which the superframe is active (for example, receiver enabled), including the beacon frame transmission time.	0-15
finalCapSlot	Specifies the final superframe slot utilized by the CAP.	0-15
ble	This field is set to one if the frames transmitted to the beaconing device during its CAP are required to start in (or before) the *macBattLifeExtPeriods* full backoff periods after the IFS period following the beacon. Otherwise, the BLE subfield is set to zero.	0-1
panCoordinator	This field is set to one if the beacon frame is being transmitted by the PAN coordinator. Otherwise, the PAN Coordinator subfield is set to zero.	0-1
associationPermit	This field is set to one if *macAssociationPermit* is set to TRUE (for example, the coordinator is accepting association to the PAN). The association permit bit is set to zero if the coordinator is currently not accepting association requests on its network.	0-1

3.2.2.7 panDescriptor_t

Synopsis:

```
typedef STRUCT panDescriptor_tag
{
    uint64_t            coordAddress;
    uint16_t            coordPanId;
    addrModeType_t      coordAddrMode;
    logicalChannelId_t  logicalChannel;
    resultType_t        securityFailure;
    macSuperframeSpec_t superframeSpec;
```

```
    bool_t              gtsPermit;
    uint8_t             linkQuality;
    uint8_t             timeStamp[3];
    macSecurityLevel_t  securityLevel;
    keyIdModeType_t     keyIdMode;
    uint64_t            keySource;
    uint8_t             keyIndex;
} panDescriptor_t;
```

Member	Description	Value
coordAddress	The address of the coordinator, as specified in the received beacon frame.	uint64_t
coordPANId	PAN identifier of the coordinator, as specified in the received beacon frame.	uint16_t
coordAddrMode	The coordinator addressing mode corresponding to the received beacon frame. This field can take one of the following values:	addrModeType_t
	gAddrModeNoAddress_c	0x00
	gAddrModeShortAddress_c	0x02
	gAddrModeExtendedAddress_c	0x03
	gAddrModeReserved_c	0x01
logicalChannel	The current logical channel occupied by the network. It can take one of the following values:	logicalChannelId_t
	gLogicalChannel0_c	0
	…	
	gLogicalChannel11_c	11
	gLogicalChannel12_c	12
	…	…
	gLogicalChannel26_c	26
	…	
	gLogicalChannel128_c	128
securityFailure	*gSuccess_c* if there was no error in the security processing of the frame. One of the other following status codes indicating an error in the security processing otherwise.	resultType_t
	gSuccess_c	0x00
	gCounterError_c	0xdb
	gImproperKeyType_c	0xdc
	gImproperSecurityLevel_c	0xdd
	gSecurityError_c	0xe4
	gUnavailableKey_c	0xf3
	gUnsupportedLegacy_c	0xde
	gUnsupportedSecurity_c	0xdf
superframeSpec	The superframe specification, as specified in the received beacon frame.	macSuperframeSpec_t
gtsPermit	TRUE if the beacon is from the PAN coordinator that is accepting GTS requests.	bool_t
linkQuality	The LQI, at which the network beacon was received. Lower values represent lower LQI.	uint8_t
timeStamp	The time, in symbols, at which the data were transmitted. This is a 24-bit value.	uint8_t
securityLevel	Indicates the security level to be used. It can take one of the following values:	macSecurityLevel_t
	gMacSecurityNone_c	0x00
	gMacSecurityMic32_c	0x01
	gMacSecurityMic64_c	0x02
	gMacSecurityMic128_c	0x03

Member	Description	Value
	gMacSecurityEnc_c	0x04
	gMacSecurityEncMic32_c	0x05
	gMacSecurityEncMic64_c	0x06
	gMacSecurityEncMic128_c	0x07
keyIdMode	This parameter is ignored if the **securityLevel** parameter is set to *gMacSecurityNone_c.* It can take one of the following values:	keyIdModeType_t
	gKeyIdMode0_c	0x00
	gKeyIdMode1_c	0x01
	gKeyIdMode2_c	0x02
	gKeyIdMode3_c	0x03
keySource	Indicates the originator of the key to be used. Ignored if the **keyIdMode** parameter is ignored or set to *gKeyIdMode0_c.*	uint64_t
keyIndex	Indicates the index of the key to be used. Ignored if the **keyIdMode** parameter is ignored or set to *gKeyIdMode0_c.*	uint16_t

3.2.2.8 panDescriptorBlock_t

This type describes a dynamic list that provides the PAN descriptor list when returning the result of a specific Scan procedure.

Synopsis:

```
typedef STRUCT panDescriptor_tag
{
  panDescriptor_t               panDescriptorList[gScanResultsPerBlock_c];
  uint8_t                       panDescriptorCount;
  struct panDescriptorBlock_tag *pNext;
} panDescriptor_t;
```

Member	Description	Value
panDescriptorList	Array containing PAN Descriptors.	panDescriptor_t
panDescriptorCount	Length of the PAN Descriptor list.	uint8_t
pNext	Pointer to the next array structure element in the PAN Descriptor list.	struct panDescriptorBlock_tag *

3.2.2.9 keyIdLookupDescriptor_t

Synopsis:

```
typedef struct keyIdLookupDescriptor_tag
{
    keyIdModeType_t             keyIdMode;
    uint64_t                    keySource;
    uint8_t                     keyIndex;
    addrModeType_t              deviceAddrMode;
    uint16_t                    devicePANId;
    uint64_t                    deviceAddress;
} keyIdLookupDescriptor_t;
```

Member	Description	Value
keyIdMode	–	keyIdModeType_t
keySource	–	uint64_t
keyIndex	–	uint8_t
deviceAddrMode	–	addrModeType_t

Member	Description	Value
devicePANId	–	uint16_t
deviceAddress	–	uint64_t

3.2.2.10 deviceDescriptor_t

Synopsis:

```
typedef struct deviceDescriptor_tag
{
    uint16_t                PANId;
    uint16_t                shortAddress;
    uint64_t                extAddress;
    uint32_t                frameCounter;
    bool_t                  exempt;
} deviceDescriptor_t;
```

Member	Description	Value
PANId	–	uint16_t
shortAddress	–	uint16_t
extAddress	–	uint64_t
frameCounter	–	uint32_t
exempt	–	bool_t

3.2.2.11 keyDeviceDescriptor_t

Synopsis:

```
typedef struct keyDeviceDescriptor_tag
{
    uint8_t                 deviceDescriptorHandle;
    bool_t                  uniqueDevice;
    bool_t                  blackListed;
} keyDeviceDescriptor_t;
```

Member	Description	Value
deviceDescriptorHandle	–	uint8_t
uniqueDevice	–	bool_t
blackListed	–	bool_t

3.2.2.12 keyUsageDescriptor_t

Synopsis:

```
typedef struct keyUsageDescriptor_tag
{
    uint8_t                   frameType;
    uint8_t                   commandFrameIdentifier;
} keyUsageDescriptor_t;
```

Member	Description	Value

frameType	–	uint8_t
commandFrameIdentifier	–	uint8_t

3.2.2.13 keyDescriptor_t

Synopsis:

```
typedef struct keyDescriptor_tag
{
    keyIdLookupDescriptor_t*  keyIdLookupList;
    uint8_t                   keyIdLookupListEntries;
    keyDeviceDescriptor_t*    keyDeviceList;
    uint8_t                   keyDeviceListEntries;
} keyDescriptor_t;
```

Member	Description	Value
keyIdLookupList	–	keyIdLookupDescriptor_t*
keyIdLookupListEntries	–	uint8_t
keyDeviceList	–	keyDeviceDescriptor_t*
keyDeviceListEntries	–	uint8_t

3.2.2.14 securityLevelDescriptor_t

Synopsis:

```
typedef struct securityLevelDescriptor_tag
{
    uint8_t                   frameType;
    uint8_t                   commandFrameIdentifier;
    uint8_t                   securityMinimum;
    bool_t                    deviceOverrideSecurityMinimum;
} securityLevelDescriptor_t;
```

Member	Description	Value
frameType	–	uint8_t
commandFrameIdentifier	–	uint8_t
securityMinimum	–	uint8_t
deviceOverrideSecurityMinimum	–	bool_t

3.2.2.15 macCslEntry_t

Synopsis:

```
typedef struct macCslEntry_tag
{
    phyTime_t          nextSample;
    uint16_t           checksum;
    uint16_t           period;
    bool_t             inUse;
}macCslEntry_t;
```

Member	Description	Value
nextSample	Absolute time for the node channel sample.	phyTime_t
checksum	Checksum of the addressing information of the node.	uint16_t
period	Interval between node channel samples.	uint16_t
inUse	Entry is currently active.	bool_t

3.2.2.16 macRitIe_t

This type describes the LE RIT Information element.

Synopsis:

```
typedef PACKED_STRUCT macRitIe_tag
{
    uint8_t                    T0;
    uint8_t                    N;
    uint16_t                  T;
} macRitIe_t;
```

Member	Description	Value
T0	Time to first listen interval.	uint8_t
N	Number of repeat listen.	uint8_t
T	Repeat listen interval.	uint16_t

3.2.2.17 macRitEntry_t

Synopsis:

```
typedef struct macRitEntry_tag
{
    phyTime_t          lastTimestamp;
    macRitIe_t         ritIe;
    uint16_t           checksum;
    bool_t             inUse;
}macRitEntry_t;
```

Member	Description	Value
lastTimestamp	Absolute time for the last RIT Data Request of the node.	phyTime_t
ritIe	Listening schedule of the node, if any.	macRitIe_t
checksum	Checksum of the addressing information of the node.	uint16_t
inUse	Entry is currently active.	bool_t

3.2.2.18 macEBIEList_t

This type describes the list of Information Element IDs that can be included in an Enhanced Beacon.

Synopsis:

```
typedef macPayloadIeId_t macEBIEList_t[4];
```

3.2.2.19 macLinkOptions_t

Synopsis:

```
typedef struct macLinkOptions_tag
```

```
{
    uint8_t tx:1;
    uint8_t rx:1;
    uint8_t shared:1;
    uint8_t timekeeping:1;
    uint8_t reserved:4;
}macLinkOptions_t;
```

Member	Description	Value
tx	Tx Link	bitfield
rx	Rx Link	bitfield
shared	Shared Link (only for Tx)	bitfield
timekeeping	Timekeeping (only for Rx)	bitfield
reserved	Reserved for future use	bitfield

3.2.2.20 macSlotframeIe_t

Synopsis:

```
typedef PACKED_STRUCT macSlotframeIe_tag
{
    uint8_t  macSlotframeHandle;
    uint16_t macSlotframeSize;
}macSlotframeIe_t;
```

Member	Description	Value
macSlotframeHandle	Unique identifier of the slotframe	uint8_t
macSlotframeSize	Slotframe size in timeslots	uint16_t

3.2.2.21 macTschLinkIe_t

Synopsis:

```
typedef PACKED_STRUCT macTschLinkIe_tag
{
    uint16_t         timeslot;
    uint16_t         channelOffset;
    macLinkOptions_t macLinkOptions;
} macTschLinkIe_t;
```

Member	Description	Value
timeslot	Timeslot of the link inside the slotframe	uint16_t
channelOffset	Channel offset used for computing channel	uint16_t
macLinkOptions	Link options	macLinkOptions_t

3.2.2.22 macSlotframe_t

Synopsis:

```
typedef struct macSlotframe_tag
{
    macSlotframeIe_t macSlotframeIe;
    bool_t           inUse;
}macSlotframe_t;
```

Member	Description	Value
macSlotframeIe	Slotframe IE as included in EB	macSlotframeIe_t
inUse	Slotframe is currently active	bool_t

3.2.2.23 macLink_t

Synopsis:

```
typedef struct macLink_tag
{
    uint16_t         macLinkHandle;
    uint16_t         macNodeAddress;
    macLinkType_t    macLinkType;
    uint8_t          slotframeHandle;
    macTschLinkIe_t  macLinkIe;
    bool_t           inUse;
}macLink_t;
```

Member	Description	Value
macLinkHandle	Unique identifier of Link	uint16_t
macNodeAddress	Node address of the Link	uint16_t
macLinkType	Link type normal or advertising	macLinkType_t
slotframeHandle	Slotframe identifier for the Link	uint8_t
macLinkIe	Link IE as included in EB	macTschLinkIe_t
inUse	Link is currently active	bool_t

3.2.2.24 macTimeslotTemplate_t

Synopsis:

```
typedef PACKED_STRUCT macTimeslotTemplate_tag
{
    uint8_t  macTimeslotTemplateId;
    uint16_t macTsCCAOffset;
    uint16_t macTsCCA;
    uint16_t macTsTxOffset;
    uint16_t macTsRxOffset;
    uint16_t macTsRxAckDelay;
    uint16_t macTsTxAckDelay;
    uint16_t macTsRxWait;
    uint16_t macTsAckWait;
    uint16_t macTsRxTx;
    uint16_t macTsMaxAck;
    uint16_t macTsMaxTx;
```

```
    uint16_t macTsTimeslotLength;
}macTimeslotTemplate_t;
```

Member	Description	Value
macTimeslotTemplateId	Identifier of Timeslot Template	uint8_t
macTsCCAOffset	The time between the beginning of timeslot and start of CCA operation, in µs	uint16_t
macTsCCA	Duration of CCA, in µs	uint16_t
macTsTxOffset	The time between the beginning of the timeslot and the start of frame transmission, in µs	uint16_t
macTsRxOffset	Beginning of the timeslot to when the receiver shall be listening, in µs	uint16_t
macTsRxAckDelay	End of frame to when the transmitter shall listen for Acknowledgment, in µs	uint16_t
macTsTxAckDelay	End of frame to start of Acknowledgment, in µs	uint16_t
macTsRxWait	The time to wait for start of frame, in µs	uint16_t
macTsAckWait	The minimum time to wait for start of an Acknowledgment, in µs	uint16_t
macTsRxTx	Transmit to Receive turnaround, in µs	uint16_t
macTsMaxAck	Transmission time to send Acknowledgment, in µs	uint16_t
macTsMaxTx	Transmission time to send the maximum length frame, in µs	uint16_t
macTsTimeslotLength	The total length of the timeslot including any unused time after frame transmission and Acknowledgment, in µs	uint16_t

3.2.2.25 macHoppingSequenceIe_t

Synopsis:

```
typedef PACKED_STRUCT macHoppingSequenceIe_tag
{
    uint8_t  macHoppingSequenceId;
    uint8_t  macChannelPage;
    uint16_t macNumberOfChannels;
    uint32_t macPhyConfiguration;
    // macExtendedBitmap;
    //uint16_t macHoppingSequenceLength;
    // macHoppingSequenceList;
    //uint16_t macCurrentHop;
}macHoppingSequenceIe_t;
```

Member	Description	Value
macHoppingSequenceId	Unique identifier of Hopping Sequence	uint8_t
macChannelPage	Channel page in use	uint8_t
macNumberOfChannels	Maximum number of channels supported by current PHY mode	uint16_t
macPhyConfiguration	PHY configuration, channel list is in the channel bitmap	uint32_t

3.2.2.26 macTschSynchronizationIe_t

Synopsis:

```
typedef PACKED_STRUCT macTschSynchronizationIe_tag
{
    uint8_t macASN[5];
    uint8_t macJoinPriority;
}macTschSynchronizationIe_t;
```

Member	Description	Value
macASN	TSCH Absolute Sequence Number	uint8[]

macJoinPriority	The lowest join priority from the TSCH Synchronization IE in an Enhanced beacon, when the device joined the network + 1.	uint8_t

3.2.2.27 macTschSlotframeAndLinkIe_t

Synopsis:

```
typedef PACKED_STRUCT macTschSlotframeAndLinkIe_tag
{
    uint8_t numberOfSlotframes;
    //macTschSlotframeIe_t * numberOfSlotframes
}macTschSlotframeAndLinkIe_t;
```

Member	Description	Value
numberOfSlotframes	Number of slotframes included in the IE	uint8_t

3.2.2.28 macTschSlotframeIe_t

Synopsis:

```
typedef PACKED_STRUCT macTschSlotframeIe_tag
{
    macSlotframeIe_t slotframe;
    uint8_t          numberOfLinks;
    //macTschLinkIe_t * numberOfLinks
}macTschSlotframeIe_t;
```

Member	Description	Value
slotframe	Slotframe IE	macSlotframeIe_t
numberOfLinks	Number of links included in the IE	uint8_t

3.2.2.29 macAckTimeCorrectionIe_t

Synopsis:

```
typedef PACKED_STRUCT macAckTimeCorrectionIe_tag
{
    uint16_t timeSyncInfo;
}macAckTimeCorrectionIe_t;
```

Member	Description	Value
timeSyncInfo	TSCH Time synchronization as included in the ACK / NACK IE	uint16_t

3.2.2.30 macTschNeighbor_t

Synopsis:

```
typedef struct macTschNeighbor_tag
{
    uint16_t nodeAddr;
    uint16_t keepAlivePeriod;
    uint16_t cntTsSinceLastPacket;
}macTschNeighbor_t;
```

Member	Description	Value

nodeAddr	Node address of the clock source	uint16_t
keepAlivePeriod	Keep alive period set for the clock source	uint16_t
cntTsSinceLastPacket	Timeslot counter since last exchanged packet	uint16_t

3.3 SAP type definitions

Registration of the callback functions by the higher layer is done using the API described in the following sections.

3.3.1 MCPS SAP

The MAC data service is accessed through the MAC common part sublayer (MCPS) data SAP. The MCPS-SAP supports the transport of SSCS protocol data units (SPDUs) between peer SSCS entities.

These structure types, used with the MCPS SAP, are listed below.

3.3.1.1 mcpsDataReq_t

The MCPS-DATA.request primitive is generated by a local entity when a data SPDU (MSDU) is to be transferred to a peer SSCS entity.

Synopsis:

```
typedef STRUCT mcpsDataReq_tag
{
    uint16_t                srcPanId; /* Not in Spec */
    uint64_t                srcAddr; /* Not in Spec */
    addrModeType_t          srcAddrMode;
    addrModeType_t          dstAddrMode;
    uint16_t                dstPANId;
    uint64_t                dstAddr;
    uint16_t                msduLength;
    uint8_t                 msduHandle;
    macTxOptions_t          txOptions;
    macSecurityLevel_t      securityLevel;
    keyIdModeType_t         keyIdMode;
    uint64_t                keySource;
    uint8_t                 keyIndex;
    uint8_t                 *pMsdu;
} mcpsDataReq_t;
```

Direction: NWK → MAC

Member	Description	Value
srcPanId	PAN identifier of the entity from which the MSDU is being sent.	uint16_t
srcAddr	The individual device address of the entity from which the MSDU is being sent.	uint64_t
srcAddrMode	The source addressing mode for this primitive and subsequent MPDU. This field can take one of the following values:	addrModeType_t
	gAddrModeNoAddress_c	0x00
	gAddrModeShortAddress_c	0x02
	gAddrModeExtendedAddress_c	0x03
	gAddrModeReserved_c	0x01
dstAddrMode	The destination addressing mode for this primitive and subsequent MPDU. This field can take one of the following values:	addrModeType_t
	gAddrModeNoAddress_c	0x00

Member	Description	Value
	gAddrModeShortAddress_c	0x02
	gAddrModeExtendedAddress_c	0x03
	gAddrModeReserved_c	0x01
dstPANId	PAN identifier of the entity to which the MSDU is being transferred.	uint16_t
dstAddr	The individual device address of the entity to which the MSDU is being transferred.	uint64_t
msduLength	The number of octets contained in the MSDU to be transmitted by the MAC sublayer entity.	uint16_t
msduHandle	The handle associated with the MSDU to be transmitted by the MAC sublayer entity.	uint8_t
txOptions	Indicates the transmission options for this MSDU. Can take one or more of the following values:	macTxOptions_t
	gMacTxOptionsAck_c	0x01
	gMacTxOptionGts_c	0x02
	gMacTxOptionIndirect_c	0x04
securityLevel	Indicates the security level to be used. Can take one of the following values:	macSecurityLevel_t
	gMacSecurityNone_c	0x00
	gMacSecurityMic32_c	0x01
	gMacSecurityMic64_c	0x02
	gMacSecurityMic128_c	0x03
	gMacSecurityEnc_c	0x04
	gMacSecurityEncMic32_c	0x05
	gMacSecurityEncMic64_c	0x06
	gMacSecurityEncMic128_c	0x07
keyIdMode	This parameter is ignored if the **securityLevel** parameter is set to *gMacSecurityNone_c.* Can take one of the following values:	keyIdModeType_t
	gKeyIdMode0_c	0x00
	gKeyIdMode1_c	0x01
	gKeyIdMode2_c	0x02
	gKeyIdMode3_c	0x03
keySource	Indicates the originator of the key to be used. Ignored if the **keyIdMode** parameter is ignored or set to *gKeyIdMode0_c.*	uint64_t
keyIndex	Indicates the index of the key to be used. Ignored if the **keyIdMode** parameter is ignored or set to *gKeyIdMode0_c*	uint16_t
pMsdu	Pointer to a set of octets forming the MSDU to be transmitted by the MAC sublayer entity.	uint8_t*

3.3.1.2 mcpsDataCnf_t

The MCPS-DATA.confirm primitive reports the results of a request to transfer a data SPDU (MSDU) from a local SSCS entity to a single peer SSCS entity.

Synopsis:

```
typedef STRUCT mcpsDataCnf_tag
{
    uint8_t                 msduHandle;
    resultType_t            status;
    uint32_t                timestamp;
} mcpsDataCnf_t;
```

Direction: MAC → NWK

Member	Description	Value

Member	Description	Value
msduHandle	The handle associated with the MSDU being confirmed.	uint8_t
status	The status of the last MSDU transmission. Can take one of the following values:	resultType_t
	gSuccess_c	0x00
	gTransactionOverflow_c	0xf1
	gTransactionExpired_c	0xf0
	gChannelAccessFailure_c	0xe1
	gInvalidAddress_c	0xf5
	gInvalidGts_c	0xe6
	gNoAck_c	0xe9
	gCounterError_c	0xdb
	gFrameTooLong_c	0xe5
	gUnavailableKey_c	0xf3
	gUnsupportedSecurity_c	0xdf
	gInvalidParameter_c	0xe8
timestamp	Optional. The time (in symbols), at which the data were transmitted. This is a 24-bit value.	uint32_t

3.3.1.3 mcpsDataInd_t

The MCPS-DATA.indication primitive indicates the transfer of data SPDU (MSDU) from the MAC sublayer to the local SSCS entity.

Synopsis:

```
typedef STRUCT mcpsDataInd_tag
{
    uint64_t                dstAddr;
    uint16_t                dstPanId;
    addrModeType_t          dstAddrMode;
    uint64_t                srcAddr;
    uint16_t                srcPanId;
    addrModeType_t          srcAddrMode;
    uint16_t                msduLength;
    uint8_t                 mpduLinkQuality;
    uint8_t                 dsn;
    uint32_t                timestamp;
    macSecurityLevel_t      securityLevel;
    keyIdModeType_t         keyIdMode;
    uint64_t                keySource;
    uint8_t                 keyIndex;
    uint8_t                 *pMsdu;
} mcpsDataInd_t;
```

Direction: MAC → NWK

Member	Description	Value
srcAddrMode	The source addressing mode for this primitive and subsequent MPDU. This field can take one of the following values:	addrModeType_t
	gAddrModeNoAddress_c	0x00
	gAddrModeShortAddress_c	0x02
	gAddrModeExtendedAddress_c	0x03
	gAddrModeReserved_c	0x01
srcPANId	PAN identifier of the entity from which the MSDU was received.	uint16_t
srcAddr	The individual device address of the entity from which the MSDU was received.	uint64_t

Member	Description	Value
dstAddrMode	The destination addressing mode for this primitive and subsequent MPDU. This field can take one of the following values:	addrModeType_t
	gAddrModeNoAddress_c	0x00
	gAddrModeShortAddress_c	0x02
	gAddrModeExtendedAddress_c	0x03
	gAddrModeReserved_c	0x01
dstPANId	PAN identifier of the entity to which the MSDU is being transferred.	uint16_t
dstAddr	The individual device address of the entity to which the MSDU is being transferred.	uint64_t
msduLength	The number of octets contained in the MSDU being indicated by the MAC sublayer entity.	uint16_t
mpduLinkQuality	The LQI value measured during the reception of the MPDU. Lower values represent lower LQI	uint8_t
dsn	The DSN of the received data frame.	uint8_t
timestamp	Optional. The time, in symbols, at which the data were received. This is a 24-bit value.	uint32_t
securityLevel	Indicates the security level to be used.	macSecurityLevel_t
	gMacSecurityNone_c	0x00
	gMacSecurityMic32_c	0x01
	gMacSecurityMic64_c	0x02
	gMacSecurityMic128_c	0x03
	gMacSecurityEnc_c	0x04
	gMacSecurityEncMic32_c	0x05
	gMacSecurityEncMic64_c	0x06
	gMacSecurityEncMic128_c	0x07
keyIdMode	This parameter is ignored if the **securityLevel** parameter is set to *gMacSecurityNone_c.*	keyIdModeType_t
	gKeyIdMode0_c	0x00
	gKeyIdMode1_c	0x01
	gKeyIdMode2_c	0x02
	gKeyIdMode3_c	0x03
keySource	Indicates the originator of the key to be used. Ignored if the **keyIdMode** parameter is ignored or set to *gKeyIdMode0_c.*	uint64_t
keyIndex	Indicates the index of the key to be used. Ignored if the **keyIdMode** parameter is ignored or set to *gKeyIdMode0_c*	uint16_t
pMsdu	Pointer to a set of octets forming the MSDU to be transmitted by the MAC sublayer entity.	uint8_t*

3.3.1.4 mcpsPurgeReq_t

The MCPS-PURGE.request primitive allows the next higher layer to purge an MSDU from the transaction queue.

