7. MAC sublayer specification

This clause specifies the MAC sublayer of IEEE P802.15.4REVb/D6. The MAC sublayer handles all access to the physical radio channel and is responsible for the following tasks:

— Generating network beacons if the device is a coordinator.
— Synchronizing to network beacons.
— Supporting PAN association and disassociation.
— Supporting device security.
— Employing the CSMA-CA mechanism for channel access.
— Handling and maintaining the GTS mechanism.
— Providing a reliable link between two peer MAC entities.

Constants and attributes that are specified and maintained by the MAC sublayer are written in the text of this clause in italics. Constants have a general prefix of “a”, e.g., *aBaseSlotDuration*, and are listed in Table 85 (see 7.4.1). Attributes have a general prefix of “mac”, e.g., *macAckWaitDuration*, and are listed in Table 86 (see 7.4.2), while the security attributes are listed in Table 88 (see 7.6.1).

7.1 MAC sublayer service specification

The MAC sublayer provides an interface between the SSCS and the PHY. The MAC sublayer conceptually includes a management entity called the MLME. This entity provides the service interfaces through which layer management functions may 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.

Figure 29 depicts the components and interfaces of the MAC sublayer.

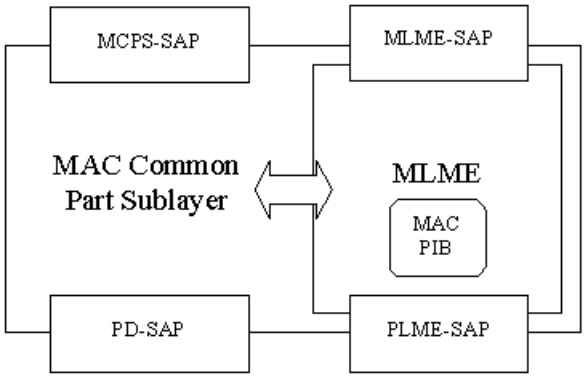


Figure 29—The MAC sublayer reference model

The MAC sublayer provides two services, accessed through two SAPs:

— The MAC data service, accessed through the MAC common part sublayer (MCPS) data SAP (MCPS-SAP), and
— The MAC management service, accessed through the MLME-SAP.

These two services provide the interface between the SSCS and the PHY, via the PD-SAP and PLME-SAP interfaces (see 6.2). In addition to these external interfaces, an implicit interface also exists between the MLME and the MCPS that allows the MLME to use the MAC data service.

7.1.1 MAC data service

The MCPS-SAP supports the transport of SSCS protocol data units (SPDUs) between peer SSCS entities. Table 40 lists the primitives supported by the MCPS-SAP. Primitives marked with a diamond (♦) are optional for an RFD. These primitives are discussed in the subclauses referenced in the table.

Table 40—MCPS-SAP primitives

MCPS-SAP primitive	Request	Confirm	Indication
MCPS-DATA	7.1.1.1	7.1.1.2	7.1.1.3
MCPS-PURGE	7.1.1.4♦	7.1.1.5♦	—

7.1.1.1 MCPS-DATA.request

The MCPS-DATA.request primitive requests the transfer of a data SPDU (i.e., MSDU) from a local SSCS entity to a single peer SSCS entity.

7.1.1.1.1 Semantics of the service primitive

The semantics of the MCPS-DATA.request primitive are as follows:

```
MCPS-DATA.request (
                  SrcAddrMode,
                  DstAddrMode,
                  DstPANId,
                  DstAddr,
                  msduLength,
                  msdu,
                  msduHandle,
                  TxOptions,
                  SecurityLevel,
                  KeyIdMode,
                  KeySource,
                  KeyIndex
                  )
```

Table 41 specifies the parameters for the MCPS-DATA.request primitive.

7.1.1.1.2 Appropriate usage

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

7.1.1.1.3 Effect on receipt

On receipt of the MCPS-DATA.request primitive, the MAC sublayer entity begins the transmission of the supplied MSDU.

Table 41—MCPS-DATA.request parameters

Name	Type	Valid range	Description
SrcAddrMode	Integer	0x00–0x03	The source addressing mode for this primitive and subsequent MPDU. This value can take one of the following values: 0x00 = no address (addressing fields omitted, see 7.2.1.1.8). 0x01 = reserved. 0x02 = 16-bit short address. 0x03 = 64-bit extended address.
DstAddrMode	Integer	0x00–0x03	The destination addressing mode for this primitive and subsequent MPDU. This value can take one of the following values: 0x00 = no address (addressing fields omitted, see 7.2.1.1.6). 0x01 = reserved. 0x02 = 16-bit short address. 0x03 = 64-bit extended address.
DstPANId	Integer	0x0000–0xffff	The 16-bit PAN identifier of the entity to which the MSDU is being transferred.
DstAddr	Device address	As specified by the DstAddrMode parameter	The individual device address of the entity to which the MSDU is being transferred.
msduLength	Integer	≤ *aMaxMACPayloadSize*	The number of octets contained in the MSDU to be transmitted by the MAC sublayer entity.
msdu	Set of octets	—	The set of octets forming the MSDU to be transmitted by the MAC sublayer entity.
msduHandle	Integer	0x00–0xff	The handle associated with the MSDU to be transmitted by the MAC sublayer entity.
TxOptions	Bitmap	3 bit field	The 3 bits (b_0, b_1, b_2) indicate the transmission options for this MSDU. For b_0, 1 = acknowledged transmission, 0 = unacknowledged transmission. For b_1, 1 = GTS transmission, 0 = CAP transmission for a beacon-enabled PAN. For b_2, 1 = indirect transmission, 0 = direct transmission. For a nonbeacon-enabled PAN, bit b_1 should always be set to 0.
SecurityLevel	Integer	0x00–0x07	The security level to be used (see Table 96).
KeyIdMode	Integer	0x00–0x03	The mode used to identify the key to be used (see Table 97). This parameter is ignored if the SecurityLevel parameter is set to 0x00.
KeySource	Set of 0, 4 or 8 octets	As specified by the KeyIdMode parameter	The originator of the key to be used (see 7.6.2.4.1). This parameter is ignored if the KeyIdMode parameter is ignored or set to 0x00.
KeyIndex	Integer	0x01–0xff	The index of the key to be used (see 7.6.2.4.2). This parameter is ignored if the KeyIdMode parameter is ignored or set to 0x00.

The MAC sublayer builds an MPDU to transmit from the supplied arguments. The flags in the SrcAddrMode and DstAddrMode parameters correspond to the addressing subfields in the frame control field (see 7.2.1.1) and are used to construct both the frame control and addressing fields of the MHR. If both the SrcAddrMode and the DstAddrMode parameters are set to 0x00 (i.e., addressing fields omitted), the MAC sublayer will issue the MCPS-DATA.confirm primitive with a status of INVALID_ADDRESS.

If the msduLength parameter is greater than *aMaxMACSafePayloadSize*, the MAC sublayer will set the frame version subfield of the frame control field to one, indicating an IEEE P802.15.4REVb/D6 enhanced frame.

The TxOptions parameter indicates how the MAC sublayer data service transmits the supplied MSDU. If the TxOptions parameter specifies that an acknowledged transmission is required, the acknowledgment request subfield of the frame control field will be set to one (see 7.5.6.4).

If the TxOptions parameter specifies that a GTS transmission is required, the MAC sublayer will determine whether it has a valid GTS (for GTS usage rules, see 7.5.7.3). If a valid GTS could not be found, the MAC sublayer will issue the MCPS-DATA.confirm primitive with a status of INVALID_GTS. If a valid GTS was found, the MAC sublayer will defer, if necessary, until the GTS. If the TxOptions parameter specifies that a GTS transmission is not required, the MAC sublayer will transmit the MSDU using either slotted CSMA-CA in the CAP for a beacon-enabled PAN or unslotted CSMA-CA for a nonbeacon-enabled PAN. Specifying a GTS transmission in the TxOptions parameter overrides an indirect transmission request.

If the TxOptions parameter specifies that an indirect transmission is required and this primitive is received by the MAC sublayer of a coordinator, the data frame is sent using indirect transmission, i.e., the data frame is added to the list of pending transactions stored on the coordinator and extracted at the discretion of the device concerned using the method described in 7.5.6.3. Transactions with a broadcast destination address will be transmitted using the mechanism described in 7.2.1.1.3. Transactions with a unicast destination address can then be extracted at the discretion of each device concerned using the method described in 7.5.6.3. If there is no capacity to store the transaction, the MAC sublayer will discard the MSDU and issue the MCPS-DATA.confirm primitive with a status of TRANSACTION_OVERFLOW. If there is capacity to store the transaction, the coordinator will add the information to the list. If the transaction is not handled within *macTransactionPersistenceTime*, the transaction information will be discarded and the MAC sublayer will issue the MCPS-DATA.confirm primitive with a status of TRANSACTION_EXPIRED. The transaction handling procedure is described in 7.5.5. If the TxOptions parameter specifies that an indirect transmission is required and either the device receiving this primitive is not a coordinator, the destination address is not present or the TxOptions parameter also specifies a GTS transmission, the indirect transmission option will be ignored.

If the TxOptions parameter specifies that an indirect transmission is not required, the MAC sublayer will transmit the MSDU using CSMA-CA either in the CAP for a beacon-enabled PAN or immediately for a nonbeacon-enabled PAN. If the TxOptions parameter specifies that a direct transmission is required and the MAC sublayer does not receive an acknowledgment from the recipient after *macMaxFrameRetries* retransmissions (see 7.5.6.4), it will discard the MSDU and issue the MCPS-DATA.confirm primitive with a status of NO_ACK.

If the SecurityLevel parameter is set to a valid value other than 0x00, indicating that security is required for this frame, the MAC sublayer will set the security enabled subfield of the frame control field to one. The MAC sublayer will perform outgoing processing on the frame based on the DstAddr, SecurityLevel, KeyIdMode, KeySource and KeyIndex parameters, as described in 7.5.8.2.1. If any error occurs during outgoing frame processing, the MAC sublayer will discard the frame and issue the MCPS-DATA.confirm primitive with the error status returned by outgoing frame processing.

If the requested transaction is too large to fit in the CAP or GTS, as appropriate, the MAC sublayer shall discard the frame and issue the MCPS-DATA.confirm primitive with a status of FRAME_TOO_LONG.

If the transmission uses CSMA-CA and the CSMA-CA algorithm failed due to adverse conditions on the channel, and the TxOptions parameter specifies that a direct transmission is required, the MAC sublayer will discard the MSDU and issue the MCPS-DATA.confirm primitive with a status of CHANNEL_ACCESS_FAILURE.

If the MPDU was successfully transmitted and, if requested, an acknowledgment was received, the MAC sublayer will issue the MCPS-DATA.confirm primitive with a status of SUCCESS.

If any parameter in the MCPS-DATA.request primitive is not supported or is out of range, the MAC sublayer will issue the MCPS-DATA.confirm primitive with a status of INVALID_PARAMETER.

7.1.1.2 MCPS-DATA.confirm

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.

7.1.1.2.1 Semantics of the service primitive

The semantics of the MCPS-DATA.confirm primitive are as follows:

```
MCPS-DATA.confirm       (
                        msduHandle,
                        status,
                        Timestamp
                        )
```

Table 42 specifies the parameters for the MCPS-DATA.confirm primitive.

7.1.1.2.2 When generated

The MCPS-DATA.confirm primitive is generated by the MAC sublayer entity in response to an MCPS-DATA.request primitive. The MCPS-DATA.confirm primitive returns a status of either SUCCESS, indicating that the request to transmit was successful, or the appropriate error code. The status values are fully described in 7.1.1.1.3 and subclauses referenced by 7.1.1.1.3.

7.1.1.2.3 Appropriate usage

On receipt of the MCPS-DATA.confirm primitive, the SSCS of the initiating device is notified of the result of its request to transmit. If the transmission attempt was successful, the status parameter will be set to SUCCESS. Otherwise, the status parameter will indicate the error.

7.1.1.3 MCPS-DATA.indication

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

Table 42—MCPS-DATA.confirm parameters

Name	Type	Valid range	Description
msduHandle	Integer	0x00–0xff	The handle associated with the MSDU being confirmed.
status	Enumeration	SUCCESS, TRANSACTION_OVERFLOW, TRANSACTION_EXPIRED, CHANNEL_ACCESS_FAILURE, INVALID_ADDRESS, INVALID_GTS, NO_ACK, COUNTER_ERROR, FRAME_TOO_LONG, KEY_ERROR, UNAVAILABLE_KEY, UNSUPPORTED_SECURITY or INVALID_PARAMETER	The status of the last MSDU transmission.
Timestamp	Integer	0x000000–0xffffff	Optional. The time, in symbols, at which the data were transmitted (see 7.5.4.1). The value of this parameter will only be considered valid if the value of the status parameter is SUCCESS; if the status parameter is not equal to SUCCESS, the value of the Timestamp parameter shall not be used for any other purpose. The symbol boundary is described by *macSyncSymbolOffset* (see Table 86). This is a 24-bit value, and the precision of this value shall be a minimum of 20 bits, with the lowest 4 bits being the least significant.

7.1.1.3.1 Semantics of the service primitive

The semantics of the MCPS-DATA.indication primitive are as follows:

```
MCPS-DATA.indication (
                     SrcAddrMode,
                     SrcPANId,
                     SrcAddr,
                     DstAddrMode,
                     DstPANId
                     DstAddr,
                     msduLength,
                     msdu,
                     mpduLinkQuality,
                     DSN,
                     Timestamp,
                     SecurityLevel,
                     KeyIdMode,
                     KeySource,
                     KeyIndex
                     )
```

Table 43 specifies the parameters for the MCPS-DATA.indication primitive.

Table 43—MCPS-DATA.indication parameters

Name	Type	Valid range	Description
SrcAddrMode	Integer	0x00–0x03	The source addressing mode for this primitive corresponding to the received MPDU. This value can take one of the following values: 0x00 = no address (addressing fields omitted). 0x01 = reserved. 0x02 = 16-bit short address. 0x03 = 64-bit extended address.
SrcPANId	Integer	0x0000–0xffff	The 16-bit PAN identifier of the entity from which the MSDU was received.
SrcAddr	Device address	As specified by the SrcAddrMode parameter	The individual device address of the entity from which the MSDU was received.
DstAddrMode	Integer	0x00–0x03	The destination addressing mode for this primitive corresponding to the received MPDU. This value can take one of the following values: 0x00 = no address (addressing fields omitted). 0x01 = reserved. 0x02 = 16-bit short device address. 0x03 = 64-bit extended device address.
DstPANId	Integer	0x0000–0xffff	The 16-bit PAN identifier of the entity to which the MSDU is being transferred.
DstAddr	Device address	As specified by the DstAddrMode parameter	The individual device address of the entity to which the MSDU is being transferred.

Table 43—MCPS-DATA.indication parameters *(continued)*

Name	Type	Valid range	Description
msduLength	Integer	≤ *aMaxMACFrame-Size*	The number of octets contained in the MSDU being indicated by the MAC sublayer entity.
msdu	Set of octets	—	The set of octets forming the MSDU being indicated by the MAC sublayer entity.
mpduLinkQuality	Integer	0x00–0xff	Link quality indication (LQI) value measured during reception of the MPDU. Lower values represent lower LQI (see 6.9.8).
DSN	Integer	0x00–0xff	The data sequence number of the received data frame.
Timestamp	Integer	0x000000–0xffffff	Optional. The time, in symbols, at which the data were received (see 7.5.4.1). The symbol boundary is described by *macSyncSymbolOffset* (see Table 86). This is a 24-bit value, and the precision of this value shall be a minimum of 20 bits, with the lowest 4 bits being the least significant.
SecurityLevel	Integer	0x00–0x07	The security level purportedly used by the received data frame (see Table 96).
KeyIdMode	Integer	0x00–0x03	The mode used to identify the key purportedly used by the originator of the received frame (see Table 97). This parameter is invalid if the SecurityLevel parameter is set to 0x00.
KeySource	Set of 0, 4 or 8 octets	As specified by the KeyIdMode parameter	The originator of the key purportedly used by the originator of the received frame (see 7.6.2.4.1). This parameter is invalid if the KeyIdMode parameter is invalid or set to 0x00.
KeyIndex	Integer	0x01–0xff	The index of the key purportedly used by the originator of the received frame (see 7.6.2.4.2). This parameter is invalid if the KeyIdMode parameter is invalid or set to 0x00.

7.1.1.3.2 When generated

The MCPS-DATA.indication primitive is generated by the MAC sublayer and issued to the SSCS on receipt of a data frame at the local MAC sublayer entity that passes the appropriate message filtering operations as described in 7.5.6.2.

7.1.1.3.3 Appropriate usage

On receipt of the MCPS-DATA.indication primitive, the SSCS is notified of the arrival of data at the device. If the primitive is received while the device is in promiscuous mode, the parameters will be set as specified in 7.5.6.5.

7.1.1.4 MCPS-PURGE.request

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

This primitive is optional for an RFD.

7.1.1.4.1 Semantics of the service primitive

The semantics of the MCPS-PURGE.request primitive are as follows:

```
MCPS-PURGE.request      (
                        msduHandle
                        )
```

Table 44 specifies the parameters for the MCPS-PURGE.request primitive.

Table 44—MCPS-PURGE.request parameters

Name	Type	Valid range	Description
msduHandle	Integer	0x00–0xff	The handle of the MSDU to be purged from the transaction queue.

7.1.1.4.2 Appropriate usage

The MCPS-PURGE.request primitive is generated by the next higher layer whenever a MSDU is to be purged from the transaction queue.

7.1.1.4.3 Effect on receipt

On receipt of the MCPS-PURGE.request primitive, the MAC sublayer attempts to find in its transaction queue the MSDU indicated by the msduHandle parameter. If an MSDU has left the transaction queue, the handle will not be found, and the MSDU can no longer be purged. If an MSDU matching the given handle is found, the MSDU is discarded from the transaction queue, and the MAC sublayer issues the MCPS-PURGE.confirm primitive with a status of SUCCESS. If an MSDU matching the given handle is not found, the MAC sublayer issues the MCPS-PURGE.confirm primitive with a status of INVALID_HANDLE.

7.1.1.5 MCPS-PURGE.confirm

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.

This primitive is optional for an RFD.

7.1.1.5.1 Semantics of the service primitive

The semantics of the MCPS-PURGE.confirm primitive are as follows:

```
MCPS-PURGE.confirm      (
                        msduHandle,
                        status
                        )
```

Table 45 specifies the parameters for the MCPS-PURGE.confirm primitive.

Table 45—MCPS-PURGE.confirm parameters

Name	Type	Valid range	Description
msduHandle	Integer	0x00–0xff	The handle of the MSDU requested to be purge from the transaction queue.
status	Enumeration	SUCCESS or INVALID_HANDLE	The status of the request to be purged an MSDU from the transaction queue.

7.1.1.5.2 When generated

The MCPS-PURGE.confirm primitive is generated by the MAC sublayer entity in response to an MCPS-PURGE.request primitive. The MCPS-PURGE.confirm primitive returns a status of either SUCCESS, indicating that the purge request was successful, or INVALID_HANDLE, indicating an error. The status values are fully described in 7.1.1.4.3.

7.1.1.5.3 Appropriate usage

On receipt of the MCPS-PURGE.confirm primitive, the next higher layer is notified of the result of its request to purge an MSDU from the transaction queue. If the purge request was successful, the status parameter will be set to SUCCESS. Otherwise, the status parameter will indicate the error.

7.1.1.6 Data service message sequence chart

Figure 30 illustrates a sequence of messages necessary for a successful data transfer between two devices. Figure 86 and Figure 87 (see 7.7) also illustrate this, including the steps taken by the PHY.

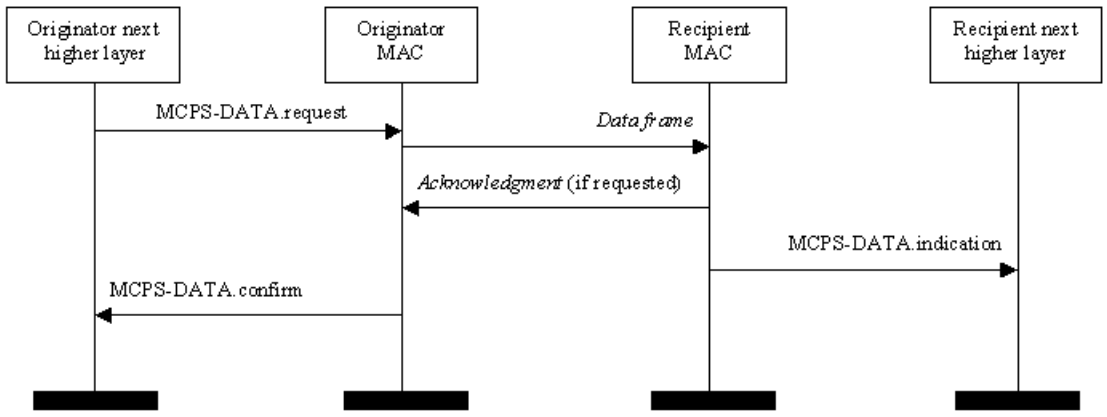


Figure 30—Message sequence chart describing the MAC data service

7.1.2 MAC management service

The MLME-SAP allows the transport of management commands between the next higher layer and the MLME. Table 46 summarizes the primitives supported by the MLME through the MLME-SAP interface. Primitives marked with a diamond (♦) are optional for an RFD. Primitives marked with an asterisk (*) are optional for both device types (i.e., RFD and FFD). The primitives are discussed in the subclauses referenced in the table.

Table 46—Summary of the primitives accessed through the MLME-SAP

Name	Request	Indication	Response	Confirm
MLME-ASSOCIATE	7.1.3.1	7.1.3.2♦	7.1.3.3♦	7.1.3.4
MLME-DISASSOCIATE	7.1.4.1	7.1.4.2		7.1.4.3
MLME-BEACON-NOTIFY		7.1.5.1		
MLME-GET	7.1.6.1			7.1.6.2
MLME-GTS	7.1.7.1*	7.1.7.3*		7.1.7.2*
MLME-ORPHAN		7.1.8.1♦	7.1.8.2♦	
MLME-RESET	7.1.9.1			7.1.9.2
MLME-RX-ENABLE	7.1.10.1*			7.1.10.2*
MLME-SCAN	7.1.11.1			7.1.11.2
MLME-COMM-STATUS		7.1.12.1		
MLME-SET	7.1.13.1			7.1.13.2
MLME-START	7.1.14.1♦			7.1.14.2♦
MLME-SYNC	7.1.15.1*			
MLME-SYNC-LOSS		7.1.15.2		
MLME-POLL	7.1.16.1			7.1.16.2

7.1.3 Association primitives

MLME-SAP association primitives define how a device becomes associated with a PAN.

All devices shall provide an interface for the request and confirm association primitives. The indication and response association primitives are optional for an RFD.

7.1.3.1 MLME-ASSOCIATE.request

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

7.1.3.1.1 Semantics of the service primitive

The semantics of the MLME-ASSOCIATE.request primitive are as follows:

```
MLME-ASSOCIATE.request          (
                                LogicalChannel,
                                ChannelPage,
                                CoordAddrMode,
                                CoordPANId,
                                CoordAddress,
                                CapabilityInformation,
                                SecurityLevel,
                                KeyIdMode,
                                KeySource,
                                KeyIndex
                                )
```

Table 47 specifies the parameters for the MLME-ASSOCIATE.request primitive.

Table 47—MLME-ASSOCIATE.request parameters

Name	Type	Valid range	Description
LogicalChannel	Integer	Selected from the available logical channels supported by the PHY (see 6.1.2).	The logical channel on which to attempt association.
ChannelPage	Integer	Selected from the available channel pages supported by the PHY (see 6.1.2).	The channel page on which to attempt association.
CoordAddrMode	Integer	0x02–0x03	The coordinator addressing mode for this primitive and subsequent MPDU. This value can take one of the following values: 2=16-bit short address. 3=64-bit extended address.
CoordPANId	Integer	0x0000–0xffff	The identifier of the PAN with which to associate.
CoordAddress	Device address	As specified by the CoordAddrMode parameter.	The address of the coordinator with which to associate.
CapabilityInformation	Bitmap	See 7.3.1.2	Specifies the operational capabilities of the associating device.

Table 47—MLME-ASSOCIATE.request parameters

Name	Type	Valid range	Description
SecurityLevel	Integer	0x00–0x07	The security level to be used (see Table 96).
KeyIdMode	Integer	0x00–0x03	The mode used to identify the key to be used (see Table 97). This parameter is ignored if the SecurityLevel parameter is set to 0x00.
KeySource	Set of 0, 4 or 8 octets	As specified by the KeyId-Mode parameter	The originator of the key to be used (see 7.6.2.4.1). This parameter is ignored if the KeyIdMode parameter is ignored or set to 0x00.
KeyIndex	Integer	0x01–0xff	The index of the key to be used (see 7.6.2.4.2). This parameter is ignored if the KeyIdMode parameter is ignored or set to 0x00.

7.1.3.1.2 Appropriate usage

The MLME-ASSOCIATE.request primitive is generated by the next higher layer of an unassociated device and issued to its MLME to request an association with a PAN through a coordinator. If the device wishes to associate through a coordinator on a beacon-enabled PAN, the MLME may optionally track the beacon of that coordinator prior to issuing this primitive.

7.1.3.1.3 Effect on receipt

On receipt of the MLME-ASSOCIATE.request primitive, the MLME of an unassociated device first updates the appropriate PHY and MAC PIB attributes and then generates an association request command (see 7.3.1), as dictated by the association procedure described in 7.5.3.1.

The SecurityLevel parameter specifies the level of security to be applied to the association request command frame. Typically, the association request command should not be implemented using security. However, if the device requesting association shares a key with the coordinator, then security may be specified in this case.

If the SecurityLevel parameter is set to a valid value other than 0x00, indicating that security is required for this frame, the MLME will set the security enabled subfield of the frame control field to one. The MAC sublayer will perform outgoing processing on the frame based on the CoordAddress, SecurityLevel, KeyIdMode, KeySource and KeyIndex parameters, as described in 7.5.8.2.1. If any error occurs during outgoing frame processing, the MLME will discard the frame and issue the MLME-ASSOCIATE.confirm primitive with the error status returned by outgoing frame processing.

If the association request command cannot be sent to the coordinator due to the CSMA-CA algorithm indicating a busy channel, the MLME will issue the MLME-ASSOCIATE.confirm primitive with a status of CHANNEL_ACCESS_FAILURE.

If the MLME successfully transmits an association request command, the MLME will expect an acknowledgment in return. If an acknowledgment is not received, the MLME will issue the MLME-ASSOCIATE.confirm primitive with a status of NO_ACK (see 7.5.6.4).

If the MLME of an unassociated device successfully receives an acknowledgment to its association request command, the MLME will wait for a response to the request (see 7.5.3.1). If the MLME of the device does not receive a response, it will issue the MLME-ASSOCIATE.confirm primitive with a status of NO_DATA.

If the MLME of the device extracts an association response command frame from the coordinator, it will then issue the MLME-ASSOCIATE.confirm primitive with a status equal to the contents of the association status field in the association response command (see 7.3.2.3).

On receipt of the association request command, the MLME of the coordinator issues the MLME-ASSOCIATE.indication primitive.

If any parameter in the MLME-ASSOCIATE.request primitive is either not supported or out of range, the MLME will issue the MLME-ASSOCIATE.confirm primitive with a status of INVALID_PARAMETER.

7.1.3.2 MLME-ASSOCIATE.indication

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

7.1.3.2.1 Semantics of the service primitive

The semantics of the MLME-ASSOCIATE.indication primitive are as follows:

```
MLME-ASSOCIATE.indication    (
                             DeviceAddress,
                             CapabilityInformation,
                             SecurityLevel,
                             KeyIdMode,
                             KeySource,
                             KeyIndex
                             )
```

Table 48 specifies the parameters for the MLME-ASSOCIATE.indication primitive.

Table 48—MLME-ASSOCIATE.indication parameters

Name	Type	Valid range	Description
DeviceAddress	Device address	An extended 64-bit IEEE address.	The address of the device requesting association.
CapabilityInformation	Bitmap	See 7.3.1.2	The operational capabilities of the device requesting association.
SecurityLevel	Integer	0x00–0x07	The security level purportedly used by the received MAC command frame (see Table 96).
KeyIdMode	Integer	0x00–0x03	The mode used to identify the key purportedly used by the originator of the received frame (see Table 97). This parameter is invalid if the SecurityLevel parameter is set to 0x00.

Table 48—MLME-ASSOCIATE.indication parameters

Name	Type	Valid range	Description
KeySource	Set of 0, 4 or 8 octets	As specified by the KeyIdMode parameter	The originator of the key purportedly used by the originator of the received frame (see 7.6.2.4.1). This parameter is invalid if the KeyIdMode parameter is invalid or set to 0x00.
KeyIndex	Integer	0x01–0xff	The index of the key purportedly used by the originator of the received frame (see 7.6.2.4.2). This parameter is invalid if the KeyIdMode parameter is invalid or set to 0x00.

7.1.3.2.2 When generated

The MLME-ASSOCIATE.indication primitive is generated by the MLME of the coordinator and issued to its next higher layer to indicate the reception of an association request command (see 7.3.1).

7.1.3.2.3 Appropriate usage

When the next higher layer of a coordinator receives the MLME-ASSOCIATE.indication primitive, the coordinator determines whether to accept or reject the unassociated device using an algorithm outside the scope of this standard. The next higher layer of the coordinator then issues the MLME-ASSOCIATE.response primitive to its MLME.

The association decision and the response should become available at the coordinator within a time of *macResponseWaitTime* (see 7.5.3.1). After this time, the device requesting association attempts to extract the association response command frame from the coordinator, using the method described in 7.5.6.3, in order to determine whether the association was successful.

7.1.3.3 MLME-ASSOCIATE.response

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

7.1.3.3.1 Semantics of the service primitive

The semantics of the MLME-ASSOCIATE.response primitive are as follows:

```
MLME-ASSOCIATE.response     (
                            DeviceAddress,
                            AssocShortAddress,
                            status,
                            SecurityLevel,
                            KeyIdMode,
                            KeySource,
                            KeyIndex
                            )
```

Table 49 specifies the parameters for the MLME-ASSOCIATE.response primitive.

Table 49—MLME-ASSOCIATE.response parameters

Name	Type	Valid range	Description
DeviceAddress	Device address	An extended 64 bit IEEE address	The address of the device requesting association.
AssocShortAddress	Integer	0x0000–0xffff	The 16-bit short device address allocated by the coordinator on successful association. This parameter is set to 0xffff if the association was unsuccessful.
status	Enumeration	See 7.3.2.3	The status of the association attempt.
SecurityLevel	Integer	0x00–0x07	The security level to be used (see Table 96).
KeyIdMode	Integer	0x00–0x03	The mode used to identify the key to be used (see Table 97). This parameter is ignored if the SecurityLevel parameter is set to 0x00.
KeySource	Set of 0, 4 or 8 octets	As specified by the KeyIdMode parameter	The originator of the key to be used (see 7.6.2.4.1). This parameter is ignored if the KeyIdMode parameter is ignored or set to 0x00.
KeyIndex	Integer	0x01–0xff	The index of the key to be used (see 7.6.2.4.2). This parameter is ignored if the KeyIdMode parameter is ignored or set to 0x00.

7.1.3.3.2 Appropriate usage

The MLME-ASSOCIATE.response primitive is generated by the next higher layer of a coordinator and issued to its MLME in order to respond to the MLME-ASSOCIATE.indication primitive.

7.1.3.3.3 Effect on receipt

When the MLME of a coordinator receives the MLME-ASSOCIATE.response primitive, it generates an association response command (see 7.3.2). The command frame is sent to the device requesting association using indirect transmission, i.e., the command frame is added to the list of pending transactions stored on the coordinator and extracted at the discretion of the device concerned using the method described in 7.5.6.3.

If the SecurityLevel parameter is set to a valid value other than 0x00, indicating that security is required for this frame, the MLME will set the security enabled subfield of the frame control field to one. The MAC sublayer will perform outgoing processing on the frame based the DeviceAddress, SecurityLevel, KeyIdMode, KeySource and KeyIndex parameters, as described in 7.5.8.2.1. If any error occurs during outgoing frame processing, the MLME will discard the frame and issue the MLME-COMM-STATUS.indication primitive with the error status returned by outgoing frame processing.

Upon receipt of the MLME-ASSOCIATE.response primitive, the coordinator attempts to add the information contained in the primitive to its list of pending transactions. If there is no capacity to store the transaction, the MAC sublayer will discard the frame and issue the MLME-COMM-STATUS.indication primitive with a status of TRANSACTION_OVERFLOW. If there is capacity to store the transaction, the coordinator will add the information to the list. If the transaction is not handled within *macTransactionPersistenceTime*, the transaction information will be discarded and the MAC sublayer will issue the MLME-COMM-STA-

TUS.indication primitive with a status of TRANSACTION_EXPIRED. The transaction handling procedure is described in 7.5.5.

If the frame was successfully transmitted and an acknowledgment was received, if requested, the MAC sublayer will issue the MLME-COMM-STATUS.indication primitive with a status of SUCCESS.

If any parameter in the MLME-ASSOCIATE.response primitive is not supported or is out of range, the MAC sublayer will issue the MLME-COMM-STATUS.indication primitive with a status of INVALID_PARAMETER.

7.1.3.4 MLME-ASSOCIATE.confirm

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.

7.1.3.4.1 Semantics of the service primitive

The semantics of the MLME-ASSOCIATE.confirm primitive are as follows:

```
MLME-ASSOCIATE.confirm      (
                            AssocShortAddress,
                            status,
                            SecurityLevel,
                            KeyIdMode,
                            KeySource,
                            KeyIndex
                            )
```

Table 50 specifies the parameters for the MLME-ASSOCIATE.confirm primitive.

7.1.3.4.2 When generated

The MLME-ASSOCIATE.confirm primitive is generated by the initiating MLME and issued to its next higher layer in response to an MLME-ASSOCIATE.request primitive. If the request was successful, the status parameter will indicate a successful association, as contained in the status field of the association response command. Otherwise, the status parameter indicates either an error code from the received association response command or the appropriate error code from Table 50. The status values are fully described in 7.1.3.1.3 and subclauses referenced by 7.1.3.1.3.

7.1.3.4.3 Appropriate usage

On receipt of the MLME-ASSOCIATE.confirm primitive, the next higher layer of the initiating device is notified of the result of its request to associate with a coordinator. If the association attempt was successful, the status parameter will indicate a successful association, as contained in the status field of the association response command, and the device will be provided with a 16-bit short address (see Table 87). If the association attempt was unsuccessful, the address will be equal to 0xffff, and the status parameter will indicate the error.

7.1.3.5 Association message sequence charts

Figure 31 illustrates a sequence of messages that may be used by a device that is not tracking the beacon of the coordinator (see 7.5.6.3) to successfully associate with a PAN. Figure 82 and Figure 83 (see 7.7) illus-

Table 50—MLME-ASSOCIATE.confirm parameters

Name	Type	Valid range	Description
AssocShortAddress	Integer	0x0000–0xffff	The short device address allocated by the coordinator on successful association. This parameter will be equal to 0xffff if the association attempt was unsuccessful.
status	Enumeration	The value of the status field of the association response command (see 7.3.2.3), SUCCESS, CHANNEL_ACCESS_FAILURE, NO_ACK, NO_DATA, COUNTER_ERROR, FRAME_TOO_LONG, IMPROPER_KEY_TYPE, IMPROPER_SECURITY_LEVEL, KEY_ERROR, SECURITY_ERROR, UNAVAILABLE_DEVICE, UNAVAILABLE_KEY, UNAVAILABLE_SECURITY_LEVEL, UNSUPPORTED_LEGACY, UNSUPPORTED_SECURITY INVALID_PARAMETER.	The status of the association attempt.

Table 50—MLME-ASSOCIATE.confirm parameters

Name	Type	Valid range	Description
SecurityLevel	Integer	0x00–0x07	If the primitive was generated following failed outgoing processing of an association request command: The security level to be used (see Table 96). If the primitive was generated following receipt of an association response command: The security level purportedly used by the received frame (see Table 96).
KeyIdMode	Integer	0x00–0x03	If the primitive was generated following failed outgoing processing of an association request command: The mode used to identify the key to be used (see Table 97). This parameter is ignored if the SecurityLevel parameter is set to 0x00. If the primitive was generated following receipt of an association response command: The mode used to identify the key purportedly used by the originator of the received frame (see Table 97). This parameter is invalid if the SecurityLevel parameter is set to 0x00.
KeySource	Set of 0, 4 or 8 octets	As specified by the KeyIdMode parameter	If the primitive was generated following failed outgoing processing of an association request command: The originator of the key to be used (see 7.6.2.4.1). This parameter is ignored if the KeyIdMode parameter is ignored or set to 0x00. If the primitive was generated following receipt of an association response command: The originator of the key purportedly used by the originator of the received frame (see 7.6.2.4.1). This parameter is invalid if the KeyIdMode parameter is invalid or set to 0x00.

Table 50—MLME-ASSOCIATE.confirm parameters

Name	Type	Valid range	Description
KeyIndex	Integer	0x01–0xff	If the primitive was generated following failed outgoing processing of an association request command: The index of the key to be used (see 7.6.2.4.2). This parameter is ignored if the KeyIdMode parameter is ignored or set to 0x00. If the primitive was generated following receipt of an association response command: The index of the key purportedly used by the originator of the received frame (see 7.6.2.4.2). This parameter is invalid if the KeyIdMode parameter is invalid or set to 0x00.

trate this same scenario, including steps taken by the PHY, for a device associating with a coordinator and for a coordinator allowing association by a device, respectively.

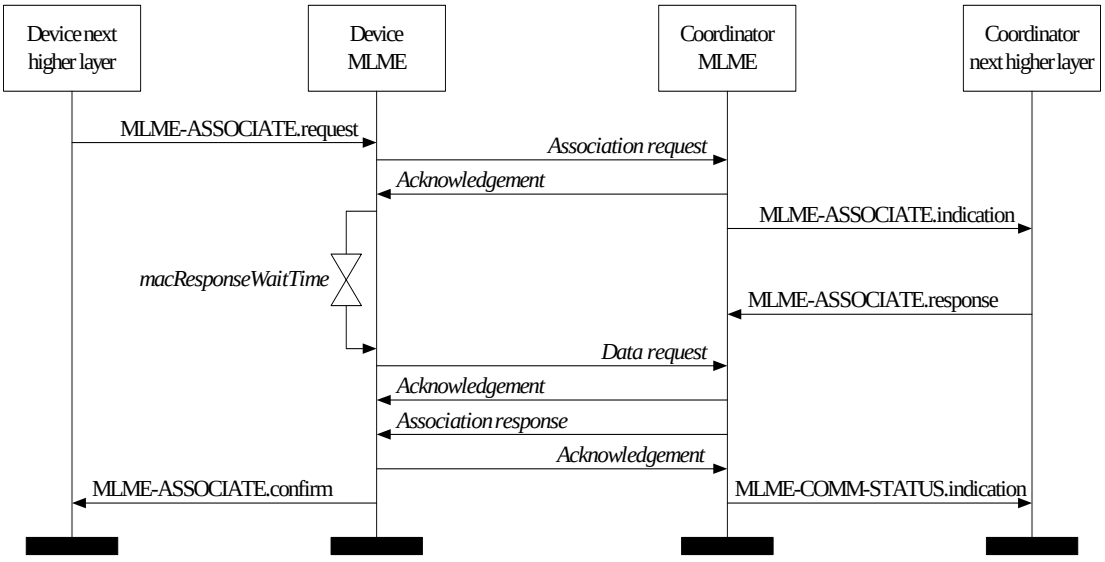


Figure 31—Message sequence chart for association

7.1.4 Disassociation primitives

The MLME-SAP disassociation primitives define how a device can disassociate from a PAN.

All devices shall provide an interface for these disassociation primitives.

7.1.4.1 MLME-DISASSOCIATE.request

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.

7.1.4.1.1 Semantics of the service primitive

The semantics of the MLME-DISASSOCIATE.request primitive are as follows:

```
MLME-DISASSOCIATE.request       (
                                DeviceAddrMode,
                                DevicePANId,
                                DeviceAddress,
                                DisassociateReason,
                                TxIndirect,
                                SecurityLevel,
                                KeyIdMode,
                                KeySource,
                                KeyIndex
                                )
```

Table 51 specifies the parameters for the MLME-DISASSOCIATE.request primitive.

Table 51—MLME-DISASSOCIATE.request parameters

Name	Type	Valid range	Description
DeviceAddrMode	Integer	0x02–0x03	The addressing mode of the device to which to send the disassociation notification command.
DevicePANId	Integer	0x0000–0xffff	The PAN identifier of the device to which to send the disassociation notification command.
DeviceAddress	Device address	As specified by the DeviceAddrMode parameter.	The address of the device to which to send the disassociation notification command.
DisassociateReason	Integer	0x00–0xff	The reason for the disassociation (see 7.3.3.2).
TxIndirect	Boolean	TRUE or FALSE	TRUE if the disassociation notification command is to be sent indirectly.

Table 51—MLME-DISASSOCIATE.request parameters

Name	Type	Valid range	Description
SecurityLevel	Integer	0x00–0x07	The security level to be used (see Table 96).
KeyIdMode	Integer	0x00–0x03	The mode used to identify the key to be used (see Table 97). This parameter is ignored if the SecurityLevel parameter is set to 0x00.
KeySource	Set of 0, 4 or 8 octets	As specified by the KeyIdMode parameter	The originator of the key to be used (see 7.6.2.4.1). This parameter is ignored if the KeyIdMode parameter is ignored or set to 0x00.
KeyIndex	Integer	0x01–0xff	The index of the key to be used (see 7.6.2.4.2). This parameter is ignored if the KeyIdMode parameter is ignored or set to 0x00.

7.1.4.1.2 Appropriate usage

The MLME-DISASSOCIATE.request primitive is generated by the next higher layer of an associated device and issued to its MLME to request disassociation from the PAN. It is also generated by the next higher layer of the coordinator and issued to its MLME to instruct an associated device to leave the PAN.

7.1.4.1.3 Effect on receipt

On receipt of the MLME-DISASSOCIATE.request primitive, the MLME compares the DevicePANId parameter with *macPANId*. If the DevicePANId parameter is not equal to *macPANId*, the MLME issues the MLME-DISASSOCIATE.confirm primitive with a status of INVALID_PARAMETER. If the DevicePANId parameter is equal to *macPANId*, the MLME evaluates the primitive address fields.

If the DeviceAddrMode parameter is equal to 0x02 and the DeviceAddress parameter is equal to *macCoordShortAddress* or if the DeviceAddrMode parameter is equal to 0x03 and the DeviceAddress parameter is equal to *macCoordExtendedAddress*, the TxIndirect parameter is ignored and the MLME sends a disassociation notification command (see 7.3.3) to its coordinator in the CAP for a beacon-enabled PAN or immediately for a nonbeacon-enabled PAN.

If the DeviceAddrMode parameter is equal to 0x02 and the DeviceAddress parameter is not equal to *macCoordShortAddress* or if the DeviceAddrMode parameter is equal to 0x03 and the DeviceAddress parameter is not equal to *macCoordExtendedAddress*, and this primitive was received by the MLME of a coordinator with the TxIndirect parameter set to TRUE, the disassociation notification command will be sent using indirect transmission, i.e. the command frame is added to the list of pending transactions stored on the coordinator and extracted at the discretion of the device concerned using the method described in 7.5.6.3.

If the DeviceAddrMode parameter is equal to 0x02 and the DeviceAddress parameter is not equal to *macCoordShortAddress* or if the DeviceAddrMode parameter is equal to 0x03 and the DeviceAddress parameter is not equal to *macCoordExtendedAddress*, and this primitive was received by the MLME of a coordinator with the TxIndirect parameter set to FALSE, the MLME sends a disassociation notification command to the device in the CAP for a beacon-enabled PAN or immediately for a nonbeacon-enabled PAN.

Otherwise, the MLME issues the MLME-DISASSOCIATE.confirm primitive with a status of INVALID_PARAMETER and does not generate a disassociation notification command.

If the disassociation notification command is to be sent using indirect transmission and there is no capacity to store the transaction, the MLME will discard the frame and issue the MLME-DISASSOCIATE.confirm primitive with a status of TRANSACTION_OVERFLOW. If there is capacity to store the transaction, the coordinator will add the information to the list. If the transaction is not handled within *macTransactionPersistenceTime*, the transaction information will be discarded and the MLME will issue the MLME-DISASSOCIATE.confirm with a status of TRANSACTION_EXPIRED. The transaction handling procedure is described in 7.5.5.

If the disassociation notification command cannot be sent due to a CSMA-CA algorithm failure, and this primitive was either received by the MLME of a coordinator with the TxIndirect parameter set to FALSE or it was received by the MLME of a device, the MLME will issue the MLME-DISASSOCIATE.confirm primitive with a status of CHANNEL_ACCESS_FAILURE.

If the SecurityLevel parameter is set to a valid value other than 0x00, indicating that security is required for this frame, the MLME will set the security enabled subfield of the frame control field to one. The MAC sublayer will perform outgoing processing on the frame based on the DeviceAddress, SecurityLevel, KeyIdMode, KeySource and KeyIndex parameters, as described in 7.5.8.2.1. If any error occurs during outgoing frame processing, the MLME will discard the frame and issue the MLME-DISASSOCIATE.confirm primitive with the error status returned by outgoing frame processing.

If the MLME successfully transmits a disassociation notification command, the MLME will expect an acknowledgment in return. If an acknowledgment is not received, and this primitive was either received by the MLME of a coordinator with the TxIndirect parameter set to FALSE or it was received by the MLME of a device, the MLME will issue the MLME-DISASSOCIATE.confirm primitive with a status of NO_ACK (see 7.5.6.4).

If the MLME successfully transmits a disassociation notification command and receives an acknowledgment in return, the MLME will issue the MLME-DISASSOCIATE.confirm primitive with a status of SUCCESS.

On receipt of the disassociation notification command, the MLME of the recipient issues the MLME-DISASSOCIATE.indication primitive.

If any parameter in the MLME-DISASSOCIATE.request primitive is not supported or is out of range, the MLME will issue the MLME-DISASSOCIATE.confirm primitive with a status of INVALID_PARAMETER.

7.1.4.2 MLME-DISASSOCIATE.indication

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

7.1.4.2.1 Semantics of the service primitive

The semantics of the MLME-DISASSOCIATE.indication primitive are as follows:

```
MLME-DISASSOCIATE.indication    (
                                DeviceAddress,
                                DisassociateReason,
                                SecurityLevel,
                                KeyIdMode,
                                KeySource,
                                KeyIndex
                                )
```

Table 52 specifies the parameters for the MLME-DISASSOCIATE.indication primitive.

Table 52—MLME-DISASSOCIATE.indication parameters

Name	Type	Valid range	Description
DeviceAddress	Device address	An extended 64-bit IEEE address	The address of the device requesting disassociation.
DisassociateReason	Integer	0x00–0xff	The reason for the disassociation (see 7.3.3.2).
SecurityLevel	Integer	0x00–0x07	The security level purportedly used by the received MAC command frame (see Table 96).
KeyIdMode	Integer	0x00–0x03	The mode used to identify the key purportedly used by the originator of the received frame (see Table 97). This parameter is invalid if the SecurityLevel parameter is set to 0x00.
KeySource	Set of 0, 4 or 8 octets	As specified by the KeyIdMode parameter	The originator of the key purportedly used by the originator of the received frame (see 7.6.2.4.1). This parameter is invalid if the KeyIdMode parameter is invalid or set to 0x00.
KeyIndex	Integer	0x01–0xff	The index of the key purportedly used by the originator of the received frame (see 7.6.2.4.2). This parameter is invalid if the KeyIdMode parameter is invalid or set to 0x00.

7.1.4.2.2 When generated

The MLME-DISASSOCIATE.indication primitive is generated by the MLME and issued to its next higher layer on receipt of a disassociation notification command.

7.1.4.2.3 Appropriate usage

The next higher layer is notified of the reason for the disassociation.

7.1.4.3 MLME-DISASSOCIATE.confirm

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

7.1.4.3.1 Semantics of the service primitive

The semantics of the MLME-DISASSOCIATE.confirm primitive are as follows:

```
MLME-DISASSOCIATE.confirm   (
                            status,
                            DeviceAddrMode,
                            DevicePANId,
                            DeviceAddress
                            )
```

Table 53 specifies the parameters for the MLME-DISASSOCIATE.confirm primitive.

Table 53—MLME-DISASSOCIATE.confirm parameters

Name	Type	Valid range	Description
status	Enumeration	SUCCESS, TRANSACTION_OVERFLOW, TRANSACTION_EXPIRED, NO_ACK, CHANNEL_ACCESS_FAILURE, COUNTER_ERROR, FRAME_TOO_LONG, KEY_ERROR, UNAVAILABLE_KEY, UNSUPPORTED_SECURITY or INVALID_PARAMETER	The status of the disassociation attempt.
DeviceAddrMode	Integer	0x02–0x03	The addressing mode of the device that has either requested disassociation or been instructed to disassociate by its coordinator.
DevicePANId	Integer	0x0000–0xffff	The PAN identifier of the device that has either requested disassociation or been instructed to disassociate by its coordinator.
DeviceAddress	Device address	As specified by the DeviceAddrMode parameter.	The address of the device that has either requested disassociation or been instructed to disassociate by its coordinator.

