ZigBee RF4CE Specification
Version 1.00

ZigBee Document 094945r00ZB	
March 17th, 2009	
Sponsored by: ZigBee Alliance	
Accepted by	This document has been accepted for release by the ZigBee Alliance Board of Directors
Abstract	The ZigBee RF4CE specification describes the protocol infrastructure and services available to applications operating on the ZigBee RF4CE platform
Keywords	RF4CE, Stack, Network, Application, Discovery, Pairing, Security

March 17th, 2009

Notice of use and disclosure

The ZigBee Specification is available to individuals, companies and institutions free of charge for all non-commercial purposes (including university research, technical evaluation, and development of non-commercial software, tools, or documentation). No part of this specification may be used in development of a product for sale without becoming a member of ZigBee Alliance.

Elements of ZigBee Alliance specifications may be subject to third party intellectual property rights, including without limitation, patent, copyright or trademark rights (such a third party may or may not be a member of ZigBee). ZigBee is not responsible and shall not be held responsible in any manner for identifying or failing to identify any or all such third party intellectual property rights.

This document and the information contained herein are provided on an “AS IS” basis and ZigBee DISCLAIMS ALL WARRANTIES EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO (A) ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OF THIRD PARTIES (INCLUDING WITHOUT LIMITATION ANY INTELLECTUAL PROPERTY RIGHTS INCLUDING PATENT, COPYRIGHT OR TRADEMARK RIGHTS) OR (B) ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE OR NONINFRINGEMENT. IN NO EVENT WILL ZIGBEE BE LIABLE FOR ANY LOSS OF PROFITS, LOSS OF BUSINESS, LOSS OF USE OF DATA, INTERRUPTION OF BUSINESS, OR FOR ANY OTHER DIRECT, INDIRECT, SPECIAL OR EXEMPLARY, INCIDENTIAL, PUNITIVE OR CONSEQUENTIAL DAMAGES OF ANY KIND, IN CONTRACT OR IN TORT, IN CONNECTION WITH THIS DOCUMENT OR THE INFORMATION CONTAINED HEREIN, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH LOSS OR DAMAGE. All Company, brand and product names may be trademarks that are the sole property of their respective owners.

The above notice and this paragraph must be included on all copies of this document that are made.

ZigBee Alliance, Inc.

2400 Camino Ramon, Suite 375

San Ramon, CA 94583

Table of Contents

List of Figures

List of Tables

1 Introduction

1.1 Definitions

Controller	A PAN participant that has ZigBee RF4CE functionality.
Destination	The device to which an IEEE 802.15.4 transmission is directed.
Device	An object that has IEEE 802.15.4 functionality.
Node	A device that has ZigBee RF4CE functionality.
Originator	The device from which a ZigBee RF4CE transmission is sent.
Pair	A logical association between two nodes.
PAN	A personal area network as defined in IEEE 802.15.4.
PAN coordinator	A device that can create its own PAN.
PAN participant	A device that can join a PAN.
RC network	Multiple, interoperating, RC PANs.
RC PAN	A PAN consisting exclusively of ZigBee RF4CE nodes.
Recipient	The device to which a ZigBee RF4CE transmission is directed.
Source	The device from which an IEEE 802.15.4 transmission is sent.
Target	A PAN coordinator that has ZigBee RF4CE functionality.

1.2 Conformance levels

The following words, used throughout this document, have specific meanings:

May A key word indicating a course of action permissible within the limits of the standard (*may* equals *is permitted*).

Shall A key word indicating mandatory requirements to be strictly followed in order to conform to the standard; deviations from shall are prohibited (*shall* equals *is required to*).

Should A key word indicating that, among several possibilities, one is recommended as particularly suitable, without mentioning or excluding others; that a certain course of action is preferred but not necessarily required; or, that (in the negative form) a certain course of action is deprecated but not prohibited (*should* equals *is recommended that*).

1.3 Abbreviations

APDU	Application protocol data unit
AV	Audio visual
CE	Consumer electronics
CEC	Consumer electronics control
ED	Energy detection
EUI	Extended unique identifier
DFA	Dynamic frequency agility
DLNA	Digital living network alliance
FFD	Full function device

HDMI	High definition multimedia interface
ID	Identifier
IEEE	Institute of electrical and electronic engineers
IR	Infrared
LQI	Link quality indication
MAC	Medium access control
MCPS	Medium access control common part sub-layer
MFRC	Multi-function remote control
NFR	Network layer frame footer
NHR	Network layer frame header
NIB	Network information base
NLDE	Network layer data entity
NLME	Network layer management entity
NPDU	Network protocol data unit
NSDU	Network service data unit
NWK	Network
ORG	Originator
OSI	Open systems interconnection
PAN	Personal area network
PHY	Physical
POS	Personal operating space
RC	Remote control
REC	Recipient
RF	Radio frequency
RF4CE	Radio frequency for consumer electronics
SAP	Service access point
WPAN	Wireless personal area network

1.4 Conventions

1.4.1 Number formats

In this specification hexadecimal numbers are prefixed with the designation “0x” and binary numbers are prefixed with the designation “0b”. All other numbers are assumed to be decimal.

1.4.2 Transmission order

The frames in this specification are described as a sequence of fields in a specific order. All frame formats 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 MAC in the order from the octet containing the lowest numbered bits to the octet containing the highest numbered bits.

1.4.3 Timing values

All timing values within this specification are specified in terms of MAC symbols. One MAC symbol is equal to 16µs. Where appropriate, absolute time values are presented in both MAC symbols and actual time in parenthesis.

1.4.4 Message sequence charts

During this specification, message sequence charts are used to illustrate the flow of various operations. Instances are labeled with the layer (APL for the application or NWK for the network) followed by the node type (ORG for the originator or REC for the recipient). Primitives are shown in normal style but, for simplicity, without the entity prefix (i.e. NLDE or NLME), e.g. “NLME-PAIR.response” becomes “PAIR.response”. Over the air command frames are labeled in italic text.

1.4.5 Reserved values

Unless otherwise specified, all reserved fields appearing in a frame structure (c.f. Figure 18) shall be set to zero on transmission and ignored upon reception. Reserved values appearing in multi-value fields (c.f. Table 33) shall not be used.

1.5 References

[R1] The Institute of Electrical and Electronics Engineers, Inc., IEEE Std. 802.15.4-2006, IEEE Standard for Information Technology. Telecommunications and Information Exchange between Systems. Local and Metropolitan Area Networks. Specific Requirements. Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (WPANs). New York: IEEE Press. 2006.

[R2] ZigBee RF4CE: Vendor ID List, ZigBee Alliance document 09xxxx.

[R3] ZigBee RF4CE: Device Type List, ZigBee Alliance document 09xxxx.

[R4] ZigBee RF4CE: Profile ID List, ZigBee Alliance document 09xxxx.

2 General description

2.1 Introduction

The ZigBee RF4CE standard defines an RC network that defines a simple, robust and low-cost communication network that allows wireless connectivity in applications in the CE domain. The ZigBee RF4CE standard enhances the IEEE 802.15.4 standard by providing a simple networking layer and standard application profiles that can be used to create a multi-vendor interoperable solution for use within the home.

Some of the characteristics of an RC network are as follows:

- Operation in the 2.4GHz frequency band according to IEEE 802.15.4.
- Frequency agile solution operating over 3 channels.
- Incorporates power saving mechanisms for all device classes.
- Service discovery mechanism with full application confirmation.
- Pairing mechanism with full application confirmation.
- Multiple star topology with inter-PAN communication.
- Various transmission options including broadcast.
- Security key generation mechanism.
- Utilizes the industry standard AES-128 security scheme.
- Specifies a simple RC control profile for CE products.
- Allows standard or vendor specific profiles to be added.

2.2 Network topology

An RC PAN is composed of two types of device: a target node and a controller node. A target node has full PAN coordinator capabilities and can start a network in its own right. Both types of node can join networks started by target nodes by pairing with that target. Multiple RC PANs form an RC network and nodes in the network can communicate between RC PANs.

In order to communicate with a target node, a controller node first switches to the channel and assumes the PAN identifier of the destination RC PAN. It then uses the network address, allocated through the pairing procedure, to identify itself on the RC PAN and thus communicate with the desired target node.

Figure 1 illustrates an example ZigBee RF4CE topology which includes three target nodes: a TV, a DVD and a CD player and each target node creates its own RC PAN. The TV, DVD and CD player also have dedicated RCs which are paired to each appropriate target node. A multi-function RC, capable of controlling all three target nodes itself, is added to the network by successively pairing to the desired target nodes. The DVD is also paired with the TV so that an external channel can be selected on the TV when a DVD is played.

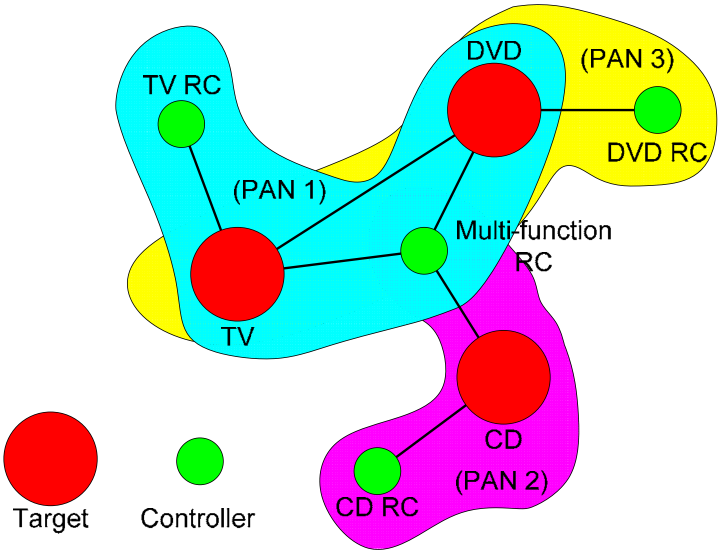


Figure 1 – Example RC network topology

As a consequence, this RC network consists of three separate RC PANs: one managed by the TV (PAN 1), containing the TV RC, the multi-function RC and the DVD; a second managed by the CD player (PAN 2), containing the CD RC and the multi-function RC and a third managed by the DVD (PAN3), containing the DVD RC, multi-function RC and the TV.

2.3 Architecture

The ZigBee RF4CE architecture is defined in terms of a number of blocks or *layers* in order to simplify the standard. Each layer is responsible for one part of the standard and offers services to the next higher layer and utilizes services from the next lower layer. The interfaces between the layers serve to define the logical links that are described in this standard. The layout of the layers is based on the open systems interconnection (OSI) seven-layer model.

Figure 2 illustrates the ZigBee RF4CE stack architecture. The ZigBee RF4CE protocol is designed to be built onto the IEEE 802.15.4 standard MAC and PHY layers and provides networking functionality and standard application profiles which can interface to the end user application. Manufacturer specific extensions to standard profiles can be defined by sending vendor specific data frames within a standard profile. In addition, vendor specific profiles can also be defined.

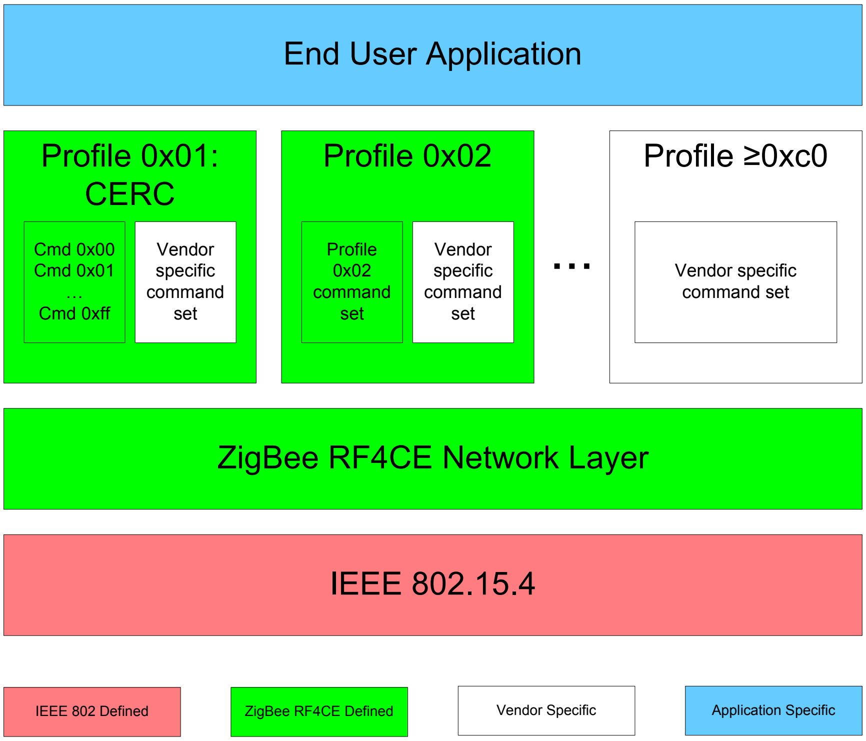


Figure 2 – The ZigBee RF4CE stack architecture

2.4 The ZigBee RF4CE NWK layer

The NWK layer provides two services: the NWK layer data service, interfacing to the NWK layer data entity (NLDE) and the NWK layer management service, interfacing to the NWK layer management entity (NLME). These services are accessed through the NWK layer data entity SAP (NLDE-SAP) and the NWK layer management entity SAP (NLME-SAP).

The NWK layer data service enables the transmission and reception of NWK protocol data units (NPDUs) across the MAC data service. The NWK layer management service permits service discovery, pairing, unpairing, receiver control, node initialization and NIB attribute manipulation.

2.4.1 2.4GHz band frequencies

A ZigBee RF4CE node operates in the 2.4GHz frequency band, as specified by IEEE 802.15.4. However, in an attempt to be robust against other common sources of interference in this band, only a small subset of channels is used - namely channels 15, 20 and 25. A target node can choose to start its network on the best available channel at startup time and so an RC network may operate over one or more of the available three channels.

2.4.2 Frequency agility

All ZigBee RF4CE nodes support frequency agility across all three permitted channels. As described above, a target node selects its own initial channel, based on the channel conditions during startup. During the course of the life of the target node, however, the channel conditions may vary to such an extent that the channel becomes compromised and communication becomes problematic. If this happens, the target node can elect to switch to another channel that may provide a better level of service.

Each node paired to the target records the channel on which communication is expected. However, in the event that the target switches to another channel, the node can attempt transmission on the other channels until communication with the target is reacquired. The node can then record the new channel accordingly for the next time communication is attempted.

2.4.3 Node initialization

A ZigBee RF4CE node initializes itself according to whether it is a target or a controller. Controller nodes simply configure the stack according to this standard and start operating normally. Target nodes configure the stack as controller nodes do but then attempt to start a network.

To do this, the target node first performs an energy detection scan which allows it to obtain information on the usage of each available channel, thus allowing it to select a suitable channel on which to operate. The target node then performs an active scan which allows it to determine the identifiers of any other IEEE 802.15.4 PANs (ZigBee RF4CE or otherwise) operating on the selected channel, thus allowing a unique PAN identifier to be selected for its network. The target node then begins operating normally.

2.4.4 Power saving

Power saving is an important consideration for a ZigBee RF4CE node. As a consequence, this standard defines a power save mechanism that allows both controller nodes as well as target nodes to manage their power consumption by entering a power saving mode. The power saving mechanism is under the control of each application.

A node can manipulate its receiver in a number of ways:

- The receiver can be enabled until further notice (e.g. when a TV comes out of standby).
- The receiver can be enabled for a finite period (e.g. when a TV enters standby mode and wants to engage the power saving mode).
- The receiver can be disabled until further notice (e.g. when an RC enters a dormant state due to none of its buttons being pressed).

When the power saving mode is engaged, the receiver is enabled for an application defined duration (known as the active period) and then disabled. This mechanism is then repeated at an application defined interval (known as the duty cycle). Other nodes can still communicate with a node in power saving mode by targeting the transmission during the active period. The result is a node that periodically enables its receiver for only a short time, hence allowing it to conserve power but still be able to be active on the network.

2.4.5 NWK frames

The ZigBee RF4CE NWK layer defines three frame types: standard data, network command and vendor specific data. Standard data frames transport application data from standard application profiles. Network command frames transport frames which allow the network layer to accomplish certain tasks such as discovery or pairing. Vendor specific data frames transport vendor specific application data.

The general NWK frame format is illustrated in Figure 3.

Octets: 1	4	0/1	0/2	Variable	0/4
Frame control	Frame counter	Profile identifier	Vendor identifier	Frame payload	Message integrity code
Header				Payload	Footer

Figure 3 – General schematic view of a NWK frame

The fields of the general NWK frame are described below:

- Frame control: control information for the frame
- Frame counter: incrementing counter to detect duplicates and prevent replay attacks (security)
- Profile identifier: the application frame format being transported
- Vendor identifier: to allow vendor extensions
- Frame payload: contains the application frame
- Message integrity code: to provide authentication (security)

2.4.6 Transmission options

The ZigBee RF4CE protocol defines a number of transmission options that can be used by an application and combined as appropriate:

- Acknowledged: Originator data is confirmed by the recipient
- Unacknowledged: Originator data is not confirmed by the recipient
- Unicast: Originator data is sent to a specific recipient
- Broadcast: Originator data is sent to all recipients
- Multiple channel: Originator attempts transmission using frequency re-acquisition mechanism
- Single channel: Originator attempts transmission on the expected channel

2.4.7 Discovery

A ZigBee RF4CE node can perform service discovery in an attempt to find other suitable nodes that can be paired to. Discovery can be attempted repeatedly on all three channels for a fixed duration or until a sufficient number of responses have been received. Service discovery is only available to nodes that are not currently in power saving mode.

During discovery, a number of pieces of information are exchanged between both devices. This information is passed to the application which can then make a decision whether it should respond. The information exchanged is as follows:

- Node capabilities: The type of the node (i.e. target or controller), whether the node is mains or battery powered and whether it supports security.
- Vendor information: The ZigBee RF4CE SIG allocated vendor identifier and a freeform vendor string specifying vendor specific identification (e.g. a serial number).
- Application information: A short user defined which describes the application functionality of the node (e.g. "lounge TV"), a device type list specifying which types of device are supported (e.g. a combo device may support both "TV" and "DVD" functionality) and a profile identifier list specifying which application profiles are supported by the node (e.g. the CERC profile or a vendor specific profile).
- Requested device type: The type of device being requested through the discovery (e.g. a multifunction remote control may be searching for "TV" functionality).

2.4.8 Pairing

Once a node has determined, through discovery, that there is another node within communication range offering compatible services, it can set up a pairing link in order to begin communication. Nodes within an RC network may only communicate directly with other nodes on the network if a pairing link exists between the originator and the recipient nodes.

A pairing link can be established on request from the application by exchanging a similar set of information as was exchanged during discovery. The application on the target node can choose whether to accept the pair (e.g. only if it has capacity to store the pairing link) and confirms the pairing request back to the originator node.

If the pairing request was successful, both nodes store a pairing link in their respective pairing tables. This allows an originator to communicate with a target and the target to communicate back to the originator. Each entry in the pairing table contains all the information necessary for the network layer to transmit a frame to the target node. This removes the burden of addressing, etc. from the application layer which can simply supply an index into the pairing table in order to communicate with another device.

Each entry in the pairing table contains the following information:

- Pairing reference
- Source network address

- Destination logical channel
- Destination IEEE address
- Destination PAN identifier
- Destination network address
- Recipient node capabilities
- Recipient frame counter
- Security link key

2.4.9 Security

Like any other wireless network, an RC network could be vulnerable to both passive eavesdropping and unauthorised tampering of the messages being transferred between nodes. To solve this, the ZigBee RF4CE standard provides a cryptographic mechanism to protect the transmissions. This mechanism provides the following security services:

- Data confidentiality: To ensure that the data contained in a ZigBee RF4CE transmission can only be disclosed to the intended recipient.
- Data authenticity: To ensure that the intended recipient of a ZigBee RF4CE transmission knows that the data was sent from a trusted source and not modified during transmission.
- Replay protection: To ensure that a secure transmission cannot simply be repeated by an attacking device if overheard.

A 128-bit cryptographic key is generated by the recipient of the pairing request and exchanged with the originator. The key is stored in the respective pairing tables.