Synopsis:

```
typedef STRUCT mcpsPurgeReq_tag
{
    uint8_t                 msduHandle;
} mcpsPurgeReq_t;
```

Direction: NWK → MAC

Member	Description	Value
msduHandle	The handle of the MSDU requested to be purged from the transaction	uint8_t

	queue.	

3.3.1.5 mcpsPurgeCnf_t

The MCPS-PURGE.confirm primitive allows the MAC sublayer to notify the next higher layer of the success of its request to purge an MSDU from the transaction queue.

Synopsis:

```
typedef STRUCT mcpsPurgeCnf_tag
{
    uint8_t                 msduHandle;
    resultType_t          status;
} mcpsPurgeCnf_t;
```

Direction: MAC → NWK

Member	Description	Value
msduHandle	The handle of the MSDU requested to be purged from the transaction queue.	uint8_t
status	The status of the request to purge an MSDU from the transaction queue. Can take one of the following values:	resultType_t
	gSuccess_c	0x00
	gInvalidHandle_c	0xe7

3.3.1.6 mcpsPromInd_t

The MCPS-DATA.indication primitive indicates the transfer of a data SPDU (MSDU) from the MAC sublayer to the local SSCS entity.

Synopsis:

```
typedef STRUCT mcpsPromInd_tag
{
    uint8_t                 mpduLinkQuality;
    uint32_t                timeStamp;
    uint8_t                 msduLength;
    uint8_t                 *pMsdu;
} mcpsPromInd_t;
```

Direction: MAC → NWK

Member	Description	Value
mpduLinkQuality	The LQI value measured during reception of the MPDU. Lower values represent lower LQI.	uint8_t
timeStamp	The time, in symbols, at which the data were received. This is a 24-bit value.	uint32_t
msduLength	The number of octets contained in the MSDU being indicated by the MAC sublayer entity.	uint8_t
pMsdu	Pointer to a set of octets forming the MSDU being indicated by the MAC sublayer entity.	uint8_t[]

3.3.2 MLME SAP

The MAC sublayer conceptually includes a management entity called the MLME. This entity provides the service interfaces through which the layer management functions can be called.

The MLME is also responsible for maintaining a database of managed objects pertaining to the MAC sublayer. This database is referred to as the MAC sublayer PIB.

These structure types, used with the MLME SAP, are listed below.

3.3.2.1 mlmeAssociateReq_t

The MLME-ASSOCIATE.request primitive enables a device to request an association with a coordinator.

Synopsis:

```
typedef STRUCT mlmeAssociateReq_tag
{
    uint64_t                coordAddress;
    uint16_t                coordPanId;
    addrModeType_t          coordAddrMode;
    logicalChannelId_t      logicalChannel;
    macSecurityLevel_t      securityLevel;
    keyIdModeType_t         keyIdMode;
    uint64_t                keySource;
    uint8_t                 keyIndex;
    macCapabilityInfo_t     capabilityInfo;
    channelPageId_t         channelPage;
} mlmeAssociateReq_t;
```

Direction: NWK → MAC

Member	Description	Value
coordAddress	The address of the coordinator with which to associate.	uint64_t
coordPANId	The identifier of the PAN with which to associate.	uint16_t
coordAddrMode	The coordinator addressing mode for this primitive and subsequent MPDU. This field can take one of the following values:	addrModeType_t
	gAddrModeShortAddress_c	0x02
	gAddrModeExtendedAddress_c	0x03
logicalChannel	Indicates the logical channel on which to attempt association. Can take one of the following values:	logicalChannelId_t
	gLogicalChannel0_c	0
	…	
	gLogicalChannel11_c	11
	gLogicalChannel12_c	12
	…	…
	gLogicalChannel26_c	26
	…	
	gLogicalChannel128_c	128
securityLevel	Indicates the security level to be used. Can take one of the following values:	macSecurityLevel_t
	gMacSecurityNone_c	0x00
	gMacSecurityMic32_c	0x01
	gMacSecurityMic64_c	0x02

Member	Description	Value
	gMacSecurityMic128_c	0x03
	gMacSecurityEnc_c	0x04
	gMacSecurityEncMic32_c	0x05
	gMacSecurityEncMic64_c	0x06
	gMacSecurityEncMic128_c	0x07
keyIdMode	This parameter is ignored if the **securityLevel** parameter is set to *gMacSecurityNone_c*. Can take one of the following values:	keyIdModeType_t
	gKeyIdMode0_c	0x00
	gKeyIdMode1_c	0x01
	gKeyIdMode2_c	0x02
	gKeyIdMode3_c	0x03
keySource	Indicates the originator of the key to be used. Ignored if the **keyIdMode** parameter is ignored or set to *gKeyIdMode0_c*	uint64_t
keyIndex	Indicates the index of the key to be used. Ignored if the **keyIdMode** parameter is ignored or set to *gKeyIdMode0_c.*	uint16_t
capabilityInformation	The operational capabilities of the device requesting association. Can take one or more of the following values:	macCapabilityInfo_t
	gCapInfoAltPanCoord_c	0x01
	gCapInfoDeviceFfd_c	0x02
	gCapInfoPowerMains_c	0x04
	gCapInfoRxWhenIdle_c	0x08
	gCapInfoSecurity_c	0x40
	gCapInfoAllocAddr_c	0x80
channelPage	The channel page on which to attempt association. Can take one of the following values:	channelPageId_t
	gChannelPageId0_c	0
	gChannelPageId1_c	1
	gChannelPageId2_c	2
	gChannelPageId9_c	9

3.3.2.2 mlmeAssociateCnf_t

The MLME-ASSOCIATE.confirm primitive is used to inform the next higher layer of the initiating device whether its request to associate was successful or unsuccessful.

Synopsis:

```
typedef STRUCT mlmeAssociateCnf_tag
{
    uint16_t                assocShortAddress;
    resultType_t            status;
    macSecurityLevel_t      securityLevel;
    keyIdModeType_t         keyIdMode;
    uint64_t                keySource;
    uint8_t                 keyIndex;
} mlmeAssociateCnf_t;
```

Direction: MAC → NWK

Member	Description	Value
assocShortAddress	The short device address allocated by the coordinator on successful association. This parameter will be equal to **0xffff** if the association attempt was unsuccessful.	uint16_t
status	The status of the association attempt. Can take one of the	resultType_t

Member	Description	Value
	following values:	
	gSuccess_c	0x00
	gChannelAccessFailure_c	0xe1
	gNoAck_c	0xe9
	gNoData_c	0xe9
	gCounterError_c	0xdb
	gFrameTooLong_c	0xe5
	gImproperKeyType_c	0xdc
	gImproperSecurityLevel_c	0xdd
	gSecurityError_c	0xe4
	gUnsupportedLegacy_c	0xde
	gUnavailableKey_c	0xf3
	gUnsupportedSecurity_c	0xdf
	gInvalidParameter_c	0xe8
securityLevel	Indicates the security level to be used. Can take one of the following values:	macSecurityLevel_t
	gMacSecurityNone_c	0x00
	gMacSecurityMic32_c	0x01
	gMacSecurityMic64_c	0x02
	gMacSecurityMic128_c	0x03
	gMacSecurityEnc_c	0x04
	gMacSecurityEncMic32_c	0x05
	gMacSecurityEncMic64_c	0x06
	gMacSecurityEncMic128_c	0x07
keyIdMode	This parameter is ignored if the **securityLevel** parameter is set to *gMacSecurityNone_c.* Can take one of the following values:	keyIdModeType_t
	gKeyIdMode0_c	0x00
	gKeyIdMode1_c	0x01
	gKeyIdMode2_c	0x02
	gKeyIdMode3_c	0x03
keySource	Indicates the originator of the key to be used. Ignored if the **keyIdMode** parameter is ignored or set to *gKeyIdMode0_c.*	uint64_t
keyIndex	Indicates the index of the key to be used. Ignored if the **keyIdMode** parameter is ignored or set to *gKeyIdMode0_c.*	uint16_t

3.3.2.3 mlmeAssociateRes_t

The MLME-ASSOCIATE.response primitive is used to initiate a response to an MLME-ASSOCIATE.indication primitive.

Synopsis:

```
typedef STRUCT mlmeAssociateRes_tag
{
    uint16_t                assocShortAddress;
    macSecurityLevel_t      securityLevel;
    keyIdModeType_t         keyIdMode;
    uint64_t                keySource;
    uint8_t                 keyIndex;
    resultType_t            status;
} mlmeAssociateRes_t;
```

Direction: NWK → MAC

Member	Description	Value
deviceAddress	The address of the device requesting association.	uint64_t
assocShortAddress	Contains the short device address allocated by the coordinator on successful association. This parameter is set to 0xffff if the association was unsuccessful.	uint16_t
securityLevel	Indicates the security level to be used. Can take one of the following values:	macSecurityLevel_t
	gMacSecurityNone_c	0x00
	gMacSecurityMic32_c	0x01
	gMacSecurityMic64_c	0x02
	gMacSecurityMic128_c	0x03
	gMacSecurityEnc_c	0x04
	gMacSecurityEncMic32_c	0x05
	gMacSecurityEncMic64_c	0x06
	gMacSecurityEncMic128_c	0x07
keyIdMode	This parameter is ignored if the **securityLevel** parameter is set to *gMacSecurityNone_c.* Can take one of the following values:	keyIdModeType_t
	gKeyIdMode0_c	0x00
	gKeyIdMode1_c	0x01
	gKeyIdMode2_c	0x02
	gKeyIdMode3_c	0x03
keySource	Indicates the originator of the key to be used. Ignored if the **keyIdMode** parameter is ignored or set to *gKeyIdMode0_c.*	uint64_t
keyIndex	Indicates the index of the key to be used. Ignored if the **keyIdMode** parameter is ignored or set to *gKeyIdMode0_c.*	uint16_t
status	The status of the association attempt. Can take one of the following values:	resultType_t
	gAssociationSuccessful_c	0x00
	gPanAtCapacity_c	0x01
	gPanAccessDenied_c	0x02

3.3.2.4 mlmeAssociateInd_t

The MLME-ASSOCIATE.indication primitive is used to indicate the reception of an association request command.

Synopsis:

```
typedef STRUCT mlmeAssociateInd_tag
{
    uint64_t                deviceAddress;
    macSecurityLevel_t      securityLevel;
    keyIdModeType_t         keyIdMode;
    uint64_t                keySource;
    uint8_t                 keyIndex;
    macCapabilityInfo_t     capabilityInfo;
} mlmeAssociateInd_t;
```

Direction: MAC → NWK

Member	Description	Value
deviceAddress	The address of the device requesting association.	uint64_t
securityLevel	Indicates the security level to be used. Can take one of the following values:	macSecurityLevel_t
	gMacSecurityNone_c	0x00
	gMacSecurityMic32_c	0x01
	gMacSecurityMic64_c	0x02
	gMacSecurityMic128_c	0x03

Member	Description	Value
	gMacSecurityEnc_c	0x04
	gMacSecurityEncMic32_c	0x05
	gMacSecurityEncMic64_c	0x06
	gMacSecurityEncMic128_c	0x07
keyIdMode	This parameter is ignored if the **securityLevel** parameter is set to *gMacSecurityNone_c.* Can take one of the following values:	keyIdModeType_t
	gKeyIdMode0_c	0x00
	gKeyIdMode1_c	0x01
	gKeyIdMode2_c	0x02
	gKeyIdMode3_c	0x03
keySource	Indicates the originator of the key to be used. Ignored if the **keyIdMode** parameter is ignored or set to *gKeyIdMode0_c.*	uint64_t
keyIndex	Indicates the index of the key to be used. Ignored if the **keyIdMode** parameter is ignored or set to *gKeyIdMode0_c.*	uint16_t
capabilityInformation	The operational capabilities of the device requesting association. Can take one or more of the following values:	macCapabilityInfo_t
	gCapInfoAltPanCoord_c	0x01
	gCapInfoDeviceFfd_c	0x02
	gCapInfoPowerMains_c	0x04
	gCapInfoRxWhenIdle_c	0x08
	gCapInfoSecurity_c	0x40
	gCapInfoAllocAddr_c	0x80

3.3.2.5 mlmeDisassociateReq_t

The MLME-DISASSOCIATE.request primitive is used by an associated device to notify the coordinator of its intent to leave the PAN. It is also used by the coordinator to instruct an associated device to leave the PAN.

Synopsis:

```
void MlmeDisassociateReq (mlmeDisassociateReq_t* pParam);

typedef STRUCT mlmeDisassociateReq_tag
{
    addrModeType_t          deviceAddrMode;
    uint16_t                devicePANId;
    uint64_t                deviceAddress;
    macDisassociateReason_t disassociateReason;
    bool_t                  txIndirect;
    macSecurityLevel_t      securityLevel;
    keyIdModeType_t         keyIdMode;
    uint64_t                keySource;
    uint8_t                 keyIndex;
} mlmeDisassociateReq_t;
```

Direction: NWK → MAC

Member	Description	Value
deviceAddrMode	The addressing mode of the device to which to send the disassociation notification command. This field can take one of the following values:	addrModeType_t
	gAddrModeShortAddress_c	0x02

Member	Description	Value
	gAddrModeExtendedAddress_c	0x03
devicePANId	The PAN identifier of the device to which to send the disassociation notification command.	uint16_t
deviceAddress	The address of the device to which to send the disassociation notification command.	uint64_t
disassociateReason	Describes the reason for the disassociation. This field can take one of the following values:	macDisassociateReason_t
	gDisassociateCoord_t	0x01
	gDisassociateDevice_t	0x02
txIndirect	Indicates whether the disassociation notification shall be sent indirectly.	bool_t
securityLevel	Indicates the security level to be used. Can take one of the following values:	macSecurityLevel_t
	gMacSecurityNone_c	0x00
	gMacSecurityMic32_c	0x01
	gMacSecurityMic64_c	0x02
	gMacSecurityMic128_c	0x03
	gMacSecurityEnc_c	0x04
	gMacSecurityEncMic32_c	0x05
	gMacSecurityEncMic64_c	0x06
	gMacSecurityEncMic128_c	0x07
keyIdMode	This parameter is ignored if the **securityLevel** parameter is set to *gMacSecurityNone_c.* Can take one of the following values:	keyIdModeType_t
	gKeyIdMode0_c	0x00
	gKeyIdMode1_c	0x01
	gKeyIdMode2_c	0x02
	gKeyIdMode3_c	0x03
keySource	Indicates the originator of the key to be used. Ignored if the **keyIdMode** parameter is ignored or set to *gKeyIdMode0_c.*	uint64_t
keyIndex	Indicates the index of the key to be used. Ignored if the **keyIdMode** parameter is ignored or set to *gKeyIdMode0_c.*	uint16_t

3.3.2.6 mlmeDisassociateCnf_t

The MLME-DISASSOCIATE.confirm primitive reports the results of an MLME-DISASSOCIATE.request primitive.

Synopsis:

```
typedef STRUCT mlmeDisassociateCnf_tag
{
    uint64_t                deviceAddress;
    uint16_t                devicePanId;
    addrModeType_t          deviceAddrMode;
    resultType_t            status;
} mlmeDisassociateCnf_t;
```

Direction: MAC → NWK

Member	Description	Value
deviceAddress	The address of the device that has either requested disassociation or has been instructed to disassociate by its coordinator.	uint64_t
devicePANId	The PAN identifier of the device that has either requested	uint16_t

Member	Description	Value
	disassociation or has been instructed to disassociate by its coordinator.	
deviceAddrMode	The addressing mode of the device that has either requested disassociation or has been instructed to disassociate by its coordinator. This field can take one of the following values:	addrModeType_t
	gAddrModeShortAddress_c	0x02
	gAddrModeExtendedAddress_c	0x03
status	The status of the disassociation attempt. Can take one of the following values:	resultType_t
	gSuccess_c	0x00
	gTransactionOverflow_c	0xf1
	gTransactionExpired_c	0xf0
	gNoAck_c	0xe9
	gChannelAccessFailure_c	0xe1
	gCounterError_c	0xdb
	gFrameTooLong_c	0xe5
	gUnavailableKey_c	0xf3
	gUnsupportedSecurity_c	0xdf
	gInvalidParameter_c	0xe8

3.3.2.7 mlmeDisassociateInd_t

The MLME-DISASSOCIATE.indication primitive is used to indicate the reception of a disassociation notification command.

Synopsis:

```
typedef STRUCT mlmeDisassociateInd_tag
{
    uint64_t                deviceAddress;
    macDisassociateReason_t disassociateReason;
    macSecurityLevel_t      securityLevel;
    keyIdModeType_t         keyIdMode;
    uint64_t                keySource;
    uint8_t                 keyIndex;
} mlmeDisassociateInd_t;
```

Direction: MAC → NWK

Member	Description	Value
deviceAddress	The address of the device requesting disassociation.	uint64_t
disassociateReason	Describes the reason for the disassociation. This field can take one of the following values:	macDisassociateReason_t
	gDisassociateCoord_t	0x01
	gDisassociateDevice_t	0x02
securityLevel	Indicates the security level to be used. Can take one of the following values:	macSecurityLevel_t
	gMacSecurityNone_c	0x00
	gMacSecurityMic32_c	0x01
	gMacSecurityMic64_c	0x02
	gMacSecurityMic128_c	0x03
	gMacSecurityEnc_c	0x04
	gMacSecurityEncMic32_c	0x05
	gMacSecurityEncMic64_c	0x06
	gMacSecurityEncMic128_c	0x07
keyIdMode	This parameter is ignored if the **securityLevel** parameter is set to *gMacSecurityNone_c.* Can take	keyIdModeType_t

Member	Description	Value
	one of the following values:	
	gKeyIdMode0_c	0x00
	gKeyIdMode1_c	0x01
	gKeyIdMode2_c	0x02
	gKeyIdMode3_c	0x03
keySource	Indicates the originator of the key to be used. Ignored if the **keyIdMode** parameter is ignored or set to *gKeyIdMode0_c.*	uint64_t
keyIndex	Indicates the index of the key to be used. Ignored if the **keyIdMode** parameter is ignored or set to *gKeyIdMode0_c.*	uint16_t

3.3.2.8 mlmeBeaconNotifyInd_t

The MLME-BEACON-NOTIFY.indication primitive is used to send parameters contained within a beacon frame received by the MAC sublayer to the next higher layer.
The primitive also sends a measure of the LQI and the time the beacon frame was received.

Synopsis:

```
typedef STRUCT mlmeBeaconNotifyInd_tag
{
    uint8_t                 bsn;
    uint8_t                 pendAddrSpec;
    uint8_t                 sduLength;
    uint8_t*                pAddrList;
    panDescriptor_t*        pPanDescriptor;
    uint8_t*                pSdu;
    void*                   pBufferRoot
} mlmeBeaconNotifyInd_t;
```

Direction: MAC → NWK

Member	Description	Value
bsn	The beacon sequence number.	uint8_t
pendAddrSpec	The beacon pending address specification. Bits 0-2 indicate the number of short addresses contained in the Address List field of the beacon frame. Bits 4-6 indicate the number of extended addresses contained in the Address List field of the beacon frame.	uint8_t
sduLength	The number of octets contained in the beacon payload of the beacon frame received by the MAC sublayer.	uint8_t
pAddrList	The list of addresses of the devices for which the beacon source has data.	uint64_t*
panDescriptor	The PANDescriptor for the received beacon.	panDescriptor_t
pSdu	The set of octets comprising the beacon payload to be transferred from the MAC sublayer entity to the next higher layer.	uint8_t*
pBufferRoot	Pointer to the start of the message received by air. The upper layer must free this buffer before freeing the indication message.	void*

3.3.2.9 mlmeGetReq_t

The MLME-GET.request primitive requests information about a given PIB attribute.

Synopsis:

```
typedef STRUCT mlmeGetReq_tag
{
```

```
    pibId_t                 pibAttribute;
    uint8_t                 pibAttributeIndex;
    void*                   pibAttributeValue;
} mlmeGetReq_t;
```

Direction: NWK → MAC

Member	Description	Value
pibAttribute	The identifier of the PIB attribute to read.	pibId_t
pibAttributeIndex	The index within the table of the specified PIB attribute to read. This parameter is valid only for the MAC PIB attributes that are tables; it is ignored when accessing PHY PIB attributes.	uint8_t
pibAttributeValue	The value of the indicated PIB attribute that was read. This parameter has zero length when the status parameter is set to UNSUPPORTED_ATTRIBUTE. This field is not IEEE Std. 802.15.4 specific.	void*

NOTE

The MLME-GET.Request is a synchronous request. The status of the request is returned by the NWK_MLME_SapHandler() function. The MAC will not free the dynamic allocated memory of the request, and it will not send MLME-GET.Confirm!

3.3.2.10 mlmeGtsReq_t

The MLME-GTS.request primitive enables a device to send a request to the PAN coordinator to allocate a new GTS or to deallocate an existing GTS. This primitive is also used by the PAN coordinator to initiate a GTS deallocation.

Synopsis:

```
typedef STRUCT mlmeGtsReq_tag
{
    gtsCharacteristics_t    gtsCharacteristics;
    macSecurityLevel_t      securityLevel;
    keyIdModeType_t         keyIdMode;
    uint64_t                keySource;
    uint8_t                 keyIndex;
} mlmeGtsReq_t;
```

Direction: NWK → MAC

Member	Description	Value
gtsCharacteristics	The characteristics of the GTS request, including whether the request is for the allocation of a new GTS or the deallocation of an existing GTS.	gtsCharacteristics_t
securityLevel	Indicates the security level to be used. Can take one of the following values:	macSecurityLevel_t
	gMacSecurityNone_c	0x00
	gMacSecurityMic32_c	0x01

Member	Description	Value
	gMacSecurityMic64_c	0x02
	gMacSecurityMic128_c	0x03
	gMacSecurityEnc_c	0x04
	gMacSecurityEncMic32_c	0x05
	gMacSecurityEncMic64_c	0x06
	gMacSecurityEncMic128_c	0x07
keyIdMode	This parameter is ignored if the **securityLevel** parameter is set to *gMacSecurityNone_c.* Can take one of the following values:	keyIdModeType_t
	gKeyIdMode0_c	0x00
	gKeyIdMode1_c	0x01
	gKeyIdMode2_c	0x02
	gKeyIdMode3_c	0x03
keySource	Indicates the originator of the key to be used. Ignored if the **keyIdMode** parameter is ignored or set to *gKeyIdMode0_c.*	uint64_t
keyIndex	Indicates the index of the key to be used. Ignored if the **keyIdMode** parameter is ignored or set to *gKeyIdMode0_c.*	uint16_t

3.3.2.11 mlmeGtsCnf_t

The MLME-GTS.confirm primitive reports the results of a request to allocate a new GTS or deallocate an existing GTS.