7.1.4.3.2 When generated

The MLME-DISASSOCIATE.confirm primitive is generated by the initiating MLME and issued to its next higher layer in response to an MLME-DISASSOCIATE.request primitive. This primitive returns a status of either SUCCESS, indicating that the disassociation request was successful, or the appropriate error code. The status values are fully described in 7.1.4.1.3 and subclauses referenced by 7.1.4.1.3.

7.1.4.3.3 Appropriate usage

On receipt of the MLME-DISASSOCIATE.confirm primitive, the next higher layer of the initiating device is notified of the result of the disassociation attempt. If the disassociation attempt was successful, the status parameter will be set to SUCCESS. Otherwise, the status parameter indicates the error.

7.1.4.4 Disassociation message sequence charts

The request to disassociate may originate either from a device or the coordinator through which the device has associated. Figure 32 illustrates the sequence of messages necessary for a device to successfully disassociate itself from the PAN.

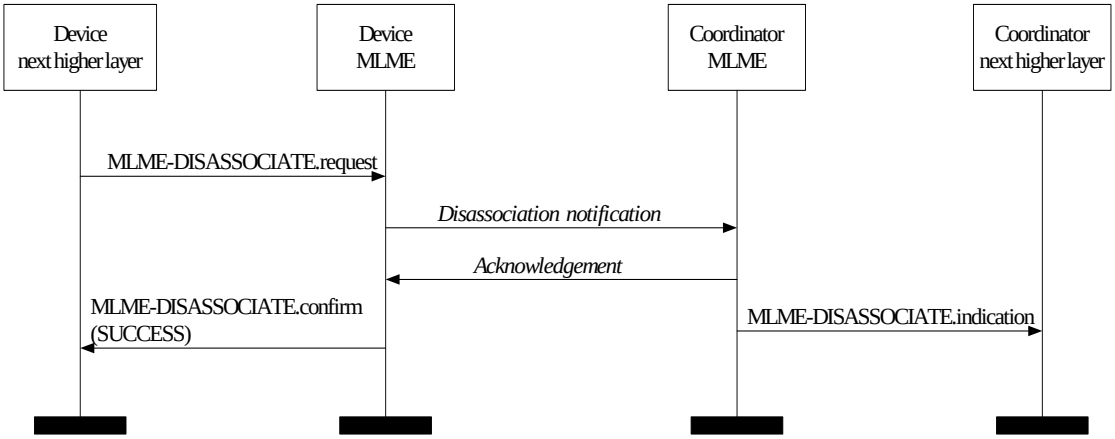


Figure 32—Message sequence chart for disassociation initiated by a device

Figure 33 illustrates the sequence necessary for a coordinator in a beacon-enabled PAN to successfully disassociate a device from its PAN using indirect transmission.

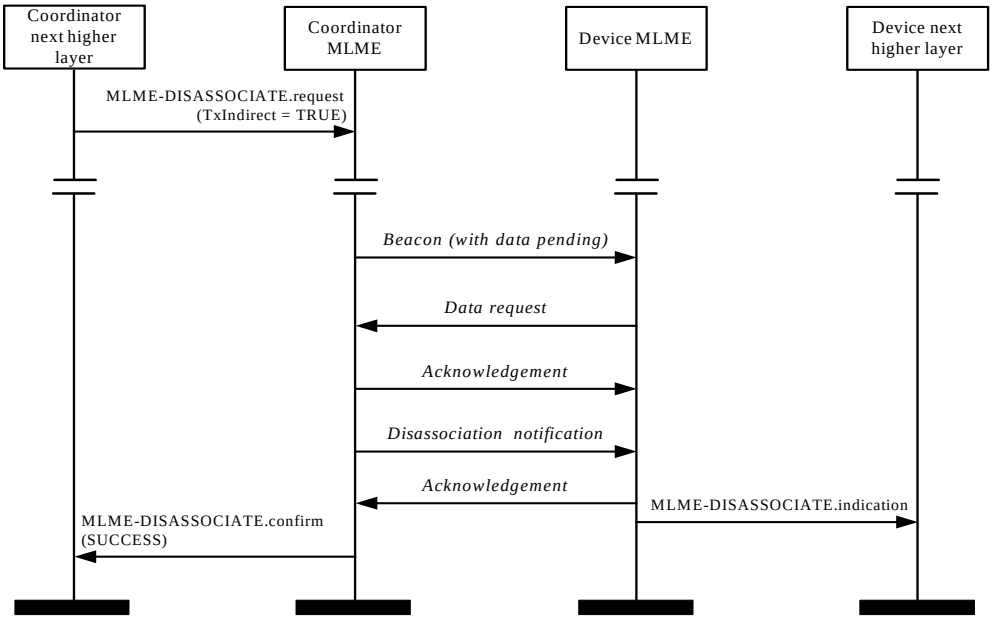


Figure 33—Message sequence chart for disassociation initiated by a coordinator, using indirect transmission, in a beacon-enabled PAN

7.1.5 Beacon notification primitive

The MLME-SAP beacon notification primitive defines how a device may be notified when a beacon is received during normal operating conditions.

All devices shall provide an interface for the beacon notification primitive.

7.1.5.1 MLME-BEACON-NOTIFY.indication

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.

7.1.5.1.1 Semantics of the service primitive

The semantics of the MLME-BEACON-NOTIFY.indication primitive are as follows:

```
MLME-BEACON-NOTIFY.indication   (
                                BSN,
                                PANDescriptor,
                                PendAddrSpec,
                                AddrList,
                                sduLength,
                                sdu
                                )
```

Table 54 specifies the parameters for the MLME-BEACON-NOTIFY.indication primitive.

Table 54—MLME-BEACON-NOTIFY.indication parameters

Name	Type	Valid range	Description
BSN	Integer	0x00–0xff	The beacon sequence number.
PANDescriptor	PANDescriptor value	See Table 55	The PANDescriptor for the received beacon.
PendAddrSpec	Bitmap	See 7.2.2.1.6	The beacon pending address specification.
AddrList	List of device addresses	—	The list of addresses of the devices for which the beacon source has data.
sduLength	Integer	0 – *aMaxBeaconPayloadLength*	The number of octets contained in the beacon payload of the beacon frame received by the MAC sublayer.
sdu	Set of octets	—	The set of octets comprising the beacon payload to be transferred from the MAC sublayer entity to the next higher layer.

Table 55 describes the elements of the PANDescriptor type.

Table 55—Elements of PANDescriptor

Name	Type	Valid range	Description
CoordAddrMode	Integer	0x02–0x03	The coordinator addressing mode corresponding to the received beacon frame. This value can take one of the following values: 2 = 16-bit short address. 3 = 64-bit extended address.
CoordPANId	Integer	0x0000–0xffff	The PAN identifier of the coordinator as specified in the received beacon frame.
CoordAddress	Device address	As specified by the CoordAddrMode parameter	The address of the coordinator as specified in the received beacon frame.
LogicalChannel	Integer	Selected from the available logical channels supported by the PHY (see 6.1.2)	The current logical channel occupied by the network.
ChannelPage	Integer	Selected from the available channel pages supported by the PHY (see 6.1.2)	The current channel page occupied by the network.
SuperframeSpec	Bitmap	See 7.2.2.1.2	The superframe specification as specified in the received beacon frame.
GTSPermit	Boolean	TRUE or FALSE	TRUE if the beacon is from the PAN coordinator that is accepting GTS requests.
LinkQuality	Integer	0x00–0xff	The LQI at which the network beacon was received. Lower values represent lower LQI (see 6.9.8).

Table 55—Elements of PANDescriptor *(continued)*

Name	Type	Valid range	Description
TimeStamp	Integer	0x000000–0xffffff	The time at which the beacon frame was received, in symbols. This value is equal to the timestamp taken when the beacon frame was received, as described in 7.5.4.1. This is a 24-bit value, and the precision of this value shall be a minimum of 20 bits, with the lowest 4 bits being the least significant.
SecurityFailure	Enumeration	SUCCESS, COUNTER_ERROR, IMPROPER_KEY_TYPE, IMPROPER_SECURITY_LEVEL, KEY_ERROR, SECURITY_ERROR, UNAVAILABLE_DEVICE, UNAVAILABLE_KEY, UNAVAILABLE_SECURITY_LEVEL, UNSUPPORTED_LEGACY, UNSUPPORTED_SECURITY	SUCCESS if there was no error in the security processing of the frame and one of the other status codes indicating an error in the security processing otherwise (see 7.5.8.2.3).
SecurityLevel	Integer	0x00–0x07	The security level purportedly used by the received beacon frame (see Table 96).
KeyIdMode	Integer	0x00–0x03	The mode used to identify the key purportedly used by the originator of the received frame (see Table 97). This parameter is invalid if the SecurityLevel parameter is set to 0x00.
KeySource	Set of 0, 4 or 8 octets	As specified by the KeyIdMode parameter	The originator of the key purportedly used by the originator of the received frame (see 7.6.2.4.1). This parameter is invalid if the KeyIdMode parameter is invalid or set to 0x00.
KeyIndex	Integer	0x01–0xff	The index of the key purportedly used by the originator of the received frame (see 7.6.2.4.2). This parameter is invalid if the KeyIdMode parameter is invalid or set to 0x00.

7.1.5.1.2 When generated

The MLME-BEACON-NOTIFY.indication primitive is generated by the MLME and issued to its next higher layer upon receipt of a beacon frame either when *macAutoRequest* is set to FALSE or when the beacon frame contains one or more octets of payload.

7.1.5.1.3 Appropriate usage

On receipt of the MLME-BEACON-NOTIFY.indication primitive, the next higher layer is notified of the arrival of a beacon frame at the MAC sublayer.

7.1.6 Primitives for reading PIB attributes

The MLME-SAP get primitives define how to read values from the PIB.

All devices shall provide an interface for these get primitives.

7.1.6.1 MLME-GET.request

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

7.1.6.1.1 Semantics of the service primitive

The semantics of the MLME-GET.request primitive are as follows:

```
MLME-GET.request        (
                        PIBAttribute,
                        PIBAttributeIndex
                        )
```

Table 56 specifies the parameters for the MLME-GET.request primitive.

Table 56—MLME-GET.request parameters

Name	Type	Valid range	Description
PIBAttribute	Integer	See Table 86 and Table 88	The identifier of the PIB attribute to read.
PIBAttributeIndex	Integer	Attribute specific; see Table 88	The index within the table of the specified PIB attribute to read. This parameter is only valid for MAC PIB attributes which are tables; it is ignored when accessing PHY PIB attributes.

7.1.6.1.2 Appropriate usage

The MLME-GET.request primitive is generated by the next higher layer and issued to its MLME to obtain information from the PIB.

7.1.6.1.3 Effect on receipt

On receipt of the MLME-GET.request primitive, the MLME checks to see if the PIB attribute is a MAC PIB attribute or PHY PIB attribute. If the requested attribute is a MAC attribute, the MLME attempts to retrieve the requested MAC PIB attribute from its database. If the identifier of the PIB attribute is not found in the database, the MLME will issue the MLME-GET.confirm primitive with a status of UNSUPPORTED_ATTRIBUTE. If the PIBAttributeIndex parameter specifies an index for a table that is out of range, the MLME will issue the MLME-GET.confirm primitive with a status of INVALID_INDEX. If the requested MAC PIB attribute is successfully retrieved, the MLME will issue the MLME-GET.confirm primitive with a status of SUCCESS.

If the requested attribute is a PHY PIB attribute, the request is passed to the PHY by issuing the PLME-GET.request primitive. Once the MLME receives the PLME-GET.confirm primitive, it will translate the received status value, since the status values used by the PHY are not the same as those used by the MLME (e.g., the status values for SUCCESS are 0x00 and 0x07 in the MAC and PHY enumeration tables, respectively). Following the translation, the MLME will issue the MLME-GET.confirm primitive to the next higher layer with the status parameter resulting from the translation and the PIBAttribute and PIBAttributeValue parameters equal to those returned by the PLME primitive.

7.1.6.2 MLME-GET.confirm

The MLME-GET.confirm primitive reports the results of an information request from the PIB.

7.1.6.2.1 Semantics of the service primitive

The semantics of the MLME-GET.confirm primitive are as follows:

```
MLME-GET.confirm        (
                        status,
                        PIBAttribute,
                        PIBAttributeIndex,
                        PIBAttributeValue
                        )
```

Table 57 specifies the parameters for the MLME-GET.confirm primitive.

Table 57—MLME-GET.confirm parameters

Name	Type	Valid range	Description
status	Enumeration	SUCCESS, UNSUPPORTED_ATTRIBUTE or INVALID_INDEX	The result of the request for PIB attribute information.
PIBAttribute	Integer	See Table 86 and Table 88	The identifier of the PIB attribute that was read.
PIBAttributeIndex	Integer	Attribute specific; see Table 88	The index within the table or array of the specified PIB attribute to read. This parameter is only valid for MAC PIB attributes which are tables or arrays; it is ignored when accessing PHY PIB attributes.
PIBAttributeValue	Various	Attribute specific; see Table 86 and Table 88	The value of the indicated PIB attribute that was read. This parameter has zero length when the status parameter is set to UNSUPPORTED_ATTRIBUTE.

7.1.6.2.2 When generated

The MLME-GET.confirm primitive is generated by the MLME and issued to its next higher layer in response to an MLME-GET.request primitive. This primitive returns a status of either SUCCESS, indicating that the request to read a PIB attribute was successful, or an error code of UNSUPPORTED_ATTRIBUTE. When an error code of UNSUPPORTED_ATTRIBUTE is returned, the PIBAttribute value parameter will be set to length zero. The status values are fully described in 7.1.6.1.3.

7.1.6.2.3 Appropriate usage

On receipt of the MLME-GET.confirm primitive, the next higher layer is notified of the results of its request to read a PIB attribute. If the request to read a PIB attribute was successful, the status parameter will be set to SUCCESS. Otherwise, the status parameter indicates the error.

7.1.7 GTS management primitives

The MLME-SAP GTS management primitives define how GTSs are requested and maintained. A device wishing to use these primitives and GTSs in general will already be tracking the beacons of its PAN coordinator.

These GTS management primitives are optional.

7.1.7.1 MLME-GTS.request

The MLME-GTS.request primitive allows 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.

7.1.7.1.1 Semantics of the service primitive

The semantics of the MLME-GTS.request primitive are as follows:

```
MLME-GTS.request        (
                        GTSCharacteristics,
                        SecurityLevel,
                        KeyIdMode,
                        KeySource,
                        KeyIndex
                        )
```

Table 58 specifies the parameters for the MLME-GTS.request primitive.

Table 58—MLME-GTS.request parameters

Name	Type	Valid range	Description
GTSCharacteristics	GTS characteristics	See 7.3.9.2	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.
SecurityLevel	Integer	0x00–0x07	The security level to be used (see Table 96).
KeyIdMode	Integer	0x00–0x03	The mode used to identify the key to be used (see Table 97). This parameter is ignored if the SecurityLevel parameter is set to 0x00.
KeySource	Set of 0, 4 or 8 octets	As specified by the KeyIdMode parameter	The originator of the key to be used (see 7.6.2.4.1). This parameter is ignored if the KeyIdMode parameter is ignored or set to 0x00.
KeyIndex	Integer	0x01–0xff	The index of the key to be used (see 7.6.2.4.2). This parameter is ignored if the KeyIdMode parameter is ignored or set to 0x00.

7.1.7.1.2 Appropriate usage

The MLME-GTS.request primitive is generated by the next higher layer of a device and issued to its MLME to request the allocation of a new GTS or to request the deallocation of an existing GTS. It is also generated

by the next higher layer of the PAN coordinator and issued to its MLME to request the deallocation of an existing GTS.

7.1.7.1.3 Effect on receipt

On receipt of the MLME-GTS.request primitive by a device, the MLME of a device attempts to generate a GTS request command (see 7.3.9) with the information contained in this primitive and, if successful, sends it to the PAN coordinator.

If *macShortAddress* is equal to 0xfffe or 0xffff, the device is not permitted to request a GTS. In this case, the MLME issues the MLME-GTS.confirm primitive containing a status of NO_SHORT_ADDRESS.

If the SecurityLevel parameter is set to a valid value other than 0x00, indicating that security is required for this frame, the MLME will set the security enabled subfield of the frame control field to one. The MAC sublayer will perform outgoing processing on the frame based on *macCoordExtendedAddress* and the SecurityLevel, KeyIdMode, KeySource and KeyIndex parameters, as described in 7.5.8.2.1. If any error occurs during outgoing frame processing, the MLME will discard the frame and issue the MLME-GTS.confirm primitive with the error status returned by outgoing frame processing.

If the GTS request command cannot be sent due to a CSMA-CA algorithm failure, the MLME will issue the MLME-GTS.confirm primitive with a status of CHANNEL_ACCESS_FAILURE.

If the MLME successfully transmits a GTS request command, the MLME will expect an acknowledgment in return. If an acknowledgment is not received, the MLME will issue the MLME-GTS.confirm primitive with a status of NO_ACK (see 7.5.6.4).

If a GTS is being allocated (see 7.5.7.2) and the request has been acknowledged, the device will wait for a confirmation via a GTS descriptor specified in a beacon frame from its PAN coordinator. If the MLME of the PAN coordinator can allocate the requested GTS, it will issue the MLME-GTS.indication primitive with the characteristics of the allocated GTS and generate a GTS descriptor with the characteristics of the allocated GTS and the 16-bit short address of the requesting device. If the MLME of the PAN coordinator cannot allocate the requested GTS, it will generate a GTS descriptor with a start slot of zero and the short address of the requesting device.

If the device receives a beacon frame from its PAN coordinator with a GTS descriptor containing a 16-bit short address that matches *macShortAddress*, the device will process the descriptor. If no descriptor for that device is received, the MLME will issue the MLME-GTS.confirm primitive with a status of NO_DATA.

If a descriptor is received that matches the characteristics requested (indicating that the PAN coordinator has approved the GTS allocation request), the MLME of the device will issue the MLME-GTS.confirm primitive with a status of SUCCESS and a GTSCharacteristics parameter with a characteristics type equal to one, indicating a GTS allocation.

If the descriptor is received with a start slot of zero (indicating that the PAN coordinator has denied the GTS allocation request), the device requesting the GTS issues the MLME-GTS.confirm primitive with a status of DENIED, indicating that the GTSCharacteristics parameter is to be ignored.

If a GTS is being deallocated (see 7.5.7.4) at the request of a device and the request has been acknowledged by the PAN coordinator, the device will issue the MLME-GTS.confirm primitive with a status of SUCCESS and a GTSCharacteristics parameter with a characteristics type equal to zero, indicating a GTS deallocation. On receipt of a GTS request command with a request type indicating a GTS deallocation, the PAN coordinator will acknowledge the frame and deallocates the GTS. The MLME of the PAN coordinator will then issue the MLME-GTS.indication primitive with the appropriate GTS characteristics. If the PAN coordinator does

not receive the deallocation request, countermeasures can be applied by the PAN coordinator to ensure consistency is maintained (see 7.5.7.6).

If the MLME of the PAN coordinator receives an MLME-GTS.request primitive indicating deallocation, the PAN coordinator will deallocate the GTS and issue the MLME-GTS.confirm primitive with a status of SUCCESS and a GTSCharacteristics parameter with a characteristics type equal to zero.

If the device receives a beacon frame from its PAN coordinator with a GTS descriptor containing a short address that matches *macShortAddress* and a start slot equal to zero, the device immediately stops using the GTS. The MLME of the device then notifies the next higher layer of the deallocation by issuing the MLME-GTS.indication primitive with a GTSCharacteristics parameter containing the characteristics of the deallocated GTS.

If any parameter in the MLME-GTS.request primitive is not supported or is out of range, the MLME will issue the MLME-GTS.confirm primitive with a status of INVALID_PARAMETER.

7.1.7.2 MLME-GTS.confirm

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

7.1.7.2.1 Semantics of the service primitive

The semantics of the MLME-GTS.confirm primitive are as follows:

```
MLME-GTS.confirm        (
                        GTSCharacteristics,
                        status
                        )
```

Table 59 specifies the parameters for the MLME-GTS.confirm primitive.

Table 59—MLME-GTS.confirm parameters

Name	Type	Valid range	Description
GTSCharacteristics	GTS characteristics	See 7.3.9.2	The characteristics of the GTS.
status	Enumeration	SUCCESS, DENIED, NO_SHORT_ADDRESS, CHANNEL_ACCESS_FAILURE, NO_ACK, NO_DATA, COUNTER_ERROR, FRAME_TOO_LONG, KEY_ERROR, UNAVAILABLE_KEY, UNSUPPORTED_SECURITYor INVALID_PARAMETER.	The status of the GTS request.

7.1.7.2.2 When generated

The MLME-GTS.confirm primitive is generated by the MLME and issued to its next higher layer in response to a previously issued MLME-GTS.request primitive.

If the request to allocate or deallocate a GTS was successful, this primitive will return a status of SUCCESS and the characteristics type field of the GTSCharacteristics parameter will have the value of one or zero, respectively. Otherwise, the status parameter will indicate the appropriate error code. The reasons for these status values are fully described in 7.1.7.1.3 and subclauses referenced by 7.1.7.1.3.

7.1.7.2.3 Appropriate usage

On receipt of the MLME-GTS.confirm primitive the next higher layer is notified of the result of its request to allocate or deallocate a GTS. If the request was successful, the status parameter will indicate a successful GTS operation. Otherwise, the status parameter will indicate the error.

7.1.7.3 MLME-GTS.indication

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

7.1.7.3.1 Semantics of the service primitive

The semantics of the MLME-GTS.indication primitive are as follows:

```
MLME-GTS.indication     (
                        DeviceAddress,
                        GTSCharacteristics,
                        SecurityLevel,
                        KeyIdMode,
                        KeySource,
                        KeyIndex
                        )
```

Table 60 specifies the parameters for the MLME-GTS.indication primitive.

7.1.7.3.2 When generated

The MLME-GTS.indication primitive is generated by the MLME of the PAN coordinator to its next higher layer whenever a GTS is allocated or deallocated following the reception of a GTS request command (see 7.3.9) by the MLME. The MLME of the PAN coordinator also generates this primitive when a GTS deallocation is initiated by the PAN coordinator itself. The characteristics type field in the GTSCharacteristics parameter will be equal to one if a GTS has been allocated or zero if a GTS has been deallocated.

This primitive is generated by the MLME of a device and issued to its next higher layer when the PAN coordinator has deallocated one of its GTSs. In this case, the characteristics type field of the GTSCharacteristics parameter is equal to zero.

7.1.7.3.3 Appropriate usage

On receipt of the MLME-GTS.indication primitive the next higher layer is notified of the allocation or deallocation of a GTS.

Table 60—MLME-GTS.indication parameters

Name	Type	Valid range	Description
DeviceAddress	Device address	0x0000–0xfffd	The 16-bit short address of the device that has been allocated or deallocated a GTS.
GTSCharacteristics	GTS characteristics	See 7.3.9.2	The characteristics of the GTS.
SecurityLevel	Integer	0x00–0x07	If the primitive was generated when a GTS deallocation is initiated by the PAN coordinator itself, the security level to be used is set to 0x00. If the primitive was generated whenever a GTS is allocated or deallocated following the reception of a GTS request command: The security level purportedly used by the received MAC command frame (see Table 96).
KeyIdMode	Integer	0x00–0x03	If the primitive was generated when a GTS deallocation is initiated by the PAN coordinator itself, this parameter is ignored. If the primitive was generated whenever a GTS is allocated or deallocated following the reception of a GTS request command: The mode used to identify the key purportedly used by the originator of the received frame (see Table 97). This parameter is invalid if the SecurityLevel parameter is set to 0x00.
KeySource	Set of 0, 4 or 8 octets	As specified by the KeyIdMode parameter	If the primitive was generated when a GTS deallocation is initiated by the PAN coordinator itself, this parameter is ignored. If the primitive was generated whenever a GTS is allocated or deallocated following the reception of a GTS request command: The originator of the key purportedly used by the originator of the received frame (see 7.6.2.4.1). This parameter is invalid if the KeyIdMode parameter is invalid or set to 0x00.
KeyIndex	Integer	0x01–0xff	If the primitive was generated when a GTS deallocation is initiated by the PAN coordinator itself, this parameter is ignored. If the primitive was generated whenever a GTS is allocated or deallocated following the reception of a GTS request command: The index of the key purportedly used by the originator of the received frame (see 7.6.2.4.2). This parameter is invalid if the KeyIdMode parameter is invalid or set to 0x00.

7.1.7.4 GTS management message sequence charts

Figure 34 and Figure 35 illustrate the sequence of messages necessary for successful GTS management. The first depicts the message flow for the case in which the device initiates the GTS allocation. The second depicts the message flow for the two cases for which a GTS deallocation occurs, first, by a device (a) and, second, by the PAN coordinator (b).

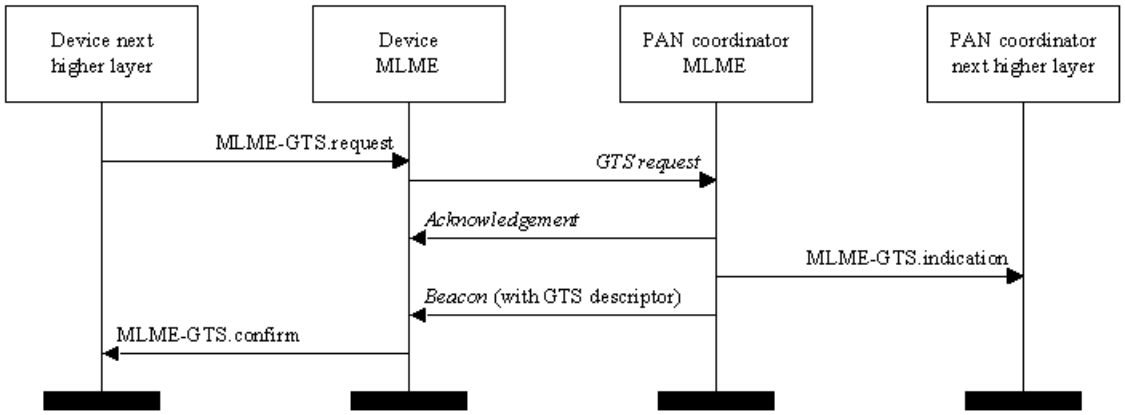


Figure 34—Message sequence chart for GTS allocation initiated by a device

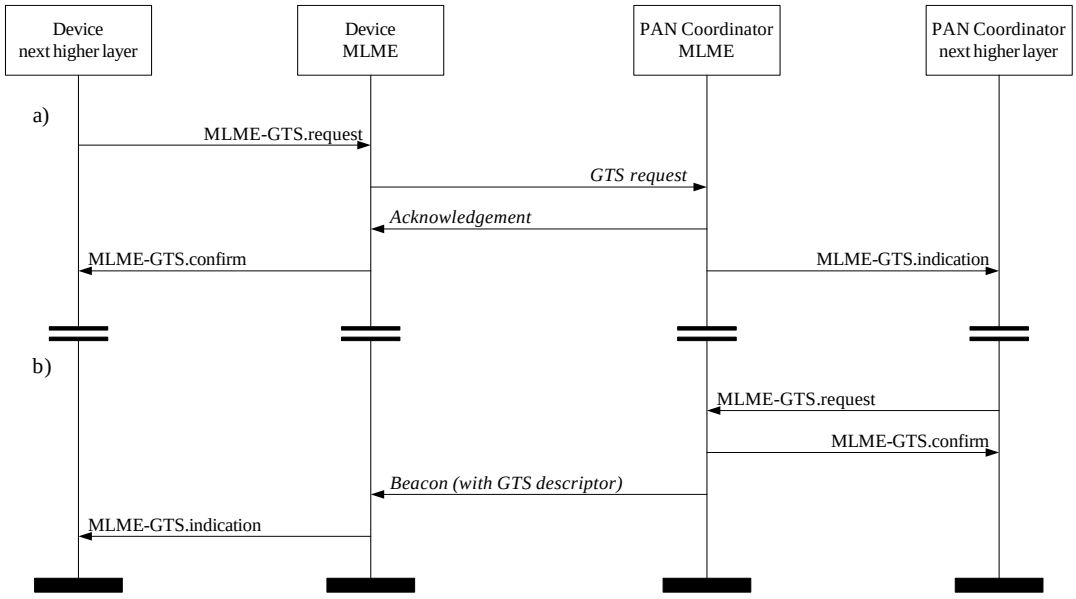


Figure 35—Message sequence chart for GTS deallocation initiated by a device (a) and the PAN coordinator (b)

7.1.8 Primitives for orphan notification

MLME-SAP orphan notification primitives define how a coordinator can issue a notification of an orphaned device.

These orphan notification primitives are optional for an RFD.

7.1.8.1 MLME-ORPHAN.indication

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

7.1.8.1.1 Semantics of the service primitive

The semantics of the MLME-ORPHAN.indication primitive are as follows:

```
MLME-ORPHAN.indication          (
                                OrphanAddress,
                                SecurityLevel,
                                KeyIdMode,
                                KeySource,
                                KeyIndex
                                )
```

Table 61 specifies the parameters for the MLME-ORPHAN.indication primitive.

Table 61—MLME-ORPHAN.indication parameters

Name	Type	Valid range	Description
OrphanAddress	Device address	Extended 64-bit IEEE address	The address of the orphaned device.
SecurityLevel	Integer	0x00–0x07	The security level purportedly used by the received MAC command frame (see Table 96).
KeyIdMode	Integer	0x00–0x03	The mode used to identify the key purportedly used by the originator of the received frame (see Table 97). This parameter is invalid if the SecurityLevel parameter is set to 0x00.
KeySource	Set of 0, 4 or 8 octets	As specified by the KeyIdMode parameter	The originator of the key purportedly used by the originator of the received frame (see 7.6.2.4.1). This parameter is invalid if the KeyIdMode parameter is invalid or set to 0x00.
KeyIndex	Integer	0x01–0xff	The index of the key purportedly used by the originator of the received frame (see 7.6.2.4.2). This parameter is invalid if the KeyIdMode parameter is invalid or set to 0x00.

7.1.8.1.2 When generated

The MLME-ORPHAN.indication primitive is generated by the MLME of a coordinator and issued to its next higher layer on receipt of an orphan notification command (see 7.3.6).

7.1.8.1.3 Appropriate usage

The effect on receipt of the MLME-ORPHAN.indication primitive is that the next higher layer is notified of the orphaned device. The next higher layer then determines whether the device was previously associated and issues the MLME-ORPHAN.response primitive to the MLME with its decision (see 7.5.2.1.4).

If the device was previously associated with the coordinator, it will send the MLME-ORPHAN.response primitive with the AssociatedMember parameter set to TRUE and the ShortAddress parameter set to the corresponding 16-bit short address allocated to the orphaned device. If the device was not previously associated with the coordinator, it will send the MLME-ORPHAN.response primitive with the AssociatedMember parameter set to FALSE.

7.1.8.2 MLME-ORPHAN.response

The MLME-ORPHAN.response primitive allows the next higher layer of a coordinator to respond to the MLME-ORPHAN.indication primitive.

7.1.8.2.1 Semantics of the service primitive

The semantics of the MLME-ORPHAN.response primitive are as follows:

```
MLME-ORPHAN.response    (
                        OrphanAddress,
                        ShortAddress,
                        AssociatedMember,
                        SecurityLevel,
                        KeyIdMode,
                        KeySource,
                        KeyIndex
                        )
```

Table 62 specifies the parameters for the MLME-ORPHAN.response primitive.

Table 62—MLME-ORPHAN.response parameters

Name	Type	Valid range	Description
OrphanAddress	Device address	Extended 64-bit IEEE address	The address of the orphaned device.
ShortAddress	Integer	0x0000–0xffff	The 16-bit short address allocated to the orphaned device if it is associated with this coordinator. The special short address 0xfffe indicates that no short address was allocated, and the device will use its 64-bit extended address in all communications. If the device was not associated with this coordinator, this field will contain the value 0xffff and be ignored on receipt.
AssociatedMember	Boolean	TRUE or FALSE	TRUE if the orphaned device is associated with this coordinator or FALSE otherwise.
SecurityLevel	Integer	0x00–0x07	The security level to be used (see Table 96).
KeyIdMode	Integer	0x00–0x03	The mode used to identify the key to be used (see Table 97). This parameter is ignored if the SecurityLevel parameter is set to 0x00.

Table 62—MLME-ORPHAN.response parameters *(continued)*

Name	Type	Valid range	Description
KeySource	Set of 0, 4 or 8 octets	As specified by the KeyIdMode parameter	The originator of the key to be used (see 7.6.2.4.1). This parameter is ignored if the KeyIdMode parameter is ignored or set to 0x00.
KeyIndex	Integer	0x01–0xff	The index of the key to be used (see 7.6.2.4.2). This parameter is ignored if the KeyIdMode parameter is ignored or set to 0x00.

7.1.8.2.2 Appropriate usage

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.

7.1.8.2.3 Effect on receipt

If the AssociatedMember parameter is set to TRUE, the orphaned device is associated with the coordinator. In this case, the MLME generates and sends the coordinator realignment command (see 7.3.8) to the orphaned device containing the value of the ShortAddress field. This command is sent in the CAP if the coordinator is on a beacon-enabled PAN or immediately otherwise. If the AssociatedMember parameter is set to FALSE, the orphaned device is not associated with the coordinator and this primitive will be ignored. If the orphaned device does not receive the coordinator realignment command following its orphan notification within *macResponseWaitTime* symbols, it will assume it is not associated to any coordinator in range.

If the SecurityLevel parameter is set to a valid value other than 0x00, indicating that security is required for this frame, the MLME will set the security enabled subfield of the frame control field to one. The MAC sublayer will perform outgoing processing on the frame based on the OrphanAddress, SecurityLevel, KeyIdMode, KeySource and KeyIndex parameters, as described in 7.5.8.2.1. If any error occurs during outgoing frame processing, the MLME will discard the frame and issue the MLME-COMM-STATUS.indication primitive with the error status returned by outgoing frame processing.

If the CSMA-CA algorithm failed due to adverse conditions on the channel, the MAC sublayer will discard the frame and issue the MLME-COMM-STATUS.indication primitive with a status of CHANNEL_ACCESS_FAILURE.

The MAC sublayer enables its receiver immediately following the transmission of the frame and waits for an acknowledgment from the recipient (see 7.5.6.4). If the MAC sublayer does not receive an acknowledgment from the recipient, it will discard the frame and issue the MLME-COMM-STATUS.indication primitive with a status of NO_ACK.

If the frame was successfully transmitted and an acknowledgment was received, if requested, the MAC sublayer will issue the MLME-COMM-STATUS.indication primitive with a status of SUCCESS.

If any parameter in the MLME-ORPHAN.response primitive is not supported or is out of range, the MAC sublayer will issue the MLME-COMM-STATUS.indication primitive with a status of INVALID_PARAMETER.

7.1.8.3 Orphan notification message sequence chart

Figure 36 illustrates the sequence of messages necessary for a coordinator to issue a notification of an orphaned device.

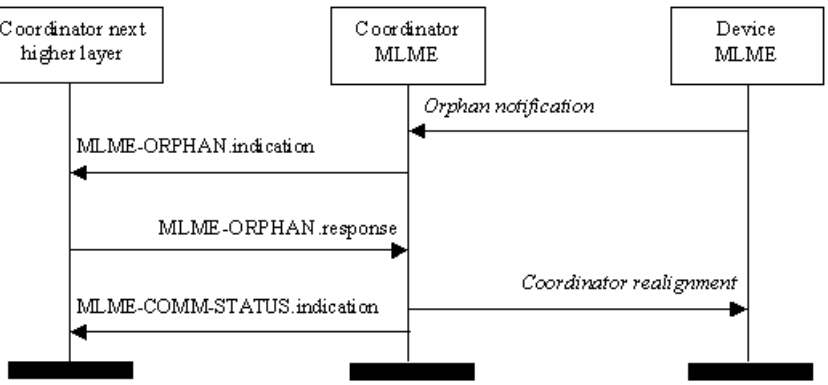


Figure 36—Message sequence chart for orphan notification

7.1.9 Primitives for resetting the MAC sublayer

MLME-SAP reset primitives specify how to reset the MAC sublayer to its default values.

All devices shall provide an interface for these reset primitives.

7.1.9.1 MLME-RESET.request

The MLME-RESET.request primitive allows the next higher layer to request that the MLME performs a reset operation.

7.1.9.1.1 Semantics of the service primitive

The semantics of the MLME-RESET.request primitive are as follows:

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

Table 63 specifies the parameter for the MLME-RESET.request primitive.

Table 63—MLME-RESET.request parameter

Name	Type	Valid range	Description
SetDefaultPIB	Boolean	TRUE or FALSE	If TRUE, the MAC sublayer is reset and all MAC PIB attributes are set to their default values. If FALSE, the MAC sublayer is reset but all MAC PIB attributes retain their values prior to the generation of the MLME-RESET.request primitive.

7.1.9.1.2 Appropriate usage

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 prior to the use of the MLME-START.request or the MLME-ASSOCIATE.request primitives.

7.1.9.1.3 Effect on receipt

On receipt of the MLME-RESET.request primitive, the MLME issues the PLME-SET-TRX-STATE.request primitive with a state of FORCE_TRX_OFF. On receipt of the PLME-SET-TRX-STATE.confirm primitive, the MAC sublayer is then set to its initial conditions, clearing all internal variables to their default values. If the SetDefaultPIB parameter is set to TRUE, the MAC PIB attributes are set to their default values.

The MLME-RESET.confirm primitive with a status of SUCCESS is issued on completion.

7.1.9.2 MLME-RESET.confirm

The MLME-RESET.confirm primitive reports the results of the reset operation.

7.1.9.2.1 Semantics of the service primitive

The semantics of the MLME-RESET.confirm primitive are as follows:

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

Table 64 specifies the parameter for the MLME-RESET.confirm primitive.

Table 64—MLME-RESET.confirm parameter

Name	Type	Valid range	Description
status	Enumeration	SUCCESS	The result of the reset operation.

7.1.9.2.2 When generated

The MLME-RESET.confirm primitive is generated by the MLME and issued to its next higher layer in response to an MLME-RESET.request primitive and following the receipt of the PLME-SET-TRX-STATE.confirm primitive.

7.1.9.2.3 Appropriate usage

On receipt of the MLME-RESET.confirm primitive, the next higher layer is notified of its request to reset the MAC sublayer. This primitive returns a status of SUCCESS indicating that the request to reset the MAC sublayer was successful.

7.1.10 Primitives for specifying the receiver enable time

MLME-SAP receiver state primitives define how a device can enable or disable the receiver at a given time.

These receiver state primitives are optional.

7.1.10.1 MLME-RX-ENABLE.request

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

7.1.10.1.1 Semantics of the service primitive

The semantics of the MLME-RX-ENABLE.request primitive are as follows:

```
MLME-RX-ENABLE.request      (
                            DeferPermit,
                            RxOnTime,
                            RxOnDuration
                            )
```

Table 65 specifies the parameters for the MLME-RX-ENABLE.request primitive.

Table 65—MLME-RX-ENABLE.request parameters

Name	Type	Valid range	Description
DeferPermit	Boolean	TRUE or FALSE	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. If the issuing device is the PAN coordinator, the term superframe refers to its own superframe. Otherwise, the term refers to the superframe of the coordinator through which the issuing device is associated.
RxOnTime	Integer	0x000000–0xffffff	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, and the precision of this value shall be a minimum of 20 bits, with the lowest 4 bits being the least significant. This parameter is ignored for nonbeacon-enabled PANs. If the issuing device is the PAN coordinator, the term superframe refers to its own superframe. Otherwise, the term refers to the superframe of the coordinator through which the issuing device is associated.
RxOnDuration	Integer	0x000000–0xffffff	The number of symbols for which the receiver is to be enabled. If this parameter is equal to 0x000000, the receiver is to be disabled.

7.1.10.1.2 Appropriate usage

The MLME-RX-ENABLE.request primitive is generated by the next higher layer and issued to the MLME to enable the receiver for a fixed duration, at a time relative to the start of the current or next superframe on a beacon-enabled PAN or immediately on a nonbeacon-enabled PAN. This primitive may also be generated

to cancel a previously generated request to enable the receiver. The receiver is enabled or disabled exactly once per primitive request.

7.1.10.1.3 Effect on receipt

The MLME will treat the request to enable or disable the receiver as secondary to other responsibilities of the device (e.g. GTSs, coordinator beacon tracking, beacon transmissions). When the primitive is issued to enable the receiver, the device will enable its receiver until either the device has a conflicting responsibility or the time specified by RxOnDuration has expired. In the case of a conflicting responsibility, the device will interrupt the receive operation. After the completion of the interrupting operation, the RxOnDuration will be checked to determine whether the time has expired. If so, the operation is complete. If not, the receiver is re-enabled until either the device has another conflicting responsibility or the time specified by RxOnDuration has expired. When the primitive is issued to disable the receiver, the device will disable its receiver unless the device has a conflicting responsibility.

On a nonbeacon-enabled PAN, the MLME ignores the DeferPermit and RxOnTime parameters and requests that the PHY enable or disable the receiver immediately. If the request is to enable the receiver, the receiver will remain enabled until RxOnDuration symbols have elapsed.

Before attempting to enable the receiver on a beacon-enabled PAN, the MLME first determines whether (RxOnTime + RxOnDuration) is less than the beacon interval, as defined by *macBeaconOrder*. If (RxOnTime + RxOnDuration) is not less than the beacon interval, the MLME issues the MLME-RX-ENABLE.confirm primitive with a status of ON_TIME_TOO_LONG.

The MLME then determines whether the receiver can be enabled in the current superframe. The PAN coordinator issuing this primitive makes the determination based on its own superframe. A device which is not the PAN coordinator makes the determination based on the superframe of the coordinator through which it is associated. If the current number of symbols measured from the start of the superframe is less than (RxOnTime – *aTurnaroundTime*), the MLME attempts to enable the receiver in the current superframe. If the current number of symbols measured from the start of the superframe is greater than or equal to (RxOnTime – *aTurnaroundTime*) and DeferPermit is equal to TRUE, the MLME defers until the next superframe and attempts to enable the receiver in that superframe. Otherwise if the MLME cannot enable the receiver in the current superframe and is not permitted to defer the receive operation until the next superframe, the MLME issues the MLME-RX-ENABLE.confirm primitive with a status of PAST_TIME.

If the RxOnDuration parameter is equal to zero, the MLME requests that the PHY disable its receiver.

If any parameter in the MLME-RX-ENABLE.request primitive is not supported or is out of range, the MAC sublayer will issue the MLME-RX-ENABLE.confirm primitive with a status of INVALID_PARAMETER.

If the request to enable or disable the receiver was successful, the MLME issues the MLME-RX-ENABLE.confirm primitive with a status of SUCCESS.

7.1.10.2 MLME-RX-ENABLE.confirm

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

7.1.10.2.1 Semantics of the service primitive

The semantics of the MLME-RX-ENABLE.confirm primitive are as follows:

```
MLME-RX-ENABLE.confirm      (
                            status
                            )
```

Table 66 specifies the parameter for the MLME-RX-ENABLE.confirm primitive.

Table 66—MLME-RX-ENABLE.confirm parameter

Name	Type	Valid range	Description
status	Enumeration	SUCCESS, PAST_TIME, ON_TIME_TOO_LONG or INVALID_PARAMETER	The result of the request to enable or disable the receiver.

7.1.10.2.2 When generated

The MLME-RX-ENABLE.confirm primitive is generated by the MLME and issued to its next higher layer in response to an MLME-RX-ENABLE.request primitive.

7.1.10.2.3 Appropriate usage

On receipt of the MLME-RX-ENABLE.confirm primitive, the next higher layer is notified of its request to enable or disable the receiver. This primitive returns a status of either SUCCESS, if the request to enable or disable the receiver was successful, or the appropriate error code. The status values are fully described in 7.1.10.1.3.

7.1.10.3 Message sequence chart for changing the state of the receiver

Figure 37 illustrates the sequence of messages necessary for enabling the receiver for a fixed duration when the device does not have any conflicting responsibilities. Part a) illustrates the case for a beacon-enabled PAN where it is assumed that both the MLME-RX-ENABLE.request has been received by the MLME without sufficient time available to enable the receiver in the current superframe and that the DeferPermit parameter is TRUE. Part b) illustrates the case for a nonbeacon-enabled PAN where the receiver is enabled immediately.

7.1.11 Primitives for channel scanning

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

All devices shall provide an interface for these scan primitives.

7.1.11.1 MLME-SCAN.request

The MLME-SCAN.request primitive is used to initiate a channel scan over a given list of channels. A device can use a 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.

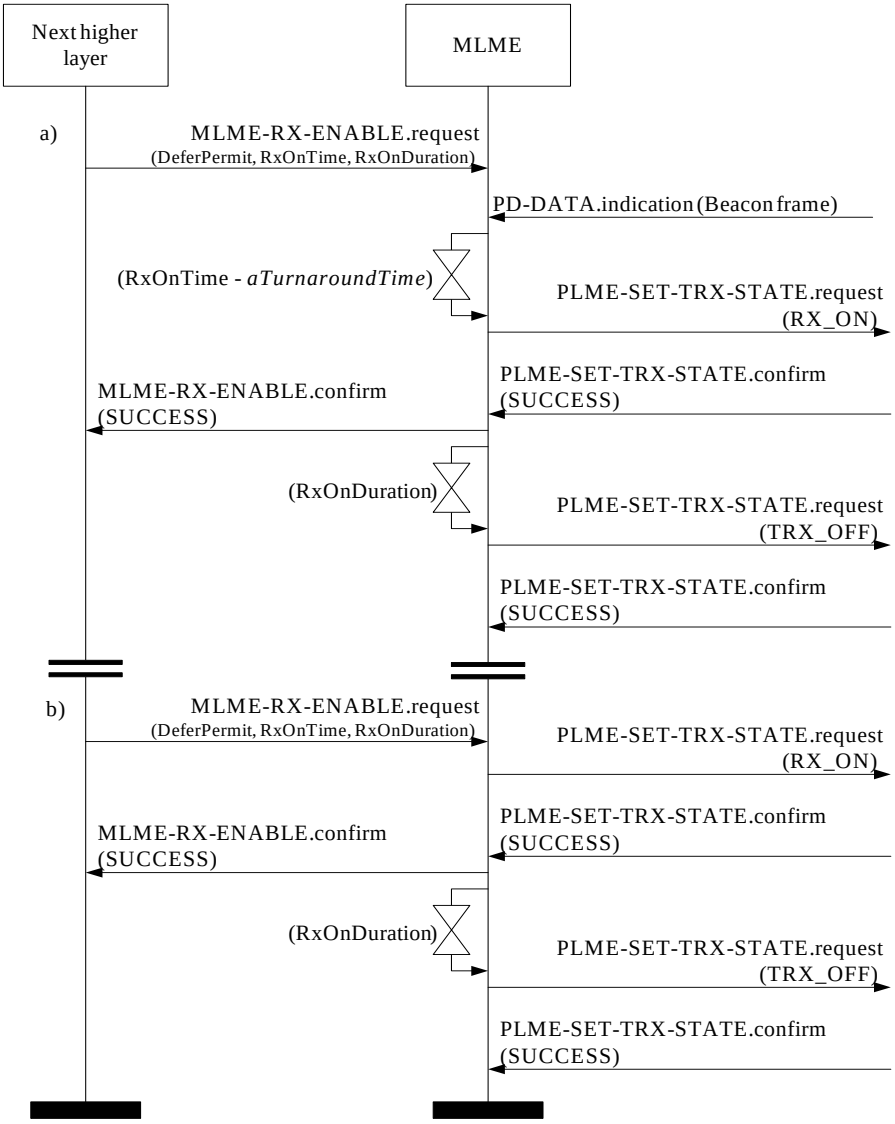


Figure 37—Message sequence chart for changing the state of the receiver

7.1.11.1.1 Semantics of the service primitive

The semantics of the MLME-SCAN.request primitive are as follows:

```
MLME-SCAN.request       (
                        ScanType,
                        ScanChannels,
                        ScanDuration,
                        ChannelPage,
                        SecurityLevel,
                        KeyIdMode,
                        KeySource,
                        KeyIndex
                        )
```

Table 67 specifies the parameters for the MLME-SCAN.request primitive.

Table 67—MLME-SCAN.request parameters

Name	Type	Valid range	Description
ScanType	Integer	0x00–0x03	Indicates the type of scan performed: 0x00 = ED scan (optional for RFD). 0x01 = active scan (optional for RFD). 0x02 = passive scan. 0x03 = orphan scan.
ScanChannels	Bitmap	27 bit field	The 27 bits (b_0, b_1,... b_{26}) indicate which channels are to be scanned (1 = scan, 0 = do not scan) for each of the 27 channels supported by the ChannelPage parameter.
ScanDuration	Integer	0–14	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. The time spent scanning each channel is [*aBaseSuperframeDuration* * (2^n + 1)] symbols, where *n* is the value of the ScanDuration parameter.
ChannelPage	Integer	0–31	The channel page on which to perform the scan (see 6.1.2).
SecurityLevel	Integer	0x00–0x07	The security level to be used (see Table 96).
KeyIdMode	Integer	0x00–0x03	The mode used to identify the key to be used (see Table 97). This parameter is ignored if the SecurityLevel parameter is set to 0x00.
KeySource	Set of 0, 4 or 8 octets	As specified by the KeyIdMode parameter	The originator of the key to be used (see 7.6.2.4.1). This parameter is ignored if the KeyIdMode parameter is ignored or set to 0x00.
KeyIndex	Integer	0x01–0xff	The index of the key to be used (see 7.6.2.4.2). This parameter is ignored if the KeyIdMode parameter is ignored or set to 0x00.