2.5 The ZigBee RF4CE application layer

The application layer of a ZigBee RF4CE node is composed of a profile component and an application specific component. The profile component can be thought of as a common language that nodes implementing the profile exchange to accomplish certain tasks, e.g. switching the channel on a TV. The application component is provided by the end manufacturer in order to add specific functionality to the commands request through the profile.

The ZigBee RF4CE standard defines standard profiles (e.g. CERC) but also permits vendors to either extend standard profiles or to define completely proprietary profiles.

The Consumer Electronics Remote Control (CERC) profile is once such standard profile defined by the ZigBee RF4CE SIG. This profile defines commands and procedures to enable CE devices (e.g. a TV, DVD or CD player) to be controlled by remote control devices.

2.6 Concept of primitives

This clause provides a brief overview of the concept of service primitives. Please refer to IEEE Std 802.2 1998 edition for more detailed information.

The services of a layer are the capabilities it offers to the user in the next higher layer or sub-layer by building its functions on the services of the next lower layer. This concept is illustrated in Figure 3, where the service hierarchy and the relationship of the two correspondent N-users and their associated N-layer (or sub-layer) peer protocol entities.

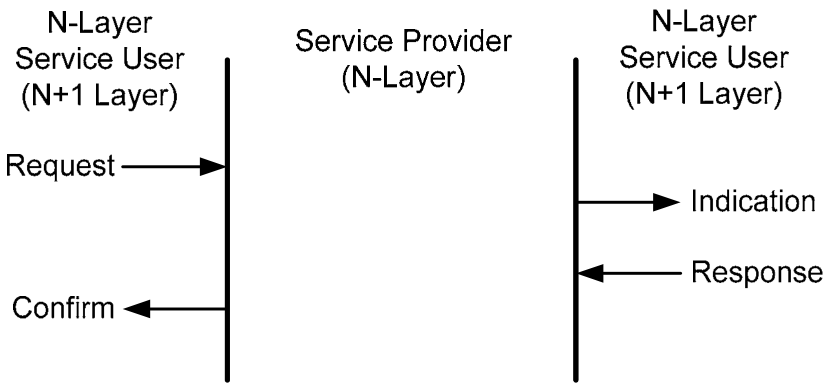


Figure 4 – Service primitives

The services are specified by describing the information flow between the N-user and the N-layer. This information flow is modeled by discrete, instantaneous events, which characterize the provision of a service. Each event consists of passing a service primitive from one layer to the other through a layer service access point associated with an N-user. Service primitives convey the required information by providing a particular service. These service primitives are an abstraction since they only specify the provided service rather than the means by which it is provided. This definition is independent of any other interface implementation.

Services are specified by describing the service primitives and parameters that characterize it. A service may have one or more related primitives that constitute the activity that is related to that particular service. Each service primitive may have zero or more parameters that convey the information required to provide the service.

A primitive can be one of four generic types:

- **Request:** The request primitive is passed from the N-user to the N-layer to request that a service is initiated.
- **Indication:** The indication primitive is passed from the N-layer to the N-user to indicate an internal N-layer event that is significant to the N-user. This event may be logically related to a remote service request, or it may be caused by an N-layer internal event.
- **Response:** The response primitive is passed from the N-user to the N-layer to complete a procedure previously invoked by an indication primitive.
- **Confirm:** The confirm primitive is passed from the N-layer to the N-user to convey the results of one or more associated previous service requests.

3 Network layer specification

3.1 NWK layer service specification

3.1.1 NWK layer data service

The NLDE-SAP supports the transport of application protocol data units (APDUs) between peer application entities. Table 1 lists the primitives supported by the NLDE-SAP. These primitives are discussed in the sub-clauses referenced in the table.

Table 1 – NLDE-SAP primitives

Name	Request	Indication	Confirm
NLDE-DATA	3.1.1.1	3.1.1.2	3.1.1.3

3.1.1.1 NLDE-DATA.request

The NLDE-DATA.request primitive requests the transfer of a data APDU (i.e. NSDU) from a local application entity to a peer application entity.

3.1.1.1.1 Semantics of the service primitive

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

```
NLDE-DATA.request       (
                        PairingRef,
                        ProfileId,
                        VendorId,
                        nsduLength,
                        nsdu,
                        TxOptions
                        )
```

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

Table 2 – NLDE-DATA.request parameters

Name	Type	Valid range	Description
PairingRef	Integer	0x00 – 0xfe	Reference into the pairing table which contains the information required to transmit the NSDU. This parameter is ignored if the TxOptions parameter specifies a broadcast transmission.
ProfileId	Integer	0x00 – 0xfe	The identifier of the profile indicating the format of the transmitted data.

Name	Type	Valid range	Description
VendorId	Vendor identifier	0x0000 or a valid vendor identifier	If the TxOptions parameter specifies that the data is vendor specific, this parameter specifies the vendor identifier. If this parameter is equal to 0x0000, the vendor identifier shall be set to *nwkcVendorIdentifier*. If the TxOptions parameter specifies that the data is not vendor specific this parameter is ignored.
nsduLength	Integer	0 – (*aMaxMACSafe-PayloadSize* – *nwkcMinNWKHeader-Overhead*) (See [R1])	The number of octets contained in the NSDU to be transmitted by the NLDE.
nsdu	Set of octets	-	The set of octets forming the NSDU to be transmitted by the NLDE.
TxOptions	Bitmap	7-bit field	Transmission options for this NSDU. For b_0 (transmission mode): 1 = broadcast transmission 0 = unicast transmission For b_1 (destination addressing mode): 1 = use destination IEEE address 0 = use destination network address For b_2 (acknowledgement mode): 1 = acknowledged transmission 0 = unacknowledged transmission For b_3 (security mode): 1 = transmit with security 0 = transmit without security For b_4 (channel agility mode): 1 = use single channel operation 0 = use multiple channel operation For b_5 (channel normalization mode): 1 = specify channel designator 0 = do not specify channel designator For b_6 (payload mode): 1 = data is vendor specific 0 = data is not vendor specific

3.1.1.1.2 When generated

The NLDE-DATA.request primitive is generated by a local application entity when a data APDU (i.e. NSDU) is to be transferred to a peer application entity.

3.1.1.1.3 Effect on receipt

On receipt of the NLDE-DATA.request primitive, the NLDE begins the transmission of the supplied NSDU.

If *nwkFrameCounter* is equal to its maximum value (0xffffffff), the NLDE will issue the NLDE-DATA.confirm primitive with a status of FRAME_COUNTER_EXPIRED and perform no further processing.

The NLDE builds an NPDU to transmit from the supplied arguments and the supplied NSDU will be placed in the NWK frame payload.

If the transmission mode flag of the TxOptions parameter indicates that a broadcast transmission is required, the NLDE shall transmit the NPDU with a destination PAN identifier and destination address of 0xffff. If the originator is a target, the source address is set to the network address of the originator. Otherwise, the source address is set to the IEEE address of the originator. In addition, the transmission will always be sent unacknowledged and with security disabled. In this case, the destination addressing mode and acknowledgement mode flags are ignored.

If the transmission mode flag of the TxOptions parameter indicates that a unicast transmission is required, the NLDE uses the PairingRef parameter to determine the recipient information with which to transmit the NPDU. If no entry exists in the pairing table with this reference, the NLDE will issue the NLDE-DATA.confirm primitive with a status of NO_PAIRING and the NPDU will not be transmitted.

If an entry exists in the pairing table with this reference, the NLDE uses the corresponding channel and addressing information contained in the pairing table entry to instruct the MAC sub-layer to transmit the frame. If the destination addressing mode flag of the TxOptions parameter indicates that a destination IEEE address is required or the recipient is a controller, the NLDE uses the destination IEEE address field of the pairing table entry. Otherwise, the NLDE uses the destination network address field in the pairing table entry. If the acknowledgement mode flag of the TxOptions parameter indicates that an acknowledgement is required, the NLDE instructs the MAC sub-layer to transmit the frame with an acknowledgement request. Otherwise, the NLDE instructs the MAC sub-layer to transmit the frame without an acknowledgement request.

If the security mode flag of the TxOptions parameter indicates that security is required for this transmission, the NLDE shall secure the frame using the procedures described in 3.5.11.3 before transmitting the frame. Otherwise, the NLDE shall transmit the frame without security.

If the channel agility mode flag of the TxOptions parameter indicates that single channel operation is required, the NLDE attempts to transmit the frame only on the current channel (as specified in the corresponding entry of the pairing table). Otherwise, the NLDE attempts to transmit the frame on multiple channels, according to the frequency agility mechanism.

If the channel normalization mode flag of the TxOptions parameter indicates that a channel designator be specified, the NLDE sets the channel designator sub-field of the frame control field according to the value of *nwkBaseChannel*. Otherwise, the NLDE sets the channel designator sub-field of the frame control field to zero.

If the payload mode flag of the TxOptions parameter indicates that the data is vendor specific, the NLDE sets the frame type sub-field of the frame control field to indicate a vendor specific data frame. Otherwise, the NLDE sets the frame type sub-field of the frame control field to indicate a standard data frame.

The NWK layer transmits the NPDU according to the transmission procedures defined in sub-clause 3.5.8 and via the MCPS-DATA.request primitive.

Once the NLDE receives the MCPS-DATA.confirm from the MAC sub-layer, it will issue the NLDE-DATA.confirm primitive along with the original pairing reference and the status value from the MCPS-DATA.confirm.

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

3.1.1.2 NLDE-DATA.indication

3.1.1.2.1 Semantics of the service primitive

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

```
NLDE-DATA.indication        (
                            PairingRef,
                            ProfileId,
                            VendorId,
                            nsduLength,
                            nsdu,
                            RxLinkQuality,
                            RxFlags
                            )
```

Table 3 specifies the parameters for the NLDE-DATA.indication primitive.

Table 3 – NLDE-DATA.indication parameters

Name	Type	Valid range	Description
PairingRef	Integer	0x00 – 0xfe	Reference into the pairing table which matched the information contained in the received NPDU.
ProfileId	Integer	0x00 – 0xff	The identifier of the profile indicating the format of the received data.
VendorId	Vendor identifier	A valid vendor identifier	If the RxFlags parameter specifies that the data is vendor specific, this parameter specifies the vendor identifier. If the RxFlags parameter specifies that the data is not vendor specific this parameter is ignored.
nsduLength	Integer	0 – (*aMaxMACSafe-PayloadSize – nwkcMinNWKHeader-Overhead*)	The number of octets contained in the NSDU received by the NLDE.
nsdu	Set of octets	-	The set of octets forming the NSDU received by the NLDE.
RxLinkQuality	Integer	0x00 – 0xff	LQI value measured during the reception of the NPDU. Lower values represent lower LQI.
RxFlags	Bitmap	3-bit field	Reception indication flags for this NSDU. For b_0 (reception mode): 1 = received as broadcast 0 = received as unicast For b_1 (security mode): 1 = received with security 0 = received without security For b_2 (payload mode): 1 = data is vendor specific 0 = data is not vendor specific

3.1.1.2.2 When generated

The NLDE-DATA.indication primitive is generated by the NLDE and issued to the application on receipt of a data frame at the local NWK layer entity.

3.1.1.2.3 Effect on receipt

On receipt of the NLDE-DATA.indication primitive, the application is notified of the arrival of data at the device. The RxLinkQuality parameter contains the LQI value reported via the MAC sub-layer of the received frame.

The reception mode flag of the RxFlags parameter indicates whether the frame was received as a broadcast or as a unicast.

The security mode flag of the RxFlags parameter indicates whether the frame was secured.

If the payload mode flag of the RxFlags parameter indicates that the frame is vendor specific, the VendorId parameter will contain the vendor identifier to use to decode the payload. Otherwise, the VendorId parameter should be ignored.

3.1.1.3 NLDE-DATA.confirm

3.1.1.3.1 Semantics of the service primitive

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

```
NLDE-DATA.confirm          (
                           Status,
                           PairingRef
                           )
```

Table 4 specifies the parameters for the NLDE-DATA.confirm primitive.

Table 4 – NLDE-DATA.confirm parameters

Name	Type	Valid range	Description
Status	Enumeration	SUCCESS, INVALID_PARAMETER, NO_PAIRING, FRAME_COUNTER_EXPIRED or any status value from the MCPS-DATA.confirm.	The status of the NSDU transmission.
PairingRef	Integer	0x00 – 0xfe	The pairing table reference for the NSDU being confirmed.

3.1.1.3.2 When generated

The NLDE-DATA.confirm primitive is generated by the NWK layer entity in response to an NLDE-DATA.request primitive. The NLDE-DATA.confirm primitive returns a status of either SUCCESS, indicating that the request to transmit was successful, or the appropriate error code. The status values are fully described in 3.1.1.1.3.

3.1.1.3.3 Effect on receipt

On receipt of the NLDE-DATA.confirm primitive, the application of the originator node is notified of the result of its request to transmit. If the transmission attempt was successful, the status parameter will be set to SUCCESS. Otherwise, the status parameter will indicate the error.

3.1.1.3.4 Data service message sequence chart

Figure 5 illustrates the sequence of messages necessary for a successful data transfer between two nodes.

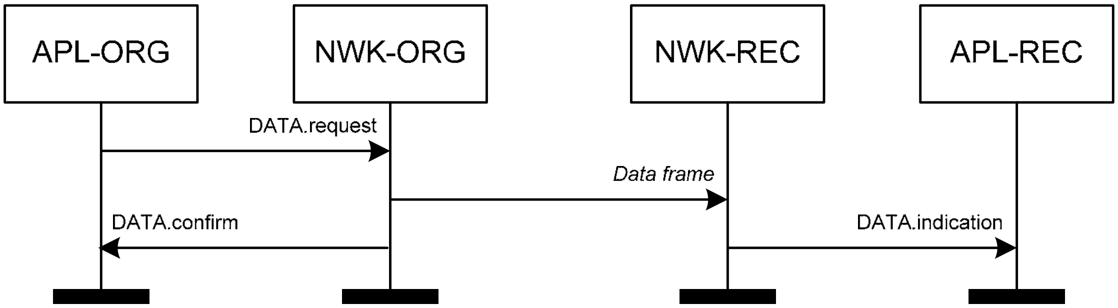


Figure 5 – Message sequence chart for the data service

3.1.2 NWK layer management service

The NLME-SAP allows the transport of management commands between the application and the NLME. Table 5 summarizes the primitives supported by the NLME through the NLME-SAP interface. The primitives are discussed in the sub-clauses referenced in the table.

Table 5 – NLME-SAP primitives

Name	Request	Indication	Response	Confirm
NLME-AUTO-DISCOVERY	3.1.2.1			3.1.2.2
NLME-COMM-STATUS		3.1.2.3		
NLME-DISCOVERY	3.1.2.4	3.1.2.5	3.1.2.6	3.1.2.7
NLME-GET	3.1.2.8			3.1.2.9
NLME-PAIR	3.1.2.10	3.1.2.11	3.1.2.12	3.1.2.13
NLME-RESET	3.1.2.14			3.1.2.15
NLME-RX-ENABLE	3.1.2.16			3.1.2.17
NLME-SET	3.1.2.18			3.1.2.19
NLME-START	3.1.2.20			3.1.2.21
NLME-UNPAIR	3.1.2.22	3.1.2.23	3.1.2.24	3.1.2.25
NLME-UPDATE-KEY	3.1.2.26			3.1.2.27

3.1.2.1 NLME-AUTO-DISCOVERY.request

The NLME-AUTO-DISCOVERY.request primitive allows the application to request the NLME automatically handles the receipt of discovery request command frames. Note that during this auto discovery response mode, the NLME does not inform the application of the arrival of discovery request command frames via the NLME-DISCOVERY.indication primitive.

3.1.2.1.1 Semantics of the service primitive

The semantics of the NLME-AUTO-DISCOVERY.request primitive are as follows:

```
NLME-AUTO-DISCOVERY.request          (
                                     RecAppCapabilities,
                                     RecDevTypeList,
                                     RecProfileIdList,
                                     AutoDiscDuration
                                     )
```

Table 6 specifies the parameters for the NLME-AUTO-DISCOVERY.request primitive.

Table 6 – NLME-AUTO-DISCOVERY.request parameters

Name	Type	Valid range	Description
RecAppCapabilities	Bitmap	See Figure 18	The application capabilities of the node issuing this primitive.
RecDevTypeList	List of integers	Each integer: 0x00 – 0xfe	The list of device types supported by the node issuing this primitive. The set of supported device types is specified in [R3] .
RecProfileIdList	List of integers	Each integer: 0x00 – 0xff	The list of profile identifiers supported by the node issuing this primitive. The set of supported profile identifiers is specified in [R4]
AutoDiscDuration	Integer	0x000000 – 0xffffff	The maximum number of MAC symbols NLME will be in auto discovery response mode.

3.1.2.1.2 When generated

The NLME-AUTO-DISCOVERY.request primitive is generated by the local application entity and issued to the NLME to request automatic discovery response mode. In this mode, the NLME determines whether to respond to received discovery request command frames according to the information contained in this primitive.

3.1.2.1.3 Effect on receipt

On receipt of this primitive, the NLME enters auto discovery response mode for at most the time specified in the AutoDiscDuration parameter. During this mode, if the node receives a command frame that is not a discovery request command frame it will be discarded.

On receipt of a discovery request command frame, the NLME checks that the requested device type field matches one of the device types listed in the RecDevTypeList parameter and that at least one of the profile identifiers listed in the profile identifier list field matches one of the profile identifiers listed in the RecProfileIdList parameter. If a match is found, the NLME waits for the reception of a second identical discovery request command frame to be received from the same node and again checks for a match.

If the second discovery request command frame matches, the NLME generates a discovery response command frame (see 3.3.2). The NLME transmits the frame on its current channel by issuing the MCPS-DATA.request primitive to the MAC sub-layer.

On receipt of an MCPS-DATA.confirm primitive from the MAC sub-layer, the NLME issues the NLME-AUTO-DISCOVERY.confirm primitive with the status value contained in the primitive.

If a second discovery request command frame is received from a different node, the NLME issues the NLME-AUTO-DISCOVERY.confirm primitive with the status of DISCOVERY_ERROR.

If a discovery request command frame is received that does not match the information contained in this primitive, the NLME discards the frame and remains in auto discovery mode.

If no matching discovery request command frames are received after the time specified in the AutoDiscDuration parameter, the NLME issues the NLME-AUTO-DISCOVERY.confirm primitive with a status of DISCOVERY_TIMEOUT.

3.1.2.2 NLME-AUTO-DISCOVERY.confirm

The NLME-AUTO-DISCOVERY.confirm primitive allows the NLME to notify the application of the status of its request to enter auto discovery response mode.

3.1.2.2.1 Semantics of the service primitive

The semantics of the NLME-AUTO-DISCOVERY.confirm primitive are as follows:

```
NLME-AUTO-DISCOVERY.confirm        (
                                   Status,
                                   SrcIEEEAddr
                                   )
```

Table 7 specifies the parameters for the NLME-AUTO-DISCOVERY.confirm primitive.

Table 7 – NLME-AUTO-DISCOVERY.confirm parameters

Name	Type	Valid range	Description
Status	Enumeration	SUCCESS, DISCOVERY_ERROR, DISCOVERY_TIMEOUT or anything returned from the MCPS-DATA.confirm primitive	The status of the auto discovery response mode.
SrcIEEEAddr	IEEE address	A valid IEEE address	The IEEE address to which the discovery response was sent.

3.1.2.2.2 When generated

The NLME-AUTO-DISCOVERY.confirm primitive is generated by the NLME and issued to the application in response to an NLME-AUTO-DISCOVERY.request primitive. This primitive returns a status of SUCCESS if a discovery response command frame was successfully transmitted in response to the reception of a discovery request command frame. Otherwise, the status parameter indicates an error code.

3.1.2.2.3 Effect on receipt

On receipt of the NLME-AUTO-DISCOVERY.confirm primitive, the application is notified of the status of the auto discovery response mode. If the NLME successfully transmitted a discovery response command frame in response to the reception of a discovery request command frame, the status parameter will indicate a successful auto discovery response and the SrcIEEEAddr parameter will contain the IEEE address of the node to which it was sent. Otherwise, the status parameter indicates an error code and the SrcIEEEAddr parameter should be ignored.