Synopsis:

```
typedef STRUCT mlmeGtsCnf_tag
{
    resultType_t            status;
    gtsCharacteristics_t    gtsCharacteristics;
} mlmeGtsCnf_t;
```

Direction: MAC → NWK

Member	Description	Value
gtsCharacteristics	The characteristics of the GTS.	gtsCharacteristics_t
status	The status of the GTS request. Can take one of the following values:	resultType_t
	gSuccess_c	0x00
	gDenied_c	0xe2
	gNoShortAddress_c	0xec
	gChannelAccessFailure_c	0xe1
	gNoAck_c	0xe9
	gNoData_c	0xeb
	gCounterError_c	0xdb
	gFrameTooLong_c	0xe5
	gUnavailableKey_c	0xf3
	gUnsupportedSecurity_c	0xdf
	gInvalidParameter_c	0xe8

3.3.2.12 mlmeGtsInd_t

The MLME-GTS.indication primitive indicates that a GTS has been allocated or that a previously allocated GTS has been deallocated.

Synopsis:

```
typedef STRUCT mlmeGtsInd_tag
{
    uint16_t                deviceAddress;
    gtsCharacteristics_t    gtsCharacteristics;
    macSecurityLevel_t      securityLevel;
    keyIdModeType_t         keyIdMode;
    uint64_t                keySource;
    uint8_t                 keyIndex;
} mlmeGtsInd_t;
```

Direction: MAC → NWK

Member	Description	Value
deviceAddress	The 16-bit short address of the device that has been allocated or deallocated a GTS.	uint16_t
gtsCharacteristics	The characteristics of the GTS.	gtsCharacteristics_t
securityLevel	Indicates the security level to be used. Can take one of the following values:	macSecurityLevel_t
	gMacSecurityNone_c	0x00
	gMacSecurityMic32_c	0x01
	gMacSecurityMic64_c	0x02
	gMacSecurityMic128_c	0x03
	gMacSecurityEnc_c	0x04
	gMacSecurityEncMic32_c	0x05
	gMacSecurityEncMic64_c	0x06
	gMacSecurityEncMic128_c	0x07
keyIdMode	This parameter is ignored if the **securityLevel** parameter is set to *gMacSecurityNone_c.* Can take one of the following values:	keyIdModeType_t
	gKeyIdMode0_c	0x00
	gKeyIdMode1_c	0x01
	gKeyIdMode2_c	0x02
	gKeyIdMode3_c	0x03
keySource	Indicates the originator of the key to be used. Ignored if the **keyIdMode** parameter is ignored or set to *gKeyIdMode0_c.*	uint64_t
keyIndex	Indicates the index of the key to be used. Ignored if the **keyIdMode** parameter is ignored or set to *gKeyIdMode0_c.*	uint16_t

3.3.2.13 **mlmeOrphanInd_t**

The MLME-ORPHAN.indication primitive allows the MLME of a coordinator to notify the next higher layer of the presence of an orphaned device.

Synopsis:

```
typedef STRUCT mlmeOrphanInd_tag
{
    uint64_t                orphanAddress;
    macSecurityLevel_t      securityLevel;
    keyIdModeType_t         keyIdMode;
    uint64_t                keySource;
```

```
    uint8_t                 keyIndex;
} mlmeOrphanInd_t;
```

Direction: MAC → NWK

Member	Description	Value
orphanAddress	The extended address of the orphaned device.	uint64_t
securityLevel	Indicates the security level to be used. Can take one of the following values:	macSecurityLevel_t
	gMacSecurityNone_c	0x00
	gMacSecurityMic32_c	0x01
	gMacSecurityMic64_c	0x02
	gMacSecurityMic128_c	0x03
	gMacSecurityEnc_c	0x04
	gMacSecurityEncMic32_c	0x05
	gMacSecurityEncMic64_c	0x06
	gMacSecurityEncMic128_c	0x07
keyIdMode	This parameter is ignored if the **securityLevel** parameter is set to *gMacSecurityNone_c.* Can take one of the following values:	keyIdModeType_t
	gKeyIdMode0_c	0x00
	gKeyIdMode1_c	0x01
	gKeyIdMode2_c	0x02
	gKeyIdMode3_c	0x03
keySource	Indicates the originator of the key to be used. Ignored if the **keyIdMode** parameter is ignored or set to *gKeyIdMode0_c.*	uint64_t
keyIndex	Indicates the index of the key to be used. Ignored if the **keyIdMode** parameter is ignored or set to *gKeyIdMode0_c.*	uint16_t

3.3.2.14 mlmeOrphanRes_t

The MLME-ORPHAN.response primitive is generated by the next higher layer and issued to its MLME when it reaches a decision about whether the orphaned device indicated in the MLME-ORPHAN.indication primitive is associated.

Synopsis:

```
typedef STRUCT mlmeOrphanRes_tag
{
    uint64_t                orphanAddress;
    uint16_t                shortAddress;
    bool_t                  associatedMember;
    macSecurityLevel_t      securityLevel;
    keyIdModeType_t         keyIdMode;
    uint64_t                keySource;
    uint8_t                 keyIndex;
} mlmeOrphanRes_t;
```

Direction: NWK → MAC

Member	Description	Value
orphanAddress	The extended address of the orphaned device.	uint64_t
shortAddress	The 16-bit short address allocated to the orphaned device if it is associated with this coordinator.	uint16_t
associatedMember	TRUE if the orphaned device is associated with this coordinator or FALSE otherwise.	bool_t

Member	Description	Value
securityLevel	Indicates the security level to be used. Can take one of the following values:	macSecurityLevel_t
	gMacSecurityNone_c	0x00
	gMacSecurityMic32_c	0x01
	gMacSecurityMic64_c	0x02
	gMacSecurityMic128_c	0x03
	gMacSecurityEnc_c	0x04
	gMacSecurityEncMic32_c	0x05
	gMacSecurityEncMic64_c	0x06
	gMacSecurityEncMic128_c	0x07
keyIdMode	This parameter is ignored if the **securityLevel** parameter is set to *gMacSecurityNone_c.* Can take one of the following values:	keyIdModeType_t
	gKeyIdMode0_c	0x00
	gKeyIdMode1_c	0x01
	gKeyIdMode2_c	0x02
	gKeyIdMode3_c	0x03
keySource	Indicates the originator of the key to be used. Ignored if the **keyIdMode** parameter is ignored or set to *gKeyIdMode0_c.*	uint64_t
keyIndex	Indicates the index of the key to be used. Ignored if the **keyIdMode** parameter is ignored or set to *gKeyIdMode0_c.*	uint16_t

3.3.2.15 mlmeResetReq_t

The MLME-RESET.request primitive is generated by the next higher layer and issued to the MLME to request a reset of the MAC sublayer to its initial conditions. The MLME-RESET.request primitive is issued before using the MLME-START.request or the MLME-ASSOCIATE.request primitives.

Synopsis:

```
typedef STRUCT mlmeResetReq_tag
{
    bool_t                  setDefaultPIB;
} mlmeResetReq_t;
```

Direction: MAC → NWK

Member	Description	Value
setDefaultPIB	If TRUE, and all MAC PIB attributes are set to their default values.	bool_t

NOTE

The MLME-RESET.Request is a synchronous request. The status of the request is returned by the NWK_MLME_SapHandler() function. The MAC will not free the dynamic allocated memory of the request, and it will not send a MLME-RESET.Confirm!

3.3.2.16 mlmeRxEnableReq_t

The MLME-RX-ENABLE.request primitive enables the next higher layer to request that the receiver is either enabled for a finite period of time or disabled.

Synopsis:

```
typedef STRUCT mlmeRxEnableReq_tag
{
    bool_t                  deferPermit;
    uint32_t                rxOnTime;
    uint32_t                rxOnDuration;
} mlmeRxEnableReq_t;
```

Direction: NWK → MAC

Member	Description	Value
deferPermit	TRUE if the requested operation can be deferred until the next superframe if the requested time has already passed. FALSE if the requested operation is only to be attempted in the current superframe. This parameter is ignored for nonbeacon-enabled PANs.	bool_t
rxOnTime	Contains the number of symbols measured from the start of the superframe before the receiver is to be enabled or disabled. This is a 24-bit value.	uint32_t
rxOnDuration	Contains the number of symbols for which the receiver is to be enabled. If this parameter is equal to 0x000000, the receiver is disabled. This is a 24-bit value.	uint32_t

3.3.2.17 mlmeRxEnableCnf_t

The MLME-RX-ENABLE.confirm primitive reports the results of the attempt to enable or disable the receiver.

Synopsis:

```
typedef STRUCT mlmeRxEnableCnf_tag
{
    resultType_t            status;
} mlmeRxEnableCnf_t;
```

Direction: MAC → NWK

Member	Description	Value
status	The result of the request to enable or disable the receiver. Can contain one of the following values:	resultType_t
	gSuccess_c	0x00
	gPastTime_c	0xf7
	gOnTimeTooLong_c	0xf6
	gInvalidParameter_c	0xe8

3.3.2.18 mlmeScanReq_t

The MLME-SAP scan primitives define how a device can determine the energy usage or the presence or absence of PANs in a communications channel.

The MLME-SCAN.request primitive is used to initiate a channel scan over a given list of channels. A device can use the channel scan to measure the energy on the channel, search

for the coordinator with which it associated, or search for all coordinators transmitting beacon frames within the POS of the scanning device.

Synopsis:

```
typedef STRUCT mlmeScanReq_tag
{
    macScanType_t           scanType;
    channelMask_t           scanChannels;
    uint8_t                 scanDuration;
    channelPageId_t         channelPage;
    macSecurityLevel_t      securityLevel;
    keyIdModeType_t         keyIdMode;
    uint64_t                keySource;
    uint8_t                 keyIndex;
} mlmeScanReq_t;
```

Direction: NWK → MAC

Member	Description	Value
scanType	Indicates the type of scan performed. Can take one of the following values:	macScanType_t
	gScanModeED_c	0x00
	gScanModeActive_c	0x01
	gScanModePassive_c	0x02
	gScanModeOrphan_c	0x03
scanChannels	Indicates which channels are to be scanned (1 = scan, 0 = do not scan) for each of the 27 channels supported by the ChannelPage parameter or up to 38 channels, depending on the Sub-1 GHz SUN PHY Mode. Can take one or more of the following values:	channelMask_t
	gChannelMask00_c	0x00000001
	gChannelMask01_c	0x00000002
	…	…
	gChannelMask26_c	0x04000000
	…	
	gChannelMask128_c	
scanDuration	This field contains a value used to calculate the length of time to spend scanning each channel for ED, active, and passive scans. This parameter is ignored for orphan scans. Valid range is 0-14.	uint8_t
channelPage	The channel page on which to perform the scan. Can take one of the following values:	channelPageId_t
	gChannelPageId0_c	0x00
	gChannelPageId1_c	0x01
	gChannelPageId2_c	0x02
	gChannelPageId9_c	0x09
securityLevel	Indicates the security level to be used. Can take one of the following values:	macSecurityLevel_t
	gMacSecurityNone_c	0x00
	gMacSecurityMic32_c	0x01
	gMacSecurityMic64_c	0x02
	gMacSecurityMic128_c	0x03
	gMacSecurityEnc_c	0x04
	gMacSecurityEncMic32_c	0x05

Member	Description	Value
	gMacSecurityEncMic64_c	0x06
	gMacSecurityEncMic128_c	0x07
keyIdMode	This parameter is ignored if the **securityLevel** parameter is set to *gMacSecurityNone_c.* Can take one of the following values:	keyIdModeType_t
	gKeyIdMode0_c	0x00
	gKeyIdMode1_c	0x01
	gKeyIdMode2_c	0x02
	gKeyIdMode3_c	0x03
keySource	Indicates the originator of the key to be used. Ignored if the **keyIdMode** parameter is ignored or set to *gKeyIdMode0_c*	uint64_t
keyIndex	Indicates the index of the key to be used. Ignored if the **keyIdMode** parameter is ignored or set to *gKeyIdMode0_c*	uint16_t

3.3.2.19 **mlmeScanCnf_t**

The MLME-SCAN.confirm primitive reports the result of the channel scan request. The panDescriptor_t type provides a link pointer for a chained list.

Synopsis:

```
typedef STRUCT mlmeScanCnf_tag
{
    resultType_t            status;
    macScanType_t           scanType;
    channelPageId_t         channelPage;
    uint8_t                 resultListSize;
    channelMask_t           unscannedChannels;
    UNION {
      uint8_t*              pEnergyDetectList;
      panDescriptorBlock_t* pPanDescriptorBlockList;
    } resList;
} mlmeScanCnf_t;
```

Direction: MAC → NWK

Member	Description	Value
status	The status of the scan attempt.	resultType_t
	gSuccess_c	0x00
	gLimitReached_c	0xfa
	gNoBeacon_c	0xea
	gScanInProgress_c	0xfc
	gCounterError_c	0xdb
	gFrameTooLong_c	0xe5
	gUnavailableKey_c	0xf3
	gUnsupportedSecurity_c	0xdf
	gInvalidParameter_c	0xe8
scanType	Indicates the type of scan performed. Can take one of the following values:	macScanType_t
	gScanModeED_c	0x00
	gScanModeActive_c	0x01
	gScanModePassive_c	0x02
	gScanModeOrphan_c	0x03
channelPage	The channel page on which the scan was performed. Can take one of the following values:	channelPageId_t
	gChannelPageId0_c	0

Member	Description	Value
	gChannelPageId1_c	1
	gChannelPageId2_c	2
	gChannelPageId9_c	9
unscannedChannels	Indicates which channels given in the request were not scanned (1 = not scanned, 0 = scanned or not requested). This parameter is not valid for ED scans. Can take one or more of the following values:	logicalChannelId_t
	gChannelMask00_c	0x00000001
	gChannelMask01_c	0x00000002
	…	…
	gChannelMask26_c	0x04000000
	…	
	gChannelMask37_c	0x0000002000000000
resultListSize	The number of elements returned in the appropriate result lists. This value is 0 for the result of an orphan scan.	uint8_t
energyDetectList	The list of energy measurements, one for each channel searched during an ED scan. This parameter is null for active, passive, and orphan scans.	uint8_t*
panDescriptorList	Pointer to array of PAN descriptors, one for each beacon found during an active or passive scan, if macAutoRequest is set to TRUE. This parameter is null for ED and orphan scans or when macAutoRequest is set to FALSE during an active or passive scan.	panDescriptor_t*

3.3.2.20 **mlmeCommStatusInd_t**

The MLME-SAP communication status primitive defines how the MLME communicates to the next higher layer about transmission status, when the transmission was instigated by a response primitive, and about security errors on incoming packets.

The MLME-COMM-STATUS.indication primitive allows the MLME to indicate a communications status.

Synopsis:

```
typedef STRUCT mlmeCommStatusInd_tag
{
    uint16_t                panId;
    addrModeType_t          srcAddrMode;
    uint64_t                srcAddr;
    addrModeType_t          dstAddrMode;
    uint64_t                dstAddr;
    resultType_t            status;
    macSecurityLevel_t      securityLevel;
    keyIdModeType_t         keyIdMode;
    uint64_t                keySource;
    uint8_t                 keyIndex;
}mlmeCommStatusInd_t;
```

Direction: MAC → NWK

Member	Description	Value
panId	PAN identifier of the device from which the frame was received, or to which the frame was being sent.	uint16_t

Member	Description	Value
srcAddrMode	The coordinator addressing mode for this primitive and subsequent MPDU. This field can take one of the following values:	addrModeType_t
	gAddrModeNoAddress_c	0x00
	gAddrModeShortAddress_c	0x02
	gAddrModeExtendedAddress_c	0x03
	gAddrModeReserved_c	0x01
srcAddr	Contains the individual device address of the entity, from which the frame causing the error originated.	uint64_t
dstAddrMode	The coordinator addressing mode for this primitive and subsequent MPDU. This field can take one of the following values:	addrModeType_t
	gAddrModeNoAddress_c	0x00
	gAddrModeShortAddress_c	0x02
	gAddrModeExtendedAddress_c	0x03
	gAddrModeReserved_c	0x01
dstAddr	Contains the individual device address of the device, for which the frame was intended.	uint64_t
status	The communication status. This field can take one of the following values:	resultType_t
	gSuccess_c	0x00
	gTransactionOverflow_c	0xf1
	gTransactionExpired_c	0xf0
	gChannelAccessFailure_c	0xe1
	gNoAck_c	0xe9
	gCounterError_c	0xdb
	gFrameTooLong_c	0xe5
	gImproperKeyType_c	0xdc
	gImproperSecurityLevel_c	0xdd
	gSecurityError_c	0xe4
	gUnavailableKey_c	0xf3
	gUnsupportedLegacy_c	0xde
	gUnsupportedSecurity_c	0xdf
	gInvalidParameter_c	0xe8
securityLevel	Indicates the security level to be used. Can take one of the following values:	macSecurityLevel_t
	gMacSecurityNone_c	0x00
	gMacSecurityMic32_c	0x01
	gMacSecurityMic64_c	0x02
	gMacSecurityMic128_c	0x03
	gMacSecurityEnc_c	0x04
	gMacSecurityEncMic32_c	0x05
	gMacSecurityEncMic64_c	0x06
	gMacSecurityEncMic128_c	0x07
keyIdMode	This parameter is ignored if the **securityLevel** parameter is set to *gMacSecurityNone_c.* Can take one of the following values:	keyIdModeType_t
	gKeyIdMode0_c	0x00
	gKeyIdMode1_c	0x01
	gKeyIdMode2_c	0x02
	gKeyIdMode3_c	0x03
keySource	Indicates the originator of the key to be used. Ignored if the **keyIdMode** parameter is ignored or set to *gKeyIdMode0_c.*	uint64_t
keyIndex	Indicates the index of the key to be used. Ignored if the **keyIdMode** parameter is ignored or set to *gKeyIdMode0_c.*	uint16_t

3.3.2.21 mlmeSetReq_t

The MLME-SET.request primitive is generated by the next higher layer, and issued to its MLME to write the indicated PIB attribute.

Synopsis:

```
typedef STRUCT mlmeSetReq_tag
{
    pibId_t                 pibAttribute;
    uint8_t                 pibAttributeIndex;
    void*                   pibAttributeValue;
} mlmeSetReq_t;
```

Direction: NWK → MAC

Member	Description	Value
pibAttribute	The identifier of the PIB attribute to write.	pibId_t
pibAttributeIndex	The index within the table of the specified PIB attribute to write. This parameter is valid only for MAC PIB attributes that are tables; it is ignored when accessing PHY PIB attributes.	uint8_t
pibAttributeValue	The value to write to the indicated PIB attribute.	void*

NOTE

The MLME-SET.Request is a synchronous request. The status of the request is returned by the NWK_MLME_SapHandler() function. The MAC will not free the dynamic allocated memory of the request, and it will not send MLME-SET.Confirm!

3.3.2.22 mlmeStartReq_t

The MLME-START.request primitive enables the PAN coordinator to initiate a new PAN, or to begin using a new superframe configuration. This primitive may also be used by a device already associated with an existing PAN to begin using a new superframe configuration.

Synopsis:

```
typedef STRUCT mlmeStartReq_tag
{
    uint16_t              panId;
    logicalChannelId_t    logicalChannel;
    channelPageId_t       channelPage;
    uint32_t              startTime;
    uint8_t               beaconOrder;
    uint8_t               superframeOrder;
    bool_t                panCoordinator;
    bool_t                batteryLifeExtension;
    bool_t                coordRealignment;
    macSecurityLevel_t    coordRealignSecurityLevel;
```

```
    keyIdModeType_t         coordRealignKeyIdMode;
    uint64_t                coordRealignKeySource;
    uint8_t                 coordRealignKeyIndex;
    macSecurityLevel_t      beaconSecurityLevel;
    keyIdModeType_t         beaconKeyIdMode;
    uint64_t                beaconKeySource;
    uint8_t                 beaconKeyIndex;
} mlmeStartReq_t;
```

Direction: NWK → MAC

Member	Description	Value
panId	Contains the PAN identifier to be used by the device.	uint16_t
logicalChannel	The logical channel on which to start using the new superframe configuration. This field can take one of the following values:	logicalChannelId_t
	gLogicalChannel0_c	0
	…	
	gLogicalChannel11_c	11
	gLogicalChannel12_c	12
	…	…
	gLogicalChannel26_c	26
	…	
	gLogicalChannel128_c	128
channelPage	The channel page on which to start using the new superframe configuration. Can take one of the following values:	channelPageId_t
	gChannelPageId0_c	0
	gChannelPageId1_c	1
	gChannelPageId2_c	2
	gChannelPageId9_c	9
startTime	The time at which to begin transmitting beacons. If this parameter is equal to 0x000000, beacon transmissions will begin immediately. This is a 24-bit value.	uint32_t
beaconOrder	How often the beacon is to be transmitted. A value of 15 indicates that the coordinator will not transmit periodic beacons. Valid range is 0-15.	uint8_t
superframeOrder	The length of the active portion of the superframe, including the beacon frame. If the beaconOrder parameter (BO) has a value of 15, this parameter is ignored. Valid range is 0-14.	uint8_t
panCoordinator	Indicates if the device will become the PAN coordinator of a new PAN or begin using a new superframe configuration on the PAN with which it is associated.	bool_t
batteryLifeExtension	If this value is TRUE, the receiver of the beaconing device is disabled macBattLifeExtPeriods full backoff periods after the inter-frame spacing (IFS) period following the beacon frame. If this value is FALSE, the receiver of the beaconing device remains enabled for the entire CAP.	bool_t
coordRealignment	Indicates if a coordinator realignment command is to be transmitted before changing the superframe.	bool_t
coordRealignSecurityLevel	Indicates the security level to be used for coordinator realignment command frames.	macSecurityLevel_t

Member	Description	Value
	Can take one of the following values:	
	gMacSecurityNone_c	0x00
	gMacSecurityMic32_c	0x01
	gMacSecurityMic64_c	0x02
	gMacSecurityMic128_c	0x03
	gMacSecurityEnc_c	0x04
	gMacSecurityEncMic32_c	0x05
	gMacSecurityEncMic64_c	0x06
	gMacSecurityEncMic128_c	0x07
coordRealignKeyIdMode	The field contains the mode used to identify the key to be used. This parameter is ignored if the **coordRealignSecurityLevel** parameter is set to *gMacSecurityNone_c*. Can take one of the following values:	keyIdModeType_t
	gKeyIdMode0_c	0x00
	gKeyIdMode1_c	0x01
	gKeyIdMode2_c	0x02
	gKeyIdMode3_c	0x03
coordRealignKeySource	Indicates the originator of the key to be used. Ignored if the **coordRealignKeyIdMode** parameter is ignored or set to *gKeyIdMode0_c.*	uint64_t
coordRealignKeyIndex	Indicates the index of the key to be used. Ignored if the **coordRealignKeyIdMode** parameter is ignored or set to *gKeyIdMode0_c.*	uint16_t
beaconSecurityLevel	Indicates the security level to be used in beacon frames. Can take one of the following values:	macSecurityLevel_t
	gMacSecurityNone_c	0x00
	gMacSecurityMic32_c	0x01
	gMacSecurityMic64_c	0x02
	gMacSecurityMic128_c	0x03
	gMacSecurityEnc_c	0x04
	gMacSecurityEncMic32_c	0x05
	gMacSecurityEncMic64_c	0x06
	gMacSecurityEncMic128_c	0x07
beaconKeyIdMode	The field contains the mode used to identify the key to be used. This parameter is ignored if the **beaconSecurityLevel** parameter is set to *gMacSecurityNone_c*. Can take one of the following values:	keyIdModeType_t
	gKeyIdMode0_c	0x00
	gKeyIdMode1_c	0x01
	gKeyIdMode2_c	0x02
	gKeyIdMode3_c	0x03
beaconKeySource	Indicates the originator of the key to be used. Ignored if the **beaconKeyIdMode** parameter is ignored or set to *gKeyIdMode0_c.*	uint64_t
beaconKeyIndex	Indicates the index of the key to be used. Ignored if the **beaconKeyIdMode** parameter is ignored or set to *gKeyIdMode0_c.*	uint16_t

3.3.2.23 **mlmeStartCnf_t**

The MLME-START.confirm primitive reports the results of the attempt to start using a new superframe configuration.

The MLME-START.confirm primitive is generated by the MLME, and issued to NWK layer in response to the MLME-START.request primitive.

The MLME-START.confirm primitive returns a status of either SUCCESS, indicating that the MAC sublayer has started using the new superframe configuration, or an appropriate error code.

