IEEE P802.15
Wireless Personal Area Networks

Project	IEEE P802.15 Working Group for Wireless Personal Area Networks (WPANs)	
Title	**Draft D17 Clause 7.6 Security Recommendation for Low-Rate IEEE 802.15.4 WPAN**	
Date Submitted	[15 November, 2002]	
Source	[Rene Struik] [Certicom Corporation] [5520 Explorer Drive, 4th floor, Mississauga, ON Canada L4W 5L1]	Voice: [905-501-6083] Fax: [905-507-4230] E-mail: [rstruik@certicom.com]
Re:	[]	
Abstract	[This document gives some security and security architectural recommendations to assist in improving the security and flexibility of the Draft D17 for the IEEE 802.15.4 Low-Rate WPAN]	
Purpose	[Assist sponsor ballot comment resolution for the Draft D17 for the IEEE 802.15.4 WPAN.]	
Notice	This document has been prepared to assist the IEEE P802.15. It is offered as a basis for discussion and is not binding on the contributing individual(s) or organization(s). The material in this document is subject to change in form and content after further study. The contributor(s) reserve(s) the right to add, amend or withdraw material contained herein.	
Release	The contributor acknowledges and accepts that this contribution becomes the property of IEEE and may be made publicly available by P802.15.	

Editorial note: Clause 7 provisional replacement text, to take away the current security vulnerabilities of Draft D17 (for details, see also 02/474r1).

1. MAC sublayer specification

This clause specifies the IEEE 802.15.4 medium access control (MAC) sublayer. 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 the beacons.
- — Supporting PAN association and disassociation.
- — Supporting device security.
- — Employing the CSMA-CA mechanism for channel access.
- — Handling and maintaining the guaranteed time slot 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. *aAckWaitDuration*, and are listed in Table 43. Attributes have a general prefix of “mac”, e.g. *macAssociationPermit*, and are listed in Table 44 and Table 45.

1.1 MAC sublayer service specification

The MAC sublayer provides an interface between the Service Specific Convergence Sublayer (SSCS) and the PHY layer. The MAC sublayer conceptually includes a management entity called the MAC sublayer management entity (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 PAN Information Base (PIB).

The device management entity (DME) is a layer-independent entity that may be viewed as residing in a separate management plane or “off to one side”. The exact functions of the DME are not specified in this standard, but in general this entity may be viewed as being responsible for interfacing with the layer-specific management entities through their associated interfaces in order to provide general system management functions for the device.

Figure 1 depicts the components and interfaces of the MAC sublayer.

The MAC sublayer provides two services, accessed through two service access points (SAPs). These are the MAC data service, accessed through the MAC common part sublayer (MCPS) data SAP, (MCPS-SAP), and the MAC management service, accessed through the MAC sublayer management entity SAP, (MLME-SAP). These two services provide the interface between the SSCS and the PHY layer, via the PD-SAP and PLME-SAP interfaces (See subclause 6.2). The MAC sublayer also provides the DME-MLME-SAP interface, which is identical to the MLME-SAP interface but can be used via the DME rather than the next higher layer. In addition to these external interfaces, there is also an implicit interface between the MLME and the MCPS that allows the MLME to use the MAC data service.

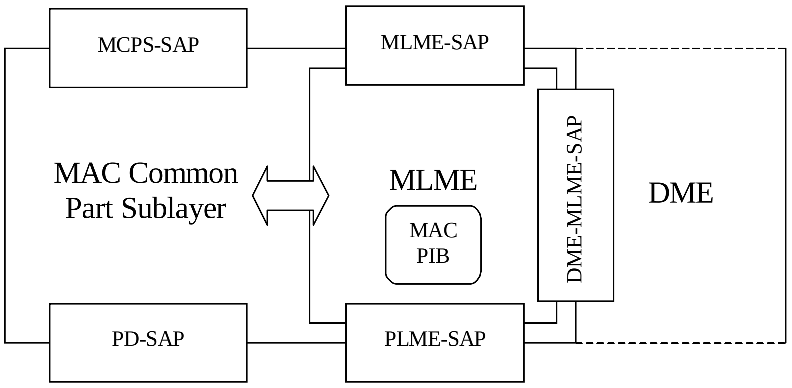


Figure 1—The MAC sublayer reference model

1.1.1 MAC data service

The MAC common part sublayer SAP (MCPS-SAP) supports the transport of SSCS protocol data units (SPDUs) between peer SSCS entities. Table 1 lists the primitives supported by the MCPS-SAP. Each of these primitives will be discussed in the following subclauses.

Table 1—MCPS-SAP primitives

MCPS-SAP Primitive	Request	Confirm	Indication
MCPS-DATA	1.1.1.1	1.1.1.2	1.1.1.3

All devices shall implement these primitives.

1.1.1.1 MCPS-DATA.request

This primitive requests the transfer of a data SPDU (MSDU) from a local SSCS entity to a single peer SSCS entity.

1.1.1.1.1 Semantics of the service primitive

This primitive shall provide the following interface:

```
MCPS-DATA.request            (
                             SrcAddrMode,
                             SrcPANId,
                             SrcAddr,
                             DstAddrMode,
                             DstPANId,
                             DstAddr,
                             msduLength,
                             msdu,
                             TxOptions
                             )
```

Table 2 specifies the parameters for the MCPS-DATA.request primitive.

1.1.1.1.2 When generated

This primitive is generated by a local SSCS entity whenever a data SPDU (MSDU) is to be transferred to a peer SSCS entity.

1.1.1.1.3 Effect on receipt

On receipt of this primitive, the MAC sublayer entity shall begin the transmission of the supplied MSDU.

The flags in the SrcAddrMode and DstAddrMode parameters correspond to the addressing subfields in the frame control field (See subclause 1.2.1.1) and shall be used to construct both the frame control and addressing fields of the MAC header.

The MAC sublayer shall build an MPDU to transmit from the supplied arguments. The TxOptions parameter indicates how the MAC sublayer data service shall transmit the supplied MSDU. The indirect transmission bit shall be ignored if either the GTS transmission bit is set to 1, the destination address is not present or the MAC sublayer receiving this primitive is not that of a coordinator.

If the TxOptions parameter specifies that a GTS transmission is required, the MAC sublayer shall defer the transmission, if necessary, until the transmit GTS, as indicated by *macTxGTSId*, provided this value is non-zero. At this time, the MAC sublayer shall transmit the MPDU without using CSMA-CA, provided that the entire transmission and acknowledgement, if requested, can be completed before the end of the GTS. If *macTxGTSId* is equal to 0, the transmission shall be aborted and the MAC sublayer shall issue the MCPS-DATA.confirm primitive with a status of INVALID_GTS. If the TxOptions parameter specifies that a GTS transmission is not required, the MAC sublayer shall transmit the MSDU using CSMA-CA either in the CAP for a beacon enabled PAN or immediately for a non-beacon enabled PAN. Specifying a GTS transmission in the TxOptions parameter shall override 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 destination address shall be added to the address list field of the beacon, indicating a pending message. Upon receipt of this primitive, the coordinator shall attempt 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 shall 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 shall add the information to the list. If the transaction is not handled within *macTransactionPersistenceTime*, the transaction information shall be discarded and the MAC sublayer shall issue the MCPS-DATA.confirm

Table 2—MCPS-DATA.request parameters

Name	Type	Valid Range	Description
SrcAddrMode	Integer	0-3	The source addressing mode for this primitive and subsequent MPDU. This value can take one of the following values: 0 = no address (ignore addressing fields). 1 = 8-bit short address. 2 = 16-bit short address. 3 = 64-bit extended address.
SrcPANId	Integer	0x0000-0xffff	The 16-bit PAN identifier of the entity from which the MSDU is being transferred.
SrcAddr	Device address	As specified by the SrcAddrMode parameter.	The individual device address of the entity from which the MSDU is being transferred.
DstAddrMode	Integer	0-3	The destination addressing mode for this primitive and subsequent MPDU. This value can take one of the following values: 0 = no address (ignore addressing fields). 1 = 8-bit short address. 2 = 16-bit short address. 3 = 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	≤*aMaxMACFrameSize*	The number of octets contained in the MSDU to be transmitted by the MAC sublayer entity.
msdu	Set of octets	-	The set of octets comprising the MSDU to be transmitted by the MAC sublayer entity.
TxOptions	Bitmap	0000 xxxx (Where x can be 0 or 1)	The transmission options for this MSDU. These shall be a bitwise OR of one or more of the following: 0x01 = acknowledged transmission. 0x02 = GTS transmission. 0x04 = indirect transmission. 0x08 = security enabled transmission.

primitive with a status of TRANSACTION_EXPIRED. The transaction handling procedure is described in subclause 1.5.5. If the TxOptions parameter specifies that an indirect transmission is required and either this primitive is not received by a coordinator or the TxOptions parameter also specifies a GTS transmission, the indirect transmission option shall be ignored. If the TxOptions parameter specifies that an indirect transmission is not required, the MAC sublayer shall transmit the MSDU using CSMA-CA either in the CAP for a beacon enabled PAN or immediately for a non-beacon enabled PAN.

If the TxOptions parameter specifies that security is not required for this frame, the MAC sublayer shall set the security enabled subfield of the frame control field to 0 (See subclause 1.2.1.1.2) and not perform any security operations on the frame. If the TxOptions parameter specifies that security is required for this frame, the MAC sublayer shall set the security enabled subfield of the frame control field to 1 and obtain the key and security information, corresponding to the DstAddr parameter, from the ACL entries in the MAC

PIB, as described in subclause 1.5.9.4.1. If an appropriate key could not be found in the ACL, the MAC sublayer shall discard the frame and issue the MCPS-DATA.confirm primitive with a status of UNAVAILABLE_KEY. If an appropriate key was found in the ACL, the MAC sublayer shall use it to apply security to the frame, according to the security information found in the ACL (See subclause 1.5.9.4). If the length of the resulting frame is longer than *aMaxMACFrameSize*, the MAC sublayer shall discard the frame and issue the MCPS-DATA.confirm primitive with a status of FRAME_TOO_LONG. If any other error occurs during the secure processing of the frame, the MAC sublayer shall discard the frame and issue the MCPS-DATA.confirm primitive with a status of FAILED_SECURITY_CHECK.

If this primitive is received by a device before it has completed the processing of a previous MCPS-DATA.request primitive that was not a request for indirect transmission, i.e. it has not yet issued the MCPS-DATA.confirm primitive, it shall discard the MSDU and issue the MCPS-DATA.confirm primitive with a status of TRANSMISSION_IN_PROGRESS.

If the transmission uses CSMA-CA and the CSMA-CA algorithm failed due to adverse conditions on the channel, the MAC sublayer shall discard the MSDU and issue the MCPS-DATA.confirm primitive with a status of CHANNEL_ACCESS_FAILURE.

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 layer. On receipt of the PLME-SET-TRX-STATE.confirm primitive with a status of either SUCCESS or TX_ON, the constructed MPDU 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 layer, depending on whether the receiver is to be enabled following the transmission.

If the TxOptions parameter specifies that an acknowledged transmission is required, the MAC sublayer shall enable its receiver immediately following the transmission of the MPDU and wait for an acknowledgment from the recipient for at most *aAckWaitDuration* symbols. If the MAC sublayer does not receive an acknowledgment within this time, it shall retry its transmission at most *aMaxFrameRetries* times. If the MAC sublayer still does not receive an acknowledgment from the recipient, it shall discard the MSDU and issue the MCPS-DATA.confirm primitive with a status of NO_ACK.

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

1.1.1.2 MCPS-DATA.confirm

This primitive reports the results of a request to transfer a data SPDU (MSDU) from a local SSCS entity to a single peer SSCS entity.

1.1.1.2.1 Semantics of the service primitive

This primitive shall provide the following interface:

```
MCPS-DATA.confirm          (
                           DstAddrMode,
                           DstPANId,
                           DstAddr,
                           status
                           )
```

Table 3 specifies the parameters for the MCPS-DATA.confirm primitive.

Table 3—MCPS-DATA.confirm parameters

Name	Type	Valid Range	Description
DstAddrMode	Integer	0-3	The destination addressing mode for this primitive. This value can take one of the following values: 0 = no address (ignore addressing fields). 1 = 8-bit short address. 2 = 16-bit short address. 3 = 64-bit extended address.
DstPANId	Integer	0x0000-0xffff	The 16-bit PAN identifier of the entity to which the MSDU was transferred.
DstAddr	Device address	As specified by the DstAddrMode parameter.	The individual device address of the entity to which the MSDU was transferred.
status	Enumeration	SUCCESS, TRANSACTION_OVERFLOW, TRANSMISSION_IN_PROGRESS, TRANSACTION_EXPIRED, CHANNEL_ACCESS_FAILURE, INVALID_GTS, NO_ACK, UNAVAILABLE_KEY, FRAME_TOO_LONG, FAILED_SECURITY_CHECK or INVALID_PARAMETER	The status of the last MSDU transmission.

1.1.1.2.2 When generated

This primitive is generated by the MAC sublayer entity in response to an MCPS-DATA.request primitive. This primitive shall return a status of either SUCCESS, indicating that the request to transmit was successful, or an error code of TRANSACTION_OVERFLOW, TRANSMISSION_IN_PROGRESS, TRANSACTION_EXPIRED, CHANNEL_ACCESS_FAILURE, INVALID_GTS, NO_ACK, UNAVAILABLE_KEY, FRAME_TOO_LONG, FAILED_SECURITY_CHECK or INVALID_PARAMETER. The reasons for these status values are fully described in subclause 1.1.1.1.3.

1.1.1.2.3 Effect on receipt

On receipt of this 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 shall be set to SUCCESS. Otherwise, the status parameter shall indicate the error.

1.1.1.3 MCPS-DATA.indication

This primitive indicates the transfer of a data SPDU (MSDU) from the MAC sublayer to the local SSCS entity.

1.1.1.3.1 Semantics of the service primitive

This primitive shall provide the following interface:

```
MCPS-DATA.indication            (
                                SrcAddrMode,
                                SrcPANId,
                                SrcAddr,
                                DstAddrMode,
                                DstPANId
                                DstAddr,
                                msduLength,
                                msdu,
                                mpduLinkQuality,
                                SecurityUse,
                                ACLEntry
                                )
```

Table 4 specifies the parameters for the MCPS-DATA.indication primitive.

1.1.1.3.2 When generated

This 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 subclause 1.5.6.2.

1.1.1.3.3 Effect on receipt

On receipt of this primitive the SSCS is notified of the arrival of data at the device.

1.1.1.4 Data service message sequence charts

Figure 2 illustrates the sequence of messages necessary for a successful data transfer between two devices. Figure 56 and Figure 57 (See subclause 1.7) illustrate this scenario, including the steps taken by the PHY layer.

Table 4—MCPS-DATA.indication parameters

Name	Type	Valid Range	Description
SrcAddrMode	Integer	0-3	The source addressing mode for this primitive corresponding to the received MPDU. This value can take one of the following values: 0 = no address (ignore addressing fields). 1 = 8-bit short address. 2 = 16-bit short address. 3 = 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	0-3	The destination addressing mode for this primitive corresponding to the received MPDU. This value can take one of the following values: 0 = no address (ignore addressing fields). 1 = 8-bit short device address. 2 = 16-bit short device address. 3 = 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.
msduLength	Integer	≤*aMaxMACFrameSize*	The number of octets contained in the MSDU being indicated by the MAC sublayer entity.
msdu	Set of octets	-	The set of octets comprising the MSDU being indicated by the MAC sublayer entity.
mpduLinkQuality	Integer	0x00-0xff	Link quality value measured during reception of the MPDU. Lower values represent lower link quality (See subclause 6.7.8).
SecurityUse	Boolean	TRUE or FALSE	An indication of whether the received data frame is using security. This value is set to TRUE if the security enable subfield was set to 1 or FALSE if the security enabled subfield was set to 0.
ACLEntry	Boolean	TRUE or FALSE	The parameter is set to TRUE if the sender of the data frame was found in the ACL. Otherwise, the parameter is set to FALSE.

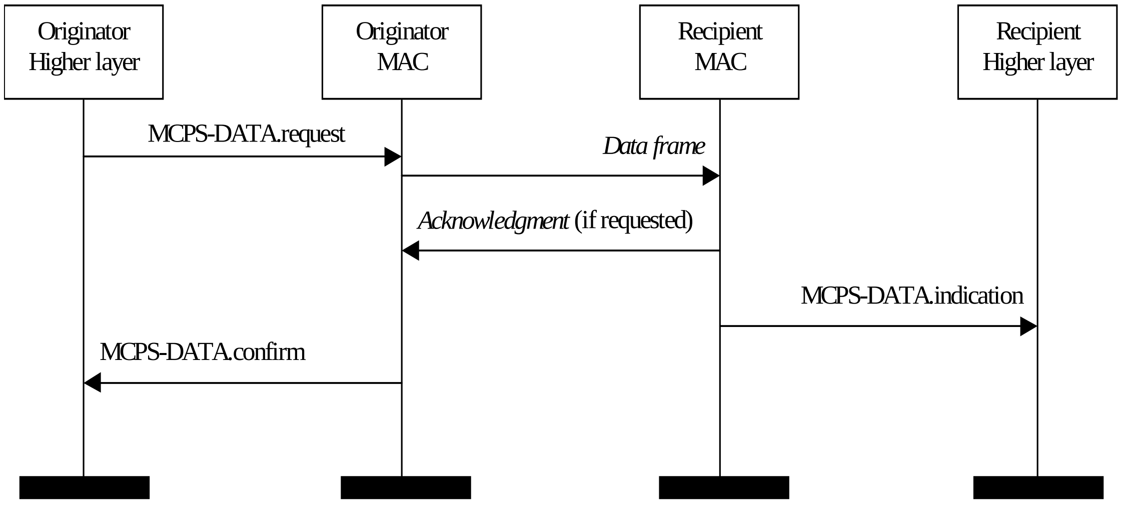


Figure 2—Message sequence chart describing the MAC data service

1.1.2 MAC management service

The MAC sublayer management entity SAP (MLME-SAP) allows the transport of management commands between the next higher layer and the MLME. An additional MAC sublayer management entity SAP (DME-MLME-SAP) allows the transport of management commands between the DME and the MLME; this interface is identical to the MLME-SAP interface. Table 5 summarizes the primitives supported by the MLME through the MLME-SAP interface. Primitives marked with an asterisk (*) are optional for an RFD. See the following subclauses for more details on the individual primitives.

Table 5—Summary of the primitives accessed through the MLME-SAP

Name	Request	Indication	Response	Confirm
MLME-ASSOCIATE	1.1.3.1	1.1.3.2*	1.1.3.3*	1.1.3.4
MLME-DISASSOCIATE	1.1.4.1	1.1.4.2		1.1.4.3
MLME-BEACON-NOTIFY		1.1.5.1		
MLME-GET	1.1.6.1			1.1.6.2
MLME-GTS	1.1.7.1*	1.1.7.3*		1.1.7.2*

Table 5—Summary of the primitives accessed through the MLME-SAP

Name	Request	Indication	Response	Confirm
MLME-ORPHAN		1.1.8.1*	1.1.8.2*	
MLME-RESET	1.1.9.1			
MLME-RX-ENABLE	1.1.10.1			
MLME-SCAN	1.1.11.1			1.1.11.2
MLME-SECURITY-ERROR		1.1.12.1		
MLME-SET	1.1.13.1			1.1.13.2
MLME-START	1.1.14.1*			1.1.14.2*
MLME-SYNC	1.1.15.1			
MLME-SYNC-LOSS		1.1.15.2		
MLME-POLL	1.1.16.1			1.1.16.2

1.1.3 Association primitives

These primitives define how a device becomes associated with a PAN.

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

1.1.3.1 MLME-ASSOCIATE.request

This primitive allows a device to request an association with a coordinator.

1.1.3.1.1 Semantics of the service primitive

This primitive shall provide the following interface:

```
MLME-ASSOCIATE.request        (
                              CoordAddrMode,
                              CoordPANId,
                              CoordAddress,
                              CapabilityInformation,
                              SecurityEnable
                              )
```

Table 6 specifies the parameters for the MLME-ASSOCIATE.request primitive.

1.1.3.1.2 When generated

This primitive is generated by the DME of an unassociated device and issued to its MLME to request an association with a coordinator.

Table 6—MLME-ASSOCIATE.request parameters

Name	Type	Valid Range	Description
CoordAddrMode	Integer	1-3	The coordinator addressing mode for this primitive and subsequent MPDU. This value can take one of the following values: 1 = 8-bit short address. 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 subclause 1.3.1.1.2)	Specifies the operational capabilities of the associating device.
SecurityEnable	Boolean	TRUE or FALSE	TRUE if security is enabled for this transfer or FALSE otherwise.

1.1.3.1.3 Effect on receipt

On receipt of this primitive, the MLME of an unassociated device shall generate an association request command (See subclause 1.3.1.1) and send it to the coordinator with the specified PAN identifier and address.

The SecurityEnable parameter specifies whether security is 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 does have knowledge of the security information of the coordinator, then security may be specified in this case. If the SecurityEnable parameter is set to FALSE, the MLME shall set the security enabled subfield of the frame control field to 0 (See subclause 1.2.1.1.2) and not perform any security operations on the frame. If the SecurityEnable parameter is set to TRUE, the MLME shall set the security enabled subfield of the frame control field to 1 and obtain the key and security information, corresponding to the CoordinatorAddress parameter, from the ACL entries in the MAC PIB, as described in subclause 1.5.9.4.1. If an appropriate key could not be found in the ACL, the MLME shall discard the frame and issue the MLME-ASSOCIATE.confirm primitive with a status of UNAVAILABLE_KEY. If an appropriate key was found in the ACL, the MLME shall use it to apply security to the frame, according to the security information found in the ACL (See subclause 1.5.9.4). If any other error occurs during the secure processing of the frame, the MLME shall discard the frame and issue the MLME-ASSOCIATE.confirm primitive with a status of FAILED_SECURITY_CHECK.

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

To transmit the association request frame, the MLME shall first enable the transmitter by issuing the PLME-SET-TRX-STATE.request primitive with a state of TX_ON to the PHY layer. On receipt of the PLME-SET-TRX-STATE.confirm primitive with a status of either SUCCESS or TX_ON, the association request command frame shall then be transmitted by issuing the PD-DATA.request primitive. Finally, on receipt of the PD-DATA.confirm primitive, the MLME shall enable the receiver by issuing the PLME-SET-TRX-STATE.request primitive with a state of RX_ON to the PHY layer in preparation for the acknowledgement.

If the MLME successfully transmits an association request command, the MLME shall expect an acknowledgment in return. If this does not occur the association request command frame shall be retried. If an acknowledgement is not received after *aMaxFrameRetries* attempts, the MLME shall issue the MLME-ASSOCIATE.confirm primitive with a status of NO_ACK.

If the MLME of an unassociated device successfully transmits an association request command, the MLME shall wait for *aResponseWaitTime* symbols for the association response command to become available. After this time, the MLME shall attempt to extract the association response command from the coordinator. If the MLME of the device does not extract a frame from the coordinator it shall issue the MLME-ASSOCIATE.confirm primitive with a status of NO_DATA.

If the MLME of the device extracts a frame from the coordinator it shall 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 subclause 1.3.1.2.4).

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

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

1.1.3.2 MLME-ASSOCIATE.indication

This primitive is used to indicate the reception of an association request command.

1.1.3.2.1 Semantics of the service primitive

This primitive shall provide the following interface:

```
MLME-ASSOCIATE.indication     (
                              DeviceAddress,
                              CapabilityInformation,
                              SecurityUse,
                              ACLEntry
                              )
```

Table 7 specifies the parameters for the MLME-ASSOCIATE.indication primitive.

1.1.3.2.2 When generated

This primitive is generated by the MLME of the coordinator and issued to its DME to indicate the reception of an association request command (See subclause 1.3.1.1).

1.1.3.2.3 Effect on receipt

When the DME of a coordinator receives this primitive, the coordinator shall determine whether to accept or reject the unassociated device using an algorithm outside the scope of this standard. The DME of the coordinator shall then issue the MLME-ASSOCIATE.response primitive to its MLME.

The association decision and the response shall occur within a time of *aResponseWaitTime*. If this does not occur, the device requesting association shall assume the association was unsuccessful.

Table 7—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 subclause 1.3.1.1.2)	The operational capabilities of the device requesting association.
SecurityUse	Boolean	TRUE or FALSE	An indication of whether the received data frame is using security. This value is set to TRUE if the security enable subfield was set to 1 or FALSE if the security enabled subfield was set to 0.
ACLEntry	Boolean	TRUE or FALSE	The parameter is set to TRUE if the sender of the MAC command frame was found in the ACL. Otherwise, the parameter is set to FALSE.

1.1.3.3 MLME-ASSOCIATE.response

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

1.1.3.3.1 Semantics of the service primitive

This primitive shall provide the following interface:

```
MLME-ASSOCIATE.response      (
                             DeviceAddress,
                             AssocShortAddress,
                             status,
                             SecurityEnable
                             )
```

Table 8 specifies the parameters for the MLME-ASSOCIATE.response primitive.

Table 8—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 short device address allocated by the coordinator on successful association. This parameter shall be set to 0x00ff if the association was unsuccessful.
status	Enumeration	(See subclause 1.3.1.2.4)	The status of the association attempt.
SecurityEnable	Boolean	TRUE or FALSE	TRUE if security is enabled for this transfer or FALSE otherwise.

1.1.3.3.2 When generated

This primitive is generated by the DME of a coordinator and issued to its MLME in order to respond to the MLME-ASSOCIATE.indication primitive.

1.1.3.3.3 Effect on receipt

When the MLME of a coordinator receives this primitive, it shall generate an association response command (See subclause 1.3.1.2). The command shall be sent to the device requesting association using indirect transmission, i.e. the address of the device shall be added to the address list field of the beacon frame, indicating a pending message, and extracted at the discretion of the device.

The SecurityEnable parameter specifies whether security is to be applied to the association response command frame. If the SecurityEnable parameter is set to FALSE, the MLME shall set the security enabled subfield of the frame control field to 0 (See subclause 1.2.1.1.2) and not perform any security operations on the frame. If the SecurityEnable parameter is set to TRUE, the MLME shall set the security enabled subfield of the frame control field to 1 and obtain the key and security information, corresponding to the DeviceAddress parameter, from the ACL entries in the MAC PIB, as described in subclause 1.5.9.4.1. If an appropriate key could not be found in the ACL, the MLME shall discard the frame and issue the MLME-SECURITY-ERROR.indication primitive with a reason code of UNAVAILABLE_KEY. If an appropriate key was found in the ACL, the MLME shall use it to apply security to the frame, according to the security information found in the ACL (See subclause 1.5.9.4). If any other error occurs during the secure processing of the frame, the MLME shall discard the frame and issue the MLME-SECURITY-ERROR.indication primitive with a reason code of FAILED_SECURITY_CHECK.

To transmit the association response command frame, the MLME shall first enable the transmitter by issuing the PLME-SET-TRX-STATE.request primitive with a state of TX_ON to the PHY layer. On receipt of the PLME-SET-TRX-STATE.confirm primitive with a status of either SUCCESS or TX_ON, the association response command frame shall then be transmitted by issuing the PD-DATA.request primitive. Finally, on receipt of the PD-DATA.confirm primitive, the MLME shall enable the receiver by issuing the PLME-SET-TRX-STATE.request primitive with a state of RX_ON to the PHY layer in preparation for the acknowledgement.

1.1.3.4 MLME-ASSOCIATE.confirm

This primitive is used to inform the DME of the initiating device whether its request to associate was successful or unsuccessful.

1.1.3.4.1 Semantics of the service primitive

This primitive shall provide the following interface:

```
MLME-ASSOCIATE.confirm        (
                              AssocShortAddress,
                              status
                              )
```

Table 9 specifies the parameters for the MLME-ASSOCIATE.confirm primitive.

1.1.3.4.2 When generated

This primitive is generated by the initiating MLME and issued to its DME in response to an MLME-ASSOCIATE.request primitive. If the request was successful, the status parameter shall indicate a successful asso-

Table 9—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 shall be equal to 0x00ff if the association attempt was unsuccessful.
status	Integer	The value of the status field of the associate response command (See subclause 1.3.1.2.4), CHANNEL_ACCESS_FAILURE, NO_ACK, NO_DATA, UNAVAILABLE_KEY, FAILED_SECURITY_CHECK or INVALID_PARAMETER.	The status of the association attempt.

ciation, as contained in the status field of the association response command. Otherwise, the status parameter shall indicate either an error code from the received association response commend or an error code of CHANNEL_ACCESS_FAILURE, NO_ACK, NO_DATA, UNAVAILABLE_KEY, FAILED_SECURITY_CHECK or INVALID_PARAMETER. The reasons for these status values are fully described in subclause 1.1.3.1.3.

1.1.3.4.3 Effect on receipt

On receipt of this primitive, the DME 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 shall indicate a successful association, as contained in the status field of the associate response command, and the device shall be provided with a short address. If this short address is in the range of 0x0000-0x00fd or 0x0100-0xffff, it may be used for communication in the PAN. If the short address is equal to 0x00fe, the device shall use its extended 64-bit address for communication in the PAN. If the association attempt was unsuccessful, the address will be equal to 0x00ff and the status parameter shall indicate the error.

1.1.3.5 Association message sequence charts

Figure 3 illustrates the sequence of messages necessary for a device to successfully associate with a PAN. Figure 53 and Figure 54 (See subclause 1.7) illustrate the sequence of messages necessary for a device to associate with a coordinator and for a coordinator to allow association by a device, respectively; these figures include steps taken by the PHY layer.

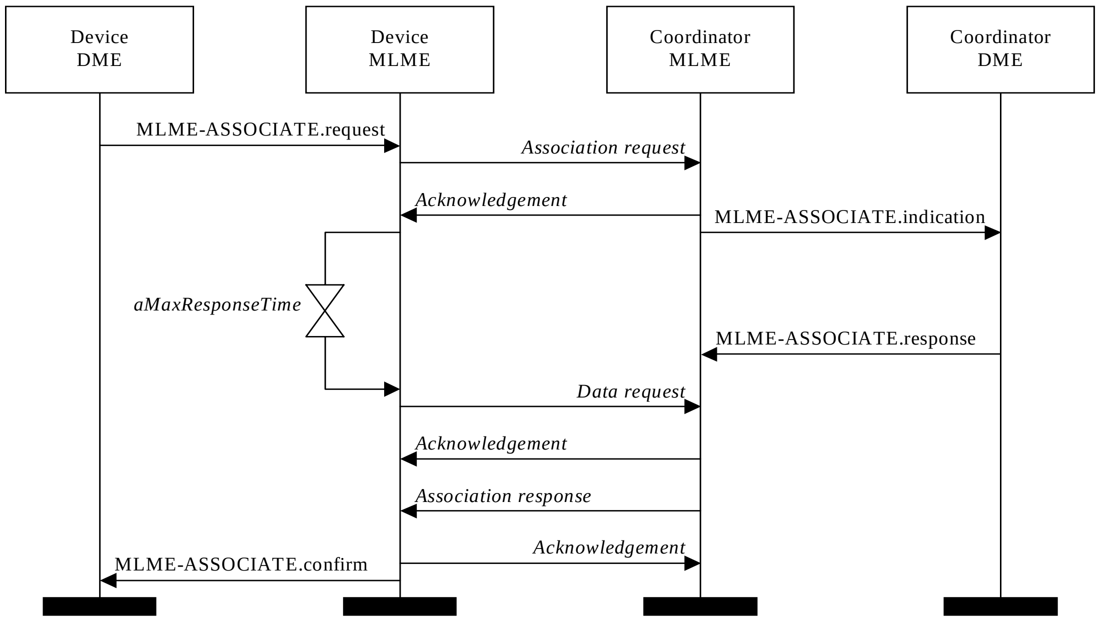


Figure 3—Message sequence chart for association

1.1.4 Disassociation primitives

These primitives define how a device can disassociate from a PAN.

All devices shall implement these primitives.

1.1.4.1 MLME-DISASSOCIATE.request

This 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.

1.1.4.1.1 Semantics of the service primitive

This primitive shall provide the following interface:

```
MLME-DISASSOCIATE.request      (
                               DeviceAddress,
                               DisassociateReason,
                               SecurityEnable
                               )
```

Table 10 specifies the parameters for the MLME-DISASSOCIATE.request primitive.

1.1.4.1.2 When generated

This primitive is generated by the DME of an associated device and issued to its MLME to request disassociation from the PAN. It is also generated by the DME of the coordinator and issued to its MLME to instruct an associated device to leave the PAN.

Table 10—MLME-DISASSOCIATE.request parameters

Name	Type	Valid Range	Description
DeviceAddress	Device address	An extended 64-bit, IEEE address.	The address of the device to which to send the disassociation notification command.
DisassociateReason	Integer	0x00-0xff	The reason for the disassociation (See subclause 1.3.1.3.2).
SecurityEnable	Boolean	TRUE or FALSE	TRUE if security is enabled for this transfer or FALSE otherwise.

1.1.4.1.3 Effect on receipt

On receipt of this primitive, the MLME shall generate a disassociation notification command (See subclause 1.3.1.3). If this primitive is received by the MLME of a coordinator, it shall send the command using indirect transmission. Otherwise, the command shall be sent to the coordinator in the CAP for a beacon enabled PAN or immediately for a non-beacon enabled PAN.

If the disassociation notification command is to be sent using indirect transmission by the coordinator, the device address shall be added to the address list field of the beacon, indicating a pending message. In this case, the coordinator shall attempt to add the information contained in the primitive to its list of pending transactions. If there is no capacity to store the transaction, the MLME shall discard the MSDU and issue the MLME-DISASSOCIATE.confirm primitive with a status of TRANSACTION_OVERFLOW. If there is capacity to store the transaction, the coordinator shall add the information to the list. If the transaction is not handled within *macTransactionPersistenceTime*, the transaction information shall be discarded and the MLME shall issue the MLME-DISASSOCIATE.confirm with a status of TRANSACTION_EXPIRED. The transaction handling procedure is described in subclause 1.5.5.

The SecurityEnable parameter specifies whether security is to be applied to the disassociation notification command frame. If the SecurityEnable parameter is set to FALSE, the MLME shall set the security enabled subfield of the frame control field to 0 (See subclause 1.2.1.1.2) and not perform any security operations on the frame. If the SecurityEnable parameter is set to TRUE, the MLME shall set the security enabled subfield of the frame control field to 1 and obtain the key and security information, corresponding to the DeviceAddress parameter, from the ACL entries in the MAC PIB, as described in subclause 1.5.9.4.1. If an appropriate key could not be found in the ACL, the MLME shall discard the frame and issue the MLME-DISASSOCIATE.confirm primitive with a status of UNAVAILABLE_KEY. If an appropriate key was found in the ACL, the MLME shall use it to apply security to the frame, according to the security information found in the ACL (See subclause 1.5.9.4). If any other error occurs during the secure processing of the frame, the MLME shall discard the frame and issue the MLME-DISASSOCIATE.confirm primitive with a status of FAILED_SECURITY_CHECK.

If the disassociation notification command cannot be sent due to a CSMA algorithm failure, the MLME shall issue the MLME-DISASSOCIATE.confirm primitive with a status of CHANNEL_ACCESS_FAILURE.