3.1.2.2.4 Auto discovery response message sequence chart

Figure 6 illustrates the sequence of messages necessary for a successful automatic discovery response mechanism.

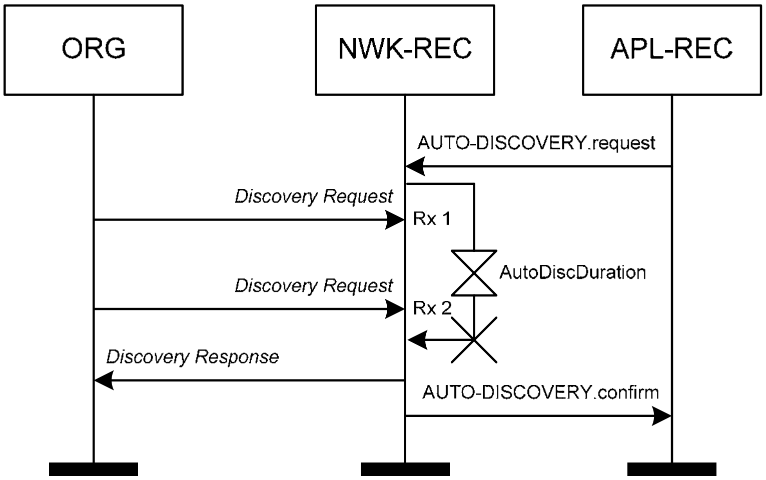


Figure 6 – Message sequence chart for auto discovery response

3.1.2.3 NLME-COMM-STATUS.indication

The NLME-COMM-STATUS.indication primitive allows the NLME to notify the application of a communication status.

3.1.2.3.1 Semantics of the service primitive

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

```
NLME-COMM-STATUS.indication     (
                                PairingRef,
                                DstPANId,
                                DstAddrMode,
                                DstAddr,
                                Status
                                )
```

Table 6 specifies the parameters for the NLME-COMM-STATUS.indication primitive.

Table 8 – NLME-COMM-STATUS.indication parameters

Name	Type	Valid range	Description
PairingRef	Integer	0x00 – 0xff	Reference into the pairing table indicating the recipient node. A value of 0xff indicates that a discovery response command frame was sent.
DstPANId	PAN identifier	A valid PAN identifier	The PAN identifier of the destination device.

Name	Type	Valid range	Description
DstAddrMode	Bitmap	1-bit field	The addressing mode used in the DstAddr parameter. For b_0 (destination addressing mode): 1 = DstAddr is 64-bit IEEE address 0 = DstAddr is 16-bit network address
DstAddr	Device address	According to the DstAddrMode parameter	The address of the destination device.
Status	Enumeration	SUCCESS, SECURITY_TIMEOUT, SECURITY_FAILURE or anything from the MCPS-DATA.confirm primitive	The status of the transmission.

3.1.2.3.2 When generated

The NLME-COMM-STATUS.indication primitive is generated by the NLME and issued to the application following a transmission instigated through a response primitive.

The NLME-COMM-STATUS.indication primitive is generated by the NLME following either the NLME-DISCOVERY.response primitive or the NLME-PAIR.response primitive. If this primitive is generated following an NLME-DISCOVERY.response primitive, the PairingRef parameter should be set to 0xff. Conversely, if this primitive is generated following an NLME-PAIR.response primitive, the PairingRef parameter should be set to the value in the ProvPairingRef parameter of the NLME-PAIR.response primitive.

This primitive returns a status of SUCCESS, indicating that the request to transmit was successful, or an appropriate error code from the MCPS-DATA.confirm primitive, following the transmission.

3.1.2.3.3 Effect on receipt

On receipt of the NLME-COMM-STATUS.indication primitive, the application is notified of the status of a transmission following a .response primitive.

3.1.2.4 NLME-DISCOVERY.request

The NLME-DISCOVERY.request primitive allows the application to request the NLME discover other devices of interest operating in the POS of the device.

3.1.2.4.1 Semantics of the service primitive

The semantics of the NLME-DISCOVERY.request primitive are as follows:

```
NLME-DISCOVERY.request        (
                              DstPANId,
                              DstNwkAddr,
                              OrgAppCapabilities,
                              OrgDevTypeList,
                              OrgProfileIdList,
                              SearchDevType,
                              DiscProfileIdListSize,
                              DiscProfileIdList,
                              DiscDuration
                              )
```

Table 9 specifies the parameters for the NLME-DISCOVERY.request primitive.

Table 9 – NLME-DISCOVERY.request parameters

Name	Type	Valid range	Description
DstPANId	PAN identifier	A valid PAN identifier	The PAN identifier of the destination device for the discovery. This value can be set to 0xffff to indicate a wildcard.
DstNwkAddr	Network address	A valid network address	The address of the destination device for the discovery. This value can be set to 0xffff to indicate a wildcard.
OrgAppCapabilities	Bitmap	See Figure 18	The application capabilities of the node issuing this primitive.
OrgDevTypeList	List of integers	Each integer: 0x00 – 0xfe	The list of device types supported by the node issuing this primitive. The set of supported device types is specified in [R3] .
OrgProfileIdList	List of integers	Each integer: 0x00 – 0xff	The list of profile identifiers disclosed as supported by the node issuing this primitive.
SearchDevType	Integer	0x00 – 0xff	The device type to discover. This value can be set to 0xff to indicate a wildcard.
DiscProfileIdListSize	Integer	0x00 – 0xff	The number of profile identifiers contained in the DiscProfileIdList parameter.
DiscProfileIdList	List of integers	Each integer: 0x00 – 0xff	The list of profile identifiers against which profile identifiers contained in received discovery response command frames will be matched for acceptance. The set of supported profile identifiers is specified in [R4]
DiscDuration	Integer	0x000000 – 0xffffff	The maximum number of MAC symbols to wait for discovery responses to be sent back from potential target nodes on each channel.

3.1.2.4.2 When generated

The NLME-DISCOVERY.request primitive is generated by the local application entity and issued to the NLME to request a discovery operation.

3.1.2.4.3 Effect on receipt

On receipt of the NLME-DISCOVERY.request primitive, the NLME generates a discovery request command frame (see 3.3.1). The NLME transmits the frame once on each channel, switching channels accordingly before requesting the MAC sub-layer transmit the frame by issuing the MCPS-DATA.request primitive.

If the MAC sub-layer fails to transmit the frame (i.e. a status of anything but SUCCESS is returned from the MCPS-DATA.confirm primitive), the NLME issues the NLME-DISCOVERY.confirm primitive with the status value returned from the MAC sub-layer.

If the MAC sub-layer successfully transmits the frame, the NLME waits for at most the time represented by the DiscDuration parameter for any discovery response commands to arrive. On receipt of each unique discovery response command frame (see 3.3.2) containing a device type that matches the SearchDevType parameter and a profile identifier that matches at least one in the DiscProfileIdList parameter, the NLME creates a new node descriptor (see Table 13) with the information contained in the discovery response command. At the end of the time represented by the DiscDuration parameter, the NLME switches to the next channel and repeats the procedure. The transmission of a discovery request command frame on all available channels is called a discovery trial. A discovery trial is performed at most *nwkMaxDiscoveryRepetitions* times, repeated at intervals of *nwkDiscoveryRepetitionInterval*.

If the number of node descriptors stored at the end of any single discovery trial is equal to *nwkMaxReportedNodeDescriptors*, the NLME issues the NLME-DISCOVERY.confirm primitive with the NodeDescList parameter containing the list of node descriptors discovered for the nodes from which discovery response commands were received and the Status parameter set to SUCCESS.

If the number of node descriptors stored at the end of any single discovery trial exceeds *nwkMaxReportedNodeDescriptors*, the NLME issues the NLME-DISCOVERY.confirm primitive with the Status parameter set to DISCOVERY_ERROR.

If, at any time during the discovery process, the number of node descriptors stored is equal to *nwkcMaxNodeDescListSize*, the NLME issues the NLME-DISCOVERY.confirm primitive with the NodeDescList parameter containing the list of node descriptors discovered for the devices from which discovery response commands were received and the Status parameter set to SUCCESS.

If, at the end of *nwkMaxDiscoveryRepetitions* discovery trials, no node descriptors are stored, the NLME issues the NLME-DISCOVERY.confirm primitive with a status of DISCOVERY_TIMEOUT.

If *nwkMaxDiscoveryRepetitions* discovery trials have been made and the procedure has not yet been terminated as described above, the NLME issues the NLME-DISCOVERY.confirm primitive with the NodeDescList parameter containing the list of node descriptors discovered for the devices from which discovery response commands were received and the Status parameter set to SUCCESS.

3.1.2.5 NLME-DISCOVERY.indication

The NLME-DISCOVERY.indication primitive allows the NLME to notify the application that a discovery request command has been received.

3.1.2.5.1 Semantics of the service primitive

The semantics of the NLME-DISCOVERY.indication primitive are as follows:

```
NLME-DISCOVERY.indication    (
                             Status,
                             SrcIEEEAddr,
                             OrgNodeCapabilities
                             OrgVendorId,
                             OrgVendorString,
                             OrgAppCapabilities,
                             OrgUserString,
                             OrgDevTypeList,
                             OrgProfileIdList,
                             SearchDevType,
                             RxLinkQuality
                             )
```

Table 10 specifies the parameters for the NLME-DISCOVERY.indication primitive.

Table 10 – NLME-DISCOVERY.indication parameters

Name	Type	Valid range	Description
Status	Enumeration	SUCCESS or NO_REC_CAPACITY	The status of the pairing table.
SrcIEEEAddr	IEEE address	A valid IEEE address	The IEEE address of the device requesting the discovery.
OrgNodeCapabilities	Bitmap	See Figure 26	The capabilities of the originator of the discovery request.
OrgVendorId	Vendor ID	A valid vendor identifier	The vendor identifier of the originator of the discovery request. The set of vendor IDs is specified in [R2] .
OrgVendorString	Octet string	7 octets	The vendor string of the originator of the discovery request.
OrgAppCapabilities	Bitmap	See Figure 18	The application capabilities of the originator of the discovery request.
OrgUserString	Character string	0 or 15 characters	The user defined identification string of the originator of the discovery request.
OrgDevTypeList	List of integers	Each integer: 0x00 – 0xfe	The list of device types supported by the originator of the discovery request. The set of supported device types is specified in [R3] .

Name	Type	Valid range	Description
OrgProfileIdList	List of integers	Each integer: 0x00 – 0xff	The list of profile identifiers supported by the originator of the discovery request. The set of supported profile identifiers is specified in [R4]
SearchDevType	Integer	0x00 – 0xff	The device type being discovered. If this is 0xff, any type is being requested.
RxLinkQuality	Integer	0x00 – 0xff	LQI value, as passed via the MAC sub-layer, of the discovery request command frame.

3.1.2.5.2 When generated

The NLME-DISCOVERY.indication primitive is generated by the NLME and issued to the application to indicate the reception of a discovery request command frame.

If the NLME has spare capacity in its pairing table for a potential pairing link with this device, it issues the primitive with a status of SUCCESS. If the NLME does not have capacity in its pairing table, it issues the primitive with a status of NO_REC_CAPACITY.

3.1.2.5.3 Effect on receipt

On receipt of the NLME-DISCOVERY.indication primitive, the application decides whether to respond based on the information contained in the primitive. The decision to respond, i.e. whether the discovery request matches the capabilities of this node, is out of the scope of this specification.

If the application decides to respond, it issues the NLME-DISCOVERY.response primitive with the source IEEE address and LQI from the received discovery request command frame, contained in this primitive, its own device type and the status value received in the NLME-DISCOVERY.indication primitive.

If the application decides not to respond, no primitive is issued.

3.1.2.6 NLME-DISCOVERY.response

The NLME-DISCOVERY.response primitive allows the application to request that the NLME respond to the discovery request command.

3.1.2.6.1 Semantics of the service primitive

The semantics of the NLME-DISCOVERY.response primitive are as follows:

```
NLME-DISCOVERY.response        (
                               Status,
                               DstIEEEAddr,
                               RecAppCapabilities,
                               RecDevTypeList,
                               RecProfileIdList,
                               DiscReqLQI
                               )
```

Table 11 specifies the parameters for the NLME-DISCOVERY.response primitive.

Table 11 – NLME-DISCOVERY.response parameters

Name	Type	Valid range	Description
Status	Enumeration	SUCCESS or NO_REC_CAPACITY	The status of the discovery request.
DstIEEEAddr	IEEE address	Valid IEEE address	The IEEE address of the device requesting discovery.
RecAppCapabilities	Bitmap	See Figure 18	The application capabilities of the node issuing this primitive.
RecDevTypeList	List of integers	Each integer: 0x00 – 0xfe	The list of device types supported by the node issuing this primitive. The set of supported device types is specified in [R3] .
RecProfileIdList	List of integers	Each integer: 0x00 – 0xff	The list of profile identifiers supported by the node issuing this primitive. The set of supported profile identifiers is specified in [R4]
DiscReqLQI	Integer	0x00 – 0xff	The LQI value from the associated NLME-DISCOVERY.indication primitive.

3.1.2.6.2 When generated

The NLME-DISCOVERY.response primitive is generated by the application and issued to its NLME in response to an NLME-DISCOVERY.indication primitive.

3.1.2.6.3 Effect on receipt

On receipt of this primitive, the NLME generates a discovery response command frame (see 3.3.2). The NLME transmits the frame on its current channel by issuing the MCPS-DATA.request primitive to the MAC sub-layer.

On receipt of an MCPS-DATA.confirm primitive from the MAC sub-layer, the NLME issues the NLME-COMM-STATUS.indication primitive with the status value contained in the primitive.

3.1.2.7 NLME-DISCOVERY.confirm

The NLME-DISCOVERY.confirm primitive allows the NLME to notify the application of the status of its request to perform a network discovery.

3.1.2.7.1 Semantics of the service primitive

The semantics of the NLME-DISCOVERY.confirm primitive are as follows:

```
NLME-DISCOVERY.confirm        (
                              Status,
                              NumNodes,
                              NodeDescList
                              )
```

Table 12 specifies the parameters for the NLME-DISCOVERY.confirm primitive.

Table 12 – NLME-DISCOVERY.confirm parameters

Name	Type	Valid range	Description
Status	Enumeration	SUCCESS, DISCOVERY_ERROR, DISCOVERY_TIMEOUT or anything returned from the MCPS-DATA.confirm primitive	The status of the network discovery attempt.
NumNodes	8-bit integer	0x00 - *nwkcMaxNodeDescListSize*	The number of discovered nodes in the NodeDescList parameter.
NodeDescList	NodeDesc	See Table 13	The list of node descriptors discovered.

Table 13 describes the elements of the NodeDesc type.

Table 13 – Elements of the NodeDesc type

Name	Type	Valid range	Description
Status	Enumeration	SUCCESS or NO_REC_CAPACITY	The status of the discovery request as reported by the responding device.
LogicalChannel	8-bit integer	See 3.5.1.1	The logical channel of the responding device.
PANId	PAN identifier	A valid PAN identifier	The PAN identifier of the responding device.
IEEEAddr	IEEE address	A valid IEEE address	The IEEE address of the responding device.
NodeCapabilities	Bitmap	See Figure 26	The capabilities of the responding node.
VendorId	Vendor identifier	A valid vendor identifier	The vendor identifier of the responding node. The set of vendor IDs is specified in [R2] .
VendorString	Octet string	7 octets	The vendor string of the responding node.
AppCapabilities	Bitmap	See Figure 18	The application capabilities of the responding node.

Name	Type	Valid range	Description
UserString	Character string	0 or 15 characters	The user defined identification string of the responding node. This field is present only if the user string specified sub-field of the AppCapabilities field is set to one.
DevTypeList	List of integers	Each integer: 0x00 – 0xfe	The list of device types supported by the responding node. The set of supported device types is specified in [R3] .
ProfileIdList	List of integers	Each integer: 0x00 – 0xff	The list of profile identifiers supported by the responding node. The set of supported profile identifiers is specified in [R4]
DiscReqLQI	8-bit integer	0x00 – 0xff	The LQI of the discovery request command frame reported by the responding device.

3.1.2.7.2 When generated

The NLME-DISCOVERY.confirm primitive is generated by the initiating NLME and issued to the application in response to an NLME-DISCOVERY.request primitive. If the request was successful, the status parameter will indicate a successful discovery attempt. Otherwise, the status parameter indicates an appropriate error code from the MCPS-DATA.confirm primitive.

3.1.2.7.3 Effect on receipt

On receipt of the NLME-DISCOVERY.confirm primitive, the application of the initiating device is notified of the result of its discovery request attempt. If the discovery attempt was successful, the status parameter will indicate a successful discovery and the application will be provided with a list of those devices that responded to the discovery request. Each element in the list contains enough information about the responding node to allow a subsequent pairing operation to proceed. If the discovery attempt was unsuccessful, the status parameter will indicate the error and all other parameters should be ignored.

3.1.2.7.4 Discovery message sequence chart

Figure 7 illustrates the sequence of messages necessary for a successful discovery attempt. Discovery is attempted on each available channel at a rate of *nwkDiscoveryRepetitionInterval* and for *nwkMaxDiscoveryRepetitions* (*n*).

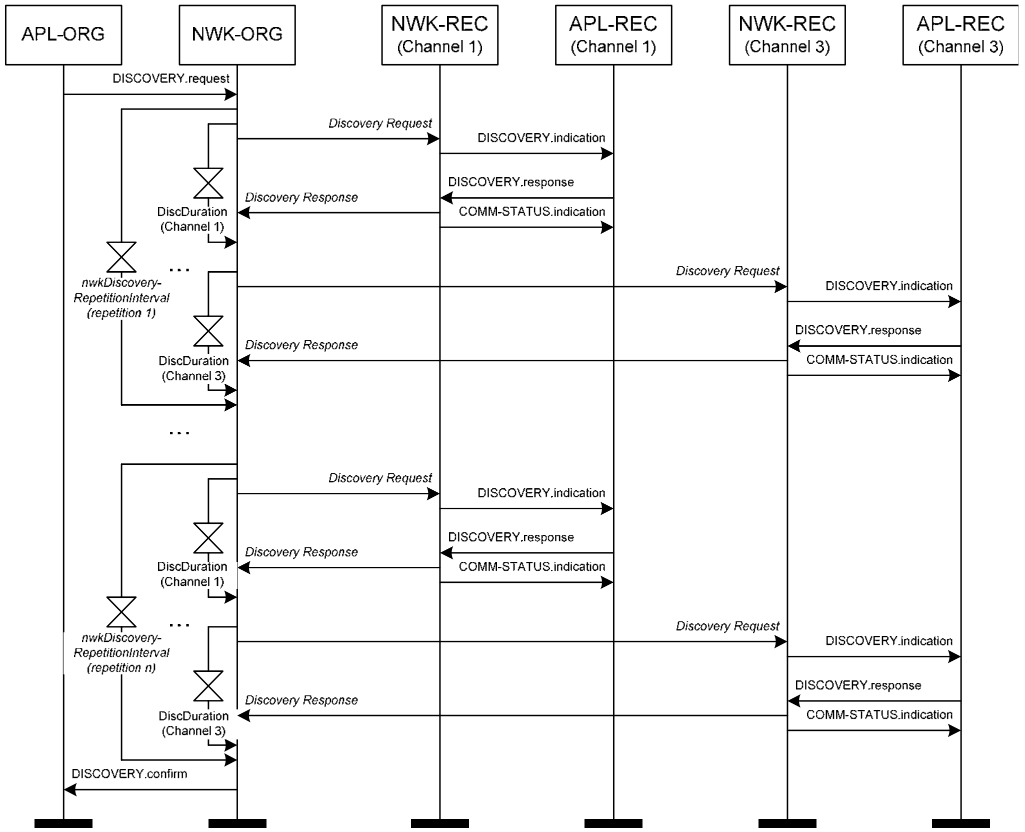


Figure 7 – Message sequence chart for discovery

3.1.2.8 NLME-GET.request

The NLME-GET.request primitive allows the application to request the value of a NIB attribute from the NLME.

3.1.2.8.1 Semantics of the service primitive

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

```
NLME-GET.request          (
                          NIBAtribute,
                          NIBAttributeIndex,
                          )
```

Table 14 specifies the parameters for the NLME-GET.request primitive.