7.1.11.1.2 Appropriate usage

The MLME-SCAN.request primitive is generated by the next higher layer and issued to its MLME to initiate a channel scan to search for activity within the POS of the device. This primitive can be used to perform an ED scan to determine channel usage, an active or passive scan to locate beacon frames containing any PAN identifier, or an orphan scan to locate a PAN to which the device is currently associated. See 7.5.2.1 for a description of each type of scan in detail.

All devices shall be capable of performing passive scans and orphan scans; ED scans and active scans are optional for an RFD. However, an RFD may support active scanning to participate in a nonbeacon-enabled network.

7.1.11.1.3 Effect on receipt

If the MLME receives the MLME-SCAN.request primitive while performing a previously initiated scan operation, it issues the MLME-SCAN.confirm primitive with a status of SCAN_IN_PROGRESS. Otherwise, the MLME initiates a scan in all channels specified in the ScanChannels parameter.

The ED scan is performed on each channel by the MLME repeatedly issuing the PLME-ED.request primitive to the PHY until [*aBaseSuperframeDuration* * (2^n + 1)] symbols, where *n* is the value of the ScanDuration parameter, have elapsed. The MLME notes the maximum energy measurement and moves on to the next channel in the channel list. See 7.5.2.1.1 for more detailed information on the ED channel scan procedure.

The active scan is performed on each channel by the MLME first sending a beacon request command (see 7.3.7). The MLME then enables the receiver and records the information contained in each received beacon in a PAN descriptor structure (see Table 55 in 7.1.5.1.1). The active scan on a particular channel terminates when the number of PAN descriptors stored equals an implementation-specified maximum or when [*aBaseSuperframeDuration* * (2^n + 1)] symbols, where *n* is the value of the ScanDuration parameter, have elapsed, whichever comes first. See 7.5.2.1.2 for more detailed information on the active channel scan procedure.

The passive scan is performed on each channel by the MLME enabling its receiver and recording the information contained in each received beacon in a PAN descriptor structure (see Table 55 in 7.1.5.1.1). The passive scan on a particular channel terminates when the number of PAN descriptors stored equals an implementation-specified maximum or when [*aBaseSuperframeDuration* * (2^n + 1)] symbols, where *n* is the value of the ScanDuration parameter, have elapsed, whichever comes first. See 7.5.2.1.3 for more detailed information on the passive channel scan procedure.

The orphan scan is performed on each channel by the MLME first sending an orphan notification command (see 7.3.6). If the device receives a coordinator realignment command in return, the MLME will disable its receiver. Otherwise, the scan is repeated on the next channel in the channel list. See 7.5.2.1.4 for more detailed information on the orphan channel scan procedure.

The SecurityLevel, KeyIdMode, KeySource and KeyIndex parameters are only used in an orphan scan.

If the SecurityLevel parameter is set to a valid value other than 0x00, indicating that security is required for this frame, the MLME will set the security enabled subfield of the frame control field to one. The MAC sublayer will perform outgoing processing on the frame based on *macCoordExtendedAddress*, the SecurityLevel, KeyIdMode, KeySource and KeyIndex parameters, as described in 7.5.8.2.1. If any error occurs during outgoing frame processing, the MLME will discard the frame and issue the MLME-SCAN.confirm primitive with the error status returned by outgoing frame processing.

The results of an ED scan are recorded in a list of ED values and reported to the next higher layer through the MLME-SCAN.confirm primitive with a status of SUCCESS.

The results of an active or passive scan are reported to the next higher layer through the MLME-SCAN.confirm primitive. If the scan is successful and *macAutoRequest* is set to TRUE, the primitive results will include a set of PAN descriptor values. If the scan is successful and *macAutoRequest* is set to FALSE, the primitive results will contain a null set of PAN descriptor values; each PAN descriptor value will be sent individually to the next higher layer using separate MLME-BEACON-NOTIFY (see 7.1.5.1) primitives. In both cases, the MLME-SCAN.confirm primitive will contain a list of unscanned channels and a status of SUCCESS.

If, during an active scan, the MLME is unable to transmit a beacon request command on a channel specified by the ScanChannels parameter due to a channel access failure, the channel will appear in the list of unscanned channels returned by the MLME-SCAN.confirm primitive. If the MLME was able to send a bea-

con request command on at least one of the channels but no beacons were found, the MLME-SCAN.confirm primitive will contain a null set of PAN descriptor values, regardless of the value of *macAutoRequest*, and a status of NO_BEACON.

The results of an orphan scan are reported to the next higher layer through the MLME-SCAN.confirm primitive. If successful, the MLME-SCAN.confirm primitive will contain a status of SUCCESS. If the MLME is unable to transmit an orphan notification command on a channel specified by the ScanChannels parameter due to a channel access failure, the channel will appear in the list of unscanned channels returned by the MLME-SCAN.confirm primitive. If the MLME was able to send an orphan notification command on at least one of the channels but the device did not receive a coordinator realignment command, the MLME-SCAN.confirm primitive will contain a status of NO_BEACON. In this case, the PANDescriptorList and EnergyDetectList parameters of the MLME-SCAN.confirm primitive are null.

If during an ED, active or passive scan, the implementation-specified maximum is reached thus terminating the scan procedure, the MAC sublayer will issue the MLME-SCAN.confirm primitive with a status of LIMIT_REACHED.

If any parameter in the MLME-SCAN.request primitive is not supported or is out of range, the MAC sublayer will issue the MLME-SCAN.confirm primitive with a status of INVALID_PARAMETER.

7.1.11.2 MLME-SCAN.confirm

The MLME-SCAN.confirm primitive reports the result of the channel scan request.

7.1.11.2.1 Semantics of the service primitive

The semantics of the MLME-SCAN.confirm primitive are as follows:

```
MLME-SCAN.confirm (
                  status,
                  ScanType,
                  ChannelPage,
                  UnscannedChannels,
                  ResultListSize,
                  EnergyDetectList,
                  PANDescriptorList
                  )
```

Table 68 specifies the parameters for the MLME-SCAN.confirm primitive.

7.1.11.2.2 When generated

The MLME-SCAN.confirm primitive is generated by the MLME and issued to its next higher layer when the channel scan initiated with the MLME-SCAN.request primitive has completed. If the MLME-SCAN.request primitive requested an active, passive, or orphan scan, the EnergyDetectList parameter will be null. If the MLME-SCAN.request primitive requested an ED or orphan scan, the PANDescriptorList parameter will be null; this is also the case if the MLME-SCAN.request primitive requested an active or passive scan with *macAutoRequest* set to FALSE. If the MLME-SCAN.request primitive requested an orphan scan, the ResultListSize parameter will be set to zero.

The MLME-SCAN.confirm primitive returns a status of either SUCCESS, indicating that the requested scan was successful, or the appropriate error code. The status values are fully described in 7.1.11.1.3 and subclauses referenced by 7.1.11.1.3.

Table 68—MLME-SCAN.confirm parameters

Name	Type	Valid range	Description
status	Enumeration	SUCCESS, LIMIT_REACHED, NO_BEACON, SCAN_IN_PROGRESS, COUNTER_ERROR, FRAME_TOO_LONG, KEY_ERROR, UNAVAILABLE_KEY, UNSUPPORTED_SECURITY or INVALID_PARAMETER	The status of the scan request.
ScanType	Integer	0x00–0x03	Indicates if the type of scan performed: 0x00 = ED scan (optional for RFD). 0x01 = active scan (optional for RFD). 0x02 = passive scan. 0x03 = orphan scan.
ChannelPage	Integer	0–31	The channel page on which the scan was performed (see 6.1.2).
UnscannedChannels	Bitmap	27 bit field	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.
ResultListSize	Integer	Implementation specific	The number of elements returned in the appropriate result lists. This value is zero for the result of an orphan scan.
EnergyDetectList	List of integers	0x00–0xff for each integer	The list of energy measurements, one for each channel searched during an ED scan. This parameter is null for active, passive, and orphan scans.
PANDescriptorList	List of PAN descriptor values	See Table 55	The list 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.

7.1.11.2.3 Appropriate usage

On receipt of the MLME-SCAN.confirm primitive, the next higher layer is notified of the results of the scan procedure. If the requested scan was successful, the status parameter will be set to SUCCESS. Otherwise, the status parameter indicates the error.

7.1.11.3 Channel scan message sequence charts

Figure 81, Figure 84, Figure 85 and Figure 88 (see 7.7) illustrate the sequence of messages necessary to perform an ED scan, a passive scan, an active scan and an orphan scan, respectively. These figures include steps taken by the PHY.

7.1.12 Communication status primitive

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 security errors on incoming packets.

All devices shall provide an interface for this communication status primitive.

7.1.12.1 MLME-COMM-STATUS.indication

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

7.1.12.1.1 Semantics of the service primitive

The semantics of the MLME-COMM-STATUS.indication primitive are as follows:

```
MLME-COMM-STATUS.indication     (
                                PANId,
                                SrcAddrMode,
                                SrcAddr,
                                DstAddrMode,
                                DstAddr,
                                status,
                                SecurityLevel,
                                KeyIdMode,
                                KeySource,
                                KeyIndex
                                )
```

Table 69 specifies the parameters for the MLME-COMM-STATUS.indication primitive.

Table 69—MLME-COMM-STATUS.indication parameters

Name	Type	Valid range	Description
PANId	Integer	0x0000–0xffff	The 16-bit PAN identifier of the device from which the frame was received or to which the frame was being sent.
SrcAddrMode	Integer	0x00–0x03	The source addressing mode for this primitive. This value can take one of the following values: 0 = no address (addressing fields omitted). 0x01 = reserved. 0x02 = 16-bit short address. 0x03 = 64-bit extended address.
SrcAddr	Device address	As specified by the SrcAddrMode parameter	The individual device address of the entity from which the frame causing the error originated.

Table 69—MLME-COMM-STATUS.indication parameters

Name	Type	Valid range	Description
DstAddrMode	Integer	0x00–0x03	The destination addressing mode for this primitive. This value can take one of the following values: 0x00 = no address (addressing fields omitted). 0x01 = reserved. 0x02 = 16-bit short address. 0x03 = 64-bit extended address.
DstAddr	Device address	As specified by the DstAddrMode parameter	The individual device address of the device for which the frame was intended.
Status	Enumeration	SUCCESS, TRANSACTION_OVERFLOW, TRANSACTION_EXPIRED, CHANNEL_ACCESS_FAILURE, NO_ACK, COUNTER_ERROR, FRAME_TOO_LONG, IMPROPER_KEY_TYPE, IMPROPER_SECURITY_LEVEL, KEY_ERROR, SECURITY_ERROR, UNAVAILABLE_DEVICE, UNAVAILABLE_KEY, UNAVAILABLE_SECURITY_LEVEL, UNSUPPORTED_LEGACY, UNSUPPORTED_SECURITY or INVALID_PARAMETER	The communications status.
SecurityLevel	Integer	0x00–0x07	If the primitive was generated following a transmission instigated through a .response primitive: The security level to be used (see Table 96). If the primitive was generated on receipt of a frame that generates an error in its security processing: The security level purportedly used by the received frame (see Table 96).

Table 69—MLME-COMM-STATUS.indication parameters

Name	Type	Valid range	Description
KeyIdMode	Integer	0x00–0x03	If the primitive was generated following a transmission instigated through a .response primitive: The mode used to identify the key to be used (see Table 97). This parameter is ignored if the SecurityLevel parameter is set to 0x00. If the primitive was generated on receipt of a frame that generates an error in its security processing: The mode used to identify the key purportedly used by the originator of the received frame (see Table 97). This parameter is invalid if the SecurityLevel parameter is set to 0x00.
KeySource	Set of 0, 4 or 8 octets	As specified by the KeyIdMode parameter	If the primitive was generated following a transmission instigated through a .response primitive: The originator of the key to be used (see 7.6.2.4.1). This parameter is ignored if the KeyIdMode parameter is ignored or set to 0x00. If the primitive was generated on receipt of a frame that generates an error in its security processing: The originator of the key purportedly used by the originator of the received frame (see 7.6.2.4.1). This parameter is invalid if the KeyIdMode parameter is invalid or set to 0x00.
KeyIndex	Integer	0x01–0xff	If the primitive was generated following a transmission instigated through a .response primitive: The index of the key to be used (see 7.6.2.4.2). This parameter is ignored if the KeyIdMode parameter is ignored or set to 0x00. If the primitive was generated on receipt of a frame that generates an error in its security processing: The index of the key purportedly used by the originator of the received frame (see 7.6.2.4.2). This parameter is invalid if the KeyIdMode parameter is invalid or set to 0x00.

7.1.12.1.2 When generated

The MLME-COMM-STATUS.indication primitive is generated by the MLME and issued to its next higher layer either following a transmission instigated through a .response primitive or on receipt of a frame that generates an error in its security processing (see 7.5.8.2.3).

The MLME-COMM-STATUS.indication primitive is generated by the MAC sublayer entity following either the MLME-ASSOCIATE.response primitive or the MLME-ORPHAN.response primitive. This primitive returns a status of either SUCCESS, indicating that the request to transmit was successful, an error code of TRANSACTION_OVERFLOW, TRANSACTION_EXPIRED, CHANNEL_ACCESS_FAILURE, NO_ACK or INVALID_PARAMETER (these status values are fully described in 7.1.3.3.3 and 7.1.8.2.3), or an error code resulting from failed security processing (these status values are fully described in 7.5.8.2.1 and 7.5.8.2.3).

7.1.12.1.3 Appropriate usage

On receipt of the MLME-COMM-STATUS.indication primitive, the next higher layer is notified of the communication status of a transmission or notified of an error that has occurred during the secure processing of incoming frame.

7.1.13 Primitives for writing PIB attributes

MLME-SAP set primitives define how PIB attributes may be written.

All devices shall provide an interface for these set primitives.

7.1.13.1 MLME-SET.request

The MLME-SET.request primitive attempts to write the given value to the indicated PIB attribute.

7.1.13.1.1 Semantics of the primitive

The semantics of the MLME-SET.request primitive are as follows:

```
MLME-SET.request        (
                        PIBAttribute,
                        PIBAttributeIndex,
                        PIBAttributeValue
                        )
```

Table 70 specifies the parameters for the MLME-SET.request primitive.

7.1.13.1.2 Appropriate usage

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

7.1.13.1.3 Effect on receipt

On receipt of the MLME-SET.request primitive, the MLME checks to see if the PIB attribute is a MAC PIB attribute or PHY PIB attribute. If the requested attribute is a MAC attribute, the MLME attempts to write the given value to the indicated MAC PIB attribute in its database. If the PIBAttribute parameter specifies an attribute that is a read-only attribute (see Table 86 and Table 88), the MLME will issue the MLME-SET.confirm primitive with a status of READ_ONLY. If the PIBAttribute parameter specifies an attribute that is not

Table 70—MLME-SET.request parameters

Name	Type	Valid range	Description
PIBAttribute	Integer	See Table 86 and Table 88	The identifier of the PIB attribute to write.
PIBAttributeIndex	Integer	Attribute specific; see Table 88	The index within the table of the specified PIB attribute to write. This parameter is only valid for MAC PIB attributes which are tables; it is ignored when accessing PHY PIB attributes.
PIBAttributeValue	Various	Attribute specific; see Table 86 and Table 88	The value to write to the indicated PIB attribute.

found in the database, the MLME will issue the MLME-SET.confirm primitive with a status of UNSUPPORTED_ATTRIBUTE. If the PIBAttributeIndex parameter specifies an index for a table that is out of range, the MLME will issue the MLME-SET.confirm primitive with a status of INVALID_INDEX. If the PIBAttributeValue parameter specifies a value that is out of the valid range for the given attribute, the MLME will issue the MLME-SET.confirm primitive with a status of INVALID_PARAMETER. If the requested MAC PIB attribute is successfully written, the MLME will issue the MLME-SET.confirm primitive with a status of SUCCESS.

If the PIBAttribute parameter indicates that *macBeaconPayloadLength* is to be set, and the length of the resulting beacon frame exceeds *aMaxPHYPacketSize* (e.g., due to the additional overhead required for security processing), the MAC sublayer shall not update *macBeaconPayloadLength* and will issue the MLME-GET.confirm primitive with a status of INVALID_PARAMETER.

If the requested attribute is a PHY PIB attribute, the request is passed to the PHY by issuing the PLME-SET.request primitive. Once the MLME receives the PLME-SET.confirm primitive, it will translate the received status value, since the status values used by the PHY are not the same as those used by the MLME (e.g., the status values for SUCCESS are 0x00 and 0x07 in the MAC and PHY enumeration tables, respectively). Following the translation, the MLME will issue the MLME-SET.confirm primitive to the next higher layer with the status parameter resulting from the translation and the PIBAttribute parameter equal to that returned by the PLME primitive.

7.1.13.2 MLME-SET.confirm

The MLME-SET.confirm primitive reports the results of an attempt to write a value to a PIB attribute.

7.1.13.2.1 Semantics of the service primitive

The semantics of the MLME-SET.confirm primitive are as follows:

```
MLME-SET.confirm        (
                        status,
                        PIBAttribute,
                        PIBAttributeIndex
                        )
```

Table 71 specifies the parameters for the MLME-SET.confirm primitive.

Table 71—MLME-SET.confirm parameters

Name	Type	Valid range	Description
status	Enumeration	SUCCESS, READ_ONLY, UNSUPPORTED_ATTRIBUTE, INVALID_INDEX or INVALID_PARAMETER	The result of the request to write the PIB attribute.
PIBAttribute	Integer	See Table 86 and Table 88	The identifier of the PIB attribute that was written.
PIBAttributeIndex	Integer	Attribute specific; see Table 88	The index within the table of the specified PIB attribute to write. This parameter is only valid for MAC PIB attributes which are tables; it is ignored when accessing PHY PIB attributes.

7.1.13.2.2 When generated

The MLME-SET.confirm primitive is generated by the MLME and issued to its next higher layer in response to an MLME-SET.request primitive. The MLME-SET.confirm primitive returns a status of either SUCCESS, indicating that the requested value was written to the indicated PIB attribute, or the appropriate error code. The status values are fully described in 7.1.13.1.3.

7.1.13.2.3 Appropriate usage

On receipt of the MLME-SET.confirm primitive, the next higher layer is notified of the result of its request to set the value of a PIB attribute. If the requested value was written to the indicated PIB attribute, the status parameter will be set to SUCCESS. Otherwise, the status parameter indicates the error.

7.1.14 Primitives for updating the superframe configuration

MLME-SAP start primitives define how an FFD can request to start using a new superframe configuration in order to initiate a PAN, begin transmitting beacons on an already existing PAN, thus facilitating device discovery, or to stop transmitting beacons.

These start primitives are optional for an RFD.

7.1.14.1 MLME-START.request

The MLME-START.request primitive allows 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.

7.1.14.1.1 Semantics of the service primitive

The semantics of the MLME-START.request primitive are as follows:

```
MLME-START.request          (
                            PANId,
                            LogicalChannel,
                            ChannelPage,
                            StartTime,
                            BeaconOrder,
                            SuperframeOrder,
                            PANCoordinator,
                            BatteryLifeExtension,
                            CoordRealignment,
                            CoordRealignSecurityLevel,
                            CoordRealignKeyIdMode,
                            CoordRealignKeySource,
                            CoordRealignKeyIndex,
                            BeaconSecurityLevel,
                            BeaconKeyIdMode,
                            BeaconKeySource,
                            BeaconKeyIndex
                            )
```

Table 72 specifies the parameters for the MLME-START.request primitive.

Table 72—MLME-START.request parameters

Name	Type	Valid range	Description
PANId	Integer	0x0000–0xffff	The PAN identifier to be used by the device.
LogicalChannel	Integer	Selected from the available logical channels specified by the ChannelPage parameter	The logical channel on which to start using the new superframe configuration.
ChannelPage	Integer	Selected from the available channel pages supported by the PHY (see 6.1.2)	The channel page on which to begin using the new superframe configuration.

Table 72—MLME-START.request parameters *(continued)*

Name	Type	Valid range	Description
StartTime	Integer	0x000000–0xffffff	The time at which to begin transmitting beacons. If this parameter is equal to 0x000000, beacon transmissions will begin immediately. Otherwise, the specified time is relative to the received beacon of the coordinator with which the device synchronizes. This parameter is ignored if either the BeaconOrder parameter has a value of 15 or the PANCoordinator parameter is TRUE. The time is specified in symbols and is rounded to a backoff slot boundary. This is a 24-bit value, and the precision of this value shall be a minimum of 20 bits, with the lowest 4 bits being the least significant.
BeaconOrder	Integer	0–15	How often the beacon is to be transmitted. A value of 15 indicates that the coordinator will not transmit periodic beacons. See 7.5.1.1 for an explanation of the relationship between the beacon order and the beacon interval.
SuperframeOrder	Integer	0–*BO* or 15	The length of the active portion of the superframe, including the beacon frame. If the BeaconOrder parameter has a value of 15, this parameter is ignored. See 7.5.1.1 for an explanation of the relationship between the superframe order and the superframe duration.
PANCoordinator	Boolean	TRUE or FALSE	If this value is TRUE, the device will become the PAN coordinator of a new PAN. If this value is FALSE, the device will begin using a new superframe configuration on the PAN with which it is associated.
BatteryLifeExtension	Boolean	TRUE or FALSE	If this value is TRUE, the receiver of the beaconing device is disabled *macBattLifeExtPeriods* full backoff periods after the interframe spacing (IFS) period following the beacon frame. If this value is FALSE, the receiver of the beaconing device remains enabled for the entire CAP. This parameter is ignored if the BeaconOrder parameter has a value of 15.

Table 72—MLME-START.request parameters *(continued)*

Name	Type	Valid range	Description
CoordRealignment	Boolean	TRUE or FALSE	TRUE if a coordinator realignment command is to be transmitted prior to changing the superframe configuration or FALSE otherwise.
CoordRealignSecurityLevel	Integer	0x00–0x07	The security level to be used for coordinator realignment command frames (see Table 96).
CoordRealignKeyIdMode	Integer	0x00–0x03	The mode used to identify the key to be used (see Table 97). This parameter is ignored if the CoordRealignSecurityLevel parameter is set to 0x00.
CoordRealignKeySource	Set of 0, 4 or 8 octets	As specified by the CoordRealignKeyIdMode parameter	The originator of the key to be used (see 7.6.2.4.1). This parameter is ignored if the CoordRealignKeyIdMode parameter is ignored or set to 0x00.
CoordRealignKeyIndex	Integer	0x01–0xff	The index of the key to be used (see 7.6.2.4.2). This parameter is ignored if the CoordRealignKeyIdMode parameter is ignored or set to 0x00.
BeaconSecurityLevel	Integer	0x00–0x07	The security level to be used for beacon frames (see Table 96).
BeaconKeyIdMode	Integer	0x00–0x03	The mode used to identify the key to be used (see Table 97). This parameter is ignored if the BeaconSecurityLevel parameter is set to 0x00.
BeaconKeySource	Set of 0, 4 or 8 octets	As specified by the BeaconKeyIdMode parameter	The originator of the key to be used (see 7.6.2.4.1). This parameter is ignored if the BeaconKeyIdMode parameter is ignored or set to 0x00.
BeaconKeyIndex	Integer	0x01–0xff	The index of the key to be used (see 7.6.2.4.2). This parameter is ignored if the BeaconKeyIdMode parameter is ignored or set to 0x00.

7.1.14.1.2 Appropriate usage

The MLME-START.request primitive is generated by the next higher layer and issued to its MLME to request that a device start using a new superframe configuration.

7.1.14.1.3 Effect on receipt

If the MLME-START.request primitive is received when *macShortAddress* is set to 0xffff, the MLME will issue the MLME-START.confirm primitive with a status of NO_SHORT_ADDRESS.

When the CoordRealignment parameter is set to TRUE, the coordinator attempts to transmit a coordinator realignment command frame as described in 7.5.2.3.2. If the transmission of the coordinator realignment command fails due to a channel access failure, the MLME will not make any changes to the superframe configuration (i.e., no PIB attributes will be changed) and will issue an MLME-START.confirm with a status of

CHANNEL_ACCESS_FAILURE. If the coordinator realignment command is successfully transmitted, the MLME updates the appropriate PIB parameters with the values of the BeaconOrder, SuperframeOrder, PANId, ChannelPage and LogicalChannel parameters, as described in 7.5.2.3.4, and will issue an MLME-START.confirm with a status of SUCCESS.

When the CoordRealignment parameter is set to FALSE, the MLME updates the appropriate PIB parameters with the values of the BeaconOrder, SuperframeOrder, PANId, ChannelPage and LogicalChannel parameters, as described in 7.5.2.3.4.

The address used by the coordinator in its beacon frames is determined by the current value of *macShortAddress*, which is set by the next higher layer before issuing this primitive. If the BeaconOrder parameter is less than 15, the MLME sets *macBattLifeExt* to the value of the BatteryLifeExtension parameter. If the BeaconOrder parameter equals 15, the value of the BatteryLifeExtension parameter is ignored.

If the SecurityLevel parameter is set to a valid value other than 0x00, indicating that security is required for this frame, the MLME will set the security enabled subfield of the frame control field to one. The MAC sublayer will perform outgoing processing on the frame, as described in 7.5.8.2.1. If the CoordRealignment parameter is set to TRUE, the CoordRealignSecurityLevel, CoordRealignKeyIdMode, CoordRealignKeySource and CoordRealignKeyIndex parameters will be used to process the MAC command frame. If the BeaconOrder parameter indicates a beacon-enabled network, the BeaconSecurityLevel, BeaconKeyIdMode, BeaconKeySource and BeaconKeyIndex parameters will be used to process the beacon frame. If any error occurs during outgoing frame processing, the MLME will discard the frame and issue the MLME-START.confirm primitive with the error status returned by outgoing frame processing.

If the length of the beacon frame exceeds *aMaxPHYPacketSize* (e.g., due to the additional overhead required for security processing), the MAC sublayer shall discard the beacon frame and issue the MLME-START.confirm primitive with a status of FRAME_TOO_LONG.

The MLME shall ignore the StartTime parameter if the BeaconOrder parameter is equal to 15, since this indicates a nonbeacon-enabled PAN. If the BeaconOrder parameter is less than 15, the MLME examines the StartTime parameter to determine the time to begin transmitting beacons; the time is defined in symbols and is rounded to a backoff slot boundary. If the PAN coordinator parameter is set to TRUE, the MLME ignores the StartTime parameter and begins beacon transmissions immediately. Setting the StartTime parameter to 0x000000 also causes the MLME to begin beacon transmissions immediately. If the PANCoordinator parameter is set to FALSE and the StartTime parameter is nonzero, the MLME calculates the beacon transmission time by adding StartTime symbols to the time, obtained from the local clock, when the MLME receives the beacon of the coordinator through which it is associated. If the time calculated causes the outgoing superframe to overlap the incoming superframe, the MLME shall not begin beacon transmissions. In this case, the MLME issues the MLME-START.confirm primitive with a status of SUPERFRAME_OVERLAP. Otherwise, the MLME then begins beacon transmissions when the current time, obtained from the local clock, equals the number of calculated symbols.

If the StartTime parameter is nonzero and the MLME is not currently tracking the beacon of the coordinator through with it is associated, the MLME will issue the MLME-START.confirm primitive with a status of TRACKING_OFF.

On completion of this procedure, the MLME responds with the MLME-START.confirm primitive. If the attempt to start using a new superframe configuration was successful, the status parameter will be set to SUCCESS. If any parameter is not supported or is out of range, the status parameter will be set to INVALID_PARAMETER.

7.1.14.2 MLME-START.confirm

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

7.1.14.2.1 Semantics of the service primitive

The semantics of the MLME-START.confirm primitive are as follows:

```
MLME-START.confirm      (
                        status
                        )
```

Table 73 specifies the parameters for the MLME-START.confirm primitive.

Table 73—MLME-START.confirm parameters

Name	Type	Valid range	Description
status	Enumeration	SUCCESS, NO_SHORT_ADDRESS, SUPERFRAME_OVERLAP, TRACKING_OFF, INVALID_PARAMETER, COUNTER_ERROR, FRAME_TOO_LONG, KEY_ERROR, UNAVAILABLE_KEY, UNSUPPORTED_SECURITY or CHANNEL_ACCESS_FAILURE	The result of the attempt to start using an updated superframe configuration.

7.1.14.2.2 When generated

The MLME-START.confirm primitive is generated by the MLME and issued to its next higher layer in response to an 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 the appropriate error code. The status values are fully described in 7.1.14.1.3 and subclauses referenced by 7.1.14.1.3.

7.1.14.2.3 Appropriate usage

On receipt of the MLME-START.confirm primitive, the next higher layer is notified of the result of its request to start using a new superframe configuration. If the MAC sublayer has been successful, the status parameter will be set to SUCCESS. Otherwise, the status parameter indicates the error.

7.1.14.3 Message sequence chart for updating the superframe configuration

Figure 38 illustrates the sequence of messages necessary for initiating beacon transmissions in an FFD. Figure 80 (see 7.7) illustrates the sequence of messages necessary for the PAN coordinator to start beaconing on a new PAN; this figure includes steps taken by the PHY.

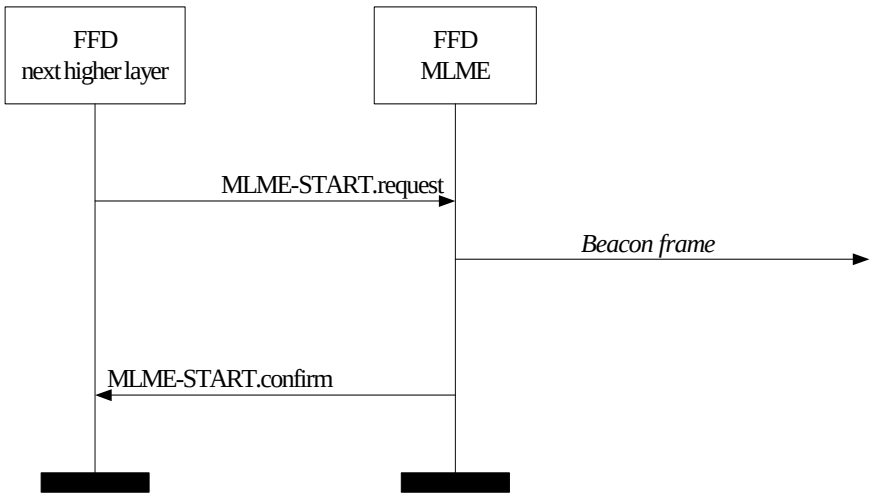


Figure 38—Message sequence chart for updating the superframe configuration

7.1.15 Primitives for synchronizing with a coordinator

MLME-SAP synchronization primitives define how synchronization with a coordinator may be achieved and how a loss of synchronization is communicated to the next higher layer.

All devices shall provide an interface for the indication primitive. The request primitive is optional.

7.1.15.1 MLME-SYNC.request

The MLME-SYNC.request primitive requests to synchronize with the coordinator by acquiring and, if specified, tracking its beacons.

7.1.15.1.1 Semantics of the service primitive

The semantics of the MLME-SYNC.request primitive are as follows:

```
MLME-SYNC.request       (
                        LogicalChannel,
                        ChannelPage,
                        TrackBeacon
                        )
```

Table 74 specifies the parameters for the MLME-SYNC.request primitive.

7.1.15.1.2 Appropriate usage

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.

7.1.15.1.3 Effect on receipt

If the MLME-SYNC.request primitive is received by the MLME on a beacon-enabled PAN, it will first set *phyCurrentPage* and *phyCurrentChannel* equal to the values of the ChannelPage and LogicalChannel

Table 74—MLME-SYNC.request parameters

Name	Type	Valid range	Description
LogicalChannel	Integer	Selected from the available logical channels supported by the PHY	The logical channel on which to attempt coordinator synchronization.
ChannelPage	Integer	Selected from the available logical channels supported by the PHY (see 6.1.2)	The channel page on which to attempt coordinator synchronization.
TrackBeacon	Boolean	TRUE or FALSE	TRUE if the MLME is to synchronize with the next beacon and attempt to track all future beacons. FALSE if the MLME is to synchronize with only the next beacon.

parameters, respectively; both attributes are updated by issuing the PLME-SET.request primitive. If the TrackBeacon parameter is equal to TRUE, the MLME will track the beacon, i.e., enable its receiver just before the expected time of each beacon so that the beacon frame can be processed. If the TrackBeacon parameter is equal to FALSE, the MLME will locate the beacon, but not continue to track it.

If this primitive is received by the MLME while it is currently tracking the beacon, the MLME will not discard the primitive, but rather treat it as a new synchronization request.

If the beacon could not be located either on its initial search or during tracking, the MLME will issue the MLME-SYNC-LOSS.indication primitive with a loss reason of BEACON_LOST.

7.1.15.2 MLME-SYNC-LOSS.indication

The MLME-SYNC-LOSS.indication primitive indicates the loss of synchronization with a coordinator.

7.1.15.2.1 Semantics of the service primitive

The semantics of the MLME-SYNC-LOSS.indication primitive are as follows:

```
MLME-SYNC-LOSS.indication       (
                                LossReason,
                                PANId,
                                LogicalChannel,
                                ChannelPage,
                                SecurityLevel,
                                KeyIdMode,
                                KeySource,
                                KeyIndex
                                )
```

Table 75 specifies the parameters for the MLME-SYNC-LOSS.indication primitive.

7.1.15.2.2 When generated

The MLME-SYNC-LOSS.indication primitive is generated by the MLME of a device and issued to its next higher layer in the event of a loss of 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.

Table 75—MLME-SYNC-LOSS.indication parameters

Name	Type	Valid range	Description
LossReason	Enumeration	PAN_ID_CONFLICT, REALIGNMENT, or BEACON_LOST	The reason that synchronization was lost.
PANId	Integer	0x0000–0xffff	The PAN identifier with which the device lost synchronization or to which it was realigned.
LogicalChannel	Integer	Selected from the available logical channels supported by the PHY (see 6.1.2).	The logical channel on which the device lost synchronization or to which it was realigned.
ChannelPage	Integer	Selected from the available channel pages supported by the PHY (see 6.1.2).	The channel page on which the device lost synchronization or to which it was realigned.

Table 75—MLME-SYNC-LOSS.indication parameters

Name	Type	Valid range	Description
SecurityLevel	Integer	0x00-0x07	If the primitive was either generated by the device itself following loss of synchronization or generated by the PAN coordinator upon detection of a PAN ID conflict, the security level is set to 0x00. If the primitive was generated following the reception of either a coordinator realignment command or a PAN ID conflict notification command: The security level purportedly used by the received MAC frame (see Table 96).
KeyIdMode	Integer	0x00–0x03	If the primitive was either generated by the device itself following loss of synchronization or generated by the PAN coordinator upon detection of a PAN ID conflict, this parameter is ignored. If the primitive was generated following the reception of either a coordinator realignment command or a PAN ID conflict notification command: The mode used to identify the key purportedly used by the originator of the received frame (see Table 97). This parameter is invalid if the SecurityLevel parameter is set to 0x00.
KeySource	Set of 0, 4 or 8 octets	As specified by the KeyIdMode parameter	If the primitive was either generated by the device itself following loss of synchronization or generated by the PAN coordinator upon detection of a PAN ID conflict, this parameter is ignored. If the primitive was generated following the reception of either a coordinator realignment command or a PAN ID conflict notification command: The originator of the key purportedly used by the originator of the received frame (see 7.6.2.4.1). This parameter is invalid if the KeyIdMode parameter is invalid or set to 0x00.

Table 75—MLME-SYNC-LOSS.indication parameters

Name	Type	Valid range	Description
KeyIndex	Integer	0x01–0xff	If the primitive was either generated by the device itself following loss of synchronization or generated by the PAN coordinator upon detection of a PAN ID conflict, this parameter is ignored. If the primitive was generated following the reception of either a coordinator realignment command or a PAN ID conflict notification command: The index of the key purportedly used by the originator of the received frame (see 7.6.2.4.2). This parameter is invalid if the KeyIdMode parameter is invalid or set to 0x00.

If a device that is associated through the PAN coordinator has detected a PAN identifier conflict and communicated it to the PAN coordinator, the MLME will issue this primitive with the LossReason parameter set to PAN_ID_CONFLICT. Similarly, if the PAN coordinator receives a PAN ID conflict notification command (see 7.3.5), the MLME will issue this primitive with the LossReason parameter set to PAN_ID_CONFLICT.

If a device has received the coordinator realignment command (see 7.3.8) from the coordinator through which it associated and the MLME was not carrying out an orphan scan, the MLME will issue this primitive with the LossReason parameter set to REALIGNMENT and the PANId, LogicalChannel, ChannelPage and security-related parameters set as described in 7.5.2.3.3.

If a device has not heard the beacon for *aMaxLostBeacons* consecutive superframes following an MLME-SYNC.request primitive, either initially or during tracking, the MLME will issue this primitive with the LossReason parameter set to BEACON_LOST. The PANId, LogicalChannel and ChannelPage parameters shall be set according to the coordinator with which synchronization was lost. The SecurityLevel parameter shall be set to zero and the KeyIdMode, KeySource, and KeyIndex parameters shall be ignored. If the beacon was being tracked, the MLME will not attempt to track the beacon any further.

7.1.15.2.3 Appropriate usage

On receipt of the MLME-SYNC-LOSS.indication primitive, the next higher layer is notified of a loss of synchronization.

7.1.15.3 Message sequence chart for synchronizing with a coordinator

Figure 39 illustrates the sequence of messages necessary for a device to synchronize with a coordinator. In a), a single synchronization request is issued. The MLME then searches for a beacon and, if found, determines whether the coordinator has any data pending for the device. If so, the data are requested as described in 7.5.6.3. In b), a track synchronization request is issued. The MLME then searches for a beacon and, if found, attempts to keep track of it using a timer that expires just before the expected time of the next beacon.

For both examples a) and b), the received beacon frames do not contain payload and *macAutoRequest* is set to TRUE. The MLME also checks for any data pending in the coordinator for the device when a beacon frame is received.

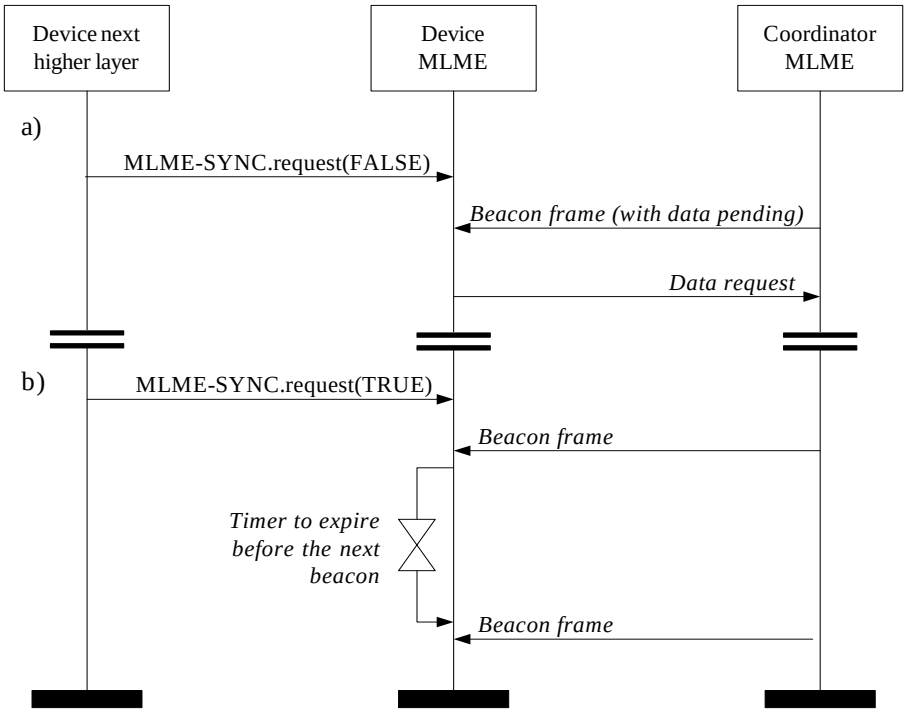


Figure 39—Message sequence chart for synchronizing to a coordinator in a beacon-enabled PAN

7.1.16 Primitives for requesting data from a coordinator

MLME-SAP polling primitives define how to request data from a coordinator.

All devices shall provide an interface for these polling primitives.

7.1.16.1 MLME-POLL.request

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

7.1.16.1.1 Semantics of the service primitive

The semantics of the MLME-POLL.request primitive are as follows:

```
MLME-POLL.request       (
                        CoordAddrMode,
                        CoordPANId,
                        CoordAddress,
                        SecurityLevel,
                        KeyIdMode,
                        KeySource,
                        KeyIndex
                        )
```

Table 76 specifies the parameter for the MLME-POLL.request primitive.

Table 76—MLME-POLL.request parameters

Name	Type	Valid range	Description
CoordAddrMode	Integer	0x02–0x03	The addressing mode of the coordinator to which the poll is intended. This parameter can take one of the following values: 2 = 16-bit short address, 3 = 64-bit extended address.
CoordPANId	Integer	0x0000–0xfffe	The PAN identifier of the coordinator to which the poll is intended.
CoordAddress	Device-Address	As specified by the CoordAddrMode parameter	The address of the coordinator to which the poll is intended.
SecurityLevel	Integer	0x00–0x07	The security level to be used (see Table 96).
KeyIdMode	Integer	0x00–0x03	The mode used to identify the key to be used (see Table 97). This parameter is ignored if the SecurityLevel parameter is set to 0x00.
KeySource	Set of 0, 4 or 8 octets	As specified by the Key-IdMode parameter	The originator of the key to be used (see 7.6.2.4.1). This parameter is ignored if the KeyIdMode parameter is ignored or set to 0x00.
KeyIndex	Integer	0x01–0xff	The index of the key to be used (see 7.6.2.4.2). This parameter is ignored if the KeyIdMode parameter is ignored or set to 0x00.

7.1.16.1.2 Appropriate usage

The MLME-POLL.request primitive is generated by the next higher layer and issued to its MLME when data are to be requested from a coordinator.

7.1.16.1.3 Effect on receipt

On receipt of the MLME-POLL.request primitive, the MLME generates and sends a data request command (see 7.3.4). If the poll is directed to the PAN coordinator, the data request command may be generated without any destination address information present. Otherwise, the data request command is always generated with the destination address information in the CoordPANId and CoordAddress parameters.

If the SecurityLevel parameter is set to a valid value other than 0x00, indicating that security is required for this frame, the MLME will set the security enabled subfield of the frame control field to one. The MAC sublayer will perform outgoing processing on the frame based on the CoordAddress, SecurityLevel, KeyIdMode, KeySource and KeyIndex parameters, as described in 7.5.8.2.1. If any error occurs during outgoing frame processing, the MLME will discard the frame and issue the MLME-POLL.confirm primitive with the error status returned by outgoing frame processing.

If the data request command cannot be sent due to a CSMA-CA algorithm failure, the MLME will issue the MLME-POLL.confirm primitive with a status of CHANNEL_ACCESS_FAILURE.

If the MLME successfully transmits a data request command, the MLME will expect an acknowledgment in return. If an acknowledgment is not received, the MLME will issue the MLME-POLL.confirm primitive with a status of NO_ACK (see 7.5.6.4).

If an acknowledgment is received, the MLME will request that the PHY enable its receiver if the frame pending subfield of the acknowledgment frame is set to one. If the frame pending subfield of the acknowledgment frame is set to zero, the MLME will issue the MLME-POLL.confirm primitive with a status of NO_DATA.

If a frame is received from the coordinator with a zero length payload or if the frame is a MAC command frame, the MLME will issue the MLME-POLL.confirm primitive with a status of NO_DATA. If a frame is received from the coordinator with nonzero length payload, the MLME will issue the MLME-POLL.confirm primitive with a status of SUCCESS. In this case, the actual data are indicated to the next higher layer using the MCPS-DATA.indication primitive (see 7.1.1.3).

If a frame is not received within *macMaxFrameTotalWaitTime* CAP symbols in a beacon-enabled PAN, or symbols in a nonbeacon-enabled PAN, even though the acknowledgment to the data request command has its frame pending subfield set to one, the MLME will issue the MLME-POLL.confirm primitive with a status of NO_DATA.

If any parameter in the MLME-POLL.request primitive is not supported or is out of range, the MLME will issue the MLME-POLL.confirm primitive with a status of INVALID_PARAMETER.

7.1.16.2 MLME-POLL.confirm

The MLME-POLL.confirm primitive reports the results of a request to poll the coordinator for data.

7.1.16.2.1 Semantics of the service primitive

The semantics of the MLME-POLL.confirm primitive are as follows:

```
MLME-POLL.confirm       (
                        status
                        )
```

Table 77 specifies the parameters for the MLME-POLL.confirm primitive.

Table 77—MLME-POLL.confirm parameters

Name	Type	Valid range	Description
status	Integer	SUCCESS, CHANNEL_ACCESS_FAILURE, NO_ACK, NO_DATA, COUNTER_ERROR, FRAME_TOO_LONG, KEY_ERROR, UNAVAILABLE_KEY, UNSUPPORTED_SECURITY or INVALID_PARAMETER	The status of the data request.

7.1.16.2.2 When generated

The MLME-POLL.confirm primitive is generated by the MLME and issued to its next higher layer in response to an MLME-POLL.request primitive. If the request was successful, the status parameter will be

equal to SUCCESS, indicating a successful poll for data. Otherwise, the status parameter indicates the appropriate error code. The status values are fully described in 7.1.16.1.3 and the subclauses referenced by 7.1.16.1.3.

7.1.16.2.3 Appropriate usage

On receipt of the MLME-POLL.confirm primitive, the next higher layer is notified of the status of the procedure to request data from the coordinator.

7.1.16.3 Message sequence chart for requesting data from a coordinator

Figure 40 illustrates the sequence of messages necessary, including the layer behavior of the device and the over-the-air interface, for a device to request data from a coordinator.

In both scenarios a) and b), a poll request is issued to the MLME, which then sends a data request command to the coordinator. In a) Figure 40, the corresponding acknowledgment has the frame pending (FP) subfield set to zero and the MLME issues the poll request confirmation immediately. In b) Figure 40, the corresponding acknowledgment has the frame pending subfield set to one and the MLME enables the receiver in anticipation of the data frame from the coordinator. On receipt of this data frame, the MLME issues a poll request confirmation followed by a data indication containing the data of the received frame.

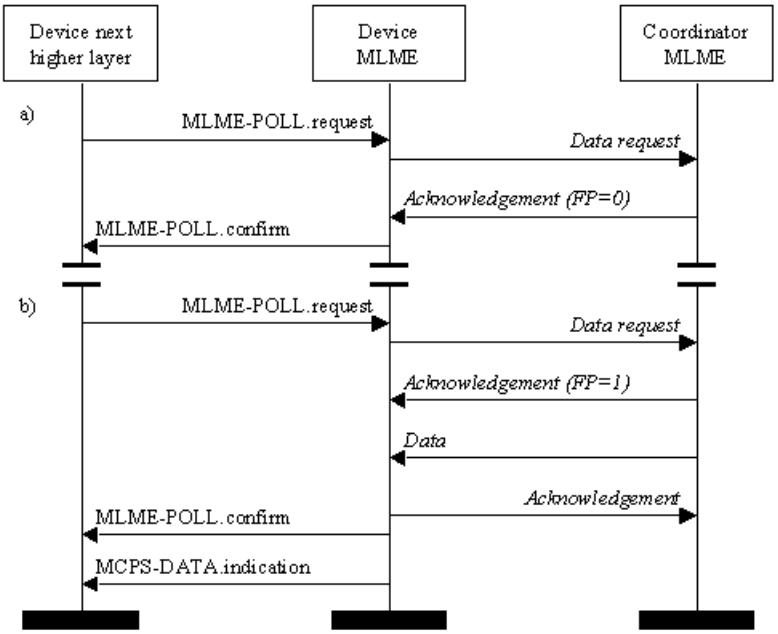


Figure 40—Message sequence chart for requesting data from the coordinator

7.1.17 MAC enumeration description

This subclause explains the meaning of the enumerations used in the primitives defined in the MAC sublayer specification. Table 78 shows a description of the MAC enumeration values.

Table 78—MAC enumerations description

Enumeration	Value	Description
SUCCESS	0x00	The requested operation was completed successfully. For a transmission request, this value indicates a successful transmission.
—	0x01–0xd7	Reserved for MAC command status and reason code values.
—	0x80–0xd7	Reserved.
BEACON_LOSS	0xe0	The beacon was lost following a synchronization request.
CHANNEL_ACCESS_FAILURE	0xe1	A transmission could not take place due to activity on the channel, i.e., the CSMA-CA mechanism has failed.
COUNTER_ERROR	0xdb	The frame counter purportedly applied by the originator of the received frame is invalid.
DENIED	0xe2	The GTS request has been denied by the PAN coordinator.
DISABLE_TRX_FAILURE	0xe3	The attempt to disable the transceiver has failed.
FRAME_TOO_LONG	0xe5	Either a frame resulting from processing has a length that is greater than *aMaxPHYPacketSize* or a requested transaction is too large to fit in the CAP or GTS.
IMPROPER_KEY_TYPE	0xdc	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.
IMPROPER_SECURITY_LEVEL	0xdd	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.
INVALID_ADDRESS	0xf5	A request to send data was unsuccessful, because neither the source address parameters nor the destination address parameters were present.
INVALID_GTS	0xe6	The requested GTS transmission failed because the specified GTS either did not have a transmit GTS direction or was not defined.
INVALID_HANDLE	0xe7	A request to purge an MSDU from the transaction queue was made using an MSDU handle that was not found in the transaction table.
INVALID_INDEX	0xf9	An attempt to write to a MAC PIB attribute that is in a table failed, because the specified table index was out of range.
INVALID_PARAMETER	0xe8	A parameter in the primitive is either not supported or is out of the valid range.
KEY_ERROR	0xd9	The key purportedly used by the originator of the received frame is available, but the originating device is blacklisted with that particular key.
LIMIT_REACHED	0xfa	A scan operation terminated prematurely, because the number of PAN descriptors stored reached an implementation-specified maximum.
NO_ACK	0xe9	No acknowledgment was received after *macMaxFrameRetries*.
NO_BEACON	0xea	A scan operation failed to find any network beacons.