To transmit the disassociation notification command frame, the MLME shall first enable the transmitter by issuing the PLME-SET-TRX-STATE.request primitive with a state of TX_ON to the PHY layer. On receipt of the PLME-SET-TRX-STATE.confirm primitive with a status of either SUCCESS or TX_ON, the disassociation notification command frame shall then be transmitted by issuing the PD-DATA.request primitive. Finally, on receipt of the PD-DATA.confirm primitive, the MLME shall enable the receiver by issuing the PLME-SET-TRX-STATE.request primitive with a state of RX_ON to the PHY layer in preparation for the acknowledgement.

If the MLME successfully transmits a disassociation notification command, the MLME shall expect an acknowledgment in return. If this does not occur the disassociation notification command frame shall be retried. If an acknowledgement is not received after *aMaxFrameRetries* attempts, the MLME shall issue the MLME-DISASSOCIATE.confirm primitive with a status of NO_ACK.

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

In the case where a device MLME receives this primitive from its DME, the disassociation notification command is sent to the coordinator specified in the DeviceAddress parameter. Similarly, in the case where a coordinator MLME receives this primitive from its DME, the disassociation notification command is sent to the device specified in the DeviceAddress parameter.

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 shall issue the MLME-DISASSOCIATE.confirm primitive with a status of INVALID_PARAMETER.

1.1.4.2 MLME-DISASSOCIATE.indication

This primitive is used to indicate the reception of a disassociation notification command.

1.1.4.2.1 Semantics of the service primitive

This primitive shall provide the following interface:

```
MLME-DISASSOCIATE.indication    (
                                DeviceAddress,
                                DisassociateReason,
                                SecurityUse,
                                ACLEntry
                                )
```

Table 11 specifies the parameters for the MLME-DISASSOCIATE.indication primitive.

1.1.4.2.2 When generated

This primitive is generated by the MLME and issued to its DME on receipt of a disassociation notification command.

1.1.4.2.3 Effect on receipt

The DME is notified of the reason for the disassociation.

1.1.4.3 MLME-DISASSOCIATE.confirm

This primitive reports the results of an MLME-DISASSOCIATE.request primitive.

Table 11—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 subclause 1.3.1.3.2).
SecurityUse	Boolean	TRUE or FALSE	An indication of whether the received data frame is using security. This value is set to TRUE if the security enable subfield was set to 1 or FALSE if the security enabled subfield was set to 0.
ACLEntry	Boolean	TRUE or FALSE	The parameter is set to TRUE if the sender of the MAC command frame was found in the ACL. Otherwise, the parameter is set to FALSE.

1.1.4.3.1 Semantics of the service primitive

This primitive shall provide the following interface:

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

Table 12 specifies the parameters for the MLME-DISASSOCIATE.confirm primitive.

Table 12—MLME-DISASSOCIATE.confirm parameters

Name	Type	Valid Range	Description
Status	Enumeration	SUCCESS, TRANSACTION_OVERFLOW, TRANSACTION_EXPIRED, NO_ACK, CHANNEL_ACCESS_FAILURE, UNAVAILABLE_KEY, FAILED_SECURITY_CHECK or INVALID_PARAMETER	The status of the disassociation attempt.

1.1.4.3.2 When generated

This primitive is generated by the initiating MLME and issued to its DME in response to an MLME-DISASSOCIATE.request primitive. This primitive shall return a status of either SUCCESS, indicating that the disassociation request was successful, or an error code of TRANSACTION_OVERFLOW, TRANSACTION_EXPIRED, NO_ACK, CHANNEL_ACCESS_FAILURE, UNAVAILABLE_KEY, FAILED_SECURITY_CHECK or INVALID_PARAMETER. The reasons for these status values are fully described in subclause 1.1.4.1.3.

1.1.4.3.3 Effect on receipt

On receipt of this primitive, the DME of the initiating device is notified of the success of the disassociation attempt. If the disassociation attempt was successful, the status parameter shall be set to SUCCESS. Otherwise, the status parameter shall indicate the error.

1.1.4.4 Disassociation message sequence charts

Figure 4 illustrates the sequence of messages necessary for successful disassociation from a PAN. The originating device may either be a device or the coordinator to which the device has associated.

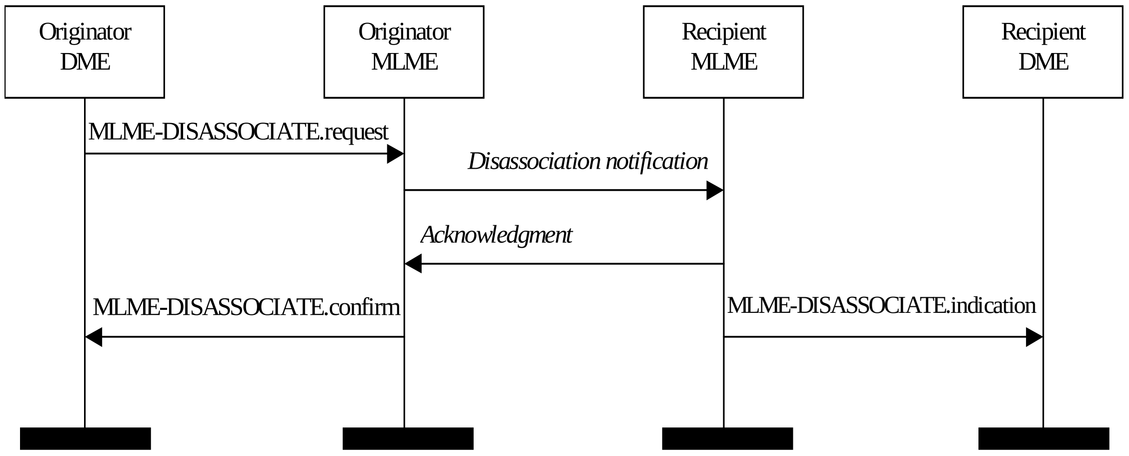

Originator DME
Originator MLME
Recipient MLME
Recipient DME
MLME-DISASSOCIATE.request
Disassociation notification
Acknowledgment
MLME-DISASSOCIATE.confirm
MLME-DISASSOCIATE.indication

Figure 4—Message sequence chart for disassociation

1.1.5 Beacon notification primitive

This primitive defines how a device may be notified when a beacon is received during normal operating conditions.

All devices shall implement this primitive.

1.1.5.1 MLME-BEACON-NOTIFY.indication

This primitive is used to send parameters contained within a beacon frame received by the MAC sublayer to the DME. The primitive also sends a measure of the link quality and the time the beacon frame was received.

1.1.5.1.1 Semantics of the service primitive

This primitive shall provide the following interface:

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

Table 13 specifies the parameters for the MLME-BEACON-NOTIFY.indication primitive.

Table 13—MLME-BEACON-NOTIFY.indication parameters

Name	Type	Valid Range	Description
BSN	Integer	0x00-0xff	The beacon sequence number.
PANDescriptor	PANDescriptor value	See Table 14	The PAN descriptor for the received beacon.
PendAddrSpec	Bitmap	(See subclause 1.2.2.1.3)	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-*aMaxBeaconMSDU-Length*	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 pay-load to be transferred from the MAC sublayer entity to the DME.

Table 14 describes the elements of the PANDescriptor type.

1.1.5.1.2 When generated

This primitive is generated by the MLME and issued to its DME upon receipt of a beacon frame either when *macAutoRequest* is set to FALSE or the beacon frame contains one or more octets of payload.

1.1.5.1.3 Effect on receipt

On receipt of this primitive, the DME is notified of the arrival of a beacon frame at the MAC sublayer.

1.1.6 Primitives for reading MAC PIB attributes

These primitives define how to read values from the MAC PIB.

All devices shall implement these primitives.

Table 14—Elements of PAN descriptor

Name	Type	Valid Range	Description
CoordAddrMode	Integer	1-3	The coordinator addressing mode for this primitive corresponding to the received beacon frame. This value can take one of the following values: 1 = 8-bit short address. 2 = 16-bit short address. 3 = 64-bit extended address.
CoordPANId	2 octets	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	The current logical channel occupied by the network.
SuperframeSpec	Bitmap	(See subclause 1.2.2.1.2)	The superframe specification as specified in the received beacon frame.
LinkQuality	Integer	0x00-0xff	The link quality at which the network beacon was received. Lower values represent lower link quality (See subclause 6.7.8).
TimeStamp	Integer	0x000000-0xffffff	The time at which the beacon frame was received, in symbols. This value shall be equal to the timestamp taken when the beacon frame was received, as described in subclause 1.5.4.1. The precision of this value shall be a minimum of 20-bits, with the lowest four bits being the least significant.
SecurityUse	Boolean	TRUE or FALSE	An indication of whether the received data frame is using security. This value is set to TRUE if the security enable subfield was set to 1 or FALSE if the security enabled subfield was set to 0.
ACLEntry	Boolean	TRUE or FALSE	The parameter is set to TRUE if the sender of the beacon frame was found in the ACL. Otherwise, the parameter is set to FALSE.
SecurityFailure	Boolean	TRUE or FALSE	The parameter is set to TRUE if there was an error in the security processing of the frame. Otherwise, the parameter is set to FALSE.

1.1.6.1 MLME-GET.request

This primitive requests information about a given MAC PIB attribute.

1.1.6.1.1 Semantics of the service primitive

The primitive shall provide the following interface:

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

Table 15 specifies the parameters for the MLME-GET.request primitive.

Table 15—MLME-GET.request parameters

Name	Type	Valid Range	Description
PIBAttribute	Enumeration	See Table 44 and Table 45	The identifier of the MAC PIB attribute to read.

1.1.6.1.2 When generated

This primitive is generated by the DME and issued to its MLME to obtain information from the MAC PIB.

1.1.6.1.3 Effect on receipt

On receipt of this primitive, the MLME shall attempt 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 shall issue the MLME-GET.confirm primitive with a status of UNSUPPORTED_ATTRIBUTE.

If the requested MAC PIB attribute is successfully retrieved, the MLME shall issue the MLME-GET.confirm primitive with a status of SUCCESS.

1.1.6.2 MLME-GET.confirm

This primitive reports the results of an information request from the MAC PIB.

1.1.6.2.1 Semantics of the service primitive

The primitive shall provide the following interface:

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

Table 16 specifies the parameters for the MLME-GET.confirm primitive.

1.1.6.2.2 When generated

This primitive is generated by the MLME and issued to its DME in response to an MLME-GET.request primitive. This primitive shall return a status of either SUCCESS, indicating that the request to read a MAC PIB attribute was successful, or an error code of UNSUPPORTED_ATTRIBUTE. The reasons for these status values are fully described in subclause 1.1.6.1.3.

Table 16—MLME-GET.confirm parameters

Name	Type	Valid Range	Description
status	Enumeration	SUCCESS, UNSUPPORTED_ATTRIBUTE	The result of the request for MAC PIB attribute information.
PIBAttribute	Enumeration	See Table 44 and Table 45	The identifier of the MAC PIB attribute that was read.
PIBAttributeValue	Various	Attribute Specific (See Table 44 and Table 45)	The value of the indicated MAC PIB attribute that was read.

1.1.6.2.3 Effect on receipt

On receipt of this primitive, the DME is notified of the results of its request to read a MAC PIB attribute. If the request to read a MAC PIB attribute was successful, the status parameter shall be set to SUCCESS. Otherwise, the status parameter shall indicate the error.

1.1.7 GTS management primitives

These primitives define how GTSs are requested and maintained.

These primitives are optional for an RFD.

1.1.7.1 MLME-GTS.request

This primitive allows a device to send a request to the PAN coordinator to allocate a new GTS or deallocate an existing GTS.

1.1.7.1.1 Semantics of the service primitive

The primitive shall provide the following interface:

```
MLME-GTS.request            (
                            GTSId,
                            GTSCharacteristics,
                            SecurityEnable
                            )
```

Table 17 specifies the parameters for the MLME-GTS.request primitive.

Table 17—MLME-GTS.request parameters

Name	Type	Valid Range	Description
GTSId	Integer	0-7	Either 0, if requesting the allocation of a new GTS or the non-zero identifier of an existing GTS, if requesting the deallocation of a GTS.
GTSCharacteristics	GTS characteristics	(See subclause 1.3.3.1.3)	The desired characteristics of the GTS.
SecurityEnable	Boolean	TRUE or FALSE	TRUE if security is enabled for this transfer or FALSE otherwise.

1.1.7.1.2 When generated

This primitive is generated by the DME and issued to its MLME to request the allocation of a new GTS or the deallocation of an existing GTS. Each allocated GTS shall have an associated identifier with which it may be referenced.

A new GTS may be allocated by issuing this primitive with GTSId equal to 0 and GTSCharacteristics specifying the desired characteristics of the GTS.

An existing GTS may be deallocated by issuing this primitive with GTSId equal to the identifier of the GTS to deallocate and GTSCharacteristics equal to 0.

1.1.7.1.3 Effect on receipt

On receipt of this primitive, the MLME of a device shall generate a GTS request command (See subclause 1.3.3.1) if the specified GTS identifier is either equal to 0 or known to the MLME and send it to the PAN coordinator. If the GTS identifier is equal to 0, the GTS request command shall be sent with a GTS request indicating GTS allocation. Otherwise, the GTS request command shall be sent with a GTS request indicating GTS deallocation.

If the specified GTS identifier is not known to the MLME, it shall not generate a GTS request command. In this case, the MLME shall issue the MLME-GTS.confirm primitive containing a status code of INVALID_GTS_ID.

The SecurityEnable parameter specifies whether security is to be applied to the GTS request command frame. If the SecurityEnable parameter is set to FALSE, the MLME shall set the security enabled subfield of the frame control field to 0 (See subclause 1.2.1.1.2) and not perform any security operations on the frame. If the SecurityEnable parameter is set to TRUE, the MLME shall set the security enabled subfield of the frame control field to 1 and obtain the key and security information, corresponding to the address of the PAN coordinator, *macCoordExtendedAddress*, from the ACL entries in the MAC PIB, as described in subclause 1.5.9.4.1. If an appropriate key could not be found in the ACL, the MLME shall discard the frame and issue the MLME-GTS.confirm primitive with a status of UNAVAILABLE_KEY. If an appropriate key was found in the ACL, the MLME shall use it to apply security to the frame, according to the security information found in the ACL (See subclause 1.5.9.4). If any other error occurs during the secure processing of the frame, the MLME shall discard the frame and issue the MLME-GTS.confirm primitive with a status of FAILED_SECURITY_CHECK.

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

To transmit the GTS request command frame, the MLME shall first enable the transmitter by issuing the PLME-SET-TRX-STATE.request primitive with a state of TX_ON to the PHY layer. On receipt of the PLME-SET-TRX-STATE.confirm primitive with a status of either SUCCESS or TX_ON, the GTS request command frame shall then be transmitted by issuing the PD-DATA.request primitive. Finally, on receipt of the PD-DATA.confirm primitive, the MLME shall enable the receiver by issuing the PLME-SET-TRX-STATE.request primitive with a state of RX_ON to the PHY layer in preparation for the acknowledgement.

If the MLME successfully transmits a GTS request command, the MLME shall expect an acknowledgment in return. If this does not occur the GTS request command frame shall be retried. If an acknowledgement is not received after *aMaxFrameRetries* attempts, the MLME shall issue the MLME-GTS.confirm primitive with a status of NO_ACK.

If the MLME of the device successfully transmits a GTS request command, the MLME shall wait for *aResponseWaitTime* symbols for the GTS allocation command to become available. After this time, the

MLME shall attempt to extract the GTS allocation command from the PAN coordinator. If the MLME of the device does not extract a frame from the PAN coordinator it shall issue the MLME-GTS.confirm primitive with a status of NO_DATA

If the MLME of the device extracts a frame from the PAN coordinator it shall then issue the MLME-GTS.confirm primitive with a status equal to the contents of the GTS status field in the GTS allocation command (See subclause 1.3.3.2.3).

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

1.1.7.2 MLME-GTS.confirm

This primitive reports the results of a request to allocate a new GTS or deallocate an existing GTS.

1.1.7.2.1 Semantics of the service primitive

The primitive shall provide the following interface:

```
MLME-GTS.confirm              (
                              GTSId,
                              GTSLength,
                              status
                              )
```

Table 18 specifies the parameters for the MLME-GTS.confirm primitive.

Table 18—MLME-GTS.confirm parameters

Name	Type	Valid Range	Description
GTSId	Integer	0-7	The identifier of the GTS being confirmed. If the GTS allocation failed for any reason, this value shall be set to 0.
GTSLength	Integer	0-(*aNumSuperframeSlots-1)*	The length (in superframe slots) of the allocated GTS or 0 if the GTS was deallocated. If no GTS was allocated due to insufficient space in the CAP, this field will contain the maximum number of superframe slots that can be allocated to a GTS.
status	Integer	The value of the GTS status field of the GTS allocation command (See subclause 1.3.3.2.3), INVALID_GTS_ID, CHANNEL_ACCESS_FAILURE, NO_ACK, NO_DATA, UNAVAILABLE_KEY, FAILED_SECURITY_CHECK or INVALID_PARAMETER.	The status of the GTS request.

1.1.7.2.2 When generated

This primitive is generated by the MLME and issued to its DME on reception of the GTS allocation command (See subclause 1.3.3.2) following a previous MLME-GTS.request primitive.

If the GTS allocation was successful and the device requested a transmit GTS, the MLME shall set *macTxGTSId* to the identifier returned in the GTS allocation command. Similarly, if the GTS allocation was successful and the device requested a receive GTS, the MLME shall set *macRxGTSId* to the identifier returned in the GTS allocation command.

If the request to allocate or deallocate a GTS was successful, this primitive shall return a status indicating a successful GTS operation, as contained in the status field of the GTS allocation command. Otherwise, the status parameter shall indicate either an error code from the received GTS allocation command or an error code of INVALID_GTS_ID, CHANNEL_ACCESS_FAILURE, NO_ACK, NO_DATA, UNAVAILABLE_KEY, FAILED_SECURITY_CHECK or INVALID_PARAMETER. The reasons for these status values are fully described in subclause 1.1.7.1.3.

1.1.7.2.3 Effect on receipt

On receipt of this primitive the DME is notified of the result of its request to allocate or deallocate a GTS. If the request was successful, the status parameter shall indicate a successful GTS operation, as contained in the status field of the GTS allocation command. Otherwise, the status parameter shall indicate the error.

1.1.7.3 MLME-GTS.indication

This primitive indicates that a GTS has been allocated or that a previously allocated GTS has been deallocated.

1.1.7.3.1 Semantics of the service primitive

The primitive shall provide the following interface:

```
MLME-GTS.indication          (
                             DevAddrMode,
                             DevAddress,
                             GTSId,
                             GTSCharacteristics,
                             ReasonCode,
                             SecurityUse,
                             ACLEntry
                             )
```

Table 19 specifies the parameters for the MLME-GTS.indication primitive.

1.1.7.3.2 When generated

This primitive is generated by the MLME of the PAN coordinator to its DME whenever a GTS is allocated or deallocated following the reception of a GTS request command (See subclause 1.3.3.1) by the MLME. The MLME of the PAN coordinator also generates this primitive when a GTS deallocation is initiated by the PAN coordinator itself. In the case where the ReasonCode parameter is GTS_DEALLOC and the GTSId parameter is equal to 0, all GTSs have been deallocated.

Table 19—MLME-GTS.indication parameters

Name	Type	Valid Range	Description
DevAddrMode	Integer	1-2	The device addressing mode for this primitive corresponding to the received GTS request command frame. This value can take one of the following values: 1 = 8-bit short address. 2 = 16-bit short address.
DevAddress	Device address	As specified by the DevAddrMode parameter.	The address of the device that has been allocated a GTS. This parameter is undefined for GTS deallocations.
GTSId	Integer	0-7	The identifier of the GTS being allocated or deallocated. If this value is equal to 0 and the reason code is GTS_DEALLOC, all associated GTSs have been deallocated.
GTSCharacteristics	bitmap	(See subclause 1.3.3.1.3)	The specification of the GTS.
ReasonCode	Enumeration	GTS_ALLOC, GTS_DEALLOC	The reason for the indication.
SecurityUse	Boolean	TRUE or FALSE	An indication of whether the received data frame is using security. This value is set to TRUE if the security enable subfield was set to 1 or FALSE if the security enabled subfield was set to 0.
ACLEntry	Boolean	TRUE or FALSE	The parameter is set to TRUE if the sender of the MAC command frame was found in the ACL. Otherwise, the parameter is set to FALSE.

This primitive is generated by the MLME of a device and issued to its DME on receipt of a GTS allocation command (See subclause 1.3.3.2) when the PAN coordinator has made an independent decision to deallocate a GTS, i.e. the MLME of the device did not issue an MLME-GTS.request primitive prior to the receipt of the GTS allocation command. In this case, the reason code shall be GTS_DEALLOC. If the GTSId parameter is equal to 0, all GTSs associated with the device have been deallocated.

1.1.7.3.3 Effect on receipt

On receipt of this primitive the DME is notified of the allocation or deallocation of a GTS.

1.1.7.4 GTS management message sequence charts

Figure 5 and Figure 6 illustrate the sequence of messages necessary for successful GTS management. The first depicts the case for which the device initiates either the GTS allocation or deallocation procedure. The second depicts the case for which the PAN coordinator initiates the GTS deallocation procedure.

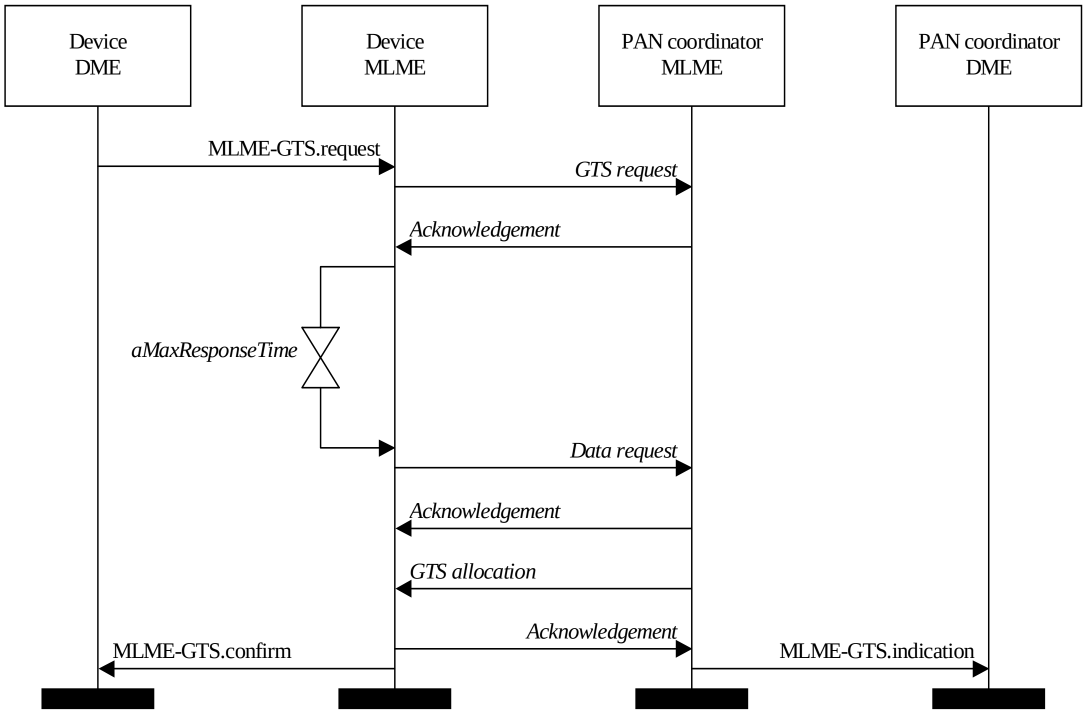


Figure 5—Message sequence chart for GTS allocation/deallocation initiated by a device

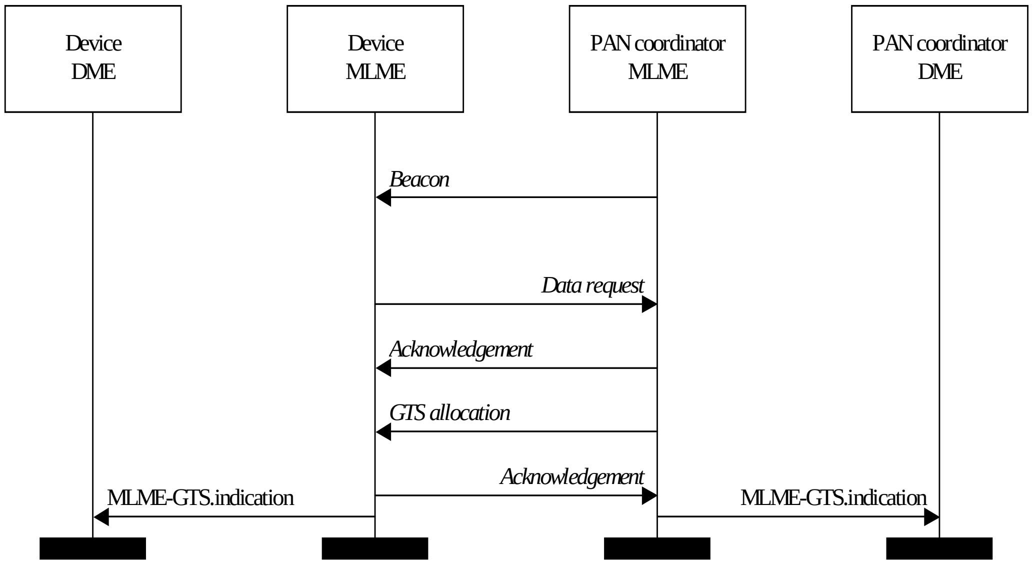


Figure 6—Message sequence chart for GTS deallocation initiated by a PAN coordinator

1.1.8 Primitives for orphan notification

These primitives define how a coordinator can issue a notification of an orphaned device.

These primitives are optional for an RFD.

1.1.8.1 MLME-ORPHAN.indication

This primitive allows the MLME of a coordinator to notify the DME of the presence of an orphaned device.

1.1.8.1.1 Semantics of the service primitive

This primitive shall provide the following interface:

```
MLME-ORPHAN.indication          (
                                OrphanAddress,
                                SecurityUse,
                                ACLEntry
                                )
```

Table 23 specifies the parameters for the MLME-ORPHAN.indication primitive.

Table 20—MLME-ORPHAN.indication parameters

Name	Type	Valid Range	Description
OrphanAddress	Device address	Extended 64-bit, IEEE address	The address of the orphaned device.
SecurityUse	Boolean	TRUE or FALSE	An indication of whether the received data frame is using security. This value is set to TRUE if the security enable subfield was set to 1 or FALSE if the security enabled subfield was set to 0.
ACLEntry	Boolean	TRUE or FALSE	The parameter is set to TRUE if the sender of the MAC command frame was found in the ACL. Otherwise, the parameter is set to FALSE.

1.1.8.1.2 When generated

This primitive is generated by the MLME of a coordinator and issued to its DME on receipt of an orphan notification command (See subclause 1.3.2.3).

1.1.8.1.3 Effect on receipt

The effect on receipt of this primitive is that the DME is notified of the orphaned device. The DME shall then determine if the device was previously associated and issue the MLME-ORPHAN.response primitive to the MLME with its decision.

The decision as to whether the device was previously associated to the coordinator and the response shall occur within a time of *aResponseWaitTime* symbols. If the device was previously associated with the coordinator, then it shall send the MLME-ORPHAN.response primitive with the AssociatedMember parameter set to TRUE and the ShortAddress parameter set to the corresponding short address allocated to the orphaned device. If the device was not previously associated with the coordinator, then it shall send the MLME-ORPHAN.response primitive with the AssociatedMember parameter set to FALSE.

1.1.8.2 MLME-ORPHAN.response

This primitive allows the DME of a coordinator to respond to the MLME-ORPHAN.indication primitive.

1.1.8.2.1 Semantics of the service primitive

This primitive shall provide the following interface:

```
MLME-ORPHAN.response          (
                              OrphanAddress,
                              ShortAddress,
                              AssociatedMember,
                              SecurityEnable
                              )
```

Table 21 specifies the parameters for the MLME-ORPHAN.response primitive.

Table 21—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 short address allocated to the orphaned device if it is associated with this coordinator. The special short address 0x00fe indicates that no short address was allocated, and the device shall use its 64-bit extended address in all communications. If the device was not associated with this coordinator, this field shall contain the value 0x00ff and shall be ignored on receipt.
AssociatedMember	Boolean	TRUE or FALSE	TRUE if the orphaned device is associated with this coordinator, FALSE otherwise.
SecurityEnable	Boolean	TRUE or FALSE	TRUE if security is enabled for this transfer or FALSE otherwise.

1.1.8.2.2 When generated

This primitive is generated by the DME and issued to its MLME when it reaches a decision as to whether the orphaned device indicated in the MLME-ORPHAN.indication primitive is associated.

1.1.8.2.3 Effect on receipt

If the AssociatedMember parameter is set to TRUE, the orphaned device is associated with the coordinator. In this case, the MLME shall generate and send the coordinator realignment command (See subclause 1.3.2.5) to the orphaned device containing the value of the ShortAddress field. This command shall be 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 shall be ignored. If the orphaned device does not receive the coordinator realignment command following its orphan notification within *aResponseWaitTime* symbols, it shall assume it is not associated to any coordinator in range.

The SecurityEnable parameter specifies whether security is to be applied to the coordinator realignment command frame. If the SecurityEnable parameter is set to FALSE, the MLME shall set the security enabled subfield of the frame control field to 0 (See subclause 1.2.1.1.2) and not perform any security operations on the frame. If the SecurityEnable parameter is set to TRUE, the MLME shall set the security enabled subfield of the frame control field to 1 and obtain the key and security information, corresponding to the ExtendedAddress parameter, from the ACL entries in the MAC PIB, as described in subclause 1.5.9.4.1. If an appropriate key could not be found in the ACL, the MLME shall discard the frame and issue the MLME-SECURITY-ERROR.indication primitive with a reason code of UNAVAILABLE_KEY. If an appropriate key was found in the ACL, the MLME shall use it to apply security to the frame, according to the security information found in the ACL (See subclause 1.5.9.4). If any other error occurs during the secure processing of the frame, the MLME shall discard the frame and issue the MLME-SECURITY-ERROR.indication primitive with a reason code of FAILED_SECURITY_CHECK.

To transmit the coordinator realignment command frame, the MLME shall first enable the transmitter by issuing the PLME-SET-TRX-STATE.request primitive with a state of TX_ON to the PHY layer. On receipt of the PLME-SET-TRX-STATE.confirm primitive with a status of either SUCCESS or TX_ON, the coordinator realignment command frame shall then be transmitted by issuing the PD-DATA.request primitive. Finally, if an acknowledgment was requested, the MLME shall enable the receiver on receipt of the PD-DATA.confirm primitive by issuing the PLME-SET-TRX-STATE.request primitive with a state of RX_ON to the PHY layer in preparation for the acknowledgement.

1.1.8.3 Orphan notification message sequence chart

Figure 7 illustrates the sequence of messages necessary for a coordinator to issue a notification of an orphaned device.

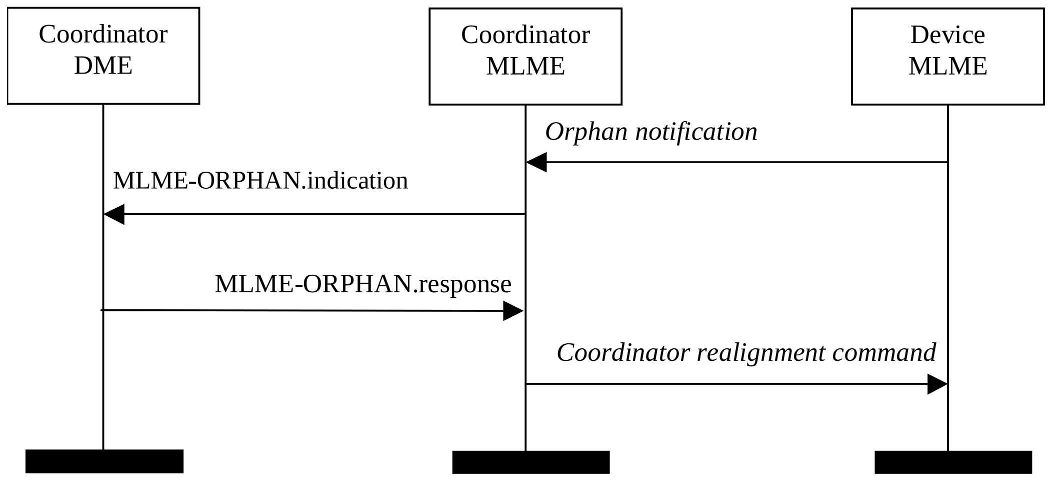


Figure 7—Message sequence chart for orphan notification

1.1.9 Primitive for resetting the MAC sublayer

This primitive specifies how to reset the MAC sublayer to its default values.

All devices shall implement this primitive.

1.1.9.1 MLME-RESET.request

This primitive allows the DME to request that the MLME performs a reset operation.

1.1.9.1.1 Semantics of the service primitive

This primitive shall provide the following interface:

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

Table 23 specifies the parameters for the MLME-RESET.request primitive.

Table 22—MLME-RESET.request parameters

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.

1.1.9.1.2 When generated

This primitive is generated by the DME and issued to its MLME to request a reset of the MAC sublayer to its initial conditions. The MLME-RESET.request primitive shall be issued prior to the use of the MLME-START.request or the MLME-ASSOCIATE.request primitives. If this primitive is issued to the MLME of an associated device or coordinator, any required disassociation attempts using the MLME-DISASSOCIATE.request primitive shall be made a priori at the discretion of the DME.

1.1.9.1.3 Effect on receipt

On receipt of this primitive, the MLME shall issue the PLME-SET-TRX-STATE.request primitive with a state of TRX_OFF. On receipt of the PLME-SET-TRX-STATE.confirm primitive, the MAC sublayer shall then be 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.

1.1.10 Primitive for specifying the receiver enable time

This primitive defines how a device can enable or disable the receiver at a given time.

All devices shall implement this primitive.

1.1.10.1 MLME-RX-ENABLE.request

This primitive contains timing information that tells the device when to enable or disable its receiver, in order to schedule a data transfer between itself and another device.

1.1.10.1.1 Semantics of the service primitive

This primitive shall provide the following interface:

```
MLME-RX-ENABLE.request          (
                                EnableRx,
                                DeferTime
                                )
```

Table 23 specifies the parameters for the MLME-RX-ENABLE.request primitive.

Table 23—MLME-RX-ENABLE.request parameters

Name	Type	Valid Range	Description
EnableRx	Boolean	TRUE or FALSE	TRUE to enable the receiver or FALSE to disable the receiver.
DeferTime	Integer	0x000000-0xffffff	The number of symbols from the start of the superframe before the specified receiver state is put into operation. The precision of this value shall be a minimum of 20-bits, with the lowest four bits being the least significant. For non-beacon enabled PANs this value shall be set to 0x000000.

1.1.10.1.2 When generated

This primitive is generated by the DME and issued to the MLME to enable or disable the receiver at a time relative to the start of the superframe. If no superframe exists, the receiver state shall be changed immediately.