Synopsis:

```
typedef STRUCT mlmeStartCnf_tag
{
    resultType_t             status;
} mlmeStartCnf_t;
```

Direction: MAC → NWK

Member	Description	Value
status	The result of the attempt to start using an updated frame configuration. Can take one of the following values:	resultType_t
	gSuccess_c	0x00
	gNoShortAddress_c	0xec
	gSuperframeOverlap_c	0xfd
	gTrackingOff_c	0xf8
	gInvalidParameter_c	0xe8
	gCounterError_c	0xdb
	gFrameTooLong_c	0xe5
	gUnavailableKey_c	0xf3
	gUnsupportedSecurity_c	0xdf
	gChannelAccessFailure_c	0xe1

3.3.2.24 mlmeSyncReq_t

The MLME-SYNC.request primitive is generated by the next higher layer of a device on a beacon-enabled PAN, and issued to its MLME to synchronize with the coordinator.

Synopsis:

```
typedef STRUCT mlmeSyncReq_tag
{
    logicalChannelId_t      logicalChannel;
    channelPageId_t         channelsPage;
    bool_t                  trackBeacon;
} mlmeSyncReq_t;
```

Direction: NWK → MAC

Member	Description	Value
logicalChannel	The logical channel on which to attempt coordinator synchronization.	logicalChannelId_t
	gLogicalChannel0_c	0
	…	
	gLogicalChannel11_c	11
	gLogicalChannel12_c	12
	…	…
	gLogicalChannel26_c	26
	…	
	gLogicalChannel128_c	128
channelPage	The channel page on which to attempt coordinator synchronization.	channelPageId_t
trackBeacon	TRUE if the MLME is to synchronize with the next beacon, and to	bool_t

	attempt to track all future beacons. FALSE if the MLME is to synchronize with only the next beacon.	

3.3.2.25 **mlmePollReq_t**

The MLME-POLL.request primitive prompts the device to request data from the coordinator.

Synopsis:

```
typedef STRUCT mlmePollReq_tag
{
    addrModeType_t          coordAddrMode;
    uint16_t                coordPANId;
    uint64_t                coordAddress;
    macSecurityLevel_t      securityLevel;
    keyIdModeType_t         keyIdMode;
    uint64_t                keySource;
    uint8_t                 keyIndex;
} mlmePollReq_t;
```

Direction: NWK → MAC

Member	Description	Value
coordAddrMode	The coordinator addressing mode to which the poll request is intended. This field can take one of the following values:	addrModeType_t
	gAddrModeShortAddress_c	0x02
	gAddrModeExtendedAddress_c	0x03
coordPANId	The PAN identifier of the coordinator for which the poll is intended.	uint16_t
coordAddress	The address of the coordinator for which the poll is intended.	uint64_t
securityLevel	Indicates the security level to be used. Can take one of the following values:	macSecurityLevel_t
	gMacSecurityNone_c	0x00
	gMacSecurityMic32_c	0x01
	gMacSecurityMic64_c	0x02
	gMacSecurityMic128_c	0x03
	gMacSecurityEnc_c	0x04
	gMacSecurityEncMic32_c	0x05
	gMacSecurityEncMic64_c	0x06
	gMacSecurityEncMic128_c	0x07
keyIdMode	This parameter is ignored if the **securityLevel** parameter is set to *gMacSecurityNone_c.* Can take one of the following values:	keyIdModeType_t
	gKeyIdMode0_c	0x00
	gKeyIdMode1_c	0x01
	gKeyIdMode2_c	0x02
	gKeyIdMode3_c	0x03
keySource	Indicates the originator of the key to be used. Ignored if the **keyIdMode** parameter is ignored or set to *gKeyIdMode0_c.*	uint64_t
keyIndex	Indicates the index of the key to be used. Ignored if the **keyIdMode** parameter is ignored or set to *gKeyIdMode0_c.*	uint16_t

3.3.2.26 **mlmeSyncLossInd_t**

This primitive is generated by the MLME of a device, and issued to its next higher layer in the event of losing synchronization with the coordinator. It is also generated by the MLME

of the PAN coordinator and issued to its next higher layer in the event of a PAN ID conflict.

Synopsis:

```
typedef STRUCT mlmeSyncLossInd_tag
{
    resultType_t            lossReason;
    uint16_t                panId;
    logicalChannelId_t      logicalChannel;
    channelPageId_t         channelPage;
    macSecurityLevel_t      securityLevel;
    keyIdModeType_t         keyIdMode;
    uint64_t                keySource;
    uint8_t                 keyIndex;
} mlmeSyncLossInd_t;
```

Direction: MAC → NWK

Member	Description	Value
lossReason	The reason why the synchronization was lost. Can take one of the following values:	resultType_t
	gPanIdConflict_c	0xee
	gRealignment_c	0xef
	gBeaconLost_c	0xe0
PANId	The PAN identifier with which the device lost the synchronization, or to which it was realigned.	uint16_t
logicalChannel	The logical channel on which the device lost the synchronization, or to which it was realigned. Can take one of the following values:	logicalChannelId_t
	gLogicalChannel0_c	0
	…	
	gLogicalChannel11_c	11
	gLogicalChannel12_c	12
	…	…
	gLogicalChannel26_c	26
	…	
	gLogicalChannel128_c	128
channelPage	The channel page on which the device lost the synchronization, or to which it was realigned. Can take one of the following values:	channelPageId_t
	gChannelPageId0_c	0
	gChannelPageId1_c	1
	gChannelPageId2_c	2
	gChannelPageId9_c	9
securityLevel	Indicates the security level to be used. Can take one of the following values:	macSecurityLevel_t
	gMacSecurityNone_c	0x00
	gMacSecurityMic32_c	0x01
	gMacSecurityMic64_c	0x02
	gMacSecurityMic128_c	0x03
	gMacSecurityEnc_c	0x04
	gMacSecurityEncMic32_c	0x05
	gMacSecurityEncMic64_c	0x06
	gMacSecurityEncMic128_c	0x07
keyIdMode	This parameter is ignored if the **securityLevel** parameter is set to *gMacSecurityNone_c.* Can take one of the following values:	keyIdModeType_t
	gKeyIdMode0_c	0x00
	gKeyIdMode1_c	0x01
	gKeyIdMode2_c	0x02
	gKeyIdMode3_c	0x03

Member	Description	Value
keySource	Indicates the originator of the key to be used. Ignored if the **keyIdMode** parameter is ignored or set to *gKeyIdMode0_c.*	uint64_t
keyIndex	Indicates the index of the key to be used. Ignored if the **keyIdMode** parameter is ignored or set to *gKeyIdMode0_c.*	uint16_t
securityLevel	Indicates the security level to be used.	macSecurityLevel_t

3.3.2.27 **mlmePollCnf_t**

The MLME-POLL.confirm primitive.

Synopsis:

```
typedef STRUCT mlmePollCnf_tag
{
    resultType_t            status;
} mlmePollCnf_t;
```

Direction: MAC → NWK

Member	Description	Value
status	The status of the data request. Can take one of the following values:	resultType_t
	gSuccess_c	0x00
	gChannelAccessFailure_c	0xe1
	gNoAck_c	0xe9
	gNoData_c	0xe9
	gCounterError_c	0xdb
	gFrameTooLong_c	0xe5
	gUnavailableKey_c	0xf3
	gUnsupportedSecurity_c	0xdf
	gInvalidParameter_c	0xe8

3.3.2.28 **mlmePollNotifyInd_t**

The MLME-POLL-NOTIFY.indication primitive reports a poll indication message to the network.

Synopsis:

```
typedef STRUCT mlmePollNotifyInd_tag
{
    addrModeType_t          srcAddrMode;
    uint64_t                srcAddr;
    uint16_t                srcPanId;
} mlmePollNotifyInd_t;
```

Direction: MAC → NWK

Member	Description	Value
srcAddrMode	The source addressing mode for this primitive. This field can take one of the following values:	addrModeType_t
	gAddrModeNoAddress_c	0x00
	gAddrModeShortAddress_c	0x02
	gAddrModeExtendedAddress_c	0x03
	gAddrModeReserved_c	0x01
srcAddr	The individual device address of the entity from which the notification was received.	uint64_t
srcPanId	PAN identifier of the entity from which the notification was received.	uint16_t

3.3.2.29 mlmeSetSlotframeReq_t

The MLME-SET-SLOTFRAME.request primitive is generated by the next higher layer, and issued to the MLME to request a set of a TSCH slotframe to the MAC sublayer.

Synopsis:

```
typedef MAC_STRUCT mlmeSetSlotframeReq_tag
{
    uint8_t                 slotframeHandle;
    macSetSlotframeOp_t     operation;
    uint16_t                size;
} mlmeSetSlotframeReq_t;
```

Direction: NWK → MAC

Member	Description	Value
slotframeHandle	Unique identifier of the slotframe.	uint8_t
operation	Operation to be performed on the slotframe.	macSetSlotframeOp_t
	gMacSetSlotframeOpAdd_c	0x00
	gMacSetSlotframeOpDelete_c	0x02
	gMacSetSlotframeOpModify_c	0x03
size	Number of timeslots in the new slotframe.	uint16_t

3.3.2.30 mlmeSetSlotframeCnf_t

The MLME-SET-SLOTFRAME.confirm primitive is generated by the MAC layer and issued to the next higher layer to confirm the set of a TSCH slotframe to the MAC sublayer.

Synopsis:

```
typedef MAC_STRUCT mlmeSetSlotframeCnf_tag
{
    uint8_t                 slotframeHandle;
    resultType_t            status;
} mlmeSetSlotframeCnf_t;
```

Direction: MAC → NWK

Member	Description	Value
slotframeHandle	Unique identifier of the slotframe to be added, deleted, or modified.	uint8_t
status	Indicates the result of the MLME-SET-SLOTFRAME.request.	resultType_t
	gSuccess_c	0x00
	gInvalidParameter_c	0xe8
	gSlotframeNotFound_c	0x03
	gMaxSlotframesExceeded_c	0x04

3.3.2.31 mlmeSetLinkReq_t

The MLME-SET-LINK.request primitive is generated by the next higher layer, and issued to the MLME to request a set of a TSCH link to the MAC sublayer.

Synopsis:

```
typedef MAC_STRUCT mlmeSetLinkReq_tag
{
    macSetLinkOp_t          operation;
    uint16_t                linkHandle;
```

```
    uint8_t                 slotframeHandle;
    uint16_t                timeslot;
    uint16_t                channelOffset;
    macLinkOptions_t        linkOptions;
    macLinkType_t           linkType;
    uint16_t                nodeAddr;
} mlmeSetLinkReq_t;
```

Direction: NWK → MAC

Member	Description	Value
operation	Operation to be performed on the link.	macSetLinkOp_t
	gMacSetLinkOpAdd_c	0x00
	gMacSetLinkOpDelete_c	0x01
	gMacSetLinkOpModify_c	0x02
linkHandle	Unique identifier (local to specified slotframe) for the link.	uint16_t
slotframeHandle	The slotframeHandle of the slotframe to which the link is associated.	uint8_t
timeslot	Timeslot of the link to be added.	uint16_t
channelOffset	The Channel offset of the link.	uint16_t
linkOptions	Link Options	macLinkOptions_t
	Bit 0: Tx	
	Bit 1: Rx	
	Bit 2: Shared (for Tx links)	
	Bit 3: Timekeeping (for Rx links)	
linkType	Type of the link; indicates if the link may be used for sending Enhanced Beacons.	macLinkType_t
	gMacLinkTypeNormal_c	0x00
	gMacLinkTypeAdvertising_c	0x01
nodeAddr	Address of the neighbor device connected via the link. 0xffff indicates that the link can be used for frames sent with broadcast address.	uint16_t

3.3.2.32 mlmeSetLinkCnf_t

The MLME-SET-LINK.confirm primitive is generated by the MAC layer, and issued to the next higher layer to confirm the set of a TSCH link to the MAC sublayer.

Synopsis:

```
typedef MAC_STRUCT mlmeSetLinkCnf_tag
{
    resultType_t            status;
    uint16_t                linkHandle;
    uint8_t                 slotframeHandle;
} mlmeSetLinkCnf_t;
```

Direction: MAC → NWK

Member	Description	Value
status	Indicates the result of the MLME-SET-LINK.request.	resultType_t
	gSuccess_c	0x00
	gInvalidParameter_c	0xe8
	gUnkonwnLink_c	0x05
	gMaxLinksExceeded_c	0x06
linkHandle	A unique (local to specified slotframe) identifier of the link.	uint16_t
slotframeHandle	The slotframeHandle of the slotframe to which the link is associated.	uint8_t

3.3.2.33 mlmeTschModeReq_t

The MLME-TSCH-MODE.request primitive is generated by the next higher layer, and issued to the MLME to request the TSCH enable or disable procedure.

Synopsis:

```
typedef MAC_STRUCT mlmeTschModeReq_tag
{
    macTschMode_t           tschMode;
} mlmeTschModeReq_t;
```

Direction: NWK → MAC

Member	Description	Value
tschMode	Target mode	macTschMode_t
	gMacTschModeOff_c	0x00
	gMacTschModeOn_c	0x01

3.3.2.34 mlmeTschModeCnf_t

The MLME-TSCH-MODE.confirm primitive is generated by the MAC layer, and issued to the next higher layer to confirm the enable or disable of the TSCH module.

Synopsis:

```
typedef MAC_STRUCT mlmeTschModeCnf_tag
{
    macTschMode_t           tschMode;
    resultType_t            status;
} mlmeTschModeCnf_t;
```

Direction: MAC → NWK

Member	Description	Value
tschMode	Target mode	macTschMode_t
	gMacTschModeOff_c	0x00
	gMacTschModeOn_c	0x01
status	Indicates result of the MLME-SET-SLOTFRAME.request.	resultType_t
	gSuccess_c	0x00
	gNoSync_c	0x07

3.3.2.35 mlmeKeepAliveReq_t

The MLME-KEEP-ALIVE.request primitive is generated by the next higher layer, and issued to the MLME to request monitor of frames exchanged with a TSCH neighbor used as clock source, and configure the keep-alive period in timeslots for that clock source.

Synopsis:

```
typedef MAC_STRUCT mlmeKeepAliveReq_tag
{
    uint16_t                dstAddr;
    uint16_t                keepAlivePeriod;
} mlmeKeepAliveReq_t;
```

Direction: NWK → MAC

Member	Description	Value
dstAddr	Address of the neighbor device with which to maintain the timing. Keep-alives with dstAddr of 0xffff do not expect to be acknowledged, and cannot be used for timekeeping.	uint16_t
keepAlivePeriod	Period in timeslots after which a frame is sent if no frames have been sent to dstAddr.	uint16_t

3.3.2.36 mlmeKeepAliveCnf_t

The MLME-KEEP-ALIVE.confirm primitive is generated by the MAC layer, and issued to the next higher layer to confirm the request to configure the keep-alive frames for a TSCH clock source.

Synopsis:

```
typedef MAC_STRUCT mlmeKeepAliveCnf_tag
{
    resultType_t            status;
} mlmeKeepAliveCnf_t;
```

Direction: MAC → NWK

Member	Description	Value
status	Indicates results of the MLME-KEEP-ALIVE.request.	resultType_t
	gSuccess_c	0x00
	gInvalidParameter_c	0xe8

3.3.2.37 mlmeBeaconReq_t

The MLME-BEACON.request primitive is generated by the next higher layer, and issued to the MLME to request the MAC layer to send a beacon or enhanced beacon frame with the specified parameters. It is used for advertising the TSCH parameters to allow new devices to join the network.

Synopsis:

```
typedef MAC_STRUCT mlmeBeaconReq_tag
{
    beaconType_t            beaconType;
    uint8_t                 channel;
    channelPageId_t         channelPage;
    uint8_t                 superframeOrder;
    macSecurityLevel_t      beaconSecurityLevel;
    keyIdModeType_t         beaconKeyIdMode;
    uint64_t                beaconKeySource;
    uint8_t                 beaconKeyIndex;
    addrModeType_t          dstAddrMode;
    uint64_t                dstAddr;
    bool_t                  bsnSuppression;
} mlmeBeaconReq_t;
```

Direction: NWK → MAC

Member	Description	Value
beaconType	Indicates whether to send a beacon or an enhanced beacon.	beaconType_t
	gMacBeacon_c	0x00
	gMacEnhancedBeacon_c	0x01
channel	The channel number to use.	uint8_t
channelPage	The channel page to use.	channelPageId_t
superframeOrder	The length of the active portion of the superframe, including the beacon frame. Valid range: 0-15.	uint8_t
beaconSecurityLevel	Indicates the security level to be used in beacon frames. Can take one of the following values:	macSecurityLevel_t
	gMacSecurityNone_c	0x00
	gMacSecurityMic32_c	0x01
	gMacSecurityMic64_c	0x02
	gMacSecurityMic128_c	0x03
	gMacSecurityEnc_c	0x04
	gMacSecurityEncMic32_c	0x05
	gMacSecurityEncMic64_c	0x06
	gMacSecurityEncMic128_c	0x07
beaconKeyIdMode	The field contains the mode used to identify the key to be used. This parameter is ignored if the **beaconSecurityLevel** parameter is set to *gMacSecurityNone_c*. Can take one of the following values:	keyIdModeType_t
	gKeyIdMode0_c	0x00
	gKeyIdMode1_c	0x01
	gKeyIdMode2_c	0x02
	gKeyIdMode3_c	0x03
beaconKeySource	Indicates the originator of the key to be used. Ignored if the **beaconKeyIdMode** parameter is ignored or set to *gKeyIdMode0_c.*	uint64_t
beaconKeyIndex	Indicates the index of the key to be used. Ignored if the **beaconKeyIdMode** parameter is ignored or set to *gKeyIdMode0_c.*	uint16_t
dstAddrMode	The destination addressing mode for this primitive and subsequent MPDU. This field can take one of the following values:	addrModeType_t
	gAddrModeNoAddress_c	0x00
	gAddrModeShortAddress_c	0x02
	gAddrModeExtendedAddress_c	0x03
	gAddrModeReserved_c	0x01
dstAddr	The individual device address of the entity to which the MSDU is being transferred.	uint64_t
bsnSuppression	If BeaconType = 0x01, then if BSNSuppression is TRUE, the EBSN is omitted from the frame and the Sequence Number Suppression field of the Frame Control field is set to 1.	bool_t

3.3.2.38 **mlmeBeaconCnf_t**

The MLME-BEACON.confirm primitive is generated by the MAC layer, and issued to the next higher layer to indicate the result of the MLME-BEACON.request primitive.

Synopsis:

```
typedef MAC_STRUCT mlmeBeaconCnf_tag
{
    resultType_t            status;
```

```
} mlmeBeaconCnf_t;
```

Direction: MAC → NWK

Member	Description	Value
status	Indicates results of the MLME-BEACON.request.	resultType_t
	gSuccess_c	0x00
	gChannelAccessFailure_c	0xe1
	gFrameTooLong_c	0xe5
	gInvalidParameter_c	0xe8

3.4 SAP function definitions

The MAC Layer can be accessed by the higher layer using the following interface functions:

3.4.1 Generic interface

These interface functions are used for initialization and general configuration of the MAC layer.

3.4.1.1 MAC_Init

This function creates and initializes all of the MAC instances that the system has been designed with.

Synopsis:

```
void MAC_Init( void );
```

Direction: NWK → MAC

3.4.1.2 BindToMAC

This function creates a logical binding with the next available (un-binded) MAC instance.

Synopsis:

```
instanceId_t BindToMAC( instanceId_t nwkId );
```

Direction: NWK → MAC

Argument	Description	Type
nwkId	The instance with which the caller layer will create the logical binding.	instanceId_t

3.4.1.3 UnBindFromMAC

This function removes the logical bind of the specified MAC instance.

Synopsis:

```
void UnBindFromMAC( instanceId_t macInstanceId );
```

Direction: NWK → MAC

Argument	Description	Type
macInstanceId	The MAC instance from which the caller layer will remove the logical binding.	instanceId_t

3.4.1.4 Mac_Task

This function is the main MAC task, which will instantiate one MAC layer.

Synopsis:

```
void Mac_Task( void const* taskParam );
```

Direction: NWK → MAC

Argument	Description	Type
taskParam	Generic parameter passed at task creation. Used for task / instance initialization.	void const*

3.4.1.5 Mac_GetState

This function checks the states of all instances of the MAC, and returns a status as follows:

- If at least one of the instances has an Active sequence, `gMacStateBusy_c` is returned
- If at least one of the instances has pending messages, `gMacStateNotEmpty_c` is returned
- If all the instances are Idle and `gMacStateIdle_c` is returned

Synopsis:

```
macState_t Mac_GetState( void );
```

Direction: NWK → MAC

3.4.1.6 Mac_RegisterSapHandlers

This function registers the MCPS and MLME SAPs, offering support for the MCPS and MLME to NWK message interactions.

Synopsis:

```
void Mac_RegisterSapHandlers( MCPS_NWK_SapHandler_t pMCPS_NWK_SapHandler,
                              MLME_NWK_SapHandler_t pMLME_NWK_SapHandler,
                              instanceId_t macInstanceId);
```

Direction: NWK → MAC

Argument	Description	Type
pMCPS_NWK_SapHandler	Pointer to the MCPS to NWK SAP Handler.	MCPS_NWK_SapHandler_t
pMLME_NWK_SapHandler	Pointer to the MLME to NWK SAP Handler function callback.	MLME_NWK_SapHandler_t
macInstanceId	MAC instance for which the SAP registration is performed.	instanceId_t

3.4.1.7 Mac_GetMaxMsduLength

This function returns the maximum MSDU that can be accommodated considering the current PHY configuration.

Synopsis:

```
uint8_t Mac_GetMaxMsduLength( mcpsDataReq_t* pParams );
```

Direction: NWK → MAC

Argument	Description	Type
pParams	Points to the MCPS-DATA.Request structure that is requested for transmission.	mcpsDataReq_t

3.4.2 MCPS SAP

The MAC data service is accessed through the MAC common part sublayer (MCPS) data SAP. The MCPS-SAP supports the transport of SSCS protocol data units (SPDUs) between peer SSCS entities.

3.4.2.1 NWK_MCPS_SapHandler

This function is part of the MCPS SAP, offering support for the NWK to MCPS message interactions.

Synopsis:

```
resultType_t NWK_MCPS_SapHandler(nwkToMcpsMessage_t* pMsg, instanceId_t
macInstanceId);
```

Direction: NWK → MAC

Argument	Description	Type
pMsg	Pointer to a structure containing the message information – the primitive identifier and primitive parameters.	nwkToMcpsMessage_t*
macInstanceId	Identifier of the MAC instance for which the primitive is called.	instanceId_t

3.4.2.2 MCPS_NWK_SapHandler_t

This type describes the function callback that is part of the MCPS SAP, offering support for the MCPS to NWK message interactions. This function callback must be registered by the NWK entity.

Synopsis:

```
typedef resultType_t (*MCPS_NWK_SapHandler_t) (mcpsToNwkMessage_t* pMsg,
instanceId_t upperInstanceId);
```

Direction: MAC → NWK

Argument	Description	Type
pMsg	Pointer to a structure containing the message information – the primitive identifier and primitive parameters.	mcpsToNwkMessage_t*
upperInstanceId	Identifier of the MAC instance for which the primitive is called.	instanceId_t

3.4.3 MLME SAP

The MAC sublayer conceptually includes a management entity called the MLME. This entity provides the service interfaces, through which layer management functions can be invoked.

The MLME is also responsible for maintaining a database of managed objects pertaining to the MAC sublayer. This database is referred to as the MAC sublayer PIB.

The MLME primitives are listed below.

3.4.3.1 NWK_MLME_SapHandler

This function is part of the MLME SAP, offering support for the NWK to MLME message interactions.

Synopsis:

```
resultType_t NWK_MLME_SapHandler( mlmeMessage_t* pMsg, instanceId_t macInstanceId
);
```

Direction: NWK → MAC

Argument	Description	Type
pMsg	Pointer to a structure containing the message information – the primitive identifier and primitive parameters.	mlmeMessage_t*
macInstanceId	Identifier of the MAC instance for which the primitive is called.	instanceId_t

3.4.3.2 MLME_NWK_SapHandler_t

This type describes the function callback that is a part of the MLME SAP, offering support for the MLME to NWK message interactions. This function callback must be registered by the NWK entity.