Table 78—MAC enumerations description *(continued)*

Enumeration	Value	Description
NO_DATA	0xeb	No response data were available following a request.
NO_SHORT_ADDRESS	0xec	The operation failed because a 16-bit short address was not allocated.
ON_TIME_TOO_LONG	0xf6	A receiver enable request was unsuccessful, because it specified a number of symbols that was longer than the beacon interval.
OUT_OF_CAP	0xed	A receiver enable request was unsuccessful because it could not be completed within the CAP. The enumeration description is not used in this standard revision, and it is only included to meet the backwards compatibility requirements for IEEE Std 802.15.4-2003.
PAN_ID_CONFLICT	0xee	A PAN identifier conflict has been detected and communicated to the PAN coordinator.
PAST_TIME	0xf7	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.
READ_ONLY	0xfb	A SET/GET request was issued with the identifier of an attribute that is read only.
REALIGNMENT	0xef	A coordinator realignment command has been received.
SCAN_IN_PROGRESS	0xfc	A request to perform a scan operation failed, because the MLME was in the process of performing a previously initiated scan operation.
SECURITY_ERROR	0xe4	Cryptographic processing of the received secured frame failed.
SUPERFRAME_OVERLAP	0xfd	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.
TRACKING_OFF	0xf8	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.
TRANSACTION_EXPIRED	0xf0	The transaction has expired and its information was discarded.
TRANSACTION_OVERFLOW	0xf1	There is no capacity to store the transaction.
TX_ACTIVE	0xf2	The transceiver was in the transmitter enabled state when the receiver was requested to be enabled. The enumeration description is not used in this standard revision, and it is only included to meet the backwards compatibility requirements for IEEE Std 802.15.4-2003.
UNAVAILABLE_KEY	0xf3	The key purportedly used by the originator of the received frame is not available.
UNAVAILABLE_DEVICE	0xda	The device information of the purported originator of the received frame is not available.
UNAVAILABLE_SECURITY_LEVEL	0xd8	The security level expected to have been applied to the received frame is not available.

Table 78—MAC enumerations description *(continued)*

Enumeration	Value	Description
UNSUPPORTED_ATTRIBUTE	0xf4	A SET/GET request was issued with the identifier of a PIB attribute that is not supported.
UNSUPPORTED_LEGACY	0xde	The received frame was purportedly secured using IEEE Std 802.15.4-2003 security, which is not supported.
UNSUPPORTED_SECURITY	0xdf	The security purportedly applied by the originator of the received frame is not supported.
—	0xfe–0xff	Reserved.

7.2 MAC frame formats

This subclause specifies the format of the MAC frame (MPDU). Each MAC frame consists of the following basic components:

a) A MHR, which comprises frame control, sequence number, address information and security-related information.
b) A MAC payload, of variable length, which contains information specific to the frame type. Acknowledgment frames do not contain a payload.
c) A MFR, which contains a FCS.

The frames in the MAC sublayer are described as a sequence of fields in a specific order. All frame formats in this subclause are depicted in the order in which they are transmitted by the PHY, from left to right, where the leftmost bit is transmitted first in time. Bits within each field are numbered from 0 (leftmost and least significant) to $k - 1$ (rightmost and most significant), where the length of the field is k bits. Fields that are longer than a single octet are sent to the PHY in the order from the octet containing the lowest numbered bits to the octet containing the highest numbered bits.

For every MAC frame, all reserved bits shall be set to zero upon transmission and shall be ignored upon receipt.

7.2.1 General MAC frame format

The MAC frame format is composed of a MHR, a MAC payload, and a MFR. The fields of the MHR appear in a fixed order, however, the addressing fields may not be included in all frames. The general MAC frame shall be formatted as illustrated in Figure 41.

<table>
<tr><th>Octets: 0/2</th><th>1</th><th>0/2</th><th>0/2/8</th><th>0/2</th><th>0/2/8</th><th>0/5/6/10/ 14</th><th>variable</th><th>0/2</th></tr>
<tr><td rowspan="2">Frame control</td><td rowspan="2">Sequence number</td><td>Destination PAN identifier</td><td>Destination address</td><td>Source PAN identifier</td><td>Source address</td><td rowspan="2">Auxiliary security header</td><td rowspan="2">Frame payload</td><td rowspan="2">FCS</td></tr>
<tr><td colspan="4">Addressing fields</td></tr>
<tr><td colspan="7">MHR</td><td>MAC payload</td><td>MFR</td></tr>
</table>

Figure 41—General MAC frame format

7.2.1.1 Frame control field

The frame control field is either 1 or 2 octets in length and contains information defining the frame type, addressing fields, and other control flags. For ease of reference, the 1-octet frame control field shall be referred to as the short frame control field in this clause.

The frame control field shall be formatted as illustrated in Figure 42.

Bits: 0–1	2	3	4	5	6	7	8–9	10–11	12	13	14–15
Frame type	Set to 0	Security enabled	Frame pending	Ack. request	PAN ID compression	Frame version	Reserved	Dest. addressing mode	Reserved	Set to 1	Source addressing mode

Figure 42—Format of the frame control field

The short frame control field shall be formatted as illustrated in Figure 43.

Bits: 0–1	2	3	4	5	6	7
Frame type	Set to 1	Security enabled	Frame pending	Ack. request	Reserved	Frame version

Figure 43—Format of the short frame control field

7.2.1.1.1 Frame type subfield

The frame type subfield is 3 bits in length and shall be set to one of the nonreserved values listed in Table 79.

Table 79—Values of the frame type subfield

Frame type value $b_1\ b_0$	Description
00	Beacon
01	Data
10	Acknowledgment
11	MAC command

7.2.1.1.2 Security enabled subfield

The security enabled subfield is 1 bit in length and shall be set to one if the frame is protected by the MAC sublayer and shall be set to zero otherwise. The auxiliary security header field of the MHR shall be present only if the security enabled subfield is set to one.

7.2.1.1.3 Frame pending subfield

The frame pending subfield is 1 bit in length and shall be set to one if the device sending the frame has more data for the recipient. This subfield shall be set to zero otherwise (see 7.5.6.3).

The frame pending subfield shall be used only in beacon frames or frames transmitted either during the CAP by devices operating on a beacon-enabled PAN or at any time by devices operating on a nonbeacon-enabled PAN.

At all other times, it shall be set to zero on transmission and ignored on reception.

7.2.1.1.4 Acknowledgment request subfield

The acknowledgment request subfield is 1 bit in length and specifies whether an acknowledgment is required from the recipient device on receipt of a data or MAC command frame. If this subfield is set to one, the recipient device shall send an acknowledgment frame only if, upon reception, the frame passes the third level of filtering (see 7.5.6.2). If this subfield is set to zero, the recipient device shall not send an acknowledgment frame.

7.2.1.1.5 PAN ID compression subfield

The PAN ID compression subfield is 1 bit in length and specifies whether the MAC frame is to be sent containing only one of the PAN identifier fields when both source and destination addresses are present. If this subfield is set to one and both the source and destination addresses are present, the frame shall only contain the destination PAN identifier field, and the source PAN identifier field shall be assumed equal to that of the destination. If this subfield is set to zero and both the source and destination addresses are present, the frame shall contain both the source and destination PAN identifier fields. If only one of the addresses is present, this subfield shall be set to zero and the frame shall contain the PAN identifier field corresponding to the address. If neither address is present, this subfield shall be set to zero and the frame shall not contain either PAN identifier field.

7.2.1.1.6 Destination addressing mode subfield

The destination addressing mode subfield is 2 bits in length and shall be set to one of the nonreserved values listed in Table 80.

If this subfield is equal to zero and the frame type subfield does not specify that this frame is an acknowledgment or beacon frame, the source addressing mode subfield shall be nonzero, implying that the frame is directed to the PAN coordinator with the PAN identifier as specified in the source PAN identifier field.

Table 80—Possible values of the destination and source addressing mode subfields

Addressing mode value $b_1\ b_0$	Description
00	PAN identifier and address field are not present.
01	Reserved.
10	Address field contains a 16-bit short address.
11	Address field contains a 64-bit extended address.

7.2.1.1.7 Frame version subfield

The frame version subfield is 2 bits in length and specifies the version number corresponding to the frame.

This subfield shall be set to 0x00 to indicate an IEEE Std 802.15.4-2003 compatible frame and 0x01 to indicate an IEEE P802.15.4REVb/D6 enhanced frame. All other subfield values shall be reserved for future use. See 7.2.3 for details on frame compatibility.

7.2.1.1.8 Source addressing mode subfield

The source addressing mode subfield is 2 bits in length and shall be set to one of the nonreserved values listed in Table 80.

If this subfield is equal to zero and the frame type subfield does not specify that this frame is an acknowledgment frame, the destination addressing mode subfield shall be nonzero, implying that the frame has originated from the PAN coordinator with the PAN identifier as specified in the destination PAN identifier field.

7.2.1.2 Sequence number field

The sequence number field is 1 octet in length and specifies the sequence identifier for the frame.

For a beacon frame, the sequence number field shall specify a BSN. For a data, acknowledgment, or MAC command frame, the sequence number field shall specify a data sequence number (DSN) that is used to match an acknowledgment frame to the data or MAC command frame.

7.2.1.3 Destination PAN identifier field

The destination PAN identifier field, when present, is 2 octets in length and specifies the unique PAN identifier of the intended recipient of the frame. A value of 0xffff in this field shall represent the broadcast PAN identifier, which shall be accepted as a valid PAN identifier by all devices currently listening to the channel.

This field shall be included in the MAC frame only if the destination addressing mode subfield of the frame control field is nonzero.

7.2.1.4 Destination address field

The destination address field, when present, is either 2 octets or 8 octets in length, according to the value specified in the destination addressing mode subfield of the frame control field (see 7.2.1.1.6), and specifies the address of the intended recipient of the frame. A 16-bit value of 0xffff in this field shall represent the broadcast short address, which shall be accepted as a valid 16-bit short address by all devices currently listening to the channel.

This field shall be included in the MAC frame only if the destination addressing mode subfield of the frame control field is nonzero.

7.2.1.5 Source PAN identifier field

The source PAN identifier field, when present, is 2 octets in length and specifies the unique PAN identifier of the originator of the frame. This field shall be included in the MAC frame only if the source addressing mode and PAN ID compression subfields of the frame control field are nonzero and equal to zero, respectively.

The PAN identifier of a device is initially determined during association on a PAN, but may change following a PAN identifier conflict resolution (see 7.5.2.2).

7.2.1.6 Source address field

The source address field, when present, is either 2 octets or 8 octets in length, according to the value specified in the destination addressing mode subfield of the frame control field (see 7.2.1.1.8), and specifies the address of the originator of the frame. This field shall be included in the MAC frame only if the source addressing mode subfield of the frame control field is nonzero.

7.2.1.7 Auxiliary security header field

The auxiliary security header field has a variable length and specifies information required for security processing, including how the frame is actually protected (security level) and which keying material from the MAC security PIB is used (see 7.6.1). This field shall only be present if the security enabled subfield is set to one. For details on formatting, see 7.6.2.

7.2.1.8 Frame payload field

The frame payload field has a variable length and contains information specific to individual frame types. If the security enabled subfield is set to one in the frame control field, the frame payload is protected as defined by the security suite selected for that frame.

7.2.1.9 FCS field

The FCS field is 2 octets in length and contains a 16-bit ITU-T CRC. The FCS is calculated over the MHR and MAC payload parts of the frame.This field shall be present only if the security enabled subfield is set to zero or if frame protection does not result in data expansion of the frame payload field (see 7.6.2.2.1).

The FCS shall be calculated using the following standard generator polynomial of degree 16:

$$G_{16}(x) = x^{16} + x^{12} + x^5 + 1 \quad (12)$$

The FCS shall be calculated for transmission using the following algorithm:

— Let $M(x) = b_0x^{k-1} + b_1x^{k-2} + \ldots + b_{k-2}x + b_{k-1}$ be the polynomial representing the sequence of bits for which the checksum is to be computed.
— Multiply $M(x)$ by x^{16}, giving the polynomial $x^{16} \times M(x)$.
— Divide $x^{16} \times M(x)$ modulo 2 by the generator polynomial, $G_{16}(x)$, to obtain the remainder polynomial, $R(x) = r_0x^{15} + r_1x^{14} + \ldots + r_{14}x + r_{15}$.
— The FCS field is given by the coefficients of the remainder polynomial, *R(x)*.

Here, binary polynomials are represented as bit strings, in highest polynomial degree first order.

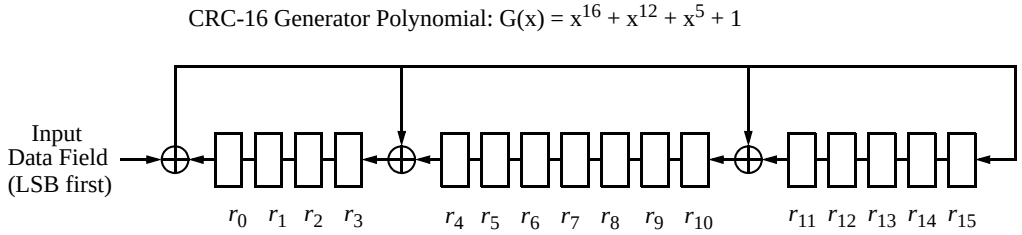
As an example, consider an acknowledgment frame with no payload and the following 3 byte MHR:

0100 0000 0000 0000 0101 0110 [leftmost bit (b_0) transmitted first in time]
b_0.....b_{23}

The FCS for this case would be the following:

0010 0111 1001 1110 [leftmost bit (r_0) transmitted first in time]
r_0.....r_{15}

A typical implementation is depicted in Figure 44.



1. Initialize the remainder register (r_0 through r_{15}) to zero.
2. Shift MHR and payload into the divider in the order of transmission (LSB first).
3. After the last bit of the data field is shifted into the divider, the remainder register contains the FCS.
4. The FCS is appended to the data field so that r_0 is transmitted first.

Figure 44—Typical FCS implementation

7.2.2 Format of individual frame types

Four frame types are defined: beacon, data, acknowledgment, and MAC command. These frame types are discussed in 7.2.2.1 through 7.2.2.4.

7.2.2.1 Beacon frame format

The beacon frame shall be formatted as illustrated in Figure 45.

Octets: 1/2	**1**	**4/10**	**0/5/6/10/ 14**	**2**	**variable**	**variable**	**variable**	**0/2**
Frame control	Sequence number	Addressing fields	Auxiliary security header	Super-frame specifi-cation	GTS fields (Figure 46)	Pending address fields (Figure 47)	Beacon payload	FCS
MHR				MAC payload				MFR

Figure 45—Beacon frame format

The GTS fields shall be formatted as illustrated in Figure 46, and the pending address fields shall be formatted as illustrated in Figure 47.

The order of the fields of the beacon frame shall conform to the order of the general MAC frame as illustrated in Figure 41.

Octets: 1	0/1	variable
GTS specification	GTS directions	GTS list

Figure 46—Format of the GTS information fields

Octets: 1	variable
Pending address specification	Address list

Figure 47—Format of the pending address information fields

7.2.2.1.1 Beacon frame MHR fields

The MHR for a beacon frame shall contain the frame control field, the sequence number field, the source PAN identifier field and the source address field.

In the frame control field, the frame type subfield shall contain the value that indicates a beacon frame, as shown in Table 79, and the source addressing mode subfield shall be set as appropriate for the address of the coordinator transmitting the beacon frame. If protection is used for the beacon, the security enabled subfield shall be set to one. The frame version subfield shall be set to one only if the security enabled subfield is set to one. If a broadcast data or command frame is pending, the frame pending subfield shall be set to one. All other subfields shall be set to zero and ignored on reception.

The sequence number field shall contain the current value of *macBSN*.

The addressing fields shall comprise only the source address fields. The source PAN identifier and source address fields shall contain the PAN identifier and address, respectively, of the device transmitting the beacon.

The auxiliary security header field, if present, shall contain the information required for security processing of the beacon frame, as specified in 7.2.1.7.

7.2.2.1.2 Superframe specification field

The superframe specification field is 16 bits in length and shall be formatted as illustrated in Figure 48.

Bits: 0-3	4-7	8-11	12	13	14	15
Beacon order	Superframe order	Final CAP slot	Battery life extension (BLE)	Reserved	PAN coordinator	Association permit

Figure 48—Format of the superframe specification field

The beacon order subfield is 4 bits in length and shall specify the transmission interval of the beacon. See 7.5.1.1 for an explanation of the relationship between the beacon order and the beacon interval.

The superframe order subfield is 4 bits in length and shall specify the length of time during which the superframe is active (i.e., receiver enabled), including the beacon frame transmission time. See 7.5.1.1 for an explanation of the relationship between the superframe order and the superframe duration.

The final CAP slot subfield is four bits in length and specifies the final superframe slot utilized by the CAP. The duration of the CAP, as implied by this subfield, shall be greater than or equal to the value specified by *aMinCAPLength*. However, an exception is allowed for the accommodation of the temporary increase in the beacon frame length needed to perform GTS maintenance (see 7.2.2.1.3).

The battery life extension (BLE) subfield is 1 bit in length and shall be set to one if frames transmitted to the beaconing device during its CAP are required to start in or before *macBattLifeExtPeriods* full backoff periods after the IFS period following the beacon. Otherwise, the BLE subfield shall be set to zero.

The PAN coordinator subfield is 1 bit in length and shall be set to one if the beacon frame is being transmitted by the PAN coordinator. Otherwise, the PAN coordinator subfield shall be set to zero.

The association permit subfield is 1 bit in length and shall be set to one if *macAssociationPermit* is set to TRUE (i.e., the coordinator is accepting association to the PAN). The association permit bit shall be set to zero if the coordinator is currently not accepting association requests on its network.

7.2.2.1.3 GTS specification field

The GTS specification field is 8 bits in length and shall be formatted as illustrated in Figure 49.

Bits: 0-2	3-6	7
GTS descriptor count	Reserved	GTS permit

Figure 49—Format of the GTS specification field

The GTS descriptor count subfield is 3 bits in length and specifies the number of three-octet GTS descriptors contained in the GTS list field of the beacon frame. If the value of this subfield is greater than zero, the size of the CAP shall be allowed to dip below *aMinCAPLength* to accommodate the temporary increase in the beacon frame length caused by the inclusion of the subfield. If the value of this subfield is zero, the GTS directions field and GTS list field of the beacon frame are not present.

The GTS permit subfield is 1 bit in length and shall be set to one if *macGTSPermit* is equal to TRUE (i.e., the PAN coordinator is accepting GTS requests). Otherwise, the GTS permit field shall be set to zero.

7.2.2.1.4 GTS directions field

The GTS directions field is 8 bits in length and shall be formatted as illustrated in Figure 50.

Bits: 0-6	7
GTS directions mask	Reserved

Figure 50—Format of the GTS directions field

The GTS directions mask subfield is 7 bits in length and contains a mask identifying the directions of the GTSs in the superframe. The lowest bit in the mask corresponds to the direction of the first GTS contained in the GTS list field of the beacon frame, with the remainder appearing in the order that they appear in the list.

Each bit shall be set to one if the GTS is a receive-only GTS or to zero if the GTS is a transmit-only GTS. GTS direction is defined relative to the direction of the data frame transmission by the device.

7.2.2.1.5 GTS list field

The size of the GTS list field is defined by the values specified in the GTS specification field of the beacon frame and contains the list of GTS descriptors that represents the GTSs that are being maintained. The maximum number of GTS descriptors shall be limited to seven.

Each GTS descriptor is 24 bits in length and shall be formatted as illustrated in Figure 51.

Bits: 0-15	16-19	20-23
Device short address	GTS starting slot	GTS length

Figure 51—Format of the GTS descriptor

The device short address subfield is 16 bits in length and shall contain the short address of the device for which the GTS descriptor is intended.

The GTS starting slot subfield is 4 bits in length and contains the superframe slot at which the GTS is to begin.

The GTS length subfield is 4 bits in length and contains the number of contiguous superframe slots over which the GTS is active.

7.2.2.1.6 Pending address specification field

The pending address specification field shall be formatted as illustrated in Figure 52.

Bits: 0—2	3	4—6	7
Number of short addresses pending	Reserved	Number of extended addresses pending	Reserved

Figure 52—Format of the pending address specification field

The number of short addresses pending subfield is 3 bits in length and indicates the number of 16-bit short addresses contained in the address list field of the beacon frame.

The number of extended addresses pending subfield is 3 bits in length and indicates the number of 64-bit extended addresses contained in the address list field of the beacon frame.

7.2.2.1.7 Address list field

The size of the address list field is determined by the values specified in the pending address specification field of the beacon frame and contains the list of addresses of the devices that currently have messages pending with the coordinator. The address list shall not contain the broadcast short address 0xffff.

The maximum number of addresses pending shall be limited to seven and may comprise both short and extended addresses. All pending short addresses shall appear first in the list followed by any extended addresses. If the coordinator is able to store more than seven transactions, it shall indicate them in its beacon on a first-come-first-served basis, ensuring that the beacon frame contains at most seven addresses.

7.2.2.1.8 Beacon payload field

The beacon payload field is an optional sequence of up to *aMaxBeaconPayloadLength* octets specified to be transmitted in the beacon frame by the next higher layer. The set of octets contained in *macBeaconPayload* shall be copied into this field.

7.2.2.2 Data frame format

The data frame shall be formatted as illustrated in Figure 53.

Octets: 1/2	**1**	**(see 7.2.2.2.1)**	**0/5/6/10/14**	**variable**	**0/2**
Frame control	Sequence number	Addressing fields	Auxiliary security header	Data payload	FCS
MHR				MAC payload	MFR

Figure 53—Data frame format

The order of the fields of the data frame shall conform to the order of the general MAC frame as illustrated in Figure 41.

7.2.2.2.1 Data frame MHR fields

The MHR for a data frame shall contain the frame control field, the sequence number field, the destination PAN identifier/address fields and/or the source PAN identifier/address fields.

In the frame control field, the frame type subfield shall contain the value that indicates a data frame, as shown in Table 79. If protection is used for the data, the security enabled subfield shall be set to one. The frame version subfield shall be set to one if either the security enabled subfield is set to one or the MAC payload field is greater than *aMaxMACSafePayloadSize*. All other subfields shall be set appropriately according to the intended use of the data frame. All reserved subfields shall be set to zero and ignored on reception.

The sequence number field shall contain the current value of *macDSN*.

The addressing fields shall comprise the destination address fields and/or the source address fields, dependent on the settings in the frame control field.

The auxiliary security header field, if present, shall contain the information required for security processing of the data frame, as specified in 7.2.1.7.

7.2.2.2.2 Data payload field

The payload of a data frame shall contain the sequence of octets that the next higher layer has requested the MAC sublayer to transmit.

7.2.2.3 Acknowledgment frame format

The acknowledgment frame shall be formatted as illustrated in Figure 54.

Octets: 1/2	1	(see 7.2.2.2.1)	0/5/6/10/14	variable	0/2
Frame control	Sequence number	Addressing fields	Auxiliary security header	Acknowledge-ment payload	FCS
MHR				MAC payload	MFR

Figure 54—Acknowledgement frame format

The order of the fields of the acknowledgment frame shall conform to the order of the general MAC frame as illustrated in Figure 41.

7.2.2.3.1 Acknowledgment frame MHR fields

The MHR for an acknowledgment frame shall contain only the frame control field and the sequence number field.

In the frame control field, the frame type subfield shall contain the value that indicates an acknowledgment frame, as shown in Table 79. If the acknowledgment frame is being sent in response to a received data request command, the device sending the acknowledgment frame shall determine whether it has data pending for the recipient. If the device can determine this before sending the acknowledgment frame (see 7.5.6.4.2), it shall set the frame pending subfield according to whether there is pending data. Otherwise, the frame pending subfield shall be set to one. If the acknowledgment frame is being sent in response to either a data frame or another type of MAC command frame, the device shall set the frame pending subfield to zero. All other subfields shall be set to zero and ignored on reception.

The sequence number field shall contain the value of the sequence number received in the frame for which the acknowledgment is to be sent.

The acknowledgement frames introduced with various proposals, such as TSCH, low-energy subgroup, and group acknolwedmgent, should be treated similarly as the various command frames and warrant a separate section. General format will follow the following principles:
- legacy 802.15.4-2006 acknowledgement frame is supported as special case (no payload, no security, etc.);
- if frame is sent with cryptographic protection, corresponding acknowledgement is sent with the same key, frame counter, and security level as indicated in security policy. Using the same key and frame counter allows compressing the auxiliary security header field and associated frame overhead entirely.
- if acknowledgement is sent with much delay, security processing which depends on correlation of frame counter with on-device time notion will fail or cause unexceptable ambiguities. This can be remedied, but may require a slight change of the incoming and outgoing frame security procedure (which may not entirely be an extension of 802.15.4-2006).
Coordination between various proposers required.

7.2.2.4 MAC command frame format

The MAC command frame shall be formatted as illustrated in Figure 55.

The order of the fields of the MAC command frame shall conform to the order of the general MAC frame as illustrated in Figure 41.

Octets: 2	1	(see 7.2.2.4.1)	0/5/6/10/14	1	variable	2
Frame control	Sequence number	Addressing fields	Auxiliary security header	Command frame identifier	Command payload	FCS
MHR				MAC payload		MFR

Figure 55—MAC command frame format

7.2.2.4.1 MAC command frame MHR fields

The MHR for a MAC command frame shall contain the frame control field, the sequence number field, the destination PAN identifier/address fields and/or the source PAN identifier/address fields.

In the frame control field, the frame type subfield shall contain the value that indicates a MAC command frame, as shown in Table 79. If the frame is to be secured, the security enabled subfield of the frame control field shall be set to one and the frame secured according to the process described in 7.5.8.1.3. Otherwise the security enabled subfield of the frame control field shall be set to zero. All other subfields shall be set appropriately according to the intended use of the MAC command frame. All reserved subfields shall be set to zero and ignored on reception.

The sequence number field shall contain the current value of *macDSN*.

The addressing fields shall comprise the destination address fields and/or the source address fields, dependent on the settings in the frame control field.

The auxiliary security header field, if present, shall contain the information required for security processing of the MAC command frame, as specified in 7.2.1.7.

7.2.2.4.2 Command frame identifier field

The command frame identifier field identifies the MAC command being used. This field shall be set to one of the nonreserved values listed in Table 82.

7.2.2.4.3 Command payload field

The command payload field contains the MAC command itself. The formats of the individual commands are described in 7.3.

7.2.3 Frame compatibility

All unsecured frames specified in IEEE P802.15.4REVb/D6 are compatible with unsecured IEEE Std 802.15.4-2003 frames with two exceptions: a coordinator realignment command frame with the channel page field present (see 7.3.8) and any frame with a MAC payload field larger than *aMaxMACSafePayloadSize* octets.

Compatibility for secured frames is shown in Table 81, which identifies the security operating modes for IEEE Std 802.15.4-2003 and IEEE P802.15.4REVb/D6.

Table 81—Frame compatibility between IEEE Std 802.15.4-2003 and IEEE P802.15.4REVb/D6

Frame control field bit assignments		Functionality
Security enabled b_3	Frame version b_{13} b_{12}	
0	00	No security. Frames are compatible between IEEE Std 802.15.4-2003 and IEEE P802.15.4REVb/D6.
0	01	No security. Frames are not compatible between IEEE Std 802.15.4-2003 and IEEE P802.15.4REVb/D6.
1	00	Secured frame in IEEE Std 802.15.4-2003 format. This type of frame is not supported in IEEE P802.15.4REVb/D6.
1	01	Secured frame in IEEE P802.15.4REVb/D6 format.

7.3 MAC command frames

The command frames defined by the MAC sublayer are listed in Table 82. An FFD shall be capable of transmitting and receiving all command frame types, with the exception of the GTS request command, while the requirements for an RFD are indicated by an “X” in the table. MAC commands shall only be transmitted in the CAP for beacon-enabled PANs or at any time for nonbeacon-enabled PANs.

How the MLME shall construct the individual commands for transmission is detailed in through 7.3.9. MAC command reception shall abide by the procedure described in 7.5.6.2.

Table 82—MAC command frames

Command frame identifier	Command name	RFD		Subclause
		Tx	Rx	
0x01	Association request	X		7.3.1
0x02	Association response		X	7.3.2
0x03	Disassociation notification	X	X	7.3.3
0x04	Data request	X		7.3.4
0x05	PAN ID conflict notification	X		7.3.5
0x06	Orphan notification	X		7.3.6
0x07	Beacon request			7.3.7
0x08	Coordinator realignment		X	7.3.8
0x09	GTS request			7.3.9
0x0a–0xff	Reserved			—

7.3.1 Association request command

The association request command allows a device to request association with a PAN through the PAN coordinator or a coordinator.

This command shall only be sent by an unassociated device that wishes to associate with a PAN. A device shall only associate with a PAN through the PAN coordinator or a coordinator allowing association, as determined through the scan procedure.

All devices shall be capable of transmitting this command, although an RFD is not required to be capable of receiving it.

The association request command shall be formatted as illustrated in Figure 56.

octets: (see 7.2.2.4)	**1**	**1**
MHR fields	Command frame identifier (see Table 82)	Capability information

Figure 56—Association request command format

7.3.1.1 MHR fields

The source addressing mode subfield of the frame control field shall be set to three (64-bit extended addressing). The destination addressing mode subfield shall be set to the same mode as indicated in the beacon frame to which the association request command refers.

The frame pending subfield of the frame control field shall be set to zero and ignored upon reception, and the acknowledgment request subfield shall be set to one.

The destination PAN identifier field shall contain the identifier of the PAN to which to associate. The destination address field shall contain the address from the beacon frame that was transmitted by the coordinator to which the association request command is being sent. The source PAN identifier field shall contain the broadcast PAN identifier (i.e., 0xffff). The source address field shall contain the value of *aExtendedAddress*.

7.3.1.2 Capability information field

The capability information field shall be formatted as illustrated in Figure 57.

bits: 0	**1**	**2**	**3**	**4-5**	**6**	**7**
Alternate PAN coordinator	Device type	Power source	Receiver on when idle	Reserved	Security capability	Allocate address

Figure 57—Capability information field format

The alternate PAN coordinator subfield is 1 bit in length and shall be set to one if the device is capable of becoming the PAN coordinator. Otherwise, the alternate PAN coordinator subfield shall be set to zero.

The device type subfield is 1 bit in length and shall be set to one if the device is an FFD. Otherwise, the device type subfield shall be set to zero to indicate an RFD.

The power source subfield is 1 bit in length and shall be set to one if the device is receiving power from the alternating current mains. Otherwise, the power source subfield shall be set to zero.

The receiver on when idle subfield is 1 bit in length and shall be set to one if the device does not disable its receiver to conserve power during idle periods. Otherwise, the receiver on when idle subfield shall be set to zero.

The security capability subfield is 1 bit in length and shall be set to one if the device is capable of sending and receiving cryptographically protected MAC frames as specified in 7.5.8.2; it shall be set to zero otherwise.

The allocate address subfield is 1 bit in length and shall be set to one if the device wishes the coordinator to allocate a 16-bit short address as a result of the association procedure. Otherwise, it shall be set to zero.

7.3.2 Association response command

The association response command allows the PAN coordinator or a coordinator to communicate the results of an association attempt back to the device requesting association.

This command shall only be sent by the PAN coordinator or coordinator to a device that is currently trying to associate.

All devices shall be capable of receiving this command, although an RFD is not required to be capable of transmitting it.

The association response command shall be formatted as illustrated in Figure 58.

octets: (see 7.2.2.4)	**1**	**2**	**1**
MHR fields	Command frame identifier (see Table 82)	Short address	Association status

Figure 58—Association response command format

7.3.2.1 MHR fields

The destination addressing mode and source addressing mode subfields of the frame control field shall each be set to three (i.e., 64-bit extended addressing).

The frame pending subfield of the frame control field shall be set to zero and ignored upon reception, and the acknowledgment request subfield shall be set to one.

The PAN ID compression subfield of the frame control field shall be set to one. In accordance with this value of the PAN ID compression subfield, the destination PAN identifier field shall contain the value of *macPANId*, while the source PAN identifier field shall be omitted. The destination address field shall contain the extended address of the device requesting association. The source address field shall contain the value of *aExtendedAddress*.

7.3.2.2 Short address field

If the coordinator was not able to associate this device to its PAN, the short address field shall be set to 0xffff, and the association status field shall contain the reason for the failure. If the coordinator was able to associate the device to its PAN, this field shall contain the short address that the device may use in its communications on the PAN until it is disassociated.

A short address field value equal to 0xfffe shall indicate that the device has been successfully associated with a PAN, but has not been allocated a short address. In this case, the device shall communicate on the PAN using only its 64-bit extended address.

7.3.2.3 Association status field

The association status field shall contain one of the nonreserved values listed in Table 83.

Table 83—Valid values of the association status field

Association status	Description
0x00	Association successful.
0x01	PAN at capacity.
0x02	PAN access denied.
0x03–0x7f	Reserved.
0x80–0xff	Reserved for MAC primitive enumeration values.

7.3.3 Disassociation notification command

Either the PAN coordinator, a coordinator or an associated device may send the disassociate notification command.

All devices shall implement this command.

The disassociation notification command shall be formatted as illustrated in Figure 59.

octets: (see 7.2.2.4)	1	1
MHR fields	Command frame identifier (see Table 82)	Disassociation reason

Figure 59—Disassociation notification command format

7.3.3.1 MHR fields

The destination addressing mode subfield of the frame control field shall be set according to the addressing mode specified by the corresponding primitive. The source addressing mode subfield shall be set to three (i.e., 64-bit extended addressing).

The frame pending subfield of the frame control field shall be set to zero and ignored upon reception, and the acknowledgment request subfield shall be set to one.

The PAN ID compression subfield of the frame control field shall be set to one. In accordance with this value of the PAN ID compression subfield, the destination PAN identifier field shall contain the value of *macPANId*, while the source PAN identifier field shall be omitted. If the coordinator wants an associated device to leave the PAN, then the destination address field shall contain the address of the device being removed from the PAN. If an associated device wants to leave the PAN, then the destination address field shall contain the value of either *macCoordShortAddress*, if the destination addressing mode subfield is equal to two, or *macCoordExtendedAddress*, if the destination addressing mode subfield is equal to three. The source address field shall contain the value of *aExtendedAddress*.

7.3.3.2 Disassociation reason field

The disassociation reason field shall contain one of the nonreserved values listed in Table 84.

Table 84—Valid disassociation reason codes

Disassociate reason	Description
0x00	Reserved.
0x01	The coordinator wishes the device to leave the PAN.
0x02	The device wishes to leave the PAN.
0x03–0x7f	Reserved.
0x80–0xff	Reserved for MAC primitive enumeration values.

7.3.4 Data request command

The data request command is sent by a device to request data from the PAN coordinator or a coordinator.

There are three cases for which this command is sent. On a beacon-enabled PAN, this command shall be sent by a device when *macAutoRequest* is equal to TRUE and a beacon frame indicating that data are pending for that device is received from its coordinator. The coordinator indicates pending data in its beacon frame by adding the address of the recipient of the data to the address list field. This command shall also be sent when instructed to do so by the next higher layer on reception of the MLME-POLL.request primitive. In addition, a device may send this command to the coordinator *macResponseWaitTime* symbols after the acknowledgment to an association request command.

All devices shall be capable of transmitting this command, although an RFD is not required to be capable of receiving it.

The data request command shall be formatted as illustrated in Figure 60.

octets: (see 7.2.2.4)	**1**
MHR fields	Command frame identifier (see Table 82)

Figure 60—Data request command format

If the data request command is being sent in response to the receipt of a beacon frame indicating that data are pending for that device, the destination addressing mode subfield of the frame control field may be set to zero (i.e., destination addressing information not present) if the beacon frame indicated in its superframe specification field (see 7.2.2.1.2) that it originated from the PAN coordinator (see 7.2.1.1.6) or set otherwise according to the coordinator to which the data request command is directed. If the destination addressing information is to be included, the destination addressing mode subfield shall be set according to the value of *macCoordShortAddress*. If *macCoordShortAddress* is equal to 0xfffe, extended addressing shall be used; the destination addressing mode subfield shall be set to three, and the destination address field shall contain the value of *macCoordExtendedAddress*. Short addressing shall be used otherwise; the destination addressing mode subfield shall be set to two, and the destination address field shall contain the value of *macCoordShortAddress*.

If the data request command is being sent in response to the receipt of a beacon frame indicating that data are pending for that device, the source addressing mode subfield shall be set according to the addressing mode used for the pending address. If the source addressing mode subfield is set to two, short addressing shall be used; the source address field shall contain the value of *macShortAddress*. Otherwise extended addressing shall be used; the source addressing mode subfield shall be set to three, and the source address field shall contain the value of *aExtendedAddress*.

If the data request command is triggered by the reception of an MLME-POLL.request primitive from the next higher layer, then the destination addressing information shall be the same as that contained in the primitive. The source addressing mode subfield shall be set according to the value of *macShortAddress*. If *macShortAddress* is less than 0xfffe, short addressing shall be used. Extended addressing shall be used otherwise.

If the data request command is being sent following the acknowledgment to an association request command frame, the destination addressing mode subfield of the frame control field shall be set according to the coordinator to which the data request command is directed. If *macCoordShortAddress* is equal to 0xfffe, extended addressing shall be used. Short addressing shall be used otherwise. The source addressing mode subfield shall be set to use extended addressing.

If the destination addressing mode subfield is set to zero (i.e., destination addressing information not present), the PAN ID compression subfield of the frame control field shall be set to zero and the source PAN identifier shall contain the value of *macPANId*. Otherwise, the PAN ID compression subfield shall be set to one. In this case and in accordance with the PAN ID compression subfield, the destination PAN identifier field shall contain the value of *macPANId*, while the source PAN identifier field shall be omitted.

The frame pending subfield of the frame control field shall be set to zero and ignored upon reception, and the acknowledgment request subfield shall be set to one.

7.3.5 PAN ID conflict notification command

The PAN ID conflict notification command is sent by a device to the PAN coordinator when a PAN identifier conflict is detected.

All devices shall be capable of transmitting this command, although an RFD is not required to be capable of receiving it.

The PAN ID conflict notification command shall be formatted as illustrated in Figure 61.

The destination addressing mode and source addressing mode subfields of the frame control field shall both be set to three (i.e., 64-bit extended addressing).

octets: (see 7.2.2.4)	1
MHR fields	Command frame identifier (see Table 82)

Figure 61—PAN ID conflict notification command format

The frame pending subfield of the frame control field shall be set to zero and ignored upon reception, and the acknowledgment request subfield shall be set to one.

The PAN ID compression subfield of the frame control field shall be set to one. In accordance with this value of the PAN ID compression subfield, the destination PAN identifier field shall contain the value of *macPANId*, while the source PAN identifier field shall be omitted. The destination address field shall contain the value of *macCoordExtendedAddress*. The source address field shall contain the value of *aExtendedAddress*.

7.3.6 Orphan notification command

The orphan notification command is used by an associated device that has lost synchronization with its coordinator.

All devices shall be capable of transmitting this command, although an RFD is not required to be capable of receiving it.

The orphan notification command shall be formatted as illustrated in Figure 62.

octets: 15	1
MHR fields	Command frame identifier (see Table 82)

Figure 62—Orphan notification command format

The source addressing mode subfield of the frame control field shall be set to three (i.e., 64-bit extended addressing). The destination addressing mode subfield shall be set to two (i.e., 16-bit short addressing).

The frame pending subfield and acknowledgment request subfield of the frame control field shall be set to zero and ignored upon reception.

The PAN ID compression subfield of the frame control field shall be set to one. In accordance with this value of the PAN ID compression subfield, the destination PAN identifier field shall contain the value of the broadcast PAN identifier (i.e., 0xffff), while the source PAN identifier field shall be omitted. The destination address field shall contain the broadcast short address (i.e., 0xffff). The source address field shall contain the value of *aExtendedAddress*.

7.3.7 Beacon request command

The beacon request command is used by a device to locate all coordinators within its POS during an active scan.

This command is optional for an RFD.

The beacon request command shall be formatted as illustrated in Figure 63.

octets: 7	1
MHR fields	Command frame identifier (see Table 82)

Figure 63—Beacon request command format

The destination addressing mode subfield of the frame control field shall be set to two (i.e., 16-bit short addressing), and the source addressing mode subfield shall be set to zero (i.e., source addressing information not present).

The frame pending subfield of the frame control field shall be set to zero and ignored upon reception. The acknowledgment request subfield and security enabled subfield shall also be set to zero.

The destination PAN identifier field shall contain the broadcast PAN identifier (i.e., 0xffff). The destination address field shall contain the broadcast short address (i.e., 0xffff).

7.3.8 Coordinator realignment command

The coordinator realignment command is sent by the PAN coordinator or a coordinator either following the reception of an orphan notification command from a device that is recognized to be on its PAN or when any of its PAN configuration attributes change due to the receipt of an MLME-START.request primitive.

If this command is sent following the reception of an orphan notification command, it is sent directly to the orphaned device. If this command is sent when any PAN configuration attributes (i.e., PAN identifier, short address, logical channel or channel page) change, it is broadcast to the PAN.

All devices shall be capable of receiving this command, although an RFD is not required to be capable of transmitting it.

The coordinator realignment command shall be formatted as illustrated in Figure 64.

octets: 17/18/23/24	1	2	2	1	2	0/1
MHR fields	Command frame identifier (see Table 82)	PAN identifier	Coordinator short address	Logical channel	Short address	Channel page

Figure 64—Coordinator realignment command format

7.3.8.1 MHR fields

The destination addressing mode subfield of the frame control field shall be set to three (e.g., 64-bit extended addressing) if the command is directed to an orphaned device or set to two (e.g., 16-bit short addressing) if it is to be broadcast to the PAN. The source addressing mode subfield of the frame control field shall be set to three (e.g., 64-bit extended addressing).

The frame pending subfield of the frame control field shall be set to zero and ignored upon reception.

The acknowledgment request subfield of the frame control field shall be set to one if the command is directed to an orphaned device or set to zero if the command is to be broadcast to the PAN.

The frame version subfield shall be set to 0x01 if the channel page field is present. Otherwise it shall be set as specified in 7.2.3.

The destination PAN identifier field shall contain the broadcast PAN identifier (e.g., 0xffff). The destination address field shall contain the extended address of the orphaned device if the command is directed to an orphaned device. Otherwise, the destination address field shall contain the broadcast short address (e.g., 0xffff). The source PAN identifier field shall contain the value of *macPANId*, and the source address field shall contain the value of *aExtendedAddress*.

7.3.8.2 PAN identifier field

The PAN identifier field shall contain the PAN identifier that the coordinator intends to use for all future communications.

7.3.8.3 Coordinator short address field

The coordinator short address field shall contain the value of *macShortAddress*.

7.3.8.4 Logical channel field

The logical channel field shall contain the logical channel that the coordinator intends to use for all future communications.

7.3.8.5 Short address field

If the coordinator realignment command is broadcast to the PAN, the short address field shall be set to 0xffff and ignored on reception.

If the coordinator realignment command is sent directly to an orphaned device, this field shall contain the short address that the orphaned device shall use to operate on the PAN. If the orphaned device does not have a short address, because it always uses its 64-bit extended address, this field shall contain the value 0xfffe.

7.3.8.6 Channel page field

The channel page field, if present, shall contain the channel page that the coordinator intends to use for all future communications. This field may be omitted if the new channel page is the same as the previous channel page.

7.3.9 GTS request command

The GTS request command is used by an associated device that is requesting the allocation of a new GTS or the deallocation of an existing GTS from the PAN coordinator. Only devices that have a 16-bit short address less than 0xfffe shall send this command.

This command is optional.

The GTS request command shall be formatted as illustrated in Figure 65.

octets: 7	1	1
MHR fields	Command frame identifier (see Table 82)	GTS characteristics

Figure 65—GTS request command format

7.3.9.1 MHR fields

The destination addressing mode subfield of the frame control field shall be set to zero (e.g., destination addressing information not present), and the source addressing mode subfield shall be set to two (e.g., 16-bit short addressing).

The frame pending subfield of the frame control field shall be set to zero and ignored upon reception, and the acknowledgment request subfield shall be set to one.

The source PAN identifier field shall contain the value of *macPANId*, and the source address field shall contain the value of *macShortAddress*.

7.3.9.2 GTS characteristics field

The GTS characteristics field shall be formatted as illustrated in Figure 66.

bits: 0–3	4	5	6–7
GTS length	GTS direction	Characteristics type	Reserved

Figure 66—GTS characteristics field format

The GTS length subfield shall contain the number of superframe slots being requested for the GTS.

The GTS direction subfield shall be set to one if the GTS is to be a receive-only GTS. Conversely, this subfield shall be set to zero if the GTS is to be a transmit-only GTS. GTS direction is defined relative to the direction of data frame transmissions by the device.

The characteristics type field shall be set to one if the characteristics refers to a GTS allocation or zero if the characteristics refers to a GTS deallocation.

7.4 MAC constants and PIB attributes

This subclause specifies the constants and attributes required by the MAC sublayer.

7.4.1 MAC constants

The constants that define the characteristics of the MAC sublayer are presented in Table 85.

7.4.2 MAC PIB attributes

The MAC PIB comprises the attributes required to manage the MAC sublayer of a device. The attributes contained in the MAC PIB are presented in Table 86. Attributes marked with a dagger (†) are read-only

Table 85—MAC sublayer constants

Constant	Description	Value
aBaseSlotDuration	The number of symbols forming a superframe slot when the superframe order is equal to 0 (see 7.5.1.1).	60
aBaseSuperframeDuration	The number of symbols forming a superframe when the superframe order is equal to 0.	*aBaseSlotDuration* * *aNumSuperframeSlots*
aExtendedAddress	The 64-bit (IEEE) address assigned to the device.	Device specific
aGTSDescPersistenceTime	The number of superframes in which a GTS descriptor exists in the beacon frame of the PAN coordinator.	4
aMaxBeaconOverhead	The maximum number of octets added by the MAC sublayer to the MAC payload of a beacon frame.	75
aMaxBeaconPayloadLength	The maximum size, in octets, of a beacon payload.	*aMaxPHYPacketSize* – *aMaxBeaconOverhead*
aMaxLostBeacons	The number of consecutive lost beacons that will cause the MAC sublayer of a receiving device to declare a loss of synchronization.	4
aMaxMACSafePayloadSize	The maximum number of octets that can be transmitted in the MAC payload field of an unsecured MAC frame that will be guaranteed not to exceed *aMaxPHYPacketSize*.	*aMaxPHYPacketSize* – *aMaxMPDUUnsecuredOverhead*
aMaxMACPayloadSize	The maximum number of octets that can be transmitted in the MAC payload field.	*aMaxPHYPacketSize* – *aMinMPDUOverhead*
aMaxMPDUUnsecuredOverhead	The maximum number of octets added by the MAC sublayer to the PSDU without security.	25
aMaxSIFSFrameSize	The maximum size of an MPDU, in octets, that can be followed by a short interframe spacing (SIFS) period.	18
aMinCAPLength	The minimum number of symbols forming the CAP. This ensures that MAC commands can still be transferred to devices when GTSs are being used. An exception to this minimum shall be allowed for the accommodation of the temporary increase in the beacon frame length needed to perform GTS maintenance (see 7.2.2.1.3).	440
aMinMPDUOverhead	The minimum number of octets added by the MAC sublayer to the PSDU.	9
aNumSuperframeSlots	The number of slots contained in any superframe.	16
aUnitBackoffPeriod	The number of symbols forming the basic time period used by the CSMA-CA algorithm.	20

attributes (i.e., attribute can only be set by the MAC sublayer), which can be read by the next higher layer using the MLME-GET.request primitive. All other attributes can be read or written by the next higher layer using the MLME-GET.request or MLME-SET.request primitives, respectively. Attributes marked with a diamond (◆) are optional for an RFD; attributes marked with an asterisk (*) are optional for both device types (i.e., RFD and FFD).

The read-only attribute *macAckWaitDuration* is dependent on a combination of constants and PHY PIB attributes. The formula for relating the constants and attributes is as follows:

macAckWaitDuration = (13)

$$aUnitBackoffPeriod + aTurnaroundTime + phySHRDuration + \lceil 6 \cdot phySymbolsPerOctet \rceil$$

where the number six represents the number of PHY header octets plus the number of PSDU octets in an acknowledgment frame.