1.1.10.1.3 Effect on receipt

On a beacon enabled PAN and provided there is sufficient time, the MLME shall request that the PHY change the state of the receiver so that the change of state occurs at the specified number of symbols from the start of the superframe. Sufficient time shall be available if the current number of symbols from the start of the superframe plus *aTurnaroundTime* is less than or equal to the requested defer time. If sufficient time is not available, the MLME shall request that the PHY change the state of the receiver at the specified defer time from the start of the next superframe.

On a non-beacon enabled PAN, the MLME shall request that the PHY change the state of the receiver immediately.

The MLME shall request that the PHY change the state of the receiver by issuing the PLME-SET-TRX-STATE.request primitive with a state of RX_ON if EnableRx is equal to TRUE or TRX_OFF if EnableRx is equal to FALSE.

1.1.10.2 Message sequence chart for changing the state of the receiver

Figure 8 illustrates the sequence of messages necessary for changing the state of the receiver. In a) there is sufficient time to change the state of the receiver in the current superframe. In b) sufficient time is not available in the current superframe and the state of the receiver is changed in the next superframe.

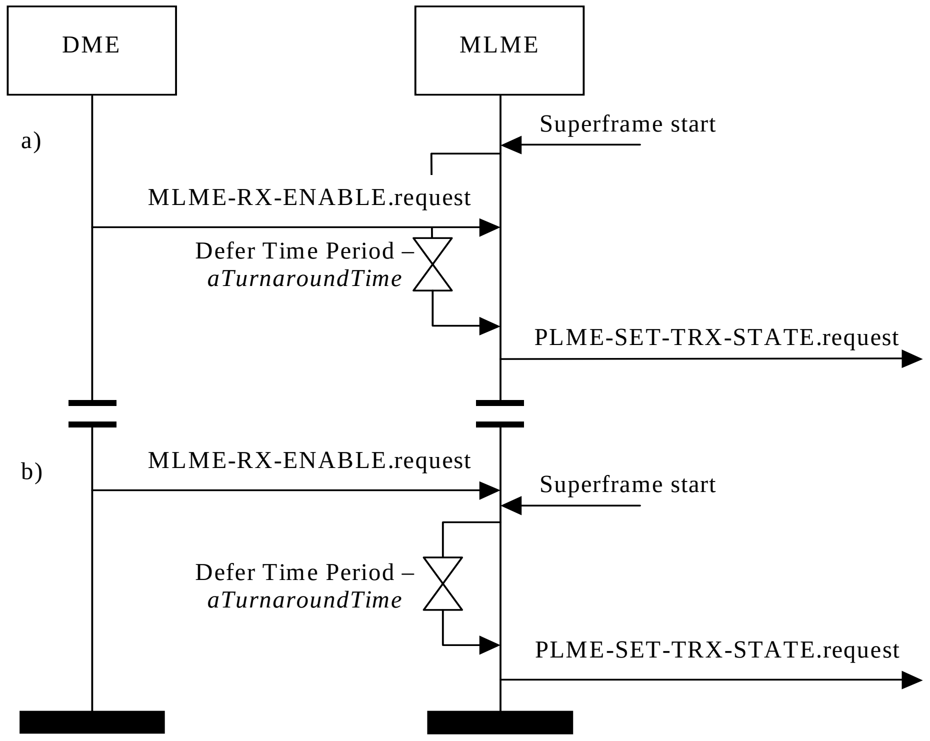

DME
MLME
a)
Superframe start
MLME-RX-ENABLE.request
Defer Time Period – *aTurnaroundTime*
PLME-SET-TRX-STATE.request
b)
MLME-RX-ENABLE.request
Superframe start
Defer Time Period – *aTurnaroundTime*
PLME-SET-TRX-STATE.request

Figure 8—Message sequence chart for changing the state of the receiver

1.1.11 Primitives for channel scanning

These primitives define how a device can determine the energy usage or the presence or absence of PANs in a communications channel.

All devices shall implement these primitives.

1.1.11.1 MLME-SCAN.request

This 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 a coordinator transmitting beacon frames containing a specific PAN identifier or search for all coordinators transmitting beacon frames within the radio sphere of the scanning device.

1.1.11.1.1 Semantics of the service primitive

This primitive shall provide the following interface:

```
MLME-SCAN.request            (
                             ScanType,
                             ScanChannels,
                             ScanDuration
                             )
```

Table 24 specifies the parameters for the MLME-SCAN.request primitive.

Table 24—MLME-SCAN.request parameters

Name	Type	Valid Range	Description
ScanType	Integer	0-3	Indicates the type of scan performed: 0 = energy detection scan (FFD only), 1 = active scan (FFD only), 2 = passive scan, 3 = orphan scan.
ScanChannels	Bitmap	32 bit field	The five most significant bits (b_{27}, ... , b_{31}) are reserved. The 27 least significant bits (b_0, b_1, ... b_{26}) shall indicate which channels are to be scanned (1=scan, 0=do not scan) for each of the 27 valid channels (See subclause 6.1.2).
ScanDuration	Integer	0-14	A value used to calculate the length of time to spend scanning each channel. The time spent scanning each channel is (*aBaseSuperframeDuration* * (2^n + 1)) symbols, where *n* is the value of the ScanDuration parameter.

1.1.11.1.2 When generated

This primitive is generated by the DME and issued to its MLME to initiate a channel scan to search for activity within the radio sphere of influence of the device. This primitive can be used to perform an energy detection 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 beacon frame containing a specific PAN identifier. The MAC functional description subclause describes each type of scan in detail (See subclause 1.5.2.1).

Energy detection or active scans can be performed prior to an FFD starting operation as a PAN coordinator. Passive scans can be performed prior to selecting a PAN for association. Orphan scans can be performed to attempt to locate a specific coordinator with which communication has been lost.

All devices shall be capable of performing passive scans and orphan scans; energy detection scans and active scans are optional for an RFD.

1.1.11.1.3 Effect on receipt

When the MLME receives this primitive, it shall initiate a scan in all channels specified in the ScanChannels parameter. The MLME shall suspend all beacon transmissions for the duration of the scan.

An energy detection scan allows a device to obtain a measure of the peak energy in each requested channel. The energy detection scan shall be performed on each channel by the MLME repeatedly issuing the PLME-ED.request primitive to the PHY layer until (*aBaseSuperframeDuration* * (2^n + 1)) symbols, where *n* is the value of the ScanDuration parameter, have elapsed. The MLME shall note the maximum energy measurement and move on to the next channel in the channel list. The energy detection scan shall terminate when each of the channels specified in the channel list have been scanned.

An active scan is used by an FFD to locate all coordinators transmitting beacon frames within its radio sphere of influence. The active scan shall be performed on each channel by the MLME first sending a beacon request command (See subclause 1.3.2.4). The MLME shall then enable the receiver and record all beacons heard, until either the number of beacons found equals an implementation specified limit or until (*aBaseSuperframeDuration* * (2^n + 1)) symbols, where *n* is the value of the ScanDuration parameter, have elapsed. Where possible, the scan shall be repeated on each channel and shall terminate when the number of beacons found equals an implementation specified limit or every channel in the set of available channels has been scanned.

A passive scan, like an active scan, is used to locate all coordinators transmitting beacon frames within the radio sphere of influence of the scanning device; the difference is that the passive scan is a receive only operation and does not transmit the beacon request command. The passive scan shall be performed on each channel by the MLME enabling its receiver and recording all beacons heard, until either the number of beacons found equals an implementation specified limit or until (*aBaseSuperframeDuration* * (2^n + 1)) symbols, where *n* is the value of the ScanDuration parameter, have elapsed. Where possible, the scan shall be repeated on each channel and terminate when the number of beacons found equals an implementation specified limit or every channel in the set of available channels has been scanned.

An orphan scan is used to locate the coordinator with which the scanning device had previously associated. The orphan scan shall be performed on each channel by the MLME first sending an orphan notification command (See subclause 1.3.2.3). The MLME shall then enable its receiver for at most (*aBaseSuperframeDuration* * (2^n + 1) symbols, where *n* is the value of the ScanDuration parameter. If the device receives a coordinator realignment command within this time, the MLME shall disable its receiver. Otherwise, the scan shall be repeated on the next channel in the channel list. The scan shall terminate when the device receives a coordinator realignment command (See subclause 1.3.2.5) or every channel in the set of available channels has been scanned.

The results of an energy detection scan shall be recorded in a list of energy detection values and reported to the DME through the MLME-SCAN.confirm primitive with a status of SUCCESS.

The results of an active or passive scan shall be recorded in a set of PANDescriptor values and reported to the DME through the MLME-SCAN.confirm primitive. If no beacons were found, the MLME-SCAN.confirm primitive shall contain a null set of PANDescriptor values and a status of NO_BEACON. Otherwise, the MLME-SCAN.confirm primitive will contain the set of PANDescriptor values found, a list of unscanned channels and a status of SUCCESS.

The results of an orphan scan shall be reported to the DME through the MLME-SCAN.confirm primitive. If successful, the MLME-SCAN.confirm primitive shall contain a status of SUCCESS. If the device did not receive a coordinator realignment command, the MLME-SCAN.confirm primitive shall contain a status of NO_BEACON. In either case, the PANDescriptor and EnergyDetectList parameters of the MLME-SCAN.confirm primitive shall be null.

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

1.1.11.2 MLME-SCAN.confirm

This primitive reports the success of the channel scan request.

1.1.11.2.1 Semantics of the service primitive

This primitive shall provide the following interface:

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

Table 25 specifies the parameters for the MLME-SCAN.confirm primitive.

Table 25—MLME-SCAN.confirm parameters

Name	Type	Valid Range	Description
status	Enumeration	SUCCESS, NO_BEACON, INVALID_PARAMETER	The status of the scan request.
ScanType	Integer	0-3	Indicates if the type of scan performed: 0 = energy detection scan (FFD only), 1 = active scan (FFD only), 2 = passive scan, 3 = orphan scan.
UnscannedChannels	Bitmap	32 bit field	Indicates which channels given in the request were not completely scanned (1=scan incomplete, 0=scanned or not requested). This parameter is only valid for passive or active scans.
ResultListSize	Integer	Implementation specific.	The number of elements returned in the appropriate result lists. This value shall be 0 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 energy detection scan. This parameter is null for active, passive and orphan scans.
PANDescriptorList	List of PANDescriptor values	See Table 14	The list of PAN descriptors, one for each beacon found during an active or passive scan. This parameter is null for energy detection and orphan scans.

1.1.11.2.2 When generated

This primitive is generated by the MLME and issued to its DME 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 shall be null. If the MLME-SCAN.request primitive requested an energy detection or orphan scan, the PANDescriptorList parameter shall be null. If the MLME-SCAN.request primitive requested an orphan scan, the ResultListSize parameter shall be set to 0.

This primitive shall return a status of either SUCCESS, indicating that the requested scan was successful, or an error code of NO_BEACON or INVALID_PARAMETER. The reasons for these status values are fully described in subclause 1.1.11.1.3.

1.1.11.2.3 Effect on receipt

On receipt of this primitive, the DME is notified of the results of the scan procedure. If the requested scan was successful, the status parameter shall be set to SUCCESS. Otherwise, the status parameter shall indicate the error.

1.1.11.3 Channel scan message sequence charts

Figure 52 and Figure 55 (See subclause 1.7) illustrate the sequence of messages necessary to perform an energy detection scan and a passive scan, respectively. These figures include steps taken by the PHY layer.

1.1.12 Security management primitive

This primitive defines how the MLME communicates security related events to the DME.

All devices shall implement this primitive.

1.1.12.1 MLME-SECURITY-ERROR.indication

This primitive allows the MLME to indicate a failed security processing operation.

1.1.12.1.1 Semantics of the service primitive

This primitive shall provide the following interface:

```
MLME-SECURITY-ERROR.indication (
                                PANId,
                                SrcAddrMode,
                                SrcAddr,
                                DstAddrMode,
                                DstAddr,
                                ReasonCode
                                )
```

Table 26 specified the parameters for the MLME-SECURITY-ERROR.indication primitive.

Table 26—MLME-SECURITY-ERROR.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	0-3	The source addressing mode for this primitive. This value can take one of the following values: 0 = no address (ignore addressing fields). 1 = 8-bit short address. 2 = 16-bit short address. 3 = 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.
DstAddrMode	Integer	0-3	The destination addressing mode for this primitive. This value can take one of the following values: 0 = no address (ignore addressing fields). 1 = 8-bit short address. 2 = 16-bit short address. 3 = 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.
ReasonCode	Enumeration	UNAVAILABLE_KEY, FRAME_TOO_LONG, FAILED_SECURITY_CHECK	The reason for the security error.

1.1.12.1.2 When generated

This primitive is generated by the MLME and issued to its DME upon receipt of a message, either from the DME or from another device, requiring security whenever one of the following two conditions occurs: the appropriate key is not available in the ACL or the device is unable to apply security to the frame.

If the appropriate key is not available in the ACL, the MAC sublayer shall discard the MSDU and issue this primitive with a reason code of UNAVAILABLE_KEY. If the appropriate key is available in the ACL but the secure processing of the frame results in a frame longer than *aMaxMACFrameSize*, the MLME shall issue this primitive with a status of FRAME_TOO_LONG. If the appropriate key is available in the ACL but an error occurs during the secure processing of the frame, the MAC sublayer shall discard the MSDU and issue this primitive with a reason code of FAILED_SECURITY_CHECK.

1.1.12.1.3 Effect on receipt

On receipt of this primitive, the DME is notified that an error has occurred during the secure processing of an outgoing or incoming frame.

1.1.13 Primitives for writing MAC PIB attributes

These primitives define how MAC PIB attributes may be written.

All devices shall implement these primitives.

1.1.13.1 MLME-SET.request

This primitive attempts to write the given value to the indicated MAC PIB attribute.

1.1.13.1.1 Semantics of the primitive

This primitive shall provide the following interface:

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

Table 27 specifies the parameters for the MLME-SET.request primitive.

Table 27—MLME-SET.request parameters

Name	Type	Valid Range	Description
PIBAttribute	Enumeration	See Table 44 and Table 45	The identifier of the MAC PIB attribute to write.
PIBAttributeValue	Various	Attribute Specific (See Table 44 and Table 45)	The value to write to the indicated MAC PIB attribute.

1.1.13.1.2 When generated

This primitive is generated by the DME and issued to its MLME to write the indicated MAC PIB attribute.

1.1.13.1.3 Effect on receipt

On receipt of this primitive, the MLME shall attempt to write the given value to the indicated MAC PIB attribute in its database. If the PIBAttribute parameter specifies an attribute that is not found in the database (See Table 44 and Table 45), the MLME shall issue the MLME-SET.confirm primitive with a status of UNSUPPORTED_ATTRIBUTE. If the PIBAttibuteValue parameter specifies a value that is out of the valid range for the given attribute, the MLME shall issue the MLME-SET.confirm primitive with a status of INVALID_PARAMETER.

If the requested MAC PIB attribute is successfully written, the MLME shall issue the MLME-SET.confirm primitive with a status of SUCCESS.

1.1.13.2 MLME-SET.confirm

This primitive reports the results of an attempt to write a value to a MAC PIB attribute.

1.1.13.2.1 Semantics of the service primitive

This primitive shall provide the following interface:

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

Table 28 specifies the parameters for the MLME-SET.confirm primitive.

Table 28—MLME-SET.confirm Parameters

Name	Type	Valid Range	Description
status	Enumeration	SUCCESS, UNSUPPORTED_ATTRIBUTE, INVALID_PARAMETER	The result of the request to write the MAC PIB attribute.
PIBAttribute	Enumeration	See Table 44 and Table 45	The identifier of the MAC PIB attribute that was written.

1.1.13.2.2 When generated

This primitive is generated by the MLME and issued to its DME in response to an MLME-SET.request primitive. This primitive shall return a status of either SUCCESS, indicating that the requested value was written to the indicated MAC PIB attribute, or an error code of UNSUPPORTED_ATTRIBUTE or INVALID_PARAMETER. The reasons for these status values are fully described in subclause 1.1.13.1.3.

1.1.13.2.3 Effect on receipt

On receipt of this primitive, the DME is notified of the result of its request to set the value of a MAC PIB attribute. If the requested value was written to the indicated MAC PIB attribute, the status parameter shall be set to SUCCESS. Otherwise, the status parameter shall indicate the error.

1.1.14 Primitives for starting beacon transmissions

These primitives define how a device can start transmitting beacons. This may be used by an FFD to either initiate a PAN or to begin transmitting beacons on an already existing PAN, facilitating device discovery.

These primitives are optional for an RFD.

1.1.14.1 MLME-START.request

This primitive makes a request for the device to begin transmitting beacons.

1.1.14.1.1 Semantics of the service primitive

This primitive shall provide the following interface:

```
MLME-START.request              (
                                PANId,
                                LogicalChannel,
                                BeaconOrder,
                                SuperframeOrder,
                                PANCoordinator,
                                SecurityEnable
                                )
```

Table 29 specifies the parameters for the MLME-START.request primitive.

1.1.14.1.2 When generated

This primitive is generated by the DME and issued to its MLME to request that a device begin transmitting beacons.

1.1.14.1.3 Effect on receipt

If this primitive is received when *macShortAddress* is set to 0x00ff, the MLME shall issue the MLME-START.confirm primitive with a status of NO_SHORT_ADDRESS.

On receipt of this primitive, the MLME shall set *macBeaconOrder* and *macSuperframeOrder* to the values of the BeaconOrder and SuperframeOrder parameters, respectively. In the case where the PANCoordinator parameter is set to TRUE, the MLME shall update *macPANId* and *phyCurrentChannel* with the values of the PANId and LogicalChannel parameters, respectively, before beginning operation as the PAN coordinator of a new PAN. In the case where the PANCoordinator parameter is set to FALSE, the MAC sublayer shall simply begin transmitting beacons on the PAN with which it has associated.

The address used by the coordinator in its beacon frames shall be determined by the current value of *macShortAddress*, which shall be set by the DME before issuing this primitive.

The SecurityEnable parameter specifies whether security is to be applied to all following beacon frames. If the SecurityEnable parameter is set to FALSE, the MAC sublayer shall set the security enabled subfield of the frame control field to 0 (See subclause 1.2.1.1.2) and not perform any security operations on the frame. If the SecurityEnable parameter is set to TRUE, the MAC sublayer shall set the security enabled subfield of the frame control field to 1 and obtain the key and security information, corresponding to the broadcast address, from the ACL entries in the MAC PIB, as described in subclause 1.5.9.4.1. If an appropriate key could not be found in the ACL, the MAC sublayer shall discard the frame and issue the MLME-START.confirm primitive with a status of UNAVAILABLE_KEY. If an appropriate key was found in the ACL, the

Table 29—MLME-START.request parameters

Name	Type	Valid Range	Description
PANId	Integer	0x0000-0xffff	The PAN identifier to be used by the beacon.
LogicalChannel	Integer	Selected from the available logical channels supported by the PHY	The logical channel on which to start transmitting beacons.
BeaconOrder	Integer	0-15	This specifies how often the beacon is to be transmitted. The BeaconOrder, *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 will not transmit a beacon.
SuperframeOrder	Integer	0-*BO* or 15	This specifies the length of the active portion of the superframe, including the beacon frame. The SuperframeOrder, *SO*, and the superframe duration, *SD*, are related as follows: for $0 \leq SO \leq \mathrm{BO} \leq 14$, $SD = aBaseSuperframeDuration * 2^{SO}$ symbols. If SO = 15, the superframe will not be active after the beacon.
PANCoordinator	Boolean	TRUE or FALSE	If this value is TRUE, the device shall become the PAN coordinator of a new PAN. If this value is FALSE, the device shall begin transmitting beacons on the PAN with which it is associated.
SecurityEnable	Boolean	TRUE or FALSE	TRUE if security is enabled for beacon transmissions or FALSE otherwise.

MAC sublayer shall use it to apply security to the frame, according to the security information found in the ACL (See subclause 1.5.9.4). If the length of the resulting frame is longer than *aMaxMACFrameSize*, the MLME shall issue the MLME-START.confirm primitive with a status of FRAME_TOO_LONG. If any other error occurs during the secure processing of the frame, the MAC sublayer shall discard the frame and issue the MLME-START.confirm primitive with a status of FAILED_SECURITY_CHECK.

To transmit the beacon frame, the MLME shall first enable the transmitter by issuing the PLME-SET-TRX-STATE.request primitive with a state of TX_ON to the PHY layer. On receipt of the PLME-SET-TRX-STATE.confirm primitive with a status of either SUCCESS or TX_ON, the beacon frame shall then be transmitted by issuing the PD-DATA.request primitive. Finally, on receipt of the PD-DATA.confirm primitive and if the active portion of the superframe extends beyond the beacon frame transmission (See subclause 1.5.1.1), the MLME of the coordinator shall enable the receiver by issuing the PLME-SET-TRX-STATE.request primitive with a state of RX_ON to the PHY layer. If the active portion of the superframe ends after the beacon frame transmission, the receiver shall not be enabled.

On completion of this procedure, the MLME shall respond with the MLME-START.confirm primitive. If the attempt to begin transmitting beacons was successful, the status parameter shall be set to SUCCESS. If any parameter is not supported or is out of range, the status parameter shall be set to INVALID_PARAMETER.

1.1.14.2 MLME-START.confirm

This primitive reports the results of the attempt to begin transmitting beacons.

1.1.14.2.1 Semantics of the service primitive

This primitive shall provide the following interface:

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

Table 30 specifies the parameters for the MLME-START.confirm primitive.

Table 30—MLME-START.confirm parameters

Name	Type	Valid Range	Description
status	Enumeration	SUCCESS, NO_SHORT_ADDRESS, UNAVAILABLE_KEY, FRAME_TOO_LONG, FAILED_SECURITY_CHECK or INVALID_PARAMETER	The result of the attempt to begin beacon transmission.

1.1.14.2.2 When generated

This primitive is generated by the MLME and issued to its DME in response to an MLME-START.request primitive. This primitive shall return a status of either SUCCESS, indicating that the MAC sublayer has begun to transmit beacons, or an error code of NO_SHORT_ADDRESS, UNAVAILABLE_KEY, FRAME_TOO_LONG, FAILED_SECURITY_CHECK or INVALID_PARAMETER. The reasons for these status values are fully described in subclause 1.1.14.1.3.

1.1.14.2.3 Effect on receipt

On receipt of this primitive, the DME is notified of the result of its request to start beaconing. If the MAC sublayer has begun to transmit beacons, the status parameter shall be set to SUCCESS. Otherwise, the status parameter shall indicate the error.

1.1.14.3 Message sequence chart for starting beacon transmissions

Figure 9 illustrates the sequence of messages necessary for initiating beacon transmissions in an FFD. Figure 51 (See subclause 1.7) illustrates the sequence of messages necessary for a PAN coordinator to start beaconing on a new PAN; this figure includes steps taken by the PHY layer.

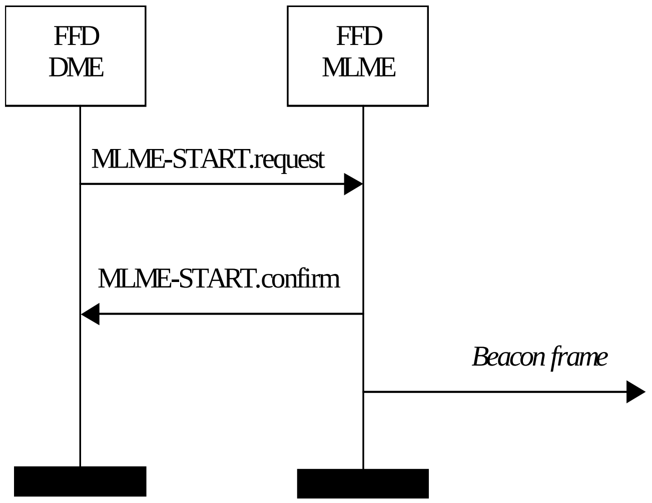


Figure 9—Message sequence chart for starting beacon transmissions

1.1.15 Primitives for synchronizing with a coordinator

These primitives define how synchronization with a coordinator may be achieved and how a loss of synchronization is communicated to the DME.

All devices shall implement these primitives.

1.1.15.1 MLME-SYNC.request

This primitive requests to synchronize with the coordinator by acquiring and, if specified, tracking its beacons.

1.1.15.1.1 Semantics of the service primitive

This primitive shall provide the following interface:

```
MLME-SYNC.request              (
                               TrackBeacon
                               )
```

Table 31 specifies the parameters for the MLME-SYNC.request primitive.

Table 31—MLME-SYNC.request parameters

Name	Type	Valid Range	Description
TrackBeacon	Boolean	TRUE or FALSE	TRUE if the beacon is to be tracked or FALSE if the beacon is to be located only once.

1.1.15.1.2 When generated

This primitive is generated by the DME of a device on a beacon enabled PAN and issued to its MLME to synchronize with the coordinator.

1.1.15.1.3 Effect on receipt

If this primitive is received by the MLME on a beacon enabled PAN, it shall enable its receiver and search for the current network beacon. If the TrackBeacon parameter is equal to TRUE, the MLME shall 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 shall locate the beacon but not continue to track it.

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

1.1.15.2 MLME-SYNC-LOSS.indication

This primitive indicates the loss of synchronization with a coordinator.

1.1.15.2.1 Semantics of the service primitive

This primitive shall provide the following interface:

```
MLME-SYNC-LOSS.indication          (
                                   LossReason
                                   )
```

Table 32 specifies the parameters for the MLME-SYNC-LOSS.indication primitive.

Table 32—MLME-SYNC-LOSS.indication parameters

Name	Type	Valid Range	Description
LossReason	Enumeration	PAN_ID_CONFLICT, COORDINATOR_LOST, REALIGNMENT or BEACON_LOST	The reason that synchronization was lost.

1.1.15.2.2 When generated

This primitive is generated by the MLME of a device and issued to its DME in the event of a loss of synchronization with the coordinator. It is also generated by the MLME of a PAN coordinator and issued to its DME in the event of a PAN ID conflict.

If a device has detected a PAN identifier conflict and communicated it to the coordinator, the MLME shall issue this primitive with a loss reason of PAN_ID_CONFLICT. Similarly if a PAN coordinator receives a PAN ID conflict notification command (See subclause 1.3.2.2), the MLME shall issue this primitive with a loss reason of PAN_ID_CONFLICT.

If a device has failed to communicate with the coordinator for *aMaxCommunicationsFailures* successive times, the MLME shall issue this primitive with a loss reason of COORDINATOR_LOST. A single communications failure occurs when a device transaction fails to reach the coordinator, i.e. no acknowledgement is received to any of the *aMaxFrameRetries* transmission attempts.

If a device has received the coordinator realignment command (See subclause 1.3.2.5), the MLME shall issue this primitive with a loss reason of REALIGNMENT.

If a device has lost the beacon following an MLME-SYNC.request primitive, either initially or during tracking, the MLME shall issue this primitive with a loss reason of BEACON_LOST.

1.1.15.2.3 Effect on receipt

On receipt of this primitive, the DME is notified of a loss of synchronization.

1.1.15.3 Message sequence chart for synchronizing with a coordinator

Figure 10 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 is requested as described in subclause 1.5.6.3. 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 which expires just before the expected time of the next beacon. The MLME also checks for any data pending in the coordinator for the device whenever a beacon frame is received.

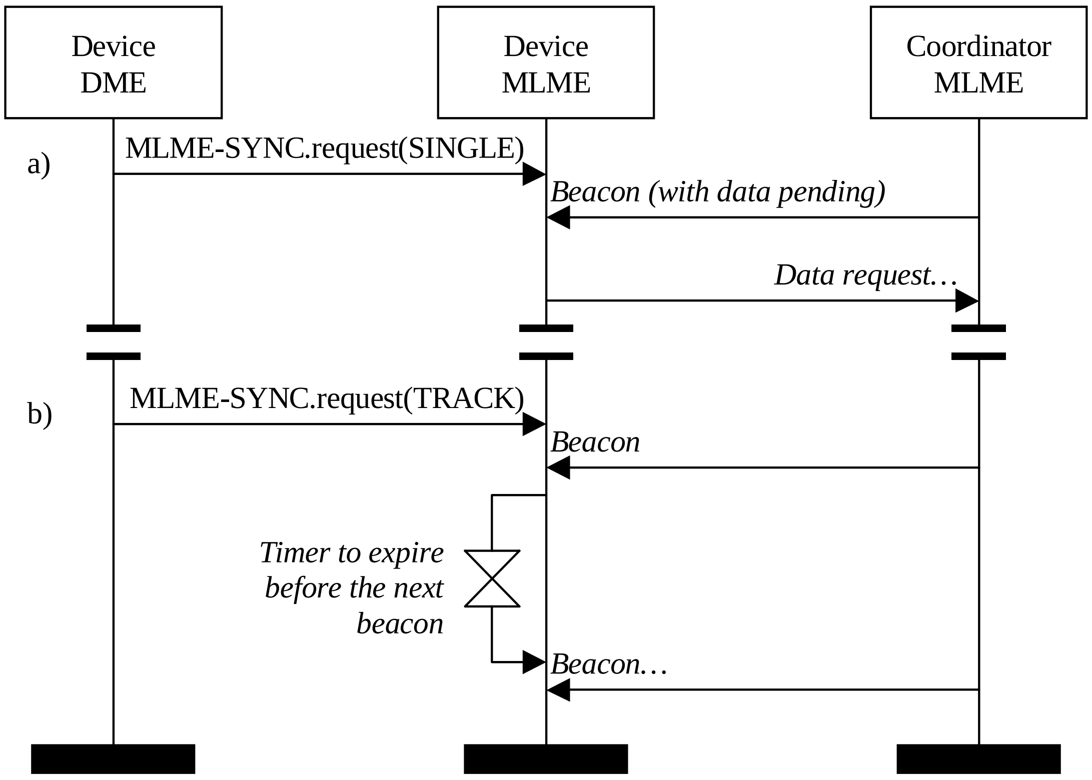


Figure 10—Message sequence chart for synchronizing to a coordinator in a beacon enabled PAN

1.1.16 Primitives for requesting data from a coordinator

These primitives define how to request data from a coordinator.

All devices shall implement these primitives.

1.1.16.1 MLME-POLL.request

This primitive prompts the device to request data from the coordinator.

1.1.16.1.1 Semantics of the service primitive

This primitive shall provide the following interface:

```
MLME-POLL.request              (
                               SecurityEnable
                               )
```

Table 31 specifies the parameter for the MLME-POLL.request primitive.

Table 33—MLME-POLL.request parameters

Name	Type	Valid Range	Description
SecurityEnable	Boolean	TRUE or FALSE	TRUE if security is enabled for this transfer or FALSE otherwise.

1.1.16.1.2 When generated

This primitive is generated by the DME and issued to its MLME when data is to be requested from a coordinator.

1.1.16.1.3 Effect on receipt

On receipt of this primitive the MLME shall generate and send a data request command to the coordinator with which it is currently associated.

The SecurityEnable parameter specifies whether security is to be applied to the data request command frame. If the SecurityEnable parameter is set to FALSE, the MLME shall set the security enabled subfield of the frame control field to 0 (See subclause 1.2.1.1.2) and not perform any security operations on the frame. If the SecurityEnable parameter is set to TRUE, the MLME shall set the security enabled subfield of the frame control field to 1 and obtain the key and security information, corresponding to *macCoordExtendedAddress*, from the ACL entries in the MAC PIB, as described in subclause 1.5.9.4.1. If an appropriate key could not be found in the ACL, the MLME shall discard the frame and issue the MLME-POLL.confirm primitive with a status of UNAVAILABLE_KEY. If an appropriate key was found in the ACL, the MLME shall use it to apply security to the frame, according to the security information found in the ACL (See subclause 1.5.9.4). If any other error occurs during the secure processing of the frame, the MLME shall discard the frame and issue the MLME-POLL.confirm primitive with a status of FAILED_SECURITY_CHECK.

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

To transmit the data request frame, the MLME shall first enable the transmitter by issuing the PLME-SET-TRX-STATE.request primitive with a state of TX_ON to the PHY layer. On receipt of the PLME-SET-TRX-STATE.confirm primitive with a status of either SUCCESS or TX_ON, the data request command frame shall then be transmitted by issuing the PD-DATA.request primitive. Finally, on receipt of the PD-DATA.confirm primitive, the MLME shall enable the receiver by issuing the PLME-SET-TRX-STATE.request primitive with a state of RX_ON to the PHY layer in preparation for the acknowledgement.

If the MLME successfully transmits a data request command, the MLME shall expect an acknowledgment in return. If this does not occur the data request command frame shall be retried. If an acknowledgement is not received after *aMaxFrameRetries* attempts, the MLME shall issue the MLME-POLL.confirm primitive with a status of NO_ACK.

If an acknowledgement is received, the MLME shall request that the PHY enable its receiver if the frame pending subfield of the acknowledgement frame is set to 1. If the frame pending subfield of the acknowledgement frame is set to 0, the MLME shall 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 shall issue the MLME-POLL.confirm primitive with a status of NO_DATA. If a frame is received from the coordinator with non-zero length payload, the MLME shall issue the MLME-POLL.confirm primitive with a status of SUCCESS. In this case, the data itself shall be indicated to the next higher layer using the MCPS-DATA.indication primitive (see subclause 1.1.1.3).

If a frame is not received within *aMaxFrameResponseTime* symbols even though the acknowledgement to the data request command has its frame pending subfield set to 1, the MLME shall 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 shall issue the MLME-POLL.confirm primitive with a status of INVALID_PARAMETER.

1.1.16.2 MLME-POLL.confirm

This primitive reports the results of a request to poll the coordinator for data.

1.1.16.2.1 Semantics of the service primitive

The primitive shall provide the following interface:

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

Table 18 specifies the parameters for the MLME-POLL.confirm primitive.

Table 34—MLME-POLL.confirm parameters

Name	Type	Valid Range	Description
status	Integer	SUCCESS, CHANNEL_ACCESS_FAILURE, NO_ACK, NO_DATA, UNAVAILABLE_KEY, FAILED_SECURITY_CHECK or INVALID_PARAMETER	The status of the data request.

1.1.16.2.2 When generated

This primitive is generated by the MLME and issued to its DME in response to an MLME-POLL.request primitive. If the request was successful, the status parameter shall be equal to SUCCESS indicating a successful poll for data. Otherwise, the status parameter shall indicate an error code of CHANNEL_ACCESS_FAILURE, NO_ACK, NO_DATA, UNAVAILABLE_KEY, FAILED_SECURITY_CHECK or INVALID_PARAMETER. The reasons for these status values are fully described in subclause 1.1.16.1.3.

1.1.16.2.3 Effect on receipt

On receipt of this primitive, the DME is notified of the status of the procedure to request data from the coordinator.