Table 14 – NLME-GET.request parameters

Name	Type	Valid range	Description
NIBAttribute	Integer	See Table 48	The identifier of the NIB attribute to read.

Name	Type	Valid range	Description
NIBAttributeIndex	Integer	Attribute specific	The index within the table or array of the specified NIB attribute to read. This parameter is valid only for NIB attributes that are tables or arrays.

3.1.2.8.2 When generated

The NLME-GET.request primitive is generated by the application and issued to the NLME to obtain information from the NIB.

3.1.2.8.3 Effect on receipt

On receipt of the NLME-GET.request primitive, the NLME checks to see if the NIB attribute exists. If the attribute exists, the NLME attempts to receive the requested NIB attribute from its database. If the identifier of the NIB attribute is not found in the database, the NLME will issue the NLME-GET.confirm primitive with a status of UNSUPPORTED_ATTRIBUTE. If the NIBAttributeIndex parameter specifies an index for a table that is out of range, the NLME will issue the NLME-GET.confirm primitive with a status of INVALID_INDEX. If the requested NIB attribute is successfully retrieved, the NLME will issue the NLME-GET.confirm primitive with a status of SUCCESS and the value of the requested attribute in the NIBAttributeValue parameter.

3.1.2.9 NLME-GET.confirm

The NLME-GET.confirm primitive allows the NLME to notify the application of the status of its request for the value of a NIB attribute.

3.1.2.9.1 Semantics of the service primitive

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

```
NLME-GET.confirm          (
                          Status,
                          NIBAttribute,
                          NIBAttributeIndex,
                          NIBAttributeValue
                          )
```

Table 15 specifies the parameters for the NLME-GET.confirm primitive.

Table 15 – NLME-GET.confirm parameters

Name	Type	Valid range	Description
Status	Enumeration	SUCCESS, UNSUPPORTED_-ATTRIBUTE or INVALID INDEX	The status of the request for NIB attribute information.
NIBAttribute	Integer	See Table 48	The identifier of the NIB attribute that was read.
NIBAttributeIndex	Integer	Attribute specific	The index within the table or array of the specified NIB attribute that was read. This parameter is valid only for NIB attributes that are tables or arrays.

Name	Type	Valid range	Description
NIBAttributeValue	Various	Attribute specific	The value of the NIB attribute that was read. This value has a zero length when the Status parameter is not equal to SUCCESS.

3.1.2.9.2 When generated

The NLME-GET.confirm primitive is generated by the NLME and issued to the application in response to an NLME-GET.request primitive. This primitive returns a status of either SUCCESS, indicating that the request to read a NIB attribute was successful, or an error code of INVALID_INDEX or UNSUPPORTED_ATTRIBUTE. The status values are fully described in 3.1.2.6.3.

3.1.2.9.3 Effect on receipt

On receipt of the NLME-GET.confirm primitive, the application is notified of the results of its request to read a NIB attribute. If the request to read a NIB attribute was successful, the status parameter will be set to SUCCESS. Otherwise, the status parameter indicates the error.

3.1.2.10 NLME-PAIR.request

The NLME-PAIR.request primitive allows the application to request the NLME pair with another device. This primitive would normally be issued following a discovery operation via the NLME-DISCOVERY.request primitive.

3.1.2.10.1 Semantics of the service primitive

The semantics of the NLME-PAIR.request primitive are as follows:

```
NLME-PAIR.request        (
                         LogicalChannel,
                         DstIEEEAddr,
                         DstPANId,
                         OrgAppCapabilities,
                         OrgDevTypeList,
                         OrgProfileIdList,
                         KeyExTransferCount
                         )
```

Table 16 specifies the parameters for the NLME-PAIR.request primitive.

Table 16 – NLME-PAIR.request parameters

Name	Type	Valid range	Description
LogicalChannel	Integer	See 3.5.1.1	The logical channel of the device with which to pair.
DstPANId	PAN identifier	A valid PAN identifier	The PAN identifier of the device with which to pair.
DstIEEEAddr	IEEE address	A valid IEEE address	The IEEE address of the device with which to pair.

Name	Type	Valid range	Description
OrgAppCapabilities	Bitmap	See Figure 18	The application capabilities of the node issuing this primitive.
OrgDevTypeList	List of integers	Each integer: 0x00 – 0xfe	The list of device types supported by the node issuing this primitive. The set of supported device types is specified in [R3] .
OrgProfileIdList	List of integer	Each integer: 0x00 – 0xff	The list of profile identifiers supported by the node issuing this primitive. The set of supported profile identifiers is specified in [R4]
KeyExTransferCount	Integer	0x00 – 0xff	The number of transfers the target should use to exchange the link key with the pairing originator.

3.1.2.10.2 When generated

The NLME-PAIR.request primitive is generated by the local application entity and issued to the NLME to request a pairing operation.

3.1.2.10.3 Effect on receipt

On receipt of the NLME-PAIR.request primitive, the NLME checks whether the requested pairing link is a duplicate of an already existing pairing table entry. If a duplicate entry exists, that pairing table entry will be updated. If a duplicate entry does not exist, the NLME checks whether it has capacity in its pairing table to store the potential new pairing link. If the NLME has no capacity in its pairing table, it issues the NLME-PAIR.confirm primitive with a status of NO_ORG_CAPACITY and performs no further processing.

The NLME then generates a pair request command (see 3.3.3), with the information supplied in the primitive. Before the NLME transmits the frame, it changes *phyCurrentChannel* to the requested channel by issuing the MLME-SET.request command to the MAC sub-layer. Finally, the NLME transmits the pair request command by issuing the MCPS-DATA.request primitive to the MAC sub-layer.

If the MAC sub-layer fails to transmit the frame (i.e. a status of anything but SUCCESS is returned from the MCPS-DATA.confirm primitive), the NLME issues the NLME-PAIR.confirm primitive with the status value returned from the MAC sub-layer and performs no further processing.

If the MAC sub-layer successfully transmits the frame, the NLME waits *nwkResponseWaitTime* symbols for the corresponding pair response command (see 3.3.4) to arrive. If a pair response command is not received within this time, the NLME issues the NLME-PAIR.confirm primitive with a status of NO_RESPONSE and performs no further processing.

On receipt of a pair response command frame with a status field not equal to SUCCESS, the NLME issues the NLME-PAIR.confirm primitive with a status equal to that received in the pair response command frame and performs no further processing.

On receipt of a pair response command frame with a status field equal to SUCCESS, the NLME creates a new entry or updates an existing entry, as determined above, in the pairing table with the information contained in the pair response command frame and marks it as provisional.

The NLME then checks whether security is required for this pairing link. If security is not required for this pairing link the NLME moves the pairing link from provisional to active and issues the NLME-PAIR.confirm primitive with the pairing table index at which the pairing link was created or updated and a status of SUCCESS and performs no further processing.

If security is required for this pairing link, the NLME waits for the reception of (n + 1) key seed command frames from the target, where n is equal to the value of the KeyExTransferCount parameter. If all the transfers are not received within ((n + 1) * *nwkcMaxKeySeedWaitTime*) symbols, the NLME removes the provisional pairing link, issues the NLME-PAIR.confirm primitive with a status of SECURITY_TIMEOUT and performs no further processing.

If all the transfers are received within ((*n* + 1) * *nwkcMaxKeySeedWaitTime*) symbols, the NLME generates a ping request command (see 3.3.7) and transmits it to the pairing recipient by issuing the MCPS-DATA.request primitive to the MAC sub-layer.

If the MAC sub-layer fails to transmit the frame (i.e. a status of anything but SUCCESS is returned from the MCPS-DATA.confirm primitive), the NLME removes the provisional pairing link, issues the NLME-PAIR.confirm primitive with the status value returned from the MAC sub-layer and performs no further processing.

If the MAC sub-layer successfully transmits the frame, the NLME waits *nwkResponseWaitTime* symbols for the corresponding ping response command (see 3.3.8) to arrive. If a ping response command frame is not received within this time, the NLME removes the provisional pairing link, issues the NLME-PAIR.confirm primitive with a status of NO_RESPONSE and performs no further processing.

If a ping response command frame is received within this time, the NLME verifies the information it contains. If the information is not verified, the NLME removes the provisional pairing link, issues the NLME-PAIR.confirm primitive with a status of SECURITY_FAILURE and performs no further processing. If the information is verified, the NLME moves the pairing table entry from provisional to active and issues the NLME-PAIR.confirm primitive with the pairing table index at which the pairing link was created or updated and a status of SUCCESS.

3.1.2.11 NLME-PAIR.indication

The NLME-PAIR.indication primitive allows the NLME to notify the application of the reception of a pairing request command (see 3.3.3).

3.1.2.11.1 Semantics of the service primitive

The semantics of the NLME-PAIR.indication primitive are as follows:

```
NLME-PAIR.indication          (
                              Status,
                              SrcPANId,
                              SrcIEEEAddr,
                              OrgNodeCapabilities,
                              OrgVendorId,
                              OrgVendorString,
                              OrgAppCapabilities,
                              OrgUserString,
                              OrgDevTypeList,
                              OrgProfileIdList,
                              ProvPairingRef,
                              )
```

Table 17 specifies the parameters for the NLME-PAIR.indication primitive.

Table 17 – NLME-PAIR.indication parameters

Name	Type	Valid range	Description
Status	Enumeration	SUCCESS, NO_REC_CAPACITY or DUPLICATE_PAIRING	The status of the provisional pairing.

Name	Type	Valid range	Description
SrcPANId	PAN identifier	A valid PAN identifier	The PAN identifier of the device requesting the pair.
SrcIEEEAddr	IEEE address	A valid IEEE address	The IEEE address of the device requesting the pair.
OrgNodeCapabilties	Bitmap	See Figure 26	The capabilities of the originator of the pair request.
OrgVendorId	Vendor ID	A valid vendor identifier	The vendor identifier of the originator of the pair request. The set of vendor IDs is specified in [R2] .
OrgVendorString	Octet string	7 octets	The vendor string of the originator of the pair request.
OrgAppCapabilities	Bitmap	See Figure 18	The application capabilities of the originator of the pair request.
OrgUserString	Character string	0 or 15 characters	The user defined identification string of the originator of the pair request.
OrgDevTypeList	List of integers	Each integer: 0x00 – 0xfe	The list of device types supported by the originator of the pair request. The set of supported device types is specified in [R3] .
OrgProfileIdList	List of integers	Each integer: 0x00 – 0xff	The list of profile identifiers supported by the originator of the pair request. The set of supported profile identifiers is specified in [R4]
ProvPairingRef	Integer	0x00 – 0xff	The pairing reference that will be used if this pairing request is successful or 0xff if there is no further capacity in the pairing table.

3.1.2.11.2 When generated

The NLME-PAIR.indication primitive is generated by the NLME and issued to the application to indicate the reception of a pair request command frame.

The Status parameter allows the NLME to notify the application of the status of the provisional pairing entry. If the NLME detects that the requested pairing link is a duplicate of an already existing pairing table entry, the NLME sets the Status parameter to DUPLICATE_PAIRING and sets the ProvPairingRef parameter to the index of the duplicate entry.

If there is no free entry in the pairing table for the new pairing link, the NLME sets the Status parameter to NO_REC_CAPACITY and sets the ProvPairingRef parameter to 0xff.

Otherwise, the NLME sets the Status parameter to SUCCESS and sets the ProvPairingRef parameter to the index of the next free pairing table entry.

3.1.2.11.3 Effect on receipt

On receipt of the NLME-PAIR.indication primitive, the application decides how to respond based on the information supplied in the primitive. The decision to respond, i.e. whether the pair request matches the capabilities of this node, is out of the scope of this specification.

The ProvPairingRef parameter indicates the next free pairing table entry at which this pairing link will be created. If this parameter is equal to 0xff, there is no free entry in the pairing table and the application issues the NLME-PAIR.response primitive with a status of NO_REC_CAPACITY.

If the application decides to allow the pair, it issues the NLME-PAIR.response primitive with its own device type and a status of SUCCESS. If the application decides not to allow the pair, it issues the NLME-PAIR.response primitive with a status of NOT_PERMITTED.

3.1.2.12 NLME-PAIR.response

The NLME-PAIR.response primitive allows the application to request that the NLME respond to a pairing request command (see 3.3.3).

3.1.2.12.1 Semantics of the service primitive

The semantics of the NLME-PAIR.response primitive are as follows:

```
NLME-PAIR.response          (
                            Status,
                            DstPANId,
                            DstIEEEAddr,
                            RecAppCapabilities,
                            RecDevTypeList,
                            RecProfileIdList,
                            ProvPairingRef,
                            )
```

Table 18 specifies the parameters for the NLME-PAIR.response primitive.

Table 18 – NLME-PAIR.response parameters

Name	Type	Valid range	Description
Status	Enumeration	SUCCESS, NO_REC_CAPACITY or NOT_PERMITTED	The status of the pairing request.

Name	Type	Valid range	Description
DstPANId	PAN identifier	A valid PAN identifier	The PAN identifier of the device requesting the pair.
DstIEEEAddr	IEEE address	A valid IEEE address	The IEEE address of the device requesting the pair.
RecAppCapabilities	Bitmap	See Figure 18	The application capabilities of the node issuing this primitive.
RecDevTypeList	List of integers	Each integer: 0x00 – 0xfe	The list of device types supported by the node issuing this primitive. The set of supported device types is specified in [R3] .
RecProfileIdList	List of integers	Each integer: 0x00 – 0xff	The list of profile identifiers supported by the node issuing this primitive. The set of supported profile identifiers is specified in [R4]
ProvPairingRef	Integer	0x00 – 0xff	The reference to the provisional pairing entry if the pair was accepted or 0xff otherwise.

3.1.2.12.2 When generated

The NLME-PAIR.response primitive is generated by the application and issued to its NLME in response to an NLME-PAIR.indication primitive.

3.1.2.12.3 Effect on receipt

On receipt of this primitive and if the Status parameter is equal to SUCCESS, the NLME updates the entry in the pairing table corresponding to the ProvPairingRef parameter by specifying a network address for the originator of the pairing link. If the capabilities of the originator node indicate that it is a controller device, the NLME allocates a network address for that device. Otherwise, the network address is specified as 0xfffe.

The NLME then generates a pair response command frame (see 3.3.4). The NLME transmits the frame on its current channel by issuing the MCPS-DATA.request primitive to the MAC sub-layer.

On receipt of an MCPS-DATA.confirm primitive from the MAC sub-layer with a status of anything but SUCCESS, the NLME issues the NLME-COMM-STATUS.indication primitive with the status value contained in the MCPS-DATA.confirm primitive, removes the provisional pairing link and performs no further processing.

If an MCPS-DATA.confirm primitive is received from the MAC sub-layer with a status of SUCCESS but the application did not accept the pair (i.e. the Status parameter of the NLME-PAIR.response primitive was not equal to SUCCESS), the NLME removes the provisional pairing link and performs no further processing.

If an MCPS-DATA.confirm primitive is received from the MAC sub-layer with a status of SUCCESS and the application accepted the pair (i.e. the Status parameter of the NLME-PAIR.response primitive was equal to SUCCESS), the NLME determines whether security is required for this pairing link. If security is not required for this pairing link, the NLME issues the NLME-COMM-STATUS.indication primitive with a status value of SUCCESS, moves the pairing link from provisional to active and performs no further processing.

If security is required for this pairing link, the NLME generates a security link key and attempts to transfer it to the originator. To transfer the key, the NLME shall send (*n* + 1) key seed command frames to the pairing originator, where *n* is equal to the value of the key exchange transfer count field of the pair request command frame.

If all the transfers do not complete within ((*n* + 1) * *nwkcMaxKeySeedWaitTime*) symbols, the NLME issues the NLME-COMM-STATUS.indication primitive with a status of SECURITY_TIMEOUT, removes the provisional pairing link and performs no further processing.

If all the transfers complete within ((*n* + 1) * *nwkcMaxKeySeedWaitTime*) symbols, the NLME waits for at most *nwkResponseWaitTime* to receive a ping request command frame from the originator of the pair request command frame. If a ping request command frame is not received within *nwkResponseWaitTime*, the NLME issues the NLME-COMM-STATUS.indication primitive with a status of SECURITY_FAILURE, removes the provisional pairing link and performs no further processing.

If a ping request command frame is received within *nwkResponseWaitTime*, the NLME verifies the data contained in the ping request command frame. If the ping request command frame cannot be verified in this way, the NLME issues the NLME-COMM-STATUS.indication primitive with a status of SECURITY_FAILURE, removes the provisional pairing link and performs no further processing.

If the ping request command frame was verified as described above, the NLME generates a ping response command frame and transmits it back to the originator of the ping request command frame by issuing the MCPS-DATA.request primitive to the MAC sub-layer.

On receipt of an MCPS-DATA.confirm primitive from the MAC sub-layer with a status other than SUCCESS, the NLME issues the NLME-COMM-STATUS.indication primitive with the value returned from the MAC sub-layer, removes the provisional pairing link and performs no further processing.

On receipt of an MCPS-DATA.confirm primitive from the MAC sub-layer with a status of SUCCESS, the NLME issues the NLME-COMM-STATUS.indication primitive with a status value of SUCCESS and then moves the pairing link from provisional to active.

3.1.2.13 NLME-PAIR.confirm

The NLME-PAIR.confirm primitive allows the NLME to notify the application of the status of its request to pair with another device.

3.1.2.13.1 Semantics of the service primitive

The semantics of the NLME-PAIR.confirm primitive are as follows:

```
NLME-PAIR.confirm          (
                           Status,
                           PairingRef,
                           RecVendorId,
                           RecVendorString,
                           RecAppCapabilities,
                           RecUserString,
                           RecDevTypeList,
                           RecProfileIdList
                           )
```

Table 19 specifies the parameters for the NLME-PAIR.confirm primitive.

Table 19 – NLME-PAIR.confirm parameters

Name	Type	Valid range	Description
Status	Enumeration	SUCCESS, NO_ORG_CAPACITY, NO_REC_CAPACITY, NO_RESPONSE, NOT_PERMITTED, SECURITY_FAILURE, SECURITY_TIMEOUT or anything returned from the MCPS-DATA.confirm primitive.	The status of the pair attempt.
PairingRef	Integer	0x00 – 0xff	The pairing table reference for this pairing link. If the Status parameter is not equal to SUCCESS, this value will be equal to 0xff.
RecVendorId	Vendor ID	A valid vendor identifier	The vendor identifier of the originator of the pair response. The set of vendor IDs is specified in [R2] .
RecVendorString	Octet string	7 octets	The vendor string of the originator of the pair response.
RecAppCapabilities	Bitmap	See Figure 18	The application capabilities of the originator of the pair response.
RecUserString	Character string	0 or 15 characters	The user defined identification string of the originator of the pair response.
RecDevTypeList	List of integers	Each integer: 0x00 – 0xfe	The list of device types supported by the originator of the pair response. The set of supported device types is specified in [R3] .
RecProfileIdList	List of integers	Each integer: 0x00 – 0xff	The list of profile identifiers supported by the originator of the pair response. The set of supported profile identifiers is specified in [R4]

3.1.2.13.2 When generated

The NLME-PAIR.confirm primitive is generated by the initiating NLME and issued to the application in response to an NLME-PAIR.request primitive. If the request was successful, the status parameter will indicate a successful pair, as contained in the Status field of the pair response command frame (see 3.3.4). Otherwise, the status parameter indicates either an error code from the received pair response command frame or an appropriate error code from the MCPS-DATA.confirm primitive.

3.1.2.13.3 Effect on receipt

On receipt of the NLME-PAIR.confirm primitive, the application of the originating device is notified of the result of its pair request. If the pair attempt was successful, the status parameter will indicate a successful pair, as contained in the Status field of the pair response command frame, and the device will be provided with the pairing reference on which the pair was created. If the pair attempt was unsuccessful, the status parameter will indicate the error and all other parameters are invalid.

3.1.2.13.4 Pairing message sequence chart

Figure 8 illustrates the sequence of messages necessary for a successful pairing attempt. Note that this message sequence chart also illustrates the security key exchange procedure which is only included if the nodes at both ends of the pairing link can support security.