The attribute *macMaxFrameTotalWaitTime* may be set by the next higher layer and is dependent upon a combination of PHY and MAC PIB attributes and constants. The formula relating the attributes and constants is as follows:

macMaxFrameTotalWaitTime = (14)

$$\left[\left(\sum_{k=0}^{m-1} 2^{macMinBE+k}\right) + (2^{macMaxBE} - 1) \cdot (macMaxCSMABackoffs - m)\right] \bullet aUnitBackoffPeriod + phyMaxFrameDuration$$

where m = min(*macMaxBE-macMinBE, macMaxCSMABackoffs).*

Table 86—MAC PIB attributes

Attribute	Identifier	Type	Range	Description	Default
macAckWaitDuration†	0x40	Integer	See Equation (13)	The maximum number of symbols to wait for an acknowledgment frame to arrive following a transmitted data frame. This value is dependent on the supported PHY, which determines both the selected logical channel and channel page. The calculated value is the time to commence transmitting the ACK plus the length of the ACK frame. The commencement time is described in 7.5.6.4.2.	Dependent on currently selected PHY, indicated by *phyCurrentPage*
macAssociatedPANCoord	0x56	Boolean	TRUE or FALSE	This indicates whether the device is associated to the PAN through the PAN coordinator. A value of TRUE indicates the device has associated through the PAN coordinator. Otherwise, the value is set to FALSE.	FALSE
macAssociationPermit◆	0x41	Boolean	TRUE or FALSE	Indication of whether a coordinator is currently allowing association. A value of TRUE indicates that association is permitted.	FALSE

Table 86—MAC PIB attributes *(continued)*

Attribute	Identifier	Type	Range	Description	Default
macAutoRequest	0x42	Boolean	TRUE or FALSE	Indication of whether a device automatically sends a data request command if its address is listed in the beacon frame. A value of TRUE indicates that the data request command is automatically sent. This attribute also affects the generation of the MLME-BEACON-NOTIFY.indication primitive (see 7.1.5.1.2).	TRUE
macBattLifeExt	0x43	Boolean	TRUE or FALSE	Indication of whether battery life extension, through the reduction of coordinator receiver operation time during the CAP, is enabled. A value of TRUE indicates that it is enabled. Also, see 7.5.1.4 for an explanation of how this attribute affects the backoff exponent in the CSMA-CA algorithm.	FALSE
macBattLifeExtPeriods	0x44	Integer	6-41	In battery life extension mode, the number of backoff periods during which the receiver is enabled after the IFS following a beacon. This value is dependent on the supported PHY and is the sum of three terms: Term 1: The value $2^x - 1$, where x is the maximum value of *macMinBE* in battery life extension mode (equal to two). This term is thus equal to 3 backoff periods. Term 2: The duration of the initial contention window length (see 7.5.1.4). This term is thus equal to 2 backoff periods. Term 3: The preamble field length and the SFD field length of the supported PHY (see Table 19 and Table 20 in Clause 6), summed together and rounded up (if necessary) to an integer number of backoff periods.	Dependent on currently selected PHY, indicated by *phyCurrentPage*
macBeaconPayload◆	0x45	Set of octets	—	The contents of the beacon payload.	NULL

Table 86—MAC PIB attributes *(continued)*

Attribute	Identifier	Type	Range	Description	Default
macBeaconPayload-Length◆	0x46	Integer	0 – *aMax-BeaconPayload-Length*	The length, in octets, of the beacon payload.	0
macBeaconOrder◆	0x47	Integer	0–15	Specification of how often the coordinator transmits its beacon. If *BO* = 15, the coordinator will not transmit a periodic beacon. See 7.5.1.1 for an explanation of the relationship between the beacon order and the beacon interval.	15
macBeaconTxTime†◆	0x48	Integer	0x000000 –0xffffff	The time that the device transmitted its last beacon frame, in symbol periods. The measurement shall be taken at the same symbol boundary within every transmitted beacon frame, the location of which is implementation specific. This is a 24-bit value, and the precision of this value shall be a minimum of 20 bits, with the lowest four bits being the least significant.	0x000000
macBSN◆	0x49	Integer	0x00–0xff	The sequence number added to the transmitted beacon frame.	Random value from within the range
macCoordExtended-Address	0x4a	IEEE address	An extended 64-bit IEEE address	The 64-bit address of the coordinator through which the device is associated.	—
macCoordShort-Address	0x4b	Integer	0x0000–0xffff	The 16-bit short address assigned to the coordinator through which the device is associated. A value of 0xfffe indicates that the coordinator is only using its 64-bit extended address. A value of 0xffff indicates that this value is unknown.	0xffff
macDSN	0x4c	Integer	0x00–0xff	The sequence number added to the transmitted data or MAC command frame.	Random value from within the range
*macGTSPermit**	0x4d	Boolean	TRUE or FALSE	TRUE if the PAN coordinator is to accept GTS requests. FALSE otherwise.	TRUE

Table 86—MAC PIB attributes *(continued)*

Attribute	Identifier	Type	Range	Description	Default
macMaxBE	0x57	Integer	3–8	The maximum value of the backoff exponent, BE, in the CSMA-CA algorithm. See 7.5.1.4 for a detailed explanation of the backoff exponent.	5
macMaxCSMABackoffs	0x4e	Integer	0–5	The maximum number of backoffs the CSMA-CA algorithm will attempt before declaring a channel access failure.	4
macMaxFrameTotalWaitTime	0x58	Integer	See Equation (14)	The maximum number of CAP symbols in a beacon-enabled PAN, or symbols in a nonbeacon-enabled PAN, to either wait for a frame intended as a response to a data request frame or to wait for a broadcast frame following a beacon with the frame pending subfield set to one. This attribute, which shall only be set by the next higher layer, is dependent upon *macMinBE*, *macMaxBE*, *macMaxCSMABackoffs* and the number of symbols per octet. See 7.4.2 for the formula relating the attributes.	Dependent on currently selected PHY, indicated by *phyCurrentPage*
macMaxFrameRetries	0x59	Integer	0–7	The maximum number of retries allowed after a transmission failure.	3
macMinBE	0x4f	Integer	0–*macMaxBE*	The minimum value of the backoff exponent, BE, in the CSMA-CA algorithm. See 7.5.1.4 for a detailed explanation of the backoff exponent.	3
macMinLIFSPeriod†		Integer	See Table 3 in Clause 6	The minimum number of symbols forming a long interframe spacing (LIFS) period.	Dependent on currently selected PHY, indicated by *phyCurrentPage*
macMinSIFSPeriod†		Integer	See Table 3 in Clause 6	The minimum number of symbols forming a short interframe spacing (SIFS) period.	Dependent on currently selected PHY, indicated by *phyCurrentPage*

Table 86—MAC PIB attributes *(continued)*

Attribute	Identifier	Type	Range	Description	Default
macPANId	0x50	Integer	0x0000–0xffff	The 16-bit identifier of the PAN on which the device is operating. If this value is 0xffff, the device is not associated.	0xffff
macPromiscuousMode◆	0x51	Boolean	TRUE or FALSE	This indicates whether the MAC sublayer is in a promiscuous (receive all) mode. A value of TRUE indicates that the MAC sublayer accepts all frames received from the PHY.	FALSE
macResponseWaitTime	0x5a	Integer	2–64	The maximum time, in multiples of *aBaseSuperframeDuration*, a device shall wait for a response command frame to be available following a request command frame.	32
macRxOnWhenIdle	0x52	Boolean	TRUE or FALSE	This indicates whether the MAC sublayer is to enable its receiver during idle periods. For a beacon-enabled PAN, this attribute is only relevant during the CAP of the incoming superframe. For a nonbeacon-enabled PAN, this attribute is relevant at all times.	FALSE
macSecurityEnabled	0x5d	Boolean	TRUE or FALSE	An indicator of whether the MAC sublayer has security enabled. A value of TRUE indicates that security is enabled, while a value of FALSE indicates that security is disabled.	FALSE
macShortAddress	0x53	Integer	0x0000–0xffff	The 16-bit address that the device uses to communicate in the PAN. If the device is the PAN coordinator, this value shall be chosen before a PAN is started. Otherwise, the address is allocated by a coordinator during association. A value of 0xfffe indicates that the device has associated but has not been allocated an address. A value of 0xffff indicates that the device does not have a short address.	0xffff

Table 86—MAC PIB attributes *(continued)*

Attribute	Identifier	Type	Range	Description	Default
macSuperframeOrder†◆	0x54	Integer	0–15	This specifies the length of the active portion of the outgoing superframe, including the beacon frame. If *SO* = 15, the superframe will not be active following the beacon. See 7.5.1.1 for an explanation of the relationship between the superframe order and the superframe duration.	15
macSyncSymbolOffset†	0x5b	Integer	0x000–0x100 for the 2.4GHz PHY 0x000–0x400 for the 868/915MHz PHY	The offset, measured in symbols, between the symbol boundary at which the MLME captures the timestamp of each transmitted or received frame, and the onset of the first symbol past the SFD, namely, the first symbol of the length field.	Implementation specific
macTimestampSupported†	0x5c	Boolean	TRUE or FALSE	This indicates whether the MAC sublayer supports the optional timestamping feature for incoming and outgoing data frames.	Implementation specific
macTransactionPersistenceTime◆	0x55	Integer	0x0000–0xffff	The maximum time (in unit periods) that a transaction is stored by a coordinator and indicated in its beacon. The unit period is governed by *macBeaconOrder*, *BO*, as follows: For $0 \le BO \le 14$, the unit period will be *aBaseSuperframeDuration* * 2^{BO}. For *BO = 15*, the unit period will be *aBaseSuperframeDuration*.	0x01f4

7.5 MAC functional description

This subclause provides a detailed description of the MAC functionality. Subclause 7.5.1 describes the following two mechanisms for channel access: contention based and contention free. Contention-based access allows devices to access the channel in a distributed fashion using a CSMA-CA backoff algorithm. Contention-free access is controlled entirely by the PAN coordinator through the use of GTSs.

The mechanisms used for starting and maintaining a PAN are described in 7.5.2. Channel scanning is used by a device to assess the current state of a channel (or channels), locate all beacons within its POS, or locate a particular beacon with which it has lost synchronization. Before starting a new PAN, the results of a channel scan can be used to select an appropriate logical channel and channel page, as well as a PAN identifier that is not being used by any other PAN in the area. Because it is still possible for the POS of two PANs with the same PAN identifier to overlap, a procedure exists to detect and resolve this situation. Following a channel scan and suitable PAN identifier selection, an FFD can begin operating as the PAN coordinator. Also

described in the subclause is a method to allow a beaconing FFD to discover other such devices during normal operations, i.e., when not scanning.

The mechanisms to allow devices to join or leave a PAN are defined in 7.5.3. The association procedure describes the conditions under which a device may join a PAN and the conditions necessary for a coordinator to permit devices to join. Also described is the disassociation procedure, which can be initiated by the associated device or its coordinator.

The mechanisms to allow devices to acquire and maintain synchronization with a coordinator are described in 7.5.4. Synchronization on a beacon-enabled PAN is described after first explaining how a coordinator generates beacon frames. Following this explanation, synchronization on a nonbeacon-enabled PAN is described. Also described is a procedure to reestablish communication between a device and its coordinator, as it is possible that a device may lose synchronization in the case of either a beacon-enabled or a nonbeacon-enabled PAN.

IEEE P802.15.4REVb/D6 has been designed so that application data transfers can be controlled by the devices on a PAN rather than by the coordinator. The procedures the coordinator uses to handle multiple transactions while preserving this requirement are described in 7.5.5.

The mechanisms for transmitting, receiving, and acknowledging frames, including frames sent using indirect transmission, are described in 7.5.6. In addition, methods for retransmitting frames are also described.

The mechanisms for allocating and deallocating a GTS are described in 7.5.7. The deallocation process may result in the fragmentation of the GTS space, i.e., an unused slot or slots. The subclause describes a mechanism to resolve fragmentation.

The MAC sublayer uses the mechanisms described in 7.5.8 for all incoming and outgoing frames.

Throughout this subclause, the receipt of a frame is defined as the successful receipt of the frame by the PHY and the successful verification of the FCS by the MAC sublayer, as described in 7.2.1.9.

7.5.1 Channel access

This subclause describes the mechanisms for accessing the physical radio channel.

7.5.1.1 Superframe structure

A coordinator on a PAN can optionally bound its channel time using a superframe structure. A superframe is bounded by the transmission of a beacon frame and can have an active portion and an inactive portion. The coordinator may enter a low power (sleep) mode during the inactive portion.

The structure of this superframe is described by the values of *macBeaconOrder* and *macSuperframeOrder.* The MAC PIB attribute *macBeaconOrder*, describes the interval at which the coordinator shall transmit its beacon frames. The value of *macBeaconOrder*, *BO*, and the beacon interval, *BI*, are related as follows: for $0 \leq BO \leq 14$, $BI = aBaseSuperframeDuration * 2^{BO}$ symbols. If $BO = 15$, the coordinator shall not transmit beacon frames except when requested to do so, such as on receipt of a beacon request command. The value of *macSuperframeOrder* shall be ignored if $BO = 15$.

The MAC PIB attribute *macSuperframeOrder* describes the length of the active portion of the superframe, which includes the beacon frame. The value of *macSuperframeOrder*, *SO*, and the superframe duration, *SD*, are related as follows: for $0 \leq SO \leq BO \leq 14$, $SD = aBaseSuperframeDuration * 2^{SO}$ symbols. If $SO = 15$, the superframe shall not remain active after the beacon. If $BO = 15$, the superframe shall not exist (the value of *macSuperframeOrder* shall be ignored), and *macRxOnWhenIdle* shall define whether the receiver is enabled during periods of transceiver inactivity.

The active portion of each superframe shall be divided into *aNumSuperframeSlots* equally spaced slots of duration 2^{SO} * *aBaseSlotDuration* and is composed of three parts: a beacon, a CAP and a CFP. The beacon shall be transmitted, without the use of CSMA, at the start of slot 0, and the CAP shall commence immediately following the beacon. The start of slot 0 is defined as the point at which the first symbol of the beacon PPDU is transmitted. The CFP, if present, follows immediately after the CAP and extends to the end of the active portion of the superframe. Any allocated GTSs shall be located within the CFP.

The MAC sublayer shall ensure that the integrity of the superframe timing is maintained, e.g. compensating for clock drift error.

PANs that wish to use the superframe structure (referred to as a beacon-enabled PAN) shall set *macBeaconOrder* to a value between 0 and 14, both inclusive, and *macSuperframeOrder* to a value between 0 and the value of *macBeaconOrder*, both inclusive.

PANs that do not wish to use the superframe structure (referred to as a nonbeacon-enabled PAN) shall set both *macBeaconOrder* and *macSuperframeOrder* to 15. In this case, a coordinator shall not transmit beacons, except upon receipt of a beacon request command; all transmissions, with the exception of acknowledgment frames and any data frame that quickly follows the acknowledgment of a data request command (see 7.5.6.3), shall use an unslotted CSMA-CA mechanism to access the channel. In addition, GTSs shall not be permitted.

An example of a superframe structure is shown in Figure 67. In this case, the beacon interval, *BI*, is twice as long as the active superframe duration, *SD*, and the CFP contains two GTSs.

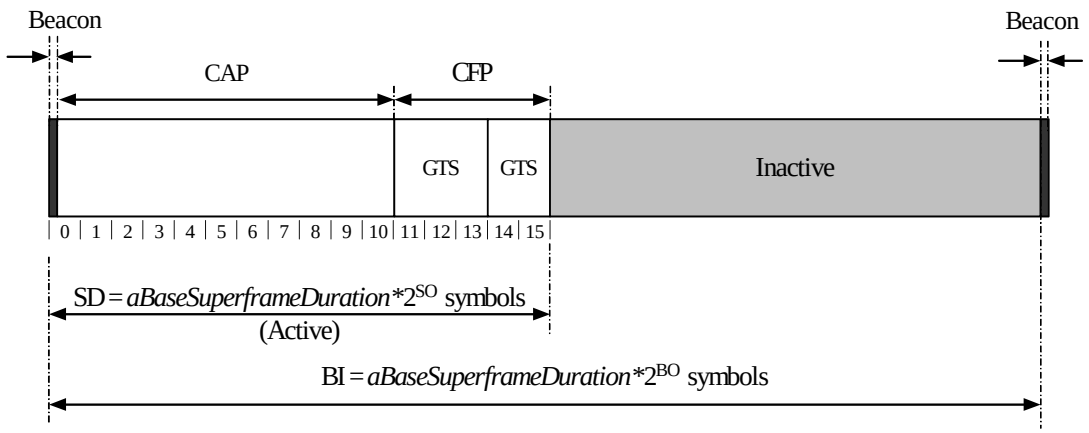


Figure 67—An example of the superframe structure

7.5.1.1.1 CAP

The CAP shall start immediately following the beacon and complete before the beginning of the CFP on a superframe slot boundary. If the CFP is zero length, the CAP shall complete at the end of the active portion of the superframe. The CAP shall be at least *aMinCAPLength* symbols, unless additional space is needed to temporarily accommodate the increase in the beacon frame length needed to perform GTS maintenance (see 7.2.2.1.3), and shall shrink or grow dynamically to accommodate the size of the CFP.

All frames, except acknowledgment frames and any data frame that quickly follows the acknowledgment of a data request command (see 7.5.6.3), transmitted in the CAP shall use a slotted CSMA-CA mechanism to access the channel. A device transmitting within the CAP shall ensure that its transaction is complete (i.e., including the reception of any acknowledgment) one IFS period (see 7.5.1.3) before the end of the CAP. If this is not possible, the device shall defer its transmission until the CAP of the following superframe.

MAC command frames shall always be transmitted in the CAP.

7.5.1.1.2 CFP

The CFP shall start on a slot boundary immediately following the CAP and it shall complete before the end of the active portion of the superframe. If any GTSs have been allocated by the PAN coordinator, they shall be located within the CFP and occupy contiguous slots. The CFP shall therefore grow or shrink depending on the total length of all of the combined GTSs.

No transmissions within the CFP shall use a CSMA-CA mechanism to access the channel. A device transmitting in the CFP shall ensure that its transmissions are complete one IFS period (see 7.5.1.3) before the end of its GTS.

7.5.1.2 Incoming and outgoing superframe timing

On a beacon-enabled PAN, a coordinator that is not the PAN coordinator shall maintain the timing of both the superframe in which its coordinator transmits a beacon (the incoming superframe) and the superframe in which it transmits its own beacon (the outgoing superframe). The relative timing of these superframes is defined by the StartTime parameter of the MLME-START.request primitive (see 7.1.14.1 and 7.5.2.4). The relationship between incoming and outgoing superframes is illustrated in Figure 68.

The beacon order and superframe order shall be equal for all superframes on a PAN. All devices shall interact with the PAN only during the active portion of a superframe.

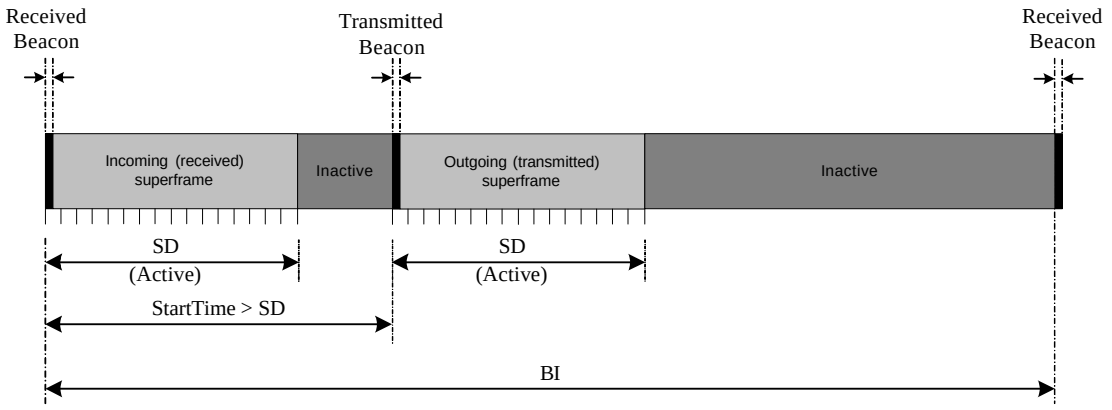


Figure 68—The relationship between incoming and outgoing beacons

7.5.1.3 IFS

The MAC sublayer needs a finite amount of time to process data received by the PHY. To allow for this, two successive frames transmitted from a device shall be separated by at least an IFS period; if the first transmission requires an acknowledgment, the separation between the acknowledgment frame and the second transmission shall be at least an IFS period. The length of the IFS period is dependent on the size of the frame that has just been transmitted. Frames (i.e., MPDUs) of up to *aMaxSIFSFrameSize* octets in length shall be followed by a SIFS period of a duration of at least *macMinSIFSPeriod* symbols. Frames (i.e., MPDUs) with

lengths greater than *aMaxSIFSFrameSize* octets shall be followed by a LIFS period of a duration of at least *macMinLIFSPeriod* symbols. These concepts are illustrated in Figure 69.

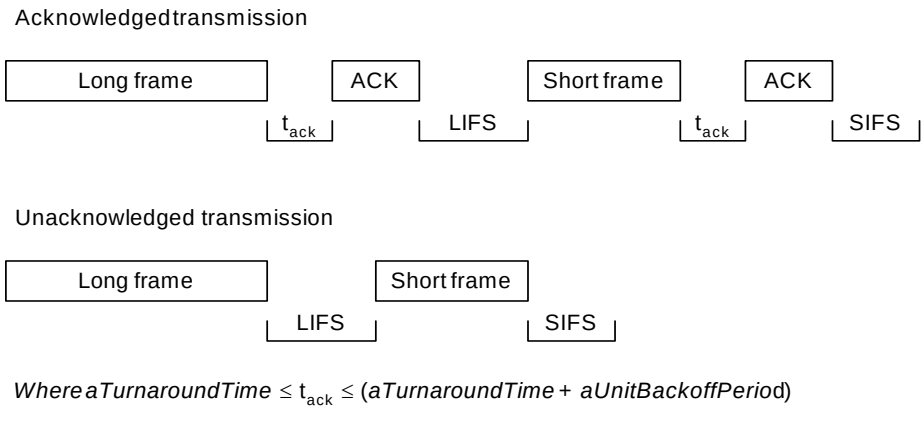


Figure 69—IFS

The CSMA-CA algorithm shall take this requirement into account for transmissions in the CAP.

7.5.1.4 The CSMA-CA algorithm

The CSMA-CA algorithm shall be used before the transmission of data or MAC command frames transmitted within the CAP, unless the frame can be quickly transmitted following the acknowledgment of a data request command (see 7.5.6.3 for timing requirements). The CSMA-CA algorithm shall not be used for the transmission of beacon frames in a beacon-enabled PAN, acknowledgment frames, or data frames transmitted in the CFP.

If periodic beacons are being used in the PAN, the MAC sublayer shall employ the slotted version of the CSMA-CA algorithm for transmissions in the CAP of the superframe. Conversely, if periodic beacons are not being used in the PAN or if a beacon could not be located in a beacon-enabled PAN, the MAC sublayer shall transmit using the unslotted version of the CSMA-CA algorithm. In both cases, the algorithm is implemented using units of time called backoff periods, where one backoff period shall be equal to *aUnitBackoffPeriod* symbols.

In slotted CSMA-CA, the backoff period boundaries of every device in the PAN shall be aligned with the superframe slot boundaries of the PAN coordinator, i.e., the start of the first backoff period of each device is aligned with the start of the beacon transmission. In slotted CSMA-CA, the MAC sublayer shall ensure that the PHY commences all of its transmissions on the boundary of a backoff period. In unslotted CSMA-CA, the backoff periods of one device are not related in time to the backoff periods of any other device in the PAN.

Each device shall maintain three variables for each transmission attempt: *NB*, *CW* and *BE*. *NB* is the number of times the CSMA-CA algorithm was required to backoff while attempting the current transmission; this value shall be initialized to zero before each new transmission attempt. *CW* is the contention window length, defining the number of backoff periods that need to be clear of channel activity before the transmission can commence; this value shall be initialized to two before each transmission attempt and reset to two each time the channel is assessed to be busy. The *CW* variable is only used for slotted CSMA-CA. *BE* is the backoff exponent, which is related to how many backoff periods a device shall wait before attempting to assess a channel. In unslotted systems, or slotted systems with the received BLE subfield (see Figure 48) set to zero, *BE* shall be initialized to the value of *macMinBE*. In slotted systems with the received BLE subfield set to

one, this value shall be initialized to the lesser of two and the value of *macMinBE*. Note that if *macMinBE* is set to zero, collision avoidance will be disabled during the first iteration of this algorithm.

Although the receiver of the device is enabled during the CCA analysis portion of this algorithm, the device may discard any frames received during this time.

Figure 70 illustrates the steps of the CSMA-CA algorithm. When using slotted CSMA-CA, the MAC sublayer shall first initialize *NB*, *CW*, and *BE* and then locate the boundary of the next backoff period [step (1)]. For unslotted CSMA-CA, the MAC sublayer shall initialize *NB* and *BE* and then proceed directly to step (2).

The MAC sublayer shall delay for a random number of complete backoff periods in the range 0 to $2^{BE} - 1$ [step (2)] and then request that the PHY perform a CCA [step (3)]. In a slotted CSMA-CA system, the CCA shall start on a backoff period boundary. In an unslotted CSMA-CA system, the CCA shall start immediately.

In a slotted CSMA-CA system with the BLE subfield set to zero, the MAC sublayer shall ensure that, after the random backoff, the remaining CSMA-CA operations can be undertaken and the entire transaction can be transmitted before the end of the CAP. Note that any bit padding used by the supported PHY (see 6.7.2.2) must be considered in making this determination. If the number of backoff periods is greater than the remaining number of backoff periods in the CAP, the MAC sublayer shall pause the backoff countdown at the end of the CAP and resume it at the start of the CAP in the next superframe. If the number of backoff periods is less than or equal to the remaining number of backoff periods in the CAP, the MAC sublayer shall apply its backoff delay and then evaluate whether it can proceed. The MAC sublayer shall proceed if the remaining CSMA-CA algorithm steps (i.e., two CCA analyses), the frame transmission, and any acknowledgment can be completed before the end of the CAP. If the MAC sublayer can proceed, it shall request that the PHY perform the CCA in the current superframe. If the MAC sublayer cannot proceed, it shall wait until the start of the CAP in the next superframe and apply a further random backoff delay [step (2)] before evaluating whether it can proceed again.

In a slotted CSMA-CA system with the BLE subfield set to one, the MAC sublayer shall ensure that, after the random backoff, the remaining CSMA-CA operations can be undertaken and the entire transaction can be transmitted before the end of the CAP. The backoff countdown shall only occur during the first *macBattLifeExtPeriods* full backoff periods after the end of the IFS period following the beacon. The MAC sublayer shall proceed if the remaining CSMA-CA algorithm steps (two CCA analyses), the frame transmission, and any acknowledgment can be completed before the end of the CAP, and the frame transmission will start in one of the first *macBattLifeExtPeriods* full backoff periods after the IFS period following the beacon. If the MAC sublayer can proceed, it shall request that the PHY perform the CCA in the current superframe. If the MAC sublayer cannot proceed, it shall wait until the start of the CAP in the next superframe and apply a further random backoff delay [step (2)] before evaluating whether it can proceed again.

If the channel is assessed to be busy [step (4)], the MAC sublayer shall increment both *NB* and *BE* by one, ensuring that *BE* shall be no more than *macMaxBE*. The MAC sublayer in a slotted CSMA-CA system shall also reset *CW* to two. If the value of *NB* is less than or equal to *macMaxCSMABackoffs*, the CSMA-CA algorithm shall return to step (2). If the value of *NB* is greater than *macMaxCSMABackoffs*, the CSMA-CA algorithm shall terminate with a channel access failure status.

If the channel is assessed to be idle [step (5)], the MAC sublayer in a slotted CSMA-CA system shall ensure that the contention window has expired before commencing transmission. To do this, the MAC sublayer shall first decrement *CW* by one and then determine whether it is equal to zero. If it is not equal to zero, the CSMA-CA algorithm shall return to step (3). If it is equal to zero, the MAC sublayer shall begin transmission of the frame on the boundary of the next backoff period. If the channel is assessed to be idle in an unslotted CSMA-CA system, the MAC sublayer shall begin transmission of the frame immediately.

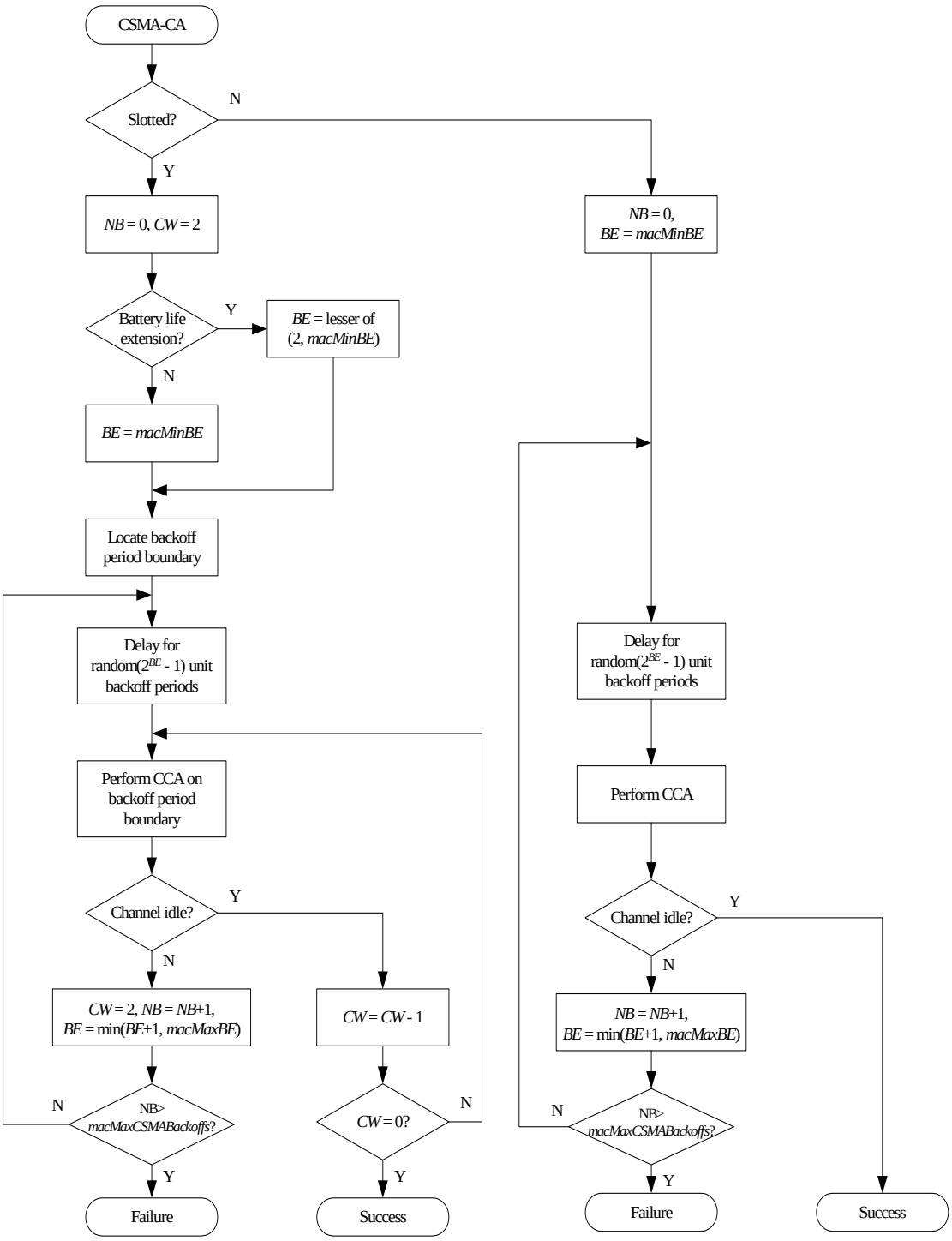


Figure 70—The CSMA-CA algorithm

7.5.2 Starting and maintaining PANs

This subclause specifies the procedures for scanning through channels, identifying PAN identifier conflicts, and starting PANs.

7.5.2.1 Scanning through channels

All devices shall be capable of performing passive and orphan scans across a specified list of channels. In addition, an FFD shall be able to perform ED and active scans. The next higher layer should submit a scan request for a particular channel page containing a list of channels chosen only from the channels specified by *phyChannelsSupported* for that particular channel page.

A device is instructed to begin a channel scan through the MLME-SCAN.request primitive. Channels are scanned in order from the lowest channel number to the highest. For the duration of the scan, the device shall suspend beacon transmissions, if applicable, and shall only accept frames received over the PHY data service that are relevant to the scan being performed. Upon the conclusion of the scan, the device shall recommence beacon transmissions. The results of the scan shall be returned via the MLME-SCAN.confirm primitive.

7.5.2.1.1 ED channel scan

An ED scan allows a device to obtain a measure of the peak energy in each requested channel. This could be used by a prospective PAN coordinator to select a channel on which to operate prior to starting a new PAN. During an ED scan, the MAC sublayer shall discard all frames received over the PHY data service.

An ED scan over a specified set of logical channels is requested using the MLME-SCAN.request primitive with the ScanType parameter set to indicate an ED scan. For each logical channel, the MLME shall first switch to the channel, by setting *phyCurrentChannel* and *phyCurrentPage* accordingly, and then repeatedly perform an ED measurement for [*aBaseSuperframeDuration* * (2^n + 1)] symbols, where *n* is the value of the ScanDuration parameter in the MLME-SCAN.request primitive. An ED measurement is performed by the MLME issuing the PLME-ED.request (see 6.2.2.3), which is guaranteed to return a value. The maximum ED measurement obtained during this period shall be noted before moving on to the next channel in the channel list. A device shall be able to store between one and an implementation-specified maximum number of channel ED measurements.

The ED scan shall terminate when either the number of channel ED measurements stored equals the implementation-specified maximum or energy has been measured on each of the specified logical channels.

7.5.2.1.2 Active channel scan

An active scan allows a device to locate any coordinator transmitting beacon frames within its POS. This could be used by a prospective PAN coordinator to select a PAN identifier prior to starting a new PAN, or it could be used by a device prior to association.

During an active scan, the MAC sublayer shall discard all frames received over the PHY data service that are not beacon frames. If a beacon frame is received that contains the address of the scanning device in its list of pending addresses, the scanning device shall not attempt to extract the pending data.

Before commencing an active scan, the MAC sublayer shall store the value of *macPANId* and then set it to 0xffff for the duration of the scan. This enables the receive filter to accept all beacons rather than just the beacons from its current PAN (see 7.5.6.2). On completion of the scan, the MAC sublayer shall restore the value of *macPANId* to the value stored before the scan began.

An active scan over a specified set of logical channels is requested using the MLME-SCAN.request primitive with the ScanType parameter set to indicate an active scan. For each logical channel, the device shall first switch to the channel, by setting *phyCurrentChannel* and *phyCurrentPage* accordingly, and send a beacon request command (see 7.3.7). Upon successful transmission of the beacon request command, the device shall enable its receiver for at most [*aBaseSuperframeDuration* * (2^n + 1)] symbols, where *n* is the value of the ScanDuration parameter. During this time, the device shall reject all nonbeacon frames and record the information contained in all unique beacons in a PAN descriptor structure (see Table 55 in 7.1.5.1.1). If a beacon frame is received when *macAutoRequest* is set to TRUE, the list of PAN descriptor structures shall be stored by the MAC sublayer until the scan is complete; at this time, the list shall be sent to the next higher layer in the PANDescriptorList parameter of the MLME-SCAN.confirm primitive. A device shall be able to store between one and an implementation-specified maximum number of PAN descriptors. A beacon frame shall be assumed to be unique if it contains both a PAN identifier and a source address that has not been seen before during the scan of the current channel. If a beacon frame is received when *macAutoRequest* is set to FALSE, each recorded PAN descriptor is sent to the next higher layer in a separate MLME-BEACON-NOTIFY.indication primitive. A received beacon frame containing one or more octets of payload shall also cause the PAN descriptor to be sent to the next higher layer via the MLME-BEACON-NOTIFY.indication primitive.

If a protected beacon frame is received, i.e., the security enabled subfield in the frame control field is set to one, the device shall attempt to unsecure the beacon frame using the unsecuring process described in 7.5.8.2.3.

The security-related elements of the PAN descriptor corresponding to the beacon (see Table 55) shall be set to the corresponding parameters returned by the unsecuring process. The SecurityFailure element of the PAN descriptor shall be set to SUCCESS if the status from the unsecuring process is SUCCESS and set to one of the other status codes indicating an error in the security processing otherwise.

The information from the unsecured frame shall be recorded in the PAN descriptor even if the status from the unsecuring process indicated an error.

If a coordinator of a beacon-enabled PAN receives the beacon request command, it shall ignore the command and continue transmitting its periodic beacons as usual. If a coordinator of a nonbeacon-enabled PAN receives this command, it shall transmit a single beacon frame using unslotted CSMA-CA.

If *macAutoRequest* is set to TRUE, the active scan on a particular channel shall terminate when the number of beacons found equals the implementation-specified limit or the channel has been scanned for the full time, as specified in 7.5.2.1.2. If *macAutoRequest* is set to FALSE, the active scan on a particular channel shall terminate when the channel has been scanned for the full time. If a channel was not scanned for the full time, it shall be considered to be unscanned.

If *macAutoRequest* is set to TRUE, the entire scan procedure shall terminate when the number of PAN descriptors stored equals the implementation-specified maximum or every channel in the set of available channels has been scanned. If *macAutoRequest* is set to FALSE, the entire scan procedure shall only terminate when every channel in the set of available channels has been scanned.

7.5.2.1.3 Passive channel scan

A passive scan, like an active scan, allows a device to locate any coordinator transmitting beacon frames within its POS. The beacon request command, however, is not transmitted. This type of scan could be used by a device prior to association.

During a passive scan, the MAC sublayer shall discard all frames received over the PHY data service that are not beacon frames. If a beacon frame is received that contains the address of the scanning device in its list of pending addresses, the scanning device shall not attempt to extract the pending data.

Before commencing a passive scan, the MAC sublayer shall store the value of *macPANId* and then set it to 0xffff for the duration of the scan. This enables the receive filter to accept all beacons rather than just the beacons from its current PAN (see 7.5.6.2). On completion of the scan, the MAC sublayer shall restore the value of *macPANId* to the value stored before the scan began.

A passive scan over a specified set of logical channels is requested using the MLME-SCAN.request primitive with the ScanType parameter set to indicate a passive scan. For each logical channel, the device shall first switch to the channel, by setting *phyCurrentChannel* and *phyCurrentPage* accordingly, and then enable its receiver for at most [*aBaseSuperframeDuration* * (2^n + 1)] symbols, where *n* is the value of the ScanDuration parameter. During this time, the device shall reject all nonbeacon frames and record the information contained in all unique beacons in a PAN descriptor structure (see Table 55 in 7.1.5.1.1). If a beacon frame is received when *macAutoRequest* is set to TRUE, the list of PAN descriptor structures shall be stored by the MAC sublayer until the scan is complete; at this time, the list shall be sent to the next higher layer in the PANDescriptorList parameter of the MLME-SCAN.confirm primitive. A device shall be able to store between one and an implementation- specified maximum number of PAN descriptors. A beacon frame shall be assumed to be unique if it contains both a PAN identifier and a source address that has not been seen before during the scan of the current channel. If a beacon frame is received when *macAutoRequest* is set to FALSE, each recorded PAN descriptor is sent to the next higher layer in a separate MLME-BEACON-NOTIFY.indication primitive. Once the scan is complete, the MLME-SCAN.confirm shall be issued to the next higher layer with a null PANDescriptorList. A received beacon frame containing one or more octets of payload shall also cause the PAN descriptor to be sent to the next higher layer via the MLME-BEACON-NOTIFY.indication primitive.

If a protected beacon frame is received (i.e., the security enabled subfield in the frame control field is set to one), the device shall attempt to unsecure the beacon frame using the unsecuring process described in 7.5.8.2.3.

The security-related elements of the PAN descriptor corresponding to the beacon (see Table 55) shall be set to the corresponding parameters returned by the unsecuring process. The SecurityFailure element of the PAN descriptor shall be set to SUCCESS if the status from the unsecuring process is SUCCESS and set to one of the other status codes indicating an error in the security processing otherwise.

The information from the unsecured frame shall be recorded in the PAN descriptor even if the status from the unsecuring process indicated an error.

If *macAutoRequest* is set to TRUE, the passive scan on a particular channel shall terminate when the number of beacons found equals the implementation specified limit or the channel has been scanned for the full time, as specified in 7.5.2.1.3. If *macAutoRequest* is set to FALSE, the passive scan on a particular channel shall terminate when the channel has been scanned for the full time. If a channel was not scanned for the full time, it shall be considered to be unscanned.

If *macAutoRequest* is set to TRUE, the entire scan procedure shall terminate when the number of PAN descriptors stored equals the implementation-specified maximum or every channel in the set of available channels has been scanned. If *macAutoRequest* is set to FALSE, the entire scan procedure shall only terminate when every channel in the set of available channels has been scanned.

7.5.2.1.4 Orphan channel scan

An orphan scan allows a device to attempt to relocate its coordinator following a loss of synchronization. During an orphan scan, the MAC sublayer shall discard all frames received over the PHY data service that are not coordinator realignment command frames.

An orphan scan over a specified set of logical channels is requested using the MLME-SCAN.request primitive with the ScanType parameter set to indicate an orphan scan. For each logical channel, the device shall

first switch to the channel, by setting *phyCurrentChannel* and *phyCurrentPage* accordingly, and then send an orphan notification command (see 7.3.6). Upon successful transmission of the orphan notification command, the device shall enable its receiver for at most *macResponseWaitTime* symbols. If the device successfully receives a coordinator realignment command (see 7.3.8) within this time, the device shall terminate the scan.

If a coordinator receives the orphan notification command, the MLME shall send the MLME-ORPHAN.indication primitive to the next higher layer. The next higher layer should then search its device list for the device indicated by the primitive. If the next higher layer finds a record of the device, it should send a coordinator realignment command to the orphaned device using the MLME-ORPHAN.response primitive. The process of searching for the device and sending the coordinator realignment command shall occur within *macResponseWaitTime* symbols. The coordinator realignment command shall contain its current PAN identifier, *macPANId*, its current logical channel and channel page, and the 16-bit short address of the orphaned device. If the next higher layer of the coordinator finds no record of the device, it should ignore the MLME-ORPHAN.indication primitive and not send the MLME-ORPHAN.response primitive; therefore, the MLME shall not send a coordinator realignment command.

The orphan scan shall terminate when the device receives a coordinator realignment command or the specified set of logical channels has been scanned.

7.5.2.2 PAN identifier conflict resolution

In some instances a situation could occur in which two PANs exist in the same POS with the same PAN identifier. If this conflict happens, the PAN coordinator and its devices shall perform the PAN identifier conflict resolution procedure.

This procedure is optional for an RFD.

7.5.2.2.1 Detection

The PAN coordinator shall conclude that a PAN identifier conflict is present if either of the following apply:

— A beacon frame is received by the PAN coordinator with the PAN coordinator subfield (see 7.2.2.1.2) set to one and the PAN identifier equal to *macPANId*.
— A PAN ID conflict notification command (see 7.3.5) is received by the PAN coordinator from a device associated with it on its PAN.

A device that is associated through the PAN coordinator (i.e., *macAssociatedPANCoord* is set to TRUE) shall conclude that a PAN identifier conflict is present if the following applies:

— A beacon frame is received by the device with the PAN coordinator subfield set to one, the PAN identifier equal to *macPANId*, and an address that is equal to neither *macCoordShortAddress* nor *macCoordExtendedAddress*.

A device, which is associated through a coordinator that is not the PAN coordinator, shall not be capable of detecting a PAN identifier conflict.

7.5.2.2.2 Resolution

On the detection of a PAN identifier conflict by a device, it shall generate the PAN ID conflict notification command (see 7.3.5) and send it to its PAN coordinator. Since the PAN ID conflict notification command contains an acknowledgment request (see 7.3.3.1), the PAN coordinator shall confirm its receipt by sending an acknowledgment frame. Once the device has received the acknowledgment frame from the PAN coordinator, the MLME shall issue an MLME-SYNC-LOSS.indication primitive with the LossReason parameter

set to PAN_ID_CONFLICT. If the device does not receive an acknowledgment frame, the MLME shall not inform the next higher layer of the PAN identifier conflict.

On the detection of a PAN identifier conflict by the PAN coordinator, the MLME shall issue an MLME-SYNC-LOSS.indication to the next higher layer with the LossReason parameter set to PAN_ID_CONFLICT. The next higher layer of the PAN coordinator may then perform an active scan and, using the information from the scan, select a new PAN identifier. The algorithm for selecting a suitable PAN identifier is out of the scope of this standard. If the next higher layer does select a new PAN identifier, it may then issue an MLME-START.request with the CoordRealignment parameter set to TRUE in order to realign the PAN, as described in 7.5.2.3.

7.5.2.3 Starting and realigning a PAN

This subclause specifies procedures for the PAN coordinator starting a PAN, coordinators realigning a PAN and devices being realigned on a PAN.

7.5.2.3.1 Starting a PAN

A PAN should be started by an FFD only after having first performed a MAC sublayer reset, by issuing the MLME-RESET.request primitive with the SetDefaultPIB parameter set to TRUE, an active channel scan and a suitable PAN identifier selection. The algorithm for selecting a suitable PAN identifier from the list of PAN descriptors returned from the active channel scan procedure is out of the scope of this standard. In addition, an FFD should set *macShortAddress* to a value less than 0xffff.

An FFD is instructed to begin operating a PAN through the use of the MLME-START.request primitive (see 7.1.14.1) with the PANCoordinator parameter set to TRUE and the CoordRealigment parameter set to FALSE. On receipt of this primitive, the MAC sublayer shall update the superframe configuration and channel parameters as specified in 7.5.2.3.4. After completing this, the MAC sublayer shall issue the MLME-START.confirm primitive with a status of SUCCESS and begin operating as the PAN coordinator.

7.5.2.3.2 Realigning a PAN

If a coordinator receives the MLME-START.request primitive (see 7.1.14.1) with the CoordRealignment parameter set to TRUE, the coordinator shall attempt to transmit a coordinator realignment command containing the new parameters for PANId, LogicalChannel and, if present, ChannelPage.

When the coordinator is already transmitting beacons and the CoordRealignment parameter is set to TRUE, the next scheduled beacon shall be transmitted on the current channel using the current superframe configuration, with the frame pending subfield of the frame control field set to one. Immediately following the transmission of the beacon, the coordinator realignment command shall also be transmitted on the current channel using CSMA-CA.

When the coordinator is not already transmitting beacons and the CoordRealignment parameter is set to TRUE, the coordinator realignment command shall be transmitted immediately on the current channel using CSMA-CA.

If the transmission of the coordinator realignment command fails due to a channel access failure, the MLME shall notify the next higher layer by issuing the MLME-START.confirm primitive with a status of CHANNEL_ACCESS_FAILURE. The next higher layer may then choose to issue the MLME-START.request primitive again.

Upon successful transmission of the coordinator realignment command, the new superframe configuration and channel parameters shall be put into operation as described in 7.5.2.3.4 at the subsequent scheduled bea-

con, or immediately if the coordinator is not already transmitting beacons, and the MAC sublayer shall issue the MLME-START.confirm primitive with a status of SUCCESS.

7.5.2.3.3 Realignment in a PAN

If a device has received the coordinator realignment command (see 7.3.8) from the coordinator through which it associated and the MLME was not carrying out an orphan scan, the MLME shall issue the MLME-SYNC-LOSS.indication primitive with the LossReason parameter set to REALIGNMENT and the PANId, LogicalChannel, ChannelPage and the security-related parameters set to the respective fields in the coordinator realignment command. The next higher layer of a coordinator may then issue an MLME-START.request primitive with the CoordRealignment parameter set to TRUE. The next higher layer of a device that is not a coordinator may instead change the superframe configuration or channel parameters through use of the MLME-SET.request primitive.

7.5.2.3.4 Updating superframe configuration and channel PIB attributes

To update the superframe configuration and channel attributes, the MLME shall assign values from the MLME-START.request primitive parameters to the appropriate PIB attributes. The MLME shall set *macBeaconOrder* to the value of the BeaconOrder parameter. If *macBeaconOrder* is equal to 15, the MLME will also set *macSuperframeOrder* to 15. In this case, this primitive configures a nonbeacon-enabled PAN. If *macBeaconOrder* is less than 15, the MAC sublayer will set *macSuperframeOrder* to the value of the SuperframeOrder parameter. The MAC sublayer shall also update *macPANID* with the value of the PANId parameter and update *phyCurrentPage* and *phyCurrentChannel* with the values of the ChannelPage and LogicalChannel parameters, respectively, by twice issuing the PLME-SET.request primitive.

7.5.2.4 Beacon generation

A device shall be permitted to transmit beacon frames only if *macShortAddress* is not equal to 0xffff.

An FFD shall use the MLME-START.request primitive to begin transmitting beacons only if the BeaconOrder parameter is less than 15. The FFD may begin beacon transmission either as the PAN coordinator of a new PAN or as a device on a previously established PAN, depending upon the setting of the PANCoordinator parameter (see 7.1.14.1). The FFD shall begin beacon transmission on a previously established PAN only once it has successfully associated with that PAN.

If the FFD is the PAN coordinator (i.e., the PANCoordinator parameter is set to TRUE), the MAC sublayer shall ignore the StartTime parameter and begin beacon transmissions immediately. Setting the StartTime parameter to zero shall also cause the MAC sublayer to begin beacon transmissions immediately. If the FFD is not the PAN coordinator and the StartTime parameter is nonzero, the time to begin beacon transmissions shall be calculated using the following method. The StartTime parameter, which is rounded to a backoff slot boundary, shall be added to the time, obtained from the local clock, when the MAC sublayer receives the beacon of the coordinator through which it is associated. The MAC sublayer shall then begin beacon transmissions when the current time, obtained from the local clock, equals the number of calculated symbols. In order for the beacon transmission time to be calculated by the MAC sublayer, the MAC sublayer shall first track the beacon of the coordinator through which it is associated. If the MLME-START.request primitive is issued with a nonzero StartTime parameter and the MAC sublayer is not currently tracking the beacon of its coordinator, the MLME shall not begin beacon transmissions but shall instead issue the MLME-START.confirm primitive with a status of TRACKING_OFF.