Synopsis:

```
typedef resultType_t (*MLME_NWK_SapHandler_t) (nwkMessage_t* pMsg, instanceId_t
upperInstanceId);
```

Direction: MAC → NWK

Argument	Description	Type
pMsg	Pointer to a structure containing the message information – the primitive identifier and primitive parameters.	genericMsg_t*
upperInstanceId	Identifier of the MAC instance for which the primitive is called	instanceId_t

3.4.4 Phy to MAC SAP

The following functions are exposed towards the PHY layer, offering the upwards interface of the communication stack from PHY to MAC.

3.4.4.1 PD_MAC_SapHandler

This function is part of the PHY data service, offering support for the PD-SAP to MAC interactions.

Synopsis:

```
resultType_t PD_MAC_SapHandler( void* pMsg, instanceId_t macInstanceId );
```

Direction: PHY → MAC

Argument	Description	Type
pMsg	Pointer to a structure containing the message information – the primitive identifier and primitive parameters.	void *
macInstanceId	Identifier of the MAC instance for which the primitive is called.	instanceId_t

3.4.4.2 PLME_MAC_SapHandler

This function is a part of the PHY management service, offering support for the PLME-SAP to MAC interactions.

Synopsis:

```
resultType_t PLME_MAC_SapHandler( void* pMsg, instanceId_t macInstanceId );
```

Direction: PHY → MAC

Argument	Description	Type
pMsg	Pointer to a structure containing the message information – the primitive identifier and primitive parameters.	void *
macInstanceId	Identifier of the MAC instance for which the primitive is called.	instanceId_t

4 IEEE 802.15.4 PHY interface

The Freescale PHY Layer deals with the physical burst, which is to be sent and/or received. It performs modulation and demodulation, transmitter and receiver switching, fragmentation, scrambling, interleaving, and error correction coding. The communication with the upper protocol layers is carried out through the Layer 1 Interface. The PHY Layer is capable of executing the following sequences:

- I (Idle)
- R (Receive Sequence conditionally followed by a TxAck)
- T (Transmit Sequence)
- C (Standalone CCA)
- CCCA (Continuous CCA) TR (Transmit / Receive Sequence – transmit unconditionally followed by either an R or RxAck)

NOTE

For Sub-1 GHz PHY, the CCA and TR states are not available. Receive Sequence conditionally followed by a TxAck or Transmit Sequence conditionally followed by RxAck are software managed.

In addition to these sequences, the PHY Layer also integrates a packet processor, which determines whether the packet is MAC-compliant, and if it is, it checks whether it is addressed to the end device. Another feature of the packet processor is Source Address Matching, which can be viewed as an extension of packet filtering. However, its function is very specific to its intended application (data-polling and indirect queue management by a PAN Coordinator).

4.1 PHY features

The PHY features, such as I, R, T, C, CCCA, and TR sequences, packet processor filtering and source address filtering, can be implemented in hardware or emulated in software depending on the transceiver used.

4.1.1 Sequence manager

The sequence manager is a state machine that controls the timing of all transmit, receive and CCA operations. Sequences can be initiated directly by the MAC Layer or automatically at the expiration of a timer.

4.1.1.1 Idle sequence

When a request to enter the **idle sequence** is received from the MAC Layer, if not already in this state, the PHY executes an orderly warm-down of the transceiver, and sends it into the idle state. After this operation is completed, a confirm primitive is sent to the MAC Layer. Requesting to enter the **idle sequence** is the proper way to abort any other sequence.

4.1.1.2 Receive sequence

The **receive sequence** is used to put the transceiver into the Rx state for the reception of an incoming data transmission. Although the reception of ACK frames is possible using the R sequence, the recommended way is to use TR sequences. This is because reception of an ACK frame follows the transmission of a MAC data or command frame with a designated Sequence Number, which must match the ACK frame Sequence Number. If an R sequence is used instead, there is no Sequence Number to match against, and the ACK frame is passed to the MAC Layer using the PD-DATA.indication primitive.

The R sequence must be used to receive all IEEE 802.15.4 PHY and MAC compliant frames, including reserved frame types. The PHY Layer must execute the R sequence as follows:

1. Set a timer to trigger the start of sequence (optional).
2. Wait for timer trigger.
3. Set a timer to timeout the sequence execution (optional)[1].
4. Execute Rx warm-up of the transceiver.
5. Wait for the transceiver notification of a received packet.

[1] if a timeout occurs at any point during the rest of the process, the sequence must be cancelled and a plmeTimeoutInd primitive must be issued.

6. Transfers the payload into internal buffer[2].
7. If the CRC passes, and in non-promiscuous mode filter rules checking passes[3]:
   - Execute Rx warm-down of the transceiver.
   - If automatic Ack is enabled and non-promiscuous or active promiscuous mode is enabled conditioned by the packet being addressed to the device, the PHY Layer checks Ack Request bit in Frame Control field.
     - If the frame type is Data Request check Source Address Matching. If a match is detected assert the Frame Pending bit in the Frame Control field of the Ack frame.
     - Frame version is copied from the received frame to the Ack frame
     - Sets a timer that triggers at 192 µs after receiving the packet (IEEE 802.15.4 RX-to-TX turnaround time)[4] minus the Tx warm-up period.
     - Execute Tx warm-up of the transceiver.
     - Initiate transmission.
     - Wait for transceiver notification of completed transmission.
     - Perform Tx warm-down.
8. Mark the PHY as being idle.
9. Notify the MAC Layer of the received packet (using PD-DATA.indication primitive).

4.1.1.3 Transmit sequence

The transmit sequence is used to put the transceiver into the Tx state for transmission of an outgoing MAC data or command frame. Although the transmission of ACK frames is possible using the T sequence, the recommended way is to use R sequences with auto ACK enabled. This is because the transmission of an ACK frame follows the reception of a MAC data or command frame with a designated Sequence Number, which must be copied to the ACK frame Sequence Number field. If a T sequence is used instead, there is no Sequence Number to copy, and the ACK frame must be created by the MAC Layer.

Sequence T allows for the insertion of one or two CCA measurements before the transmission to ensure that the channel is idle. All CCA measurements must indicate an idle channel to proceed with the transmission. If the channel is determined to be busy, the sequence must be terminated and a PD-DATA.confirm primitive with a status of CHANNEL_BUSY must be issued.

---

[2] For SubGHz PHY, the transceiver sends notification for SFD received. Each byte of the payload is transferred into the buffer and filters are applied.

[3] If either the CRC or filter rules checking fails, the payload must be discarded and the PHY must continue waiting for a transceiver notification of a received packet.

[4] For SubGHz PHY, a 1000 µs period is set (IEEE 802.15.4g Rx-to-Tx turnaround time).

Warning ! To maintain the Turnaround time of 1000 µs, no interrupts can last longer than 50 µs, if a sequence with turnaround time is ongoing.

The T sequence must be used to transmit all IEEE 802.15.4 PHY and MAC compliant frames including reserved frame types. The PHY Layer must execute the T sequence as follows:

1. Set a timer to trigger the start of transmission (optional).
2. Wait for timer trigger.
3. Set a timer to timeout the sequence execution (optional)[5].
4. If CCA before Tx is required:
    - Execute Rx warm-up of the transceiver.
    - Initiate CCA measurement[6].
    - If CCA indicates a busy channel:
        - Terminate sequence and issue a PD-DATA.confirm primitive with a status of CHANNEL_BUSY.
    - Else if CCA indicates channel idle:
        - If slotted mode is not used, proceed to Rx warm-down.
    - Else if slotted mode is used[7], initiate a second CCA, 320 μs after initiating first CCA:
        - If CCA indicates a busy channel, terminate sequence and issues a PD-DATA.confirm primitive with a status of CHANNEL_BUSY.
    - Else if CCA indicates channel idle.
5. Execute Rx warm-down if slotted mode is used, wait 320 μs after second CCA.
6. Execute Tx warm-up and pass data to the transceiver.
7. Wait for transceiver notification of completed transmission.
8. Perform Tx warm-down.
9. Mark the PHY as being idle.
10. Issue a PD-DATA.confirm primitive with a status of SUCCESS.

At any time before executing Tx warm-up and passing data to the transceiver, the PHY Layer must calculate the CRC of the frame, and populate the FCS field.

4.1.1.4 Standalone CCA sequence

During the standalone CCA sequence, the PHY Layer executes a single CCA measurement and reports the result to the MAC Layer.

The execution of a C sequence is as follows:

- Set a timer to trigger the start of sequence (optional).
- Wait for timer trigger.

[5] If a timeout occurs at any point during the rest of the process, the sequence must be cancelled and a plmeTimeoutInd primitive must be issued.
[6] For SubGHz PHY, a timer is set for the CCA duration period to sample RSSI level for the CCA.
[7] For SubGHz PHY, slotted mode is not available.

- Set a timer to timeout the sequence execution (optional)[8].
- Execute Rx warmup of the transceiver.
- Initiate CCA measurement in the transceiver.
- Wait for the transceiver notification of a completed measurement.
- Perform Rx warm-down.
- Mark the PHY as being idle.
- Issue a PLME-CCA.confirm primitive.

4.1.1.5 Continuous CCA sequence9

This sequence is designed to accommodate situations where channel availability may be infrequent. During CCCA sequence, the PHY repeats the standalone CCA measurement until an idle channel condition is found. This sequence is used as a part of T or TR sequence instead of the normal CCA sequence.

The execution of the CCCA sequence is as follows:

- Set a timer to trigger the start of the sequence (optional).
- Wait for timer trigger.
- Executes Rx warm-up of the transceiver.
- While previous CCA measurement indicates channel busy.
- Initiate CCA measurement in the transceiver.
- Wait for transceiver notification of completed measurement.
- Performs Rx warm-down.
- Marks the PHY as being idle.
- Issues a PLME-CCA.confirm primitive Transmit / Receive Sequence[10].

Sequence TR is a combination of transmit / receive sequences. The sequence is executed as a concatenation of one transmit operation followed by one receive operation.

There are two types of TR sequences depending on auto ACK being enabled, and the ACK request bit in the frame control field of the transmitted frame being asserted. In this case, the R part of the sequence becomes receive ACK only.

For both cases, the sequence T that constitutes the first half of the operation is identical to the standalone T sequence. If either auto ACK is disabled or the transmitted frame does not request ACK, then the TR sequence is executed as a T sequence followed by an R sequence.

[8] If a timeout occurs at any point during the rest of the process, the sequence must be cancelled, and a plmeTimeoutInd primitive must be issued.
[9] For SubGHz PHY, Continuous CCA Sequence is not available.
[10] For SubGHz PHY, Transmit / Receive Sequence is not available.

The R sequence that constitutes the second half of the operation and is identical to the standalone version, can be followed by an additional automatically transmitted ACK frame if auto ACK is enabled and the incoming frame requires an acknowledgement.

However, if the second half of the R sequence is a receive ACK only operation, the PHY Layer must filter all incoming packets, looking only for an acknowledge frame, whose sequence number matches the sequence number of the frame transmitted in the T sequence portion. All non-matching frames are discarded, and, after each frame discarded, the sequence manager will continue the R sequence. The only exception is in the active promiscuous mode, when a non-matching received frame is not discarded, but passed through to the MAC Layer instead.

4.1.2 Packet processor filtering

The packet processor parses packets to verify compliance with the 802.15.4 MAC frame format. The Frame Control Field, two octets in length, contains subfields which encode "instructions" on how to parse the remainder of the MHR (MAC Header). The structure of the Frame Control Field is shown in the table below.

Table 3 Frame control field structure

Bits: 0-2	3	4	5	6	7-9	10-11	12-13	14-15
Frame Type	Security Enabled	Frame Pending	Ack Request	PAN ID Compression	Reserved	Dest. Addressing Mode	Frame Version	Source Addressing Mode

The packet processor utilizes the Frame Control Fields subfields in the following manner:

Table 4 Usage of the frame control fields by the packet processor

FCF subfield	Utilization by the packet processor
Frame Type[11]	Interprets the remaining MHR as specific to Beacon, Ack, Data, Command, or Reserved frame types. Each frame type has a unique MHR structure, and different parsing rules apply.
Security Enabled	If Security Enabled=1 and Frame Version>0 (frame versions 2006 and later), an Auxiliary Security Header field will be present in the MHR and will need to be further parsed by the packet processor if this is a MAC Command frame (see 4.1.3 Source Address Matching section below). There is no other use of Security Enabled by the packet processor.
Frame Pending	Ignored by the packet processor.
Ack. Request	Will be stored internally by the sequence manager. An auto-TxAck frame will follow the incoming receive frame, if the necessary conditions are met.
PAN ID Compression	Used for addressing mode rules-checking and addressing field parsing.

[11] For SubGHz PHY, Multipurpose Frame Type (Low Energy Wake-Up Frame) is also allowed. [IEEE 802.15.4g]

Table 4 Usage of the frame control fields by the packet processor

FCF subfield	Utilization by the packet processor
Reserved	Ignored by the packet processor.
Destination Addressing Mode	Used for addressing mode rules-checking and addressing field parsing.
Frame Version	Frame Version is checked against the allowed frame versions. Frame Version is captured by the sequence manager. If an auto-TxAck frame follows the incoming receive frame, the captured Frame Version will be copied into the Frame Control Field of the transmitted frame. If Security Enabled=1 and Frame Version>0 (frame versions 2006 and later), an Auxiliary Security Header field will be present in the MHR, and the MHR must be further parsed by the packet processor, if this is a MAC Command frame (see 4.1.3 Source Address Matching section).
Source Addressing Mode	Used for addressing mode rules-checking and addressing field parsing.

The Sequence Number field is directly following the Frame Control Field. The Sequence Number field of the MHR is captured by the packet processor. If an auto-TxAck frame follows the incoming receive frame, the captured Sequence Number is copied to the transmitted Acknowledge packet, and the captured Frame Version is inserted into the Frame Control Field of the transmitted packet.

The Addressing Fields follow the Sequence Number. The format of the Addressing Fields depends on the Source and Destination Addressing Mode subfields of the Frame Control Field. The Addressing Modes are defined in the following table:

Addressing mode value	Description
0x00	PAN identifier and address fields are not present.
0x01	Reserved.
0x02	Address field contains a 16-bit short address.
0x03	Address field contains a 64-bit extended address.

The packet processor uses the Source and Destination Address Modes shown above to extract the Source PAN ID and Address (if present), the Destination PAN ID, and Address (if present) from the MHR, according to the table below:

Frame type	Destination addressing mode	Source addressing mode	PAN ID compression	Addressing fields
Acknowledge only (reject all other frame types)	0x00	0x00	0x00	None
Data or Command (reject ACK and Beacon frames)	0x02 or 0x03	0x00	0x00	Dest. PAN ID + Dest. Addr.

Frame type	Destination addressing mode	Source addressing mode	PAN ID compression	Addressing fields
Beacon (all devices), or, Data or Command (PAN Coordinator only) (reject ACK frames)	0x00	0x02 or 0x03	0x00	Src. PAN ID + Src. Addr.
Data or Command (reject ACK and Beacon)	0x02 or 0x03	0x02 or 0x03	0x00	Dest. PAN ID + Dest. Addr. + Src. PAN ID + Src. Addr.
Reject all frames	0x00	0x00	0x01	-
Reject all frames	0x02 or 0x03	0x00	0x01	-
Reject all frames	0x00	0x02 or 0x03	0x01	-
Data or Command (reject ACK and Beacon frames)	0x02 or 0x03	0x02 or 0x03	0x01	Dest. PAN ID + Dest. Addr. + Src. PAN ID
Reject all frames	0x01	-	-	-
Reject all frames	-	0x01	-	-

The Source PAN ID and Address, and Destination PAN ID and Address, extracted by the packet processor, are then checked against the gPhyPibPanId_c, gPhyPibShortAddress_c, and gPhyPibLongAddress_c PHY PIBs, depending on the Frame Type of the incoming packet, to determine:

- Is the addressing mode combination valid for this frame type?
- Do the address fields indicate that the packet is indeed addressed to the end device?

4.1.3 Source address matching

The 802.15.4 wireless MAC standard envisions a scenario, where an end device interrogates a coordinator, as to whether the coordinator is storing data (that means a pending message) for the end device. The situation arises in a beacon-enabled network, when a coordinator includes in its transmitted beacon frame the MAC address (long or short) of the end device in its “Pending Address Fields”. The “Pending Address Fields” are a part of the required MAC payload of the beacon frame. The “Pending Address Fields” contain a list of end device addresses for which messages are pending. In this scenario, an end device that finds its address included in the “Pending Address Fields” of the received beacon frame, must respond to the beacon (coordinator) with a MAC Command of “data request” type. Alternatively, in non-beacon-enabled networks, an end device may periodically wake up and “poll” a coordinator to determine whether a message is pending for the end device. In either case, the coordinator stores messages for its end devices in its “indirect queue”. The coordinator must respond to an incoming MAC Command data request (which must have Ack Request=1 in its Frame Control Field), with an Acknowledge frame containing a Frame Pending subfield indicating the presence (or absence) of a message for the requesting end device, in the coordinator’s indirect queue.

The PHY Layer must implement 12 entry checksum buffers, each being 16 bits long. The checksum is calculated in the following manner:

Destination Addressing Mode 2 (short address)	Checksum = (Destination PAN ID + DstAddr[15:0]) % 65536
Destination Addressing Mode 3 (long address)	Checksum = (Destination PAN ID + DstAddr[15:0]) % 65536 Checksum = (Checksum + DstAddr[31:15]) % 65536 Checksum = (Checksum + DstAddr[47:32]) % 65536 Checksum = (Checksum + DstAddr[63:48]) % 65536

The implementation must permit the MAC layer to write and erase entries from this buffer. If an attempt is made to write to a location that is not empty, the request must be denied. The MAC Layer is responsible for calculating the checksum for insertion requests. Checksums stored in the buffer do not have to be contiguous. The PHY Layer must mark unpopulated or erased indexes as unused.

Upon reception of a Data Request frame, the PHY Layer must calculate the checksum for the incoming frame, and perform a search operation through the used locations in the buffer. If a matching sequence is found, an ACK frame with the Frame Pending bit set must be issued, together with a PD-DATA.indication containing the Data Request frame sent to the upper layer, and, otherwise, an ACK frame with the Frame Pending bit set to zero must be issued without any further communication with the MAC Layer. The PHY Layer must not remove any entry from the buffer, unless it is requested by the upper layer. The request to remove an unpopulated index is always successful.

4.2 Inter-layer communication

The PHY sublayer provides two services: the PHY data service, and the PHY management service interfacing to the PHY sublayer management entity (PLME) service access point (SAP) (known as PLME-SAP). The PHY data service enables the transmission and reception of PHY protocol data units (PSDUs) over the media (radio).

The PHY Layer interfaces to the MAC Layer through **function calls** and **function callbacks**.

If the interface primitives are implemented as *function calls*, the MAC Layer calls the exposed functions (provided by the PHY Layer) to issue commands / requests.

If the interface primitives are implemented as *function callbacks*, these are implemented by the MAC Layer, and registered as callbacks by sending their pointers to the PHY Layer through a dedicated function.

4.2.1 Constant macro definitions for 2.4 GHz PHY

The following defines refer to entities used in the 2.4GHz PHY API elements.

4.2.1.1 gMinPHYPacketSize_c

This define is used to limit the minimum number of octets for a packet to be considered valid.

Synopsis:

```
#define gMinPHYPacketSize_c    5
```

4.2.1.2 gMaxPHYPacketSize_c

This define is used to limit the maximum number of octets that the PHY can transmit or receive.

Synopsis:

```
#define gMaxPHYPacketSize_c    127
```

4.2.1.3 gCCATime_c

This define is used to set the CCA duration in symbols.

Synopsis:

```
#define gPhyCCATime_c    8
```

4.2.1.4 gPhyTurnaroundTime_c

This define is used to set the Rx-to-Tx or Tx-to-Rx maximum turnaround time in symbols.

Synopsis:

```
#define gPhyTurnaroundTime_c    12
```

4.2.1.5 gPhySHRDuration_c

This define is used to set the duration of the synchronization header in symbols.

Synopsis:

```
#define gPhySHRDuration_c    10
```

4.2.1.6 gPhySymbolsPerOctet_c

This define is used to set the number of symbols per octet for the current PHY.

Synopsis:

```
#define gPhySymbolsPerOctet_c    2
```

4.2.1.7 gPhyFCSSize_c

This define is used to set the length of the FCS field in bytes.

Synopsis:

```
#define gPhyFCSSize_c    2
```

4.2.1.8 gPhyMaxFrameDuration_c

This define is used to set the maximum number of symbols in a frame.

Synopsis:

```
#define gPhySymbolsPerOctet_c    (gPhySHRDuration_c + (gMaxPHYPacketSize_c + 1) *
gPhySymbolsPerOctet_c)
```

4.2.2 Constant macro definitions for Sub-1 GHz PHY

The following defines refer to entities used in the Sub-1 GHz PHY API elements.

4.2.2.1 gPhyTaskStackSize_c

This define is used to set Stack size in octets for one PHY Task.

NOTE

This parameter must not be changed.

Synopsis:

```
#define gPhyTaskStackSize_c    600
```

4.2.2.2 gPhyTaskPriority_c

This define is used to set the PHY Task priority level in the operating system.

NOTE

The PHY Task must have the highest priority. This parameter must not be changed.

Synopsis:

```
#define gPhyTaskPriority_c    0 /* osPriorityRealtime */
```

4.2.2.3 gPhySymbolsPerOctet_c

This define is used to set the number of symbols per octet for the current PHY.

Synopsis:

```
#define gPhySymbolsPerOctet_c    8
```

4.2.2.4 gPhyMRFSKPHRLength_c

This define is used to set the length of the PHR in octets for the MRFSK PHY.

Synopsis:

```
#define gPhyMRFSKPHRLength_c    2
```

4.2.2.5 gPhyFSKPreambleLength_c

This define is used to set the length of the Preamble in octets for the MRFSK PHY.

Synopsis:

```
#define gPhyFSKPreambleLength_c    16
```

4.2.2.6 gPhyMRFSKSFDLength_c

This define is used to set the length of the SFD in octets for the MRFSK PHY.

Synopsis:

```
#define gPhyMRFSKSFDLength_c    2
```

4.2.2.7 gMinPHYPacketSize_c

This define is used to limit the minimum number of octets for a packet to be considered valid.

Synopsis:

```
#define gMinPHYPacketSize_c    5
```

4.2.2.8 gMaxPHYPacketSize_c

This define is used to limit the maximum number of octets that the PHY can transmit or receive.

Synopsis:

```
#define gMaxPHYPacketSize_c    254
```

4.2.2.9 gCCADurationDefault_c

This define is used to set the default CCA duration (in symbols) used for initialization sequence.

Synopsis:

```
#define gPhyCCADuration_c    13
```

4.2.2.10 gPhySHRDuration_c

This define is used to set the duration of the synchronization header (SHR) (in symbols) for the current PHY.

Synopsis:

```
#define gPhySHRDuration_c    (gPhySymbolsPerOctet_c * (gPhyFSKPreambleLength_c + gPhyMRFSKSFDLength_c))
```

4.2.2.11 gPhyMaxFrameDuration_c

This define is used to set the maximum number of symbols in a frame.

Synopsis:

```
#define gPhyMaxFrameDuration_c    (gPhySHRDuration_c + (gPhyMRFSKPHRLength_c + gMaxPHYPacketSize_c) * gPhySymbolsPerOctet_c)
```

4.2.2.12 Frequency band selection

The following defines are used to set the specific frequency band.

Synopsis:

```
#define gFreq169_400__169_475MHz_c 0 /* Europe 169 */
#define gFreqBand_470__510MHz_d    0 /* China 470 - 576 */
#define gFreqBand_779__787MHz_d    0 /* China 779 - 787 */
#define gFreqBand_863__870MHz_d    0 /* Europe 863 – 870 */
#define gFreqBand_902__928MHz_d    1 /* U.S. 902 – 928 */
#define gFreqBand_920__928MHz_d    0 /* Japan 920 – 928 */
```

NOTE

Select only one frequency band at a time.

4.2.2.13 Frequency band ID

The following defines are used to set the specific frequency band ID for the selected frequency band.

Synopsis:

```
#define gFreqBandId_d gFreq902__928MHz_c
```

4.2.2.14 PHY mode default

The following define is used to configure the default PHY mode.