1.1.16.3 Message sequence chart for requesting data from a coordinator

Figure 11 illustrates the sequence of messages necessary for a device to request data from a coordinator. In

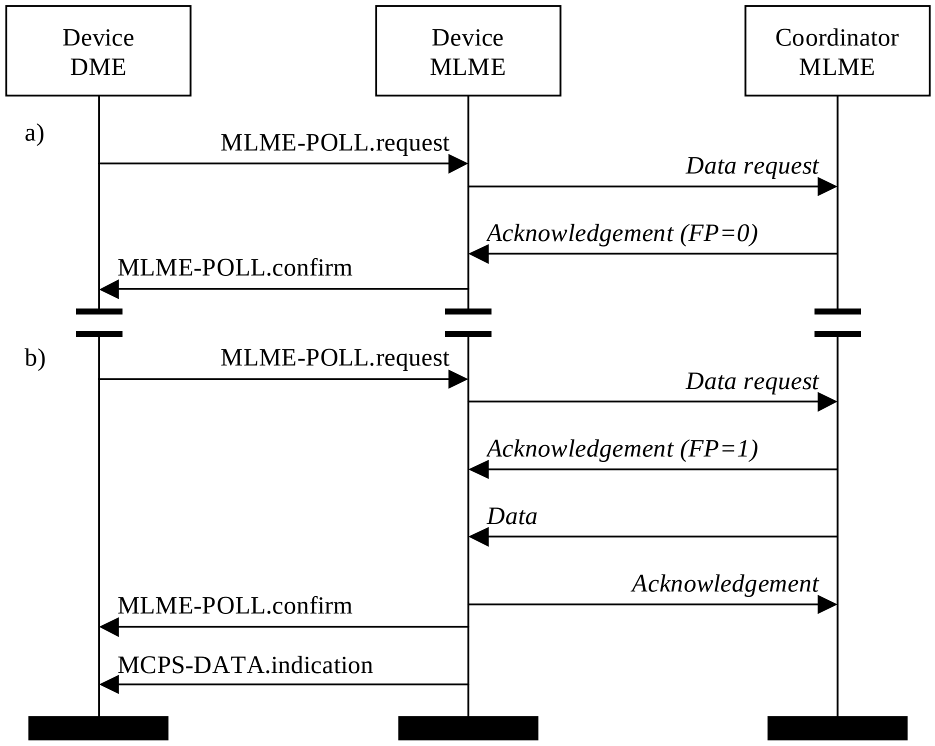


Figure 11—Message sequence chart for requesting data from the coordinator

both cases, a poll request is issued to the MLME, which then sends a data request command to the coordinator. In a), the corresponding acknowledgement has the frame pending (FP) subfield set to 0 and the MLME issues the poll request confirmation immediately. In b), the corresponding acknowledgement has the frame pending (FP) subfield set to 1 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.

1.1.17 MAC enumeration description

This subclause explains the meaning of the enumerations used in the primitives defined in the MAC sublayer specification. Table 35 shows a description of the MAC enumeration values.

1.2 MAC frame formats

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

a) A MAC header, which comprises frame control, sequence number and address 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 MAC footer, which contains a frame check sequence.

Table 35—MAC enumerations description

Enumeration	Value	Description
SUCCESS	0x00	The requested operation was completed successfully. For a transmission request, this indicates a successful transmission.
-	0x01-0x7f	Reserved for MAC command status and reason code values.
-	0x80-0xdf	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.
COORDINATOR_LOST	0xe2	Synchronization has been lost due to a coordinator communications failure.
FAILED_SECURITY_CHECK	0xe3	The received frame induces a failed security check according to the security suite.
FRAME_TOO_LONG	0xe4	The frame resulting from secure processing has a length that is greater than *aMACMaxFrameSize*.
GTS_ALLOC	0xe5	The specified GTS has been allocated.
GTS_DEALLOC	0xe6	The specified GTS has been deallocated.
INVALID_GTS	0xe7	The requested GTS transmission failed due to the specified GTS either not having a transmit GTS direction or not being defined.
INVALID_GTS_ID	0xe8	A device has requested to deallocate a GTS with an invalid GTS identifier.
INVALID_PARAMETER	0xe9	A parameter in the primitive is out of the valid range.
NO_ACK	0xea	No acknowledgment was received after *aMaxFrameRetries*.
NO_BEACON	0xeb	A scan operation failed to find any network beacons.

Table 35—MAC enumerations description

Enumeration	Value	Description
NO_DATA	0xec	No response data was available following a request.
NO_SHORT_ADDRESS	0xed	The operation failed due to a short address not being allocated.
PAN_ID_CONFLICT	0xee	A PAN identifier conflict has been detected and communicated to the PAN coordinator.
REALIGNMENT	0xef	A coordinator realignment command has been received.
SUCCESS	0xf0	The requested operation was completed successfully. For a transmission request, this indicates a successful transmission.
TRANSACTION_EXPIRED	0xf1	The transaction has expired and its information discarded.
TRANSACTION_OVERFLOW	0xf2	There is no capacity to store the transaction.
TRANSMISSION_IN_PROGRESS	0xf3	A transmission was requested before a previous transmission had completed.
UNAVAILABLE_KEY	0xf4	The appropriate key is not available in the ACL.
UNSUPPORTED_ATTRIBUTE	0xf5	A SET/GET request was issued with the identifier of a PIB attribute that is not supported.

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.

The MAC 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 left-most bit is transmitted first in time.

1.2.1 General MAC frame format

The MAC frame format is composed of a MAC header, a MAC payload and a MAC footer. The fields of the MAC header 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 12.

1.2.1.1 Frame control field

The frame control field is 16-bits in length and contains information defining the frame type, addressing fields and other control flags. The frame control field shall be formatted as illustrated in Figure 13.

Figure 12—General MAC frame format

Octets: 2	1	0/2	0/1/2/8	0/2	0/1/2/8	variable	2
Frame control	Sequence number	Destination PAN identifier	Destination address	Source PAN identifier	Source address	Frame payload	Frame check sequence
		Addressing fields					
MAC header						MAC payload	MAC footer

Figure 13—Format of the frame control field

Bits: 0-2	3	4	5	6-9	10-11	12-13	14-15
Frame type	Security enabled	Frame pending	Ack. req.	Reserved	Dest. addressing mode	Reserved	Source addressing mode

1.2.1.1.1 Frame type subfield

The frame type subfield is three bits in length and shall be set to one of the non-reserved values listed in Table 36.

Table 36—Values of the frame type subfield

Frame type value b_2 b_1 b_0	Description
000	Beacon
001	Data
010	Acknowledgment
011	MAC command
100-111	Reserved.

1.2.1.1.2 Security enabled subfield

The security enabled subfield is one bit in length and shall be set to 0 if the frame is not cryptographically protected by the MAC sublayer. If the security enabled subfield is set to 1, the frame shall be protected using the keys stored in the MAC PIB for the security relationship indicated by the current frame. The cryptographic operations used to protect the frame are defined by the security suite selected for that security relationship. If no security suite is defined for that relationship, the security enabled subfield shall be set to 0.

1.2.1.1.3 Frame pending subfield

The frame pending subfield is one bit in length and shall be set to 1 if the device sending the frame has additional data to send to the recipient following the current transfer. If more data is pending, the recipient shall retrieve it by sending another data request command to the device. If the device sending the frame does not have any more data for the recipient, this subfield shall be set to 0.

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

At all other times it shall be set to 0 on transmission and ignored on reception.

1.2.1.1.4 Acknowledgment request subfield

The acknowledgment request subfield is one 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 field is set to 1, the recipient device shall send an acknowledgement frame after determining that the frame is valid (see subclause 1.5.6.2 for a list of requirements necessary for frame validity). If this field is set to 0, the recipient device shall not send an acknowledgement frame after determining that the frame is valid.

1.2.1.1.5 Destination addressing mode subfield

The destination addressing mode subfield is two bits in length and shall be set to one of the values listed in Table 37.

If this subfield is equal to 0 and the frame type subfield does not specify that this is an acknowledgement or beacon frame, the source addressing mode subfield shall be non-zero, implying that the frame is directed to the coordinator with the PAN identifier as specified in the source PAN identifier field.

Table 37—Possible values of the destination/source addressing mode subfield

Addressing mode value **b_1 b_0**	**Description**
00	PAN identifier and address field are not present.
01	Address field contains an 8-bit, short address.
10	Address field contains a 16-bit, short address.
11	Address field contains a 64-bit, extended address.

1.2.1.1.6 Source addressing mode subfield

The source addressing mode subfield is two bits in length and shall be set to one of the values listed in Table 37.

If this subfield is equal to 0 and the frame type subfield does not specify that this is an acknowledgement frame, the destination addressing mode subfield shall be non-zero, implying that the frame has originated from the coordinator with the PAN identifier as specified in the destination PAN identifier field.

1.2.1.2 Sequence number field

The sequence number field is 8-bits in length and specifies a unique sequence identifier for the frame.

In the case of a beacon frame, the sequence number field shall specify a beacon sequence number (BSN). Each coordinator shall store a 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. The coordinator shall copy the value of the *macBSN* attribute into the sequence number field of a beacon frame and shall increment it by one after each beacon is sent.

In the case of 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. Each device shall store a 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. The device shall copy the value of the *macDSN* attribute into the sequence number field of a data frame or MAC command frame for each data transfer.

If an acknowledgement is not requested, the MAC sublayer of the originating device shall increment *macDSN* by one after successfully transmitting the data or MAC command frame. If an acknowledgment is requested, the recipient device shall copy the DSN received in the data or MAC command frame into the DSN field of the corresponding acknowledgment frame. Following the successful reception of the acknowledgment frame by the originating device, the MAC sublayer shall increment *macDSN* by one. If the acknowledgement was not received after *aAckWaitDuration* symbols, the MAC sublayer of the originating device shall retransmit the frame using the same value of *macDSN*. If the acknowledgement was still not received after *aMaxFrameRetries* retransmissions, the MAC sublayer of the originating device shall increment *macDSN* by one.

1.2.1.3 Destination PAN identifier field

The destination PAN identifier field is 16-bits 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 non-zero.

1.2.1.4 Destination address field

The destination address field is either 8-, 16- or 64-bits in length, according to the value specified in the destination addressing mode subfield of the frame control field (See subclause 1.2.1.1.5), and specifies the address of the intended recipient of the frame. An 8-bit value of 0xff in this field shall represent the broadcast short address which shall be accepted as a valid 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 non-zero.

1.2.1.5 Source PAN identifier field

The source PAN identifier field is 16-bits 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 subfield of the frame control field is non-zero.

The PAN identifier of a device is initially determined during association on a PAN but may change following a PAN identifier conflict resolution (See subclause 1.5.2.2).

1.2.1.6 Source address field

The source address field is either 8-, 16- or 64-bits in length, according to the value specified in the destination addressing mode subfield of the frame control field (See subclause 1.2.1.1.6), 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 non-zero.

1.2.1.7 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 1 in the frame control field, the frame payload is protected as defined by the security suite selected for that relationship.

1.2.1.8 Frame check sequence field

The frame check sequence (FCS) field is 16-bits in length and contains a 16-bit ITU-T cyclic redundancy code (CRC). The FCS is calculated over the MAC header and MAC payload parts of the frame.

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

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

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

— Let $M(x) = b_0 x^{k-1} + b_1 x^{k-2} + \ldots + b_{k-2}x + b_{k-1}$ 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, $R(x) = r_0 x^{15} + r_1 x^{14} + \ldots + r_{14}x + r_{15}$.
— The FCS field is given by the remainder coefficients.

As an example, consider an acknowledgement frame with no payload and the following 3-byte MAC header:

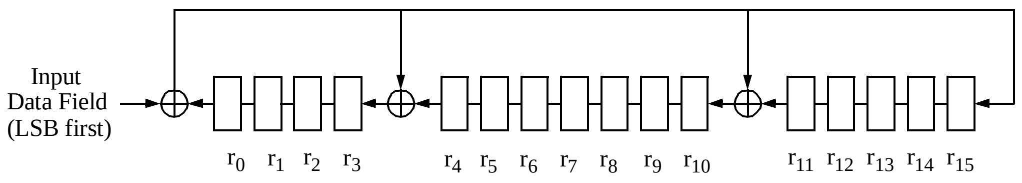
0100 0000 0000 0000 0101 0110 [leftmost bit (b_0) transmitted first in time]
b_0...b_{23}

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 14.

CRC-16 Generator Polynomial: $G(x) = x^{16} + x^{12} + x^5 + 1$


Input
Data Field
(LSB first)
r_0 r_1 r_2 r_3 r_4 r_5 r_6 r_7 r_8 r_9 r_{10} r_{11} r_{12} r_{13} r_{14} r_{15}

1. Initialize the remainder register (r_0 through r_{15}) to zero.
2. Shift MAC header 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 such that r_0 is transmitted first.

Figure 14—Typical FCS implementation

1.2.2 Format of individual frame types

There are four defined frame types: beacon, data, acknowledgment and MAC command. Each of these frame types will be discussed in the following subclauses.

1.2.2.1 Beacon frame format

The beacon frame shall be formatted as illustrated in Figure 15.

Figure 15—Beacon frame format

Octets: 2	1	3/4/10	2	1	variable	variable	2
Frame control	Sequence number	Addressing fields	Superframe specification	Pending address specification	Address list	Beacon payload	Frame check sequence
MAC header			MAC payload				MAC footer

The order of the fields of the beacon frame shall conform to the order of the general MAC frame as illustrated in Table 12.

1.2.2.1.1 Beacon frame MAC header field

The MAC header field 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 field shall contain the value which indicates a beacon frame, as shown in Table 36 and the source addressing mode field shall be set as appropriate for the address of the beacon. If security is used for the beacon, the security enabled subfield shall be set to one. All other fields shall be set to zero and ignored on reception.

The sequence number field shall contain the current value of *macBSN*. Once the beacon has been sent, the transmitting device shall increment *macBSN* by one.

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.

1.2.2.1.2 Superframe specification field

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

Figure 16—Format of the superframe specification field

Bits: 0-3	4-7	8-11	12-13	14	15
Beacon order	Superframe order	Final CAP slot	Reserved	PAN Coordinator	Association permit

The beacon order subfield is four bits in length and shall specify the transmission interval of the beacon. If *BO* is the value of the beacon order, the beacon interval, *BI*, shall be computed as follows: *BI* = *aBaseSuperframe Duration* * 2^{BO} symbols, where $0 \le BO \le 14$.

The superframe order subfield is four bits in length and shall specify the length of time during which the superframe is active (receiver enabled), including the beacon frame transmission time. The coordinator shall only interact with its PAN during the active superframe. If *SO* is the value of the superframe order, the superframe duration, *SD*, shall be computed as follows. For $0 \le SO \le BO \le 14$, *SD = aBaseSuperframeDuration* * 2^{SO} symbols. If *SO* = 15, the superframe shall not be active following the transmission of the beacon.

The final CAP slot subfield is four bits in length and specifies the final superframe slot utilized by the contention access period (CAP). The duration of the CAP, as implied by this subfield, shall be greater than or equal to that specified by *aMinCAPLength*.

The PAN coordinator subfield is one bit in length and shall be set to 1 if the beacon frame is being transmitted by the PAN coordinator. Otherwise the PAN coordinator subfield shall be set to 0.

The association permit subfield is one bit in length and shall be set to 1 if *macAssociationPermit* is set to TRUE (the coordinator is accepting association on the PAN). The association permit bit shall be set to 0 if the coordinator is currently not accepting association requests on its network.

1.2.2.1.3 Pending address specification field

The pending address specification field shall be formatted as illustrated in Figure 17.

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

Figure 17—Format of the pending address specification field

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

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

1.2.2.1.4 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 address of the devices that currently have messages pending with the coordinator.

The maximum number of addresses pending shall be limited to seven and may be comprised of both short and extended addresses. All pending short addresses shall appear first in the list followed by any extended addresses.

1.2.2.1.5 Beacon frame payload field

The payload field is an optional sequence of up to *aMaxBeaconMSDULength* octets specified to be transmitted in the beacon frame by the DME. If *macBeaconMSDULength* is non-zero, the set of octets contained in *macBeaconMSDU* shall be copied into this field.

If security is required on an outgoing beacon frame, the sequence of octets in the payload field shall be processed according to the security suite corresponding to *aExtendedAddress.*

If the security enabled subfield of the frame control field of an incoming frame is set to 0, the beacon payload field shall contain the intended sequence of octets to be passed to the next higher layer. If the security enabled subfield of the frame control field of an incoming frame is set to 1, the device shall process the beacon payload field according to the security suite corresponding to *macCoordExtendedAddress* in order to determine the intended sequence of octets to be passed to the next higher layer.

If a device receives a beacon with a payload field present, it shall indicate it to the DME and then process the information contained in the superframe specification and address list fields. If the MAC sublayer receives a beacon without a payload field present, it shall immediately interpret and process the information contained in the superframe specification and address list fields.

1.2.2.2 Data frame format

The data frame shall be formatted as illustrated in Figure 18.

The order of the fields of the data frame shall conform to the order of the general MAC frame as illustrated in Table 12.

1.2.2.2.1 Data frame MAC header field

The MAC header field 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.

Figure 18—Data frame format

Octets: 2	1	(See subclause 1.2.2.2.1)	variable	2
Frame control	Sequence number	Addressing fields	Data payload	Frame check sequence
MAC header			MAC payload	MAC footer

In the frame control field, the frame type subfield shall contain the value which indicates a data frame, as shown in Table 36. All other subfields shall be set appropriately according to the intended use of the data frame.

The sequence number field shall contain the current value of *macDSN*. When the data transaction has completed (i.e. including the reception of any acknowledgment), the device shall increment *macDSN* by one.

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

1.2.2.2.2 Data payload field

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

If security is required on an outgoing data frame, the sequence of octets in this field shall be processed according to the security suite corresponding to either the destination address, if present, or *macCoordExtendedAddress*, if the destination address field is not present.

If the security enabled subfield of the frame control field of an incoming frame is set to 0, the data payload field shall contain the intended sequence of octets to be passed to the next higher layer. If the security enabled subfield of the frame control field of an incoming frame is set to 1, the device shall process the data payload field according to its selected security suite in order to determine the intended sequence of octets to be passed to the next higher layer.

1.2.2.3 Acknowledgment frame format

The acknowledgment frame shall be formatted as illustrated in Figure 19.

Figure 19—Acknowledgment frame format

Octets: 2	1	2
Frame control	Sequence number	Frame check sequence
MAC header		MAC footer

The order of the fields of the acknowledgment frame shall conform to the order of the general MAC frame as illustrated in Table 12.

1.2.2.3.1 Acknowledgment frame MAC header field

The MAC header field 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 which indicates an acknowledgment frame, as shown in Table 36. The frame pending subfield shall be set according to whether the device sending the acknowledgement frame has more data pending for the recipient. All other subfields shall be set to 0 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.

1.2.2.4 MAC Command Frame Format

The MAC command frame shall be formatted as illustrated in Figure 20.

Figure 20—MAC command frame format

Octets: 2	**1**	**(See subclause 1.2.2.4.1)**	**1**	**variable**	**2**
Frame control	Sequence number	Addressing fields	Command frame identifier	Command payload	Frame check sequence
MAC header			MAC payload		MAC footer

The order of the fields of the MAC command frame shall conform to the order of the general MAC frame as illustrated in Table 12.

1.2.2.4.1 MAC command frame MAC header field

The MAC header field 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 which indicates a MAC command frame, as shown in Table 36. All other subfields shall be set appropriately according to the intended use of the MAC command frame.

The sequence number field shall contain the current value of *macDSN*. When the transaction has completed (i.e. including the reception of the acknowledgment, if requested), the device shall increment *macDSN* by one.

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

1.2.2.4.2 Command frame identifier field

The command frame identifier field identifies the MAC command being used. This field shall be set to one of the non-reserved values listed in Table 38.

1.2.2.4.3 Command payload field

The command payload field contains the MAC command itself.

If security is required on an outgoing MAC command frame, the sequence of octets in this field shall be processed according to the security suite corresponding to either the destination address, if present, or *macCoordExtendedAddress*, if the destination address field is not present.

If the security enabled subfield of the frame control field of an incoming frame is set to 0, the command payload field shall contain the intended MAC command. If the security enabled subfield of the frame control field of an incoming frame is set to 1, the device shall process the command payload field according to its selected security suite in order to determine the intended MAC command.

The format of the individual commands are described in subclause 1.3.

1.3 MAC command frames

The command frames defined by the MAC sublayer are listed in Table 38. An FFD shall be capable of transmitting and receiving all command frame types, while the requirements for an RFD are indicated in the table. MAC commands shall only be transmitted in the CAP for beacon enabled PANs or at any time for non-beacon enabled PANs.

See the following subclauses for more details on the individual commands.

Table 38—MAC command frames

Command frame identifier	Command name	RFD		Subclause
		Tx	Rx	
0x01	Association request	X		1.3.1.1
0x02	Association response		X	1.3.1.2
0x03	Disassociation notification	X	X	1.3.1.3
0x04	Data request	X		1.3.2.1
0x05	PAN ID conflict notification			1.3.2.2
0x06	Orphan notification	X		1.3.2.3
0x07	Beacon request			1.3.2.4
0x08	Coordinator realignment		X	7.3.2.5
0x09	GTS request			7.3.3.1
0x0a	GTS allocation			7.3.3.2
0x0b-0xff	Reserved			-

1.3.1 Association and disassociation

This set of commands is used to allow devices to associate with or disassociate from a PAN.

1.3.1.1 Association request command

The association request command allows a device to request association with 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 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 21.

Figure 21—Association request command format

octets: 16/17/23	1	1
MAC header fields	Command frame identifier (Table 38)	Capability information

1.3.1.1.1 MAC header fields

The fields of the MAC header portion of the general MAC frame format (See Table 12) shall be specified as follows.

The source addressing mode subfield of the frame control field shall be set to 3 (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.

If security is used for the association request command, the security enabled subfield shall be set to 1 and the frame shall be processed according to the method defined by the security suite corresponding to the destination address. Otherwise, the security enabled subfield shall be set to 0.

The frame pending subfield of the frame control field shall be set to 0 and ignored upon reception, and the acknowledgment request subfield shall be set to 1.

The destination PAN identifier field shall contain the identifier of the PAN to which to associate. The destination address field shall contain the address contained in 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 (0xffff). The source address field shall contain the value of *aExtendedAddress*.

1.3.1.1.2 Capability information field

The capability information field shall be formatted as illustrated in Figure 22.

The alternate PAN coordinator subfield is one bit in length and shall be set to 1 if the device is capable of becoming a PAN coordinator. Otherwise the alternate PAN coordinator subfield shall be set to 0.

The device type subfield is one bit in length and shall be set to 1 if the device is an FFD. Otherwise the device type subfield shall be set to 0, indicating an RFD.

Figure 22—Capability information field format

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

The power source subfield is one bit in length and shall be set to 1 if the device is receiving power from the alternating current mains. Otherwise the power source subfield shall be set to 0.

The receiver on when idle subfield is one bit in length and shall be set to 1 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 0.

The security capability subfield is one bit in length and shall be set to 1 if the device is capable of sending and receiving MAC frames secured using the security suite specified in subclause 1.6. Otherwise the security capability subfield shall be set to 0.

The allocate address subfield is one bit in length and shall be set to 1 if the device wishes the coordinator to allocate a short address as a result of the association procedure. The short address shall be either 8 or 16 bits in length, depending on which short addressing mode is being used by the coordinator; the mode being used by the coordinator can be inferred from the short address of the coordinator, as described in Table 47. If this subfield is set to 0, the special short address of 0x00fe shall be allocated to the device and returned through the association response command. In this case, the device shall only communicate on the PAN using its 64-bit extended address.

1.3.1.2 Association response command

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

This command shall only be sent by the 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 23.

Figure 23—Association response command format

octets: 23	1	2	2	1
MAC header fields	Command frame identifier (Table 38)	Short address	Coordinator short address	Association status

1.3.1.2.1 MAC header fields

The fields of the MAC header portion of the general MAC frame format (See Table 12) shall be specified as follows.

The destination addressing mode and source addressing mode subfields of the frame control field shall each be set to 3 (64-bit extended addressing).

If security is used for the association response command, the security enabled subfield shall be set to 1 and the frame shall be processed according to the method defined by the security suite corresponding to the destination address. Otherwise, the security enabled subfield shall be set to 0.

The frame pending subfield of the frame control field shall be set to 0 and ignored upon reception, and the acknowledgment request subfield shall be set to 1.

The destination and source PAN identifier fields shall contain the value of *macPANId*. The destination address field shall contain the extended address of the device requesting association. The source address field shall contain the value of *aExtendedAddress*.

1.3.1.2.2 Short address field

The short address field is 16-bits in length.

If the coordinator was not able to associate this device to its PAN, this field shall be set to 0x00ff 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.

If the coordinator supports the 8-bit short addressing mode, the least significant octet shall contain the short address of the device and the most significant octet shall contain all zeros. If the coordinator supports the 16-bit short addressing mode, the short address shall be greater than 0x00ff.

A short address field value equal to 0x00fe 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 only communicate on the PAN using its 64-bit extended address.

1.3.1.2.3 Coordinator short address

The coordinator short address field is 16-bits in length and contains the short address that is currently being used by the coordinator. If the coordinator set up or associated with a PAN supporting the 8-bit short addressing mode, the least significant octet of this field shall contain the address of the coordinator and the most significant octet shall contain all zeros. If the coordinator set up or associated with a PAN supporting the 16-bit short addressing mode, this entire field shall contain the address of the coordinator and the address shall be greater than 0x00ff.

1.3.1.2.4 Association status field

The association status field is 8-bits in length and shall contain one of the non-reserved values listed in Table 39.

1.3.1.3 Disassociation notification command

Either the 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 24.

Table 39—Valid values of the association status field

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

Figure 24—Disassociation notification command format

octets: 23	1	1
MAC header fields	Command frame identifier (Table 38)	Disassociation reason

1.3.1.3.1 MAC header fields

The fields of the MAC header portion of the general MAC frame format (See Table 12) shall be specified as follows.

The destination addressing mode and source addressing mode subfields of the frame control field shall both be set to 3 (64-bit extended addressing).

If security is used for the disassociation notification command, the security enabled subfield shall be set to 1 and the frame shall be processed according to the method defined by the security suite corresponding to the destination address. Otherwise, the security enabled subfield shall be set to 0.

The frame pending subfield of the frame control field shall be set to 0 and ignored upon reception, and the acknowledgment request subfield shall be set to 1.

The destination and source PAN identifier fields shall contain the value of *macPANId*. If the coordinator wants an associated device to leave the PAN, then the destination address field shall contain the extended 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 *macCoordExtendedAddress*. The source address field shall contain the value of *aExtendedAddress*.

1.3.1.3.2 Disassociation reason field

The disassociation reason field is 8-bits in length and shall contain one of the non-reserved values listed in Table 40.

1.3.2 Coordinator interaction

This set of commands is used to allow devices to interact with a coordinator.

Table 40—Valid disassociation reason codes

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

1.3.2.1 Data request command

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

On a beacon enabled PAN, this command shall be sent by a device whenever *macAutoRequest* is equal to TRUE and a beacon frame is received from its coordinator indicating that data is pending for that device. 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 DME on reception of the MLME-POLL.request primitive. In addition, a device may send this command to the coordinator *aResponseWaitTime* symbols after the acknowledgement to a request command, such as an association or a GTS request.

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 25.

Figure 25—Data request command format

octets: 6/7/13	1
MAC header fields	Command frame identifier (Table 38)

1.3.2.1.1 MAC header fields

The fields of the MAC header portion of the general MAC frame format (See Table 12) shall be specified as follows.

The destination addressing mode subfield of the frame control field shall be set to 0 (destination addressing information not present). The source addressing mode subfield shall be set to 3 (64-bit extended address) if the value of *macShortAddress* is equal to 0x00fe. Otherwise, the source addressing mode subfield shall be set to the short addressing mode supported on the PAN.

+If security is used for the data request command, the security enabled subfield shall be set to 1 and the frame shall be processed according to the method defined by the security suite corresponding to *macCoordExtendedAddress*. Otherwise, the security enabled subfield shall be set to 0.

The frame pending subfield of the frame control field shall be set to 0 and ignored upon reception, and the acknowledgment request subfield shall be set to 1.

The source PAN identifier field shall contain the value of *macPANId*. The source address field shall contain the value of *aExtendedAddress* if the value of *macShortAddress* is equal to 0xfe. Otherwise, the source address field shall be set to the value of *macShortAddress*.

1.3.2.2 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.

This command is optional for an RFD.

The PAN ID conflict notification command shall be formatted as illustrated in Figure 26.

Figure 26—PAN ID conflict notification command format

octets: 23	**1**
MAC header fields	Command frame identifier (Table 38)

1.3.2.2.1 MAC header fields

The fields of the MAC header portion of the general MAC frame format (See Table 12) shall be specified as follows.

The destination addressing mode and source addressing mode subfields of the frame control field shall both be set to 3 (64-bit extended addressing).

If security is used for the PAN ID conflict notification command, the security enabled subfield shall be set to 1 and the frame shall be processed according to the method defined by the security suite corresponding to *macCoordExtendedAddress*. Otherwise, the security enabled subfield shall be set to 0.

The frame pending subfield of the frame control field shall be set to 0 and ignored upon reception, and the acknowledgment request subfield shall be set to 1.

The destination PAN identifier and source PAN identifier subfields shall each contain the value of *macPANId*. The destination address field shall contain the value of *macCoordExtendedAddress*. The source address subfield shall contain the value of *aExtendedAddress*.

1.3.2.3 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 27.

Figure 27—Orphan notification command format

octets: 16	1
MAC header fields	Command frame identifier (Table 38)

1.3.2.3.1 MAC header fields

The fields of the MAC header portion of the general MAC frame format (See Table 12) shall be specified as follows.

The source addressing mode subfield of the frame control field shall be set to 3 (64-bit extended addressing). The destination addressing mode subfield shall be set to 1 (8-bit short addressing).

If security is used for the orphan notification command, the security enabled subfield shall be set to 1 and the frame shall be processed according to the method defined by the security suite corresponding to *macCoordExtendedAddress*. Otherwise, the security enabled subfield shall be set to 0.

The frame pending and acknowledgment request subfields of the frame control field shall be set to 0 and ignored upon reception.

The destination PAN identifier and source PAN identifier subfields shall each contain the broadcast PAN identifier (0xffff). The destination address field shall contain the broadcast short address (0xff). The source address subfield shall contain the value of *aExtendedAddress*.

1.3.2.4 Beacon request command

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

This command is optional for an RFD.

The beacon request command shall be formatted as illustrated in Figure 28.

Figure 28—Beacon request command format

octets: 6	1
MAC header fields	Command frame identifier (Table 38)

1.3.2.4.1 MAC header fields

The fields of the MAC header portion of the general MAC frame format (See Table 12) shall be specified as follows.

The destination addressing mode subfield of the frame control field shall be set to 1 (8-bit short addressing), and the source addressing mode subfield shall be set to 0 (source addressing information not present).

The frame pending subfield of the frame control field shall be set to 0 and ignored upon reception, and the acknowledgment request subfield shall also be set to 0.

The destination PAN identifier subfield shall contain the broadcast PAN identifier (0xffff). The destination address field shall contain the broadcast short address (0xff).

1.3.2.5 Coordinator realignment command

The coordinator realignment command is sent by 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.

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 (PAN identifier or logical channel) change, it is broadcast to the PAN as a courtesy to any devices currently able to receive.

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 29.

Figure 29—Coordinator realignment command format

octets: 16 or 23	1	2	1	2
MAC header fields	Command frame identifier (Table 38)	PAN identifier	Logical channel	Short address

1.3.2.5.1 MAC header fields

The fields of the MAC header portion of the general MAC frame format (See Table 12) shall be specified as follows.

The destination addressing mode subfield shall be set to 3 (64-bit extended addressing) if the command is directed to an orphaned device or to 1 (8-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 3 (64-bit extended addressing).

If security is used for the coordinator realignment command directed to an orphaned device, the security enabled subfield shall be set to 1 and the frame shall be processed according to the method defined by the security suite corresponding to the destination address. Otherwise, the security enabled subfield shall be set to 0.

The frame pending subfield of the frame control field shall be set to 0 and ignored upon reception.

The acknowledgment request subfield of the frame control field shall be set to 1 if the command is directed to an orphaned device or to 0 if the command is to be broadcast to the PAN.

The destination PAN identifier field shall contain the broadcast PAN identifier (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 (0xff). The source PAN identifier field shall contain the value of *macPANId* and the source address field shall contain the value of *aExtendedAddress*.

1.3.2.5.2 PAN identifier field

The PAN identifier field is 16-bits in length and shall contain the PAN identifier which the coordinator intends to use for all future communications.

1.3.2.5.3 Logical channel field

The logical channel field is 8-bits in length and shall contain the logical channel which the coordinator intends to use for all future communications.

1.3.2.5.4 Short address field

The short address field is 16-bits in length. If the coordinator realignment command is broadcast to the PAN, this field shall be set to 0x00ff 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 device shall use to operate on the PAN. If the coordinator set up or associated with a PAN supporting the 8-bit short addressing mode, the least significant octet of this field shall contain the address of the orphaned device and the most significant octet shall contain all zeros. If the coordinator set up or associated with a PAN supporting the 16-bit short addressing mode, this entire field shall contain the address of the orphaned device and the address shall be greater than 0x00ff.

1.3.3 GTS allocation and deallocation

This set of commands is used to manage GTSs. A device can use these commands to request the allocation of a new GTS, the deallocation of an existing GTS or the allocation confirmation of an existing GTS. The PAN coordinator can use these commands to allocate a new GTS, deallocate an existing GTS or reallocate an existing GTS.

This set of commands is optional for an RFD.

1.3.3.1 GTS request command

The GTS request command is used by an associated device that is requesting the allocation of a new GTS, the deallocation of an existing GTS or the allocation confirmation of an existing GTS from the PAN coordinator. Only those devices that have a valid short address shall send this command, i.e. the value of *macShortAddress* is not equal to 0x00fe or 0x00ff.

The GTS request command shall be formatted as illustrated in Figure 30.

1.3.3.1.1 MAC header fields

The fields of the MAC header portion of the general MAC frame format (See Table 12) shall be specified as follows.

Figure 30—GTS request command format

octets: 6 or 7	**1**	**1**	**1**
MAC header fields	Command frame identifier (Table 38)	GTS control	GTS characteristics

The destination addressing mode subfield of the frame control field shall be set to 0 (destination addressing information not present), and the source addressing mode subfield shall be set to the short addressing mode supported on the PAN.

If security is used for the GTS request command, the security enabled subfield shall be set to 1 and the frame shall be protected by the method defined by the security suite corresponding to *macCoordExtendedAddress*. Otherwise, the security enabled subfield shall be set to 0.

The frame pending subfield of the frame control field shall be set to 0 and ignored upon reception, and the acknowledgment request subfield shall be set to 1.

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

1.3.3.1.2 GTS control field

The GTS control field is 8-bits in length and shall be formatted as illustrated in Figure 31.

Figure 31—The GTS control format

bits: 0-1	**2-4**	**5-7**
Request type	Reserved	GTS identifier

The request type subfield is 2 bits in length and specifies the type of GTS request being performed. This subfield shall contain one of the non-reserved values listed in Table 41.

Table 41—Valid values of the request type subfield

Request type value b_1 b_0	**Description**
00	GTS allocation
01	GTS deallocation
10	GTS confirmation
11	Reserved

The GTS identifier subfield is 3 bits in length and specifies the identifier of the GTS to which the GTS request command refers. In the case where a GTS allocation is being requested, the GTS identifier subfield shall contain the value 0. In all other cases, the GTS identifier field shall contain the non-zero identifier of a previously allocated GTS.