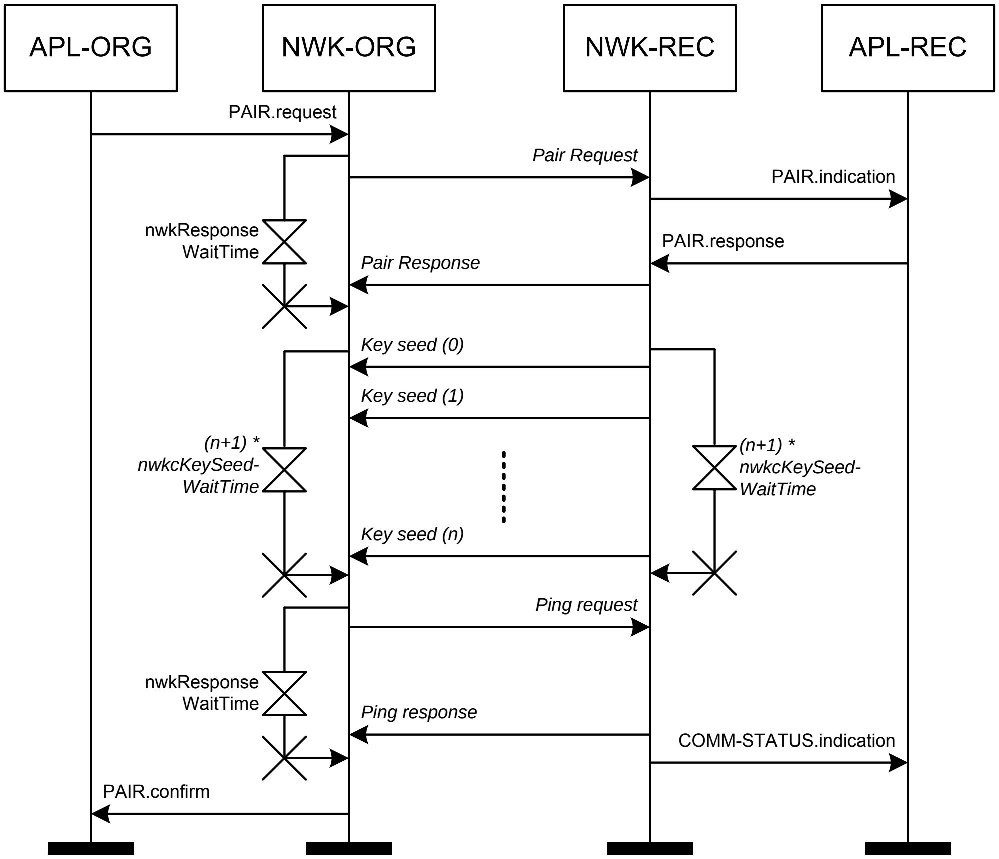


Figure 8 – Message sequence chart for pairing

3.1.2.14 NLME-RESET.request

The NLME-RESET.request primitive allows the application entity to request a reset of the NWK layer.

3.1.2.14.1 Semantics of the service primitive

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

```
NLME-RESET.request          (
                            SetDefaultNIB
                            )
```

Table 20 specifies the parameters for the NLME-RESET.request primitive.

Table 20 – NLME-RESET.request parameters

Name	Type	Valid range	Description
SetDefaultNIB	Boolean	TRUE or FALSE	If TRUE, the NWK layer is reset and all NIB attributes are set to their default values. If FALSE, the NWK layer is reset but all NIB attributes retain their values prior to the generation of the NLME-RESET.request primitive.

3.1.2.14.2 When generated

The NLME-RESET.request primitive is generated by the application and issued to the NLME to request a reset of the NWK layer to its initial conditions. The NLME-RESET.request primitive is issued prior to the use of the NLME-START.request primitive.

3.1.2.14.3 Effect on receipt

On receipt of the NLME-RESET.request primitive, the NLME issues the MLME-RESET.request primitive with the SetDefaultPIB parameter set to the value of the SetDefaultNIB parameter. On receipt of the MLME-RESET.confirm primitive, the NWK layer is then set to its initial conditions, clearing all internal variables to their default values. If the SetDefaultNIB parameter is set to TRUE, the NIB attributes are set to their default values.

Note: issuing this primitive with the SetDefaultNIB parameter set to TRUE will remove all entries from the pairing table.

3.1.2.15 NLME-RESET.confirm

The NLME-RESET.confirm primitive allows the NLME to notify the application of the status of its request to reset the NWK layer.

3.1.2.15.1 Semantics of the service primitive

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

```
NLME-RESET.confirm          (
                            Status
                            )
```

Table 21specifies the parameters for the NLME-RESET.confirm primitive.

Table 21 – NLME-RESET.confirm parameters

Name	Type	Valid range	Description
Status	Enumeration	SUCCESS	The status of the reset request.

3.1.2.15.2 When generated

The NLME-RESET.confirm primitive is generated by the NLME and issued to the application in response to an NLME-RESET.request primitive.

3.1.2.15.3 Effect on receipt

On receipt of the NLME-RESET.confirm primitive, the application is notified of its request to reset the NWK layer. This primitive returns a status of SUCCESS indicating that the request to reset the NWK layer was successful.

3.1.2.16 NLME-RX-ENABLE.request

The NLME-RX-ENABLE.request primitive allows the application to request that the receiver is either enabled (for a finite period or until further notice) or disabled.

3.1.2.16.1 Semantics of the service primitive

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

```
NLME-RX-ENABLE.request          (
                                RxOnDuration
                                )
```

Table 22 specifies the parameters for the NLME-RX-ENABLE.request primitive.

Table 22 – NLME-RX-ENABLE.request parameters

Name	Type	Valid range	Description
RxOnDuration	Integer	0x000000 – 0xffffff	The number of MAC symbols for which the receiver is to be enabled. For power saving mode, this value should correspond to the value of *nwkActivePeriod*. If this parameter is equal to 0x000000, the receiver is to be disabled until further notice, allowing the node to enter dormant power save mode. If this parameter is equal to 0xffffff, the receiver is to be enabled until further notice, allowing the node to come out of power save mode.

3.1.2.16.2 When generated

The NLME-RX-ENABLE.request primitive is generated by the application and issued to the NLME to either enable the receiver (for a fixed duration or until further notice) or to disable the receiver.

3.1.2.16.3 Effect on receipt

On receipt of the NLME-RX-ENABLE.request primitive, the NLME checks the value of the RxOnDuration parameter. If the value of this parameter is equal to 0xffffff, the NLME sets *nwkInPowerSave* to FALSE and sets *macRxOnWhenIdle* to TRUE by issuing the MLME-SET.request primitive to the MAC sub-layer. If the value of this parameter is any value other than 0x000000, the NLME sets *nwkInPowerSave* to TRUE and sets *macRxOnWhenIdle* to FALSE by issuing the MLME-SET.request primitive to the MAC sub-layer. If the value of this parameter is equal to 0x000000, the NLME sets *nwkInPowerSave* to TRUE.

The NLME then issues the MLME-RX-ENABLE.request primitive to the MAC sub-layer with the DeferPermit, RxOnTime and RxOnDuration parameters set to FALSE, 0x000000 and to the value of the RxOnDuration parameter from the NLME-RX-ENABLE.request primitive, respectively.

The NLME then waits until it receives the MLME-RX-ENABLE.confirm primitive from the MAC sub-layer and issues the NLME-RX-ENABLE.confirm primitive with the Status parameter set equal to the Status parameter received in the MLME-RX-ENABLE.confirm primitive.

3.1.2.17 NLME-RX-ENABLE.confirm

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

3.1.2.17.1 Semantics of the service primitive

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

```
NLME-RX-ENABLE.confirm          (
                                Status
                                )
```

Table 23 specifies the parameters for the NLME-RX-ENABLE.confirm primitive.

Table 23 – NLME-RX-ENABLE.confim parameters

Name	Type	Valid range	Description
Status	Enumeration	As returned by the MLME-RX-ENABLE.confirm primitive	The result of the request to enable or disable the receiver.

3.1.2.17.2 When generated

The NLME-RX-ENABLE.confirm primitive is generated by the NLME and issued to the application in response to an NLME-RX-ENABLE.request primitive.

3.1.2.17.3 Effect on receipt

On receipt of the NLME-RX-ENABLE.confirm primitive, the application is notified of its request to enable or disable the receiver. This primitive returns a status of either SUCCESS, if the request to enable or disable the receiver was successful, or an appropriate error code.

3.1.2.17.4 Message sequence chart for changing the state of the receiver

Figure 9 illustrates the sequence of messages necessary for manipulating the receiver state in various ways. Figure 9 a) illustrates how the application can enabled the receiver until further notice, e.g. when a target comes out of its power save mode. Figure 9 b) illustrates how the application can disable the receiver until further notice, e.g. when a controller has finished transmitting and wishes to sleep.

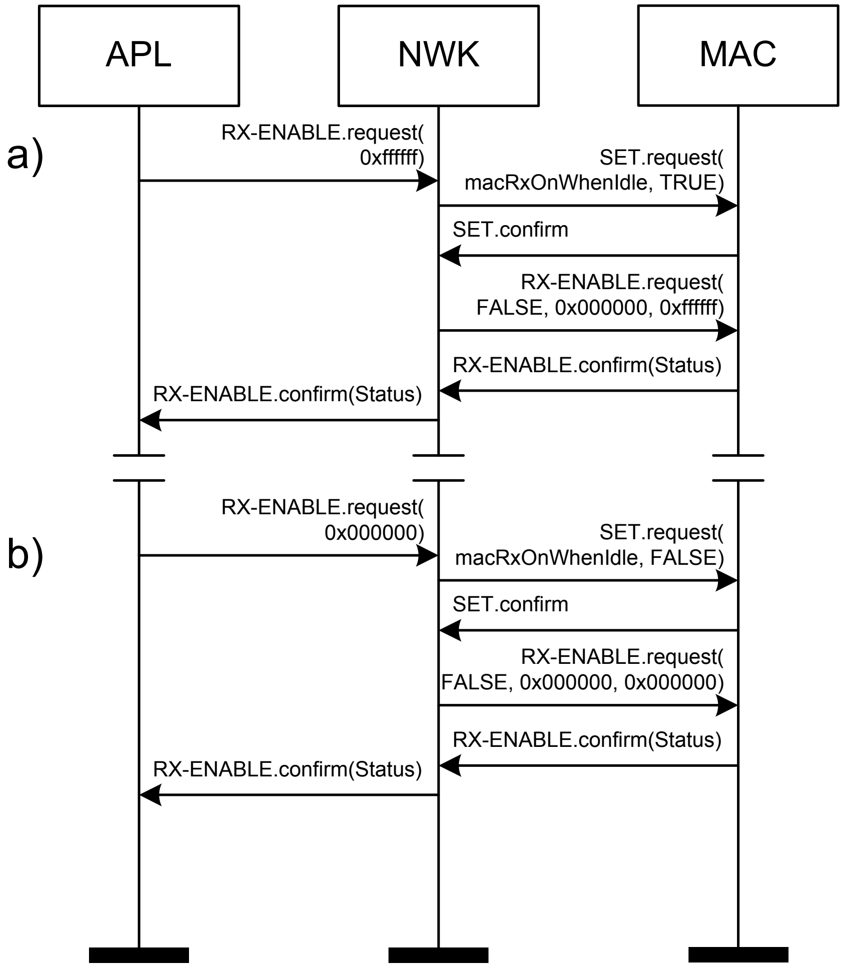


Figure 9 – Message sequence chart for manipulating the receiver

3.1.2.18 NLME-SET.request

The NLME-SET.request primitive allows the application to request the NLME change the value of a NIB attribute.

3.1.2.18.1 Semantics of the service primitive

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

```
NLME-SET.request          (
                          NIBAtribute,
                          NIBAttributeIndex,
                          NIBAttributeValue
                          )
```

Table 24 specifies the parameters for the NLME-SET.request primitive.

Table 24 – NLME-SET.request parameters

Name	Type	Valid range	Description
NIBAttribute	Integer	See Table 48	The identifier of the NIB attribute to write.
NIBAttributeIndex	Integer	Attribute specific	The index within the table or array of the specified NIB attribute to write. This parameter is valid only for NIB attributes that are tables or arrays.
NIBAttributeValue	Various	Attribute specific	The value of the indicated attribute to write.

3.1.2.18.2 When generated

The NLME-SET.request primitive is generated by the application and issued to the NLME to write the indicated NIB attribute.

3.1.2.18.3 Effect on receipt

On receipt of the NLME-SET.request primitive, the NLME checks to see if the NIB attribute exists. If the attribute exists, the NLME attempts to write the given value to the indicated NIB attribute in its database. If the identifier of the NIB attribute is not found in the database, the NLME will issue the NLME-SET.confirm primitive with a status of UNSUPPORTED_ATTRIBUTE. If the NIBAttributeIndex parameter specifies an index for a table that is out of range, the NLME will issue the NLME-SET.confirm primitive with a status of INVALID_INDEX. If the requested NIB attribute is successfully written, the NLME will issue the NLME-SET.confirm primitive with a status of SUCCESS.

3.1.2.19 NLME-SET.confirm

The NLME-SET.confirm primitive allows the NLME to notify the application of the status of its request to change the value of a NIB attribute.

3.1.2.19.1 Semantics of the service primitive

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

```
NLME-SET.confirm        (
                        Status,
                        NIBAttribute,
                        NIBAttributeIndex
                        )
```

Table 25 specifies the parameters for the NLME-SET.confirm primitive.

Table 25 – NLME-SET.confirm parameters

Name	Type	Valid range	Description
Status	Enumeration	SUCCESS, UNSUPPORTED_-ATTRIBUTE or INVALID INDEX	The status of the request to set PIB attribute information.

Name	Type	Valid range	Description
NIBAttribute	Integer	See Table 48	The identifier of the NIB attribute that was written.
NIBAttributeIndex	Integer	Attribute specific	The index within the table or array of the specified NIB attribute that was written. This parameter is valid only for NIB attributes that are tables or arrays.

3.1.2.19.2 When generated

The NLME-SET.confirm primitive is generated by the NLME and issued to the application in response to an NLME-SET.request primitive. The NLME-SET.confirm primitive returns a status of either SUCCESS, indicating that the requested value was written to the indicated NIB attribute, or an appropriate error code. The status values are fully described in 3.1.2.18.3.

3.1.2.19.3 Effect on receipt

On receipt of the NLME-SET.confirm primitive, the application is notified of the result of its request to write the value of a NIB attribute. If the requested value was written to the indicated NIB attribute, the status parameter will be set to SUCCESS. Otherwise, the status parameter indicates the error.

3.1.2.20 NLME-START.request

The NLME-START.request primitive allows the application to request the NLME start a network.

3.1.2.20.1 Semantics of the service primitive

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

```
NLME-START.request                (
                                  )
```

The NLME-START.request primitive does not have any parameters.

3.1.2.20.2 When generated

The NLME-START.request primitive is generated by the application and issued to the NLME to request that a device goes through its startup procedure and enter normal operation.

3.1.2.20.3 Effect on receipt

On receipt of the NLME-START.request primitive, the NLME evaluates the node type field of *nwkcNodeCapabilities*. If this field indicates that the node is a controller, the NLME performs the general device startup procedure and issues the NLME-START.confirm primitive with a status of SUCCESS.

If the node type field of *nwkcNodeCapabilities* indicates that the node is a target, the NLME first performs the general device startup procedure. The NLME will then issue the MLME-SCAN.request primitive to request the MAC sub-layer perform an energy detect scan with the ScanChannels parameter equal to *nwkcChannelMask* and the ScanDuration parameter set to *nwkScanDuration*. On receipt of the MLME-SCAN.confirm primitive, the NLME sets *nwkBaseChannel* to the channel with the least detected energy.

The NLME will then issue the MLME-SCAN.request primitive again to request the MAC sub-layer perform an active scan with the ScanChannels parameter equal to *nwkcChannelMask* and the ScanDuration parameter set to *nwkScanDuration*. On receipt of the MLME-SCAN.confirm primitive with a status of SUCCESS or LIMIT_REACHED, the NLME chooses a PAN identifier that is unique across the PANs detected during the active scan and sets *macPANId* to this value. The device shall then set the value of *macShortAddress* to a randomly generated value in the permitted range. The NLME will then issue the NLME-START.confirm primitive with a status of SUCCESS.

If either of the two scan requests returns an error, the NLME issues the NLME-START.confirm primitive with a status equal to that returned from the scan that failed.

3.1.2.21 NLME-START.confirm

The NLME-START.confirm primitive allows the NLME to notify the application of the status of its request to start a network.

3.1.2.21.1 Semantics of the service primitive

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

```
NLME-START.confirm          (
                            Status
                            )
```

Table 26 specifies the parameters for the NLME-START.confirm primitive.

Table 26 – NLME-START.confirm parameters

Name	Type	Valid range	Description
Status	Enumeration	SUCCESS or any error status values returned from the MLME-SCAN.confirm primitives.	The status of the start attempt.

3.1.2.21.2 When generated

The NLME-START.confirm primitive is generated by the NLME and issued to the application in response to an NLME-START.request primitive. The NLME-START.confirm primitive returns a status of SUCCESS, indicating that the NWK layer has been initialized and, if a target start was requested, that an RC network has been started. If an error occurred, the status parameter will indicate the error.

3.1.2.21.3 Effect on receipt

On receipt of the NLME-START.confirm primitive, the application is notified of the result of its request to start operating. If the request was successful, the status parameter will be set to SUCCESS. Otherwise, the status parameter indicates the error.

3.1.2.22 NLME-UNPAIR.request

The NLME-UNPAIR.request primitive allows the application to request the NLME removes a pairing link with another device both in the local and remote pairing tables.

3.1.2.22.1 Semantics of the service primitive

The semantics of the NLME-UNPAIR.request primitive are as follows:

```
NLME-UNPAIR.request                (
                                   PairingRef
                                   )
```

Table 27 specifies the parameters for the NLME-UNPAIR.request primitive.

Table 27 – NLME-UNPAIR.request parameters

Name	Type	Valid range	Description
PairingRef	Integer	0x00 – 0xfe	The reference into the local pairing table of the entry that is to be removed.

3.1.2.22.2 When generated

The NLME-UNPAIR.request primitive is generated by the local application entity and issued to the NLME to request the removal of a pairing link.

3.1.2.22.3 Effect on receipt

On receipt of the NLME-UNPAIR.request primitive, the NLME searches the pairing table for an entry corresponding to the supplied reference. If no corresponding entry is found, the NLME issues the NLME-UNPAIR.confirm primitive with a status of NO_PAIRING.

If a corresponding entry exists in the pairing table, the NLME generates an unpair request command, with the information contained in the pairing table entry. The NLME then transmits the unpair request command frame by issuing the MCPS-DATA.request primitive to the MAC sub-layer.

On receipt of the MCPS-DATA.confirm primitive from the MAC sub-layer, the NLME removes the pairing table entry and issues the NLME-UNPAIR.confirm primitive with the same status value returned from the MAC sub-layer.

3.1.2.23 NLME-UNPAIR.indication

The NLME-UNPAIR.indication primitive allows the NLME to notify the application of the removal of a pairing link by another device.

3.1.2.23.1 Semantics of the service primitive

The semantics of the NLME-UNPAIR.indication primitive are as follows:

```
NLME-UNPAIR.indication             (
                                   PairingRef
                                   )
```

Table 28 specifies the parameters for the NLME-UNPAIR.indication primitive.

Table 28 – NLME-UNPAIR.indication parameters

Name	Type	Valid range	Description
PairingRef	Integer	0x00 – 0xfe	The pairing table reference that has been removed from the pairing table.

3.1.2.23.2 When generated

The NLME-UNPAIR.indication primitive is generated by the initiating NLME and issued to the application to indicate the reception of an unpair request command frame.

3.1.2.23.3 Effect on receipt

On receipt of the NLME-UNPAIR.indication primitive, the application of the initiating device is notified of the removal of the pairing link. At this point, the application may extract information from the pairing table as necessary in order to inform the user.

Once the application has all the required information it needs, it issues the NLME-UNPAIR.response primitive with the pairing reference supplied in the NLME-UNPAIR.indication primitive.