If a device misses between one and (*aMaxLostBeacons*-1) consecutive beacon frames from its coordinator, the device shall continue to transmit its own beacons based on both *macBeaconOrder* (see 7.5.2.3.4) and its local clock. If the device then receives a beacon frame from its coordinator and therefore does not lose synchronization, the device shall resume transmitting its own beacons based on the StartTime parameter and the incoming beacon. If a device does lose synchronization with its coordinator, the MLME of the device shall

issue the MLME-SYNC-LOSS.indication primitive to the next higher layer and immediately stop transmitting its own beacons. The next higher layer may, at any time following the reception of the MLME-SYNC-LOSS.indication primitive, resume beacon transmissions by issuing a new MLME-START.request primitive.

On receipt of the MLME-START.request primitive, the MAC sublayer shall set the PAN identifier in *macPANId* and use this value in the source PAN identifier field of the beacon frame. The address used in the source address field of the beacon frame shall contain the value of *aExtendedAddress* if *macShortAddress* is equal to 0xfffe or *macShortAddress* otherwise.

The time of transmission of the most recent beacon shall be recorded in *macBeaconTxTime* and shall be computed so that its value is taken at the same symbol boundary in each beacon frame, the location of which is implementation specific. The symbol boundary, which is specified by the *macSyncSymbolOffset* attribute, is the same as that used in the timestamp of the incoming beacon frame, as described in 7.5.4.1.

All beacon frames shall be transmitted at the beginning of each superframe at an interval equal to *aBaseSuperframeDuration* * 2^n symbols, where *n* is the value of *macBeaconOrder* (the construction of the beacon frame is specified in 7.2.2.1).

Beacon transmissions shall be given priority over all other transmit and receive operations.

7.5.2.5 Device discovery

The PAN coordinator or a coordinator indicates its presence on a PAN to other devices by transmitting beacon frames. This allows the other devices to perform device discovery.

A coordinator that is not the PAN coordinator shall begin transmitting beacon frames only when it has successfully associated with a PAN. The transmission of beacon frames by the device is initiated through the use of the MLME-START.request primitive with the PANCoordinator parameter set to FALSE. On receipt of this primitive, the MLME shall begin transmitting beacons based on the StartTime parameter (see 7.5.2.4) using the identifier of the PAN with which the device has associated, *macPANId*, and its extended address, *aExtendedAddress*, if *macShortAddress* is equal to 0xfffe, or its short address, *macShortAddress*, otherwise. A beacon frame shall be transmitted at a rate of one beacon frame every *aBaseSuperframeDuration* * 2^n symbols, where *n* is the value of *macBeaconOrder*.

7.5.3 Association and disassociation

This subclause specifies the procedures for association and disassociation.

7.5.3.1 Association

A device shall attempt to associate only after having first performed a MAC sublayer reset, by issuing the MLME-RESET.request primitive with the SetDefaultPIB parameter set to TRUE, and then having completed either an active channel scan (see 7.5.2.1.2) or a passive channel scan (see 7.5.2.1.3). The results of the channel scan would have then been used to choose a suitable PAN. The algorithm for selecting a suitable PAN with which to associate from the list of PAN descriptors returned from the channel scan procedure is out of the scope of this standard.

Following the selection of a PAN with which to associate, the next higher layers shall request through the MLME-ASSOCIATE.request primitive that the MLME configures the following PHY and MAC PIB attributes to the values necessary for association:

— *phyCurrentChannel* shall be set equal to the LogicalChannel parameter of the MLME-ASSOCIATE.request primitive.

— *phyCurrentPage* shall be set equal to the ChannelPage parameter of the MLME-ASSOCIATE.request primitive.
— *macPANId* shall be set equal to the CoordPANId parameter of the MLME-ASSOCIATE.request primitive.
— *macCoordExtendedAddress* or *macCoordShortAddress*, depending on which is known from the beacon frame from the coordinator through which it wishes to associate, shall be set equal to the CoordAddress parameter of the MLME-ASSOCIATE.request primitive.

A coordinator shall allow association only if *macAssociationPermit* is set to TRUE. Similarly, a device should attempt to associate only with a PAN through a coordinator that is currently allowing association, as indicated in the results of the scanning procedure. If a coordinator with *macAssociationPermit* set to FALSE receives an association request command from a device, the command shall be ignored.

In order to optimize the association procedure on a beacon-enabled PAN, a device may begin tracking the beacon of the coordinator through which it wishes to associate a priori. This is achieved by the next higher layer issuing the MLME-SYNC.request primitive with the TrackBeacon parameter set to TRUE.

A device that is instructed to associate with a PAN, through the MLME-ASSOCIATE.request primitive, shall try to associate only with an existing PAN and shall not attempt to start its own PAN.

The MAC sublayer of an unassociated device shall initiate the association procedure by sending an association request command (see 7.3.1) to the coordinator of an existing PAN; if the association request command cannot be sent due to a channel access failure, the MAC sublayer shall notify the next higher layer. Since the association request command contains an acknowledgment request (see 7.3.3.1), the coordinator shall confirm its receipt by sending an acknowledgment frame.

The acknowledgment to an association request command does not mean that the device has associated. The next higher layer of the coordinator needs time to determine whether the current resources available on the PAN are sufficient to allow another device to associate. The next higher layer should make this decision within *macResponseWaitTime* symbols. If the next higher layer of the coordinator finds that the device was previously associated on its PAN, all previously obtained device-specific information should be removed. If sufficient resources are available, the next higher layer should allocate a 16-bit short address to the device and the MAC sublayer shall generate an association response command (see 7.3.2) containing the new address and a status indicating a successful association. If sufficient resources are not available, the next higher layer of the coordinator should inform the MAC sublayer, and the MLME shall generate an association response command containing a status indicating a failure (see Table 83). The association response command shall be sent to the device requesting association using indirect transmission, i.e., the association response command frame shall be added to the list of pending transactions stored on the coordinator and extracted at the discretion of the device concerned using the method described in 7.5.6.3.

If the allocate address subfield of the capability information field (see 7.3.1.2) of the association request command is set to one, the next higher layer of the coordinator shall allocate a 16-bit address with a range depending on the addressing mode supported by the coordinator, as described in Table 87. If the allocate address subfield of the association request command is set to zero, the 16-bit short address shall be equal to 0xfffe. A short address of 0xfffe is a special case that indicates that the device has associated, but has not been allocated a short address by the coordinator. In this case, the device shall use only its 64-bit extended address to operate on the network.

On receipt of the acknowledgment to the association request command, the device shall wait for at most *macResponseWaitTime* symbols for the coordinator to make its association decision; the PIB attribute *macResponseWaitTime* is a network topology dependent parameter and may be set to match the specific requirements of the network that a device is trying to join. If the device is tracking the beacon, it shall attempt to extract the association response command from the coordinator whenever it is indicated in the beacon frame. If the device is not tracking the beacon, it shall attempt to extract the association response command from

the coordinator after *macResponseWaitTime* symbols. If the device does not extract an association response command frame from the coordinator within *macResponseWaitTime* symbols, the MLME shall issue the MLME-ASSOCIATE.confirm primitive with a status of NO_DATA, and the association attempt shall be deemed a failure. In this case, the next higher layer shall terminate any tracking of the beacon. This is achieved by issuing the MLME-SYNC.request primitive with the TrackBeacon parameter set to FALSE.

Since the association response command contains an acknowledgment request (see 7.3.3.1), the device requesting association shall confirm its receipt by sending an acknowledgment frame. If the association status field of the command indicates that the association was successful, the device shall store the address contained in the 16-bit short address field of the command in *macShortAddress*; communication on the PAN using this short address shall depend on its range, as described in Table 87. If the original beacon selected for association following a scan contained the short address of the coordinator, the extended address of the coordinator, contained in the MHR of the association response command frame, shall be stored in *macCoordExtendedAddress.*

Table 87—Usage of the 16-bit short address

Value of *macShortAddress*	Description
0x0000–0xfffd	If a source address is included, the device shall use short source addressing mode for beacon and data frames and the appropriate source addressing mode specified in 7.3 for MAC command frames.
0xfffe	If a source address is included, the device shall use extended source addressing mode for beacon and data frames and the appropriate source addressing mode specified in 7.3 for MAC command frames.
0xffff	The device is not associated and therefore shall not perform any data frame communication. The device shall use the appropriate source addressing mode specified in 7.3 for MAC command frames.

If the association status field of the command indicates that the association was unsuccessful, the device shall set *macPANId* to the default value (0xffff).

7.5.3.2 Disassociation

The disassociation procedure is initiated by the next higher layer by issuing the MLME-DISASSOCIATE.request primitive to the MLME.

When a coordinator wants one of its associated devices to leave the PAN, the MLME of the coordinator shall send the disassociation notification command in the manner specified by the TxIndirect parameter of the MLME-DISASSOCIATE.request primitive previously sent by the next higher layer. If TxIndirect is TRUE, the MLME of the coordinator shall send the disassociation notification command to the device using indirect transmission, i.e., the disassociation notification command frame shall be added to the list of pending transactions stored on the coordinator and extracted at the discretion of the device concerned using the method described in 7.5.6.3. If the command frame is not successfully extracted by the device, the coordinator should consider the device disassociated. Otherwise, the MLME shall send the disassociation notification command to the device directly. In this case, if the disassociation notification command cannot be sent due to a channel access failure, the MAC sublayer shall notify the next higher layer.

Since the disassociation command contains an acknowledgment request (see 7.3.3.1), the receiving device shall confirm its receipt by sending an acknowledgment frame. If the direct or indirect transmission fails, the coordinator should consider the device disassociated.

If an associated device wants to leave the PAN, the MLME of the device shall send a disassociation notification command to its coordinator. If the disassociation notification command cannot be sent due to a channel access failure, the MAC sublayer shall notify the next higher layer. Since the disassociation command contains an acknowledgment request (see 7.3.3.1), the coordinator shall confirm its receipt by sending an acknowledgment frame. However, even if the acknowledgment is not received, the device should consider itself disassociated.

If the source address contained in the disassociation notification command is equal to *macCoordExtendedAddress*, the device should consider itself disassociated. If the command is received by a coordinator and the source is not equal to *macCoordExtendedAddress*, it shall verify that the source address corresponds to one of its associated devices; if so, the coordinator should consider the device disassociated. If none of the above conditions is satisfied, the command shall be ignored.

An associated device shall disassociate itself by removing all references to the PAN; the MLME shall set *macPANId*, *macShortAddress*, *macAssociatedPANCoord*, *macCoordShortAddress* and *macCoordExtendedAddress* to the default values. The next higher layer of a coordinator should disassociate a device by removing all references to that device.

The next higher layer of the requesting device shall be notified of the result of the disassociation procedure through the MLME-DISASSOCIATE.confirm primitive.

7.5.4 Synchronization

This subclause specifies the procedures for coordinators to generate beacon frames and for devices to synchronize with a coordinator. For PANs supporting beacons, synchronization is performed by receiving and decoding the beacon frames. For PANs not supporting beacons, synchronization is performed by polling the coordinator for data.

7.5.4.1 Synchronization with beacons

All devices operating on a beacon-enabled PAN (i.e., *macBeaconOrder* < 15) shall be able to acquire beacon sychronization in order to detect any pending messages or to track the beacon. Devices shall be permitted to acquire beacon synchronization only with beacons containing the PAN identifier specified in *macPANId*. If *macPANId* specifies the broadcast PAN identifier (0xffff), a device shall not attempt to acquire beacon synchronization.

A device is instructed to attempt to acquire the beacon through the MLME-SYNC.request primitive. If tracking is specified in the MLME-SYNC.request primitive, the device shall attempt to acquire the beacon and keep track of it by regular and timely activation of its receiver. If tracking is not specified, the device shall either attempt to acquire the beacon only once or terminate the tracking after the next beacon if tracking was enabled through a previous request.

To acquire beacon synchronization, a device shall enable its receiver and search for at most [*aBaseSuperframeDuration* * (2^n + 1)] symbols, where *n* is the value of *macBeaconOrder*. If a beacon frame containing the current PAN identifier of the device is not received, the MLME shall repeat this search. Once the number of missed beacons reaches *aMaxLostBeacons*, the MLME shall notify the next higher layer by issuing the MLME-SYNC-LOSS.indication primitive with a loss reason of BEACON_LOSS.

The MLME shall timestamp each received beacon frame at the same symbol boundary within each frame, the location of which described by the *macSyncSymbolOffset* attribute. The symbol boundary shall be the same as that used in the timestamp of the outgoing beacon frame, stored in *macBeaconTxTime*. The timestamp value shall be that of the local clock of the device at the time of the symbol boundary. The timestamp is intended to be a relative time measurement that may or may not be made absolute, at the discretion of the implementer.

If a protected beacon frame is received (i.e., the security enabled subfield in the frame control field is set to one), the device shall attempt to unsecure the beacon frame using the unsecuring process described in 7.5.8.2.3.

If the status from the unsecuring process is not SUCCESS, the MLME shall issue an MLME-COMM-STATUS.indication primitive with the status parameter set to the status from the unsecuring process, indicating the error.

The security-related elements of the PAN descriptor corresponding to the beacon (see Table 55) shall be set to the corresponding parameters returned by the unsecuring process. The SecurityFailure element of the PAN descriptor shall be set to SUCCESS if the status from the unsecuring process is SUCCESS and set to one of the other status codes indicating an error in the security processing otherwise.

If a beacon frame is received, the MLME shall discard the beacon frame if the source address and the source PAN identifier fields of the MHR of the beacon frame do not match the coordinator source address (*macCoordShortAddress* or *macCoordExtendedAddress*, depending on the addressing mode) and the PAN identifier of the device (*macPANId*).

If a valid beacon frame is received and *macAutoRequest* is set to FALSE, the MLME shall indicate the beacon parameters to the next higher layer by issuing the MLME-BEACON-NOTIFY.indication primitive. If a beacon frame is received and *macAutoRequest* is set to TRUE, the MLME shall first issue the MLME-BEACON-NOTIFY.indication primitive if the beacon contains any payload. The MLME shall then compare its address with those addresses in the address list field of the beacon frame. If the address list field contains the 16-bit short or 64-bit extended address of the device and the source PAN identifier matches *macPANId*, the MLME shall follow the procedure for extracting pending data from the coordinator (see 7.5.6.3).

If beacon tracking is activated, the MLME shall enable its receiver at a time prior to the next expected beacon frame transmission, i.e., just before the known start of the next superframe. If the number of consecutive beacons missed by the MLME reaches *aMaxLostBeacons*, the MLME shall respond with the MLME-SYNC-LOSS.indication primitive with a loss reason of BEACON_LOST.

7.5.4.2 Synchronization without beacons

All devices operating on a nonbeacon-enabled PAN (*macBeaconOrder* = 15) shall be able to poll the coordinator for data at the discretion of the next higher layer.

A device is instructed to poll the coordinator when the MLME receives the MLME-POLL.request primitive. On receipt of this primitive, the MLME shall follow the procedure for extracting pending data from the coordinator (see 7.5.6.3).

7.5.4.3 Orphaned device realignment

If the next higher layer receives repeated communications failures following its requests to transmit data, it may conclude that it has been orphaned. A single communications failure occurs when a device transaction fails to reach the coordinator, i.e., an acknowledgment is not received after *macMaxFrameRetries* attempts at sending the data. If the next higher layer concludes that it has been orphaned, it may instruct the MLME to either perform the orphaned device realignment procedure, or to reset the MAC sublayer and then perform the association procedure.

If the decision has been made by the next higher layer to perform the orphaned device realignment procedure, it will have issued an MLME-SCAN.request with the ScanType parameter set to orphan scan and the ScanChannel parameter containing the list of channels to be scanned. Upon receiving this primitive, the MAC sublayer shall begin an orphan scan, as described in 7.5.2.1.4.

If the orphan scan is successful (i.e., its PAN has been located), the device shall update its MAC PIB with the PAN information contained in the coordinator realignment command (see 7.3.8).

7.5.5 Transaction handling

Because IEEE P802.15.4REVb/D6 favors very low cost devices that, in general, will be battery powered, transactions can be instigated from the devices themselves rather than from the coordinator. In other words, either the coordinator needs to indicate in its beacon when messages are pending for devices or the devices themselves need to poll the coordinator to determine whether they have any messages pending. Such transfers are called indirect transmissions.

The coordinator shall begin handling a transaction on receipt of an indirect transmission request either via the MCPS-DATA.request primitive or via a request from the MLME to send a MAC command instigated by a primitive from the next higher layer, such as the MLME-ASSOCIATE.response primitive (see 7.1.3.3). On completion of the transaction, the MAC sublayer shall indicate a status value to the next higher layer. If a request primitive instigated the indirect transmission, the corresponding confirm primitive shall be used to convey the appropriate status value. Conversely, if a response primitive instigated the indirect transmission, the MLME-COMM-STATUS.indication primitive shall be used to convey the appropriate status value. The MLME-COMM-STATUS.indication primitive can be related to its corresponding response primitive by examining the destination address field.

The information contained in the indirect transmission request forms a transaction, and the coordinator shall be capable of storing at least one transaction. On receipt of an indirect transmission request, if there is no capacity to store another transaction, the MAC sublayer shall indicate to the next higher layer a status of TRANSACTION_OVERFLOW in the appropriate corresponding primitive.

If the coordinator is capable of storing more than one transaction, it shall ensure that all the transactions for the same device are sent in the order in which they arrived at the MAC sublayer. For each transaction sent, if another exists for the same device, the MAC sublayer shall set its frame pending subfield to one, indicating the additional pending data.

Each transaction shall persist in the coordinator for at most *macTransactionPersistenceTime*. If the transaction is not successfully extracted by the appropriate device within this time, the transaction information shall be discarded and the MAC sublayer shall indicate to the next higher layer a status of TRANSACTION_EXPIRED in the appropriate corresponding primitive. In order to be successfully extracted, an acknowledgment shall be received if one was requested.

If the transaction was successful, the transaction information shall be discarded and the MAC sublayer shall indicate to the next higher layer a status of SUCCESS in the appropriate corresponding primitive.

If the coordinator transmits beacons, it shall list the addresses of the devices to which each transaction is associated in the address list field and indicate the number of addresses in the pending address specification field of the beacon frame. If the coordinator is able to store more than seven pending transactions, it shall indicate them in its beacon on a first-come-first-served basis, ensuring that the beacon frame contains at most seven addresses. For transactions requiring a GTS, the PAN coordinator shall not add the address of the recipient to its list of pending addresses in the beacon frame. Instead it shall transmit the transaction in the GTS allocated for the device (see 7.5.7.3).

On a beacon-enabled PAN, if there is a transaction pending for the broadcast address, the frame pending subfield of the frame control field in the beacon frame shall be set to one, and the pending message shall be transmitted immediately following the beacon using the CSMA-CA algorithm. If there is a second message pending for the broadcast address, its transmission shall be delayed until the following superframe. Only one broadcast message shall be allowed to be sent indirectly per superframe.

On a beacon-enabled PAN, a device that receives a beacon containing its address in the list of pending addresses shall attempt to extract the data from the coordinator. On a nonbeacon-enabled PAN, a device shall attempt to extract the data from the coordinator on receipt of the MLME-POLL.request primitive. The procedure for extracting pending data from the coordinator is described in 7.5.6.3. If a device receives a beacon with the frame pending subfield set to one, it shall leave its receiver enabled for up to *macMaxFrameTotalWaitTime* symbols to receive the broadcast data frame from the coordinator.

7.5.6 Transmission, reception, and acknowledgment

This subclause describes the fundamental procedures for transmission, reception, and acknowledgment.

7.5.6.1 Transmission

Each device shall store its current DSN value in the MAC PIB attribute *macDSN* and initialize it to a random value; the algorithm for choosing a random number is out of the scope of this standard. Each time a data or a MAC command frame is generated, the MAC sublayer shall copy the value of *macDSN* into the sequence number field of the MHR of the outgoing frame and then increment it by one. Each device shall generate exactly one DSN regardless of the number of unique devices with which it wishes to communicate. *macDSN* shall be permitted to roll over.

Each coordinator shall store its current BSN value in the MAC PIB attribute *macBSN* and initialize it to a random value; the algorithm for choosing a random number is out of the scope of this standard. Each time a beacon frame is generated, the MAC sublayer shall copy the value of *macBSN* into the sequence number field of the MHR of the outgoing frame and then increment it by one. *macBSN* shall be permitted to roll over.

It should be noted that both the DSN and BSN are 8-bit values and therefore have limited use to the next higher layer (e.g., in the case of the DSN, in detecting retransmitted frames).

The source address field, if present, shall contain the address of the device sending the frame. When a device has associated and has been allocated a 16-bit short address (i.e., *macShortAddress* is not equal to 0xfffe or 0xffff), it shall use that address in preference to its 64-bit extended address (i.e., *aExtendedAddress*) wherever possible. When a device has not yet associated to a PAN or *macShortAddress* is equal to 0xffff, it shall use its 64-bit extended address in all communications requiring the source address field. If the source address field is not present, the originator of the frame shall be assumed to be the PAN coordinator, and the destination address field shall contain the address of the recipient.

The destination address field, if present, shall contain the address of the intended recipient of the frame, which may be either a 16-bit short address or a 64-bit extended address. If the destination address field is not present, the recipient of the frame shall be assumed to be the PAN coordinator, and the source address field shall contain the address of the originator.

If both destination and source addressing information is present, the MAC sublayer shall compare the destination and source PAN identifiers. If the PAN identifiers are identical, the PAN ID compression subfield of the frame control field shall be set to one, and the source PAN identifier shall be omitted from the transmitted frame. If the PAN identifiers are different, the PAN ID compression subfield of the frame control field shall be set to zero, and both destination and source PAN identifier fields shall be included in the transmitted frame. If only either the destination or the source addressing information is present, the PAN ID compression subfield of the frame control field shall be set to zero, and the PAN identifier field of the single address shall be included in the transmitted frame.

If the frame is to be transmitted on a beacon-enabled PAN, the transmitting device shall attempt to find the beacon before transmitting. If the beacon is not being tracked (see 7.5.4.1) and hence the device does not know where the beacon will appear, it shall enable its receiver and search for at most [*aBaseSuperframe-*

Duration * (2^n + 1)] symbols, where *n* is the value of *macBeaconOrder*, in order to find the beacon. If the beacon is not found after this time, the device shall transmit the frame following the successful application of the unslotted version of the CSMA-CA algorithm (see 7.5.1.4). Once the beacon has been found, either after a search or due to its being tracked, the frame shall be transmitted in the appropriate portion of the superframe. Transmissions in the CAP shall follow a successful application of the slotted version of the CSMA-CA algorithm (see 7.5.1.4), and transmissions in a GTS shall not use CSMA-CA.

If the frame is to be transmitted on a nonbeacon-enabled PAN, the frame shall be transmitted following the successful application of the unslotted version of the CSMA-CA algorithm (see 7.5.1.4).

For either a beacon-enabled PAN or a nonbeacon-enabled PAN, if the transmission is direct and originates due to a primitive issued by the next higher layer and the CSMA-CA algorithm fails, the next higher layer shall be notified. If the transmission is indirect and the CSMA-CA algorithm fails, the frame shall remain in the transaction queue until it is requested again and successfully transmitted or until the transaction expires.

The device shall process the frame using the outgoing frame security procedure described in 7.5.8.2.1.

If the status from the outgoing frame security procedure is not SUCCESS, the MLME shall issue the corresponding .confirm or MLME-COMM-STATUS.indication primitive with the status parameter set to the status from the outgoing frame security procedure, indicating the error, and, where relevant, with the security-related parameters set to the corresponding parameters used in the outgoing frame security procedure.

To transmit the frame, the MAC sublayer shall first enable the transmitter by issuing the PLME-SET-TRX-STATE.request primitive with a state of TX_ON to the PHY. On receipt of the PLME-SET-TRX-STATE.confirm primitive with a status of either SUCCESS or TX_ON, the constructed frame shall then be transmitted by issuing the PD-DATA.request primitive. Finally, on receipt of the PD-DATA.confirm primitive, the MAC sublayer shall disable the transmitter by issuing the PLME-SET-TRX-STATE.request primitive with a state of RX_ON or TRX_OFF to the PHY, depending on whether the receiver is to be enabled following the transmission. In the case where the acknowledgment request subfield of the frame control field is set to one, the MAC sublayer shall enable the receiver immediately following the transmission of the frame by issuing the PLME-SET-TRX-STATE.request primitive with a state of RX_ON to the PHY.

7.5.6.2 Reception and rejection

Each device may choose whether the MAC sublayer is to enable its receiver during idle periods. During these idle periods, the MAC sublayer shall still service transceiver task requests from the next higher layer. A transceiver task shall be defined as a transmission request with acknowledgment reception, if required, or a reception request. On completion of each transceiver task, the MAC sublayer shall request that the PHY enables or disables its receiver, depending on the values of *macBeaconOrder* and *macRxOnWhenIdle*. If *macBeaconOrder* is less than 15, the value of *macRxOnWhenIdle* shall be considered relevant only during idle periods of the CAP of the incoming superframe. If *macBeaconOrder* is equal to 15, the value of *macRxOnWhenIdle* shall be considered relevant at all times.

Due to the nature of radio communications, a device with its receiver enabled will be able to receive and decode transmissions from all devices complying with IEEE P802.15.4REVb/D6 that are currently operating on the same channel and are in its POS, along with interference from other sources. The MAC sublayer shall, therefore, be able to filter incoming frames and present only the frames that are of interest to the upper layers.

For the first level of filtering, the MAC sublayer shall discard all received frames that do not contain a correct value in their FCS field in the MFR (see 7.2.1.9). The FCS field shall be verified on reception by recalculating the purported FCS over the MHR and MAC payload of the received frame and by subsequently comparing this value with the received FCS field. The FCS field of the received frame shall be considered to be correct if these values are the same and incorrect otherwise.

The second level of filtering shall be dependent on whether the MAC sublayer is currently operating in promiscuous mode. In promiscuous mode, the MAC sublayer shall pass all frames received after the first filter directly to the upper layers without applying any more filtering or processing. The MAC sublayer shall be in promiscuous mode if *macPromiscuousMode* is set to TRUE.

If the MAC sublayer is not in promiscuous mode (i.e., *macPromiscuousMode* is set to FALSE), it shall accept only frames that satisfy all of the following third level filtering requirements:

— The frame type subfield shall not contain a reserved frame type.
— The frame version subfield shall not contain a reserved value.
— If a destination PAN identifier is included in the frame, it shall match *macPANId* or shall be the broadcast PAN identifier (0xffff).
— If a short destination address is included in the frame, it shall match either *macShortAddress* or the broadcast address (0xffff). Otherwise, if an extended destination address is included in the frame, it shall match *aExtendedAddress*.
— If the frame type indicates that the frame is a beacon frame, the source PAN identifier shall match *macPANId* unless *macPANId* is equal to 0xffff, in which case the beacon frame shall be accepted regardless of the source PAN identifier.
— If only source addressing fields are included in a data or MAC command frame, the frame shall be accepted only if the device is the PAN coordinator and the source PAN identifier matches *macPANId*.

With overhead reduction, one should be able to accept an incoming frame also if it does not always have an FCS. Moreover, if destination addressing fields are omitted from the sent frame, the current reception procedure does not allow reception by a device not being the PAN coordinator.
In my mind, we need the following (but this could be "revision material"):
- The frame type subfield shall not contain a reserved frame type.
- The frame version subfield shall not contain a reserved value.
- If a destination PAN identifier is included in the frame, it shall be in the macPANIdTable (i.e., a set, not a singleton). This table shall include the broadcast PANId. Note: this would also suit 802.15.4f).
- If a short destination address is included in the frame, the (short address, PANId) pair shall be in the DeviceDescriptor. Note: this allows more than one pair (in practice, mostly one), and allows smooth transitions where new short addresses are handed out.
If the frame type is a beacon frame, the source PANId shall be accepted if it is in the sourcePANIdTable (i.e., a set, not a singleton). This table would provide as special case the wild card (which currently is 0xfff). Note: this seems to be close to the proposal by Ember re ZigBee device discovery. If a device discovers a network, it may already have some idea as to PANId (more or less, network ids) it may be interested in (e.g., preconfigured set, a la corporate WiFi access point ids).
If only source addressing fields are included in a frame, the frame may be accepted if the source occurs in a SourceTable (source address filtering). Note: I have not implemented this technique yet (Item #6 with 08/828r9), due to the fact that some of the details require more discussion amongst technical experts.
If no addressing fields are present, the device may be accepted if the frame could possibly be valid, due to side information (e.g., TSCH time slot for A --> B communications). Similarly, with unsecured ACKs 802.15.4-2006 style, one never had addressing fields and only after reading the above several times did I see it would pass third level filtering. Perhaps, we can clean this up and make this less cumbersome?

If any of the third level filtering requirements are not satisfied, the MAC sublayer shall discard the incoming frame without processing it further. If all of the third level filtering requirements are satisfied, the frame shall be considered valid and processed further. For valid frames that are not broadcast, if the frame type subfield indicates a data or MAC command frame and the acknowledgment request subfield of the frame control field is set to one, the MAC sublayer shall send an acknowledgment frame. Prior to the transmission of the acknowledgment frame, the sequence number included in the received data or MAC command frame shall be copied into the sequence number field of the acknowledgment frame. This step will allow the transaction originator to know that it has received the appropriate acknowledgment frame.

I could nowhere find info as to what info is stored by the originator of a frame that needs to be acknowledged (also not in 7.5.6.4.3, so it seems). In my mind, one needs the following for A to B communication that requires (secured) acknowledgement: A stores the following tuple (B, frame counter, time stamp, key used, security level used). Note that, with 802.15.4e, the DSN entry is part of the frame counter, so this would include (B, DSN). One should only send an ACK if the frame was really received properly (i.e., after incoming frame seurity processing).

If the PAN ID compression subfield of the frame control field is set to one and both destination and source addressing information is included in the frame, the MAC sublayer shall assume that the omitted source PAN identifier field is identical to the destination PAN identifier field.

The device shall process the frame using the incoming frame security procedure described in 7.5.8.2.3.

If the status from the incoming frame security procedure is not SUCCESS, the MLME shall issue the corresponding .confirm or MLME-COMM-STATUS.indication primitive with the status parameter set to the status from the incoming frame security procedure, indicating the error, and with the security-related parameters set to the corresponding parameters returned by the unsecuring process.

If the valid frame is a data frame, the MAC sublayer shall pass the frame to the next higher layer. This is achieved by issuing the MCPS-DATA.indication primitive containing the frame information. The security-related parameters of the MCPS-DATA.indication primitive shall be set to the corresponding parameters returned by the unsecuring process.

If the valid frame is a MAC command or beacon frame, it shall be processed by the MAC sublayer accordingly and a corresponding .confirm or .indication primitive may be sent to the next higher layer. The security-related parameters of the corresponding .confirm or .indication primitive shall be set to the corresponding parameters returned by the unsecuring process.

7.5.6.3 Extracting pending data from a coordinator

A device on a beacon-enabled PAN can determine whether any frames are pending for it by examining the contents of the received beacon frame, as described in 7.5.4.1. If the address of the device is contained in the address list field of the beacon frame and *macAutoRequest* is TRUE, the MLME of the device shall send a data request command (see 7.3.4) to the coordinator during the CAP with the acknowledgment request subfield of the frame control field set to one; the only exception to this is if the beacon frame is received while performing an active or passive scan (see 7.5.2.1). There are two other cases for which the MLME shall send a data request command to the coordinator. The first case is when the MLME receives the MLME-POLL.request primitive. In the second case, a device may send a data request command *macResponseWaitTime* symbols after the acknowledgment to a request command frame, such as during the association procedure. If the data request is intended for the PAN coordinator, the destination address information may be omitted.

If the data request command originated from an MLME-POLL.request primitive, the MLME shall perform the security process on the data request command based on the SecurityLevel, KeyIdMode, KeySource and KeyIndex parameters of the MLME-POLL.request primitive, according to 7.5.8.2.1. Otherwise, the MLME shall perform the security process on the data request command based on the *macAutoRequestSecurityLevel*, *macAutoRequestKeyIdMode*, *macAutoRequestKeySource* and *macAutoRequestKeyIndex* PIB attributes, according to 7.5.8.2.1.

On successfully receiving a data request command, the coordinator shall send an acknowledgment frame, thus confirming its receipt. If the coordinator has enough time to determine whether the device has a frame pending before sending the acknowledgment frame (see 7.5.6.4.2), it shall set the frame pending subfield of the frame control field of the acknowledgment frame accordingly to indicate whether a frame is actually pending for the device. If this is not possible, the coordinator shall set the frame pending subfield of the acknowledgment frame to one.

On receipt of the acknowledgment frame with the frame pending subfield set to zero, the device shall conclude that there are no data pending at the coordinator.

On receipt of the acknowledgment frame with the frame pending subfield set to one, a device shall enable its receiver for at most *macMaxFrameTotalWaitTime* CAP symbols in a beacon-enabled PAN, or symbols in a

nonbeacon-enabled PAN, to receive the corresponding data frame from the coordinator. If there is an actual data frame pending within the coordinator for the requesting device, the coordinator shall send the frame to the device using one of the mechanisms described in this subclause. If there is no data frame pending for the requesting device, the coordinator shall send a data frame without requesting acknowledgment to the device containing a zero length payload, indicating that no data are present, using one of the mechanisms described in this subclause.

The data frame following the acknowledgment of the data request command shall be transmitted using one of the following mechanisms:

— Without using CSMA-CA, if the MAC sublayer can commence transmission of the data frame between *aTurnaroundTime* and (*aTurnaroundTime* + *aUnitBackoffPeriod*) symbols, on a backoff slot boundary, and there is time remaining in the CAP for the message, appropriate IFS, and acknowledgment (see 6.4.1 for an explanation of *aTurnAroundTime*). If a requested acknowledgment frame is not received following this data frame, the process shall begin anew following the receipt of a new data request command.
— Using CSMA-CA, otherwise.

If the requesting device does not receive a data frame from the coordinator within *macMaxFrameTotalWaitTime* CAP symbols in a beacon-enabled PAN, or symbols in a nonbeacon-enabled PAN, or if the requesting device receives a data frame from the coordinator with a zero length payload, it shall conclude that there are no data pending at the coordinator. If the requesting device does receive a data frame from the coordinator, it shall send an acknowledgment frame, if requested, thus confirming receipt.

If the frame pending subfield of the frame control field of the data frame received from the coordinator is set to one, the device still has more data pending with the coordinator. In this case it may extract the data by sending a new data request command to the coordinator.

7.5.6.4 Use of acknowledgments and retransmissions

A data or MAC command frame shall be sent with the acknowledgment request subfield of its frame control field set appropriately for the frame. A beacon or acknowledgment frame shall always be sent with the acknowledgment request subfield set to zero. Similarly, any frame that is broadcast shall be sent with its acknowledgment request subfield set to zero.

7.5.6.4.1 No acknowledgment

A frame transmitted with its acknowledgment request subfield set to zero shall not be acknowledged by its intended recipient. The originating device shall assume that the transmission of the frame was successful.

The message sequence chart in Figure 71 shows the scenario for transmitting a single frame of data from an originator to a recipient without requiring an acknowledgment. In this case, the originator transmits the data frame with the acknowledgment request (AR) subfield of the frame control field equal to zero.

7.5.6.4.2 Acknowledgment

A frame transmitted with the acknowledgment request subfield of its frame control field set to one shall be acknowledged by the recipient. If the intended recipient correctly receives the frame, it shall generate and send an acknowledgment frame containing the same DSN from the data or MAC command frame that is being acknowledged.

The transmission of an acknowledgment frame in a nonbeacon-enabled PAN or in the CFP shall commence *aTurnaroundTime* symbols after the reception of the last symbol of the data or MAC command frame. The transmission of an acknowledgment frame in the CAP shall commence either *aTurnaroundTime* symbols

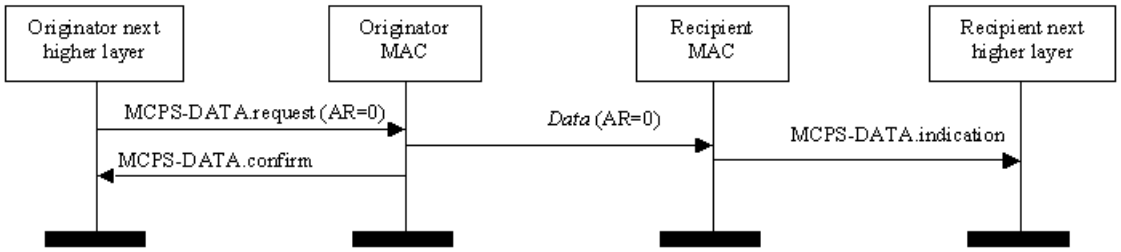


Figure 71—Successful data transmission without an acknowledgment

after the reception of the last symbol of the data or MAC command frame or at a backoff slot boundary. In the latter case, the transmission of an acknowledgment frame shall commence between *aTurnaroundTime* and (*aTurnaroundTime* + *aUnitBackoffPeriod*) symbols after the reception of the last symbol of the data or MAC command frame. The constant *aTurnaroundTime* is defined in Table 22 (in 6.4.1).

The message sequence chart in Figure 72 shows the scenario for transmitting a single frame of data from an originator to a recipient with an acknowledgment. In this case, the originator indicates to the recipient that it requires an acknowledgment by transmitting the data frame with the acknowledgment request (AR) subfield of the frame control field set to one.

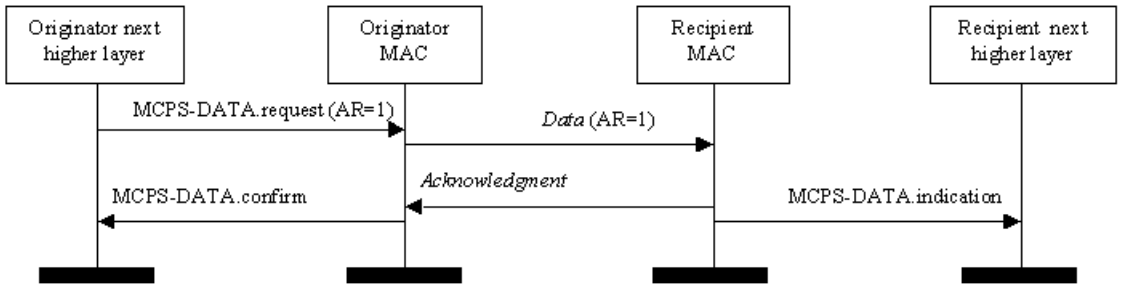


Figure 72—Successful data transmission with an acknowledgment

7.5.6.4.3 Retransmissions

A device that sends a frame with the acknowledgment request subfield of its frame control field set to zero shall assume that the transmission was successfully received and shall hence not perform the retransmission procedure.

A device that sends a data or MAC command frame with its acknowledgment request subfield set to one shall wait for at most *macAckWaitDuration* symbols for the corresponding acknowledgment frame to be received. If an acknowledgment frame is received within *macAckWaitDuration* symbols and contains the same DSN as the original transmission, the transmission is considered successful, and no further action regarding retransmission shall be taken by the device. If an acknowledgment is not received within *macAck-WaitDuration* symbols or an acknowledgment is received containing a DSN that was not the same as the original transmission, the device shall conclude that the single transmission attempt has failed.

If a single transmission attempt has failed and the transmission was indirect, the coordinator shall not retransmit the data or MAC command frame. Instead, the frame shall remain in the transaction queue of the

coordinator and can only be extracted following the reception of a new data request command. If a new data request command is received, the originating device shall transmit the frame using the same DSN as was used in the original transmission.

If a single transmission attempt has failed and the transmission was direct, the device shall repeat the process of transmitting the data or MAC command frame and waiting for the acknowledgment, up to a maximum of *macMaxFrameRetries* times. The retransmitted frame shall contain the same DSN as was used in the original transmission. Each retransmission shall only be attempted if it can be completed within the same portion of the superframe, i.e., the CAP or a GTS in which the original transmission was attempted. If this timing is not possible, the retransmission shall be deferred until the same portion in the next superframe. If an acknowledgment is still not received after *macMaxFrameRetries* retransmissions, the MAC sublayer shall assume the transmission has failed and notify the next higher layer of the failure.

7.5.6.5 Promiscuous mode

A device may activate promiscuous mode by setting *macPromiscuousMode*. If the MLME is requested to set *macPromiscuousMode* to TRUE, the MLME shall then request that the PHY enable its receiver. This request is achieved when the MLME issues the PLME-SET-TRX-STATE.request primitive with a state of RX_ON.

When in promiscuous mode, the MAC sublayer shall process received frames according to 7.5.6.2 and pass all frames correctly received to the next higher layer using the MCPS-DATA.indication primitive. The source and destination addressing mode parameters shall each be set to 0x00, the MSDU parameter shall contain the MHR concatenated with the MAC payload (see Figure 41), and the msduLength parameter shall contain the total number of octets in the MHR concatenated with the MAC payload. The mpduLinkQuality shall be valid.

If the MLME is requested to set *macPromiscuousMode* to FALSE, the MLME shall request that the PHY set its receiver to the state specified by *macRxOnWhenIdle*. This is achieved by the MLME issuing the PLME-SET-TRX-STATE.request primitive (see 6.2.2.7) with the state set accordingly.

7.5.6.6 Transmission scenarios

Due to the imperfect nature of the radio medium, a transmitted frame does not always reach its intended destination. Figure 73 illustrates three different data transmission scenarios:

— *Successful data transmission.* The originator MAC sublayer transmits the data frame to the recipient via the PHY data service. In waiting for an acknowledgment, the originator MAC sublayer starts a timer that will expire after *macAckWaitDuration* symbols. The recipient MAC sublayer receives the data frame, sends an acknowledgment back to the originator, and passes the data frame to the next higher layer. The originator MAC sublayer receives the acknowledgment from the recipient before its timer expires and then disables and resets the timer. The data transfer is now complete, and the originator MAC sublayer issues a success confirmation to the next higher layer.
— *Lost data frame.* The originator MAC sublayer transmits the data frame to the recipient via the PHY data service. In waiting for an acknowledgment, the originator MAC sublayer starts a timer that will expire after *macAckWaitDuration* symbols. The recipient MAC sublayer does not receive the data frame and so does not respond with an acknowledgment. The timer of the originator MAC sublayer expires before an acknowledgment is received, and therefore the data transfer has failed. If the transmission was direct, the originator retransmits the data, and this entire sequence may be repeated up to a maximum of *macMaxFrameRetries* times; if a data transfer attempt fails a total of (1 + *macMaxFrameRetries*) times, the originator MAC sublayer will issue a failure confirmation to the next higher layer. If the transmission was indirect, the data frame will remain in the transaction queue until either another request for the data is received and correctly acknowledged or until *macTransactionPersistenceTime* is reached. If *macTransactionPersistenceTime* is reached, the transaction infor-

mation will be discarded and the MAC sublayer will issue a failure confirmation to the next higher layer.

— *Lost acknowledgment frame.* The originator MAC sublayer transmits the data frame to the recipient via the PHY data service. In waiting for an acknowledgment, the originator MAC sublayer starts a timer that will expire after *macAckWaitDuration* symbols. The recipient MAC sublayer receives the data frame, sends an acknowledgment back to the originator, and passes the data frame to the next higher layer. The originator MAC sublayer does not receive the acknowledgment frame and its timer expires, and therefore the data transfer has failed. If the transmission was direct, the originator retransmits the data, and this entire sequence may be repeated up to a maximum of *macMaxFrameRetries* times; if a data transfer attempt fails a total of (1 + *macMaxFrameRetries*) times, the originator MAC sublayer will issue a failure confirmation to the next higher layer. If the transmission was indirect, the data frame will remain in the transaction queue until either another request for the data is received and correctly acknowledged or until *macTransactionPersistenceTime* is reached. If *macTransactionPersistenceTime* is reached, the transaction information will be discarded and the MAC sublayer will issue a failure confirmation to the next higher layer.

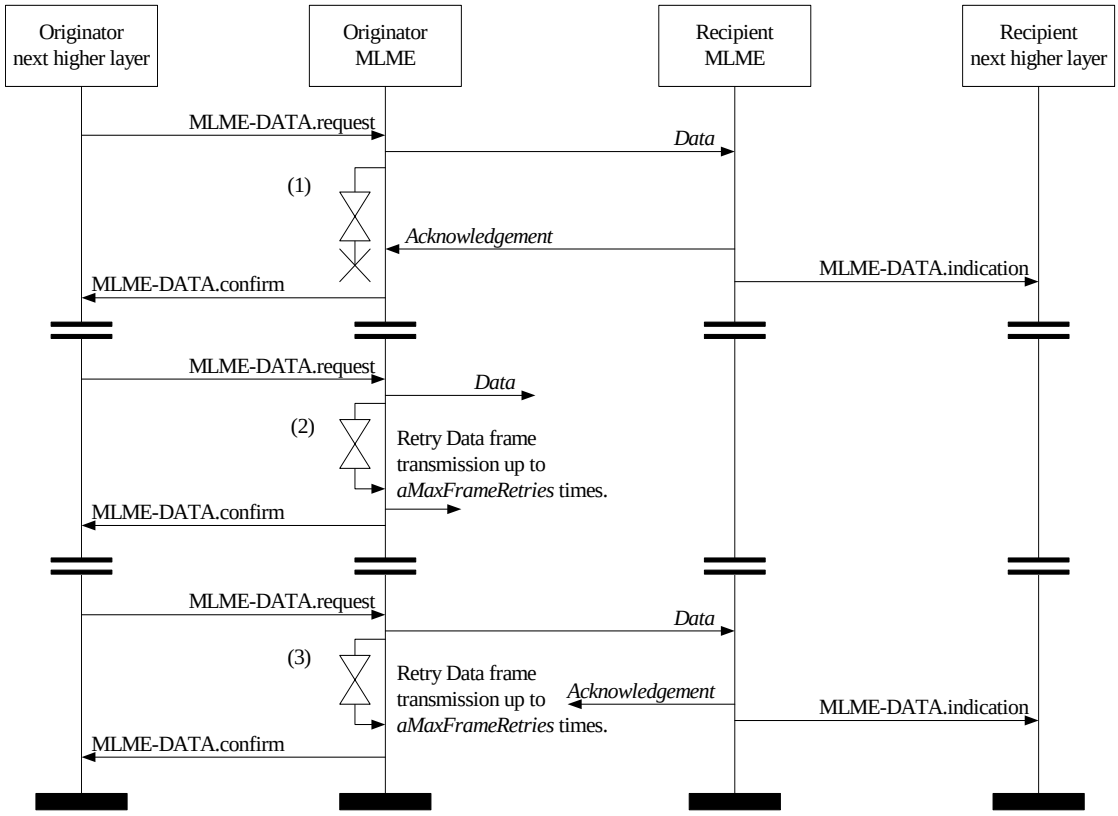


Figure 73—Transmission scenarios, using direct transmission, for frame reliability

7.5.7 GTS allocation and management

A GTS allows a device to operate on the channel within a portion of the superframe that is dedicated (on the PAN) exclusively to that device. A GTS shall be allocated only by the PAN coordinator, and it shall be used only for communications between the PAN coordinator and a device associated with the PAN through the PAN coordinator. A single GTS may extend over one or more superframe slots. The PAN coordinator may allocate up to seven GTSs at the same time, provided there is sufficient capacity in the superframe.

A GTS shall be allocated before use, with the PAN coordinator deciding whether to allocate a GTS based on the requirements of the GTS request and the current available capacity in the superframe. GTSs shall be allocated on a first-come-first-served basis, and all GTSs shall be placed contiguously at the end of the superframe and after the CAP. Each GTS shall be deallocated when the GTS is no longer required, and a GTS can be deallocated at any time at the discretion of the PAN coordinator or by the device that originally requested the GTS. A device that has been allocated a GTS may also operate in the CAP.

A data frame transmitted in an allocated GTS shall use only short addressing.

The management of GTSs shall be undertaken by the PAN coordinator only. To facilitate GTS management, the PAN coordinator shall be able to store all the information necessary to manage seven GTSs. For each GTS, the PAN coordinator shall be able to store its starting slot, length, direction, and associated device address.

The GTS direction, which is relative to the data flow from the device that owns the GTS, is specified as either transmit or receive. The device address and direction shall, therefore, uniquely identify each GTS. Each device may request one transmit GTS and/or one receive GTS. For each allocated GTS, the device shall be able to store its starting slot, length, and direction. If a device has been allocated a receive GTS, it shall enable its receiver for the entirety of the GTS. In the same way, the PAN coordinator shall enable its receiver for the entirety of the GTS if a device has been allocated a transmit GTS. If a data frame is received during a receive GTS and an acknowledgment is requested, the device shall transmit the acknowledgment frame as usual. Similarly, a device shall be able to receive an acknowledgment frame during a transmit GTS.

A device shall attempt to allocate and use a GTS only if it is currently tracking the beacons. The MLME is instructed to track beacons by issuing the MLME-SYNC.request primitive with the TrackBeacon parameter set to TRUE. If a device loses synchronization with the PAN coordinator, all its GTS allocations shall be lost.

The use of GTSs is optional.