A PAN coordinator receiving the GTS request command with a request type of deallocate GTS or confirm GTS shall discard the GTS characteristics field.

1.3.3.1.3 GTS characteristics field

The GTS characteristics field is 8-bits in length and shall be formatted as illustrated in Figure 32.

Figure 32—The GTS characteristics format

bits: 0-3	4-6	7
GTS length	Reserved	GTS direction

The GTS length subfield is 4 bits in length and shall contain the number of superframe slots being requested for the GTS.

The GTS direction subfield is 1 bit in length and shall be set to 1 if the GTS is to be a receive only GTS. Conversely, this subfield shall be set to 0 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.

1.3.3.2 GTS allocation command

The GTS allocation command is used by the PAN coordinator either in response to a GTS request command from an associated device or to inform an associated device of the deallocation or reallocation of one of its GTSs.

The GTS allocation command shall be formatted as illustrated in Figure 33.

Figure 33—GTS allocation command format

octets: 6 or 7	1	2	1
MAC header fields	Command frame identifier (Table 38)	GTS specification	GTS status

1.3.3.2.1 MAC header fields

The fields of the MAC header portion of the general MAC frame format (See Table 12) shall be specified as follows.

The destination addressing mode subfield of the frame control field shall be set to the short addressing mode supported on the PAN, and source addressing mode subfield shall be set to 0 (source addressing information not present).

If security is used for the GTS allocation command, the security enabled subfield shall be set to 1 and the frame shall be protected by the method defined by the security suite corresponding to the destination address. Otherwise, the security enabled subfield shall be set to 0.

The frame pending subfield of the frame control field shall be set to 0 and ignored upon reception, and the acknowledgment request subfield of the frame control field shall be set to 1.

The destination PAN identifier field shall contain the value of *macPANId* and the destination address field shall contain the short address of the device that requested or owns the GTS.

1.3.3.2.2 GTS specification field

The GTS specification field is 16-bits in length and shall be formatted as illustrated in Figure 34.

Figure 34—The GTS specification format

Bits: 0-2	**3**	**4-7**	**8-11**	**12-15**
GTS identifier	GTS direction	GTS start slot	GTS length	Reserved

The GTS identifier subfield is 3 bits in length and specifies the identifier of the GTS to which the GTS allocation command refers. In the case where a new GTS has been allocated following a request, the GTS identifier field shall contain the unique identifier with which the GTS shall be referenced in all future communications. In the case where a new GTS has not been allocated following a request, the GTS identifier field shall contain the value 0.

The GTS direction subfield shall be set to 1 if the GTS is a receive only GTS. Conversely, this subfield shall be set to 0 if the GTS is a transmit only GTS. GTS direction is defined relative to the direction of data frame transmissions by the device.

The GTS start slot subfield is 4 bits in length and specifies the starting slot for the GTS, relative to the start of the superframe and can be a number between 0 and (*aNumSuperframeSlots*-1). In the case where a new GTS has been allocated following a request, the GTS start slot field shall contain the first superframe slot at which the new GTS can be used. In the case where an existing GTS has been reallocated, the GTS start slot field shall contain the first superframe slot at which the reallocated GTS can be used. In the case where an existing GTS is to be deallocated or an allocation request has failed, the GTS start slot field shall contain the value 0 and ignored on reception.

The GTS length field is 4-bits in length and specifies the length of the GTS in superframe slots. In the case where a new GTS has been allocated following a request, the GTS length field shall contain the requested length of the GTS. In the case where a new GTS has not been allocated following a request, the GTS length field shall contain the current number of slots remaining in the CAP that can be allocated to a GTS. In the case where an existing GTS has been reallocated, the GTS length field shall contain the same GTS length used before reallocation. In the case where a GTS is being deallocated, the GTS length field shall contain the value 0.

1.3.3.2.3 GTS status field

The GTS status field is 8-bits in length and shall contain one of the non-reserved values listed in Table 42.

Table 42—Valid values of the GTS status field

Value	GTS status description
0x00	Reserved for MAC primitive enumeration value.
0x01	The requested GTS has been successfully allocated.
0x02	The indicated GTS has been successfully deallocated.
0x03	The indicated GTS has been successfully re-allocated.
0x04	The characteristics of the indicated GTS have been successfully confirmed to the originator of the GTS request command.
0x05	The GTS allocation attempt has failed due to insufficient space in the CAP.
0x06	The GTS allocation attempt has failed because the PAN coordinator has already allocated its maximum number of GTSs.
0x07	The GTS allocation attempt has failed because the PAN coordinator is currently processing a previous GTS request.
0x08	The requested GTS operation has failed due to the specification of an invalid GTS identifier in the GTS request command.
0x09-0x7f	Reserved.
0x80-0xff	Reserved for MAC primitive enumeration values.

1.4 MAC constants and PIB attributes

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

1.4.1 MAC constants

The constants that define the characteristics of the MAC sublayer are presented in Table 43.

1.4.2 MAC PAN information base

The MAC PIB comprises the attributes required to manage the MAC sublayer of a device. Each of these attributes can be read or written using the MLME-GET.request and MLME-SET.request primitives, respectively. The attributes contained in the MAC PIB are presented in Table 44; those attributes marked with an asterisk (*) are optional for an RFD. The MAC PIB security related attributes are presented in Table 45 and Table 46.

Table 43—MAC sublayer constants

Constant	Description	Value
aAckWaitDuration	The maximum number of symbols to wait for an acknowledgment frame to arrive following a transmitted data frame.	32 + 11 * *physSymbolsPerOctet*
aBaseSlotDuration	The number of symbols comprising a superframe slot when the superframe order is equal to 0.	60
aBaseSuperframeDuration	The number of symbols comprising a superframe when the superframe order is equal to 0.	*aBaseSlotDuration* * *aNumSuperframeSlots*
aExtendedAddress	The 64-bit (IEEE) address assigned to the device.	Device specific
aLIFSPeriod	The number of symbols comprising a LIFS period.	40
aMaxBE	The maximum value of the backoff exponent in the CSMA-CA algorithm.	5
aMaxBeaconOverhead	The maximum number of octets added by the MAC sublayer to the payload of its beacon frame.	74
aMaxBeaconPayloadSize	The maximum size, in octets, of a beacon payload.	*aMaxPHYPacketSize* - *aMaxBeaconOverhead*
aMaxCommunicationsFailures	The maximum number of successive communication failures before a device declares itself orphaned. A single communications failure occurs when a transaction fails to reach its intended destination, i.e. an acknowledgement is not received after *aMaxFrameRetries* attempts.	3
aMaxFrameOverhead	The maximum number of octets added by the MAC sublayer to its payload without security. If security is required on a frame, its secure processing may inflate the frame length so that it is greater than this value. In this case, an error will be generated through the appropriate .confirm or MLME-SECURITY-ERROR.indication primitives.	25
aMaxFrameResponseTime	The maximum number of symbols to wait for a frame intended as a response to a data request frame.	1180

Table 43—MAC sublayer constants

Constant	Description	Value
aMaxFrameRetries	The maximum number of retries allowed after a transmission failure.	3
aMaxLostBeacons	The maximum number of consecutive beacons the MAC sublayer can miss without declaring a loss of synchronization.	4
aMaxMACFrameSize	The maximum number of octets that can be transmitted in the MAC frame payload field.	*aMaxPHYPacketSize* - *aMaxFrameOverhead*
aMaxSIFSFrameSize	The maximum size of a frame, in octets, that can be followed by a SIFS period.	18
aMinCAPLength	The minimum number of symbols comprising the CAP. This ensures that MAC commands can still be transferred to devices when GTSs are being used.	420
aNumSuperframeSlots	The number of slots contained in any superframe.	16
aResponseWaitTime	The maximum number of symbols a device shall wait for a response command to be available following a request command.	32 * *aBaseSuperframeDuration*
aSIFSPeriod	The number of symbols comprising a SIFS period.	12
aSuperframesBeforeSnooze	The number of superframes of no activity before the coordinator enters snooze mode.	5
aUnitBackoffPeriod	The number of symbols comprising the basic time period used by the CSMA-CA algorithm.	20

1.5 MAC functional description

This subclause provides a detailed description of the MAC functionality. Subclause 1.5.1 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 Carrier Sense Multiple Access-Collision Avoidance (CSMA-CA) backoff algorithm. Contention free access is controlled entirely by the PAN coordinator through the use of Guaranteed Time Slots (GTSs).

The mechanisms used for starting and maintaining a PAN are described in subclause 1.5.2. Channel scanning is used by a device to either assess the current state of a channel (or channels), locate all beacons within its radio sphere of influence 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 a PAN ID that is not being used by any other PAN in the area. Because it is still possible for the radio sphere of influence 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 subclause 1.5.3. 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.

Table 44—MAC PIB attributes

Attribute	Identifier	Type	Range	Description	Default
*macAssociationPermit**	0x40	Boolean	TRUE or FALSE	This indicates whether a coordinator is currently allowing association. A value of TRUE indicates that association is permitted.	FALSE
macAutoRequest	0x41	Boolean	TRUE or FALSE	This indicates 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 will be automatically sent.	TRUE
*macBeaconMSDU**	0x42	Set of octets	-	The contents of the beacon payload.	NULL
*macBeaconMSDU-Length**	0x43	Integer	0-*aMax-Beacon-PayloadSize*	The length, in octets, of the beacon payload.	0
*macBeaconOrder**	0x44	Integer	0-15	This specifies how often the coordinator will transmit a beacon. The *macBeaconOrder*, *BO*, and the beacon interval, *BI*, are related as follows: for $0 \le BO \le 14$, $BI = aBaseSuperframeDuration * 2^{BO}$ symbols. If $BO= 15$, the coordinator will not transmit a beacon.	0
*macBeaconTxTime**	0x45	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. The precision of this value shall be a minimum of 20-bits, with the lowest four bits being the least significant.	0x000000
*macBSN**	0x46	Integer	0x00-0xff	The sequence number for the current beacon frame.	Random value from within the range.
macCoordExtendedAddress	0x47	IEEE address	An extended 64-bit, IEEE address	The 64-bit address of the coordinator with which the device is associated.	-

Table 44—MAC PIB attributes

Attribute	Identifier	Type	Range	Description	Default
macCoordShortAddress	0x48	Integer	0x0000-0xffff	The 16-bit short address assigned to the coordinator with which the device is associated. If the coordinator supports the 8-bit short addressing mode, this value shall be less than 0x00fe. If the coordinator supports the 16-bit short addressing mode, this value shall be greater than 0x00ff. A value of 0x00fe indicates that the coordinator is only using its 64-bit extended address. A value of 0x00ff indicates that this value is unknown.	0x00ff
macDSN	0x49	Integer	0x00-0xff	The sequence number for the current data or MAC command frame.	Random value from within the range.

Table 44—MAC PIB attributes

Attribute	Identifier	Type	Range	Description	Default
macMaxCSMABack-offs	0x4a	Integer	0-5	The maximum number of backoffs the CSMA-CA algorithm will attempt before declaring a channel access failure.	4
macMinBE	0x4b	Integer	0-3	The minimum value of the backoff exponent in the CSMA-CA algorithm. Note that if this value is set to 0, collision avoidance will be disabled during the first iteration of the algorithm.	3
macPANId	0x4c	Integer	0x0000-0xffff	The 16-bit identifier of the PAN on which the device is operating.	0x0000
*macPromiscuous-Mode**	0x4d	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 sub-layer accepts all frames received from the PHY.	FALSE
*macRxGTSId**	0x4e	Integer	0-7	The identifier of the current receive GTS if allocated or 0 if a receive GTS is not allocated. For the PAN coordinator this value identifies the currently active receive GTS corresponding to a transmit GTS on a device.	0
macShortAddress	0x4f	Integer	0x0000-0xffff	The 16-bit address allocated to the device during association. If the coordinator supports the 8-bit short addressing mode, this value shall be less than 0x00fe. If the coordinator supports the 16-bit short addressing mode, this value shall be greater than 0x00ff. A value of 0x00fe indicates that the device has associated but has not been allocated an address. A value of 0x00ff indicates that the device has not associated.	0x00ff
*macSnoozePermit**	0x50	Boolean	TRUE or FALSE	This indicates whether the coordinator is permitted to enter snooze mode. A value of TRUE indicates that snoozing is permitted.	TRUE
*macSuperframeOrder**	0x51	Integer	0-15	This specifies the length of the active portion of the superframe, including the beacon frame. The *macSuperframeOrder*, *SO*, and the superframe duration, *SD*, are related as follows: for $0 \leq SO \leq$ BO ≤ 14, *SD* = *aBaseSuperframeDuration* * 2^{SO} symbols. If SO = 15, the superframe will not be active following the beacon.	0
*macTransactionPersistenceTime**	0x52	Integer	0x0000-0xffff	The maximum time (in superframe periods) that a transaction is stored by a coordinator and indicated in its beacon.	0x01f4

Table 44—MAC PIB attributes

Attribute	Identifier	Type	Range	Description	Default
*macTxGTSId**	0x53	Integer	0-7	The identifier of the current transmit GTS if allocated or 0 if a transmit GTS is not allocated. For the PAN coordinator this value identifies the currently active transmit GTS corresponding to a receive GTS on a device.	0

Table 45: MAC PIB security attributes

Attribute	Identifier	Type	Range	Description	Default
macACLEntry-DescriptorSet	0x70	Set of ACLDescriptor values (See Table 46)	Variable	A set of ACL entries, each containing address information, security suite information and security material to be used to protect frames between the MAC sublayer and the specified device.	Null set
macDefaultSecurity	0x71	Boolean	TRUE or FALSE	This indicates whether the device is able to transmit secure frames to or accept secure frames from devices that are not explicitly listed in the ACL. It is also used to communicate with multiple devices at once. A value of TRUE indicates that such transmissions are permitted.	FALSE
macDefaultSecurityMaterial	0x72	octet string	Variable	The specific security material to be used to protect frames between the MAC sublayer and devices not in the ACL. (See subclause 1.6.1.8)	Empty string
macDefaultSecuritySuite	0x73	Integer	0x00 - 0xff	The unique identifier of the security suite to be used to protect communications between the MAC and devices not in the ACL as specified in Table 50.	0
macSecurity-Mode	0x74	Integer	0-2	The identifier of the security mode in use as specified in 1.5.9. 0 = Unsecured mode 1 = ACL mode 2 = Secured mode	0

The mechanisms to allow devices to acquire and maintain synchronization with a coordinator are described in subclause 1.5.4. Synchronization on a beacon enabled PAN is described after first explaining how a coordinator generates beacon frames. Following this, synchronization on a non-beacon enabled PAN is described. A procedure to re-establish communication between a device and its coordinator has been created, as it is possible that a device may lose synchronization in either the case of a beacon enabled or non-beacon enabled PAN.

The IEEE 802.15.4 protocol has been designed so that application data transfers can be controlled by the devices on a PAN rather than by the coordinator. Subclause 1.5.5 describes the procedures the coordinator uses to handle multiple transactions while preserving this requirement.

The mechanisms for transmitting, receiving and acknowledging frames, including frames sent using indirect transmission, are described in subclause 1.5.6. In addition, methods for retransmitting frames and resolving duplicate received frames is also described.

Table 46—Elements of ACL entry descriptor

Attribute	Type	Range	Description	Default
ACLExtendedAddress	IEEE Address	Any valid 64-bit device address	The 64-bit extended IEEE address of the device in this ACL entry.	Device specific
ACLShortAddress	Integer	0x0000-0xffff	The 16-bit short address of the device in this ACL entry. If the device uses 8-bit short addressing mode, this value shall be less than 0x00fe. If the device uses 16-bit short addressing mode, this value shall be greater than 0x00ff. A value of 0x00fe indicates that the device is only using its 64-bit extended address. A value of 0x00ff indicates that this value is unknown.	0x00ff
ACLPANId	Integer	0x0000-0xffff	The 16-bit PAN identifier of the device in this ACL entry.	Device specific
ACLSecurityMaterial-Length	Integer	0-26	The number of octets contained in *ACLSecurityMaterial.*	21
ACLSecurityMaterial	Octet string	Variable	The specific keying material to be used to protect frames between the MAC sublayer and the device indicated by the associated ACLExtendedAddress (See subclause 1.6.1.8).	Empty string
ACLSecuritySuite	Integer	0x00-0x07	The unique identifier of the security suite to be used to protect communications between the MAC sublayer and the device indicated by the associated ACLExtendedAddress as specified in Table 50.	0x00

In some instances, the quality of communication on the current channel may degrade. Subclause 1.5.7 describes a procedure that attempts to re-establish clear communication by dynamically switching to an alternative channel.

The mechanisms for allocating and deallocating a GTS are described in subclause 1.5.8. 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.

When the next higher layer or DME requests that security be implemented prior to transmission of a frame or upon receipt of a frame, the MAC sublayer uses the mechanisms defined in subclause 1.5.9.

Throughout this subclause, the receipt of a frame is defined as the successful receipt of the frame by the PHY layer and the successful verification of the frame check sequence by the MAC sublayer, as described in subclause 1.2.1.8.

1.5.1 Channel access

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

1.5.1.1 Superframe structure

The coordinator on an IEEE 802.15.4 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 shall only interact with its PAN during the active portion of the superframe and so 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. 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).

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 contention access period (CAP) and a contention free period (CFP). The beacon shall be transmitted, without the use of CSMA, at the start of slot 0, and the CAP shall commence immediately after the beacon. The CFP, if present, follows immediately after the CAP and extends to the end of the active portion of the superframe. Any allocated guaranteed time slots (GTSs) shall be located within the CFP.

.

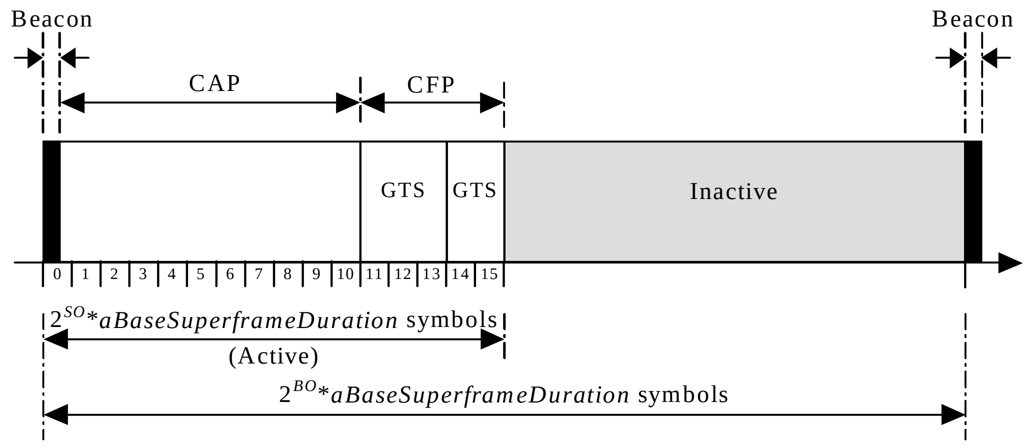


Figure 35—An example of the superframe structure

PANs that wish to use the superframe structure shall set *macBeaconOrder* to a value between 0 and 14 and *macSuperframeOrder* to a value between 0 and the value of *macBeaconOrder.*

PANs that do not wish to use the superframe structure (referred to as a non-beacon enabled PAN) shall set both *macBeaconOrder* and *macSuperframeOrder* to 15. In this case, a coordinator shall not transmit beacons and all transmissions, with the exception of acknowledgment frames and any data frame that quickly follows the acknowledgment of a data request command (See subclause 1.5.6.3), shall use an unslotted CSMA-CA mechanism to access the channel. In addition, GTSs shall not be permitted.

An example of the superframe structure is shown in Figure 35. In this case, the beacon interval, *BI*, is twice as long as the active superframe duration, *SD*, and the CFP contains two GTSs.

1.5.1.1.1 The contention access period (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 superframe. The CAP shall be at least *aMinCAPLength* symbols 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 subclause 1.5.6.3), 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 inter-frame spacing period (See subclause 1.5.1.2) 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.

1.5.1.1.2 The contention free period (CFP)

The CFP shall start on a slot boundary immediately following the CAP and it shall complete before the start of the next beacon. 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 inter-frame spacing period (See subclause 1.5.1.2) before the end of its GTS.

1.5.1.2 Inter-frame spacing

The MAC sublayer needs a finite amount of time to process data received by the PHY layer. To allow for this, all transmissions shall be followed by an inter-frame spacing period, the length of which is dependent on the size of the frame that has just been transmitted. Frames of up to *aMaxSIFSFrameSize* in length shall be followed by a short inter-frame spacing (SIFS) period of duration *aSIFSPeriod* symbols. Frames with lengths greater than *aMaxSIFSFrameSize* shall be followed by a long inter-frame spacing (LIFS) of duration *aLIFSPeriod* symbols.

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

1.5.1.3 The CSMA-CA algorithm

The CSMA-CA algorithm shall be used before the transmission of data or MAC command frames not transmitted within the CFP. The CSMA-CA algorithm shall not be used for the transmission of beacon frames, acknowledgement frames, data frames transmitted in the CFP or data or MAC command frames that can be quickly transmitted in response to a data request command, immediately following the acknowledgement (See subclause 1.5.6.3).

If 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 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. Transmissions in slotted CSMA-CA shall only commence 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 0 before each 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 2 before each transmission attempt and reset to 2 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; this value shall be initialized to the value of *macMinBE*. Note that if *macMinBE* is set to 0, collision avoidance will be disabled during the first iteration of this algorithm.

Although the receiver of the device is enabled during the channel assessment portion of this algorithm, the device shall discard any frames received during this time.

Figure 36 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 (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 (2) and then request that the PHY perform a clear channel assessment (CCA) (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, 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. 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 (two CCA analyses), the frame transmission and any acknowledgement 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 repeat the evaluation.

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

If the channel is assessed to be idle (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 if it is equal to 0. If it is not equal to 0, the CSMA-CA algorithm shall return to step (3). If it is equal to 0, 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.

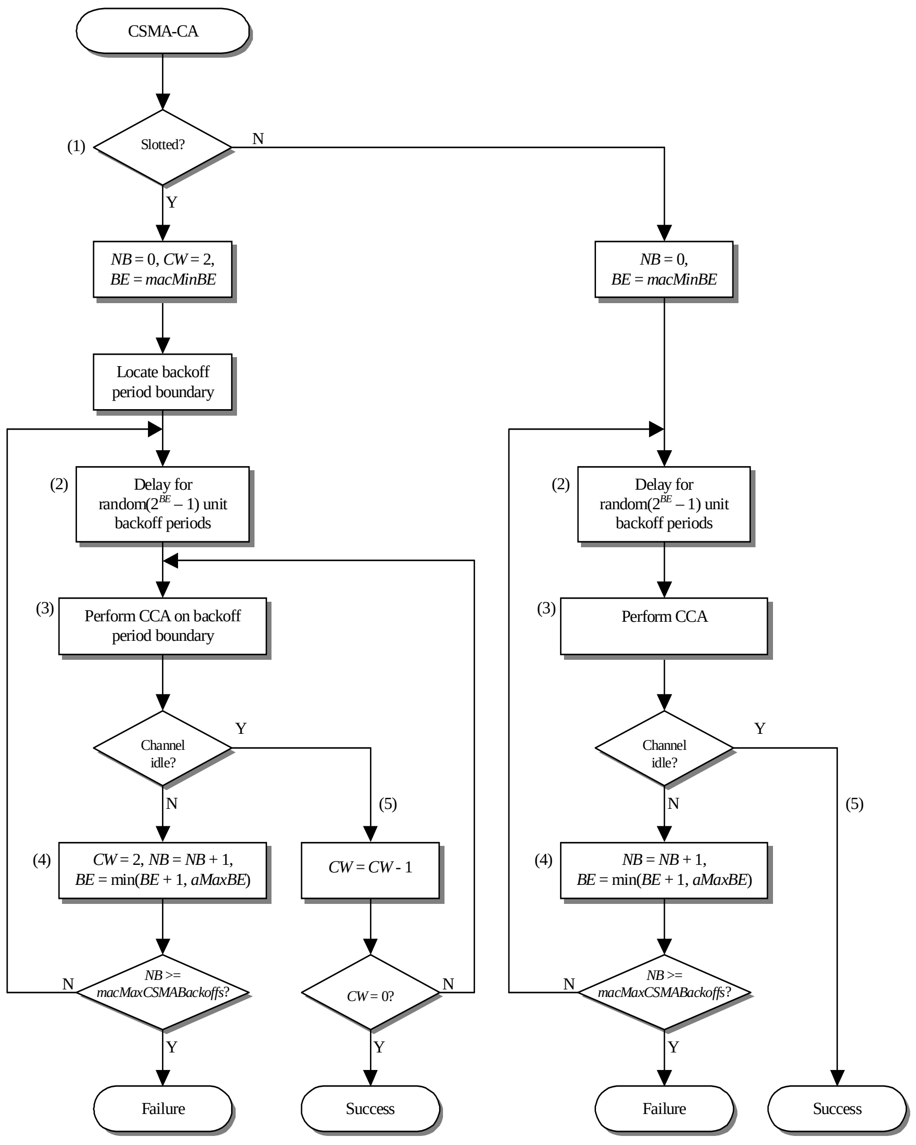

CSMA-CA
(1)
Slotted?
N
Y
NB = 0, CW = 2, BE = macMinBE
NB = 0, BE = macMinBE
Locate backoff period boundary
(2)
Delay for random(2^BE – 1) unit backoff periods
(3)
Perform CCA on backoff period boundary
Perform CCA
Channel idle?
(4)
CW = 2, NB = NB + 1, BE = min(BE + 1, aMaxBE)
NB = NB + 1, BE = min(BE + 1, aMaxBE)
(5)
CW = CW - 1
NB >= macMaxCSMABackoffs?
CW = 0?
Failure
Success

Figure 36—The CSMA-CA algorithm

1.5.2 Starting and maintaining PANs

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

1.5.2.1 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 energy detection and active scans.

A device is instructed to begin a channel scan through the MLME-SCAN.request primitive. For the duration of the scan, the device shall suspend beacon transmissions, if applicable, and 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.

1.5.2.1.1 Energy detection channel scan

An energy detection scan allows an FFD 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 in which to operate prior to starting a new PAN or to switch to during dynamic channel selection.

An energy detection scan over a specified set of logical channels is requested using the MLME-SCAN.request primitive with the ScanType parameter set to indicate an energy detection scan. For each logical channel, the MLME shall first switch to the channel, by setting *phyCurrentChannel* accordingly, and then repeatedly perform an energy detection measurement for (*aBaseSuperframeDuration* * $(2^n + 1)$) symbols, where n is the value of the ScanDuration parameter in the MLME-SCAN.request primitive. An energy detection measurement is performed by the DME issuing the PLME-ED.request (See subclause 6.2.2.3), which is guaranteed to return a value. The maximum energy detection measurement obtained during this period shall be noted.

The energy detection scan shall terminate when energy has been measured on each of the specified logical channels.

1.5.2.1.2 Active channel scan

An active scan allows an FFD to locate any coordinator transmitting beacon frames within its radio sphere of influence. This could be used by a prospective PAN coordinator to select a PAN identifier prior to starting a new PAN.

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* accordingly, and send a beacon request command (See subclause 1.3.2.4). The device shall then enable its receiver for at most (*aBaseSuperframeDuration* * $(2^n + 1)$) symbols, where n is the value of the ScanDuration parameter in the MLME-SCAN.request primitive. During this time, the device shall reject all non-beacon frames and record the information contained in all unique beacons in a PAN descriptor structure (See Table 29). If a beacon frame is received with the security enabled subfield set to 1, the device shall attempt to process the beacon frame for security, as described in subclause 1.5.9. Any errors encountered during the secure processing of the beacon frame shall be ignored and the beacon information shall be recorded in a PAN descriptor with the SecurityUse, ACLEntry and SecurityFailure fields set accordingly (See Table 14).

If a coordinator that is currently in its snooze mode receives the beacon request command, it shall exit snooze mode and begin transmitting beacons at the next superframe boundary according to the value of *macBeaconOrder*. If a coordinator of a non-beacon enabled PAN receives this command, it shall transmit a single beacon frame using unslotted CSMA-CA.

The active scan shall terminate when the number of beacons found equals an implementation specified limit or the set of available channels have each been scanned.

1.5.2.1.3 Passive channel scan

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

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* accordingly, and then enable its receiver for at most (*aBaseSuperframeDuration* * (2^n + 1)) symbols, where *n* is the value of the ScanDuration parameter in the MLME-SCAN.request primitive. During this time, the device shall reject all non-beacon frames and record the information contained in all unique beacons in a PAN descriptor structure (See Table 29). If a beacon frame is received with the security enabled subfield set to 1, the device shall attempt to process the beacon frame for security, as described in subclause 1.5.9. Any errors encountered during the secure processing of the beacon frame shall be ignored and the beacon information shall be recorded in a PAN descriptor with the SecurityUse, ACLEntry and SecurityFailure fields set accordingly (See Table 14).

The passive scan shall terminate when the number of beacons found equals an implementation specified limit or the set of available channels have each been scanned.

1.5.2.1.4 Orphan channel scan

An orphan scan allows a device to attempt to relocate its coordinator following a loss of synchronization.

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* accordingly, and then send an orphan notification command (See subclause 1.3.2.3). The device shall then enable its receiver for at most (*aBaseSuperframeDuration* * (2^n + 1)) symbols, where *n* is the value of the ScanDuration parameter in the MLME-SCAN.request primitive. If the device successfully receives a coordinator realignment command (See subclause 1.3.2.5) within this time, the device shall disable its receiver.

If a coordinator receives the orphan notification command, it shall search its device list for the device sending the command. If the coordinator finds a record of the device, it shall send a coordinator realignment command to the orphaned device. The process of searching for the device and sending the coordinator realignment command shall occur within *aResponseWaitTime* symbols. The coordinator realignment command shall contain its current PAN identifier, *macPANId*, its current logical channel and the short address of the orphaned device. If a coordinator finds no record of the device, it shall ignore the command and 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.

1.5.2.2 PAN identifier conflict resolution

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

This procedure is optional for an RFD.

1.5.2.2.1 Detection

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

- A beacon frame is received by the PAN coordinator with the PAN coordinator subfield (See subclause 1.2.2.1.2) set to 1 and the PAN identifier equal to *macPANId*.

- A PAN ID conflict notification command (See subclause 1.3.2.2) is received by the PAN coordinator from a device on its PAN.

A device 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 1, the PAN identifier equal to *macPANId* and an address which is neither *macCoordShortAddress* nor *macCoordExtendedAddress*.

1.5.2.2.2 Resolution

On the detection of a PAN identifier conflict, the coordinator shall first perform an active scan and then, 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. The coordinator shall then broadcast the coordinator realignment command containing the new PAN identifier with the source PAN identifier field equal to the value in *macPANId*. Once the coordinator realignment command has been sent, the coordinator shall set *macPANId* to the new PAN identifier.

On the detection of a PAN identifier conflict by a device, it shall generate the PAN ID conflict notification command (See subclause 1.3.2.2) and send it to the PAN coordinator. If the PAN ID conflict notification command is received correctly, the PAN coordinator shall send an acknowledgment frame, thus confirming receipt. The PAN coordinator shall then resolve the conflict as described above.

1.5.2.3 Starting a PAN

A PAN shall be started by an FFD only after an active channel scan has been performed and a suitable PAN identifier selection has been made. 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 shall select a short address that is not equal to 0x00ff and set *macShortAddress* equal to this value. The short addressing mode of all addresses allocated on the PAN shall match the addressing mode selected by the PAN coordinator when the PAN is started, as described in Table 47.

Table 47—PAN coordinator addressing modes

Value of *macShortAddress*	Description
0x0000-0x00fd	The PAN shall use 8-bit short addressing mode and addresses shall be allocated in the range 0x0000-0x00fd.
0x00fe	The PAN shall use 64-bit extended addressing mode and short addresses shall not be allocated.
0x00ff	The PAN cannot be started.
0x0100-0xffff	The PAN shall use 16-bit short addressing mode and addresses shall be allocated in the range 0x0100-0xffff.

An FFD is instructed to begin operating a PAN through the use of the MLME-START.request primitive with the PANCoordinator parameter set to TRUE. On receipt of this primitive, the MAC sublayer shall set the logical channel in *phyCurrentChannel* and the PAN identifier in *macPANId*. After completing this, the MAC sublayer shall respond with the MLME-START.confirm primitive and begin operating as a PAN coordinator.

1.5.2.4 Beacon generation

An FFD shall use the MLME-START.request primitive to begin transmitting beacons. The FFD may either begin beacon transmission 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 subclause 1.1.14.1). On receipt of this 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 beacon shall depend on the value of *macShortAddress* as shown in Table 48.

Table 48—Addresses used in the beacon frame

Value of *macShortAddress*	Description
0x0000-0x00fd	The beacon shall contain an 8-bit short address.
0x00fe	The beacon shall contain a 64-bit extended address.
0x00ff	Invalid address - do not begin beacon transmission.
0x0100-0xffff	The beacon shall contain a 16-bit short address.

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 shall be chosen to be the same as that used in the timestamp of the incoming beacon frame, as described in subclause 1.5.4.1.

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 beacon frame shall be constructed as specified in subclause 1.2.2.1.

In order to limit the traffic on the radio channel and hence the used bandwidth, a coordinator device may move into a quiescent state called snooze mode. The coordinator shall enter snooze mode only if all of the following conditions are satisfied:

- *macSnoozePermit* is TRUE.
- There are no allocated GTS.
- *aSuperframesBeforeSnooze* superframes have elapsed, during which no messages were pending and no device activity was detected on the channel.

While in snooze mode, the coordinator shall cease the transmission of beacons, while still listening for messages from devices during the active portion of the superframe and keeping track of the superframe boundaries. The coordinator shall preserve the values of *macBeaconOrder* and *macSuperframeOrder* while in snooze mode.

The coordinator shall be able to move out of snooze mode in less than 2^n**aBaseSuperframeDuration* symbols, where *n* is the value of *macBeaconOrder*. This means that if the stimulus to move out of snooze mode occurs close to the end of a superframe, the coordinator will be ready to transmit its beacon at the start of the superframe after next.

A transition from snooze mode to normal mode shall occur when any of the above conditions for snooze become unsatisfied.

1.5.2.5 Device discovery

An FFD may indicate its presence on a PAN to other devices by transmitting beacon frames. This allows the other devices to perform device discovery.

An FFD, that is not the PAN coordinator, shall only begin transmitting beacon frames 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 using the identifier of the PAN with which the device has associated, *macPANId*, and its short address, *macShortAddress*. 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*. In the case where *macShortAddress* is equal to 0x00fe, the device shall transmit beacons only using its 64-bit extended address, *aExtendedAddress*.

1.5.3 Association and disassociation

This subclause specifies the procedures for association and disassociation.

1.5.3.1 Association

A device shall only attempt to associate after having first performed a MAC sublayer reset, by issuing the MLME-RESET.request primitive, and secondly completed a passive channel scan (See subclause 1.5.2.1.3) after which a suitable PAN selection has been made. 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.

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

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

An unassociated device shall initiate the association procedure by sending an associate request command (See subclause 1.3.1.1) to the coordinator of an existing PAN. If the association request command is received correctly, the coordinator shall send an acknowledgment frame, thus confirming receipt.