Synopsis:

```
#define gPhyModeDefault_d gPhyMode1_c
```

4.2.2.15 AFC enabled

The following define is used to enable the Automatic Frequency Correction.

Synopsis:

```
#define gAfcEnabled_d 0
```

4.2.2.16 AFC Rx timeout

The following define is used to configure the receiver timeout when AFC is enabled, and no SFD is received.

Synopsis:

```
#define gAfcRxTimeout_c (gPhySHRDuration_c) /* [symbols] - Min Preamble + SFD +
variable duration */
```

4.2.2.17 RSSI threshold

The following define is used to configure the default receiver sensitivity threshold.

Synopsis:

```
#define gRssiThreshold_c        0xB4 /* -90 dBm */
```

4.2.3 Common constant macro definitions

4.2.3.1 gPhyInstancesCnt_c

This define is used to set the number of supported PHY instances. Currently, only one PHY instance is supported.

Synopsis:

```
#define gPhyInstancesCnt_c    1
```

4.2.3.2 gMaxPhyTimers_c

This define is used to set the maximum number of simultaneous events that can be scheduled in PHY.

Synopsis:

```
#define gMaxPhyTimers_c    5
```

4.2.3.3 gPhyIndirectQueueSize_c

This define is used to set the maximum number of the indirect queue entries.

Synopsis:

```
#define gPhyIndirectQueueSize_c    12
```

4.2.3.4 gPhySeqStartAsap_c

This define is used as a start time to signal that a current sequence is handled as soon as possible by the PHY layer.

Synopsis:

```
#define gPhySeqStartAsap_c    ((phyTime_t)(-1))
```

4.3 Data type definition / SAP type definitions

The PHY provides two services, accessed through two SAPs: the PHY data service, accessed through the PHY data SAP (PD-SAP), and the PHY management service, accessed through the PLME-SAP.

4.3.1 Common data types definitions

The following data types are used in the MAC-PHY interface. Members of the structures that define the payload of the service and callback functions, described later in detail, use these data types.

4.3.1.1 phyStatus_t

This type enumerates all the possible statuses of primitives that require passing a status to the MAC Layer.

Synopsis:

```
typedef enum
{
    gPhyChannelBusy_c = 0x00,
    gPhyBusyRx_c = 0x01,
    gPhyBusyTx_c = 0x02,
    gPhyChannelIdle_c = 0x04,
```

```
    gPhyInvalidParameter_c = 0x05,
    gPhyRxOn_c = 0x06,
    gPhySuccess_c = 0x07,
    gPhyTRxOff_c = 0x08,
    gPhyTxOn_c = 0x09
    gPhyUnsupportedAttribute_c = 0x0A,
    gPhyReadOnly_c = 0x0B,
    gPhyIndexUsed_c = 0x11,
    gPhyNoAck_c = 0x14,
    gPhyFramePending_c = 0x15,
    gPhyBusy_c = 0xF1,
    gPhyInvalidPrimitive_c = 0xF2
}phyStatus_t;
```

Member	Value	Description
gPhyChannelBusy_c	0x00	The CCA attempt has detected a busy channel.
gPhyBusyRx_c	0x01	The transceiver is asked to change its state while receiving.
gPhyBusyTx_c	0x02	The transceiver is asked to change its state while transmitting.
gPhyChannelIdle_c	0x04	The CCA attempt has detected an idle channel.
gPhyInvalidParameter_c	0x05	A SET request was issued with a parameter in the primitive that is out of the valid range.
gPhyRxOn_c	0x06	The transceiver is in the receiver enabled state.
gPhySuccess_c	0x07	A SET/GET, an ED operation, a data request, an indirect queue insert, or a transceiver state change was successful.
gPhyTRxOff_c	0x08	The transceiver is in the transceiver disabled state.
gPhyTxOn_c	0x09	The transceiver is in the transmitter enabled state.
gPhyUnsupportedAttribute_c	0x0A	A SET/GET request was issued with the identifier of an attribute that is not supported.
gPhyReadOnly_c	0x0B	A SET request was issued with the identifier of an attribute that is read-only.
gPhyIndexUsed_c	0x11	The indirect queue insert operation has detected an used index.
gPhyNoAck_c	0x14	No ACK was received for the last transmission.
gPhyFramePending_c	0x15	The ACK of a Data Request frame indicates a pending frame in the coordinator's indirect TX queue.
gPhyBusy_c	0xF1	The current request cannot be handled, because the PHY is busy.
gPhyInvalidPrimitive_c	0xF2	The set was not completed, because the primitive is not in the valid range.

4.3.1.2 phySlottedTx_t12

This type enumerates the possible transmission modes in respect to slotted or unslotted modes.

Used by:

PdDataReq()

PlmeSetTRxStateReq()

[12] Not supported by SubGHz PHY.

Synopsis:

```
typedef enum
{
    gPhySlottedTx_c = 0x0c,
    gPhyUnslottedTx_c = 0x0d
}phySlottedTx_t;
```

Member	Value	Description
gPhySlottedTx _c	0x0c	The TX operation must be performed in slotted mode.
gPhyUnslottedTx_c	0x0d	The TX operation must be performed in unslotted mode.

4.3.1.3 phyCCAType_t

This type is used to indicate, whether the CCA operations are required before the transmissions, and together with phySlottedTx_t to determine, whether there are more than one needed.

Used by:

PhyPdDataRequest()

PhyPlmeCcaEdRequest()

Synopsis:

```
typedef enum
{
    gPhyEnergyDetectMode_c = 0x00,
    gPhyCCAMode1_c = 0x01,
    gPhyCCAMode2_c = 0x02,
    gPhyCCAMode3_c = 0x03,
    gPhyNoCCABeforeTx_c = 0x04
}phyCCAType_t;
```

Member	Value	Description
gPhyEnergyDetectMode_c	0x00	Energy Detect must be performed.
gPhyCCAMode1_c	0x01	CCA Mode 1 must be performed before the TX operation.
gPhyCCAMode2_c	0x02	CCA Mode 2 must be performed before the TX operation.
gPhyCCAMode3_c	0x03	CCA Mode 3 must be performed before the TX operation.
gPhyNoCCABeforeTx_c	0x04	No CCA must be performed before the TX operation.

4.3.1.4 phyContCCAMode_t[13]

This type is used to indicate, whether a Continuous CCA operation is required.

Used by:

PhyPlmeCcaEdRequest()

Synopsis:

```
typedef enum
{
    gPhyEnergyDetectMode_c = 0x00,
    gPhyCCAMode1_c = 0x01,
    gPhyCCAMode2_c = 0x02,
    gPhyCCAMode3_c = 0x03,
    gPhyNoCCABeforeTx_c = 0x04
}phyCCAType_t;
```

Member	Value	Description
gPhyEnergyDetectMode_c	0x00	Energy Detect must be performed.
gPhyCCAMode1_c	0x01	CCA Mode 1 must be performed before the TX operation.
gPhyCCAMode2_c	0x02	CCA Mode 2 must be performed before the TX operation.
gPhyCCAMode3_c	0x03	CCA Mode 3 must be performed before the TX operation.
gPhyNoCCABeforeTx_c	0x04	No CCA must be performed before the TX operation.

4.3.1.5 phyState_t

This type is used to enumerate the possible states to set the transceiver to. Setting the transceiver into any Tx state is done by issuing pdDataReq_t that does not use this type for any member.

Used by:

PlmeSetTRxStateReq()

Synopsis:

```
typedef enum
{
    gPhyForceTRxOff_c = 0x03,
    gPhySetRxOn_c = 0x12,
    gPhySetTRxOff_c = 0x13,
}phyState_t;
```

Member	Value	Description
gPhyForceTRxOff_c	0x03	The transceiver is to be switched off immediately.
gPhySetRxOn_c	0x12	The transceiver is to be configured into the receiver-enabled state.
gPhySetTRxOff_c	0x13	The transceiver is to be configured into the transceiver-disabled state.

[13] Not supported by SubGHz PHY.

4.3.1.6 phyAckRequired_t

This type is used to filter the next received frames, and to accept only Ack frames.

Used by:

PhyPdDataRequest()

Synopsis:

```
typedef enum
{
    gPhyRxAckRqd_c    = 0x00,
    gPhyNoAckRqd_c    = 0x01,
    gPhyEnhancedAckReq_c = 0x02
}phyTimeStatus_t;
```

Member	Value	Description
gPhyRxAckRqd_c	0x00	A receive Ack frame is expected to follow the transmit frame.
gPhyNoAckRqd_c	0x01	An ordinary receive frame follows the transmit frame.
gPhyEnhancedAckReq_c[14]	0x02	A receive Enhanced Ack frame is expected to follow the transmit frame.

4.3.1.7 phyPibId_t

This type enumerates all PHY PIB IDs. The PIBs can be read and written to by the upper layer, and are used to configure certain parameters and modes of operation for the PHY Layer. The PIBs that refer to physical parameters, such as carrier frequency or transmission power, are usually mirrored in hardware, but in case the IEEE 802.15.4 hardware acceleration is used, there are protocol-oriented PIBs also stored in hardware, such as addresses, PAN ID, promiscuous mode setting, and so on.

Used by:

PlmeSetPIBRequest()

PlmeGetPIBRequest()

Synopsis:

```
typedef enum
{
    gPhyPibCurrentChannel_c  = 0x00,
    gPhyPibCurrentPage_c     = 0x01,
    gPhyPibTransmitPower_c   = 0x02,
    gPhyPibLongAddress_c     = 0x03,
    gPhyPibShortAddress_c    = 0x04,
    gPhyPibPanId_c           = 0x05,
    gPhyPibPanCoordinator_c  = 0x06,
    gPhyPibSrcAddrEnable_c   = 0x07,
    gPhyPibPromiscuousMode_c = 0x08,
    gPhyPibAutoAckEnable_c   = 0x09,
```

[14] only for SubGHz Phy.

```
    gPhyPibFrameVersion_c    = 0x0A,
    gPhyPibFrameEnable_c     = 0x0B,
    gPhyPibAckFramePending_c = 0x0C,
    gPhyPibRxOnWhenIdle      = 0x0D,
    gPhyPibFrameWaitTime_c   = 0x0E,
    gPhyPibDeferTxIfRxBusy_c = 0x0F,
    gPhyPibLastTxAckFP_c     = 0x10,
    gPhyPibCurrentMode_c            = 0x20,
    gPhyPibFSKPreambleRepetitions_c = 0x21,
    gPhyPibFSKScramblePSDU_c        = 0x22,
    gPhyPibCCADuration_c            = 0x23,
    gPhyPibCSLRxEnabled_c           = 0x24,
    gPhyPibFreqBandId_c             = 0x25,
    gPhyPibAckWaitDuration_c        = 0x26,
    gPhyPibTschEnabled_c            = 0x27,
    gPhyPibTschTimeslotInfo_c       = 0x28
}phyPibId_t;
```

Member	Value	Description
gPhyPibCurrentChannel_c	0x00	The channel currently used.
gPhyPibCurrentPage_c	0x01	The channel page currently used.
gPhyPibTransmitPower_c	0x02	The power used for TX operations.
gPhyPibLongAddress_c	0x03	The MAC long address to be used by the PHY's source address matching feature.
gPhyPibShortAddress_c	0x04	The MAC short address to be used by the PHY's source address matching feature.
gPhyPibPanId_c	0x05	The MAC PAN ID to be used by the PHY's source address matching feature.
gPhyPibPanCoordinator_c	0x06	Indicates, whether the device is a PAN coordinator or not.
gPhyPibSrcAddrEnable_c	0x07	Enables or disables the PHY's source address matching feature.
gPhyPibPromiscuousMode_c	0x08	Selects between normal, promiscuous, and active promiscuous mode.
gPhyPibAutoAckEnable_c	0x09	Enables or disables automatic transmission of ACK frames.
gPhyPibFrameVersion_c	0x0A	Used in checking for allowed frame versions (0x00 – any version accepted, 0x01 – accept Frame Version 0 packets, 0x02 – accept Frame Version 1 packets, 0x03 – accept Frame Version 0 and 1 packets.
gPhyPibFrameEnable_c	0x0B	Used for enabling or disabling the reception of MAC frames.
gPhyPicAckFramePending_c	0x0C	Used to copy its contents to the outgoing ACK frame's Frame Pending field, as a response to a received Data Request frame with Source Address Mathing disabled.
gPhyPibRxOnWhenIdle_c	0x0D	Enable RX when the radio is IDLE.
gPhyPibFrameWaitTime_c	0x0E	The number of symbols that the Rx should be on after receiving an ACK with FP=1.
gPhyPibDeferTxIfRxBusy_c	0x0F	If a packet is being received, transmission is deffered.
gPhyPibLastTxAckFP_c	0x10	Indicates, whether the last ACK sent to the MAC Data Request had the frame pending flag set.
gPhyPibCurrentMode_c	0x20	Used to set or get the current operating PHY mode. Only for Sub-1 GHz PHY.
gPhyPibFSKPreambleRepetitions_c	0x21	Used to set or get the number of 1 octet patterns in the preamble. Only for Sub-1 GHz PHY.
gPhyPibFSKScramblePSDU_c	0x22	Enables or disables the data whitening feature. Only for Sub-1 GHz PHY.

Member	Value	Description
gPhyPibCCADuration_c	0x23	Set or get the CCA duration specified in symbols. The valid range is 8 – 1000. Only for Sub-1 GHz PHY.
gPhyCSLRxEnabled_c	0x24	Enable or disable the CSL mode for the Rx sequences.
gPhyPibFreqBandId_c	0x25	Sub-1 GHz Frequency Band Identifier.
gPhyPibAckWaitDuration_c	0x26	ACK receiver on period.
gPhyPibTschEnabled_c	0x27	Enable or disable TSCH.
gPhyPibTschTimeslotInfo_c	0x28	The TSCH timings inside a timeslot.

4.3.2 Sub-1 GHz specific enumerations definition

The following enumerations define the data types used in the MAC-PHY interface. Members of the structures that define the payload of the service and callback functions (described later in detail) use these data types.

4.3.2.1 phyMode_t

This type is used to enumerate all the PHY modes available. The available PHY modes depend on the selected Frequency Band (See *0 – IEEE Standard for Local and metropolitan area networks*).

Used by:

PhyPib_SetCurrentPhyMode()

Synopsis:

```
typedef enum
{
    gPhyMode1_c    = 0x00,
    gPhyMode2_c    = 0x01,
    gPhyMode3_c    = 0x02,
    gPhyMode4_c    = 0x03,
    gPhyMode1ARIB_c    = 0x04,
    gPhyMode2ARIB_c    = 0x05,
    gPhyMode3ARIB_c    = 0x06
}phyMode_t;
```

Member	Value	Description
gPhyMode1_c	0x00	Set PHY Mode 1.
gPhyMode2_c	0x01	Set PHY Mode 2.
gPhyMode3_c	0x02	Set PHY Mode 3.
gPhyMode4_c	0x03	Set PHY Mode 4.
gPhyMode1ARIB_c	0x04	Set PHY Mode 1 for ARIB standard. 920 – 928 MHz only.
gPhyMode2ARIB_c	0x05	Set PHY Mode 2 for ARIB standard. 920 – 928 MHz only.
gPhyMode3ARIB_c	0x06	Set PHY Mode 3 for ARIB standard. 920 – 928 MHz only.

4.3.2.2 phyFreqBand_t

This type is used to enumerate all the Frequency Band Ids available.

Synopsis:

```
typedef enum
{
  gFreq169_400__169_475MHz_c = 0x00 // 169  (Europe)
  gFreq470__510MHz_c = 0x02,    // 470-510   (China)
  gFreq779__787MHz_c = 0x03,    // 779-787   (China)
  gFreq863__870MHz_c = 0x04,    // 863-870   (Europe)
  gFreq902__928MHz_c = 0x07,    // 902-928   (U.S.)
  gFreq920__928MHz_c = 0x09,    // 920-928   (Japan) - Includes ARIB modes
}phyFreqBand_t;
```

Member	Value	Description
gFreq169_400__169_475MHz_c	0x00	Europe 169 Frequency Band Id
gFreq470__510MHz_c	0x02	China Frequency Band Id
gFreq779__787MHz_c	0x03	China Frequency Band Id
gFreq863__870MHz_c	0x04	Europe Frequency Band Id
gFreq902__928MHz_c	0x07	US Frequency Band Id
gFreq920__928MHz_c	0x09	Japan Frequency Band Id

4.3.3 PD SAP type definitions

The PHY data service is accessed through the PHY data SAP (PD-SAP). The PD-SAP supports the transport of MAC Protocol Data Units (MPDUs) between peer MAC sublayer entities.

These PD-SAP primitives are listed below.

4.3.3.1 macToPdDatamessage_t

This is a message sent by the MAC layer, containing the data request for the PHY layer.

Synopsis:

```
typedef struct macToPdDataMessage_tag
{
    phyMessageId_t             msgType;
    uint8_t                    macInstance;
    union
    {
        pdDataReq_t            dataReq;
        pdIndQueueInsertReq_t  indQueueInsertReq;
        pdIndQueueRemoveReq_t  indQueueRemoveReq;
    } msgData;
} macToPdDataMessage_t;
```

Direction: MAC → PHY

Member	Description	Value
msgType	The requested operation sent by the MAC layer.	phyMessageId_t
	gPdIndQueueInsertReq_c	
	gPdIndQueueRemove_c	
	gPdDataReq_c	
macInstance	Id of the MAC instance to be serviced.	uint8_t
dataReq	Descriptor of the PD-DATA.Request primitive.	pdDataReq_t
indQueueInsertReq	Descriptor of the PD-INDQUEUE INSERT.Request primitive.	pdIndQueueInsert_t
indQueueRemoveReq	Descriptor of the PD-INDQUEUE REMOVE.Request primitive.	pdIndQueueRemove_t

4.3.3.2 pdDataReq_t

The PD-DATA.request primitive is generated by the MAC Layer when a MAC data frame (MPDU) is ready to be transferred to the PHY Layer, becoming payload for the PHY frame (PSDU). Upon the reception of this primitive, the PHY Layer will arm either a T or TRxAck sequence, depending on the Acknowledgment Request subfield bit included in the Frame Control field, which is part of the MHR.

The MAC Layer must also provide information about performing the CCA operations before transmission:

Table 5 CCA operations before transmission

CCABeforeTx	slottedTx	Number of CCA measurements performed by PHY
0	X	0
1	0	1
1	1	2
2	0	Continuous CCA
2	1	reserved

For a continuous CCA, the PHY Layer must first arm a CCCA sequence, and after it completes, it must arm a T sequence without any further CCA operations.

The psduLength parameter represents the number of octets contained in the PSDU to be transmitted by the PHY Layer without the last two octets containing the FCS field. The PHY Layer calculates the CRC of the MAC frame, and then populates the FCS field.

Synopsis:

```
typedef struct pdDataReq_tag
{
    uint32_t                startTime;
    uint32_t                txDuration;
    phySlottedMode_t        slottedTx;
    phyCCAType_t            CCABeforeTx;
    phyAckRequired_t        ackRequired;
    uint8_t                 psduLength;
    phyPHR_t                phyHeader;              /* SubGhz Phy only */
```

```
    uint8_t                 macDataIndex;           /* SubGHz Phy only */
    uint8_t                 fillFifoBlockLength;    /* SubGHz Phy only */
    uint8_t*                pPsdu;
} pdDataReq_t;
```

Direction: MAC → PHY

Member	Description	Value
startTime	The start time of the Data Request sequence. A value of gPhySeqStartAsap_c to start immediately.	uint32_t
txDuration	The computed duration for the Data Request frame.	uint32_t
slottedTx	Indicates, whether the slotted mode is used for this transmission.	phySlottedTx_t
	gPhySlottedTx _c	0x0c
	gPhyUnslottedTx_c	0x0d
CCABeforeTx	Indicates, whether the CCA is used before this transmission.	phyCCAType_t
	gPhyCCAMode1_c	0x01
	gPhyCCAMode2_c	0x02
	gPhyNoCCABeforeTx_c	0x03
ackRequired	Indicates, whether the Ack is required for this transmission.	phyAckRequired_t
	gPhyRxAckRqd_c	0x01
	gPhyNoAckRqd_c	0x02
	gPhyEnahncedAckReq_c – Sub-1GHz Phy only	0x03
psduLength	The number of octets contained in the PSDU to be transmitted by the PHY Layer, without the last two octets containing the FCS field.	uint8_t
phyHeader	Sub-1 GHz PHY only. Used to form the PHY header before sending.	phyPHR_t
macDataIndex	Sub-1 GHz PHY only. Used to store the index of the currently sent byte.	uint8_t
fillFifoBlockLength	Sub-1 GHz PHY only. The block length to be prefilled in transceiver's fifo.	uint8_t
pPsdu	A pointer to the set of octets forming the PSDU to be transmitted by the PHY Layer.	uint8_t*

4.3.3.3 pdIndQueueInsertReq_t

The PD-INDQUEUEINSERT.Request primitive is generated by the MAC Layer when a packet is inserted into the MAC indirect queue. A 16-bit checksum derived from the Destination Address and Destination PAN ID is passed to the PHY Layer. The PHY Layer, both in hardware implementations and emulated in software, must keep a 12-entry database of checksums, and facilitate writing into it through the use of this primitive.

Synopsis:

```
typedef struct pdIndQueueInsertReq_tag
{
    uint8_t             index;
    uint16_t            checksum;
} pdIndQueueInsert_t;
```

Direction: MAC → PHY

Member	Description	Value
index	The index where the checksum is to be inserted. Accepted values are 0x00 – 0x0b.	uint8_t
checksum	The calculated checksum used for indirect transmissions.	uint16_t

4.3.3.4 pdIndQueueRemoveReq_t

The PD-INDQUEUEREMOVE.Request primitive is generated by the MAC Layer when a packet is removed from the MAC indirect queue, and the index, at which the packet's checksum is stored, gets passed on to the PHY Layer. The PHY Layer, both in hardware implementations and emulated in software, must facilitate erasing entries from its database of checksums through the use of this primitive.

Synopsis:

```
typedef struct pdIndQueueRemoveReq_tag
{
    uint8_t            index;
} pdIndQueueRemove_t;
```

Direction: MAC → PHY

Member	Description	Value
index	The index where the checksum is to be inserted. Accepted values are 0x00 – 0x0b.	uint8_t

4.3.3.5 phyPHR_t

Used internally by the Sub-1 GHz PHY to store the PHY Header before sending it to the transceiver.

Synopsis:

```
typedef struct phyPHR_tag[15]
{
    union{
        uint16_t mask;
        uint8_t  byteAccess[2];
        struct{
            uint16_t     modeSwitch          :1;
            uint16_t     reserved            :2;
            uint16_t     fcsType             :1;
            uint16_t     dataWhitening       :1;
            uint16_t     frameLength         :11;
        };
    };
} phyPHR_t;
```

[15] Only for SubGHz PHY.

Direction: Sub-1 GHz PHY Internal

Member	Description	Value
modeSwitch	Set to 1 to indicate that a mode switch shall occur.	1 bit
reserved	Set to 0.	2 bits
fcsType	Set to 0 corresponds to a 4-octet FCS.	1 bit
dataWhitening	Set to 1 when data whitening is used.	1 bit
frameLength	PSDU length of the packet.	11 bits

4.3.3.6 phyTxParams_t

Passed by the MAC layer to specify that a standalone CCA must be used, or that the ACK is required for the sequence.

Synopsis:

```
typedef struct phyTxParams_tag
{
    bool_t        useStandaloneCcaBeforeTx;
    uint8_t       numOfCca;
    phyAckRequired_t      ackRequired;
} phyTxParams_t;
```

Direction: MAC → PHY

Member	Description	Value
useStandaloneCcaBeforeTx	Set to 1 to send a CCA or ED request before the Tx sequence.	bool_t
numOfCca	The number of CCA samples to be taken.	uint8_t
ackRequired	Indicates, whether an Ack is required for this transmission.	phyAckRequired_t

4.3.3.7 pdDataToMacMessage_t

Used by the PHY State Machine to send the PD-DATA.indication and PD-DATA.confirm messages to the MAC Layer.