7.5.7.1 CAP maintenance

The PAN coordinator shall preserve the minimum CAP length of *aMinCAPLength* and take preventative action if the minimum CAP is not satisfied. However, an exception shall be allowed for the accommodation of the temporary increase in the beacon frame length needed to perform GTS maintenance. If preventative action becomes necessary, the action chosen is left up to the implementation, but may include one or more of the following:

— Limiting the number of pending addresses included in the beacon.
— Not including a payload field in the beacon frame.
— Deallocating one or more of the GTSs

7.5.7.2 GTS allocation

A device is instructed to request the allocation of a new GTS through the MLME-GTS.request primitive, with GTS characteristics set according to the requirements of the intended application.

To request the allocation of a new GTS, the MLME shall send the GTS request command (see 7.3.9) to the PAN coordinator. The characteristics type subfield of the GTS characteristics field of the request shall be set to one (GTS allocation), and the length and direction subfields shall be set according to the desired characteristics of the required GTS. Since the GTS request command contains an acknowledgment request (see 7.3.3.1), the PAN coordinator shall confirm its receipt by sending an acknowledgment frame.

On receipt of a GTS request command indicating a GTS allocation request, the PAN coordinator shall first check if there is available capacity in the current superframe, based on the remaining length of the CAP and the desired length of the requested GTS. The superframe shall have available capacity if the maximum number of GTSs has not been reached and allocating a GTS of the desired length would not reduce the length of the CAP to less than *aMinCAPLength*. GTSs shall be allocated on a first-come-first-served basis by the PAN coordinator provided there is sufficient bandwidth available. The PAN coordinator shall make this decision within *aGTSDescPersistenceTime* superframes.

On receipt of the acknowledgment to the GTS request command, the device shall continue to track beacons and wait for at most *aGTSDescPersistenceTime* superframes. If no GTS descriptor for the device appears in the beacon within this time, the MLME of the device shall notify the next higher layer of the failure. This notification is achieved when the MLME issues the MLME-GTS.confirm primitive (see 7.1.7.2) with a status of NO_DATA.

When the PAN coordinator determines whether capacity is available for the requested GTS, it shall generate a GTS descriptor with the requested specifications and the 16-bit short address of the requesting device. If the GTS was allocated successfully, the PAN coordinator shall set the start slot in the GTS descriptor to the superframe slot at which the GTS begins and the length in the GTS descriptor to the length of the GTS. In addition, the PAN coordinator shall notify the next higher layer of the new GTS. This notification is achieved when the MLME of the PAN coordinator issues the MLME-GTS.indication primitive (see 7.1.7.3) with the characteristics of the allocated GTS. If there was not sufficient capacity to allocate the requested GTS, the start slot shall be set to zero and the length to the largest GTS length that can currently be supported. The PAN coordinator shall then include this GTS descriptor in its beacon and update the GTS specification field of the beacon frame accordingly. The PAN coordinator shall also update the final CAP slot subfield of the superframe specification field of the beacon frame, indicating the final superframe slot utilized by the decreased CAP. The GTS descriptor shall remain in the beacon frame for *aGTSDescPersistenceTime* superframes, after which it shall be removed automatically. The PAN coordinator shall be allowed to reduce its CAP below *aMinCAPLength* to accommodate the temporary increase in the beacon frame length due to the inclusion of the GTS descriptor.

On receipt of a beacon frame containing a GTS descriptor corresponding to *macShortAddress*, the device shall process the descriptor. The MLME of the device shall then notify the next higher layer of whether the GTS allocation request was successful. This notification is achieved when the MLME issues the MLME-GTS.confirm primitive with a status of SUCCESS (if the start slot in the GTS descriptor was greater than zero) or DENIED (if the start slot was equal to zero or if the length did not match the requested length).

7.5.7.3 GTS usage

When the MAC sublayer of a device that is not the PAN coordinator receives an MCPS-DATA.request primitive (see 7.1.1.1) with the TxOptions parameter indicating a GTS transmission, it shall determine whether it has a valid transmit GTS. If a valid GTS is found, the MAC sublayer shall transmit the data during the GTS, i.e., between its starting slot and its starting slot plus its length. At this time, the MAC sublayer shall transmit the MPDU immediately without using CSMA-CA, provided the requested transaction can be completed before the end of the GTS. If the requested transaction cannot be completed before the end of the current GTS, the MAC sublayer shall defer the transmission until the specified GTS in the next superframe. Note that the MAC must allow for the PHY overhead (see 6.7.2.2) in making this determination.

If the device has any receive GTSs, the MAC sublayer of the device shall ensure that the receiver is enabled at a time prior to the start of the GTS and for the duration of the GTS, as indicated by its starting slot and its length.

When the MAC sublayer of the PAN coordinator receives an MCPS-DATA.request primitive with the TxOptions parameter indicating a GTS transmission, it shall determine whether it has a valid receive GTS corresponding to the device with the requested destination address. If a valid GTS is found, the PAN coordi-

nator shall defer the transmission until the start of the receive GTS. In this case, the address of the device with the message requiring a GTS transmission shall not be added to the list of pending addresses in the beacon frame (see 7.5.5). At the start of the receive GTS, the MAC sublayer shall transmit the data without using CSMA-CA, provided the requested transaction can be completed before the end of the GTS. If the requested transaction cannot be completed before the end of the current GTS, the MAC sublayer shall defer the transmission until the specified GTS in the next superframe.

For all allocated transmit GTSs (relative to the device), the MAC sublayer of the PAN coordinator shall ensure that its receiver is enabled at a time prior to the start and for the duration of each GTS.

Before commencing transmission in a GTS, each device shall ensure that the data transmission, the acknowledgment, if requested, and the IFS, suitable to the size of the data frame, can be completed before the end of the GTS.

If a device misses the beacon at the beginning of a superframe, it shall not use its GTSs until it receives a subsequent beacon correctly. If a loss of synchronization occurs due to the loss of the beacon, the device shall consider all of its GTSs deallocated.

7.5.7.4 GTS deallocation

A device is instructed to request the deallocation of an existing GTS through the MLME-GTS.request primitive (see 7.1.7.1), using the characteristics of the GTS it wishes to deallocate. From this point onward, the GTS to be deallocated shall not be used by the device, and its stored characteristics shall be reset.

To request the deallocation of an existing GTS, the MLME shall send the GTS request command (see 7.3.9) to the PAN coordinator. The characteristics type subfield of the GTS characteristics field of the request shall be set to zero (i.e., GTS deallocation), and the length and direction subfields shall be set according to the characteristics of the GTS to deallocate. Since the GTS request command contains an acknowledgment request (see 7.3.3.1), the PAN coordinator shall confirm its receipt by sending an acknowledgment frame. On receipt of the acknowledgment to the GTS request command, the MLME shall notify the next higher layer of the deallocation. This notification is achieved when the MLME issues the MLME-GTS.confirm primitive (see 7.1.7.2) with a status of SUCCESS and a GTSCharacteristics parameter with its characteristics type subfield set to zero. If the GTS request command is not received correctly by the PAN coordinator, it shall determine that the device has stopped using its GTS by the procedure described in 7.5.7.6.

On receipt of a GTS request command with the characteristics type subfield of the GTS characteristics field set to zero (GTS deallocation), the PAN coordinator shall attempt to deallocate the GTS. If the GTS characteristics contained in the GTS request command do not match the characteristics of a known GTS, the PAN coordinator shall ignore the request. If the GTS characteristics contained in the GTS request command match the characteristics of a known GTS, the MLME of the PAN coordinator shall deallocate the specified GTS and notify the next higher layer of the change. This notification is achieved when the MLME issues the MLME-GTS.indication primitive (see 7.1.7.3) with a GTSCharacteristics parameter containing the characteristics of the deallocated GTS and a characteristics type subfield set to zero. The PAN coordinator shall also update the final CAP slot subfield of the superframe specification field of the beacon frame, indicating the final superframe slot utilized by the increased CAP. It shall not add a descriptor to the beacon frame to describe the deallocation.

GTS deallocation may be initiated by the PAN coordinator either due to a deallocation request from the next higher layer, the GTS expiring (see 7.5.7.6) or maintenance required to maintain the minimum CAP length, *aMinCAPLength* (see 7.5.7.1).

When a GTS deallocation is initiated by the next higher layer of the PAN coordinator, the MLME shall receive the MLME-GTS.request primitive with the GTS characteristics field of the request set to zero (i.e.,

GTS deallocation), and the length and direction subfields set according to the characteristics of the GTS to deallocate.

When a GTS deallocation is initiated by the PAN coordinator either due to the GTS expiring or due to CAP maintenance, the MLME shall notify the next higher layer of the change. This notification is achieved when the MLME issues the MLME-GTS.indication primitive with a GTSCharacteristics parameter containing the characteristics of the deallocated GTS and a characteristics type subfield set to zero.

In the case of any deallocation initiated by PAN coordinator, the PAN coordinator shall deallocate the GTS and add a GTS descriptor into its beacon frame corresponding to the deallocated GTS, but with its starting slot set to zero. The descriptor shall remain in the beacon frame for *aGTSDescPersistenceTime* superframes. The PAN coordinator shall be allowed to reduce its CAP below *aMinCAPLength* to accommodate the temporary increase in the beacon frame length due to the inclusion of the GTS descriptor.

On receipt of a beacon frame containing a GTS descriptor corresponding to *macShortAddress* and a start slot equal to zero, the device shall immediately stop using the GTS. The MLME of the device shall then notify the next higher layer of the deallocation. This notification is achieved when the MLME issues the MLME-GTS.indication primitive with a GTSCharacteristics parameter containing the characteristics of the deallocated GTS and a characteristics type subfield set to zero.

7.5.7.5 GTS reallocation

The deallocation of a GTS may result in the superframe becoming fragmented. For example, Figure 74 shows three stages of a superframe with allocated GTSs. In stage 1, three GTSs are allocated starting at slots 14, 10, and 8, respectively. If GTS 2 is now deallocated (stage 2), there will be a gap in the superframe dur-

ing which nothing can happen. To solve this, GTS 3 will have to be shifted to fill the gap, thus increasing the size of the CAP (stage 3).

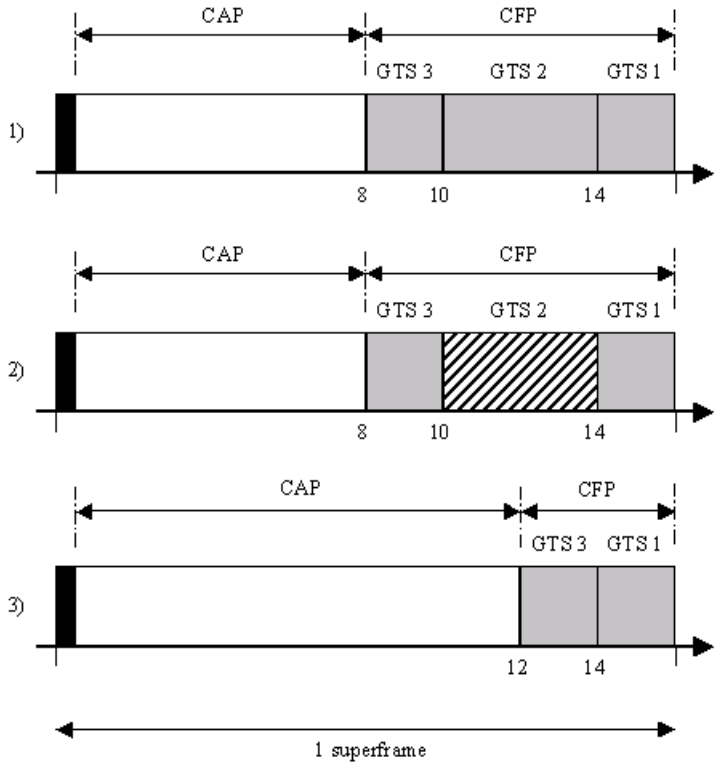


Figure 74—CFP defragmentation on GTS deallocations

The PAN coordinator shall ensure that any gaps occurring in the CFP, appearing due to the deallocation of a GTS, are removed to maximize the length of the CAP.

When a GTS is deallocated by the PAN coordinator, it shall add a GTS descriptor into its beacon frame indicating that the GTS has been deallocated. If the deallocation is initiated by a device, the PAN coordinator shall not add a GTS descriptor into its beacon frame to indicate the deallocation. For each device with an allocated GTS having a starting slot lower than the GTS being deallocated, the PAN coordinator shall update the GTS with the new starting slot and add a GTS descriptor to its beacon corresponding to this adjusted GTS. The new starting slot is computed so that no space is left between this GTS and either the end of the CFP, if the GTS appears at the end of the CFP, or the start of the next GTS in the CFP.

In situations where multiple reallocations occur at the same time, the PAN coordinator may choose to perform the reallocation in stages. The PAN coordinator shall keep each GTS descriptor in its beacon for *aGTSDescPersistenceTime* superframes.

On receipt of a beacon frame containing a GTS descriptor corresponding to *macShortAddress* and a direction and length corresponding to one of its GTSs, the device shall adjust the starting slot of the GTS corresponding to the GTS descriptor and start using it immediately.

In cases where it is necessary for the PAN coordinator to include a GTS descriptor in its beacon, it shall be allowed to reduce its CAP below *aMinCAPLength* to accommodate the temporary increase in the beacon

frame length. After *aGTSDescPersistenceTime* superframes, the PAN coordinator shall remove the GTS descriptor from the beacon.

7.5.7.6 GTS expiration

The MLME of the PAN coordinator shall attempt to detect when a device has stopped using a GTS using the following rules:

- For a transmit GTS, the MLME of the PAN coordinator shall assume that a device is no longer using its GTS if a data frame is not received from the device in the GTS at least every 2**n* superframes, where *n* is defined below.
- For receive GTSs, the MLME of the PAN coordinator shall assume that a device is no longer using its GTS if an acknowledgment frame is not received from the device at least every 2**n* superframes, where *n* is defined below. If the data frames sent in the GTS do not require acknowledgment frames, the MLME of the PAN coordinator will not be able to detect whether a device is using its receive GTS. However, the PAN coordinator is capable of deallocating the GTS at any time.

The value of *n* is defined as follows:

$n = 2^{(8-macBeaconOrder)}$ $\quad 0 \le macBeaconOrder \le 8$

$n = 1$ $\quad 9 \le macBeaconOrder \le 14$

7.5.8 Frame security

The MAC sublayer is responsible for providing security services on specified incoming and outgoing frames when requested to do so by the higher layers. IEEE P802.15.4REVb/D6 supports the following security services (see 5.5.6 for definitions):

- Data confidentiality
- Data authenticity
- Replay protection

The information determining how to provide the security is found in the security-related PIB (see Table 88).

7.5.8.1 Security-related MAC PIB attributes

The security-related MAC PIB attributes contain:

- Key table (*macKeyTable*, *macKeyTableEntries*)
- Device table (*macDeviceTable*, *macDeviceTableEntries*)
- Minimum security level table (*macSecurityLevelTable*, *macSecurityLevelTableEntries*)
- Frame counter (*macFrameCounter*)
- Automatic request attributes (*macAutoRequestSecurityLevel*, *macAutoRequestFrameCounterMode*, *macAutoRequestKeyIdMode*, *macAutoRequestKeySource*, *macAutoRequestKeyIndex*)
- Default key source (*macDefaultKeySource*)
- PAN coordinator address (*macPANCoordExtendedAddress*, *macPANCoordShortAddress*)
- Key source table (*macKeySourceTable*, *macKeySourceTableEntries*)
- Device time (*macCurrentTime*)

7.5.8.1.1 Key table

The key table holds key descriptors (keys with related key-specific information) that are required for security processing of outgoing and incoming frames. Key-specific information in the key table is identified

based on information explicitly contained in the requesting primitive or in the received frame, as described in the outgoing frame key retrieval procedure (see 7.5.8.2.2) and the incoming frame key retrieval procedure (see 7.5.8.2.4), as well as in the KeyDescriptor lookup procedure (see 7.5.8.2.6) and the KeySourceDescriptor lookup procedure (see 7.5.8.2.9).

7.5.8.1.2 Device table

The device table holds device descriptors (device-specific addressing information and security-related information) that, when combined with key-specific information from the key table, provide all the keying material needed to secure outgoing (see 7.5.8.2.1) and unsecure incoming frames (see 7.5.8.2.3). Device-specific information in the device table is identified based on the originator of the frame, as described in the incoming frame device retrieval procedure (see 7.5.8.2.5) and the DeviceDescriptor lookup procedure (see 7.5.8.2.8).

7.5.8.1.3 Minimum security level table

The minimum security level table holds information regarding the security level the device expects to have been applied by the originator of a frame, depending on frame type and, if it concerns a MAC command frame, the command frame identifier. Security processing of an incoming frame will fail if the frame is not adequately protected, as described in the incoming frame security procedure (see 7.5.8.2.3) and in the incoming security level checking procedure (see 7.5.8.2.11).

7.5.8.1.4 Frame counter

The 6-octet frame counter is used to provide replay protection and semantic security of the cryptographic building block used for securing outgoing frames. The frame counter is included in each secured frame and is one of the elements required for the unsecuring operation at the recipient(s). The frame counter is incremented each time an outgoing frame is secured, as described in the outgoing frame security procedure (see 7.5.8.2.1). When the frame counter is used, it is scaled-down towards a 4 1/2-octet value for usage with a particular key. When this scaled frame counter reaches its maximum value of 0xffffffffff, the associated keying material can no longer be used, thus requiring this particular key to be updated. This provides a mechanism for ensuring that the keying material for every frame is unique and, thereby, provides for sequential freshness. The frame counter may be included with the secured frame in a compressed format, thus allowing bandwidth savings in scenarios where the full frame counter value can be faithfully reconstructed from the comprssed frame counter value, as contained in an incoming frame, and locally maintained status information.

7.5.8.1.5 Automatic request attributes

Automatic request attributes hold all the information needed to secure outgoing frames generated automatically and not as a result of a higher layer primitive, as is the case with automatic data requests.

7.5.8.1.6 Default key source

The default key source is information commonly shared between originator and receipient(s) of a secured frame, which, when combined with additional information explicitly contained in the requesting primitive or in the received frame, allows an originator or a recipient to determine the key required for securing or unsecuring this frame, respectively. This provides a mechanism for significantly reducing the overhead of security information contained in secured frames in particular use cases (see 7.5.8.2.2 and 7.5.8.2.4).

7.5.8.1.7 PAN coordinator address

The address of the PAN coordinator is information commonly shared between all devices in a PAN, which, when combined with additional information explicitly contained in the requesting primitive or in the

received frame, allows an originator of a frame directed to the PAN coordinator or a recipient of a frame originating from the PAN coordinator to determine the key and security-related information required for securing or unsecuring, respectively, this frame (see 7.5.8.2.2 and 7.5.8.2.4).

7.5.8.1.8 Key source table

The key source table holds key source descriptors (key-specific information) that, when combined with additional information explicitly contained in the requesting primitive or in the received frame, allow an originator or a recipient to determine the key required for securing or unsecuring this frame, respectively (see 7.5.8.2.2 and 7.5.8.2.4). For received frames, this information may be either implicitly derived from the addressing fields of the frame or explicitly indicated in the frame by its originator, as described in the outgoing frame key retrieval procedure (see 7.5.8.2.2) and the incoming frame key retrieval procedure (see 7.5.8.2.4).

7.5.8.1.9 Device time

The 6-octet device time is exploited to enhance the security functionality, by providing the capability to detect whether a received frame was purportedly sent relatively recently. This detection is implemented by correlating frame counter with the device time, when securing outgoing frames and unsecuring incoming frames. Device time is expressed with granularity of 16kHz. When the device time reaches its maximum value, security can no longer be used.

7.5.8.2 Functional description

A device may optionally implement security. A device that does not implement security shall not provide a mechanism for the MAC sublayer to perform any cryptographic transformation on incoming and outgoing frames, nor require any PIB attributes associated with security. A device that implements security shall provide a mechanism for the MAC sublayer to provide cryptographic transformations on incoming and outgoing frames using information in the PIB attributes associated with security when the *macSecurityEnabled* attribute is set to TRUE.

If the MAC sublayer is required to transmit a frame or receives an incoming frame, the MAC sublayer shall process the frame as specified in 7.5.8.2.1 and 7.5.8.2.3, respectively.

7.5.8.2.1 Outgoing frame security procedure

The inputs to this procedure are the frame to be secured, and the SecurityLevel, FrameCounterMode, KeyIdMode, KeySource and KeyIndex parameters from the originating primitive or automatic request PIB attributes. The outputs from this procedure are the status of the procedure and, if this status is SUCCESS, the secured frame.

The outgoing frame security procedure involves the following steps:

a) The procedure shall set *macCurrentTime* to the current absolute device time, measured in 16kHz granularity. If this procedure fails, the procedure shall return with a status of TIME_ERROR (TBD).
b) If the security enabled subfield of the frame control field of the frame to be secured is set to zero, the procedure shall set the security level to zero.
c) If the security enabled subfield of the frame control field of the frame to be secured is set to one, the procedure shall set the security level to the SecurityLevel parameter. If the resulting security level is zero, the procedure shall return with a status of UNSUPPORTED_SECURITY.
d) If the *macSecurityEnabled* attribute is set to FALSE and the security level is unequal to zero, the procedure shall return with a status of UNSUPPORTED_SECURITY.
e) The procedure shall determine whether the frame to be secured satisfies the constraint on the maximum length of MAC frames, as follows:

i) The procedure shall set the length M, in octets, of the authentication field to zero if the security level is equal to zero and shall determine this value from the security level and Table 96 otherwise.
ii) The procedure shall determine the length AuxLen, in octets, of the auxiliary security header (see 7.6.2) using KeyIdMode and the security level.
iii) The procedure shall determine the data expansion as AuxLen+M.
iv) The procedure shall check whether the length of the frame to be secured, including data expansion and frame check sequence, is less than or equal to *aMaxPHYPacketSize*. If this check fails, the procedure shall return with a status of FRAME_TOO_LONG.

f) If the security level is zero, the procedure shall set the secured frame to be the frame to be secured and return with the secured frame and a status of SUCCESS.
g) The procedure shall obtain the KeyDescriptor using the outgoing frame key retrieval procedure as described in 7.5.8.2.2. If that procedure fails, the procedure shall return with a status of UNAVAILABLE_KEY.
h) If the Blacklisted element of the KeyDescriptor is set to TRUE, the procedure shall return with a status of KEY_ERROR.
i) The procedure shall set the frame counter to the maximum value of the *macFrameCounter*and the *macCurrentTime* attributes and shall set the scaled frame counter to the difference between the frame counter and the KeyOffset element of the KeyDescriptor.
j) If the scaled frame counter value is negative, the procedure shall return with a status of KEY_ERROR. If the scaled frame counter is greater than or equal to 0xffffffff, the procedure shall return with a status of COUNTER_ERROR.
k) The procedure shall insert the auxiliary security header into the frame, with fields set as follows:
i) The security level subfield of the security control field shall be set to the security level.
ii) The key identifier mode subfield of the security control field shall be set to the KeyIdMode parameter.
iii) The frame counter mode subfield of the security control field shall be set to the FrameCounterMode parameter.
iv) The frame counter field shall be set to the representation of the frame counter indicated by the frame counter mode [TBD].
v) If the KeyIdMode parameter is set to a value not equal to zero, the key source and key index subfields of the key identifier field shall be set to the KeySource and KeyIndex parameters, respectively.
l) The procedure shall then use *aExtendedAddress*, the scaled frame counter, the security level and the Key element of the KeyDescriptor to produce the secured frame according to the CCM* transformation process described in the security operations (see 7.6.3.4).
i) If the SecurityLevel parameter specifies the use of encryption (see Table 96), the encryption operation shall be applied only to the actual payload field within the MAC payload, i.e., the beacon payload field (see 7.2.2.1.8), command payload field (see 7.2.2.4.3), or data payload field (see 7.2.2.2.2), depending on the frame type. The corresponding payload field is passed to the CCM* transformation process described in 7.6.3.4 as the unsecured payload (see Table 99). The resulting encrypted payload shall substitute the original payload.
ii) The remaining fields in the MAC payload part of the frame shall be passed to the CCM* transformation process described in 7.6.3.4 as the non-payload fields (see Table 99).
iii) The ordering and exact manner of performing the encryption and integrity operations and the placement of the resulting encrypted data or integrity code within the MAC payload field shall be as defined in 7.6.3.4.
m) The procedure shall increment the frame counter by one and set the *macFrameCounter* attribute to the resulting value.
n) If the *macFrameCounter* element is equal to 0xffffffff, the procedure shall set the Blacklisted element of the KeyDescriptor to TRUE.
o) The procedure shall return with the secured frame and a status of SUCCESS.

7.5.8.2.2 Outgoing frame key retrieval procedure

The inputs to this procedure are the frame to be secured, and the KeyIdMode, KeySource and KeyIndex parameters from the originating primitive. The outputs from this procedure are a passed or failed status and, if passed, a KeyDescriptor.

The outgoing frame key retrieval procedure involves the following steps:

a) If the KeyIdMode parameter is set to 0x00 (implicit key identification), the procedure shall determine the key source lookup data and key source lookup size as follows:
   i) If the destination addressing mode subfield of the frame control field of the frame is set to 0x00 and the *macPANCoordShortAddress* attribute is set to a value in the range 0x0000-0xfffd (i.e., the short address is used), the key source lookup data shall be set to the 2-octet source PAN ID field of the frame right-concatenated (see B.1.1) with the 2-octet *macPANCoordShortAddress* attribute. The key source lookup size shall be set to four.
   ii) If the destination addressing mode subfield of the frame control field of the frame is set to 0x00 and the *macPANCoordShortAddress* attribute is set to 0xfffe (i.e., the extended address is used), the key source lookup data shall be set to the 8-octet *macPANCoordExtendedAddress* attribute. The key source lookup size shall be set to eight.
   iii) If the destination addressing mode subfield of the frame control field of the frame is set to 0x02, the key source lookup data shall be set to the 2-octet destination PAN ID field of the frame right-concatenated (see B.1.1) with the 2-octet destination address field of the frame. The key source lookup size shall be set to four.
   iv) If the destination addressing mode subfield of the frame control field of the frame is set to 0x03, the key source lookup data shall be set to the 8-octet destination address field of the frame. The key source lookup size shall be set to eight.

   The key index shall be set to the single octet 0x00.

b) If the KeyIdMode parameter is set to a value not equal to 0x00 (explicit key identification), the procedure shall determine the key source lookup data and key source lookup size as follows:
   i) If the KeyIdMode parameter is set to 0x01, the key source lookup data shall be set to the 8-octet *macDefaultKeySource* attribute. The key source lookup size shall be set to eight.
   ii) If the KeyIdMode parameter is set to 0x02, the key source lookup data shall be set to the 4-octet KeySource parameter. The key source lookup size shall be set to four.
   iii) If the KeyIdMode parameter is set to 0x03, the key source lookup data shall be set to the 8-octet KeySource parameter. The key source lookup size shall be set to nine.

   The key index shall be set to the 1-octet KeyIndex parameter.

c) The procedure shall obtain the KeySourceDescriptor by passing the key source lookup data and the key source lookup size to the KeySourceDescriptor lookup procedure as described in 7.5.8.2.9. If that procedure returns with a failed status, this procedure shall also return with a failed status.
d) The procedure shall obtain the KeyDescriptor by passing the KeySourceDescriptor and the key index to the KeyDescriptor lookup procedure as described in 7.5.8.2.6. If that procedure returns with a failed status, this procedure shall also return with a failed status.
e) The procedure shall return with a passed status, having obtained the KeyDescriptor.

NOTE—For broadcast frames, the outgoing frame key retrieval procedure will result in a failed status if implicit key identification is used. Hence, one needs to use explicit key identification for broadcast frames.

7.5.8.2.3 Incoming frame security procedure

The input to this procedure is the frame to be unsecured. The outputs from this procedure are the unsecured frame, the security level, the key identifier mode, the key source, the key index and the status of the procedure. All outputs of this procedure are assumed to be invalid unless and until explicitly set in this procedure. It is assumed that the PIB attributes associating KeyDescriptors in *macKeyTable* with a single, unique device or a number of devices will have been established by the next higher layer.

The incoming frame security procedure involves the following steps:

a) The procedure shall set *macCurrentTime* to the current absolute device time, measured in 16kHz granularity.If this procedure fails, the procedure shall set the unsecured frame to be the frame to be unsecured and return with the unsecured frame, the security level, the key identifier mode, the key source, the key index and a status of TIME_ERROR (TBD).

b) If the security enabled subfield of the frame control field of the frame to be unsecured is set to zero, the procedure shall set the security level to zero.

c) If the security enabled subfield of the frame control field of the frame to be unsecured is set to one and the frame version number of the frame control field of the frame to be unsecured is set to zero, the procedure shall set the unsecured frame to be the frame to be unsecured and return with the unsecured frame, the security level, the key identifier mode, the key source, the key index and a status of UNSUPPORTED_LEGACY.

d) If the security enabled subfield of the frame control field of the frame to be unsecured is set to one, the procedure shall set the security level, the key identifier mode and the frame counter mode to the corresponding subfields of the security control field of the auxiliary security header of the frame to be unsecured, and the key source and key index to the corresponding subfields of the key identifier field of the auxiliary security header of the frame to be unsecured, if present. If the resulting security level is zero, the procedure shall set the unsecured frame to be the frame to be unsecured and return with the unsecured frame, the security level, the key identifier mode, the key source, the key index and a status of UNSUPPORTED_SECURITY.

e) If the *macSecurityEnabled* attribute is set to FALSE, the procedure shall set the unsecured frame to be the frame to be unsecured and return with the unsecured frame, the security level, the key identifier mode, the key source, the key index and a status of SUCCESS if the security level is equal to zero and with the unsecured frame, the security level, the key identifier mode, the key source, the key index and a status of UNSUPPORTED_SECURITY otherwise.

f) The procedure shall obtain the SecurityLevelDescriptor by passing the frame type and depending on whether the frame is a MAC command frame, the first octet of the MAC payload (i.e., command frame identifier for a MAC command frame) to the SecurityLevelDescriptor lookup procedure described in 7.5.8.2.10. If that procedure fails, the procedure shall set the unsecured frame to be the frame to be unsecured and return with the unsecured frame, the security level, the key identifier mode, the key source, the key index and a status of UNAVAILABLE_SECURITY_LEVEL.

g) The procedure shall determine whether the frame to be unsecured conforms to the security level policy by passing the SecurityLevelDescriptor and the security level to the incoming security level checking procedure as described in 7.5.8.2.11. If that procedure returns with a failed status, the procedure shall set the unsecured frame to be the frame to be unsecured and return with the unsecured frame, the security level, the key identifier mode, the key source, the key index and a status of IMPROPER_SECURITY_LEVEL; otherwise, if that procedure returns with a passed status and the security level is equal to zero, the procedure shall set the unsecured frame to be the frame to be unsecured and return with the unsecured frame, the security level, the key identifier mode, the key source, the key index and a status of SUCCESS.

h) The procedure shall obtain the DeviceDescriptor using the incoming frame device retrieval procedure described in 7.5.8.2.5. If that procedure fails, the procedure shall set the unsecured frame to be the frame to be unsecured and return with the unsecured frame, the security level, the key identifier mode, the key source, the key index and a status of UNAVAILABLE_DEVICE.

i) If the incoming security level checking procedure of Step g above had as output the ‘conditionally passed’ status, the procedure shall set the unsecured frame to be the frame to be unsecured and return with the unsecured frame, the security level, the key identifier mode, the key source, the key index and a status of SUCCESS if the Exempt element of the DeviceDescriptor is set to TRUE and with the unsecured frame, the security level, the key identifier mode, the key source, the key index and a status of IMPROPER_SECURITY_LEVEL otherwise.

j) The procedure shall obtain the KeyDescriptor using the incoming frame key retrieval procedure described in 7.5.8.2.4. If that procedure fails, the procedure shall set the unsecured frame to be the frame to be unsecured and return with the unsecured frame, the security level, the key identifier mode, the key source, the key index and a status of UNAVAILABLE_KEY.

k) The procedure shall obtain the KeyDeviceDescriptor using the KeyDeviceDescriptor lookup procedure described in 7.5.8.2.7. If that procedure fails or if the Blacklisted element of the KeyDeviceDescriptor is set to TRUE, the procedure shall set the unsecured frame to be the frame to be unsecured and return with the unsecured frame, the security level, the key identifier mode, the key source, the key index and a status of KEY_ERROR.

l) The procedure shall determine whether the frame to be unsecured conforms to the key usage policy by passing the KeyDescriptor, the frame type and depending on whether the frame is a MAC command frame, the first octet of the MAC payload (i.e., command frame identifier for a MAC command frame) to the incoming key usage policy checking procedure as described in 7.5.8.2.12. If that procedure fails, the procedure shall set the unsecured frame to be the frame to be unsecured and return with the unsecured frame, the security level, the key identifier mode, the key source, the key index and a status of IMPROPER_KEY_TYPE.

m) The procedure shall set the frame counter to the frame counter field of the auxiliary security header of the frame to be unsecured and shall set the scaled frame counter to the difference between the frame counter and the KeyOffset element of the KeyDescriptor.

n) If the scaled frame counter value is negative, the procedure shall set the unsecured frame to be the frame to be unsecured and return with the unsecured frame, the security level, the key identifier mode, the key source, the key index and a status of KEY_ERROR.

o) If the scaled frame counter is greater than or equal to 0xffffffffff, the procedure shall set the unsecured frame to be the frame to be unsecured and return with the unsecured frame, the security level, the key identifier mode, the key source, the key index and a status of COUNTER_ERROR.

p) The procedure shall determine whether the frame counter is greater than or equal to the FrameCounter element of the DeviceDescriptor. If this check fails, the procedure shall set the unsecured frame to be the frame to be unsecured and return with the unsecured frame, the security level, the key identifier mode, the key source, the key index and a status of COUNTER_ERROR.

q) The procedure shall then use the ExtAddress element of the DeviceDescriptor, the scaled frame counter, the security level and the Key element of the KeyDescriptor to produce the unsecured frame according to the CCM* inverse transformation process described in the security operations (see 7.6.3.5).

    i) If the security level specifies the use of encryption (see Table 96), the decryption operation shall be applied only to the actual payload field within the MAC payload, i.e., the beacon payload field (see 7.2.2.1.8), command payload field (see 7.2.2.4.3), or data payload field (see 7.2.2.2.2), depending on the frame type. The corresponding payload field shall be passed to the CCM* inverse transformation process described in 7.6.3.5 as the secure payload.

    ii) The remaining fields in the MAC payload part of the frame shall be passed to the CCM* inverse transformation process described in 7.6.3.5 as the non-payload fields (see Table 101).

    iii) The ordering and exact manner of performing the decryption and integrity checking operations and the placement of the resulting decrypted data within the MAC payload field shall be as defined in 7.6.3.5.

r) If the CCM* inverse transformation process fails, the procedure shall set the unsecured frame to be the frame to be unsecured and return with the unsecured frame, the security level, the key identifier mode, the key source, the key index and a status of SECURITY_ERROR.

s) The procedure shall increment the frame counter by one and set the FrameCounter element of the DeviceDescriptor to the resulting value.

t) If the FrameCounter element is equal to 0xffffffff, the procedure shall set the Blacklisted element of the KeyDeviceDescriptor to TRUE.

u) The procedure shall return with the unsecured frame, the security level, the key identifier mode, the key source, the key index and a status of SUCCESS.

7.5.8.2.4 Incoming frame key retrieval procedure

The input to this procedure is the frame to be unsecured. The outputs from this procedure are a passed or failed status and, if passed, a KeyDescriptor.

The incoming frame key retrieval procedure involves the following steps:

a) If the key identifier mode subfield of the security control field of the auxiliary security header of the frame is set to 0x00 (implicit key identification), the procedure shall determine the key source lookup data and the key source lookup size as follows:

   i) If the source address mode of the frame control field of the frame is set to 0x00 and the *macPANCoordShortAddress* attribute is set to a value in the range 0x0000-0xfffd (i.e., the short address is used), the key source lookup data shall be set to the 2-octet destination PAN ID field of the frame right-concatenated (see B.1.1) with the 2-octet *macPANCoord-ShortAddress* attribute. The key source lookup size shall be set to four.

   ii) If the source address mode of the frame control field of the frame is set to 0x00 and the *macPANCoordShortAddress* attribute is set to 0xfffe (i.e., the extended address is used), the key source lookup data shall be set to the 8-octet *macPANCoordExtendedAddress* attribute. The key source lookup size shall be set to eight.

   iii) If the source address mode of the frame control field of the frame is set to 0x02, the key source lookup data shall be set to the 2-octet source PAN ID field of the frame, or to the 2-octet destination PAN ID field of the frame if the PAN ID compression subfield of the frame control field of the frame is set to one, right-concatenated (see B.1.1) with the 2-octet source address field of the frame. The key source lookup size shall be set to four.

   iv) If the source address mode of the frame control field of the frame is set to 0x03, the key source lookup data shall be set to the 8-octet source address field of the frame. The key source lookup size shall be set to eight.

   The key index shall be set to the single octet 0x00.

b) If the key identifier mode subfield of the security control field of the auxiliary security header of the frame is set to a value unequal to 0x00 (explicit key identification), the procedure shall determine the key source lookup data and key source lookup size as follows:

   i) If the key identifier mode is set to 0x01, the key source lookup data shall be set to the 8-octet *macDefaultKeySource* attribute. The key source lookup size shall be set to eight.

   ii) If the key identifier mode is set to 0x02, the key source lookup data shall be set to the 4-octet key source subfield of the key identifier field of the auxiliary security header. The key source lookup size shall be set to four.

   iii) If the key identifier mode is set to 0x03, the key source lookup data shall be set to the 8-octet key source subfield of the key identifier field of the auxiliary security header. The key source lookup size shall be set to eight.

   The key index shall be set to the 1-octet key index subfield of the key identifier field of the auxiliary security header.

c) The procedure shall obtain the KeySourceDescriptor by passing the key source lookup data and the key source lookup size to the KeySourceDescriptor lookup procedure as described in 7.5.8.2.9. If that procedure returns with a failed status, this procedure shall also return with a failed status.
d) The procedure shall obtain the KeyDescriptor by passing the KeySourceDescriptor and key index to the KeyDescriptor lookup procedure as described in 7.5.8.2.6. If that procedure returns with a failed status, this procedure shall also return with a failed status.
e) The procedure shall return with a passed status having obtained the KeyDescriptor.

7.5.8.2.5 Incoming frame device retrieval procedure

The input to this procedure is the frame to be unsecured. The outputs from this procedure are a passed or failed status and, if passed, a DeviceDescriptor.

The incoming frame device retrieval procedure involves the following steps:

a) The procedure shall determine the device lookup data and the device lookup size as follows:
   i) If the source address mode of the frame control field of the frame is set to 0x00 and the *macPANCoordShortAddress* attribute is set to a value in the range 0x0000-0xfffd (i.e., the short address is used), the device lookup data shall be set to the 2-octet destination PAN ID field of the frame right-concatenated (see B.1.1) with the 2-octet *macPANCoordShortAddress* attribute. The device lookup size shall be set to four.
   ii) If the source address mode of the frame control field of the frame is set to 0x00 and the *macPANCoordShortAddress* attribute is set to a 0xfffe (i.e., the extended address is used), the device lookup data shall be set to the 8-octet *macPANCoordExtendedAddress* attribute. The device lookup size shall be set to eight.
   iii) If the source address mode of the frame control field of the frame is set to 0x02, the device lookup data shall be set to the 2-octet source PAN ID field of the frame, or to the 2-octet destination PAN ID field of the frame if the PAN ID compression subfield of the frame control field of the frame is set to one, right-concatenated (see B.1.1) with the 2-octet source address field of the frame. The device lookup size shall be set to four.
   iv) If the source address mode of the frame control field of the frame is set to 0x03, the device lookup data shall be set to the 8-octet source address field of the frame. The device lookup size shall be set to eight.
b) The procedure shall obtain the DeviceDescriptor by passing the device lookup data and the device lookup size to the DeviceDescriptor lookup procedure as described in 7.5.8.2.8. If that procedure returns with a failed status, this procedure shall also return with a failed status.
c) The procedure shall return with a passed status having obtained the DeviceDescriptor.

7.5.8.2.6 KeyDescriptor lookup procedure

The inputs to this procedure are the KeySourceDescriptor and the key index. The outputs from this procedure are a passed or failed status and, if passed, a KeyDescriptor.

The KeyDescriptor lookup procedure involves the following steps:

a) For each KeyDescriptor in the *macKeyTable* attribute, the procedure shall check whether the ExtKeySource element of the KeyDescriptor is equal to the corresponding element of the KeySourceDescriptor and whether the KeyIndex element of the KeyDescriptor is equal to the key index parameter. If both checks pass (i.e., there is a match), the procedure shall return with this (matching) KeyDescriptor and a passed status.
b) The procedure shall return with a failed status.

7.5.8.2.7 KeyDeviceDescriptor lookup procedure

The inputs to this procedure are the KeyDescriptor and the DeviceDescriptor. The outputs from this procedure are a passed or failed status and, if passed, a KeyDeviceDescriptor.

The KeyDeviceDescriptor lookup procedure involves the following steps:

a) For each KeyDeviceDescriptor in the KeyDeviceList of the KeyDescriptor, the procedure shall check whether the ExtAddress element of the DeviceDescriptor is equal to the DeviceAddress element of the KeyDeviceDescriptor. If this check passes (i.e., there is a match), the procedure shall return with the KeyDeviceDescriptor and a passed status.
b) The procedure shall return with a failed status.

7.5.8.2.8 DeviceDescriptor lookup procedure

The inputs to this procedure are the device lookup data and the device lookup size. The output from this procedure are a passed or failed status and, if passed, a DeviceDescriptor.

The DeviceDescriptor lookup procedure involves the following steps:

a) For each DeviceDescriptor in the *macDeviceTable* attribute:
    i) If the device lookup size is four and the device lookup data is equal to the PAN ID element of the DeviceDescriptor right-concatenated (see B.1.1) with the ShortAddress element of the DeviceDescriptor (i.e., there is a match), this procedure shall return with the DeviceDescriptor and a passed status.
    ii) If the device lookup size is eight and the device lookup data is equal to the ExtAddress element of the DeviceDescriptor (i.e., there is a match), this procedure shall return with the DeviceDescriptor and a passed status.
b) The procedure shall return with a failed status.

7.5.8.2.9 KeySourceDescriptor lookup procedure

The inputs to this procedure are the key source lookup data and the key source lookup size. The output from this procedure are a passed or failed status and, if passed, a KeySourceDescriptor.

The KeySourceDescriptor lookup procedure involves the following steps:

a) For each KeySourceDescriptor in the *macKeySourceTable* attribute:
    i) If the key source lookup size is four and the key source lookup data is equal to the ShortKeySource element of the KeySourceDescriptor (i.e., there is a match), this procedure shall return with the KeySourceDescriptor and a passed status.
    ii) If the key source lookup size is eight and the key source lookup data is equal to the ExtKeySource element of the KeySourceDescriptor (i.e., there is a match), this procedure shall return with the KeySourceDescriptor and a passed status.
b) The procedure shall return with a failed status.

7.5.8.2.10 SecurityLevelDescriptor lookup procedure

Adapt so as to accommodate frame identifiers for other frame types, such as, e.g,, acknowledgement frame type. Incorporate similar change with

The inputs to this procedure are the frame type and the command frame identifier. The output from this procedure are a passed or failed status and, if passed, a SecurityLevelDescriptor.

The SecurityLevelDescriptor lookup procedure involves the following steps:

a) For each SecurityLevelDescriptor in the *macSecurityLevelTable* attribute:
   i) If the frame type is not equal to 0x03 and the frame type is equal to the FrameType element of the SecurityLevelDescriptor (i.e., there is a match), the procedure shall return with the SecurityLevelDescriptor and a passed status.
   ii) If the frame type is equal to 0x03, the frame type is equal to the FrameType element of the SecurityLevelDescriptor and the command frame identifier is equal to the CommandFrameIdentifier element of the SecurityLevelDescriptor, the procedure shall return with the SecurityLevelDescriptor and a passed status.
b) The procedure shall return with a failed status.

7.5.8.2.11 Incoming security level checking procedure

The inputs to this procedure are the SecurityLevelDescriptor and the incoming security level. The output from this procedure is a passed, failed, or 'conditionally passed' status.

The incoming security level checking procedure involves the following steps:

a) For each SecurityModeDescriptor in the SecurityLevelList of the SecurityLevelDescriptor, the procedure shall check whether the incoming security level is equal to the SecurityLevel element of the SecurityModeDescriptor. If this check is successful (i.e., there is a match), the procedure shall return with a passed status.
b) If the incoming security level is equal to 0x00 and the DeviceOverrideSecurityMinimum element of the SecurityLevelDescriptor is set to TRUE, the procedure shall return with a 'conditionally passed' status.
c) The procedure shall return with a failed status.

7.5.8.2.12 Incoming key usage policy checking procedure

The inputs to this procedure are the KeyDescriptor, the frame type and the command frame identifier. The output from this procedure is a passed or failed status.

The incoming key usage policy checking procedure involves the following steps:

a) For each KeyUsageDescriptor in the KeyUsageList of the KeyDescriptor:
   i) If the frame type is not equal to 0x03 and the frame type is equal to the FrameType element of the KeyUsageDescriptor, the procedure shall return with a passed status.
   ii) If the frame type is equal to 0x03, the frame type is equal to the FrameType element of the KeyUsageDescriptor and the command frame identifier is equal to the CommandFrameIdentifier element of the KeyUsageDescriptor, the procedure shall return with a passed status.
b) The procedure shall return with a failed status.

7.6 Security suite specifications

7.6.1 PIB security material

The PIB security-related attributes are presented in Table 88, Table 89, Table 90, Table 91, Table 92, Table 93, Table 94, and Table 95.

7.6.2 Auxiliary security header

The auxiliary security header field has a variable length and contains information required for security processing, including a security control field, a frame counter field, and a key identifier field. The auxiliary

Table 88— Security-related MAC PIB attributes

Attribute	Identifier	Type	Range	Description	Default
macKeyTable	0x71	List of KeyDescriptor entries (see Table 89)	–	A table of KeyDescriptor entries, each containing keys and related information required for secured communications.	(empty)
macKeyTableEntries	0x72	Integer	Implementation specific	The number of entries in *macKeyTable*.	0
macDeviceTable	0x73	List of DeviceDescriptor entries (see Table 93)	–	A table of DeviceDescriptor entries, each indicating a remote device with which this device securely communicates.	(empty)
macDeviceTableEntries	0x74	Integer	Implementation specific	The number of entries in *macDeviceTable*.	0
macSecurityLevelTable	0x75	Table of SecurityLevelDescriptor entries (see Table 92)	–	A table of SecurityLevelDescriptor entries, each with information as to the minimum security level expected depending on incoming frame type and subtype.	(empty)
macSecurityLevelTableEntries	0x76	Integer	Implementation specific	The number of entries in *macSecurityLevelTable*.	0
macFrameCounter	0x77	Integer	0x0000000000 –0xffffffffff	The outgoing frame counter for this device.	0
macAutoRequestSecurityLevel	0x78	Integer	0x00–0x07	The security level used for automatic data requests.	0x06
macAutoRequestFrameCounterMode	TBD	Integer	0x00-0x07	The frame counter mode used for automatic data requests. This attribute is invalid if the *macAutoRequestSecurityLevel* attribute is set to 0x00.	0x00
macAutoRequestKeyIdMode	0x79	Integer	0x00–0x03	The key identifier mode used for automatic data requests. This attribute is invalid if the *macAutoRequestSecurityLevel* attribute is set to 0x00.	0x00
macAutoRequestKeySource	0x7a	As specified by the *macAutoRequestKeyIdMode* parameter	–	The originator of the key used for automatic data requests. This attribute is invalid if the *macAutoRequestKeyIdMode* element is invalid or set to 0x00.	All octets 0xff

Table 88— Security-related MAC PIB attributes *(continued)*

Attribute	Identifier	Type	Range	Description	Default
macAutoRequestKeyIndex	0x7b	Integer	0x01-0xff	The index of the key used for automatic data requests. This attribute is invalid if the *macAutoRequestKeyIdMode* attribute is invalid or set to 0x00.	All octets 0xff
macDefaultKeySource	0x7c	Set of 8 octets	–	The identifier of the default key source used for key identifier mode 0x01.	All octets 0xff
macPANCoordExtendedAddress	0x7d	IEEE address	An extended 64-bit IEEE address	The 64-bit address of the PAN coordinator.	—
macPANCoordShortAddress	0x7e	Integer	0x0000–0xffff	The 16-bit short address assigned to the PAN coordinator. A value of 0xfffe indicates that the PAN coordinator is only using its 64-bit extended address. A value of 0xffff indicates that this value is unknown.	0x0000
macKeySourceTable	0x7f	List of KeySourceDescriptor entries (see Table 94)	–	A table of KeySourceDescriptor entries, each indicating key identifying information required for secured communications.	(empty)
macKeySourceTableEntries	0x80	Integer	Implementation specific	The number of entries in *macKeySourceTable*.	0
macCurrentTime	TBD	Integer	0x000000000000-0xffffffffffff	The most recent absolute device time, as measured just prior to perfoming an incoming or outgoing frame security procedure, in 16kHz accuracy	0

Table 89—Elements of KeyDescriptor

Name	Type	Range	Description
ExtKeySource	Set of 8 octets	–	The 64-bit identifier of the key source (see 7.6.2.4.1)
KeyIndex	Integer	0x00-0xff	The identifier of the key index (see 7.6.2.4.2). A value of 0x00 indicates an implicitly identified key; a value in the range 0x01-0xff indicates an explicitly identified key.
KeyOffset	Integer	0x000000000000-0xffffffffffff	The start time counter value for this key, prior to which this key shall not be used. A value of 0 indicates that there are no time restrictions as to when to start using this key.