3.1.2.24 NLME-UNPAIR.response

The NLME-UNPAIR.response primitive allows the application to notify the NLME that the pairing link indicated via the NLME-UNPAIR.indication primitive can be removed from the pairing table.

3.1.2.24.1 Semantics of the service primitive

The semantics of the NLME-UNPAIR.response primitive are as follows:

```
NLME-UNPAIR.response          (
                              PairingRef
                              )
```

Table 29 specifies the parameters for the NLME-UNPAIR.response primitive.

Table 29 – NLME-UNPAIR.response parameters

Name	Type	Valid range	Description
PairingRef	Integer	0x00 – 0xfe	The reference into the local pairing table of the entry that is to be removed.

3.1.2.24.2 When generated

The NLME-UNPAIR.response primitive is generated by the local application entity and issued to the NLME to indicate that a pairing link, previously indicated via the NLME-UNPAIR.indication primitive, can be removed from the pairing table.

3.1.2.24.3 Effect on receipt

On receipt of the NLME-UNPAIR.response primitive, the NLME removes the entry corresponding to the supplied reference. If no corresponding entry is found, the NLME ignores the primitive.

3.1.2.25 NLME-UNPAIR.confirm

The NLME-UNPAIR.confirm primitive allows the NLME to notify the application of the status of its request to remove a pair with another device.

3.1.2.25.1 Semantics of the service primitive

The semantics of the NLME-UNPAIR.confirm primitive are as follows:

```
NLME-UNPAIR.confirm          (
                              Status,
                              PairingRef
                              )
```

Table 30 specifies the parameters for the NLME-UNPAIR.confirm primitive.

Table 30 – NLME-UNPAIR.confirm parameters

Name	Type	Valid range	Description
Status	Enumeration	SUCCESS, NO_PAIRING or a status value from the MCPS-DATA.confirm primitive.	The status of the unpair attempt.
PairingRef	Integer	0x00 – 0xff	The pairing table reference for this pairing link.

3.1.2.25.2 When generated

The NLME-UNPAIR.confirm primitive is generated by the initiating NLME and issued to the application in response to an NLME-UNPAIR.request primitive. If the request was successful, the status parameter will indicate a successful removal of a pairing link. Otherwise, the status parameter indicates either an error code of NO_PAIRING or an appropriate error code from the MCPS-DATA.confirm primitive.

3.1.2.25.3 Effect on receipt

On receipt of the NLME-UNPAIR.confirm primitive, the application of the initiating device is notified of the result of its request to remove a pairing link. If the unpair attempt was successful, the status parameter will indicate a successful unpair. If the unpair attempt was unsuccessful (for example, due to a pairing reference not existing or a channel access failure on transmission of the unpair request command frame), the status parameter will indicate the error and all other parameters are invalid.

3.1.2.25.4 Message sequence chart for removing a pairing link

Figure 10 illustrates the sequence of messages necessary for the successful removal of a pairing link.

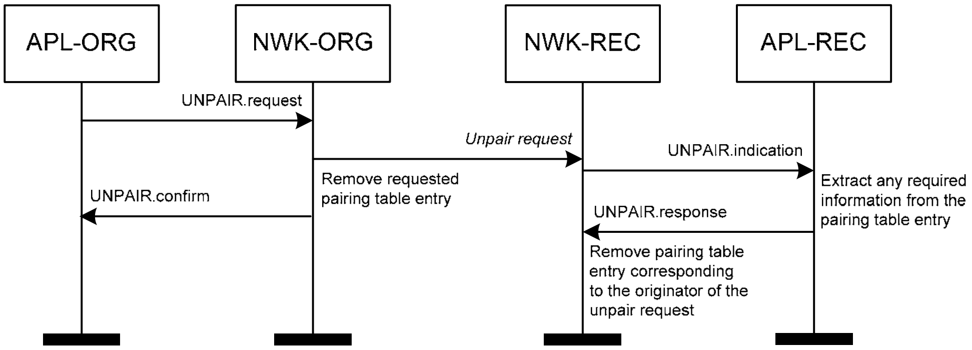

APL-ORG
NWK-ORG
NWK-REC
APL-REC
UNPAIR.request
Unpair request
UNPAIR.indication
Remove requested pairing table entry
Extract any required information from the pairing table entry
UNPAIR.confirm
UNPAIR.response
Remove pairing table entry corresponding to the originator of the unpair request

Figure 10 – Message sequence chart for removing a pairing link

3.1.2.26 NLME-UPDATE-KEY.request

The NLME-UPDATE-KEY.request primitive allows the application to request the NLME change the security link key of an entry in the pairing table.

Note that this primitive provides no network layer support to the application, such as informing the other node of the key update. This primitive should, therefore, only be used to update the key in the pairing table following an application defined key exchange mechanism. Such a mechanism is out of the scope of this specification.

3.1.2.26.1 Semantics of the service primitive

The semantics of the NLME-UPDATE-KEY.request primitive are as follows:

```
NLME-UPDATE-KEY.request          (
                                 PairingRef,
                                 NewLinkKey
                                 )
```

Table 31 specifies the parameters for the NLME-UPDATE-KEY.request primitive.

Table 31 – NLME-UPDATE-KEY.request parameters

Name	Type	Valid range	Description
PairingRef	Integer	0x00 – 0xfe	The reference into the local pairing table of the entry whose key is to be updated.
NewLinkKey	Integer	16-byte link key	The security link key to replace the key in the pairing table.

3.1.2.26.2 When generated

The NLME-UPDATE-KEY.request primitive is generated by the application and issued to the NLME to change the security link key for a pairing table entry.

3.1.2.26.3 Effect on receipt

On receipt of the NLME-UPDATE-KEY.request primitive, the NLME searches the pairing table for an entry corresponding to the supplied reference. If no corresponding entry is found, the NLME issues the NLME-UPDATE-KEY.confirm primitive with a status of NO_PAIRING.

If a corresponding entry exists in the pairing table, the NLME checks that the security capable bit is set in both the recipient capabilities field of the pairing table entry and *nwkcNodeCapabilities*. If both bits are not set, the NLME issues the NLME-UPDATE-KEY.confirm primitive with a status of NOT_PERMITTED.

If both bits are set, the NLME replaces the security link key entry with the value of the NewLinkKey parameter. It then issues the NLME-UPDATE-KEY.confirm primitive with a status of SUCCESS.

3.1.2.27 NLME-UPDATE-KEY.confirm

The NLME-UPDATE-KEY.confirm primitive allows the NLME to notify the application of the status of its request to change the security link key of a pairing table entry.

3.1.2.27.1 Semantics of the service primitive

The semantics of the NLME-UPDATE-KEY.confirm primitive are as follows:

```
NLME-UPDATE-KEY.confirm        (
                               Status,
                               PairingRef
                               )
```

Table 32 specifies the parameters for the NLME-UPDATE-KEY.confirm primitive.

Table 32 – NLME-UPDATE-KEY.confirm parameters

Name	Type	Valid range	Description
Status	Enumeration	SUCCESS, NO_PAIRING or NOT_PERMITTED	The status of the request to update the security link key.
PairingRef	Integer	0x00 – 0xfe	The reference into the local pairing table of the entry whose key is to be updated.

3.1.2.27.2 When generated

The NLME-UPDATE-KEY.confirm primitive is generated by the NLME and issued to the application in response to an NLME-UPDATE-KEY.request primitive. The NLME-UPDATE-KEY.confirm primitive returns a status of either SUCCESS, indicating that the security link key of the requested pairing table entry was updated, or an appropriate error code. The status values are fully described in 3.1.2.26.3.

3.1.2.27.3 Effect on receipt

On receipt of the NLME-UPDATE-KEY.confirm primitive, the application is notified of the result of its request to update the security link key in a pairing table entry. If the security link key of the requested pairing table entry was updated, the status parameter will be set to SUCCESS. Otherwise, the status parameter indicates the error.

3.2 NWK frame formats

This sub-clause specifies the format of the NWK frame (NPDU). Each NWK frame consists of the following components:

a) An NHR, which comprises frame control and frame counter information.

b) A NWK payload, of variable length, which contains information specific to the frame type.

c) An NFR, which comprises the security message integrity code used to secure the frame. The NFR is only included if the frame is secured.

3.2.1 General NWK frame format

The general NWK frame shall be formatted as illustrated in Figure 11

Octets: 1	4	0/1	0/2	Variable	0/4
Frame control	Frame counter	Profile identifier	Vendor identifier	Frame payload	Message integrity code
NHR				NWK Payload	NFR

Figure 11 – General NWK frame format

3.2.1.1 Frame control field

The frame control field is 1 octet in length and contains information defining the frame type and other control flags. The frame control field shall be formatted as illustrated in Figure 12. Note that the reserved sub-field shall be set to 1.

Bits: 0-1	2	3-4	5	6-7
Frame type	Security enabled	Protocol version	Reserved (=1)	Channel designator

Figure 12 – Format of the frame control field

3.2.1.1.1 Frame type sub-field

The frame type sub-field is 2 bits in length and shall be set to one of the non reserved values listed in Table 33.

Table 33 – Values of the frame type sub-field

Frame type value b_1b_0	Description
00	Reserved
01	Standard data frame
10	NWK command frame
11	Vendor specific data frame

3.2.1.1.2 Security enabled sub-field

The security enabled sub-field is 1 bit in length and shall be set to 1 if the frame is protected by the NWK layer. Otherwise it shall be set to 0. The message integrity code field of the NWK frame shall be present only if the security enabled sub-field is set to 1.

3.2.1.1.3 Protocol version sub-field

The protocol version sub-field is 2-bits in length and shall contain the current value of *nwkcProtocolVersion*.

3.2.1.1.4 Channel designator sub-field

The channel designator sub-field is 2-bits in length and specifies a channel designator corresponding to the default channel of the device transmitting the frame. This value can be set to one of the values listed in Table 34.

Table 34 – Values of the channel designator sub-field

Channel designator value b_1b_0	Description
00	Channel not specified
01	Channel 15
10	Channel 20
11	Channel 25

3.2.1.2 Frame counter field

The frame counter field is 4 octets in length and specifies a sequence identifier for the frame being transported. This field is used by the NLDE to detect duplicate transmissions and for securing the NWK frame.

3.2.1.3 Profile identifier field

The profile identifier field is 1 octet in length and specifies the profile identifier of the command being carried in the NWK payload field. The profile identifier field shall only be included for standard and vendor specific data frames.

The set of supported profile identifiers is specified in [R4]

3.2.1.4 Vendor identifier field

The vendor identifier field is 2 octets in length and specifies the vendor identifier of the command being carried in the NWK payload field. The vendor identifier field shall only be included for vendor specific data frames.

3.2.1.5 Frame payload field

The frame payload field has a variable length and contains information specific to individual frame types. If the security enabled sub-field of the frame control field is set to 1, the frame payload is protected.

3.2.1.6 Message integrity code field

The message integrity code field is 4 octets in length. The message integrity code shall be included in the frame only if the security enabled sub-field of the frame control field is set to 1. If the security enabled sub-field of the frame control field is set to zero, the message integrity field shall be omitted from the frame.

3.2.2 Format of individual frame types

3.2.2.1 Data (standard and vendor specific) frame

The data frame shall be formatted as illustrated in Figure 13.

Octets: 1	4	1	0/2	Variable	0/4
Frame control	Frame counter	Profile identifier	Vendor identifier	Data payload	Message integrity code
NHR				NWK Payload	NFR

Figure 13 – Data frame format

The order of the fields of the data frame shall conform to the order of the general NWK frame as illustrated in Figure 11.

3.2.2.1.1 Data frame NHR fields

The NHR for a data frame shall contain the frame control, frame counter and profile identifier fields. In addition, a vendor specific data frame shall also contain the vendor identifier field.

In the frame control field, the frame type shall contain the value that indicates either a standard or vendor specific data frame, as shown in Table 33. If a channel designator is to be included in the frame, the channel designator sub-field shall be set to the value that corresponds to the channel stored in *nwkBaseChannel*, according to the mapping in Table 34; otherwise it shall be set to 0b00.

The frame counter field shall contain the current value of *nwkFrameCounter*.

The profile identifier field shall contain the application specified profile identifier of the data being transported in the data payload field. The set of supported profile identifiers is specified in [R4]

The vendor identifier field shall contain the application specified vendor identifier of the data being transported in the data payload field.

3.2.2.1.2 Data payload field

The payload of a data frame shall contain the sequence of octets that the application has requested the NWK layer to transmit.

3.2.2.1.3 Data frame NFR fields

The NFR for a data frame shall contain the message integrity code field if the security enabled sub-field of the frame control field is set to 1. If the security enabled sub-field of the frame control field is set to zero, the message integrity field shall be omitted from the frame.

3.2.2.1.4 Transmitting the data frame via the MAC sub-layer

A data frame shall be transmitted via the MAC sub-layer data service. This is instigated by the NLME issuing the MCPS-DATA.request primitive with the parameters listed in Table 35.

Table 35 – MCPS-DATA.request parameters for a data frame

Parameter	Setting
SrcAddrMode	0x03 (64-bit IEEE address) if the originator is a controller and the application requested a broadcast transmission, OR 0x02 (16-bit network address) otherwise.
DstAddrMode	0x02 (16-bit network address) if the application requested a broadcast transmission or (the recipient is a target and the application requested a transmission with a network address), OR 0x03 (64-bit IEEE address) if the recipient is a controller or the application requested a transmission with an IEEE address.
DstPANId	0xffff if the recipient is a controller or the application requested a broadcast transmission, OR The PAN identifier of the destination device from the pairing table if the recipient is a target.
DstAddr	0xffff if the application requested a broadcast transmission, OR The destination IEEE address from the pairing table if the recipient is a controller or the application requested a transmission with an IEEE address, OR The destination network address from the pairing table if the recipient is a target and the application requested a transmission with a network address.
msduLength	Size of the data or vendor specific frame
msdu	The data or vendor specific frame
msduHandle	Handle for this transmission
TxOptions	0b000 (no acknowledgment requested) if the application requested an unacknowledged or broadcast transmission, OR 0b001 (acknowledgment requested) if the application requested an acknowledged transmission.
Security	Disabled.

3.2.2.2 NWK command frame

The NWK command frame shall be formatted as illustrated in Figure 14.

Octets: 1	4	1	Variable	0/4
Frame control	Frame counter	Command identifier	Command payload	Message integrity code
NHR		NWK Payload		NFR

Figure 14 – NWK command frame format

The order of the fields of the data frame shall conform to the order of the general NWK frame as illustrated in Figure 11.

3.2.2.2.1 NWK command frame NHR fields

The NHR for a NWK command frame shall contain the frame control field and the frame counter field.

In the frame control field, the frame type shall contain the value that indicates a NWK command frame, as shown in Table 33. The channel designator sub-field shall be set to zero.

The frame counter field shall contain the current value of *nwkFrameCounter*.

3.2.2.2.2 Command identifier field

The command identifier field identifies the NWK command being used. This field shall be set to one of the non-reserved values listed in Table 36.

3.2.2.2.3 Command payload field

The command payload field contains the NWK command itself. The formats of the individual commands are described in 3.3.

3.2.2.2.4 NWK command frame NFR fields

The NFR for a NWK command frame shall contain the message integrity code field if the security enabled sub-field of the frame control field is set to 1. If the security enabled sub-field of the frame control field is set to zero, the message integrity field shall be omitted from the frame.

3.2.2.2.5 Transmitting the NWK command frame via the MAC sub-layer

A NWK command frame shall be transmitted via the MAC sub-layer data service. This is instigated by the NLME issuing the MCPS-DATA.request primitive with the parameters listed in each section describing the individual command frames.

3.3 NWK command frames

The command frames defined by the NWK layer are listed in Table 36.

Table 36 – NWK command frames

Command frame identifier	Command name	Sub-clause
0x01	Discovery request	3.3.1
0x02	Discovery response	3.3.2
0x03	Pair request	3.3.3
0x04	Pair response	3.3.4
0x05	Unpair request	3.3.5
0x06	Key seed	3.3.6
0x07	Ping request	3.3.7
0x08	Ping response	3.3.8
0x09 – 0xff	Reserved	-

3.3.1 Discovery request command frame

The discovery request command frame allows a device to discover devices operating in its POS.

This command can be directed to a specific device on a specific PAN, any device on a specific PAN or to any device.

The discovery request command frame shall be formatted as illustrated in Figure 15.

Octets: (see 3.2.2.2)	1	1	9	3-26	1
NHR fields	Command identifier (see Table 36)	Node capabilities	Vendor information fields (see Figure 16)	Application information fields (see Figure 17)	Requested device type

Figure 15 – Format of the discovery request command frame

Octets: 2	7
Vendor identifier	Vendor string

Figure 16 – Format of the vendor information fields

Octets: 1	0/15	1-3	1-7
Application capabilities	User string	Device type list	Profile identifier list

Figure 17 – Format of the application information fields

3.3.1.1 NHR fields

The channel designator sub-field of the frame control field shall be set to 0b00.

3.3.1.2 Node capabilities field

The node capabilities field is 1 octet in length and shall contain the capabilities of the node transmitting the discovery request command frame. This field shall be set to the value of *nwkcNodeCapabilities*.

3.3.1.3 Vendor identifier field

The vendor identifier field is 2 octets in length and shall contain the vendor identifier of the node transmitting the discovery request command frame. This field shall be set to the value of *nwkcVendorIdentifier*.

The set of vendor IDs is specified in [R2] .

3.3.1.4 Vendor string field

The vendor string field is 7 octets in length and shall contain the vendor string of the node transmitting the discovery request command frame. This field shall be set to the value of *nwkcVendorString*.

3.3.1.5 Application capabilities field

The application capabilities field is 1 octet in length and shall contain information relating to the capabilities of the application of the node sending the discovery request command frame. The application capabilities field shall be formatted as illustrated in Figure 18.

Bits: 0	**1-2**	**3**	**4-6**	**7**
User string specified	Number of supported device types	Reserved	Number of supported profiles	Reserved

Figure 18 – Format of the application capabilities field

3.3.1.5.1 User string specified sub-field

The user string specified sub-field is 1 bit in length and shall be set to one if a user string is to be included in the frame. Otherwise, it shall be set to zero. The user string field shall be included in the frame only if the user string specified sub-field is set to one.

3.3.1.5.2 Number of supported device types sub-field

The number of supported device types sub-field is 2-bits in length and shall contain the number of device types supported by the application of the node sending the discovery request command frame. This sub-field specifies the number of device types contained in the device type list field and shall be greater than zero.

3.3.1.5.3 Number of supported profiles sub-field

The number of supported profiles sub-field is 3-bits in length and shall contain the number of profiles disclosed as supported by the application of the node sending the discovery request command frame. This sub-field specifies the number of profile identifiers contained in the profile identifier list field and shall be greater than zero.

3.3.1.6 User string field

The user string field is 15 octets in length and shall contain the user specified identification string for the node sending the discovery request command frame. This field shall be included in the frame only if the user string specified sub-field of the application capabilities field is set to one.

3.3.1.7 Device type list field

The device type list field is 1-3 octets in length and shall contain the list of device types supported by the node sending the discovery request command frame. The length of this field shall be specified by the number of supported device types sub-field of the application capabilities field.

The set of supported device types is specified in [R3] .

3.3.1.8 Profile identifier list field

The profile identifier list field is 1-7 octets in length and shall contain the list of profile identifiers disclosed as supported by the node sending the discovery request command frame. The length of this field shall be specified by the number of supported profiles sub-field of the application capabilities field.

The set of supported profile identifiers is specified in [R4]

3.3.1.9 Requested device type field

The requested device type field is 1 octet in length and shall contain the type of device the application is attempting to discover.

If this value is set to 0xff, all devices that hear the command could respond with discovery response frames. Alternatively, if this value is set to a value other than 0xff, all devices that support that device type could respond with discovery response command frames.

3.3.1.10 Transmitting the frame via the MAC sub-layer

The discovery request command frame shall be transmitted via the MAC sub-layer data service. This is instigated by the NLME issuing the MCPS-DATA.request primitive with the parameters listed in Table 37.