Synopsis:

```
typedef struct pdDataToMacMessage_tag
{
    phyMessageId_t             msgType;
    uint8_t                    macInstance;
    union
    {
        pdDataCnf_t             dataCnf;
        pdDataInd_t             dataInd;
        pdIndQueueInsertCnf_t   indQueueInsertCnf;
    }msgData;
} pdDataToMacMessage_t;
```

Direction: PHY → MAC

Member	Description	Value
msgType	The requested operation, sent by the MAC layer.	phyMessageId_t
	gPdDataInd_c	
	gPdDataCnf_c	
macInstance	Id of the MAC instance to be serviced.	uint8_t
dataCnf	Descriptor of the PD-DATA.Confirm primitive.	pdDataCnf_t
dataInd	Descriptor of the PD-DATA.Indication primitive.	pdDataInd_t
indQueueInsertCnf	Descriptor of the PD-INDQUEUE INSERT.Confirm primitive.	pdIndQueueInsertCnf_t

4.3.3.8 pdDataCnf_t

The PD-DATA.confirm primitive reports the result of a request to transfer a data MAC frame (MPDU). The status returned by PD-DATA.confirm can be SUCCESS, indicating that the request to transmit was successful, an error code of BUSY if the PHY Layer was not in the idle state (I sequence) when the PD-DATA.request was issued, or an error code of CHANNEL_BUSY if all CCA sequences indicated that the channel was busy. If the transmission occurred successfully but no valid ACK frame was received (assuming that it was requested), then an error code of NO_ACK is used.

Synopsis:

```
typedef struct pdDataCnf_tag
{
    phyStatus_t            status;
} pdDataCnf_t;
```

Direction: PHY → MAC

Member	Description	Value
status	The result of the request to transmit a packet.	phyStatus_t
	gPhySuccess_c	
	gPhyBusy_c	
	gPhyChannelBusy_c	
	gPhyNoAck_c	

4.3.3.9 pdDataInd_t

The PD-DATA.indication primitive is generated by the PHY Layer when an MPDU is ready to be transferred to the MAC Layer. Besides the PSDU itself, the primitive also returns the LQI value measured during the reception.

Synopsis:

```
typedef struct pdDataInd_tag
{
    uint32_t               timeStamp;
    uint8_t                ppduLinkQuality;
    uint8_t                psduLength;
    uint8_t *              pPsdu;
} pdDataInd_t;
```

Direction: PHY → MAC

Member	Description	Value
timeStamp	The timestamp when the reception started.	uint32_t
ppduLinkQuality	Link quality (LQI) value measured during the reception of the PPDU.	uint8_t
psduLength	The number of octets contained in the PSDU received by the PHY Layer.	uint8_t
pPsdu	The pointer to the set of octets forming the PSDU received by the PHY Layer.	uint8_t*

4.3.3.10 pdQueueInsertCnf_t

The PD-INDQUEUEINSERT.confirm primitive reports the result of PD-INDQUEUEINSERT request, which can be SUCCESS for a successful request, or an error code of INDEX_USED if the index at which the request was made is not free.

Synopsis:

```
typedef struct pdIndQueueInsertCnf_tag
{
    phyStatus_t            status;
} pdIndQueueInsertCnf _t;
```

Direction: PHY → MAC

Member	Description	Value
status	The result of the request to insert a checksum in the source address matching vector.	phyStatus_t
	gPhySuccess_c	0x07
	gPhyIndexUsed_c	0x10

4.3.4 PLME SAP type definitions

The PHY management service is accessed through the PHY Layer Management Entity SAP (PLME-SAP). The PLME-SAP enables the transport of management commands between the MLME and the PLME.

These PLME-SAP primitives are listed below.

4.3.4.1 macToPlmeMessage_t

Used by the MAC layer to send commands to the PHY layer.

Synopsis:

```
typedef struct macToPlmeMessage_tag
{
    phyMessageId_t             msgType;
    uint8_t                    macInstance;
    union
    {
        plmeEdReq_t                edReq;
        plmeCcaReq_t               ccaReq;
        plmeSetTRxStateReq_t       setTRxStateReq;
        plmeSetReq_t               setReq;
```

```
        plmeGetReq_t                getReq;
    }msgData;
} macToPlmeMessage_t;
```

Direction: MAC → PHY

Member	Description	Value
msgType	The command sent by the MAC layer.	phyMessageId_t
	gPlmeEdReq_c	
	gPlmeCcaReq_c	
	gPlmeSetReq_c	
	gPlmeGetReq_c	
	gPlmeSetTRxStateReq_c	
macInstance	Id of the MAC instance to be serviced.	uint8_t
edReq	Descriptor of the PLME-ED.Request primitive.	plmeEdReq_t
ccaReq	Descriptor of the PLME-CCA.Request primitive.	plmeCcaReq_t
setTRxStateReq	Descriptor of the PLME-SET-TRX-STATE.Request primitive.	plmeSetTRxStateReq_t
setReq	Descriptor of the PLME-SET.Request primitive.	plmeSetReq_t
getReq	Descriptor of the PLME-Get.Request primitive.	plmeGetReq_t

4.3.4.2 plmeEdReq_t

The PLME-ED.request primitive is generated by the MAC Layer when an ED measurement must be performed by the PHY Layer, which then arms a C sequence. The ED request primitive has no parameters.

In this case the message payload is NULL.

Synopsis:

```
typedef struct plmeEdReq_tag
{
    uint32_t             startTime;
} plmeEdReq_t;
```

Direction: MAC → PHY

Member	Description	Value
startTime	Start time for the ED request.	uint32_t

4.3.4.3 plmeCCAReq_t

The PLME-CCA.request primitive is generated by the MAC Layer when a CCA operation must be performed, and it is passed to the PHY Layer, which then arms a C sequence. The CCA request primitive has no parameters.

In this case the message payload is NULL.

Synopsis:

```
typedef struct plmeCcaReq_tag
{
    phyCCAType_t                ccaType;
    phyContCCAMode_t            contCcaMode;
} plmeCcaReq_t;
```

Direction: MAC → PHY

Member	Description	Value
ccaType	The type of the CCA requested by MAC.	phyCCAType_t
contCcaMode	The requested mode for the Continuous CCA Scan.	phyContCCAMode_t

4.3.4.4 plmeSetTRxStateReq_t

The PLME-SET-TRX-STATE.request primitive is generated by the MAC Layer when the transceiver state must be changed by the PHY Layer, which then arms either an I or R sequence. This primitive is also used to cancel any ongoing sequence by setting the state to FORCE_TRX_OFF. If this primitive is issued with an RX_ON or TRX_OFF argument and the PHY is busy transmitting a PPDU, a state change will occur at the end of the transmission. If this primitive is issued with TRX_OFF, and the PHY is in the RX_ON state and has already received a valid SFD, the state change will occur at the end of reception of the PPDU.

The slottedTx parameter is used by the PHY Layer during R sequence to determine whether the ensuing transmit acknowledge frame (if any) must be synchronized to a backoff slot boundary.

Arming the T or TR sequence is done exclusively using the pdDataReq_t primitive.

Synopsis:

```
typedef struct plmeSetTRxStateReq_tag
{
    phyState_t            state;
    phySlottedMode_t      slottedMode;
    uint32_t              startTime;
    uint32_t              rxDuration;
} plmeSetTRxStateReq_t;
```

Direction: MAC → PHY

Member	Description	Value
state	The new state in which to configure the transceiver.	phyState_t
phySlottedMode_t	Indicates, whether the slotted mode is used for this transmission.	phySlottedMode_t
startTime	The start time when the state change is scheduled to occur.	uint32_t
rxDuration	If the requested state is Rx, then Rx will be enabled for rxDuration symbols.	uint32_t

4.3.4.5 plmeSetReq_t

The PLME-SET.request primitive is generated by the MAC Layer to modify a PIB attribute in the PHY Layer. This primitive requires the identifier of the PIB attribute to set its value.

Synopsis:

```
typedef struct plmeSetReq_tag
{
    phyPibId_t              PibAttribute;
    uint64_t                PibAttributeValue;
} plmeSetReq_t;
```

Direction: MAC → PHY

Member	Description	Value
PibAttribute	The identifier of the PIB attribute to set.	phyPibId_t
PibAttributeValue	The value of the indicated PIB attribute to set.	uint64_t

4.3.4.6 plmeGetReq_t

The PLME-GET.request primitive is generated by the MAC Layer to request information about a PIB attribute in the PHY Layer. This primitive requires the identifier of the PIB attribute to read.

Synopsis:

```
typedef struct plmeGetReq_tag
{
    phyPibId_t              PibAttribute;
    uint64_t *              pPibAttributeValue;
} plmeGetReq_t;
```

Direction: MAC → PHY

Member	Description	Value
PibAttribute	The identifier of the PIB attribute to get.	phyPibId_t
PibAttributeValue	The value of the indicated PIB attribute to get.	uint64_t*

4.3.4.7 plmeEdCnf_t

The PLME-ED.confirm primitive is generated by the PHY Layer after the C sequence completes, and returns the response of a previous PLME-ED.request to the MAC Layer. The status returned can be SUCCESS if the measurement was successful, an error code of TX_ON if there is an ongoing T sequence, or RX_ON if the transceiver is receiving. The PLME-ED.confirm primitive returns the value of the ED measurement.

Synopsis:

```
typedef struct plmeEdCnf_tag
{
    phyStatus_t          status;
    uint8_t              energyLevel;
    uint8_t              energyLeveldB;
} plmeEdCnf_t;
```

Direction: PHY → MAC

Member	Description	Value
status	The result of the request to perform an ED measurement.	phyStatus_t
	gPhySuccess_c	0x07
	gPhyTxOn_c	0x09
	gPhyRxOn_c	0x08
energyLevel	The ED level for the current channel. If the status is not SUCCESS, the value of this parameter will be ignored.	uint8_t
energyLeveldB	The ED level for the current channel in dBm value.	uint8_t

4.3.4.8 plmeCcaCnf_t

The PLME-CCA.confirm primitive is generated by the PHY Layer after the C sequence completes, and returns the response of a previous PLME-CCA.request to the MAC Layer. The status returned can be IDLE if the channel is idle, RX_ON if the transceiver is receiving, or BUSY if there is an ongoing T sequence or the channel assessment process determined that the channel is busy.

Synopsis:

```
typedef struct plmeCcaCnf_tag
{
    phyStatus_t             status;
} plmeCcaCnf_t;
```

Direction: PHY → MAC

Member	Description	Value
status	The result of the request to perform a CCA.	phyStatus_t
	gPhyChannelIdle_c	0x04
	gPhyChannelBusy_c	0x00

4.3.4.9 plmeSetTRxStateCnf_t

The PLME-SET-TRX-STATE.confirm primitive is generated by the PHY Layer, and issued to the MAC Layer after attempting to change the operating state of the transceiver. After the request, if a state change occurs, a status of `gPhySuccess_c` is returned. Otherwise, a status describing the current state is issued (TRX_OFF or RX_ON).

Synopsis:

```
typedef struct plmeSetTRxStateCnf_tag
{
    phyStatus_t             status;
} plmeSetTRxStateCnf_t;
```

Direction: PHY → MAC

Member	Description	Value
status	The result of a request to change the state of the transceiver.	phyStatus_t
	gPhySuccess_c	0x07
	gPhyRxOn_c	0x06
	gPhyTRxOff_c	0x08

4.3.4.10 plmeSetCnf_t

The PLME-SET.confirm primitive is generated by the PHY Layer to report an attempt to modify a PHY PIB attribute to the MAC Layer. The primitive returns the PIB attribute identifier and the status, which can be UNSUPPORTED_ATTRIBUTE if the attribute is not supported, READ_ONLY if the attribute is not writable, INVALID_PARAMETER if the value is out of range for this specific attribute, or SUCCESS if the attribute was successfully written.

Synopsis:

```
typedef struct plmeSetCnf_tag
{
    phyStatus_t            status;
    phyPibId_t             PibAttribute;
} plmeSetCnf_t;
```

Direction: PHY → MAC

Member	Description	Value
status	The status of the attempt to set the requested PIB attribute.	phyStatus_t
	gPhySuccess_c	0x07
	gPhyUnsupportedAttribute_c	0x0a
	gPhyReadOnly_c	0x0b
	gPhyInvalidParameter_c	0x05
PIBAttribute	The identifier of the PIB attribute being confirmed.	phyPibId_t

4.3.4.11 plmeGetCnf_t

The PLME-GET.confirm primitive is generated by the PHY Layer to report the results of an information request from the PHY PIB to the MAC Layer. The primitive returns the PIB attribute identifier, its value, and the status, which can be UNSUPPORTED_ATTRIBUTE if the attribute is not supported or SUCCESS if the attribute was successfully retrieved.

Synopsis:

```
typedef struct plmeSetCnf_tag
{
    phyStatus_t            status;
    phyPibId_t             PibAttribute;
    uint64_t               PibAttributeValue;
} plmeSetCnf_t;
```

Direction: PHY → MAC

Member	Description	Value
status	The result of the attempt to get the requested PIB attribute information.	phyStatus_t
	gPhySuccess_c	0x07
	gPhyUnsupportedAttribute_c	0x0a
PIBAttribute	The identifier of the PIB attribute that was requested.	phyPibId_t
PIBAttributeValue	The value of the indicated PHY PIB attribute that was requested. This parameter has zero length when the status parameter is set to UNSUPPORTED_ATTRIBUTE.	uint64_t

4.3.4.12 plmeToMacMessage_t

Used by the PHY State Machine to send the PLME.confirm messages to the MAC Layer.

Synopsis:

```
typedef struct plmeToMacMessage_tag
{
    phyMessageId_t             msgType;
    uint8_t                    macInstance;
    union
    {
        plmeCcaCnf_t            ccaCnf;
        plmeEdCnf_t             edCnf;
        plmeSetTRxStateCnf_t    setTRxStateCnf;
        plmeSetCnf_t            setCnf;
        plmeGetCnf_t            getCnf;
    }msgData;
} plmeToMacMessage_t;
```

Direction: MAC → PHY

Member	Description	Value
msgType	The command sent by the MAC layer.	phyMessageId_t
	gPlmeEdReq_c	
	gPlmeCcaReq_c	
	gPlmeSetReq_c	
	gPlmeGetReq_c	
	gPlmeSetTRxStateReq_c	
macInstance	Id of the MAC instance to be serviced.	uint8_t
ccaCnf	Descriptor of the PLME-CCA.Confirm primitive.	plmeCcaCnf_t
edCnf	Descriptor of the PLME-Ed.Confirmt primitive.	plmeEdCnf_t
setTRxStateCnf	Descriptor of the PLME-SET-TRX-STATE.Confirm primitive.	plmeSetTRxStateCnf_t
setCnf	Descriptor of the PLME-SET.Confirm primitive.	plmeSetCnf_t
getCnf	Descriptor of the PLME-GET.Confirm primitive.	plmeGetcnf_t

4.3.5 Generic interface

4.3.5.1 Phy_Init

This function creates and initializes all of the PHY instances that the system is designed with.

Synopsis:

```
void Phy_Init( void );
```

Direction: APP → PHY

4.3.5.2 BindToPHY

This function creates a logical binding with the next available (un-binded) PHY instance.

Synopsis:

```
instanceId_t BindToPhy( instanceId_t macInstance );
```

Direction: MAC → PHY

Argument	Description	Type
macInstance	Instance with which the caller layer will create the logical binding.	instanceId_t

4.3.5.3 PhyPpGetState

This function checks the states of all the instances of the PHY.

Synopsis:

```
uint8_t PhyPpGetState( void );
```

Direction: MAC → PHY

4.3.5.4 Phy_RegisterSapHandlers

This function registers the PD and PLME SAPs, offering support for the PD and PLME to MAC message interactions.

Synopsis:

```
void Phy_RegisterSapHandlers( PD_MAC_SapHandler_t pPD_MAC_SapHandler,
                              PLME_MAC_SapHandler_t pPLME_MAC_SapHandler,
                              instanceId_t instanceId);
```

Direction: NWK → MAC

Argument	Description	Type
pPD_MAC_SapHandler	Pointer to the PD to MAC SAP Handler.	PD_MAC_SapHandler_t
pPLME_MAC_SapHandler	Pointer to the PLME to MAC SAP Handler function callback.	PLME_MAC_SapHandler_t
instanceId	PHY instance for which the SAP registration is performed.	instanceId_t

4.3.6 MAC to PHY SAP

The following functions are exposed towards the MAC layer, offering the upwards interface of the communication stack from MAC to PHY.

4.3.6.1 MAC_PD_SapHandler

This function is part of the MAC data service, offering support for the PD-SAP to PHY interactions.

Synopsis:

```
phyStatus_t MAC_PD_SapHandler(macToPdDataMessage_t * pMsg, instanceId_t
phyInstance );
```

Direction: PHY → MAC

Argument	Description	Type
pMsg	Pointer to a structure containing the message information – the primitive identifier and primitive parameters.	macToPdDataMessage_t *

phyInstance	Identifier of the PHY instance for which the primitive is called.	instanceId_t

4.3.6.2 MAC_PLME_SapHandler

This function is a part of the MAC management service, offering support for the PLME-SAP to PHY interactions.

Synopsis:

```
phyStatus_t MAC_PLME_SapHandler( macToPlmeMessage_t* pMsg, instanceId_t
phyInstance );
```

Direction: PHY → MAC

Argument	Description	Type
pMsg	Pointer to a structure containing the message information – the primitive identifier and primitive parameters.	macToPlmeMessage_t *
phyInstance	Identifier of the PHY instance for which the primitive is called.	instanceId_t

4.4 PHY time services

The interface described here is a dedicated API for services offered by a generic hardware timer with high resolution and with availability when the system enters a Low Power operation mode.

4.5 Constant macro definitions

4.5.1 gInvalidTimerId_c

This constant defines the identification value of an invalid timer.

Used by:

PhyTime_ScheduleEvent

Synopsis:

```
#define gInvalidTimerId_c (0xFF)
```

4.6 Data type definition

The following data types are needed for the implementation of the services exposed by the PHY Timer Module API.

4.6.1 phyTimeTimestamp_t

Synopsis:

```
typedef uint32_t phyTimeTimestamp_t;
```

Synonym	Description	Base Type
phyTimeTimestamp_t	Timestamp value reported by the hardware timer module.	uint32_t

4.6.2 phyTimeTimerId_t

Synopsis:

```
typedef uint8_t phyTimeTimerId_t;
```

Synonym	Description	Base Type
phyTimeTimerId_t	The ID of an instantiated timer.	uint8_t

4.6.3 phyTimeCallback_t

Used by:

SAPs

Synopsis:

```
typedef void (*phyTimeCallback_t)(uint32_t parameter);
```

Argument	Description	Type
parameter	Parameter passed to the callback function.	uint32_t

4.6.4 phyTimeStatus_t

PHY Timer Module result status.

Used by:

SAPs

Synopsis:

```
typedef enum
{
    gPhyTimeOk_c                    = 0x00,
    gPhyTimeAlreadyPassed_c         = 0x01,
    gPhyTimeTooClose_c              = 0x02,
    gPhyTimeTooMany_c               = 0x03,
    gPhyTimeInvalidParameter_c      = 0x04,
    gPhyTimeNotFound_c              = 0x05,
    gPhyTimeError_c                 = 0x06
}phyTimeStatus_t;
```

Constant	Description	Value
gPhyTimeOk_c	The request was performed successfully.	0x00
gPhyTimeAlreadyPassed_c	Requested time for event trigger has already passed.	0x01
gPhyTimeTooClose_c	Requested time for event trigger is too close to the current time to be processed.	0x02
gPhyTimeTooMany_c	Too many requests have been addressed.	0x03
gPhyTimeInvalidParameter_c	The request was performed with an invalid parameter.	0x04
gPhyTimeNotFound_c	The requested timer was not found.	0x05
gPhyTimeError_c	The request encoundered a hardware error.	0x06

4.6.5 phyTimeEvent_t

Used by:

SAPs

Synopsis:

```
typedef struct phyTimeEvent_tag
{
    phyTimeTimestamp_t      timestamp;
    phyTimeCallback_t       callback;
    uint32_t                parameter;
}phyTimeEvent_t;
```

Member	Description	Value
timestamp	Timestamp at which the hardware timer module will trigger the event.	phyTimeTimestamp_t
callback	Pointer to the callback function designated for the specific event.	phyTimeCallback_t
parameter	Parameter passed to the callback function.	uint32_t

4.7 Functions

Declaration of the interface functions exposed by the PHY Timer Module.

4.7.1 PhyTime_TimerInit()

This function performs the initialization of the PHY Timer Module.

Synopsis:

```
phyTimeStatus_t PhyTime_TimerInit(void);
```

Parameters: N/A

Return: Status of the request.

4.7.2 PhyTime_GetTimestamp()

This function returns the current timestamp, reported by the PHY Timer Module.

Synopsis:

```
phyTimeTimestamp_t PhyTime_GetTimestamp(void);
```

Parameters: N/A

Return: Current timestamp.

4.7.3 PhyTime_ScheduleEvent()

This function schedules a timed event. The event context is given by the configuration structure.

Synopsis:

```
phyTimeTimerId_t PhyTime_ScheduleEvent(phyTimeEvent_t *pEvent);
```

Parameters:

Argument	Description	Type
pEvent	Pointer to the structure with the event configuration that must be scheduled.	phyTimeEvent_t

Return: The ID of the scheduled timer, or gInvalidTimerId_c, if scheduling fails.

4.7.4 PhyTime_CancelEvent()

This function cancels an event that has already been scheduled.

Synopsis:

```
phyTimeStatus_t PhyTime_CancelEvent(phyTimeTimerId_t timerId);
```

Parameters:

Argument	Description	Type
timerId	The ID of the timer that must be cancelled.	phyTimeTimerId_t

Return: Status of the requestPhyTime_CancelEventsWithParam ().

This function cancels all events registered with the specified parameter.

Synopsis:

```
phyTimeStatus_t PhyTime_CancelEventsWithParam(uint32_t param);
```

Parameters:

Argument	Description	Type
param	The parameter of the event(s) to be canceled.	uint32_t

Return: Status of the request.

5 ASP interface

The Application Support Package (ASP) contains XCVR-specific functionality. It can be used for XCVR testing, and for accessing XCVR-specific features (for example FAD).

5.1 Constant macro definitions

5.1.1 gAspCapability_d

Synopsis:

```
#ifndef gAspCapability_d
#ifndef gPHY_802_15_4g_d
#define gAspCapability_d (1)
#else
#define gAspCapability_d (0)
#endif // gPHY_802_15_4g_d
#endif // gAspCapability_d
```

Description:

Enable / disable the ASP functionality.

5.1.2 gAspInterfaceId

Synopsis:

```
#define gAspInterfaceId (0)
```

Description:

The default FSCI interface used by the ASP module.