The acknowledgment to an association request command does not mean that the device has associated. The coordinator needs time to determine whether the current resources available on the PAN are sufficient to allow another device to associate. The coordinator shall make this decision within *aResponseWaitTime* symbols. If the coordinator finds that the device was previously associated on its PAN, all previously obtained device specific information shall be removed. If sufficient resources are available, the coordinator shall allocate a short address to the device and generate an association response command (See subclause 1.3.1.2) containing the new address and a status indicating a successful association. If sufficient resources are not available, the coordinator shall generate an association response command containing a status indicating a failure (See Table 39). The association response command shall be sent to the device requesting association using indirect transmission, i.e. the address of the device requesting association shall be added to the address list field of the beacon frame, indicating a pending message, and extracted at the discretion of the device. In this case, the association response command shall be added to the transaction list of the coordinator.

If the allocate address subfield of the association request command is set to 1, the coordinator shall allocate a 16-bit address with a range depending on the addressing mode supported by the coordinator, as described in Table 47. If the allocate address subfield of the association request command is set to 0, the address allocated by the coordinator shall be equal to 0x00fe. A short address of 0x00fe is a special case that indicates that the device has associated but has not been allocated a short address. In this case, the device shall only use its 64-bit extended address to operate on the network.

On receipt of the acknowledgement to the association request command, the device shall wait for *aResponseWaitTime* symbols for the coordinator to make its association decision. After this time, the device shall attempt to extract the association response command from the coordinator according to the procedure described in subclause 1.5.6.3. If the device subsequently does not extract a frame from the coordinator, it shall issue the MLME-ASSOCIATE.confirm primitive with a status of NO_DATA, and the association attempt shall be deemed a failure.

On receiving the association response command, the device requesting association shall send an acknowledgment frame, thus confirming receipt. If the association status field of the command indicates that the association was successful, the device shall store the addresses of the coordinator with which it has associated. The short address of the coordinator, contained in the association response command, shall be stored in *macCoordShortAddress* and the extended address of the coordinator, contained in the MAC header of the association response command frame, shall be stored in *macCoordExtendedAddress.* The device shall also store the address contained in the short address field in *macShortAddress* and the identifier of the new PAN in *macPANId*. Communication on the PAN using this short address shall depend on its range, as described in Table 49.

Table 49—Usage of the short address

Value of *macShortAddress*	Description
0x0000-0x00fd	The device shall use 8-bit short addressing mode with an address consisting of the least significant octet of *macShortAddress*.
0x00fe	The device shall use 64-bit extended addressing mode with an address consisting of *aExtendedAddress.*
0x00ff	The device is not associated and shall not communicate on the PAN.
0x0100-0xffff	The device shall use 16-bit short addressing mode with an address consisting of *macShortAddress.*

1.5.3.2 Disassociation

The disassociation procedure is initiated by the DME by issuing the MLME-DISASSOCIATE.request primitive to the MLME.

When a coordinator wants a device to leave the PAN, it shall send the disassociation notification command to the device using indirect transmission, i.e. the address of the device shall be added to the address list field of the beacon frame, indicating a pending message, and extracted at the discretion of the device. In this case, the disassociation notification command shall be added to the transaction list of the coordinator. If the device requests and correctly receives the disassociation notification command, it shall confirm its receipt by sending an acknowledgment frame.

If a device wants to leave the PAN, it shall send a disassociation notification command to the coordinator. If the disassociation notification command is received correctly by the coordinator, it shall confirm its receipt by sending an acknowledgment frame.

The reason for leaving the PAN shall be given in the disassociate notification command (See subclause 1.3.1.3).

At this point, the device shall be considered disassociated from the PAN, even if an acknowledgement was not received. The MLME of the device shall then set *macShortAddress* to its default value of 0x00ff.

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

1.5.4 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.

1.5.4.1 Synchronization with Beacons

All devices operating on a beacon enabled PAN (*macBeaconOrder* < 15) shall be able to acquire beacon sychronization in order to detect any pending messages or to track the beacon. Devices shall only be permitted to acquire beacon synchronization with beacons containing the PAN identifier specified in *macPANId*. If *macPANId* specifies the broadcast PAN identifier (0xffff), a device shall be able to acquire beacon synchronization with any beacon.

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 attempt to acquire the beacon only once.

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 DME 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 is implementation specific. The symbol boundary shall be chosen to be the same as that used in the timestamp of the outgoing beacon frame, stored in *macBeaconTxTime*.

If the security enabled subfield is set to 1, the MLME shall process the received beacon frame for security, as described in subclause 1.5.9. If the secure processing fails, the frame shall be discarded and the MLME shall issue the MLME-SECURITY-ERROR.indication primitive, indicating the error.

If a beacon frame is received and *macAutoRequest* is set to FALSE, the MLME shall indicate the beacon parameters to the DME 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 in the address list field of the beacon frame. If the address list field contains the short or 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 subclause 1.5.6.3). If the short or extended address of the device is contained in the address list field and the source PAN identifier does not match *macPANId*, the MLME shall discard the beacon frame.

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 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.

1.5.4.2 Synchronization without beacons

All devices operating on a non-beacon enabled PAN (*macBeaconOrder* = 15) shall be able to poll the coordinator for data at the discretion of the DME.

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 subclause 1.5.6.3).

1.5.4.3 Orphaned device realignment

After *aMaxCommunicationsFailures* successive unsuccessful attempts at communicating with the coordinator with which it has associated, a device shall conclude that it has been orphaned. A single communications failure occurs when a device transaction fails to reach the coordinator, i.e. an acknowledgement is not received after *aMaxFrameRetries* attempts at sending the data. When the MLME concludes that it is orphaned, it shall notify the DME by issuing the MLME-SYNC-LOSS.indication primitive with a loss reason of COORDINATOR_LOST. On receiving this primitive, the DME shall instruct the MLME to either perform the orphaned device realignment procedure or reset the MAC sublayer and then perform the association procedure.

If the decision has been made by the DME to perform the orphaned device realignment procedure, it shall issue an MLME-SCAN.request with the ScanType parameter set to orphan scan and the ChannelList parameter containing the list of channels to be scanned. Upon receiving this primitive, the MAC sublayer shall begin an orphan scan, as described in subclause 1.5.2.1.4.

If the orphan scan is successful (its PAN has been located), the device shall update its MAC PIB with the PAN information contained in the coordinator realignment command (See subclause 1.3.2.5). If the orphan scan was unsuccessful, the DME shall decide what further action need be taken, e.g. to retry the orphan scan or attempt to re-associate.

1.5.5 Transaction handling

Since IEEE 802.15.4 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. This means that 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 if 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 request primitive from the DME, such as the MLME-ASSOCIATE.response primitive (see subclause 1.1.3.3). The information contained in these requests forms a transaction and the coordinator shall be capable of storing at least one transaction. On receipt of this primitive, if there is no capacity to store another transaction, the MAC sublayer shall issue the confirm primitive, corresponding to the original request, with a status value of TRANSACTION_OVERFLOW.

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 1, indicating the additional pending data.

Each transaction shall persist in the coordinator for at most *macTransactionPersistenceTime*. If the transaction is not extracted by the appropriate device within this time, the transaction information shall be discarded and the MAC sublayer shall issue the confirm primitive, corresponding to the original request, with a status of TRANSACTION_EXPIRED.

The coordinator shall not enter snooze mode if any transactions exist. 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 messages, it shall indicate them in its beacon on a first-come-first-served basis, ensuring that every beacon frame contains at most seven addresses. For those 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.

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 non-beacon 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 clause 1.5.6.3.

When the transaction is complete, it shall be discarded and a confirmation of the data shall be sent to the next higher layer. If the transaction required an acknowledgement and an acknowledgement was not received, the MAC sublayer shall issue the confirm primitive, corresponding to the original request, with a status of NO_ACK. If the transaction was successful, the MAC sublayer shall issue the confirm primitive, corresponding to the original request, with a status of SUCCESS.

1.5.6 Transmission, reception and acknowledgment

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

1.5.6.1 Transmission

Prior to the transmission of a data frame or a MAC command frame, the MAC sublayer shall copy the value of *macDSN* into the sequence number field of the MAC header of the outgoing frame. Similarly, prior to the transmission of a beacon frame, the MAC sublayer shall copy the value of *macBSN* into the sequence number field of the MAC header of the outgoing frame.

The source address field, if present, shall contain the address of the device sending the frame. In the case where a device has associated and has been allocated a short address (*macShortAddress* is not equal to 0x00fe or 0x00ff), it shall use that address in preference to its 64-bit extended address (*aExtendedAddress*) wherever possible. In the case where a device has not yet associated to a PAN or *macShortAddress* is equal to 0x00fe, 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 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 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 subclause 1.5.4.1), and hence the device does not know where the beacon will appear, it shall enable its receiver and search for at most (*aBaseSuperframeDuration* * (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 subclause 1.5.1.3). Once the beacon has been found, either after a search or due to it 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 subclause 1.5.1.3), and transmissions in a transmit GTS shall not use CSMA-CA.

If the frame is to be transmitted on a non-beacon enabled PAN, the frame shall be transmitted following the successful application of the unslotted version of the CSMA-CA algorithm (See subclause 1.5.1.3).

1.5.6.2 Reception and rejection

Due to the nature of radio communications, a device will be able to receive and decode transmissions from all IEEE 802.15.4 devices currently operating in its radio sphere of influence, along with interference from other sources. The MAC sublayer shall therefore be able to filter incoming frames and only present the frames which are of interest to the upper layers.

For the first level of filtering, the MAC sublayer shall discard all received frames that are do not contain a correct value in their frame check sequence field, according to the algorithm described in subclause 1.2.1.8. The filtering applied after this 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. The MAC sublayer shall be in promiscuous mode if *macPromiscuousMode* is set to TRUE.

If the MAC sublayer is not in promiscuous mode (*macPromiscuousMode* is set to FALSE), it shall accept only those frames which satisfy all of the following requirements:

— The frame type subfield shall not contain an illegal frame type.
— 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 (0xff). Otherwise if an extended destination address is included in the frame, it shall match *aExtendedAddress*.
— If only source addressing fields are included in the frame, the source PAN identifier shall be the PAN identifier of the coordinator.

If any of the above requirements are not satisfied, the MAC sublayer shall discard the incoming frame. If all of the above requirements are satisfied, the frame shall be considered valid and processed further. For valid frames if the frame type subfield indicates a data or MAC command frame and the acknowledgement requirement subfield is set to 1, the MAC sublayer shall send an acknowledgment frame.

If the security enabled subfield is set to 1, the MAC sublayer shall process the received frame for security, as described in subclause 1.5.9. If the secure processing fails, the frame shall be discarded and the MAC sublayer shall issue the MLME-SECURITY-ERROR.indication primitive, indicating the error. The only exceptions to this are during either an active or passive scan for beacons. In both cases, information contained in the received beacon frames that fail secure processing will still be put into a PAN descriptor, with the SecurityUse, ACLEntry and SecurityFailure fields set accordingly, and passed to the DME (See subclause 1.5.2.1).

If the frame was successfully processed, the MAC sublayer shall issue the MCPS-DATA.indication primitive containing the frame information.

1.5.6.3 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 the contents of the received beacon frame, as described in subclause 1.5.4.1. If the address of the device is contained in the address list field of the beacon frame, the MLME of the device shall send a data request command (See subclause 1.3.2.1) to the coordinator during the CAP with the acknowledgment request subfield set to 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 *aResponseWaitTime* symbols after the acknowledgement to a request command, such as during the association procedure or the GTS allocation/deallocation procedure.

On successfully receiving a data request command, the coordinator shall send an acknowledgement frame, thus confirming its receipt. If the coordinator has enough time to determine whether the device has a frame pending and is still able to send the acknowledgement frame within *aAckWaitDuration* symbols, it shall set the frame pending subfield of the acknowledgement 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 acknowledgement frame to 1.

On receipt of the acknowledgement frame with the frame pending subfield set to 0, the device shall conclude that there is no data pending at the coordinator.

On receipt of the acknowledgement frame with the frame pending subfield set to 1, a device shall enable its receiver for at most *aMaxFrameResponseTime* symbols 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 below. If there is no data frame pending for the requesting device, the coordinator shall send a data frame to the device containing a zero length payload, indicating that no data is present, using one of the mechanisms described below.

The data frame following the acknowledgement 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 inter-frame spacing and acknowledgement. If a requested acknowledgement frame is not received following this data frame, all subsequent retransmissions shall be transmitted using CSMA-CA. The constant *aTurnaroundTime* is defined in the PHY layer constants table (See subclause 6.3.1).

- Using CSMA-CA, otherwise.

If the requesting device does not receive a data frame from the coordinator within *aMaxFrameResponseTime* symbols or receives a data frame from the coordinator with a zero length payload, it shall conclude that there is no data pending at the coordinator. If the requesting device does receive a data frame from the coordinator, it shall send an acknowledgement frame, if requested, thus confirming receipt.

If the frame pending subfield of the data frame, received from the coordinator, is set to 1, the device still has more data pending with the coordinator. In this case it may extract the data by repeating the above procedure.

1.5.6.4 Use of Acknowledgments

A data or MAC command frame shall be sent with its acknowledgment request subfield set appropriately for the frame. A beacon or acknowledgment frame shall always be sent with the acknowledgment request subfield set to 0. Similarly, any frame that is broadcast shall be sent with its acknowledgement request subfield set to 0.

1.5.6.4.1 No acknowledgment

A frame transmitted with its acknowledgment request (AR) subfield set to 0 shall not be acknowledged by its intended recipient. The originating device shall assume that the transmission of the frame was successful.

The MSC in Figure 37 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 *AR* subfield equal to 0.

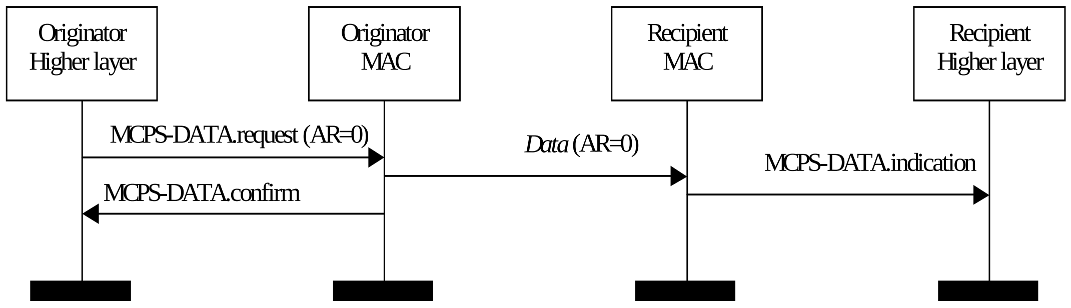


Figure 37—Successful data transmission without an acknowledgment

1.5.6.4.2 Acknowledgment

A frame transmitted with its acknowledgment request subfield set to 1 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 data sequence number from the data or MAC command frame that is being acknowledged.

The transmission of an acknowledgement frame in a non-beacon 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 acknowledgement frame in the CAP shall commence at a backoff slot boundary. In this case, the transmission of an acknowledgement 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 the PHY layer constants table (See subclause 6.4.1).

The MSC in Figure 38 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 *AR* subfield set to 1.

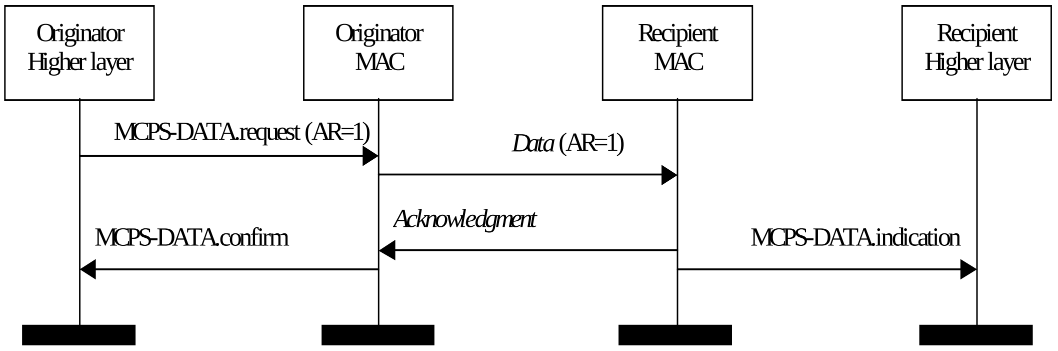


Figure 38—Successful data transmission with an acknowledgment

1.5.6.5 Retransmissions

A device that sends a frame with its acknowledgment request subfield set to 0 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 1 shall wait for at most *aAckWaitDuration* symbols for the corresponding acknowledgment frame to be received. If an acknowledgment frame is received within *aAckWaitDuration* symbols and contains the same data sequence number as the original transmission, the transmission is considered successful and no further action shall be taken by the device. If an acknowledgment is not received within *aAckWaitDuration* symbols or an acknowledgement is received containing a data sequence number that was not the same as the original transmission, the device shall repeat the process of transmitting the data or MAC command frame and waiting for the acknowledgment, up to a maximum of *aMaxFrameRetries* times. If an acknowledgment is still not received after *aMaxFrameRetries* retransmissions, the MAC sublayer shall assume the transmission has failed and notify the next higher layer of the failure. This eventuality is referred to as a communications failure.

1.5.6.6 Duplicate detection

Due to the imperfect nature of the radio medium, the originator of a data or MAC command frame may not receive the acknowledgment from the intended recipient, even though the recipient did in fact transmit the acknowledgment. In this case, the originator will retransmit the frame even though it has already been received and acknowledged by the intended recipient.

All devices shall therefore detect such duplicate receptions so that the same frame is not processed more than once. In the case of a data frame, the MAC sublayer shall ensure that a data indication is issued to the next higher layer only once. Similarly, in the case of a MAC command frame, the MAC sublayer shall ensure that the frame is processed only once.

To detect duplicate frames, the MAC sublayer shall be able to store both the address and sequence number contained in the last frame received from a particular device. A duplicate frame shall then be detected by comparing the originator address and sequence number contained in the received frame with those stored from the last frame received from that device. If these two fields are identical to the last frame received, then the received frame is a duplicate. In this case, the MAC sublayer shall acknowledge the frame but discard the data.

1.5.6.7 Transmission scenarios (informative)

Due to the imperfect nature of the radio medium, a transmitted frame does not always reach its intended destination. Figure 39 illustrates three different data transmission scenarios; these scenarios are:

1) 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 *aAckWaitDuration* symbols. The recipient MAC sublayer receives the data frame, sends an acknowledgment back to the originator and passes the 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.

2) 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 *aAckWaitDuration* 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 was received. The data transfer has failed and the originator retransmits the data. This may be repeated up to a maximum of *aMaxFrameRetries* times. If a data transfer attempt fails a total of (1 + *aMaxFrameRetries*) times, the originator MAC sublayer will issue a failure confirmation to the next higher layer.

3) 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 *aAckWaitDuration* symbols. The recipient MAC sublayer receives the data frame, sends an acknowledgment back to the originator and passes the frame to the next higher layer. The originator MAC sublayer does not receive the acknowledgment frame and its timer expires. The data transfer has failed and the originator will retransmit the data. This may be repeated up to a maximum of *aMaxFrameRetries* times. If a data transfer attempt fails a total of (1 + *aMaxFrameRetries*) times, the MAC sublayer will issue a failure confirmation to the next higher layer.

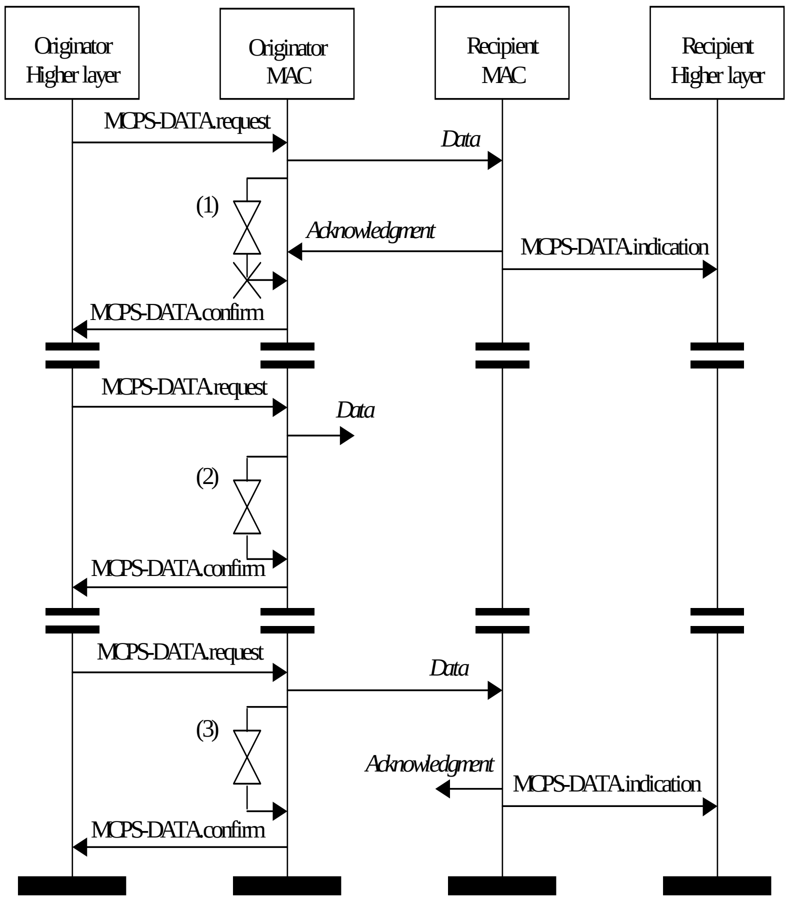


Figure 39—Transmission scenarios for frame reliability

1.5.7 Dynamic channel selection

The PAN coordinator shall initiate dynamic channel selection if the current communication requirements cannot be satisfied due to adverse conditions of the channel.

To locate another channel, the PAN coordinator shall first perform an energy detection scan. The PAN coordinator shall use the results of the energy detection scan to select the channel that has the lowest detected energy. The PAN coordinator shall then send a coordinator realignment command using the broadcast device address (0xff) on its existing channel indicating its intention to change. The PAN coordinator shall then switch to the new channel and commence transmitting its beacons. If no suitable channel is found during the energy detection scan, the PAN coordinator shall remain on its current logical channel.

Devices that become orphaned when the PAN coordinator switches channels shall perform the orphaned device realignment procedure (See subclause 1.5.4.3).

1.5.8 GTS allocation and management

A Guaranteed Time Slot (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 only be allocated by the PAN coordinator and it shall only be used for communications between the PAN coordinator and a device. A single GTS may extend over one or more superframe slots. The PAN coordinator may allocate up to seven GTSs at the same time, providing 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 requested GTS 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. 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 this, 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 identifier, 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. Each FFD device may request one transmit GTS and/or one receive GTS. For each allocated GTS, the device shall be able to store its identifier, 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. 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.

If a device loses synchronization with the PAN coordinator, all its GTS allocations shall be lost.

The use of GTSs by an RFD is optional.

1.5.8.1 CAP maintenance

The PAN coordinator shall always preserve the minimum CAP length of *aMinCAPLength*, and take preventative action if this is not satisfied. Before transmitting a beacon with a payload, the MAC sublayer of the PAN coordinator shall evaluate whether the beacon will grow to such an extent that the CAP length drops below *aMinCAPLength*. If this is the case, the MAC sublayer shall not add the payload to the beacon frame and shall then attempt to deallocate the first GTS that follows the CAP, using the procedure described in subclause 1.5.8.4. The MAC sublayer shall apply this test iteratively until the payload is able to fit in the beacon frame, while still leaving a CAP length of at least *aMinCAPLength*. If the addition of the payload to the beacon frame does not reduce the CAP length below *aMinCAPLength*, the MAC sublayer shall transmit the next beacon with the payload.

1.5.8.2 Allocating a GTS

A device is instructed to request the allocation of a new GTS through the MLME-GTS.request primitive, with a GTS identifier of 0 and 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 subclause 1.3.3.1) to the PAN coordinator. The request type shall be set to indicate a GTS allocation, the GTS identifier field shall be set to 0 and the GTS characteristics field shall be set to the desired characteristics specified in the MLME-GTS.request primitive. If the GTS request command is received correctly, the PAN coordinator shall send an acknowledgment frame, thus confirming receipt.

If a GTS request command is received while the PAN coordinator is currently servicing a previous GTS request command, it shall send the GTS allocation command with the GTS status field indicating that a GTS request is in progress.

On receipt of a GTS request command containing a GTS identifier of 0, the PAN coordinator shall first check if there is available capacity in the current superframe, based on the remaining length of the contention period 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 *aResponseWaitTime* symbols.

When the PAN coordinator determines whether capacity is available for the requested GTS, it shall assign an identifier to the GTS, storing its specifications, and generate the GTS allocation command (See subclause 1.3.3.2) containing the new GTS identifier and a status indicating a successful allocation (See Table 42). If the maximum number of GTSs has been reached, the PAN coordinator shall generate the GTS allocation command with the GTS length field set to 0 and the GTS status field indicating that the maximum number of GTSs has been reached. If capacity is not available, the PAN coordinator shall generate the GTS allocation command with the GTS identifier field set to 0, the GTS length field set equal to the current number of slots remaining in the CAP that can be allocated to a GTS and the GTS status field indicating that no space is available in the CAP. The GTS allocation command shall be sent to the device requesting allocation using indirect addressing, i.e. the address of the device requesting a GTS allocation shall be added to the address list field of the beacon, indicating a pending message and extracted at the discretion of the device. In this case, the GTS allocation command shall be added to the transaction list of the coordinator.

If capacity is available, the GTS identifier field shall contain the identifier of the new GTS, the GTS start slot shall contain the starting slot for the GTS and the GTS status field shall indicate a successful GTS allocation.

On receipt of the acknowledgement to the GTS request command, the device shall wait for *aResponseWaitTime* symbols for the PAN coordinator to make its allocation decision. After this time, the device shall attempt to extract the GTS allocation command from the PAN coordinator according to the procedure described in subclause 1.5.6.3. If the device subsequently does not extract a frame from the PAN coordinator, it shall issue the MLME-GTS.confirm primitive with a status of NO_DATA, and the allocation attempt shall be deemed a failure.

If the GTS allocation command is received correctly, the device shall send an acknowledgment frame, thus confirming receipt. The receipt of the GTS allocation command with a successful GTS status value shall indicate a successful allocation attempt; any other status value shall indicate an unsuccessful allocation attempt and the GTS shall not be used.

On receipt of the GTS allocation command with a successful GTS status, the device requesting the GTS shall store the GTS allocation information for future use. If the device requested a transmit GTS, the MLME shall set *macTxGTSId* to the identifier returned in the GTS allocation command. Similarly, if the device requested a receive GTS, the MLME shall set *macRxGTSId* to the identifier returned in the GTS allocation command. The MLME shall then issue the MLME-GTS.confirm primitive with the GTS identifier, GTS length and GTS status fields from the GTS allocation command.

On receipt of the acknowledgement frame, the PAN coordinator shall begin transmitting beacons with the final CAP slot subfield of the superframe specification field indicating the new superframe slot of the decreased CAP size. On receipt of a beacon with this new decreased CAP size, the device that has just successfully requested a new GTS allocation shall begin using the GTS.

1.5.8.3 GTS usage

When the MAC sublayer of a device receives a MCPS-DATA.request primitive with the TxOptions parameter indicating a GTS transmission, it shall transmit the data during the GTS, identified in *macTxGTSId*, 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. If *macRxGTSId* is non-zero, the MAC sublayer of the device shall enable its receiver at a time prior to the start of the GTS, indicated by this value, and disable it at the end of the GTS.

When the MAC sublayer of the PAN coordinator receives a MCPS-DATA.request primitive with the TxOptions parameter indicating a GTS transmission, it shall defer the transmission until the start of the receive GTS of the intended recipient. The addresses of those devices with messages requiring a GTS transmission shall not be added to the list of pending addresses in the beacon frame (See subclause 1.5.5). 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 inter-frame spacing, suitable to the size of the data frame, can be completed before the end of the GTS.

1.5.8.4 Deallocating a GTS

A device is instructed to request the deallocation of an existing GTS through the MLME-GTS.request primitive, using a previously allocated GTS identifier. From this point onwards, the GTS to be deallocated shall not be used by the device and either *macRxGTSId* or *macTxGTSId* shall be set to 0, depending on whether the GTS is a receive or a transmit GTS, respectively.

To request the deallocation of an existing GTS, the MLME shall send the GTS request command (See subclause 1.3.3.1) to the PAN coordinator. The request type shall be set to indicate a GTS deallocation and the GTS identifier field shall be set to the identifier of the GTS to deallocate. If the GTS request command is received correctly, the PAN coordinator shall send an acknowledgment frame, thus confirming receipt.

On receipt of a GTS request command with the request type set to indicate a GTS deallocation, the PAN coordinator shall attempt to deallocate the GTS. If the GTS identifier subfield of the GTS request command does not contain the identifier of a known GTS, the PAN coordinator shall send a GTS allocation command to the device with the supplied identifier and a status indicating an invalid GTS identifier. If the GTS identifier subfield of the GTS request command contained the identifier of a known GTS, the PAN coordinator shall remove the GTS from its list of allocated GTSs and send a GTS allocation command to the device. The GTS identifier subfield shall be set to the identifier of the GTS being deallocated and all other fields and subfields shall be set to 0.

If the GTS allocation command is received correctly, the device shall send an acknowledgment frame, thus confirming receipt. The device shall then issue the MLME-GTS.confirm primitive with the GTS identifier of the deallocated GTS.

In the case when a GTS deallocation is initiated by the PAN coordinator, the MLME shall send the GTS allocation command containing the identifier of the GTS to deallocate. If the GTS allocation command is received correctly, the device shall send an acknowledgment frame, thus confirming receipt. The device shall then set either *macRxGTSId* or *macTxGTSId* to 0, depending on the direction of the deallocated GTS and issue the MLME-GTS.indication primitive with the GTS identifier of the deallocated GTS.

1.5.8.5 GTS reallocation

The deallocation of a GTS may result in the superframe becoming fragmented. For example, Figure 40 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 during 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).

.

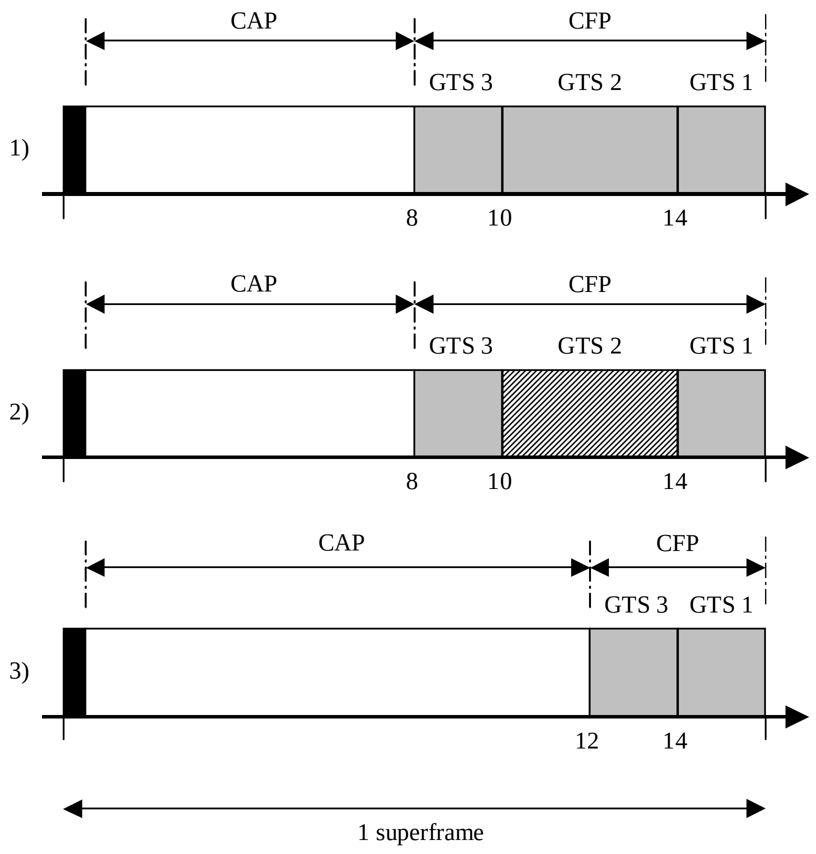


Figure 40—CFP defragmentation on GTS deallocations

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; it does this by reallocating all remaining GTSs. The reallocation shall be undertaken in descending starting slot order, starting with the GTS furthest from the CAP.

For each device with an allocated GTS, the PAN coordinator shall send a GTS allocation command (See subclause 1.3.3.2) using indirect transmission, i.e. the address of the device shall be added to the address list field of the beacon, indicating a pending message. In this case, the GTS allocation command shall be added to the transaction list of the coordinator. Each GTS allocation command shall contain the identifier and length of the existing GTS, a new starting slot (which may be the same as the original starting slot), and a GTS status value indicating that the GTS has been reallocated. The new starting slot is computed so that no space is left between this GTS and the end of the CFP (for the first GTS reallocated) or this GTS and the GTS with a higher starting slot value (for all other GTSs).

If the GTS allocation command is successfully received by a device it shall send an acknowledgement frame, thus confirming its receipt. The device shall note the new reallocated GTS specification but shall only begin using it when the CAP length, as implied by the final CAP slot subfield of the beacon frame of its PAN coordinator, increases.

When the PAN coordinator has attempted to notify each device with an allocated GTS, it shall begin to transmit beacons with the new increased CAP size. If a device has an allocated GTS and has received a valid GTS allocation command, it shall update the starting slot of the specified GTS and begin using the reallocated GTS immediately on the reception of a beacon frame containing the new increased CAP size. If a device has an allocated GTS but has not received a valid GTS allocation command, its GTSs shall be invalid and unusable on receiving a beacon frame containing the new increased CAP size. In this case, the device shall attempt to confirm the allocation of its GTSs by sending the GTS request command to the PAN coordinator with a request type indicating a GTS confirmation.

If a GTS request command with a request type of GTS confirmation is successfully received by the PAN coordinator, it shall send an acknowledgement frame, thus confirming its receipt. The device making the request shall delay *aResponseWaitTime* before sending a data request command petitioning the PAN coordinator for a response in the form of a GTS allocation command. The PAN coordinator shall then send a GTS allocation command to the device, containing the specifications of the GTS in question. If the data request command does not arrive at the PAN coordinator before the next beacon frame is transmitted, the PAN coordinator shall use indirect transmission to send the GTS allocation command, i.e. the address of the device shall be added to the address list field of the beacon, indicating a pending message. In this case, the GTS allocation command shall be added to the transaction list of the coordinator.

If a GTS allocation command with a GTS status indicating a confirmation is successfully received by the device, it shall send an acknowledgement frame, thus confirming its receipt. The device shall then update its GTS specification and begin reusing the GTS.

1.5.9 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. The IEEE 802.15.4 protocol supports the following security services (see subclause 5.4.6.1 for definitions):

— Access control
— Data encryption
— Frame integrity
— Sequential freshness

The protocol also provides the following security modes (see subclause 5.4.6.2 for definitions):

— Unsecured mode
— ACL mode
— Secured mode

The information determining how to provide the security is found in the MAC PIB (see Table 45).