Table 37 – MCPS-DATA.request parameters for a discovery request command frame

Parameter	**Setting**
SrcAddrMode	0x03 (64-bit IEEE address)
DstAddrMode	0x02 (16-bit network address)
DstPANId	0xffff or the PAN ID of the intended recipient
DstAddr	0xffff or the network address of the intended recipient
msduLength	Size of the discovery request command
msdu	The discovery request command
msduHandle	Handle for this transmission
TxOptions	0b000
Security	This frame shall be transmitted without MAC security

3.3.2 Discovery response command frame

The discovery response command frame allows a device that receives a discovery request command frame to respond with information that will allow the originating device to make a selection for pairing.

This command is unicast back to the originator of the discovery request command frame.

The discovery response command frame shall be formatted as illustrated in Figure 19.

Octets: (see 3.2.2.2)	**1**	**1**	**1**	**9**	**3-26**	**1**
NHR fields	Command identifier (see Table 36)	Status	Node capabilities	Vendor information fields (see Figure 16)	Application information fields (see Figure 17)	Discovery request LQI

Figure 19 – Format of the discovery response command frame

3.3.2.1 NHR fields

The channel designator sub-field of the frame control field shall be set to 0b00.

3.3.2.2 Status field

The status field is 1 octet in length and shall contain the result of the discovery attempt as reported by the device transmitting the discovery response command frame.

3.3.2.3 Node capabilities field

The node capabilities field is 1 octet in length and shall contain the capabilities of the node transmitting the discovery response command frame. This value shall be set to the value of *nwkcNodeCapabilities*.

3.3.2.4 Vendor identifier field

The vendor identifier field is 2 octets in length and shall contain the vendor identifier of the node transmitting the discovery response command frame. This field shall be set to the value of *nwkcVendorIdentifier*.

The set of vendor IDs is specified in [R2] .

3.3.2.5 Vendor string field

The vendor string field is 7 octets in length and shall contain the vendor string of the node transmitting the discovery response command frame. This field shall be set to the value of *nwkcVendorString*.

3.3.2.6 Application capabilities field

The application capabilities field is 1 octet in length and shall contain information relating to the capabilities of the application of the node sending the discovery response command frame. The application capabilities field shall be formatted as illustrated in Figure 18.

3.3.2.6.1 User string specified sub-field

The user string specified sub-field is 1 bit in length and shall be set to one if a user string is to be included in the frame. Otherwise, it shall be set to zero. The user string field shall be included in the frame only if the user string specified sub-field is set to one.

3.3.2.6.2 Number of supported device types sub-field

The number of supported device types sub-field is 2-bits in length and shall contain the number of device types supported by the application of the node sending the discovery response command frame. This sub-field specifies the number of device types contained in the device type list field and shall be greater than zero.

3.3.2.6.3 Number of supported profiles sub-field

The number of supported profiles sub-field is 3-bits in length and shall contain the number of profiles supported by the application of the node sending the discovery response command frame. This sub-field specifies the number of profile identifiers contained in the profile identifier list field and shall be greater than zero.

3.3.2.7 User string field

The user string field is 15 octets in length and shall contain the user specified identification string for the node sending the discovery response command frame. This field shall be included in the frame only if the user string specified sub-field of the application capabilities field is set to one.

3.3.2.8 Device type list field

The device type list field is 1-3 octets in length and shall contain the list of device types supported by the node sending the discovery response command frame. The length of this field shall be specified by the number of supported device types sub-field of the application capabilities field.

The set of supported device types is specified in [R3] .

3.3.2.9 Profile identifier list field

The profile identifier list field is 1-7 octets in length and shall contain the list of profile identifiers supported by the node sending the discovery response command frame. The length of this field shall be specified by the number of supported profiles sub-field of the application capabilities field.

The set of supported profile identifiers is specified in [R4]

3.3.2.10 Discovery request LQI field

The discovery request LQI field is 1 octet in length and shall contain the LQI measured by the PHY on reception of the original discovery request command frame and reported by the MAC sub-layer.

Note that in normal discovery mode, the LQI of the received discovery request command frame is passed to and from the application via the NLME-DISCOVERY.indication and NLME-DISCOVERY.response primitives, respectively, so that the NLME does not need to store the LQI value. Conversely, in auto discovery mode, the application is not involved and the NLME constructs this field as described.

3.3.2.11 Transmitting the frame via the MAC sub-layer

The discovery response command frame shall be transmitted via the MAC sub-layer data service. This is instigated by the NLME issuing the MCPS-DATA.request primitive with the parameters listed in Table 38.

Table 38 – MCPS-DATA.request parameters for a discovery response command frame

Parameter	**Setting**
SrcAddrMode	0x03 (64-bit IEEE address)
DstAddrMode	0x03 (64-bit IEEE address)
DstPANId	0xffff
DstAddr	IEEE address of originator
msduLength	Size of the discovery response command
msdu	The discovery response command
msduHandle	Handle for this transmission
TxOptions	0b001 (acknowledgment requested)
Security	This frame shall be transmitted without MAC security

3.3.3 Pair request command frame

The pair request command frame allows a device to request to pair with another device. Usually, this would be preceded by a discovery operation.

This command is unicast to the node to which a pairing link is required.

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

Octets: (see 3.2.2.2)	**1**	**2**	**1**	**9**	**3-26**	**1**
NHR fields	Command identifier (see Table 36)	Network address	Node capabilities	Vendor information fields (see Figure 16)	Application information fields (see Figure 17)	Key exchange transfer count

Figure 20 – Format of the pair request command frame

3.3.3.1 NHR fields

The channel designator sub-field of the frame control field shall contain the value corresponding to *nwkBaseChannel* if the node sending the pair request command frame is a target. Otherwise, this field shall be set to 0b00.

3.3.3.2 Network address field

The network address field is 2 octets in length and shall contain the network address (*macShortAddress*) of the node sending the pair request command frame if it is a target. Otherwise, this field shall be set to 0xfffe.

3.3.3.3 Node capabilities field

The node capabilities field is 1 octet in length and shall contain the capabilities of the node transmitting the pair request command frame. This field shall be set to the value of *nwkcNodeCapabilities*.

3.3.3.4 Vendor identifier field

The vendor identifier field is 2 octets in length and shall contain the vendor identifier of the node transmitting the pair request command frame. This field shall be set to the value of *nwkcVendorIdentifier*.

The set of vendor IDs is specified in [R2] .

3.3.3.5 Vendor string field

The vendor string field is 7 octets in length and shall contain the vendor string of the node transmitting the pair request command frame. This field shall be set to the value of *nwkcVendorString*.

3.3.3.6 Application capabilities field

The application capabilities field is 1 octet in length and shall contain information relating to the capabilities of the application of the node sending the pair request command frame. The application capabilities field shall be supplied by the application and shall be formatted as illustrated in Figure 18.

3.3.3.6.1 User string specified sub-field

The user string specified sub-field is 1 bit in length and shall be set to one if a user string is to be included in the frame. Otherwise, it shall be set to zero. The user string field shall be included in the frame only if the user string specified sub-field is set to one.

3.3.3.6.2 Number of supported device types sub-field

The number of supported device types sub-field is 2-bits in length and shall contain the number of device types supported by the application of the node sending the pair request command frame. This sub-field specifies the number of device types contained in the device type list field and shall be greater than zero.

3.3.3.6.3 Number of supported profiles sub-field

The number of supported profiles sub-field is 3-bits in length and shall contain the number of profiles supported by the application of the node sending the pair request command frame. This sub-field specifies the number of profile identifiers contained in the profile identifier list field and shall be greater than zero.

3.3.3.7 User string field

The user string field is 15 octets in length and shall contain the user specified identification string for the node sending the pair request command frame. This field shall be included in the frame only if the user string specified sub-field of the application capabilities field is set to one and shall be set to the value of *nwkUserString.*

3.3.3.8 Device type list field

The device type list field is 1-3 octets in length and shall contain the list of device types supported by the node sending the pair request command frame. The length of this field shall be specified by the number of supported device types sub-field of the application capabilities field. The device type list field shall be supplied by the application.

The set of supported device types is specified in [R3] .

3.3.3.9 Profile identifier list field

The profile identifier list field is 1-7 octets in length and shall contain the list of profile identifiers supported by the node sending the pair request command frame. The length of this field shall be specified by the number of supported profiles sub-field of the application capabilities field. The profile identifier list field shall be supplied by the application.

The set of supported profile identifiers is specified in [R4]

3.3.3.10 Key exchange transfer count field

The key exchange transfer count field is 1 octets in length and shall contain the number of transfers the originator wishes to use to exchange a security link key for the pairing link. This value shall be set in the range 0x00 – 0xff.

If security is not supported by the originator, this field shall be set to zero.

3.3.3.11 Transmitting the frame via the MAC sub-layer

If the pair request command frame is to be transmitted by a controller node, it shall first set *macPANId* to 0xffff.

The pair request command frame shall be transmitted via the MAC sub-layer data service. This is instigated by the NLME issuing the MCPS-DATA.request primitive with the parameters listed in Table 39.

Table 39 – MCPS-DATA.request parameters for a pair request command frame

Parameter	**Setting**
SrcAddrMode	0x03 (64-bit IEEE address)
DstAddrMode	0x03 (64-bit IEEE address)
DstPANId	PAN ID of the intended recipient
DstAddr	IEEE address of the intended recipient of this command
msduLength	Size of the pair request command
msdu	The pair request command
msduHandle	Handle for this transmission
TxOptions	0b001 (acknowledgement requested)
Security	This frame shall be transmitted without MAC security

3.3.4 Pair response command frame

The pair response command frame allows a device to respond to a pair request and pass information relevant to the pairing link back to the originator.

This command is unicast back to the originator of the pair request command frame.

The pair response command frame shall be formatted as illustrated in Figure 21.

Octets: **(see 3.2.2.2)**	**1**	**1**	**2**	**2**	**1**	**9**	**3-26**
NHR fields	Command identifier (see Table 36)	Status	Allocated network address	Network address	Node capabilities	Vendor information fields (see Figure 16)	Application information fields (see Figure 17)

Figure 21 – Format of the pair response command frame

3.3.4.1 NHR fields

The channel designator sub-field of the frame control field shall contain the value corresponding to *nwkBaseChannel*.

3.3.4.2 Status field

The status field is 1 octet in length and shall contain the result of the pairing attempt as reported by the device transmitting the pair response command frame.

3.3.4.3 Allocated network address field

The allocated network address field is 2 octets in length and shall contain a network address allocated by the node sending the pair response command frame if it is a target and the originator of the pair request command frame is a controller. Otherwise, this field shall be set to 0xfffe.

3.3.4.4 Network address field

The network address field is 2 octets in length and shall contain the network address of the node sending the pair response command frame.

3.3.4.5 Node capabilities field

The node capabilities field is 1 octet in length and shall contain the capabilities of the node transmitting the pair response command frame. This value shall be set to the value of *nwkcNodeCapabilities*.

3.3.4.6 Vendor identifier field

The vendor identifier field is 2 octets in length and shall contain the vendor identifier of the node transmitting the pair response command frame. This field shall be set to the value of *nwkcVendorIdentifier*.

The set of vendor IDs is specified in [R2] .

3.3.4.7 Vendor string field

The vendor string field is 7 octets in length and shall contain the vendor string of the node transmitting the pair response command frame. This field shall be set to the value of *nwkcVendorString*.

3.3.4.8 Application capabilities field

The application capabilities field is 1 octet in length and shall contain information relating to the capabilities of the application of the node sending the pair response command frame. The application capabilities field shall supplied by the application and shall be formatted as illustrated in Figure 18.

3.3.4.8.1 User string specified sub-field

The user string specified sub-field is 1 bit in length and shall be set to one if a user string is to be included in the frame. Otherwise, it shall be set to zero. The user string field shall be included in the frame only if the user string specified sub-field is set to one.

3.3.4.8.2 Number of supported device types sub-field

The number of supported device types sub-field is 2-bits in length and shall contain the number of device types supported by the application of the node sending the pair response command frame. This sub-field specifies the number of device types contained in the device type list field and shall be greater than zero.

3.3.4.8.3 Number of supported profiles sub-field

The number of supported profiles sub-field is 3-bits in length and shall contain the number of profiles supported by the application of the node sending the pair response command frame. This sub-field specifies the number of profile identifiers contained in the profile identifier list field and shall be greater than zero.

3.3.4.9 User string field

The user string field is 15 octets in length and shall contain the user specified identification string for the node sending the pair response command frame. This field shall be included in the frame only if the user string specified sub-field of the application capabilities field is set to one and shall be set to the value of *nwkUserString*.

3.3.4.10 Device type list field

The device type list field is 1-3 octets in length and shall contain the list of device types supported by the node sending the pair response command frame. The length of this field shall be specified by the number of supported device types sub-field of the application capabilities field. The device type list field shall be supplied by the application.

The set of supported device types is specified in [R3] .

3.3.4.11 Profile identifier list field

The profile identifier list field is 1-7 octets in length and shall contain the list of profile identifiers supported by the node sending the pair response command frame. The length of this field shall be specified by the number of supported profiles sub-field of the application capabilities field. The profile identifier list field shall be supplied by the application.

The set of supported profile identifiers is specified in [R4]

3.3.4.12 Transmitting the frame via the MAC sub-layer

The pair response command frame shall only be transmitted by a target node.

The pair response command frame shall be transmitted via the MAC sub-layer data service. This is instigated by the NLME issuing the MCPS-DATA.request primitive with the parameters listed in Table 40.

Table 40 – MCPS-DATA.request parameters for a pair response command frame

Parameter	**Setting**
SrcAddrMode	0x03 (64-bit IEEE address)
DstAddrMode	0x03 (64-bit IEEE address)
DstPANId	0xffff if the intended recipient is a controller or the PAN ID of the intended recipient otherwise.
DstAddr	IEEE address of intended recipient of this command
msduLength	Size of the pair response command
msdu	The pair response command
msduHandle	Handle for this transmission
TxOptions	0b001 (acknowledgment requested)
Security	This frame shall be transmitted without MAC security

3.3.5 Unpair request command frame

The unpair request command frame allows a device to request to remove a pairing link on a remote device.

This command is unicast to the node on which the pairing link is to be removed and shall be sent securely if a link key exists for the corresponding pairing link. Otherwise, this command shall be sent without security.

The unpair request command frame shall be formatted as illustrated in Figure 22.

Octets: **(see 3.2.2.2)**	**1**	**0/4**
NHR fields	Command identifier (see Table 36)	NFR fields

Figure 22 – Format of the unpair request command frame

3.3.5.1 NHR fields

The channel designator sub-field of the frame control field shall be set to 0b00.

3.3.5.2 NFR fields

The message integrity code field shall be included in the frame if a link key exists for the pairing link that is to be removed. Otherwise, the message integrity code field shall not be included in the frame.

3.3.5.3 Transmitting the frame via the MAC sub-layer

The unpair request command frame shall be transmitted via the MAC sub-layer data service. This is instigated by the NLME issuing the MCPS-DATA.request primitive with the parameters listed in Table 41.

Table 41 – MCPS-DATA.request parameters for an unpair request command frame

Parameter	**Setting**
SrcAddrMode	0x03 (64-bit IEEE address)
DstAddrMode	0x03 (64-bit IEEE address)
DstPANId	PAN ID from the pairing table if the intended recipient is a target or 0xffff if the intended recipient is a controller.
DstAddr	IEEE address of the intended recipient
msduLength	Size of the unpair request command
msdu	The unpair request command
msduHandle	Handle for this transmission
TxOptions	0b001 (acknowledgement requested)
Security	This frame shall be transmitted without MAC security

3.3.6 Key seed command frame

The key seed command frame allows a device to exchange security key seed values with a remote device in order to generate a security link key.

This command is unicast to a specific device and shall be sent with a transmit power of, at most, *nwkcMaxSecCmdTxPower*.

The key seed command frame shall be formatted as illustrated in Figure 23.

Octets: (see 3.2.2.2)	**1**	**1**	**80**
NHR fields	Command identifier (see Table 36)	Seed sequence number	Seed data

Figure 23 – Format of the key seed command frame

3.3.6.1 NHR fields

The channel designator sub-field of the frame control field shall be set to 0b00.

3.3.6.2 Seed sequence number field

The seed sequence number field is 1 octet in length and shall contain the sequence number of the seed contained in the seed data field.

3.3.6.3 Seed data field

The seed data field is 80 octets in length and shall contain the data for the seed identified by the seed sequence number field.

3.3.6.4 Transmitting the frame via the MAC sub-layer

The key seed command frame shall be transmitted via the MAC sub-layer data service. This is instigated by the NLME issuing the MCPS-DATA.request primitive with the parameters listed in Table 42.

Table 42 – MCPS-DATA.request parameters for an key seed command frame

Parameter	**Setting**
SrcAddrMode	0x03 (64-bit IEEE address)
DstAddrMode	0x03 (64-bit IEEE address)
DstPANId	0xffff
DstAddr	IEEE address of the intended recipient
msduLength	Size of the key seed command
msdu	The key seed command
msduHandle	Handle for this transmission
TxOptions	0b001 (acknowledgement requested)
Security	This frame shall be transmitted without MAC security

3.3.7 Ping request command frame

The ping request command frame allows a device to send a ping command frame to another device and get a response.

This command is unicast to a specific device on a specific PAN and shall always be sent securely.

The ping request command frame shall be formatted as illustrated in Figure 24.

Octets: (see 3.2.2.2)	1	1	Variable	4
NHR fields	Command identifier (see Table 36)	Ping options	Ping payload	NFR fields

Figure 24 – Format of the ping request command frame

3.3.7.1 NHR fields

The channel designator sub-field of the frame control field shall be set to 0x00.

3.3.7.2 Ping options field

The ping options field is 1 octet in length and shall contain options defined by the procedure in which the ping request command is used.

3.3.7.3 Ping payload field

The ping payload field has a variable length and shall contain a payload defined by the procedure in which the ping request command is used.

3.3.7.4 NFR fields

The message integrity code field shall always be included in the frame, generated using the security link key of the recipient.

3.3.7.5 Transmitting the frame via the MAC sub-layer

The ping request command frame shall be transmitted via the MAC sub-layer data service. This is instigated by the NLME issuing the MCPS-DATA.request primitive with the parameters listed in Table 42.

Table 43 – MCPS-DATA.request parameters for a ping request command frame

Parameter	**Setting**
SrcAddrMode	0x03 (64-bit IEEE address)
DstAddrMode	0x03 (64-bit IEEE address)
DstPANId	0xffff
DstAddr	IEEE address of the intended recipient
msduLength	Size of the ping request command
msdu	The ping request command
msduHandle	Handle for this transmission
TxOptions	0b001 (acknowledgement requested)
Security	This frame shall be transmitted without MAC security

3.3.8 Ping response command frame

The ping response command frame allows a device to respond to a ping request command frame from another device.

This command is unicast back to the originator of the ping request command frame and shall always be sent securely.

The ping response command frame shall be formatted as illustrated in Figure 25.

Octets: **(see 3.2.2.2)**	**1**	**1**	**Variable**	**4**
NHR fields	Command identifier (see Table 36)	Ping options	Ping payload	NFR fields

Figure 25 – Format of the ping response command frame

3.3.8.1 NHR fields

The channel designator sub-field of the frame control field shall be set to 0x00.

3.3.8.2 Ping options field

The ping options field is 1 octet in length and shall contain the value of the ping options field of the ping request command frame.

3.3.8.3 Ping payload field

The ping payload field has a variable length and shall contain the value of the ping payload field of the ping request command frame.

3.3.8.4 NFR fields

The message integrity code field shall always be included in the frame, generated using the security link key of the recipient.

3.3.8.5 Transmitting the frame via the MAC sub-layer

The ping response command frame shall be transmitted via the MAC sub-layer data service. This is instigated by the NLME issuing the MCPS-DATA.request primitive with the parameters listed in Table 44.

Table 44 – MCPS-DATA.request parameters for a ping response command frame

Parameter	**Setting**
SrcAddrMode	0x03 (64-bit IEEE address)
DstAddrMode	0x03 (64-bit IEEE address)
DstPANId	0xffff
DstAddr	IEEE address of the originator of the ping request
msduLength	Size of the ping response command
msdu	The ping response command
msduHandle	Handle for this transmission
TxOptions	0b001 (acknowledgement requested)
Security	This frame shall be transmitted without MAC security

3.4 NWK enumerations, constants and attributes

3.4.1 NWK enumerations

This sub-clause explains the meaning of the enumerations used in this specification. Table 45 shows a description of the NWK enumeration values. In some cases, primitives will carry results from the MAC sub-layer containing other enumeration values; these values are defined in [R1] .