5.2 Data type definitions

5.2.1 AspStatus_t

Synopsis:

```
typedef enum{
    gAspSuccess_c          = 0x00,
    gAspInvalidRequest_c   = 0xC2,
    gAspDenied_c           = 0xE2,
    gAspTooLong_c          = 0xE5,
    gAspInvalidParameter_c = 0xE8
}AspStatus_t;
```

Synonym	Description	Value
gAspSuccess_c	The ASP command execution was successful.	0x00
gAspInvalidRequest_c	The ASP request does not exist.	0xC2
gAspDenied_c	The request is not allowed at this point.	0xE2
gAspTooLong_c	The payload of the ASP request is too long.	0xE5

		ASP interface
gAspInvalidParameter_c	The request contains an invalid parameter.	0xE8

5.2.2 Telec test modes

Synopsis:

```
enum {
    gTestForceIdle_c               = 0,
    gTestPulseTxPrbs9_c            = 1,
    gTestContinuousRx_c            = 2,
    gTestContinuousTxMod_c         = 3,
    gTestContinuousTxNoMod_c       = 4,
    gTestContinuousTx2Mhz_c        = 5,
    gTestContinuousTx200Khz_c      = 6,
    gTestContinuousTx1MbpsPRBS9_c  = 7,
    gTestContinuousTxExternalSrc_c = 8,
    gTestContinuousTxNoModZero_c   = 9,
    gTestContinuousTxNoModOne_c    = 10
};
```

Synonym	Description	Value
gTestForceIdle_c	Force XCVR to idle.	0
gTestPulseTxPrbs9_c	Send a PRBS9 payload in packet mode.	1
gTestContinuousRx_c	Set XCVR in continuous Rx mode.	2
gTestContinuousTxMod_c	Transmit the 802.15.4 carrier.	3
gTestContinuousTxNoMod_c	Transmit an unmodulated signal.	4
gTestContinuousTx2Mhz_c	Transmit a 2 MHz clock signal.	5
gTestContinuousTx200Khz_c	Transmit a 200 KHz clock signal.	6
gTestContinuousTx1MbpsPRBS9_c	Transmit a continuous PRBS9 signal.	7
gTestContinuousTxExternalSrc_c	Transmit pattern from an external source.	8
gTestContinuousTxNoModZero_c	Transmit a 1 pattern.	9
gTestContinuousTxNoModOne_c	Transmit a 0 pattern.	10

5.2.3 Supported power levels

Synopsis:

```
/* Supported Power Levels */
enum {
    gAspPowerLevel_m40dBm = 0x03,
    gAspPowerLevel_m38dBm = 0x04,
    gAspPowerLevel_m36dBm = 0x05,
    gAspPowerLevel_m34dBm = 0x06,
    gAspPowerLevel_m32dBm = 0x07,
    gAspPowerLevel_m30dBm = 0x08,
    gAspPowerLevel_m28dBm = 0x09,
    gAspPowerLevel_m26dBm = 0x0A,
    gAspPowerLevel_m24dBm = 0x0B,
    gAspPowerLevel_m22dBm = 0x0C,
    gAspPowerLevel_m20dBm = 0x0D,
    gAspPowerLevel_m18dBm = 0x0E,
    gAspPowerLevel_m16dBm = 0x0F,
    gAspPowerLevel_m14dBm = 0x10,
    gAspPowerLevel_m12dBm = 0x11,
    gAspPowerLevel_m10dBm = 0x12,
    gAspPowerLevel_m8dBm  = 0x13,
    gAspPowerLevel_m6dBm  = 0x14,
    gAspPowerLevel_m4dBm  = 0x15,
```

```
    gAspPowerLevel_m2dBm  = 0x16,
    gAspPowerLevel_0dBm   = 0x17,
    gAspPowerLevel_2dBm   = 0x18,
    gAspPowerLevel_4dBm   = 0x19,
    gAspPowerLevel_6dBm   = 0x1A,
    gAspPowerLevel_8dBm   = 0x1B,
    gAspPowerLevel_10dBm  = 0x1C,
    gAspPowerLevel_12dBm  = 0x1D,
    gAspPowerLevel_14dBm  = 0x1E,
    gAspPowerLevel_16dBm  = 0x1F
};
```

5.2.4 AppAspMsgType_t

Synopsis:

```
typedef enum {
    aspMsgTypeGetTimeReq_c          = 0x00,
    aspMsgTypeGetInactiveTimeReq_c  = 0x01,
    aspMsgTypeGetMacStateReq_c      = 0x02,
    aspMsgTypeDozeReq_c             = 0x03,
    aspMsgTypeAutoDozeReq_c         = 0x04,
    aspMsgTypeAcomaReq_c            = 0x05,
    aspMsgTypeHibernateReq_c        = 0x06,
    aspMsgTypeWakeReq_c             = 0x07,
    aspMsgTypeEventReq_c            = 0x08,
    aspMsgTypeClkoReq_c             = 0x09,
    aspMsgTypeTrimReq_c             = 0x0A,
    aspMsgTypeDdrReq_c              = 0x0B,
    aspMsgTypePortReq_c             = 0x0C,
    aspMsgTypeSetMinDozeTimeReq_c   = 0x0D,
    aspMsgTypeSetNotifyReq_c        = 0x0E,
    aspMsgTypeSetPowerLevel_c       = 0x0F,
    aspMsgTypeGetPowerLevel_c       = 0x1F,
    aspMsgTypeTelecTest_c           = 0x10,
    aspMsgTypeTelecSetFreq_c        = 0x11,
    aspMsgTypeGetInactiveTimeCnf_c  = 0x12,
    aspMsgTypeGetMacStateCnf_c      = 0x13,
    aspMsgTypeDozeCnf_c             = 0x14,
    aspMsgTypeAutoDozeCnf_c         = 0x15,
    aspMsgTypeTelecSendRawData_c    = 0x16,
    aspMsgTypeSetFADState_c         = 0x17,
    aspMsgTypeSetFADThreshold_c     = 0x18,
    aspMsgTypeGetFADThreshold_c     = 0x19,
    aspMsgTypeGetFADState_c         = 0x1A,
    aspMsgTypeSetActivePromState_c  = 0x1B,
    aspMsgTypeXcvrWriteReq_c        = 0x1C,
    aspMsgTypeXcvrReadReq_c         = 0x1D,
    aspMsgTypeSetANTXState_c        = 0x20,
    aspMsgTypeGetANTXState_c        = 0x21,
    aspMsgTypeSetLQIMode_c          = 0x22,
    aspMsgTypeGetRSSILevel_c        = 0x23,
    aspMsgTypeSetMpmConfig_c        = 0x24,
    aspMsgTypeGetMpmConfig_c        = 0x25
}AppAspMsgType_t;
```

Description:

This enumeration matches the FSCI OpCode used by ASP.

5.2.5 aspEventReq_t

Synopsis:

```
typedef PACKED_STRUCT aspEventReq_tag
{   /* AspEvent.Request              */
    uint32_t eventTime;
} aspEventReq_t;
```

5.2.6 aspGetTimeReq_t

Synopsis:

```
typedef PACKED_STRUCT aspGetTimeReq_tag
{   /* AspGetTime.Request            */
    uint64_t time;
} aspGetTimeReq_t;
```

5.2.7 aspSetNotifyReq_t

Synopsis:

```
typedef PACKED_STRUCT aspSetNotifyReq_tag
{   /* AspSetNotify.Request          */
    uint8_t notifications;
} aspSetNotifyReq_t;
```

5.2.8 aspSetPowerLevelReq_t

Synopsis:

```
typedef PACKED_STRUCT aspSetPowerLevelReq_tag
{   /* AspSetPowerLevel.Request      */
    uint8_t powerLevel;
} aspSetPowerLevelReq_t;
```

5.2.9 aspGetPowerLevelReq_t

Synopsis:

```
typedef PACKED_STRUCT aspGetPowerLevelReq_tag
{   /* AspGetPowerLevel.Request      */
    uint8_t powerLevel;
} aspGetPowerLevelReq_t;
```

5.2.10 aspTelecTest_t

Synopsis:

```
typedef PACKED_STRUCT aspTelecTest_tag
{   /* AspTelecTest.Request          */
    uint8_t mode;
} aspTelecTest_t;
```

5.2.11 aspTelecsetFreq_t

Synopsis:

```
typedef PACKED_STRUCT aspTelecsetFreq_tag
{   /* AspTelecSetFreq.Request       */
    uint8_t channel;
} aspTelecsetFreq_t;
```

5.2.12 aspTelecSendRawData_t

Synopsis:

```
typedef PACKED_STRUCT aspTelecSendRawData_tag
{   /* AspTelecSendRawData.Request   */
    uint8_t  length;
    uint8_t* dataPtr;
} aspTelecSendRawData_t;
```

5.2.13 aspFADThreshold_t

Synopsis:

```
    /* AspSetFADThreshold.Request   */
typedef uint8_t aspFADThreshold_t;
```

5.2.14 aspLQIMode_t

Synopsis:

```
    /* AspSetLQIMode.Request    */
typedef uint8_t aspLQIMode_t;
```

5.2.15 aspXcvrReq_t

Synopsis:

```
typedef PACKED_STRUCT aspXcvrReq_tag
{   /* AspXcvrWrite.Request / AspXcvrRead.Request   */
    uint8_t  mode;
    uint16_t addr;
    uint8_t  len;
    uint8_t  data[4]; /* more than 4 bytes can be read/written */
} aspXcvrReq_t;
```

5.2.16 AppToAspMessage_t

Synopsis:

```
typedef PACKED_STRUCT AppToAspMessage_tag
{
    AppAspMsgType_t msgType;
    PACKED_UNION
    {
        aspEventReq_t           aspEventReq;
```

```
        aspGetTimeReq_t         aspGetTimeReq;
        aspSetPowerLevelReq_t   aspSetPowerLevelReq;
        aspGetPowerLevelReq_t   aspGetPowerLevelReq;
        aspTelecTest_t          aspTelecTest;
        aspTelecsetFreq_t       aspTelecsetFreq;
        aspTelecSendRawData_t   aspTelecSendRawData;
        aspFADThreshold_t       aspFADThreshold;
        bool_t                  aspFADState;
        bool_t                  aspANTXState;
        aspLQIMode_t            aspLQIMode;
        bool_t                  aspActivePromState;
        aspXcvrReq_t            aspXcvrData;
        mpmConfig_t             MpmConfig;
    }msgData;
} AppToAspMessage_t;
```

Description:

This union is used to send ASP request using the *APP_ASP_SapHandler()*.

5.3 Functions

The ASP layer can be interfaced using the SAP handler, or using direct function calls.

5.3.1 ASP_Init

This function initializes the ASP module, and registers the ASP Operation Groups to the FSCI. If the FSCI is not used, the *interfaceId* parameter is ignored.

Synopsis:

```
void ASP_Init( instanceId_t phyInstance, uint8_t interfaceId );
```

Parameters:

Argument	Description	Type
phyInstance	The instance of the PHY.	instanceId_t
interfaceId	The FSCI interface on which the ASP OG will be registered.	uint8_t

Return: None.

5.3.2 APP_ASP_SapHandler

This function represents the Service Access Point (SAP) of the ASP layer. This SAP is used by other layers to send requests to the ASP layer.

Synopsis:

```
AspStatus_t APP_ASP_SapHandler(AppToAspMessage_t *pMsg, instanceId_t instanceId);
```

Parameters:

Argument	Description	Type
pMsg	Pointer to an ASP request message.	AppToAspMessage_t*
interfaceId	The instance of the PHY layer.	uint8_t

Return: AspStatus_t

5.3.3 Asp_GetTimeReq

Returns the absolute time from the XCVR module.

Synopsis:

```
void Asp_GetTimeReq(uint64_t *time);
```

Parameters:

Argument	Description	Type
time	Pointer to a memory location where the time will be stored.	uint64_t*

Return: None.

5.3.4 Asp_XcvrWriteReq

This function is used for writing XCVR registers.

Synopsis:

```
AspStatus_t Asp_XcvrWriteReq
(
uint8_t mode,
uint16_t addr,
uint8_t len,
uint8_t* pData
);
```

Parameters:

Argument	Description	Type
mode	The XCVR access mode (direct / indirect).	uint8_t
Addr	The XCVR address.	uint16_t
Len	Length of the data to be written (in bytes).	uint8_t
pData	Pointer to the data to be written.	uint8_t*

Return: AspStatus_t

5.3.5 Asp_XcvrReadReq

This function is used for reading XCVR registers.

Synopsis:

```
AspStatus_t Asp_XcvrReadReq
(
uint8_t mode,
uint16_t addr,
uint8_t len,
uint8_t* pData
);
```

Parameters:

Argument	Description	Type
mode	The XCVR access mode (direct / indirect).	uint8_t
Addr	The XCVR address.	uint16_t
Len	Length of the data to be read (in bytes).	uint8_t
pData	Pointer to the data to be read.	uint8_t*

Return: AspStatus_t

5.3.6 Asp_SetPowerLevel

Set the XCVR Tx power level.

Synopsis:

```
AspStatus_t Asp_SetPowerLevel(uint8_t powerLevel);
```

Parameters:

Argument	Description	Type
powerLevel	The XCVR power level.	uint8_t

Return: AspStatus_t

5.3.7 Asp_GetPowerLevel

Returns the XCVR Tx power level.

Synopsis:

```
uint8_t     Asp_GetPowerLevel(void);
```

Parameters:

None

Return: powerLevel

5.3.8 Asp_SetActivePromState

Set the state of the Active Promiscuous feature.

Synopsis:

```
AspStatus_t Asp_SetActivePromState(bool_t state);
```

Parameters:

Argument	Description	Type
state	If ActivePromiscuous must be enabled, set to TRUE, else FALSE.	bool_t

Return: AspStatus_t

5.3.9 Asp_SetFADState

Set the state of the Fast Antenna Diversity (FAD) feature.

Synopsis:

```
AspStatus_t Asp_SetFADState(bool_t state);
```

Parameters:

Argument	Description	Type
state	If FAD should be enabled, set to TRUE, else FALSE.	bool_t

Return: AspStatus_t

5.3.10 Asp_SetFADThreshold

Set the correlator threshold for the Fast Antenna Diversity.

Synopsis:

```
AspStatus_t Asp_SetFADThreshold(uint8_t thresholdFAD);
```

Parameters:

Argument	Description	Type
thresholdFAD	Correlator threshold at which the FAD will select the antenna.	uint8_t

Return: AspStatus_t

5.3.11 Asp_SetANTXState

If FAD is enabled, this function sets the starting antenna for the FAD feature, else it simply selects the active antenna.

Synopsis:

```
AspStatus_t Asp_SetANTXState(uint8_t state);
```

Parameters:

Argument	Description	Type
state	1 – use Anntenna_1; 0 – use Antenna_0	uint8_t

Return: AspStatus_t

5.3.12 Asp_GetANTXState

Returns the active antenna (in FAD and non-FAD modes).

Synopsis:

```
uint8_t     Asp_GetANTXState(void);
```

Parameters:

None

Return:

Active antenna.

5.3.13 Asp_SetANTPadStateRequest

Set the state of the ANT pads and RxTx pads

Synopsis:

```
uint8_t     Asp_SetANTPadStateRequest(bool_t antAB_on, bool_t rxtxSwitch_on);
```

Parameters:

Argument	Description	Type
antAB_on	Enable / disable the ANT_A and ANT_B pads.	bool_t
rxtxSwitch_on	Enable / disable the RxTxSwitch pads.	bool_t

Return: gAspSuccess_c

5.3.14 Asp_SetANTPadStrengthRequest

Enable / disable hiStrength on the ANT_x and RxTx pads.

Synopsis:

```
uint8_t     Asp_SetANTPadStrengthRequest(bool_t hiStrength);
```

Parameters:

Argument	Description	Type
hiStrength	Enable / disable hiStrength on the ANTX pads.	bool_t

Return: gAspSuccess_c

5.3.15 Asp_SetANTPadInvertedRequest

Invert ANTX pad logic.

Synopsis:

```
uint8_t     Asp_SetANTPadInvertedRequest
(
bool_t invAntA,
bool_t invAntB,
bool_t invTx,
bool_t invRx
);
```

Parameters:

Argument	Description	Type
invAntA	Invert ANT_A pad logic.	bool_t
invAntB	Invert ANT_B pad logic.	bool_t
invTx	Invert Tx pad logic.	bool_t
invRx	Invert Rx pad logic.	bool_t

Return: gAspSuccess_c

5.3.16 Asp_SetLQIMode

Set the LQI mode: 1 – based on RSSI; 0 – based on correlator peaks.

NOTE

Default LQI mode is based on RSSI.

Synopsis:

```
AspStatus_t Asp_SetLQIMode(bool_t mode);
```

Parameters:

Argument	Description	Type
mode	Enable / disable LQI based on RSSI.	bool_t

Return: AspStatus_t

5.3.17 Asp_GetRSSILevel

Read current XCVR RSSI level.

Synopsis:

```
uint8_t     Asp_GetRSSILevel(void);
```

Parameters:

None.

Return: RSSI value

5.3.18 ASP_TelecSetFreq

Set the 802.15.4 channel for the Telec module.

Synopsis:

```
AspStatus_t ASP_TelecSetFreq    (uint8_t channel);
```

Parameters:

Argument	Description	Type
channel	802.15.4 channel	uint8_t

Return: AspStatus_t

5.3.19 ASP_TelecSendRawData

Send a raw data packet using the Telec module.

Synopsis:

```
AspStatus_t ASP_TelecSendRawData(uint8_t* dataPtr);
```

Parameters:

Argument	Description	Type
dataPtr	Pointer to the data to be sent over-the-air.	uint8_t*

Return: AspStatus_t

5.3.20 ASP_TelecTest

Start a Telec test.

Synopsis:

```
AspStatus_t ASP_TelecTest       (uint8_t mode);
```

Parameters:

Argument	Description	Type
Mode	Telec test mode	uint8_t

Return: AspStatus_t

6 MPM interface

The Multi-PAN Manager (MPM) module is used to enable multiple MAC instances to access a single PHY.

6.1 Constant macro definitions

6.1.1 gMpmMaxPANs_c

Synopsis:

```
#if !defined(gMpmMaxPANs_c)
#define gMpmMaxPANs_c 1
#endif
```

Description:

Defines the number of Personal Area Networks (PAN) to be used. The value must be greater than 1 to enable the MPM functionality.

6.1.2 gMpmIncluded_d

Synopsis:

```
#define gMpmIncluded_d (gMpmMaxPANs_c > 1)
```

Description:

This define illustrates the state of the MPM functionality. It must not be modified directly.

6.1.3 gMpmPhyPanRegSets_c

Synopsis:

```
#define gMpmPhyPanRegSets_c          (2)
```

Description:

Defines the number of hardware register sets that are used to store PAN-specific information.

6.1.4 gMpmUseDifferentTxPwrLevel_c

Synopsis:

```
#define gMpmUseDifferentTxPwrLevel_c (0)
```

Description:

Enable / disable different Tx power levels on every PAN. If enabled, the MPM will set the power level before the first Tx on a PAN.

6.1.5 mDefaultDualPanDwellPrescaller_c

Synopsis:

```
#define mDefaultDualPanDwellPrescaller_c (0x00)
```

Description:

Defines the default PAN switch dwell time prescaler. The default value offers a granularity of 0.5 ms to set the dwell timer.

6.1.6 mDefaultDualPanDwellTime_c

Synopsis:

```
#define mDefaultDualPanDwellTime_c       (0x06)
```

Description:

Default value of the PAN switch dwell timer, in multiples of the time base set by the prescaler. The default value offers a dwell time of (6 + 1) * 0.5 ms = 3.5 ms.

6.1.7 MPM_isPanActive

Synopsis:

```
#define  MPM_isPanActive( panIdx ) (MPM_GetRegSet(panIdx) != gMpmInvalidRegSet_c)
```

Description:

This define is used to determine, whether the PAN specified as parameter is the active one.

6.2 Data type definitions

6.2.1 mpmConfig_t

Synopsis:

```
typedef PACKED_STRUCT mpmConfig_tag{
    bool_t  autoMode;
    uint8_t dwellTime;
    uint8_t activeMAC;
}mpmConfig_t;
```

Member	Description	Value
autoMode	Enable / disable automatic PAN switch.	bool_t
dwellTime	The value of the hardware PAN switch dwell timer.	uint8_t
activeMAC	The instance of the active MAC.	uint8_t

6.2.2 panFlags_t

Synopsis:

```
typedef union panFlags_tag{
    uint16_t all;
    struct{
        uint16_t panCoordinator: 1;
        uint16_t promiscuous:    1;
        uint16_t rxOnWhenIdle:   1;
        uint16_t reserved:       13;
    };
}panFlags_t;
```

Member	Description	Value
all	Used to reference all fields from below.	uint16_t
panCoordinator	The value of the PanCoordinator PIB for a PAN.	uint16_t:1
promiscuous	The value of the Promiscuous PIB for a PAN.	uint16_t:1
rxOnWhenIdle	The value of the RxOnWhenIdle PIB for a PAN.	uint16_t:1
reserved	Unused bits. Must be set to 0.	uint16_t:13

6.2.3 panInfo_t

Synopsis:

```
typedef struct panInfo_tag{
    uint8_t         flags;
    uint8_t         macInstance;
    uint8_t         phyRegSet;
    int8_t          locked;
#if (gMpmMaxPANs_c > gMpmPhyPanRegSets_c)
    uint64_t        longAddr;
    uint16_t        shortAddr;
    uint16_t        panId;
    uint8_t         channel;
#endif
#if gMpmUseDifferentTxPwrLevel_c
    uint8_t         pwrLevel;
#endif
}panInfo_t;
```

Member	Description	Value
flags	The state of relevant MAC PIBs.	uint8_t
macInstance	The instance of the MAC.	uint8_t
phyRegSet	The id of the PHY registers set.	uint8_t
longAddr	The value of the MAC IEEE address.	uint64_t
shortAddr	The value of the MAC Short Address.	uint16_t
panId	The value of the MAC PanID.	uint16_t
channel	The 802.15.4 channel used by the MAC.	uint8_t
pwrLevel	The Tx power level used by the MAC.	uint8_t

6.3 Functions

Most of the functions below are used internally by the PHY layer, so there is no need for a direct call (except for MPM_SetConfig() and MPM_SetConfig() functions).

6.3.1 MPM_Init

This function initializes the MPM module. This function is called at the initialization of the PHY layer so there is no need for an explicit call.

Synopsis:

```
void MPM_Init( void );
```

Parameters: None.

Return: None.

6.3.2 MPM_SetConfig

This function is used by other layers to change the current MPM configuration. Read the MPM configuration first, modify it, and then write the updated configuration.

Synopsis:

```
void MPM_SetConfig( mpmConfig_t *pCfg );
```

Parameters:

Argument	Description	Type
pCfg	Pointer to the new MPM configuration.	mpmConfig_t

Return: None.

6.3.3 MPM_GetConfig

This function is used by other layers to read the current MPM configuration.

Synopsis:

```
void MPM_GetConfig( mpmConfig_t *pCfg );
```

Parameters:

Argument	Description	Type
pCfg	Pointer to a MPM configuration structure, where the current configuration will be stored.	mpmConfig_t

Return: None.

6.3.4 MPM_GetPanIndex

This function is used by the PHY to determine the MPM PAN index associated to a MAC instance.

Synopsis:

```
int32_t  MPM_GetPanIndex( instanceId_t macInstance );
```

Parameters:

Argument	Description	Type
macInstance	The instance of the MAC.	instanceId _t

Return: The MPM PAN index if found, else –1.

6.3.5 MPM_GetRegSet

This function is used by the PHY to determine the PAN register set associated to an MPM PAN index.

Synopsis:

```
uint32_t MPM_GetRegSet( uint8_t panIdx );
```

Parameters:

Argument	Description	Type
panIdx	The MPM PAN index.	uint8_t

Return: The PAN register set.

6.3.6 MPM_GetMacInstanceFromRegSet

This function is used by the PHY to determine the MAC instance associated with the specified PAN register set.

Synopsis:

```
uint32_t MPM_GetMacInstanceFromRegSet(uint32_t regSet);
```

Parameters:

Argument	Description	Type
regSet	The PHY PAN register set.	uint32_t

Return: The instance of the MAC.

6.3.7 MPM_PrepareForTx

This function is called by the PHY layer before a new Tx is initiated.

Synopsis:

```
phyStatus_t MPM_PrepareForTx( instanceId_t macInstance );
```

Parameters:

Argument	Description	Type
macInstance	The instance of the MAC which requested the Tx.	instanceId _t

Return: phyStatus_t

6.3.8 MPM_PrepareForRx

This function is called by the PHY layer before a new Rx is initiated.

Synopsis:

```
phyStatus_t MPM_PrepareForRx( instanceId_t macInstance );
```

Parameters:

Argument	Description	Type
macInstance	The instance of the MAC which requested the Rx.	instanceId _t

Return: phyStatus_t

6.3.9 MPM_GetPIB

This function is called by the PHY layer when the value of a PIB is requested by the upper layer.

Synopsis:

```
phyStatus_t MPM_GetPIB(phyPibId_t pibId, void* pValue, uint8_t panIdx);
```

Parameters:

Argument	Description	Type
pibId	The Id of the PIB.	phyPibId_t
pValue	Pointer where to store the PIB value.	void*
panIdx	The index of the PAN on which the change occurred.	uint8_t

Return: phyStatus_t

6.3.10 MPM_SetPIB

This function is called by the PHY layer when a PIB value is changed, to inform the MPM of the new PIB value.

Synopsis:

```
phyStatus_t MPM_SetPIB(phyPibId_t pibId, void* pValue, uint8_t panIdx);
```

Parameters:

Argument	Description	Type
pibId	The Id of the PIB.	phyPibId_t
pValue	Pointer to the PIB value.	void*
panIdx	The index of the PAN on which the change occurred.	uint8_t

Return: phyStatus_t

Table 6 Revision history

Rev. number	Date	Substantive changes
0	03/2015	Initial release.
1	06/2015	Changes related to IEEE 802.15.4g/e specific new features.

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support4

Document number: 802154MPAPIRM
Rev. 1
07/2015

freescale™