1.5.9.1 ACL entries

The MAC PIB ACL contains a single default ACL entry and a number of additional ACL entries. The default ACL entry is known by every device in the PAN and is used in situations in which the device needs to communicate with a second device or with multiple devices that it may not know individually. Individual ACL entries are used in situations in which the device shares a key with a specific known device.

The default ACL entry consists of *macDefaultSecurity*, which indicates if security is in use for devices not in the ACL, *macDefaultSecuritySuite*, which indicates the default security suite to use for frames to be sent or received from devices not in the ACL and *macDefaultSecurityMaterial*, which indicates the keying material to use in secure communications involving frames to be sent or received from devices not in the ACL. If *macDefaultSecurity* is set to FALSE, *macDefaultSecuritySuite* and *macDefaultSecurityMaterial* shall not be used.

The additional ACL entries are contained in *macACLEntryDescriptorSet.* Each ACL entry consists of a PAN identifier, a 64-bit extended address, a short address of the trusted device, which may include the default associated address 0x00ff, an associated security suite, and associated keying material.

1.5.9.2 Functional description of unsecured mode

Unsecured mode is the default security mode for the MAC sublayer and provides no MAC sublayer security. A device operating in unsecured mode shall not utilize the ACL entries and shall not perform any security related operations on incoming frames. When a device receives a frame while in this mode, the MAC sublayer shall perform its filtering operations on the incoming frame, as described in subclause 1.5.6.2, before checking the security enabled subfield. If the security enabled subfield in the frame is set to 1, the MAC sublayer shall discard the frame and the MLME shall issue the MLME-SECURITY-ERROR.indication primitive to the higher layer with a ReasonCode of UNAVAILABLE_KEY unless the device is performing an active or passive scan (See subclause 1.1.11.1). If the device is performing an active or passive scan, the device shall accept beacon frames with the security enabled subfield set to 1 and set the SecurityUse, ACLEntry and SecurityFailure fields of the PAN descriptor corresponding to that beacon to TRUE, FALSE and TRUE, respectively (see Table 29). If the MAC sublayer receives a frame with the security enabled subfield set to 0, the MLME shall set the SecurityUse and ACLEntry fields to FALSE in the MCPS-DATA.indication primitive (see subclause 1.1.1.3) issued to the higher layer.

1.5.9.3 Functional description of ACL mode

ACL mode provides a mechanism for the MAC sublayer to indicate to the higher layer if a received frame purportedly originated from a device in the ACL. A device operating in ACL mode shall not make any modifications to the MAC frames or perform any cryptographic operations on the frames. As a result, ACL mode provides only a means for the device to filter received frames according to the source address in the frame, but not a means to securely determine the originator of the frame. When a device receives a frame while in this mode, the MAC sublayer shall perform its filtering operations on the incoming frame, as described in subclause 1.5.6.2, before checking the security enabled subfield. If the security enabled subfield in the frame is set to 1, the MAC sublayer shall discard the frame and the MLME shall issue the MLME-SECURITY-ERROR.indication primitive to the higher layer with a ReasonCode of UNAVAILABLE_KEY, unless the device is performing an active or passive scan (See subclause 1.1.11.1). If the device is performing an active or passive scan, the device shall accept beacon frames with the security enabled subfield set to 1 and set the SecurityUse, ACLEntry and SecurityFailure fields of the PAN descriptor corresponding to that beacon to TRUE, FALSE and TRUE, respectively (see Table 29). If the MAC sublayer receives a frame with the security enabled subfield set to 0, the MLME shall set the SecurityUse field to FALSE and the ACLEntry field to TRUE or FALSE depending on whether the originator of the frame is in the ACL in the MCPS-DATA.indication primitive (see subclause 1.1.1.3) issued to the higher layer. The device shall determine if a device is in the ACL by searching *macACLEntryDescriptorSet* for an individual ACL entry with a value of ACLPANId that matches the received PAN identifier and a value of ACLExtendedAddress or ACLShortAddress that matches the received source address. If no source address is present in the frame, the ACLEntry field shall be set to FALSE.

1.5.9.4 Functional description of secured mode

Secured mode provides a mechanism for the MAC sublayer to both use the ACL functionality and provide cryptographic protection on incoming and outgoing frames. While in this mode, if the MAC sublayer receives an incoming frame or a request from the higher layer to transmit a frame, the MAC sublayer shall process the frame as specified in the following subclauses.

1.5.9.4.1 Processing outgoing frames in secured mode

While in secured mode, if the MLME receives a message from a higher layer to prepare a secure frame for transmission (i.e. the security enabled bit in the TxOptions is set to 1), the MLME shall scan the entries in the ACL for the correct entry to use. The MLME shall first search through *macACLEntryDescriptorSet* to find an entry in which the ACLPANId, ACLExtendedAddress and ACLShortAddress fields match the destination address information of the frame to be created. If a match is found, the MLME shall select the security suite from the associated ACLSecuritySuite field and the security material from the associated ACLSecurityMaterial field for use on the outgoing frame.

If the MLME is unable to locate an ACLPANId and ACLExtendedAddress or ACLShortAddress that matches the destination address information of the frame to be created, the MLME shall examine *macDefaultSecurity*. If *macDefaultSecurity* is equal to TRUE, the MLME shall select the security suite indicated in *macDefaultSecuritySuite* and the security material from *macDefaultSecurityMaterial* for use on the outgoing frame.

If the MLME is unable to locate an ACLPANId and ACLExtendedAddress or ACLShortAddress that matches the destination address information of the frame to be created and *macDefaultSecurity* is equal to FALSE, the MLME shall return an MLME-SECURITY-ERROR.indication to the higher layer with the ReasonCode set to UNAVAILABLE_KEY.

After the MLME obtains the appropriate security suite and security material from the ACL, the MLME shall first set the security enabled subfield in the frame control field to 1 before performing the cryptographic operations on the frame.

If the security suite specifies the use of encryption, the encryption operation shall be applied to data in the payload field within the MAC payload (i.e. the beacon payload field, command payload field or data payload field) of the frame only. The remaining fields shall be left unencrypted. If a frame does not contain a payload field, encryption shall not be used. The result of the encryption operation shall be inserted into the payload field of the frame in the place of the original data.

If the security suite specifies the use of an integrity code, the integrity code shall be applied to the MAC header concatenated with the MAC payload (see subclause 1.2 for information on the MAC frame format). The result of the integrity code computation shall be placed after the payload field within the MAC payload of the frame in addition to any other data in the payload field. If the payload field does not contain any data, then it shall contain only the integrity code. Integrity codes shall not be used for acknowledgement frames.

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 beacon payload, command payload or data payload field is defined by the selected security suite (See subclause 1.6).

If any of the security operations fail, the MLME shall issue the MLME-SECURITY-ERROR.indication primitive to the higher layers with a ReasonCode of FAILED_SECURITY_CHECK and shall not transmit the requested frame. If the length of the resulting frame is longer than *aMaxMACFrameSize*, the MLME shall issue the MLME-SECURITY-ERROR.indication primitive to the higher layers with a ReasonCode of FRAME_TOO_LONG and shall not transmit the requested frame.

If the security operations have been successfully performed and the payload field within the MAC payload has been modified appropriately, the device shall compute the FCS over the modified frame.

1.5.9.4.2 Processing incoming frames in secured mode

Any incoming frame may be protected by security. While in secured mode, if the MLME receives a frame it shall perform its filtering operations on the incoming frame, as described in subclause 1.5.6.2, before checking the security enabled subfield to determine if security is used for that frame.

If the security enabled subfield in the frame control field is set to 0, the device shall not perform security operations on the frame and shall process the frame as if it were in ACL mode (See subclause 1.5.9.3).

If the security enabled subfield in the frame control field is set to 1 the MLME shall scan the entries in the MAC PIB ACL for the correct entry to use. The MLME shall first search through *macACLEntryDescriptorSet* to find an entry in which the ACLPANId, ACLExtendedAddress and ACLShortAddress fields match the source address information of the frame received. If a match is found, the MLME shall select the security suite from the associated ACLSecuritySuite field and the security material from the associated ACLSecurityMaterial field for use on the incoming frame.

If the MLME is unable to locate an ACLPANId and ACLExtendedAddress or ACLShortAddress that matches the source address information of the received frame, the MLME shall examine *macDefaultSecurity*. If *macDefaultSecurity* is equal to TRUE, the MLME shall select the security suite from *macDefaultSecuritySuite* and the security material from *macDefaultSecurityMaterial* for use on the incoming frame.

If the MLME is unable to locate an ACLPANId and ACLExtendedAddress or ACLShortAddress that matches the source address information of the received frame and *macDefaultSecurity* is equal to FALSE, the MLME shall return an MLME-SECURITY-ERROR.indication to the higher layer with the ReasonCode set to UNAVAILABLE_KEY unless the device is performing an active or passive scan (See subclause 1.1.11.1). If the device is performing an active or passive scan, the device shall accept secure beacon frames for which the corresponding security material cannot be found and set the SecurityUse, ACLEntry and SecurityFailure fields of the PAN descriptor corresponding to that beacon to TRUE, FALSE and TRUE, respectively (see Table 29).

After the MLME obtains the appropriate security suite and security material from the ACL, the MAC sublayer shall apply the operations defined by the security suite using the security material to the frame.

If the security suite specifies the use of encryption, the decryption operation shall be applied to data in the payload field within the MAC payload (i.e. the beacon payload field, command payload field or data payload field) of the frame only. If a frame does not contain any data in the payload field, decryption shall not be used. The result of the decryption operation shall be inserted into the payload field of the frame in the place of the original encrypted data.

If the security suite specifies the use of an integrity code, the integrity code shall be checked by first removing the integrity code and any other security suite specific data (e.g. the frame counter and key sequence counter) from the payload field within the MAC payload and then verifying the integrity code on the MAC header concatenated with the MAC payload.

The ordering and exact manner of performing the decryption and integrity operations and the location of the security data within the payload field is defined by the security suite in use (See subclause 1.6).

If at least one of the security operations fail, the MLME shall return an MLME-SECURITY-ERROR.indication to the higher layers with the ReasonCode set to FAILED_SECURITY_CHECK and shall not perform any additional operations on the received frame unless the device is performing an active or passive scan, 1.1.11.1. If the device is performing an active or passive scan, the device shall accept beacon frames that cause a security operation failure and set the SecurityUse, ACLEntry and SecurityFailure fields of the PAN descriptor corresponding to that beacon to TRUE, TRUE and TRUE, respectively (see Table 29).

If the security operations have been successfully performed and the payload field within the MAC payload has been modified appropriately, the device shall then continue to process the frame. In the indication of the frame to the higher layer, the MAC shall set the SecurityUse field to TRUE and set the ACLEntry field to TRUE if the key used in the operations was found in *macACLEntryDescriptorSet* or FALSE if the key used in the operations was found in *macDefaultSecurityMaterial*.

1.6 Security Suite Specification

Security suites may be used when a device is operating in secured mode. A security suite consists of a set of operations to perform on MAC frames that provide security services. The security suite name indicates the symmetric cryptography algorithm, mode and integrity code bit length. The bit length of the integrity code is less than or equal to the block size of the symmetric algorithm and determines the probability that a random guess of the integrity code would be correct. This bit length does not correspond to the strength of the underlying algorithm. For all security suites in this standard, the algorithm used shall be AES (See subclause 1.6.1.7). Each device that implements security shall support the AES-CCM-64 security suite (See subclause 1.6.3) and zero or more additional security suites. Each security suite is specified by a one-octet value as shown in Table 50.

Table 50—Security suite list

Identifier	Security Suite Name	Security Services				Subclause
		Access Control	Data Encryption	Frame Integrity	Sequential Freshness (optional)	
0x00	AES-CCM*	X	X	Y/N	X	1.6.2

Editorial note: To be obtained from the Frame Header (viz. Frame Control Field, see also Slide 11 of 02/474r0): 3-bit Security Algorithm Id. For temporary use, I included the encoding hereof below:

Security Algorithm Id: 3-bits, where the leftmost bit specifies whether confidentiality is required (value: 1) or not (value: 0) and where the rightmost 2 bits specify to what extent authenticity is requires: 0-bits (value: 00), 32-bits (value: 01), 64-bits (value: 10), 128-bits (value: 11).

1.6.1 Security suite building blocks

The following definitions and methods are defined for use in the security suites specified in this standard.

1.6.1.1 Bit ordering

For security operations in this standard, a bit is defined to be an element of the set {0, 1}. An octet (also called a byte) is defined to be a bit string of length 8 arranged in the order in which it would be transmitted, where bit 0 is the first bit in the octet and bit 7 is the last bit in the octet. An octet string (also called a byte string) is an array of octets arranged in the order in which it would be transmitted, where octet 0 is the first octet. The terms first and last and leftmost and rightmost are used to distinguish the ends of octet strings (first and leftmost are equivalent; last and rightmost are equivalent). Within an octet, the terms first and last, leftmost and rightmost and low-order and high-order are used for the order of the bits (first, leftmost and low-order are equivalent; last, rightmost and high-order are equivalent).

Note that the first bit in an octet string is indexed as bit 0 of octet 0 and represents the low-order bit of the first octet. The sixteenth bit in an octet string is indexed as bit 7 of octet 2 and represents the high-order bit of the second octet.

1.6.1.2 Concatenation

In this standard, concatenation of two octet strings *a* and *b* of length *m* and *n* respectively, denoted $a||b$, consists of the octet string of length $m+n$ with the leftmost *m* octets equal to *a* and the rightmost *n* octets equal to *b*.

1.6.1.3 Integer encoding and counter incrementing

When an integer is represented as an octet string, the first octet (octet 0) corresponds to the most significant octet and the eighth bit (bit 7) within the first octet corresponds to the most significant bit. An octet string *A* of length *n,* is written as a bit string $A = a_{0,0}a_{0,1}a_{0,2} \ldots a_{n-1,6}a_{n-1,7}$. An octet string or bit string is converted to an integer *I* by assigning each bit $a_{i,j}$ the value $a_{i,j}*2^{8i+j}$ and setting *I* to be the sum of all of the values.

Editorial note RS: this big-endian representation of integers as octets is quite costly and inconsistent with other integer encodings in the draft, see also 02/474r0, Slide 12 (Note RS: I did leave it in as such for the moment, though).

As an example of integer encoding, consider the integer 11146, which corresponds to the octet string composed of the two octets 0x2b 8a. This integer can be represented as the bit string 1101 0100 0101 0001. In this case, the most significant octet (octet 0) is 0x2b and the least significant octet (octet 1) is 0x8a. The least significant bit (bit 0) of octet 0 has the value 1, as does the most significant bit (bit 7) of octet 1. The integer 1 can be represented as a four octet string by 0x00 00 00 01 and as a bit string by 0000 0000 0000 0000 0000 0000 1000 0000.

The counter incrementing operation in this standard takes as input an integer that is encoded as an octet string of length *n*. When the counter incrementing operation is invoked, the integer shall be incremented (increased by 1). If the incremented integer is less than 2^{8n}, the operation shall return the octet string encoding of the new integer. Otherwise the operation shall return an error (the counter incrementing operation has caused the integer value to roll over).

1.6.1.4 CTR encryption (REMOVE COMPLETELY)

The counter mode (CTR) symmetric encryption algorithm used in this standard consists of the generation of a key stream using a block cipher in counter mode, with a given key and nonce, and performing an XOR of the key stream with the plaintext and integrity code. A nonce is either a time stamp, a counter, or a special marker intended to prevent unauthorized message replay. The decryption operation consists of the generation of the key stream and the XOR of the key stream with the ciphertext to obtain the plaintext.

All of the above operations shall be performed as specified in Annex B.2, which defines CTR encryption in the general sense. A security suite implementing CTR encryption will specify any parameters left unspecified in Annex B.2 according to the requirements of that particular security suite.

1.6.1.5 CBC-MAC authentication (REMOVE COMPLETELY)

The cipher block chaining message authentication code (CBC-MAC) symmetric authentication algorithm used in this standard consists of the generation of an integrity code using a block cipher in CBC mode computed on a message that includes the length of the authenticated data at the beginning of the data itself. The verification operation consists of the computation of this integrity code and comparison to the received integrity code.

All of the above operations shall be performed as specified in Annex B.3. Security suites implementing CBC-MAC authentication specify the parameters that are left undefined in Annex B.3.

1.6.1.6 CTR + CBC-MAC (CCM*) combined encryption and authentication

The counter mode encryption plus cipher block chaining message authentication code (CCM*) combined symmetric encryption and authentication mechanisms used in this standard consists of the generation of an integrity code followed by the encryption of plaintext data and the integrity code. The output consists of the encrypted data and the encrypted integrity code.

The symmetric authentication operation used in this security suite consists of the generation of an integrity code using a block cipher in CBC mode computed on a nonce followed by (optional) padded authentication data followed by (optional) padded plaintext data. The verification operation consists of the computation of this integrity code and comparison to the received integrity code.

The symmetric encryption operation used in this security suite consists of the generation of a key stream using a block cipher in counter mode with a given key and nonce and performing an XOR of the key stream with the plaintext and integrity code. The decryption operation consists of the generation of the key stream and the XOR of the key stream with the ciphertext to obtain the plaintext and integrity code.

All of the above operations shall be performed as specified in Annex B.1(Note: use extended version: AES-CCM*). Security suites implementing CCM* combined encryption and authentication specify the parameters that are left undefined in Annex B.1.

1.6.1.7 AES encryption

The advanced encryption standard (AES) encryption algorithm used in this security suite shall be performed as specified in the FIPS 197 standard (See Subclause 2.5). This encryption algorithm is parameterized by the use of 128-bit keys and 128-bit block size. (Note: Clause 7.6.1.6 and Clause 7.6.1.7 should be stated in different order, since the AES-CCM* mode is only defined for block-ciphers with block length 128 bits (16 bytes).

1.6.1.8 PIB security material

This subclauses describes the formats of the security information stored in the MAC PIB. This information is dependent on the individual security suite selected and is referenced in the following subclauses.

The symmetric key is the AES key for this ACL entry that shall be used to perform either CTR encryption, CCM encryption and authentication or CBC-MAC authentication.

The frame counter is the running counter that shall be included in the payload field in the MAC payload of the MAC frame. This value shall be incremented each time a secure frame is transmitted and shall not roll-over. The frame counter shall be reset if a new key is issued.

The key sequence counter is a counter that is fixed by the higher layer that shall be included in the payload field in the MAC payload of the MAC frame. This value may be used by the higher layers to facilitate key management by allowing the higher layers to reset the frame counter and not generate the same nonce twice.

Comment note RS: It is more useful to define the key sequence counter as an octet that defines the symmetric key used. Currently, there is no way to distinguish different keys, which becomes quite a pain, when key updates are deemed necessary. (Note RS: currently, I did not change anything however-- see my Sponsor Ballot comment for rationale why the current deifnition is not very useful).

The optional external frame counter and optional external key sequence counter are fields that may be stored in the ACL entry that represents the value of the last received frame counter and key sequence counter respectively in a secure frame corresponding to this ACL entry. If the optional external frame counter and optional external key sequence counter fields are included in the ACL entry, the MAC shall use them to verify the sequential freshness of received secure frames.

~~1.6.2 AES-CTR security suite (REMOVE COMPLETELY)~~

~~The AES-CTR security suite is used when a device is operating in secured mode. The cryptographic operations in this security suite consist of performing AES-CTR encryption (or decryption) on the payload field within the MAC payload using shared data, the frame counter and the key sequence counter.~~

~~The AES-CTR security suite provides the following security services:~~

— ~~Access control, defined in subclause 5.4.6.1.1~~
— ~~Data encryption, defined in subclause 5.4.6.1.2~~
— ~~Sequential freshness, defined in subclause 5.4.6.1.4(optional)~~

~~1.6.2.1 Data formats~~

~~The following subclauses define the data formats used in this security suite.~~

~~1.6.2.1.1 MAC PIB formats~~

~~For the AES-CTR security suite, the security material stored in *macDefaultSecurityMaterial* or the ACLSecurityMaterial field in the ACL consists of a symmetric key, a frame counter and a key sequence counter as well as an optional frame counter and sequence counter that may be used for incoming frames. If these optional fields are included in the security material, the optional operations specified in subclause 1.6.2.3.2 shall be performed. When these optional operations are performed, the security suite provides the sequential freshness security service on received frames. Figure 41 specifies the order and length of the AES-CTR security material components.~~

Figure 41—AES-CTR security material

Octets: 16	4	1	(4)	(1)
Symmetric key	Frame counter	Key sequence counter	Optional external frame counter	Optional external key sequence counter

~~1.6.2.1.2 Protected payload field formats~~

~~In the AES-CTR security suite, the payload field in the MAC payload of a protected frame consists of the frame counter, the key sequence counter and the encrypted payload. Figure 42 specifies the order and length~~

~~of the subfields of the payload field of an AES-CTR secured frame. The length of the encrypted payload field is equal to the length of the payload field before it was encrypted.~~

Figure 42—AES-CTR payload field

Octets: 4	1	variable
Frame counter	Key sequence counter	Encrypted payload

~~1.6.2.1.3 CTR input blocks~~

~~In the AES-CTR security suite, the input blocks to the CTR encryption function for generating the key stream consist of a flag octet, the address of the device sending the frame, the frame and key sequence counter values and the block counter value. Figure 43 specifies the order and length of the subfields of the CTR input blocks, the use of which is described in subclauses 1.6.2.3.1 and 1.6.2.3.2. These input blocks correspond to the counters~~ ~~T_1, T_2, ... , T_n~~ ~~as specified in Annex B.2.~~

Figure 43—AES-CTR input block

Octets: 1	8	4	1	2
Flags	Source address	Frame counter	Key sequence counter	Block counter

~~The flags octet used in AES-CTR mode, which is used as padding for the input block, is formatted as specified in Figure 44. The bit values in this flag octet are chosen simply to distinguish the AES-CTR flags from the AES-CCM flags (see Annex B.1) for information on the AES-CCM flags).~~

Figure 44—AES-CTR flags field

Bits: 0	1	2-6	7
0	1	0	1

~~1.6.2.2 Security parameters~~

~~The CTR encryption operation in this security suite, as defined in subclause 1.6.1.4, shall be parameterized by the following: the underlying block cipher shall be the AES encryption algorithm as specified in subclause 1.6.1.7, the counter input blocks shall be formatted as specified in subclause 1.6.2.1.3 where each input block is the same except that the block counter is set to 0 for the first block and is incremented as specified in subclause 1.6.1.3 for each successive input block. In other words, the~~ ~~i^{th}~~ ~~input block~~ ~~T_i~~ ~~shall have the block counter value set to i-1 (see Annex B.2) for further explanation).~~

~~1.6.2.3 Security operations~~

~~The following subclauses specify the operations performed on outgoing and incoming frames.~~

~~1.6.2.3.1 Outgoing frame operations~~

~~When the AES-CTR security suite is invoked to protect an outgoing frame, the MAC sublayer shall perform the following operations:~~

1) ~~Obtain its own 64-bit extended address, *aExtendedAddress*, along with the frame counter and the sequence counter from the MAC PIB, and construct the counter input blocks as specified in subclause 1.6.2.2.~~
2) ~~Encrypt the payload field in the MAC payload of the frame using CTR encryption, as specified in subclause 1.6.1.4, with the parameters specified in subclause 1.6.2.2, and using the counter input blocks from step 1.~~
3) ~~Combine the frame counter, sequence counter and output from step 2, as specified in subclause 1.6.2.1.2, to obtain the new payload field.~~
4) ~~Increment the frame counter as specified in subclause 1.6.1.3 and, if the incrementing succeeds, insert the new counter value into the MAC PIB. If the incrementing operation fails due to the counter value rolling over, the device shall abort the operation and issue the MLME-SECURITY-ERROR.indication primitive to the higher layer with a ReasonCode of FAILED_SECURITY_CHECK.~~

~~1.6.2.3.2 Incoming frame operations~~

~~When the AES-CTR security suite is invoked to protect an incoming frame, the MAC sublayer shall perform the following operations:~~

1) ~~(optional) Ensure sequential freshness by verifying that the received key sequence counter is greater than or equal to the last known key sequence counter from that device. If the key sequence counter is equal to the last known key sequence counter, verify that the received frame counter is greater than or equal to the last known frame counter from that device. If either of these checks fail, the device shall reject the frame and issue the MLME-SECURITY-ERROR.indication primitive to the higher layer with a ReasonCode of FAILED_SECURITY_CHECK. If the checks succeed, the last known sequence counter and last known frame counter shall be set to the received values.~~
2) ~~Obtain the 64-bit extended address of the source either from the frame or from the ACL, extract the frame counter and sequence counter from the payload field in the MAC payload and construct the counter input blocks as specified in subclause 1.6.2.2. If the nonce cannot be constructed due to the data being unavailable, the device shall issue the MLME-SECURITY-ERROR.indication primitive to the higher layers with a ReasonCode of FAILED_SECURITY_CHECK.~~
3) ~~Decrypt the encrypted payload field using CTR decryption, as specified in subclause 1.6.1.4, with the parameters specified in subclause 1.6.2.2 and using the counter input blocks from step 2.~~
4) ~~Replace the existing payload field in the MAC payload with the decrypted data from step 3.~~

1.6.3 AES-CCM* security suite

The AES-CCM* security suite is used when a device is operating in secured mode. The cryptographic operations in this security suite consist of performing AES-CCM* authentication (or verification) on the MAC header concatenated with the MAC payload and encryption (or decryption) on the payload field in the MAC header using shared data, the frame counter and the SecField field. The AES-CCM* security suite can be implemented using either 0-, 32-, 64- or 128-bit integrity codes. (Obviously, 0-bit integrity corresponds to disabling data authenticity/integrity, since the integrity code is the empty string in that case.)

The AES-CCM security suite provides the following security services:

— Access control, defined in subclause 5.4.6.1.1
— Data encryption, defined in subclause 5.4.6.1.2
— Frame integrity (up to a certain level), defined in subclause 5.4.6.1.3
— Sequential freshness, defined in subclause 5.4.6.1.4 (optional)

1.6.3.1 Data formats

The following subclauses define the data formats used in this security suite.

1.6.3.1.1 MAC PIB formats

For the AES-CCM* security suite, the security material stored in *macDefaultSecurityMaterial* or the *ACLSecurityMaterial* field in the ACL consists of a symmetric key, a frame counter and a key sequence countr as well as an optional frame counter and sequence counter that may be used for incoming frames. If these optional fields are included in the security material, the optional operations specified in subclause 1.6.3.3.2 shall be performed. When these optional operations are performed, the security suite provides the sequential freshness security service on received frames. Figure 45 specifies the order and length of the AES-CCM* security material components.

Figure 45—AES-CCM* security material

Octets: 16	**4**	**1**	**(4)**	**(1)**
Symmetric key	Frame counter	Key sequence counter	Optional external frame counter	Optional external key sequence counter

1.6.3.1.2 Protected payload field formats

In the AES-CCM security suite, the payload field in the MAC payload of a protected frame consists of the frame counter, the key sequence counter, the encrypted payload and the encrypted integrity code. Figure 46 specifies the order and length of the subfields of the payload field in the MAC payload of an AES-CCM secured frame. The length of the encrypted payload field is equal to the length of the payload field before it was encrypted. The length of the encrypted integrity code subfield corresponds to the length of integrity code required, i.e. 0 octets for 0-bit, 4 octets for 32-bit, 8 octets for 64-bit or 16 octets for 128-bit data authenticity/integrity..

Figure 46—AES-CCM* MAC payload

Octets: 4	**1**	**variable**	**0, 4, 8, or 16**
Frame counter	Key sequence counter	Encrypted payload	Encrypted integrity code

1.6.3.1.3 AES-CCM* settings

In the AES-CCM* security suite, the nonce input used for the CCM* authentication and encryption function consists of data explicitly included in the frame and data that both devices can independently obtain. Figure 47 specifies the order and length of the subfields of the CCM* nonce. The 1 octet field SecField has as value the 1-octet representation of the Security Services Id (see {xref} 7.6.1), obtained by prepending this 3-bit string with five 0-bits..

Figure 47—AES-CCM* nonce

Octets: 8	4
Source address	Frame counter

1.6.3.2 Security parameters

In this security suite, the CCM* operations as defined in subclause 1.6.1.6 shall be parameterized by the following: the underlying block cipher shall be the AES encryption algorithm as specified in subclause 1.6.1.7, the length in octets of the length field L shall be 2 octets, the length *M* of the authentication field shall be 0,4,8, or 16 octets, depending upon the value of the Security Suite Service Id, and the nonce shall be formatted as specified in subclause 1.6.3.1.3.

1.6.3.3 Security operations

The following subclauses specify the operations performed on outgoing and incoming frames.

1.6.3.3.1 Outgoing frame operations

When the AES-CCM* security suite is invoked to protect an outgoing frame, the MAC sublayer shall perform the following operations:

1) Obtain its own 64-bit extended address, *aExtendedAddress*, along with the frame counter from the MAC PIB, and construct the nonce and SecField as specified in subclause 1.6.3.1.3.
2) Encrypt and authenticate the MAC header and MAC payload in the frame using CCM* authentication and encryption, as specified in subclause 1.6.1.6, with the parameters specified in subclause 1.6.3.2. Use the MAC header and non-payload fields in the MAC payload as the authentication data, *a*, the payload field in the MAC payload as the message, *m*, and the nonce and SecField computed in step 1.
3) Combine the frame counter, sequence counter and output from step 2 (including the encrypted payload and encrypted integrity code), as specified in subclause 1.6.3.1.2, to obtain the new payload field.
4) Increment the frame counter as specified in subclause 1.6.1.3 and, if the incrementing succeeds, insert the new counter value into the MAC PIB. If the incrementing operation fails due to the counter value rolling over, the device shall abort the operation and issue the MLME-SECURITY-ERROR.indication primitive to the higher layers with a ReasonCode of FAILED_SECURITY_CHECK.

1.6.3.3.2 Incoming frame operations

When the AES-CCM security suite is invoked to protect an incoming frame, the MAC sublayer shall perform the following operations:

~~1)~~ ~~REMOVE COMPLETELY: (optional) Ensure sequential freshness by verifying that the received key sequence counter is greater than or equal to the last known key sequence counter from that device. If the key sequence counter is equal to the last known key sequence counter, verify that the received frame counter is greater than or equal to the last known frame counter~~

~~from that device. If either of these checks fail, the device shall reject the frame and issue the MLME-SECURITY-ERROR.indication primitive to the higher layer with a ReasonCode of FAILED_SECURITY_CHECK. If the checks succeed, the last known sequence counter and last known frame counter shall be set to the received values.~~

2) Obtain the 64-bit extended address of the source either from the frame or from the ACL, remove the frame counter and sequence counter from the payload field in the MAC payload and construct the nonce and SecField as specified in subclause 1.6.3.1.3. If this data cannot be constructed due to the data being unavailable, the device shall reject the frame and issue the MLME-SECURITY-ERROR.indication primitive to the higher layers with a ReasonCode of FAILED_SECURITY_CHECK.
3) Decrypt the encrypted payload field and verify the integrity code using CCM* decryption and authentication, as specified in subclause 1.6.1.6, with the parameters specified in subclause 1.6.3.2. Use the MAC header and non-payload fields in the MAC payload as the authentication data, *a*, the encrypted payload field as the message, *m*, and the nonce and SecField computed in step 2. If the integrity code fails, the device shall discard the frame and issue the MLME-SECURITY-ERROR.indication primitive to the higher layers with a ReasonCode of FAILED_SECURITY_CHECK.
4) Replace the existing payload field in the MAC payload with the decrypted data from step 3.

~~1.6.4 AES-CBC-MAC security suite (REMOVE COMPLETELY)~~

~~The AES-CBC-MAC security suite is used when a device is operating in secured mode. The cryptographic operations in this security suite consist of performing AES-CBC-MAC authentication on the MAC header and MAC payload. The AES-CBC-MAC security suite can be implemented using either 32-, 64- or 128-bit integrity codes.~~

~~The AES-CBC-MAC security suite provides the following security services:~~

— ~~Access control, defined in subclause 5.4.6.1.1~~
— ~~Frame integrity, defined in subclause 5.4.6.1.3~~

~~1.6.4.1 Data formats~~

~~The following subclauses define the data formats used in this security suite.~~

~~1.6.4.1.1 MAC PIB formats~~

~~For the AES-CBC-MAC security suite, the security material stored in *macDefaultSecurityMaterial* or the ACLSecurityMaterial field in the ACL consists of a symmetric key. No state information is required between frames. Figure 48 specifies the order and length of the AES-CBC-MAC security material components.~~

Figure 48—AES-CBC-MAC security material

Octets: 16
Symmetric key

~~1.6.4.1.2 Protected payload field formats~~

~~In the AES-CBC-MAC security suite, the payload field in the MAC payload of a protected frame consists of the existing payload followed by the integrity code. Figure 49 specifies the order and length of the subfields of the payload field in the MAC payload of an AES-CBC-MAC secured frame. The length of the integrity code subfield corresponds to the length of integrity code required, i.e. 4 octets for 32-bit, 8 octets for 64-bit or 16 octets for 128-bit.~~

Figure 49—AES-CBC-MAC payload field

Octets: variable	**4, 8 or 16**
Payload	Integrity code

~~1.6.4.1.3 CBC-MAC input blocks~~

~~In the AES-CBC-MAC security suite, the input to the CBC-MAC authentication function for generating the integrity code consists of the length of the data to authenticate (not including the length field itself) followed by the MAC header followed by the MAC payload. The input is broken up into 16-octet blocks starting from the left and proceeding to the right until the last block, which may be smaller than 16-octets depending on the length of the total input. Figure 50 specifies the order and length of the subfields of the CBC-MAC input.~~

Figure 50—AES-CBC-MAC input

Octets: 1	**variable = n**	**variable = m**
Length = n + m	MAC header	MAC payload

~~1.6.4.2 Security parameters~~

~~In this security suite, the CBC-MAC operations as defined in subclause 1.6.1.5 shall be parameterized by the following: the underlying block cipher shall be the AES encryption algorithm as specified in subclause 1.6.1.7, the input to the CBC-MAC function shall be formatted as specified in subclause 1.6.4.1.3, the length of the integrity code M shall be 32, 64 or 128 bits, as required.~~

~~1.6.4.3 Security operations~~

~~The following subclauses specify the operations performed on outgoing and incoming frames.~~

~~1.6.4.3.1 Outgoing frame operations~~

~~When the AES-CBC-MAC security suite is invoked to protect an outgoing frame, the MAC sublayer shall perform the following operations:~~

~~1)~~ ~~Determine the length in octets of the MAC header concatenated with the MAC payload (before the security operations are performed) and encode that length as a 1-octet integer, as specified in subclause 1.6.1.3.~~

~~2) Compute the integrity code on the MAC header and MAC payload in the frame using CBC-MAC authentication, as specified in subclause 1.6.1.5, with the parameters specified in subclause 1.6.4.2.~~

~~3) Combine the existing payload field in the MAC payload and output from step 2, as specified in subclause 1.6.4.1.2, to obtain the new payload field.~~

~~1.6.4.3.2 Incoming frame operations~~

~~When the AES-CBC-MAC security suite is invoked to protect an incoming frame, the MAC sublayer shall perform the following operations:~~

~~1) Determine the length in octets of the MAC header concatenated with the MAC payload, before the security operations were applied, and encode that length as a 1-octet integer as specified in subclause 1.6.1.3.~~

~~2) Parse the payload field of the MAC payload into the payload and integrity code subfields, as specified in subclause 1.6.4.1.2, and verify the integrity code using CBC-MAC authentication, as specified in subclause 1.6.1.5, with the parameters specified in subclause 1.6.4.2. Use the length determined in step 1, the MAC header from the received frame and the MAC payload (without the integrity code) as the input data. If the integrity code fails, the device shall discard the frame and issue the MLME-SECURITY-ERROR.indication primitive to the higher layers with a ReasonCode of FAILED_SECURITY_CHECK.~~

~~3) Remove the integrity code from the payload field in the MAC payload.~~

1.7 MSCs illustrating MAC-PHY interaction

This subclause illustrates the main tasks performed by the IEEE 802.15.4 system. 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 a PAN coordinator to start a new PAN are shown in Figure 51. The first action the DME takes after resetting the MAC sublayer is to initiate a scan to search for other PANs in the area. The scan in the MSC refers to Figure 52, which shows the steps for performing an energy detection scan. An active scan can also be used to search the area for neighboring PANs but is not shown here.