Table 89—Elements of KeyDescriptor *(continued)*

Name	Type	Range	Description
Blacklisted	Boolean	TRUE or FALSE	Indicator as to whether the device previously communicated with this key prior to the exhaustion of the frame counter. If TRUE, this indicates that the device shall not use this key further, since it exhausted its use of the frame counter used with this key.
KeyDeviceList	List of KeyDeviceDescriptor entries (see Table 91)	–	A list of KeyDeviceDescriptor entries indicating which devices are currently using this key, including their blacklist status.
KeyDeviceListEntries	Integer	Implementation specific	The number of entries in KeyDeviceList.
KeyUsageList	List of KeyUsageDescriptor entries (see Table 90)	–	A list of KeyUsageDescriptor entries indicating which frame types this key may be used with.
KeyUsageListEntries	Integer	Implementation specific	The number of entries in KeyUsageList.
Key	Set of 16 octets	–	The actual value of the key.

Table 90—Elements of KeyUsageDescriptor

Name	Type	Range	Description
FrameType	Integer	0x00–0x03	See 7.2.1.1.1
CommandFrameIdentifier	Integer	0x00–0x09	See Table 82

Table 91—Elements of KeyDeviceDescriptor

Name	Type	Range	Description
DeviceAddress	IEEE address	Any valid 64-bit device address	The 64-bit IEEE extended address of the device in this KeyDeviceDescriptor.
Blacklisted	Boolean	TRUE or FALSE	Indicator as to whether the device indicated by DeviceAddress previously communicated with this key prior to the exhaustion of the frame counter. If TRUE, this indicates that the device shall not use this key further, since it exhausted its use of the frame counter used with this key.

security header field shall only be present if the security enabled subfield of the frame control field is set to one. The auxiliary security header field shall be formatted as illustrated in Figure 75.

Table 92—Elements of SecurityLevelDescriptor

Name	Type	Range	Description
FrameType	Integer	0x00–0x03	See 7.2.1.1.1
CommandFrameIdentifier	Integer	0x00–0x09	See Table 82
Acknowledgement-FrameIdentifier	Integer	0x00-TBD	TBD
SecurityLevelList	List of SecurityModeDescriptor entries (see Table 95)	–	A list of SecurityModeDescriptor entries indicating the security levels incoming MAC frames with the indicated frame type and, if present, command frame type or acknowledgement frame type are expected to be be secured with.
SecurityLevelListEntries	Integer	Implementation specific	The number of entries in SecurityLevelList.
DeviceOverrideSecurityMinimum	Boolean	TRUE or FALSE	Indicator as to whether originating devices for which the Exempt flag is set may override the required/expected security levels indicated by the SecurityLevelList element. If TRUE, this indicates that for originating devices with Exempt status, the incoming security level zero is acceptable, in addition to those incoming security levels indicated by the SecurityLevelList element.

Table 93—Elements of DeviceDescriptor

Name	Type	Range	Description
PANId	Device PAN ID	0x0000–0xffff	The 16-bit PAN identifier of the device in this DeviceDescriptor.
ShortAddress	Device short address	0x0000–0xffff	The 16-bit short address of the device in this DeviceDescriptor. A value of 0xfffe indicates that this device is only using its extended address. A value of 0xffff indicates that this value is unknown.
ExtAddress	IEEE address	Any valid 64-bit device address	The 64-bit IEEE extended address of the device in this DeviceDescriptor. This element is also used in unsecuring operations on incoming frames.
FrameCounter	Integer	0x000000000000–0xffffffffffff	The incoming frame counter of the device in this DeviceDescriptor. This value is used to ensure sequential freshness of frames.
Exempt	Boolean	TRUE or FALSE	Indicator as to whether the device may override the minimum security level settings defined in Table 92.

Table 94—Elements of KeySourceDescriptor

Name	Type	Range	Description
ExtKeySource	Set of 8 octets	–	The 64-bit identifier of the key source (see 7.6.2.4.1).
ShortKeySource	Set of 4 octets	–	The 32-bit identifier of the ExtKeySource in this KeySourceDescriptor. A value of 0xfffffffe indicates that only ExtKeySource is used. A value of 0xffffffff indicates that this value is unknown.

Table 95—Elements of SecurityModeDescriptor

Name	Type	Range	Description
SecurityLevel	Integer	0x00–0x07	Security level identifier (see Table 96).

Octets: 1	4	0/1/5/9
Security control	Frame counter	Key identifier

Figure 75—Format of the auxiliary security header

7.6.2.1 Integer and octet representation

The auxiliary security header is a MAC frame field (see 7.2.1.7) and, therefore, uses the representation conventions specified in 7.2.

7.6.2.2 Security control field

The security control field is 1 octet in length and is used to provide information as to what protection is applied to the frame. The security control field shall be formatted as shown in Figure 76.

Bit: 0-2	3-4	5-7
Security level	Key identifier mode	Frame counter mode

Figure 76—Security control field format

7.6.2.2.1 Security level subfield

The security level subfield is 3 bits in length and indicates the actual frame protection that is provided. This value can be adapted on a frame-by-frame basis and allows for varying levels of data authenticity (to allow minimization of security overhead in transmitted frames where required) and for optional data confidentiality. The cryptographic protection offered by the various security levels is shown in Table 96. When nontrivial protection is required, replay protection is always provided.

Table 96—Security levels available to the MAC sublayer

Security level identifier	Security control field (Figure 76) b2 b1 b0	Security attributes	Data confidentiality	Data authenticity (including length M of authentication tag, in octets)
0x00	'000'	None	OFF	NO (M = 0)
0x01	'001'	MIC-32	OFF	YES (M=4)
0x02	'010'	MIC-64	OFF	YES (M=8)
0x03	'011'	MIC-128	OFF	YES (M=16)
0x04	'100'	ENC	ON	NO (M = 0)
0x05	'101'	ENC-MIC-32	ON	YES (M=4)
0x06	'110'	ENC-MIC-64	ON	YES (M=8)
0x07	'111'	ENC-MIC-128	ON	YES (M=16)

7.6.2.2.2 Key identifier mode subfield

The key identifier mode subfield is 2 bits in length and indicates whether or not the key that is used to protect the frame can be derived implicitly or explicitly; furthermore, it is used to indicate the particular representations of the key identifier field (see 7.6.2.4) if derived explicitly. The key identifier mode subfield shall be set to one of the values listed in Table 97. The key identifier field of the auxiliary security header (see 7.6.2.4) shall only be present if this subfield has a value that is not equal to 0x00.

Table 97—Values of the key identifier mode

Key identifier mode	Key identifier mode subfield b1 b0	Description	Key identifier field length (octets)
0x00	‘00’	Key is determined implicitly from the originator and receipient(s) of the frame, as indicated in the frame header.	0
0x01	‘01’	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
0x02	‘10’	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
0x03	‘11’	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

7.6.2.2.3 Frame counter mode subfield

The frame counter mode subfield is 3 bits in length and is used to indicate the particular representation of the frame counter field (see 7.6.2.3). The frame counter mode subfield shall be set to one of the values listed in Table 97.

Table 98— Values of the frame counter mode

Frame counter mode	Frame counter mode subfield b2 b1 b0	Decription	Frame counter field length
0x00	‘000’	Frame counter as in 802.15.4-2006, without correlation frame counter and DSN entry.	4
0x01	‘001’	Full frame counter, with least significant octet in DSN entry.	5
0x02	‘010’	Compressed frame counter, with least significant octet in DSN entry.	1
0x03	‘011’	Compressed frame counter, with least significant octet in DSN entry.	0
0x04-0x07	‘100’-‘111’	reserved	–

7.6.2.3 Frame counter field

The frame counter field is 4 octets in length and represents the *macFrameCounter* attribute of the originator of a protected frame. It is used to provide semantic security of the cryptographic mechanism used to protect a frame and to offer replay protection.

7.6.2.4 Key identifier field

The key identifier field has a variable length and identifies the key that is used for cryptographic protection of outgoing frames, either explicitly or in conjunction with implicitly defined side information. The key identifier field shall only be present if the key identifier mode subfield of the security control field of the auxiliary security header (see 7.6.2.2.2) is set to a value different from 0x00. The key identifier field shall be formatted as illustrated in Figure 77.

Octets: 0/4/8	1
Key source	Key index

Figure 77—Format for the key identifier field, if present

7.6.2.4.1 Key source subfield

The key source subfield, when present, is either 4 octets or 8 octets in length, according to the value specified by the key identifier mode subfield of the security control field (see 7.6.2.2.2), and indicates the originator of a group key.

7.6.2.4.2 Key index subfield

The key index subfield is 1 octet in length and allows unique identification of different keys with the same originator.

It is the responsibility of each key originator to make sure that actively used keys that it issues have distinct key indices and that the key indices are all different from 0x00.

7.6.3 Security operations

This clause describes the parameters for the CCM* security operations, as specified in B.2.2.

7.6.3.1 Integer and octet representation

This clause uses the representation conventions specified in B.1.

7.6.3.2 CCM* Nonce

The CCM* nonce is a 13-octet string and is used for the AES-CCM* mode of operation (see B2.2). The nonce shall be formatted as shown in Figure 78, with the leftmost field in the figure defining the first (and leftmost) octets and the rightmost field defining the last (and rightmost) octet of the nonce.

Octets: 8	**4**	**1**
Source address	Frame counter	Security Field

Figure 78—CCM* nonce

Bits: 0-3	**4**	**5-7**
Ext. Frame Counter	Set to 0	Security level

Figure 79—Security Field

The source address field shall be set to the extended address *aExtendedAddress* of the device originating the frame. The frame counter field and the ext. frame counter field shall be set so that their right-concatenation, as bit-strings, has the same integer value as the scaled frame counter. The security level field shall be set to the security level.

The source address, frame counter and security level shall be represented as specified in 7.6.3.1.

7.6.3.3 CCM* prerequisites

Securing a frame involves the use of the CCM* mode encryption and authentication transformation, as described in B.3.1. Unsecuring a frame involves the use of the CCM* decryption and authentication checking process, as described in B.3.2. The prerequisites for the CCM* forward and inverse transformations are as follows:

— The underlying block cipher shall be the AES encryption algorithm as specified in B.2.1.
— The bit ordering shall be as defined in 7.6.3.1.

— The length in octets of the length field L shall be 2 octets.
— The length of the authentication field M shall be 0 octets, 4 octets, 8 octets, or 16 octets as required.

7.6.3.3.1 Authentication field length

The length of the authentication field M for the CCM* forward transformation and the CCM* inverse transformation is determined from the security level, using Table 96.

7.6.3.4 CCM* transformation data representation

This clause describes how the inputs and output of the CCM* forward transformation, as described in B.3.1, are formed:

The inputs are

— Key
— Nonce
— *a* data
— *m* data

The output is

— *c* data

7.6.3.4.1 Key and nonce data inputs

The Key data for the CCM* forward transformation is passed by the outgoing frame security procedure described in 7.5.8.2.1. The Nonce data for the CCM* transformation is constructed as described in 7.6.3.2.

7.6.3.4.2 *a* data and *m* data

In the CCM* transformation process, the data fields shall be applied as in Table 99.

Table 99—*a* data and *m* data for all security levels

Security level identifier	*a* data	*m* data
0x00	None	None
0x01	MHR || Auxiliary security header || Non-payload fields || Unsecured payload fields	None
0x02	MHR || Auxiliary security header || Non-payload fields || Unsecured payload fields	None
0x03	MHR || Auxiliary security header || Non-payload fields || Unsecured payload fields	None
0x04	None	Unsecured Payload fields
0x05	MHR || Auxiliary security header || Non-payload fields	Unsecured Payload fields
0x06	MHR || Auxiliary security header || Non-payload fields	Unsecured Payload fields
0x07	MHR || Auxiliary security header || Non-payload fields	Unsecured Payload fields

7.6.3.4.3 *c* data output

In the CCM* transformation process, the data fields that are applied, or right-concatenated and applied, represent octet strings.

The secured payload fields right-concatenated with the authentication tag shall substitute the unsecured payload field in the original unsecured frame to form the secured frame (see Table 100).

Table 100—c data for all security levels

Security level identifier	c data
0x00	None
0x01	MIC-32
0x02	MIC-64
0x03	MIC-128
0x04	Secured Payload fields
0x05	Secured Payload fields || MIC-32
0x06	Secured Payload fields || MIC-64
0x07	Secured Payload fields || MIC-128

7.6.3.5 CCM* inverse transformation data representation

This clause describes how the inputs and output of the CCM* inverse transformation, as described in B.3.2, are formed.

The inputs are

— Key
— Nonce
— *c* data
— *a* data

The output is

— *m* data

7.6.3.5.1 Key and nonce data inputs

The Key data for the CCM* inverse transformation is passed by the incoming frame security procedure described in 7.5.8.2.3. The Nonce data for the CCM* transformation is constructed as described in 7.6.3.2.

7.6.3.5.2 *c* data and *a* data

In the CCM* inverse transformation process, the data fields shall be applied as in Table 101.

Table 101—*c* data and *a* data for all security levels

Security level identifier	*c* data	*a* data
0x00	None	None
0x01	MIC-32	MHR || Auxiliary security header || Non-payload fields ||Secured payload fields
0x02	MIC-64	MHR || Auxiliary security header || Non-payload fields || Secured payload fields
0x03	MIC-128	MHR || Auxiliary security header || Non-payload fields || Secured payload fields
0x04	Secured payload fields	MHR || Auxiliary security header || Non-payload fields
0x05	Secured payload fields || MIC-32	MHR || Auxiliary security header || Non-payload fields
0x06	Secured payload fields || MIC-64	MHR || Auxiliary security header || Non-payload fields
0x07	Secured payload fields || MIC-128	MHR || Auxiliary security header || Non-payload fields

7.6.3.5.3 *m* data output

The *m* data shall then substitute secured payload fields and authentication tag in the original secured frame to form the unsecured frame.

7.7 Message sequence charts illustrating MAC-PHY interaction

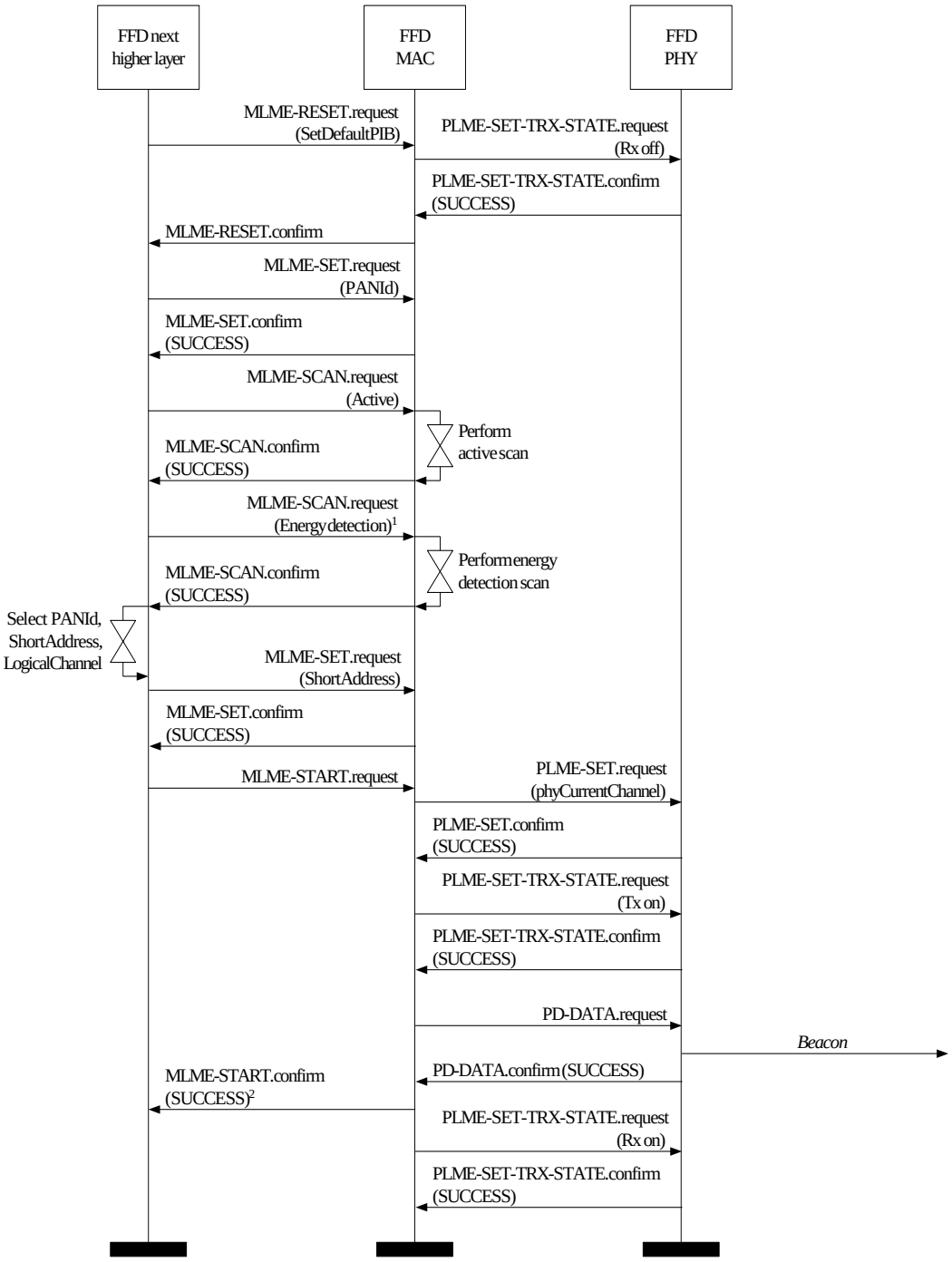
This subclause illustrates the main tasks specified in IEEE P802.15.4REVb/D6. Each task is described by use of a message sequence chart to illustrate the chronological order, rather than the exact timing, of the primitives required for each task.

The primitives necessary for the PAN coordinator to start a new PAN are shown in Figure 80. The first action the next higher layer takes after resetting the MAC sublayer is to initiate a scan to search for other PANs in the area. An active scan is required, and an ED scan may optionally be performed. The steps for performing an active scan and an ED scan are shown in Figure 85 and Figure 81, respectively.

Once a new PAN is established, the PAN coordinator is ready to accept requests from other devices to join the PAN. Figure 82 shows the primitives issued by a device requesting association, while Figure 83 illustrates the steps taken by a coordinator allowing association. In the process of joining a PAN, the device requesting association will perform either a passive or an active scan to determine which PANs in the area are allowing association; Figure 84 and Figure 85 detail the primitives necessary to complete a passive scan and an active scan, respectively.

The primitives necessary for transmitting and receiving a single data packet are shown next. The actions taken by the originator of the packet are shown in Figure 86, while the actions taken by the recipient are shown in Figure 87.

When a device becomes unable to communicate to its coordinator any longer, the device can use an orphan scan to rediscover its coordinator. The primitives necessary for the realignment of an orphaned device are shown in Figure 88.



[1] The energy detection scan is optional.
[2] The MLME-START.confirm and PLME-SET-TRX-STATE.request primitives may not need to be in the order shown.

Figure 80—PAN start message sequence chart—PAN coordinator

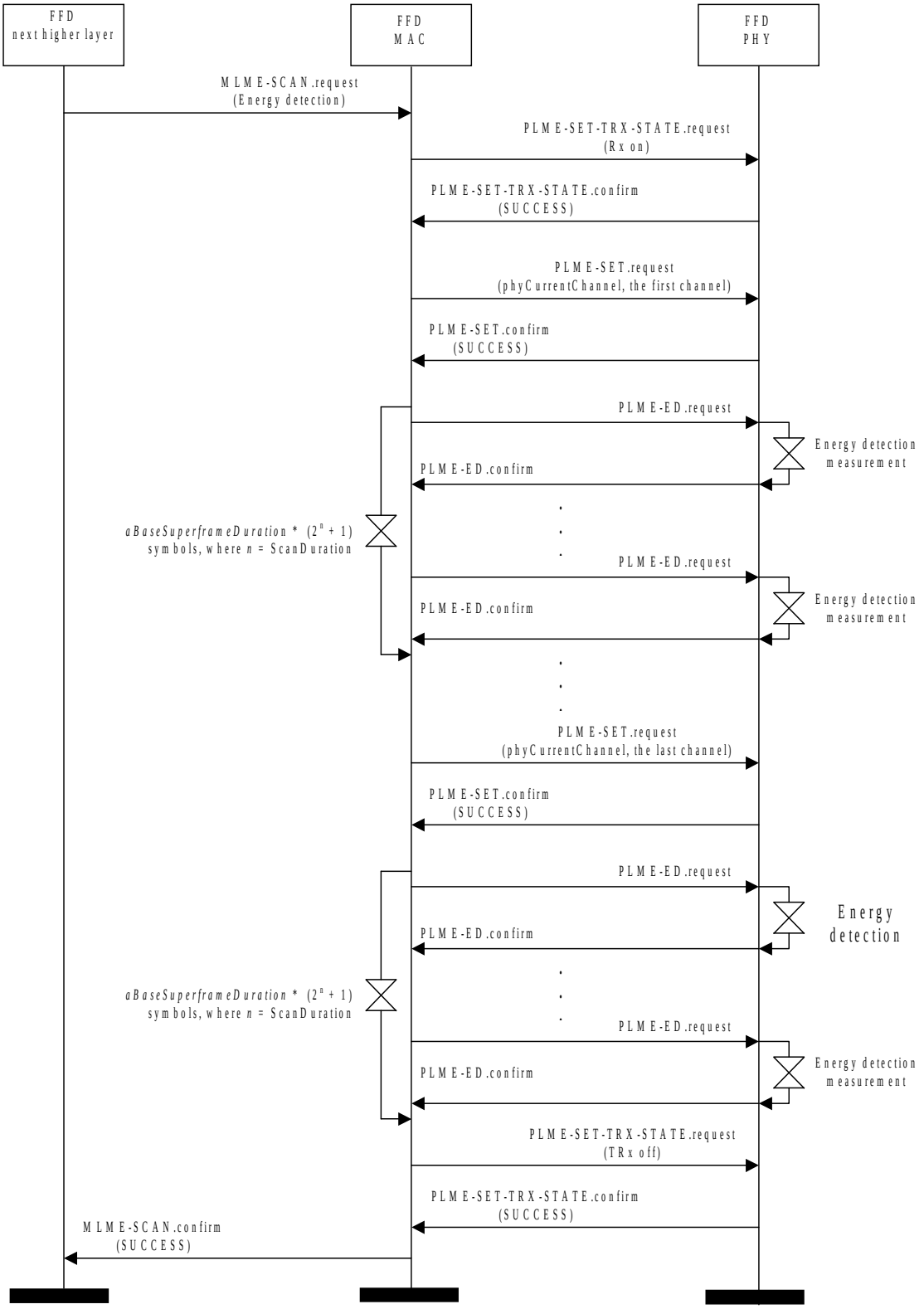


Figure 81—ED scan message sequence chart

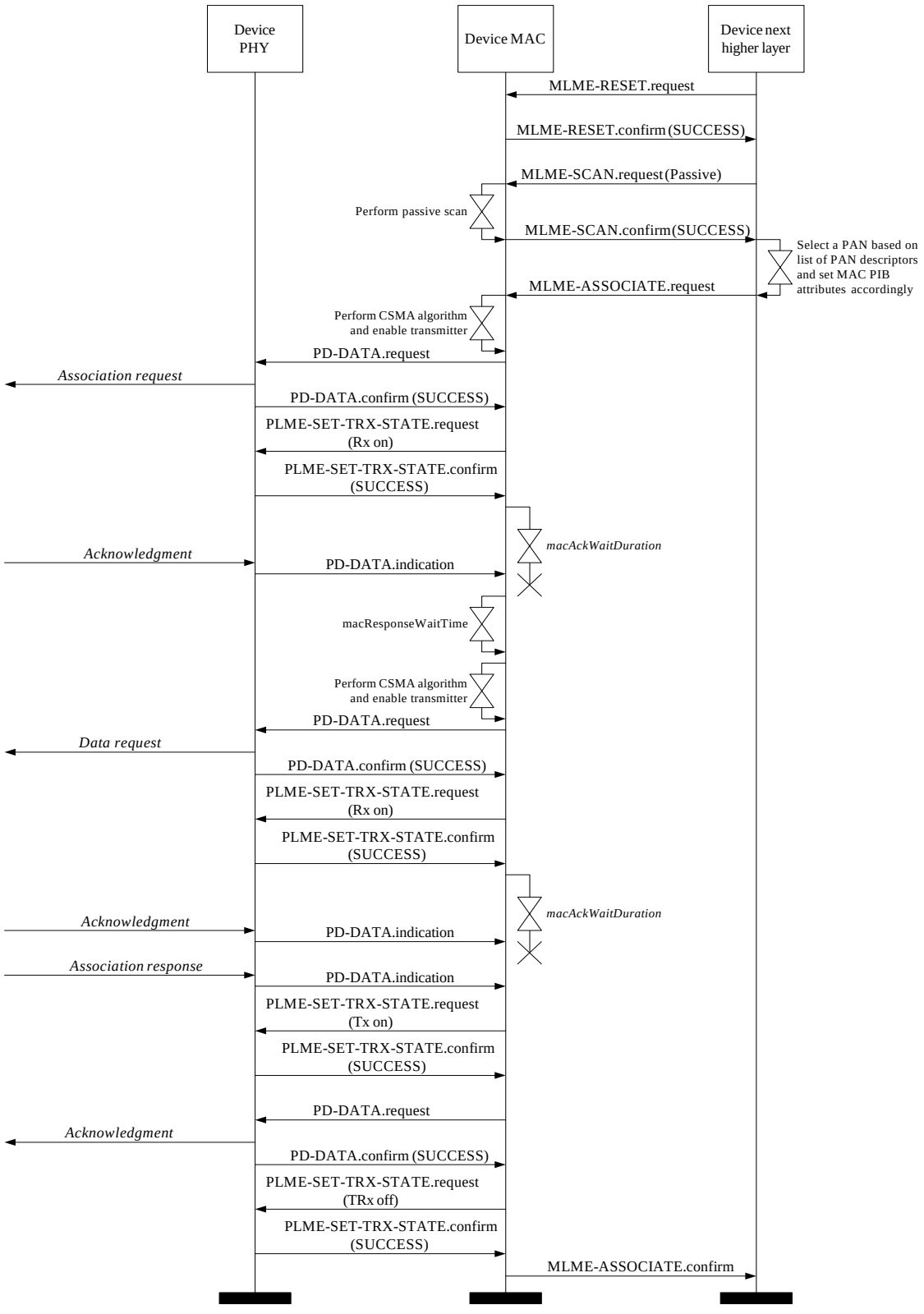


Figure 82—Association message sequence chart—device

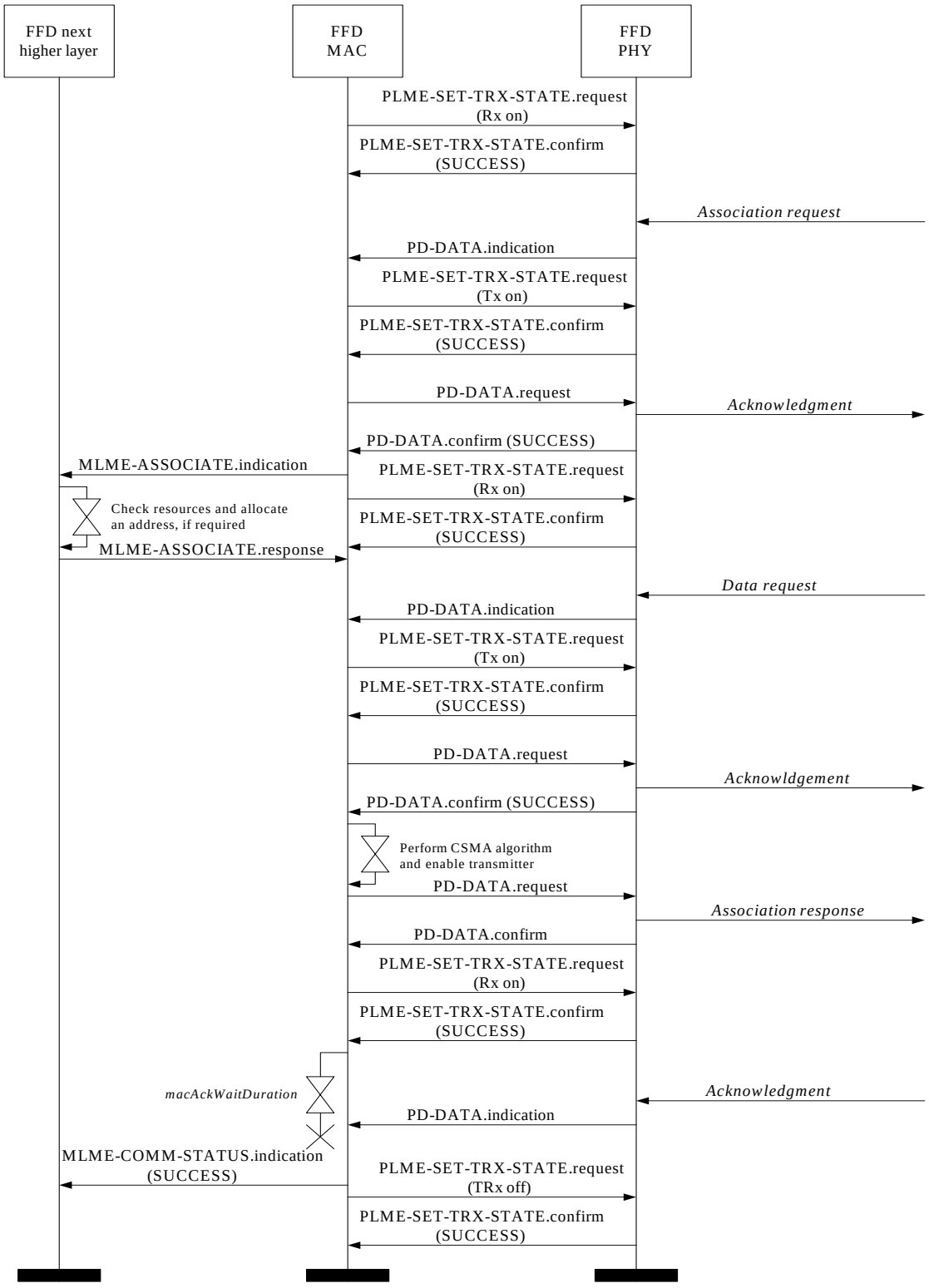


Figure 83—Association message sequence chart—coordinator

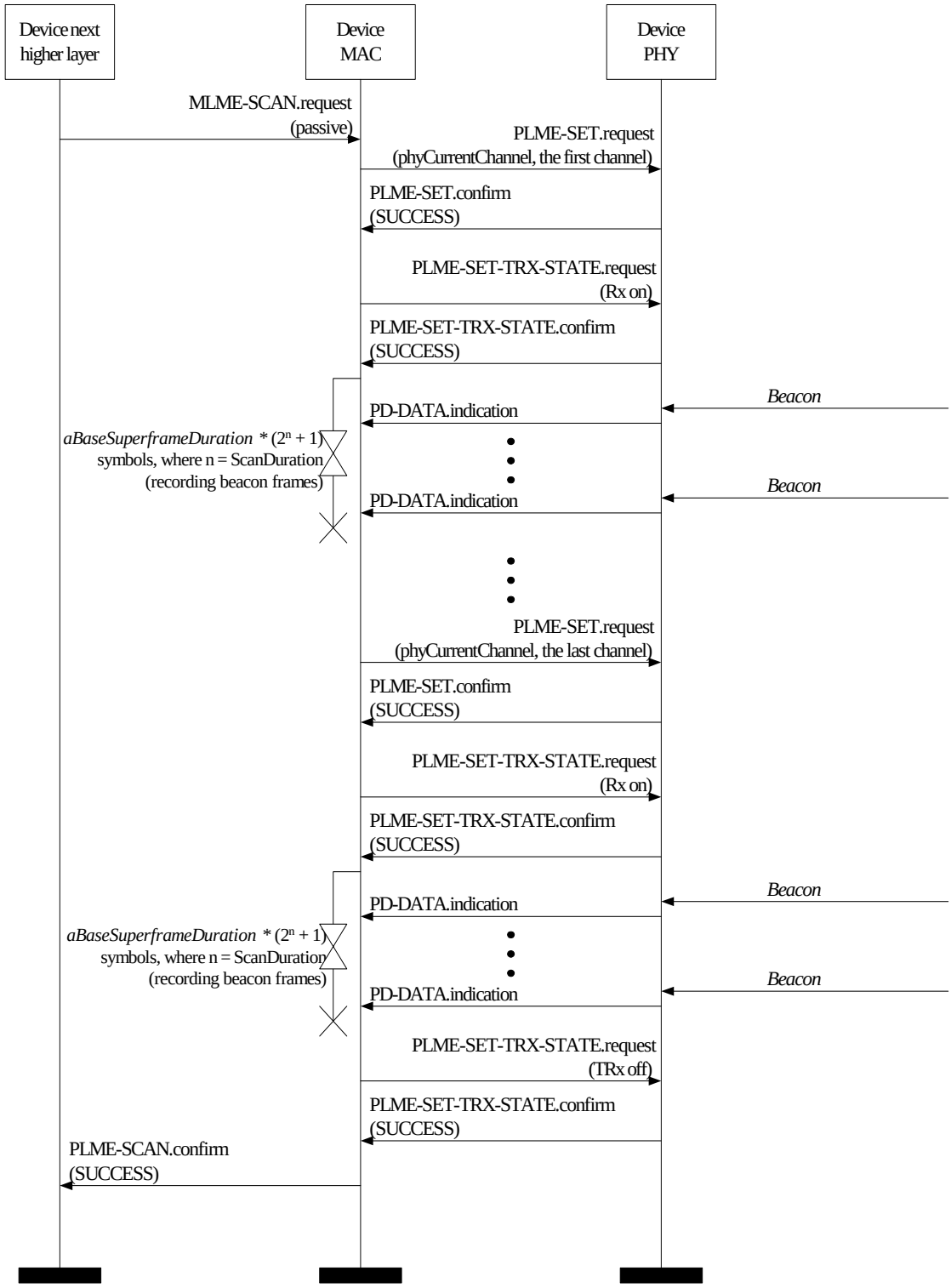


Figure 84—Passive scan message sequence chart

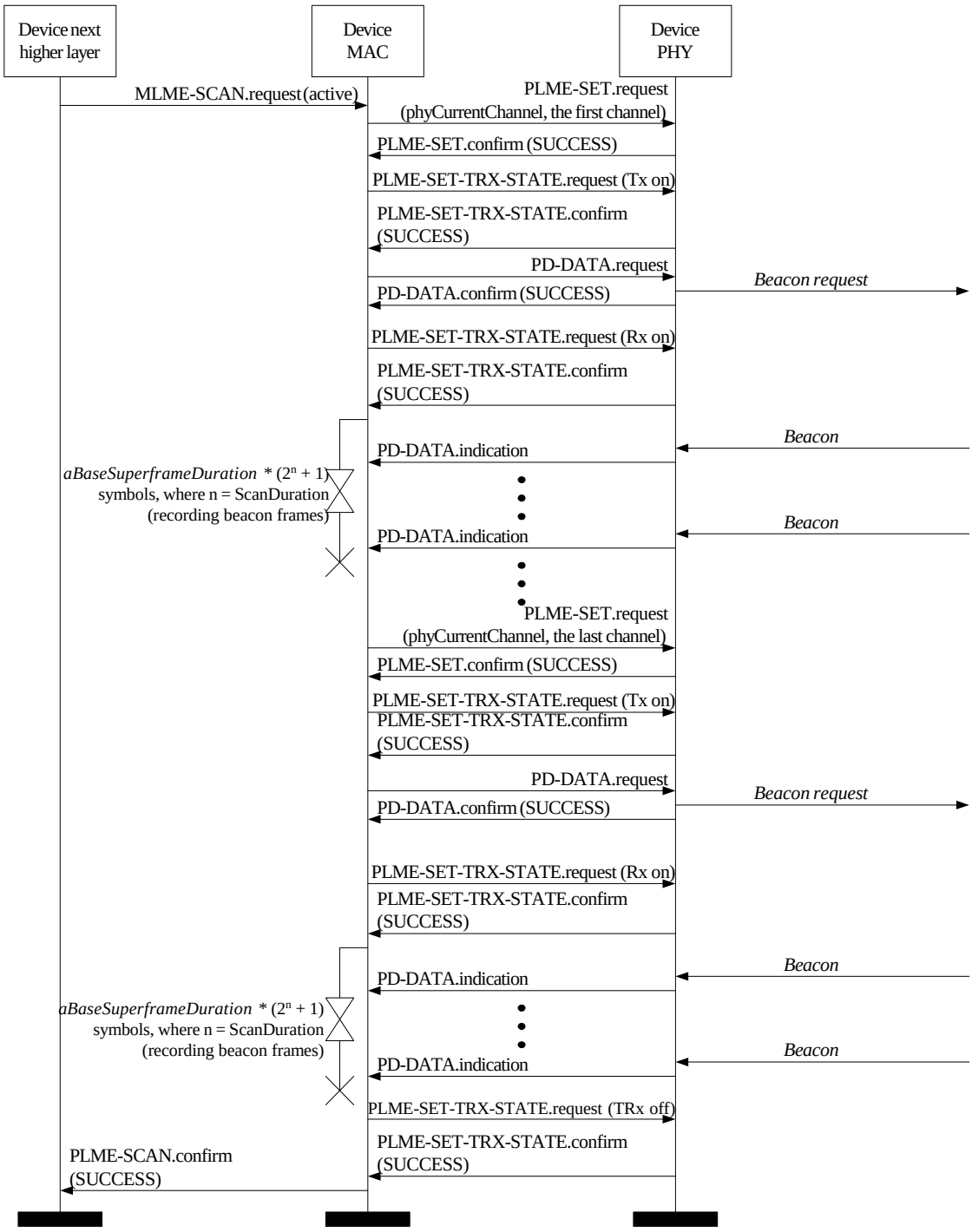


Figure 85—Active scan message sequence chart

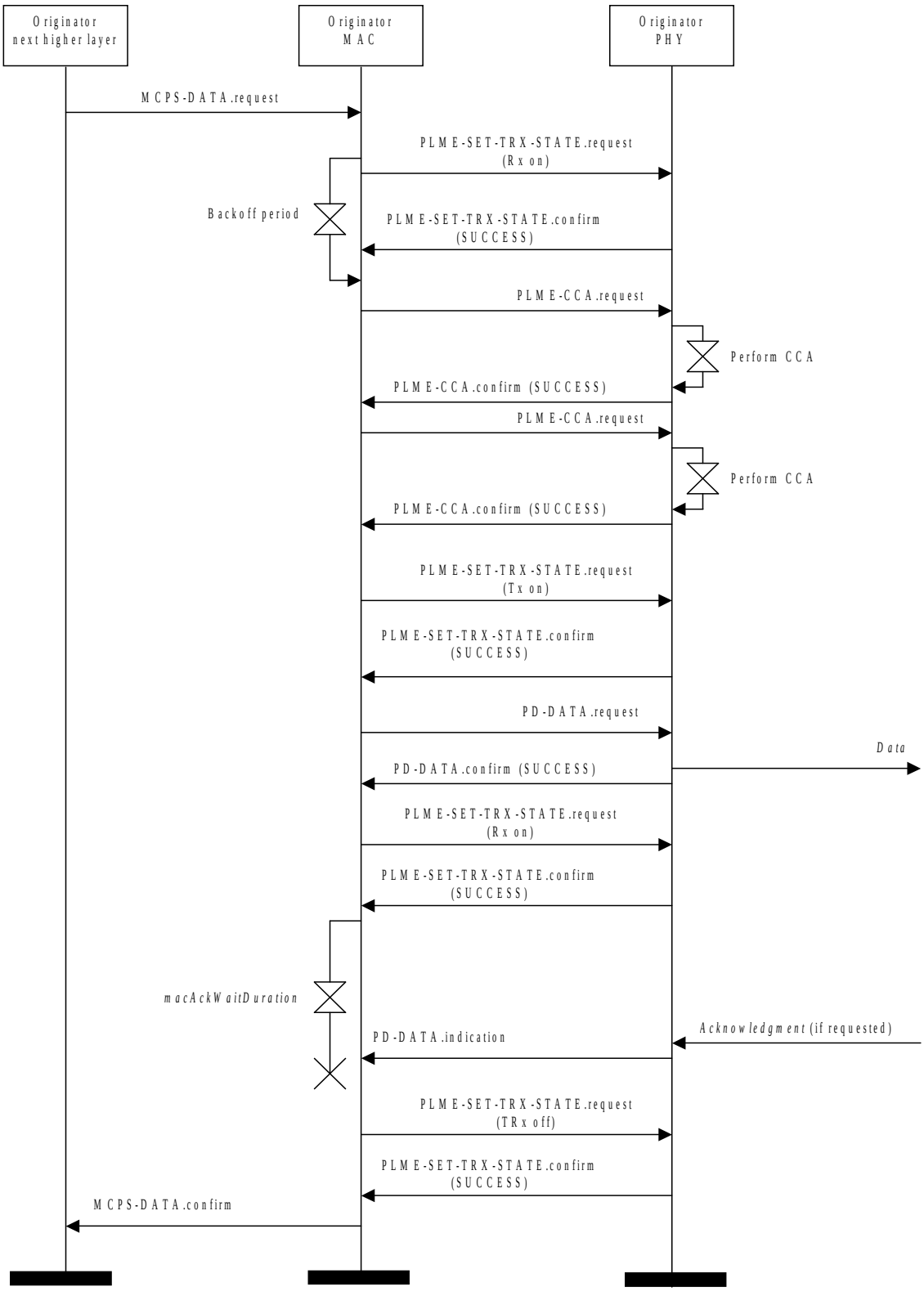


Figure 86—Data transmission message sequence chart—originator

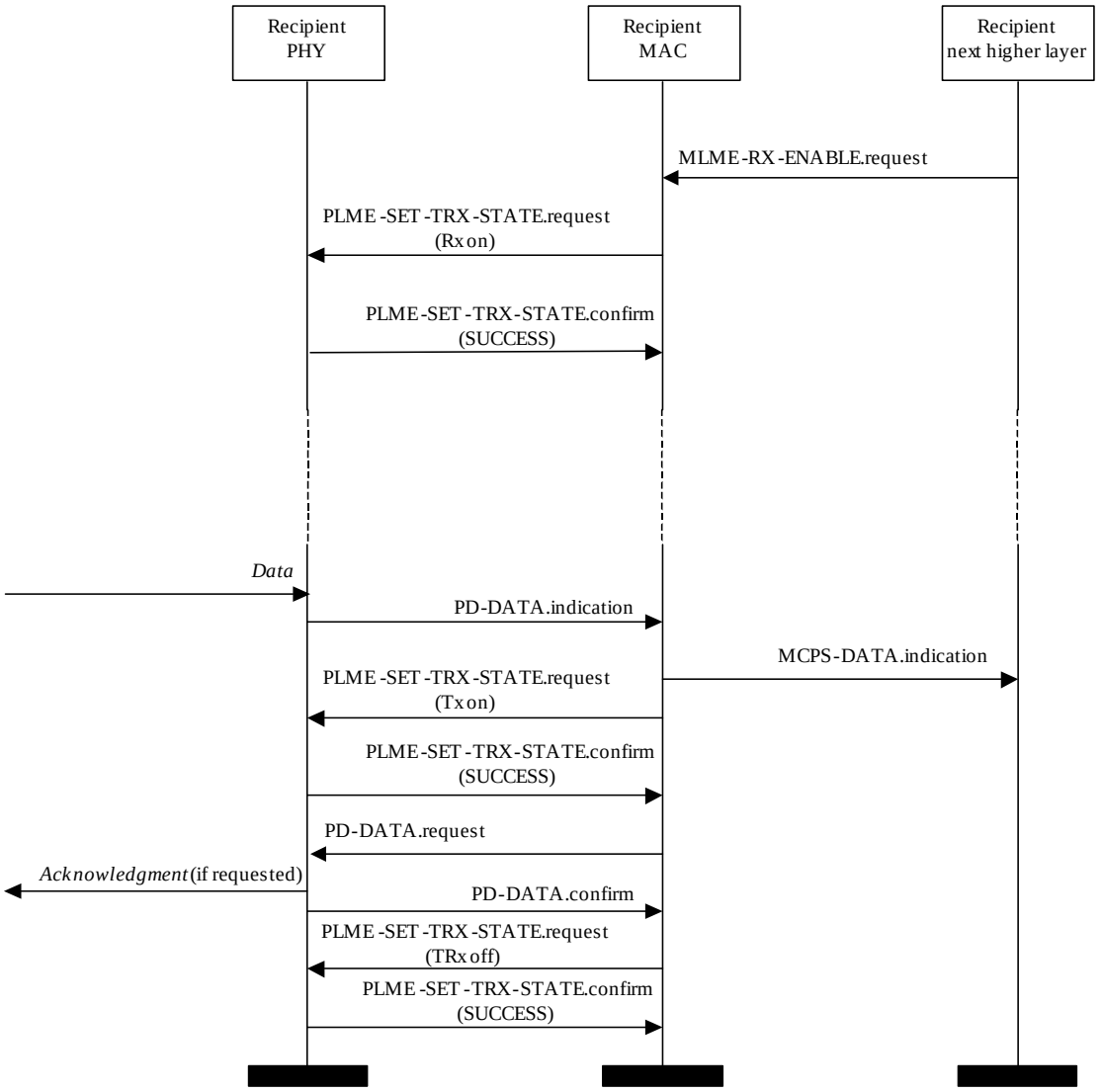


Figure 87—Data transmission message sequence chart—recipient

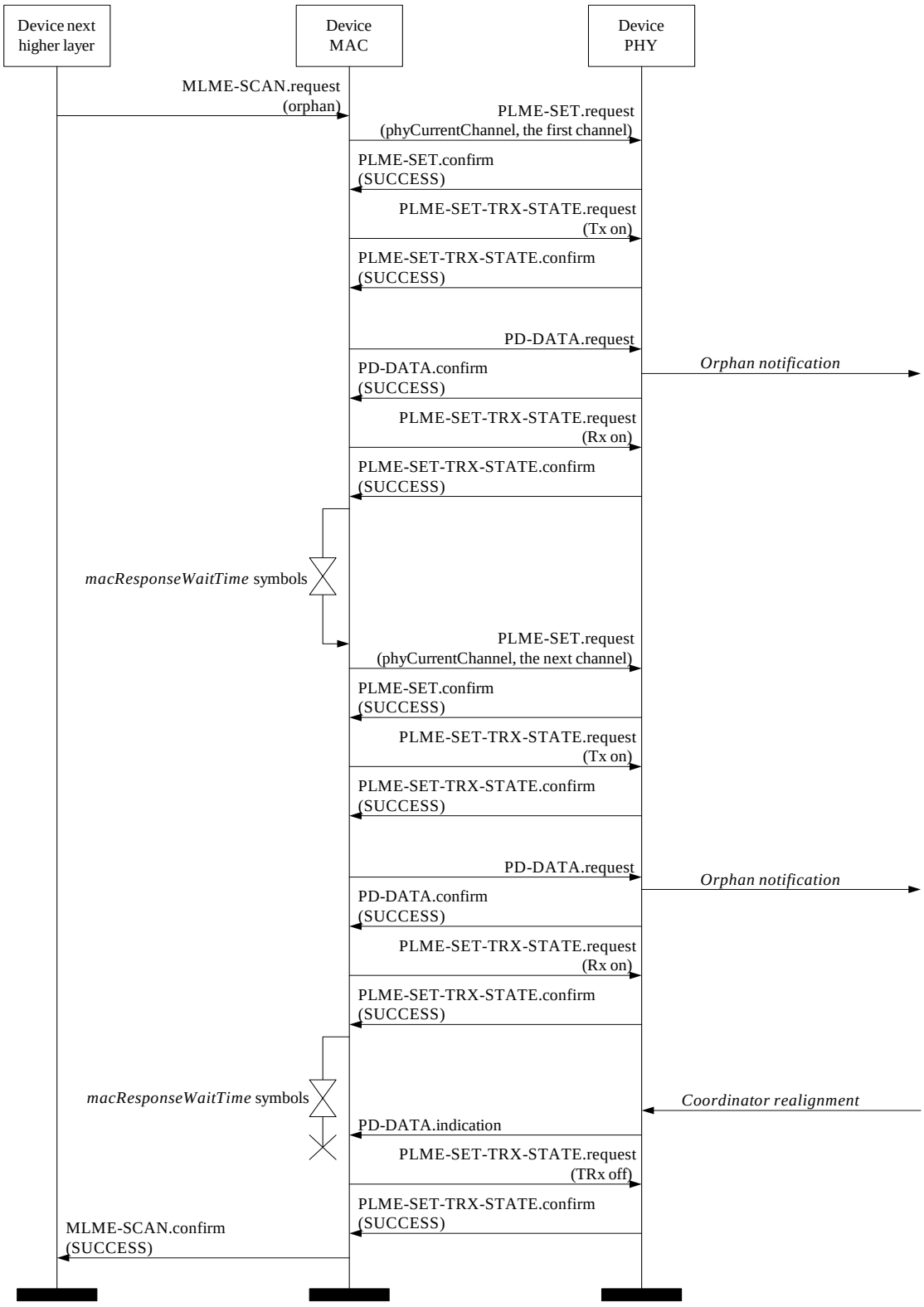


Figure 88—Orphaned device realignment message sequence chart