Once a new PAN is established, the PAN coordinator is ready to accept requests from other devices to join the PAN. Figure 53 shows the primitives issued by the device requesting association, while Figure 54 illustrates the steps taken by the PAN coordinator allowing association. In the process of joining a PAN, the device requesting association will perform a passive scan to determine which PANs in the area are allowing association; Figure 55 details the primitives necessary to complete a passive scan. The association procedure also applies to devices joining the PAN through a coordinator that is not the central PAN coordinator (See clause 3 for definitions of both coordinator and PAN coordinator).

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 56, while those taken by the recipient are shown in Figure 57.

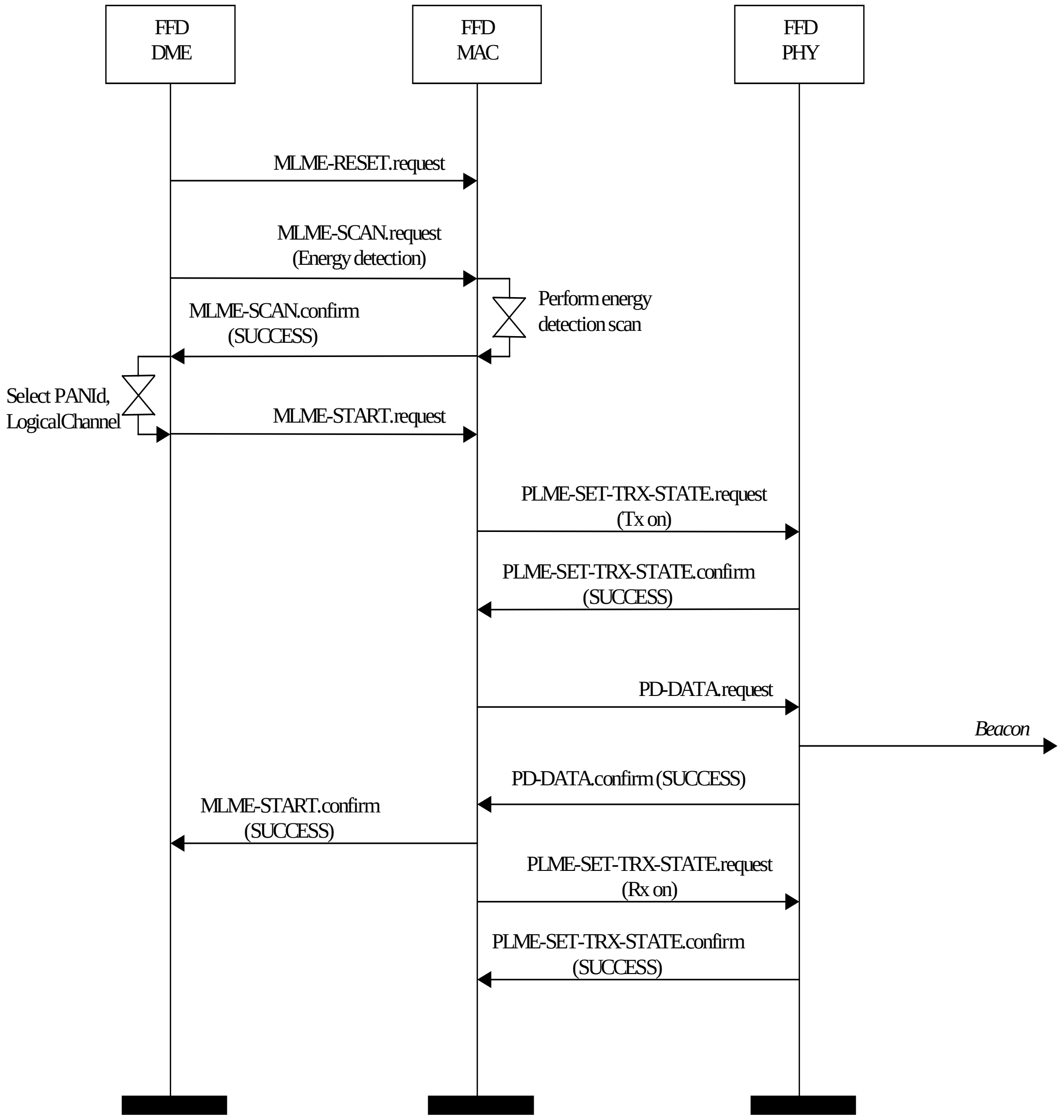

FFD DME
FFD MAC
FFD PHY
MLME-RESET.request
MLME-SCAN.request (Energy detection)
Perform energy detection scan
MLME-SCAN.confirm (SUCCESS)
Select PANId, LogicalChannel
MLME-START.request
PLME-SET-TRX-STATE.request (Tx on)
PLME-SET-TRX-STATE.confirm (SUCCESS)
PD-DATA.request
Beacon
PD-DATA.confirm (SUCCESS)
MLME-START.confirm (SUCCESS)
PLME-SET-TRX-STATE.request (Rx on)
PLME-SET-TRX-STATE.confirm (SUCCESS)

Figure 51—PAN start message sequence chart - PAN coordinator

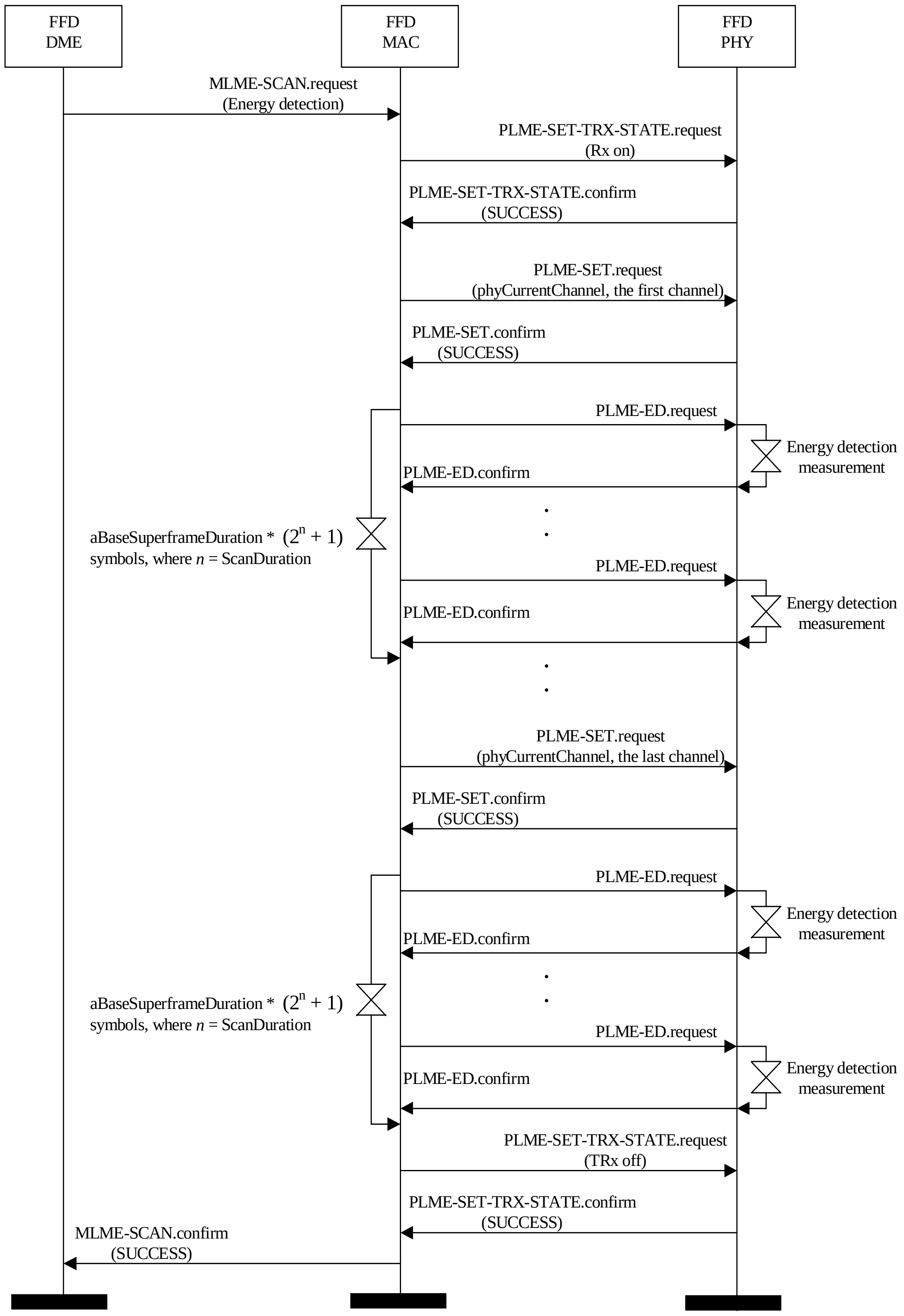

FFD DME
FFD MAC
FFD PHY
MLME-SCAN.request (Energy detection)
PLME-SET-TRX-STATE.request (Rx on)
PLME-SET-TRX-STATE.confirm (SUCCESS)
PLME-SET.request (phyCurrentChannel, the first channel)
PLME-SET.confirm (SUCCESS)
PLME-ED.request
Energy detection measurement
PLME-ED.confirm
aBaseSuperframeDuration * $(2^n + 1)$ symbols, where n = ScanDuration
PLME-ED.request
Energy detection measurement
PLME-ED.confirm
PLME-SET.request (phyCurrentChannel, the last channel)
PLME-SET.confirm (SUCCESS)
PLME-ED.request
Energy detection measurement
PLME-ED.confirm
aBaseSuperframeDuration * $(2^n + 1)$ symbols, where n = ScanDuration
PLME-ED.request
Energy detection measurement
PLME-ED.confirm
PLME-SET-TRX-STATE.request (TRx off)
PLME-SET-TRX-STATE.confirm (SUCCESS)
MLME-SCAN.confirm (SUCCESS)

Figure 52—Energy detection scan message sequence chart

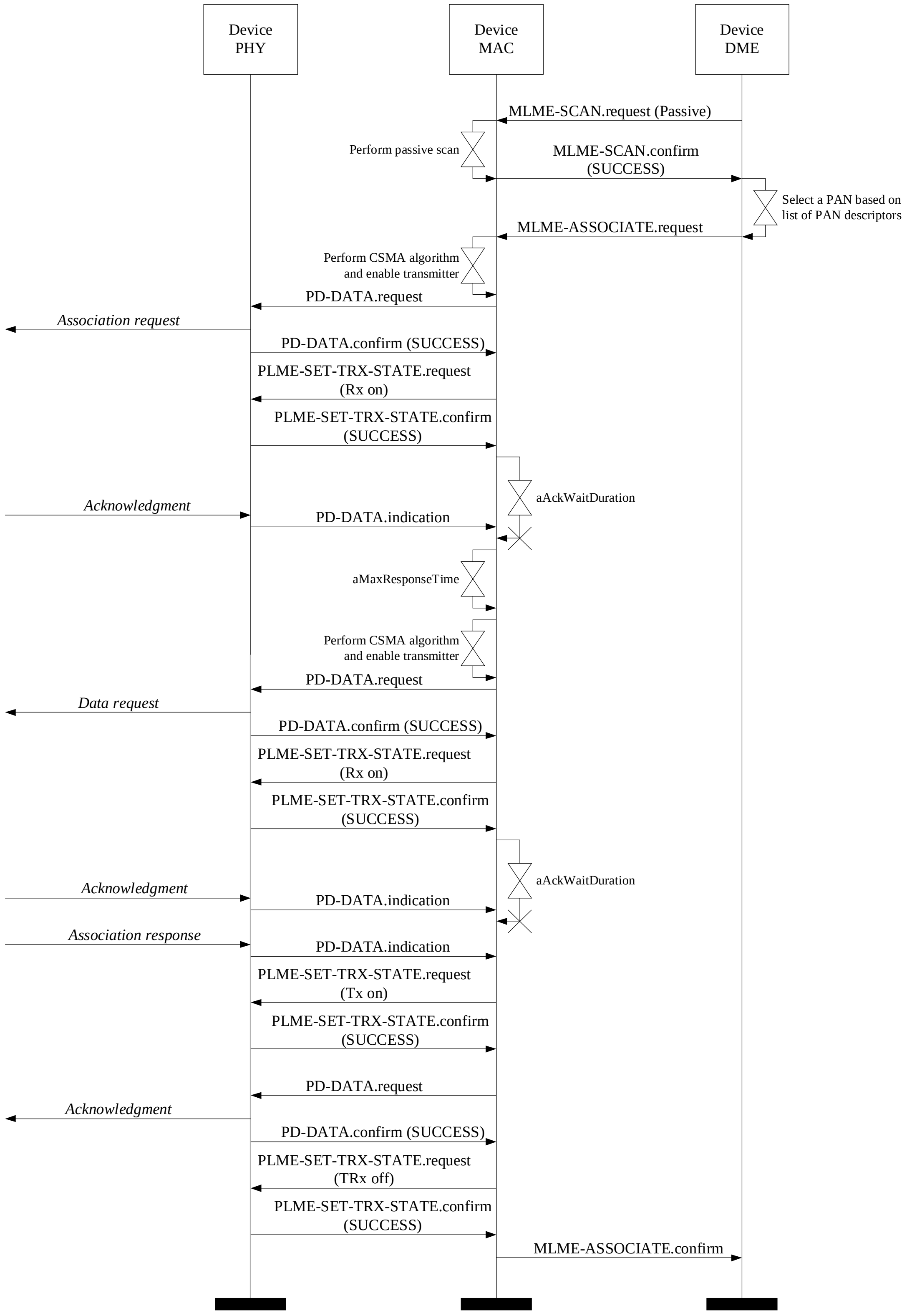

Device PHY
Device MAC
Device DME
MLME-SCAN.request (Passive)
Perform passive scan
MLME-SCAN.confirm (SUCCESS)
Select a PAN based on list of PAN descriptors
MLME-ASSOCIATE.request
Perform CSMA algorithm and enable transmitter
PD-DATA.request
Association request
PD-DATA.confirm (SUCCESS)
PLME-SET-TRX-STATE.request (Rx on)
PLME-SET-TRX-STATE.confirm (SUCCESS)
aAckWaitDuration
Acknowledgment
PD-DATA.indication
aMaxResponseTime
Perform CSMA algorithm and enable transmitter
PD-DATA.request
Data request
PD-DATA.confirm (SUCCESS)
PLME-SET-TRX-STATE.request (Rx on)
PLME-SET-TRX-STATE.confirm (SUCCESS)
aAckWaitDuration
Acknowledgment
PD-DATA.indication
Association response
PD-DATA.indication
PLME-SET-TRX-STATE.request (Tx on)
PLME-SET-TRX-STATE.confirm (SUCCESS)
PD-DATA.request
Acknowledgment
PD-DATA.confirm (SUCCESS)
PLME-SET-TRX-STATE.request (TRx off)
PLME-SET-TRX-STATE.confirm (SUCCESS)
MLME-ASSOCIATE.confirm

Figure 53—Association message sequence chart - device

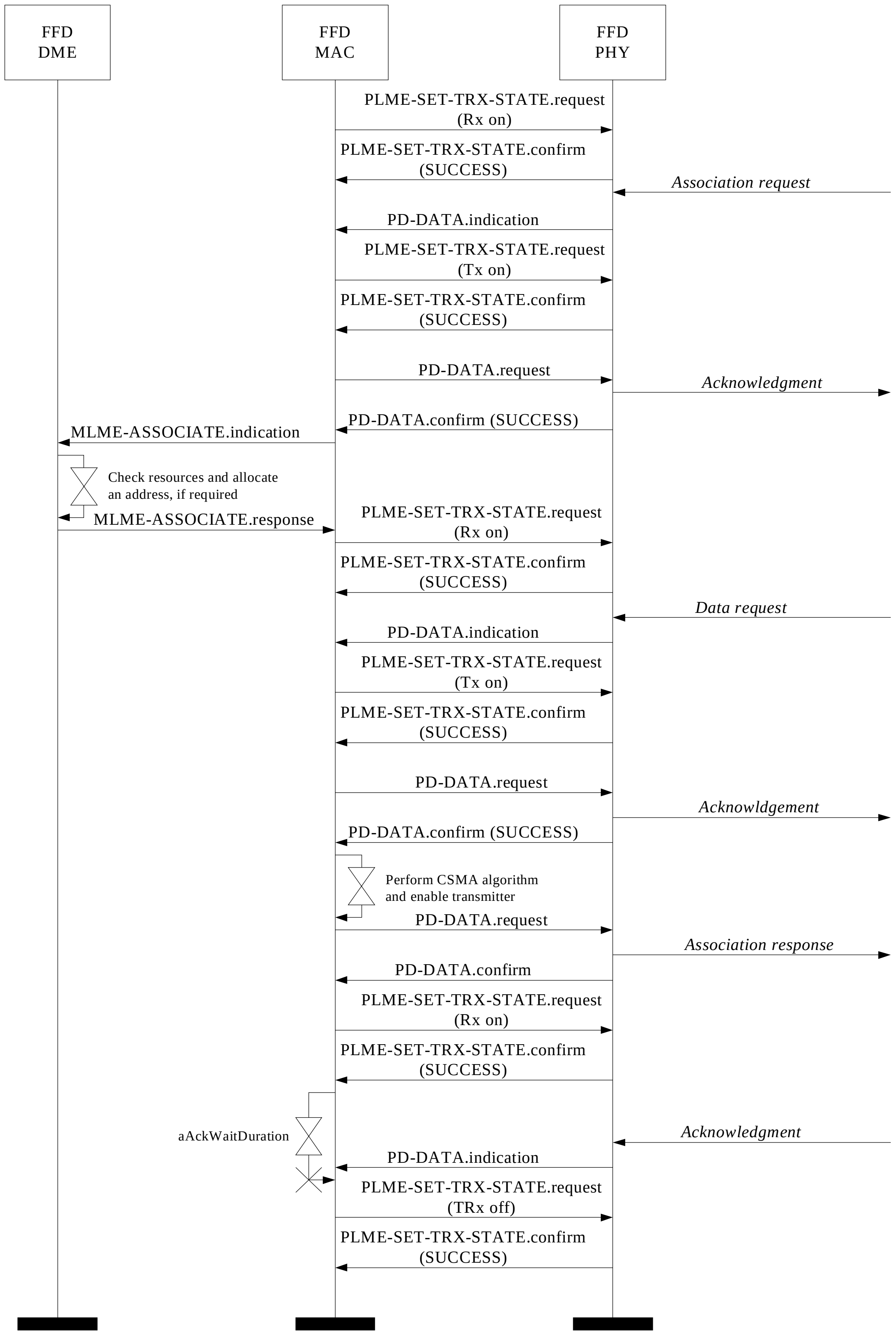

FFD DME
FFD MAC
FFD PHY
PLME-SET-TRX-STATE.request (Rx on)
PLME-SET-TRX-STATE.confirm (SUCCESS)
Association request
PD-DATA.indication
PLME-SET-TRX-STATE.request (Tx on)
PLME-SET-TRX-STATE.confirm (SUCCESS)
PD-DATA.request
Acknowledgment
PD-DATA.confirm (SUCCESS)
MLME-ASSOCIATE.indication
Check resources and allocate an address, if required
MLME-ASSOCIATE.response
PLME-SET-TRX-STATE.request (Rx on)
PLME-SET-TRX-STATE.confirm (SUCCESS)
Data request
PD-DATA.indication
PLME-SET-TRX-STATE.request (Tx on)
PLME-SET-TRX-STATE.confirm (SUCCESS)
PD-DATA.request
Acknowldgement
PD-DATA.confirm (SUCCESS)
Perform CSMA algorithm and enable transmitter
PD-DATA.request
Association response
PD-DATA.confirm
PLME-SET-TRX-STATE.request (Rx on)
PLME-SET-TRX-STATE.confirm (SUCCESS)
aAckWaitDuration
Acknowledgment
PD-DATA.indication
PLME-SET-TRX-STATE.request (TRx off)
PLME-SET-TRX-STATE.confirm (SUCCESS)

Figure 54—Association message sequence chart - coordinator

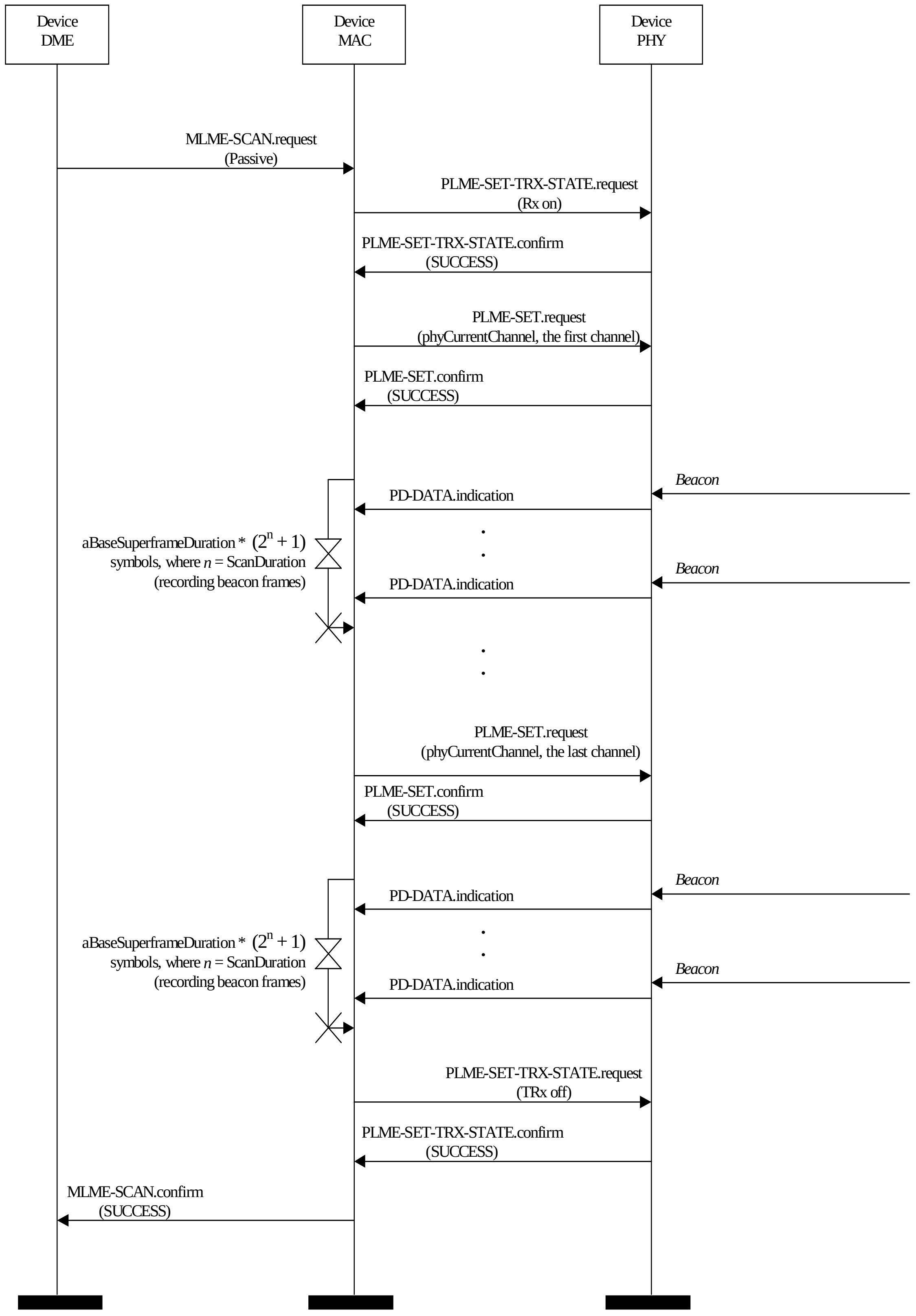

Device DME
Device MAC
Device PHY
MLME-SCAN.request (Passive)
PLME-SET-TRX-STATE.request (Rx on)
PLME-SET-TRX-STATE.confirm (SUCCESS)
PLME-SET.request (phyCurrentChannel, the first channel)
PLME-SET.confirm (SUCCESS)
Beacon
PD-DATA.indication
aBaseSuperframeDuration * $(2^n + 1)$ symbols, where n = ScanDuration (recording beacon frames)
Beacon
PD-DATA.indication
PLME-SET.request (phyCurrentChannel, the last channel)
PLME-SET.confirm (SUCCESS)
Beacon
PD-DATA.indication
aBaseSuperframeDuration * $(2^n + 1)$ symbols, where n = ScanDuration (recording beacon frames)
Beacon
PD-DATA.indication
PLME-SET-TRX-STATE.request (TRx off)
PLME-SET-TRX-STATE.confirm (SUCCESS)
MLME-SCAN.confirm (SUCCESS)

Figure 55—Passive scan message sequence chart

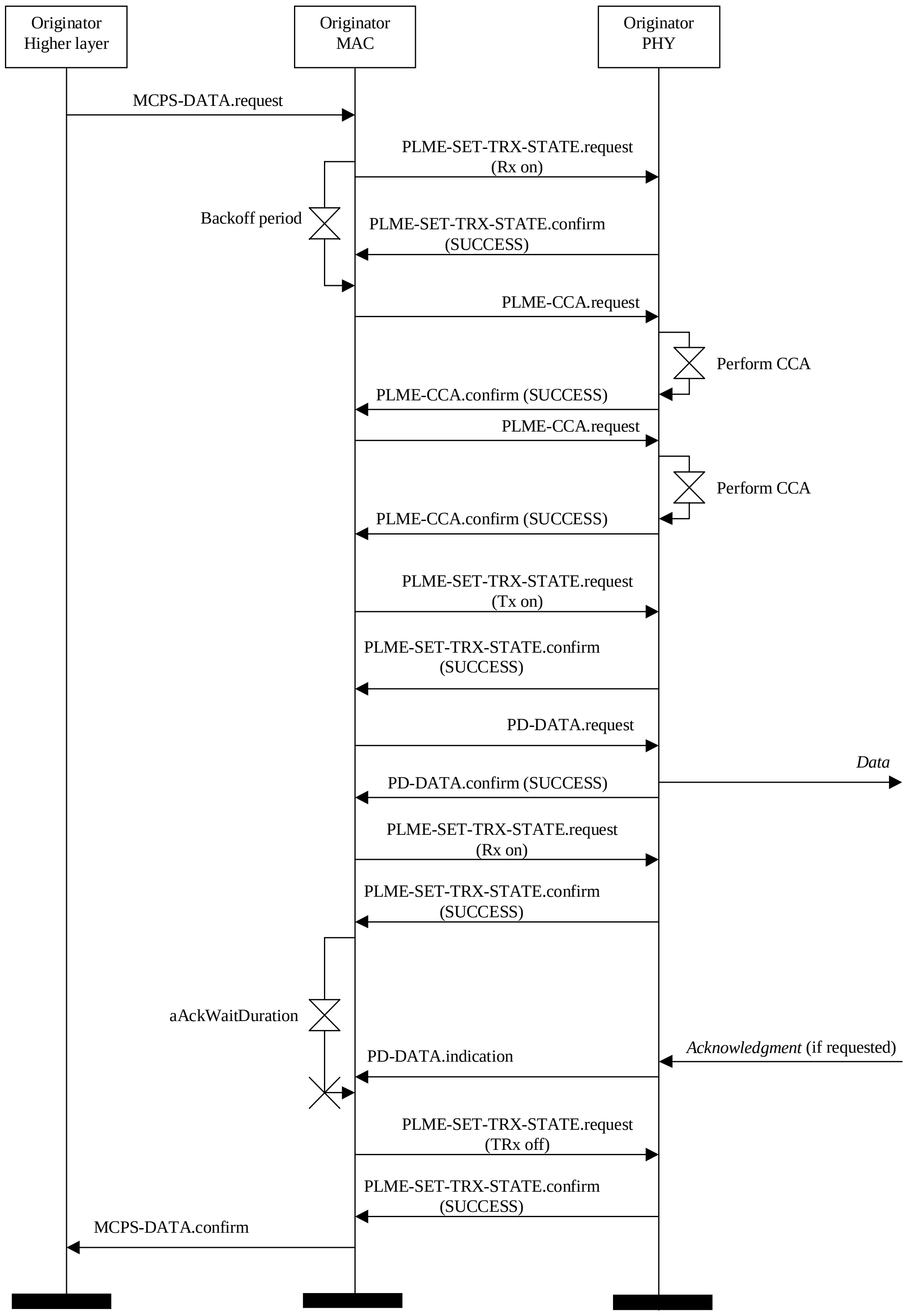

Originator Higher layer
Originator MAC
Originator PHY
MCPS-DATA.request
PLME-SET-TRX-STATE.request (Rx on)
Backoff period
PLME-SET-TRX-STATE.confirm (SUCCESS)
PLME-CCA.request
Perform CCA
PLME-CCA.confirm (SUCCESS)
PLME-CCA.request
Perform CCA
PLME-CCA.confirm (SUCCESS)
PLME-SET-TRX-STATE.request (Tx on)
PLME-SET-TRX-STATE.confirm (SUCCESS)
PD-DATA.request
Data
PD-DATA.confirm (SUCCESS)
PLME-SET-TRX-STATE.request (Rx on)
PLME-SET-TRX-STATE.confirm (SUCCESS)
aAckWaitDuration
Acknowledgment (if requested)
PD-DATA.indication
PLME-SET-TRX-STATE.request (TRx off)
PLME-SET-TRX-STATE.confirm (SUCCESS)
MCPS-DATA.confirm

Figure 56—Data transmission message sequence chart - originator

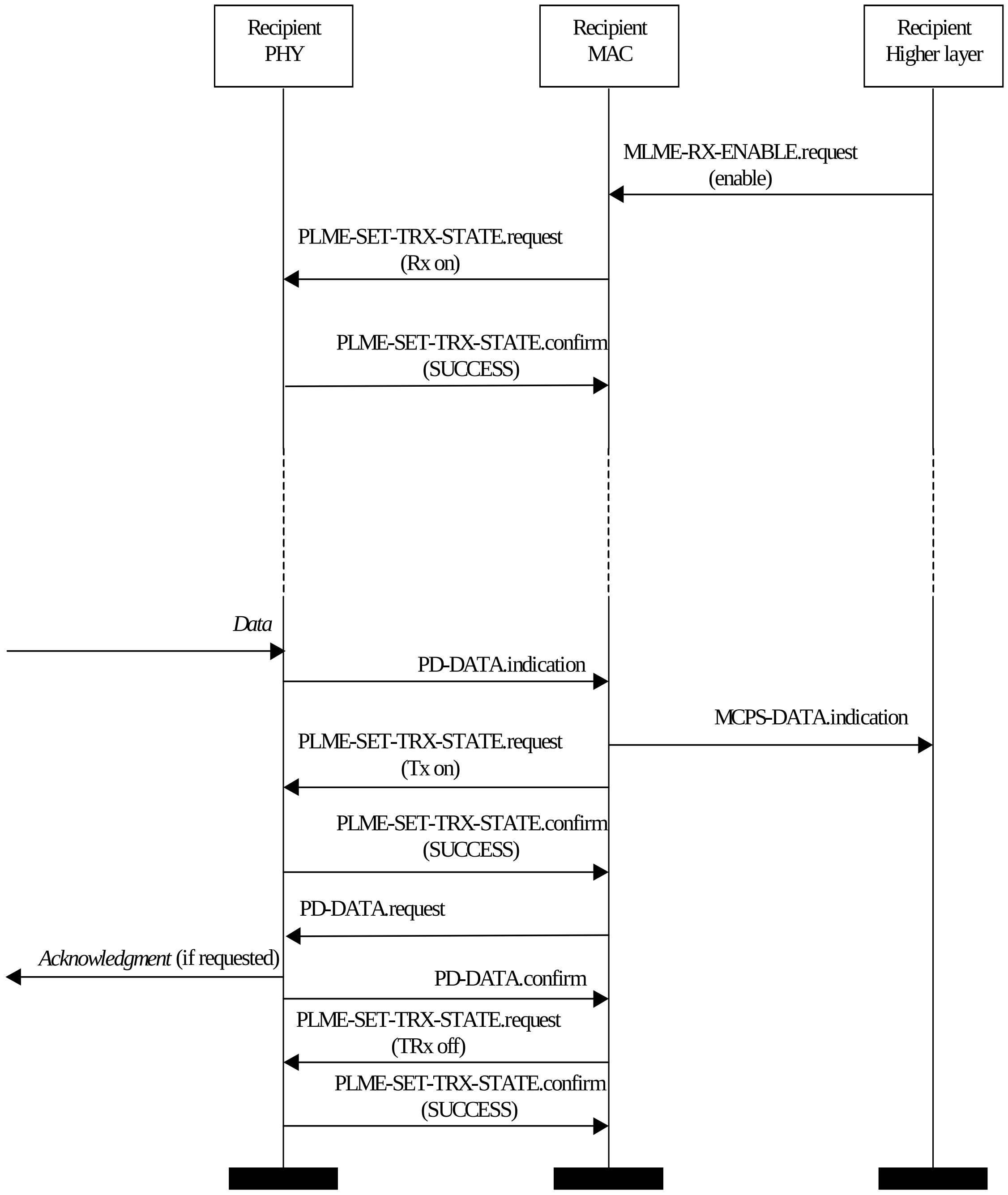

Recipient PHY
Recipient MAC
Recipient Higher layer
MLME-RX-ENABLE.request (enable)
PLME-SET-TRX-STATE.request (Rx on)
PLME-SET-TRX-STATE.confirm (SUCCESS)
Data
PD-DATA.indication
MCPS-DATA.indication
PLME-SET-TRX-STATE.request (Tx on)
PLME-SET-TRX-STATE.confirm (SUCCESS)
PD-DATA.request
Acknowledgment (if requested)
PD-DATA.confirm
PLME-SET-TRX-STATE.request (TRx off)
PLME-SET-TRX-STATE.confirm (SUCCESS)

Figure 57—Data transmission message sequence chart - recipient