Table 45 – NWK enumerations description

Value	Enumeration	Description
0x00	SUCCESS	The requested operation was completed successfully.
0xb0	NO_ORG_CAPACITY	A pairing link cannot be established since the originator node has reached its maximum number of entries in its pairing table.
0xb1	NO_REC_CAPACITY	A pairing link cannot be established since the recipient node has reached its maximum number of entries in its pairing table.
0xb2	NO_PAIRING	A pairing table entry could not be found that corresponds to the supplied pairing reference.
0xb3	NO_RESPONSE	A response frame was not received within *nwkResponseWaitTime*.
0xb4	NOT_PERMITTED	A pairing request was denied by the recipient node or an attempt to update a security link key was not possible due to one or more nodes not supporting security.
0xb5	DUPLICATE_PAIRING	A duplicate pairing table entry was detected following the receipt of a pairing request command frame.
0xb6	FRAME_COUNTER_EXPIRED	The frame counter has reached its maximum value.
0xb7	DISCOVERY_ERROR	Too many unique matched discovery request or valid response command frames were received than requested.
0xb8	DISCOVERY_TIMEOUT	No discovery request or response command frames were received during discovery.
0xb9	SECURITY_TIMEOUT	The security link key exchange or recovery procedure did not complete within the required time.
0xba	SECURITY_FAILURE	A security link key was not successfully established between both ends of a pairing link.
0xe8	INVALID_PARAMETER	A parameter in the primitive is either not supported or is out of the valid range.
0xf4	UNSUPPORTED_ATTRIBUTE	A SET/GET request was issued with the identifier of a NIB attribute that is not supported.
0xf9	INVALID_INDEX	An attempt to write to a NIB attribute that is in a table failed because the specified table index was out of range.

3.4.2 NWK constants

The constants that define the characteristics of the NWK layer are presented in Table 46.

Table 46 – NWK layer constants

Constant	Description	Value
nwkcChannelMask	The channel mask to use in all scanning operations.	See Table 47.
nwkcFrameCounterWindow	The amount that needs to be added to the frame counter if a device is reset.	1024
nwkcMACBeaconPayloadLength	The length, in octets, of the MAC beacon payload field, as used by the ZigBee RF4CE protocol.	2
nwkcMaxNodeDescListSize	The maximum number of entries supported in the node descriptors list generated through the discovery process.	Implementation specific but ≥ *nwkcMinNodeDescListSize*
nwkcMaxDutyCycle	The maximum duty cycle in MAC symbols, permitted for a power saving device.	62500 (1s)
nwkcMaxKeySeedWaitTime	The maximum time, in MAC symbols, to wait for each security link key seed exchange.	3750 (60ms) (see Equation 1)
nwkcMaxPairingTableEntries	The maximum number of entries supported in the pairing table.	Implementation specific
nwkcMaxSecCmdTxPower	The maximum acceptable power, in dBm, at which commands relating to security should be sent.	-15
nwkcMinActivePeriod	The minimum receiver on time, in MAC symbols, permitted for a power saving device.	1050 (16.8ms) (see Equation 2)
nwkcMinControllerPairingTableSize	The minimum number of pairing table entries that a controller device shall support.	1
nwkcMinNodeDescListSize	The minimum number of entries that must be supported in the node descriptor list generated through the discovery process.	3
nwkcMinNWKHeaderOverhead	The minimum overhead the NWK layer adds to an application payload	5
nwkcMinTargetPairingTableSize	The minimum number of pairing table entries that a target device shall support.	5

Constant	Description	Value
nwkcNodeCapabilities	The capabilities of this node.	Implementation specific according to the format illustrated in Figure 26.
nwkcProtocolIdentifier	The identifier of the NWK layer protocol being used by this device.	0xce
nwkcProtocolVersion	The version of the ZigBee RF4CE protocol implemented on this device.	0b01
nwkcVendorIdentifier	The manufacturer specific vendor identifier for this node.	Vendor ID corresponding to the device manufacturer
nwkcVendorString	The manufacturer specific identification string for this node.	Implementation specific format of 7 octets.

3.4.2.1 Maximum key exchange wait duration

The *nwkcMaxKeySeedWaitTime* constant shall be computed according to Equation 1.

Equation 1 – Computation for *nwkcMaxKeySeedWaitTime*

$$nwkcMaxKeySeedWaitTime = N_{at}*(T_{bo1} + T_{cca} + T_{cmd} + T_{ack} + T_{spt})$$

Where:

N_{at} Is the worst case number of NWK transmission attempts (3)

T_{bo1} Is the worst case 1st CSMA backoff period (140 symbols)

T_{cca} Is the time to perform a clear channel assessment (8 symbols)

T_{cmd} Is equal to the time to transmit a key seed command frame, assuming 64-bit addressing and no security (236 symbols)

T_{ack} Is equal to *macAckWaitDuration* (54 symbols)

T_{spt} Is the stack processing time (812 symbols)

3.4.2.2 Channel mask

The *nwkcChannelMask* constant shall contain the value illustrated in Table 47.

Table 47 – The value of the *nwkcChannelMask* constant

					b_{26}	**b_{25}**	**b_{24}**
					0	1	0
b_{23}	**b_{22}**	**b_{21}**	**b_{20}**	**b_{19}**	**b_{18}**	**b_{17}**	**b_{16}**
0	0	0	1	0	0	0	0
b_{15}	**b_{14}**	**b_{13}**	**b_{12}**	**b_{11}**	**b_{10}**	**b_9**	**b_8**
1	0	0	0	0	0	0	0
b_7	**b_6**	**b_5**	**b_4**	**b_3**	**b_2**	**b_1**	**b_0**
0	0	0	0	0	0	0	0

3.4.2.3 Minimum active period

The *nwkcMinActivePeriod* constant shall be computed according to Equation 2.

Equation 2 – Computation for *nwkcMinActivePeriod*

$$nwkcMinActivePeriod = N_{ch} * (T_{cmd} + T_{ack} + T_{ch} + T_{bo1} + T_{cca} + T_{spt}) + T_{cmd}$$

Where:

N_{ch}	Is equal to the number of channels being utilized (3).
T_{cmd}	Is equal to the time to receive a message that will wake up the node, assuming 16-bit addressing with security. (78 symbols)
T_{ack}	Is equal to *macAckWaitDuration* (54 symbols)
T_{ch}	Is equal to the time to switch channels (12 symbols)
T_{bo1}	Is the worst case 1st CSMA backoff period (140 symbols)
T_{cca}	Is the time to perform a clear channel assessment (8 symbols)
T_{spt}	Is the stack processing time (32 symbols)

3.4.2.4 Node capabilities

The *nwkcNodeCapabilities* constant shall be formatted as illustrated in Figure 26.

Bits: 0	1	2	3	4-7
Node type	Power source	Security capable	Channel normalization capable	Reserved

Figure 26 – Format of the *nwkcNodeCapabilities* constant

The node type sub-field is 1-bit in length and shall be set to one if the node is a target node. Otherwise, the node type sub-field shall be set to zero indicating a controller node.

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

The security capable sub-field is 1-bit in length and shall be set to one if the node is capable of sending and receiving cryptographically protected frames, as specified in [R1]. Otherwise, the security capable sub-field shall be set to zero.

The channel normalization sub-field is 1-bit in length and shall be set to one if the node will react to a channel change request through the channel designator sub-field of the frame control field. Otherwise, the channel normalization sub-field shall be set to zero.

3.4.3 NIB attributes

The NIB comprises attributes required to manage the NWK layer of a device. The attributes contained in the NIB are presented in Table 48.

Table 48 – NIB attributes

Attribute	Identifier	Type	Range	Description	Default
nwkActivePeriod	0x60	Integer	*nwkcMin-ActivePeriod* – 0xffffff	The active period of a device in MAC symbols.	0xffffff (268.44s)
nwkBaseChannel	0x61	Integer	15, 20 or 25	The logical channel that was chosen during device initialization.	15
nwkDiscoveryLQI-Threshold	0x62	Integer	0x00 – 0xff	The LQI threshold below which discovery requests will be rejected.	0xff
nwkDiscovery-RepetitionInterval	0x63	Integer	0x000000 – 0xffffff	The interval, in MAC symbols, at which discovery attempts are made on all channels.	0x0030d4 (200ms)
nwkDutyCycle	0x64	Integer	0x000000 – *nwkcMax-DutyCycle*	The duty cycle of a device in MAC symbols. A value of 0x000000 indicates the device is not using power saving.	0x000000
nwkFrameCounter	0x65	Integer	0x00000001 – 0xffffffff	The frame counter added to the transmitted NPDU.	0x00000001

Attribute	Identifier	Type	Range	Description	Default
nwkIndicateDiscovery-Requests	0x66	Boolean	TRUE or FALSE	Indicates whether the NLME indicates the reception of discovery request command frames to the application. TRUE indicates that the NLME notifies the application.	FALSE
nwkInPowerSave	0x67	Boolean	TRUE or FALSE	The power save mode of the node. TRUE indicates that the device is operating in power save mode.	FALSE
nwkPairingTable	0x68	List of pairing table entries (see Table 49)	Variable	The pairing table managed by the device.	Empty list.
nwkMaxDiscovery-Repetitions	0x69	Integer	0x01 – 0xff	The maximum number of discovery attempts made at the *nwkDiscovery-RepetitionInterval* rate.	0x01
nwkMaxFirstAttempt-CSMABackoffs	0x6a	Integer	0-5	The maximum number of backoffs the MAC CSMA-CA algorithm will attempt before declaring a channel access failure for the first transmission attempt.	4
nwkMaxFirstAttempt-FrameRetries	0x6b	Integer	0-7	The maximum number of MAC retries allowed after a transmission failure for the first transmission attempt.	3

Attribute	Identifier	Type	Range	Description	Default
nwkMaxReported-NodeDescriptors	0x6c	Integer	0x00 – *nwkcMaxNodeDescListSize*	The maximum number of node descriptors that can be obtained before reporting to the application.	0x03
nwkResponseWaitTime	0x6d	Integer	0x000000 – 0xffffff	The maximum time in MAC symbols, a device shall wait for a response command frame following a request command frame.	0x00186a (100ms)
nwkScanDuration	0x6e	Integer	0-14	A measure of the duration of a scanning operation, according to [R1].	6
nwkUserString	0x6f	Character string	15 characters	The user defined character string used to identify this node.	Empty string.

3.4.3.1 Pairing table

The *nwkPairingTable* attribute shall contain a list of *nwkcMaxPairingTableEntries* pairing table entries. Each entry of the pairing table shall be formatted as illustrated in Table 49.

Table 49 – Format of a pairing table entry

Field name	Type	Valid range	Description
Source network address	Integer	0x0000 – 0xfffe	The network address to be assumed by the source device.
Destination logical channel	Integer	15, 20 or 25	The expected channel of the destination device.
Destination IEEE address	64-bit IEEE address	A valid IEEE address	The IEEE address of the destination device.
Destination PAN identifier	Integer	0x0000 – 0xfffe	The PAN identifier of the destination device.
Destination network address	Integer	0x0000 – 0xfffe	The network address of the destination device.
Recipient capabilities	Bitmap	See Figure 26	The node capabilities of the recipient node.
Recipient frame counter	Integer	0x00000000 – 0xffffffff	The frame counter last received from the recipient node.
Security link key	128-bit key	A valid 128-bit key	The link key to be used to secure this pairing link.

3.5 Functional description

3.5.1 Frequency usage

3.5.1.1 Frequency channels

A ZigBee RF4CE device shall operate only at the 2.4GHz frequency band and shall utilize channels 15, 20 and 25. Channel switching operations shall use these channels in the sequence …15, 20, 25, 15, 20, etc., starting with the channel on which the destination was expected.

All scanning operations shall use the channel mask defined in *nwkcChannelMask*.

3.5.1.2 Frequency agility

All target devices shall be able to evaluate current channel conditions, according to some implementation specific mechanism. If the channel becomes compromised due to external sources of interference, the target device shall switch its base channel by changing the value of *nwkBaseChannel* and *phyCurrentChannel*.

All devices shall be able to re-acquire a device according to the mechanisms described in section 3.5.4.

3.5.1.3 Merging to a single frequency

If the application requires the use of unacknowledged or broadcast transmissions, it may attempt to merge its intended recipients onto its own channel. To do this, the application requests the transmission of data to include a channel designator, which gives a representation of its current channel, as stored in *nwkBaseChannel*.

The channel merging procedure shall only be used in outgoing data frames to those recipient nodes that support the feature (as specified by the channel normalization capable sub-field of the recipient capabilities field of the corresponding pairing table entry). Nodes that do not support this feature shall ignore the channel designator field on all incoming data frames.

The channel merging procedure shall only be used for acknowledged transmissions. If any other form of transmission is requested, the NLDE shall ignore channel merging and ensure the channel designator sub-field of the frame control field of the outgoing frame is set to 0b00.

On receipt of a request from the application to include a channel designator, the NLDE shall map *nwkBaseChannel* to the appropriate channel designator value (as listed in Table 34). This 2-bit value shall then be placed in the channel designator sub-field of the frame control field of the outgoing frame.

If the frame was successfully transmitted, the NLDE shall update the destination logical channel entry of the pairing table entry corresponding to the destination of the message with the value of *nwkBaseChannel*.

On receipt of a frame from the MAC sub-layer containing a non-zero channel designator, the NLDE of the recipient shall map the channel designator back to a logical channel number (as listed in Table 34). The NLDE shall then set *nwkBaseChannel* to this derived logical channel number and switch to that channel by setting *phyCurrentChannel*.

3.5.2 LQI mapping

The NLDE shall ensure that the PHY layer reports LQI as a measure of the RSSI of the incoming frame. The NLDE shall map the LQI values reported by the MAC sub-layer to the range 0x00 – 0xff, where 0x00 represents compliant signals detected at the receiver of -128dBm or lower and 0xff represents the compliant signals detected at the receiver of 0dBm or higher. LQI values in between this range shall be uniformly distributed between these two limits with a least 8 unique values of LQI being represented.

Note that a solution does not need to support sensitivities exactly in the range -128dBm to 0dBm but the range that is supported should be mapped as described above.

3.5.3 Addressing

Each node in an RC network shall contain a unique 64-bit IEEE address that can be exchanged, during the pairing procedure, for a 16-bit network address.

Each RC PAN shall utilize a 16-bit PAN identifier in the range 0x0000 – 0xfffe that shall be determined to be unique to the best of the ability of the target that starts the network.

A target shall allocate itself and all other devices that pair with it, a random network address on its chosen RC PAN in the range 0x0000 - 0xfffd. The target shall ensure that all network addresses that it allocates are unique on that RC PAN.

The PAN identifier 0xffff shall be used as the broadcast PAN identifier and network address 0xffff shall be used as the broadcast network address.

The ZigBee RF4CE protocol provides a mechanism to allow the use of manufacturer specific protocols by transmitting vendor specific data frames which contain a vendor identifier within the frame header.

3.5.4 Network topology

An RC PAN shall be composed of two types of device: a target node and a controller node. A target node has full PAN coordinator capabilities and shall start a network in its own right. A controller node (which could also be a target node) shall join networks started by target nodes by pairing with the target. Multiple RC PANs form an RC network and nodes can communicate between RC PANs.

A target node shall be exclusively responsible for its own RC PAN. The target node shall allocate and ensure the uniqueness of a PAN identifier for its network. A target node shall allow other devices (which may be either controllers or targets) to pair with it in order to carry out the appropriate communication operations.

A node can be paired with many other nodes and many nodes can be paired to any one node, subject to the capacity of the pairing table of each respective node.

Inter-PAN communication is used to communicate from one RC PAN to another. In order to communicate, the transmitting node first switches to the channel and assumes the PAN identifier of the destination RC PAN. It then uses the network address, allocated through the pairing procedure, to identify itself on the RC PAN and thus communicate with the desired recipient node.

Figure 27 illustrates an example ZigBee RF4CE topology which includes three target nodes: a TV, a DVD and a CD player and each target node creates its own RC PAN. The TV, DVD and CD player also have dedicated RCs which are paired to each appropriate target node. A multi-function RC, capable of controlling all three target nodes itself, is added to the network by successively pairing to the desired target nodes. The DVD is also paired with the TV so that an external channel can be selected on the TV when a DVD is played.

As a consequence, this RC network consists of three separate RC PANs: one managed by the TV, containing the TV RC, the multi-function RC and the DVD; a second managed by the DVD, containing the DVD RC, multi-function RC and the TV, and a third managed by the CD player, containing the CD RC and the multi-function RC.

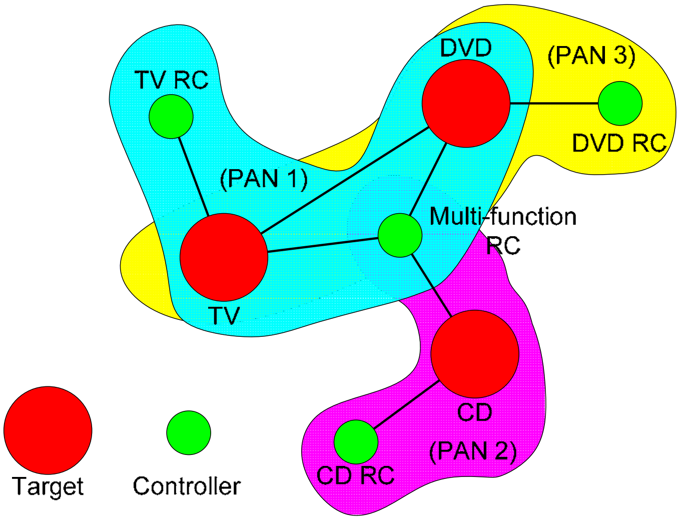


Figure 27 – Example ZigBee RF4CE network topology

3.5.5 Node initialization

All nodes shall be able to detect whether they are performing a cold start (e.g. the first startup outside the factory) or a warm start (e.g. after a battery replacement). The procedure for detecting whether to perform a cold or warm start is implementation specific and out of the scope of this specification.

Each node shall be able to store the NIB attributes, which includes the pairing table, in non volatile memory so that when a warm start is performed, their values are preserved.

3.5.5.1 Node cold start procedure

A node shall perform a cold start by first performing a cold start reset of the NWK layer and then performing the startup procedure.

The application instigates a NWK layer cold start reset by issuing the NLME-RESET.request primitive with the SetDefaultNIB parameter set to TRUE and the NLME responds by issuing the NLME-RESET.confirm primitive.

On receipt of a reset confirmation from the MAC sub-layer, the application shall instigate the startup procedure. The startup procedure is instigated by issuing the NLME-START.request primitive and the NLME responds by issuing the NLME-START.confirm primitive.

On receipt of a request for node startup, the NLME shall first configure the MAC sub-layer. In order to operate according to the ZigBee RF4CE protocol, selected MAC attributes shall be configured according to the information listed in Table 50. All other MAC attributes shall be set to their default values.

Table 50 – Initial MAC attribute settings

MAC attribute	Identifier	Initial value
macAssociationPermit	0x41	FALSE
macMaxCSMABackoffs	0x4e	0
macMaxFrameRetries	0x59	0

If the node is a controller, indicated by the node type sub-field of *nwkcNodeCapabilities* being equal to zero, the NLME shall then notify the application of the status of the startup and the node shall enter normal operation and perform no further initialization processing.

If the node is a target, indicated by the node type sub-field of *nwkcNodeCapabilities* being equal to one, the NLME shall attempt to start a new network by first performing an energy detection scan, to select a suitable channel on which to operate, and then performing an active scan, to allow the node to select a unique PAN identifier.

The NLME shall request that the MAC sub-layer performs an energy detect scan by issuing the MLME-SCAN.request primitive to the MLME. In this primitive, the ScanChannels parameter shall be equal to *nwkcChannelMask* and the ScanDuration parameter shall be equal to *nwkScanDuration*. On receipt of the MLME-SCAN.confirm primitive, the NLME shall set *phyCurrentChannel* and *nwkBaseChannel* to the channel with the least detected energy.

The NLME shall then request that the MAC sub-layer performs an active scan by issuing the MLME-SCAN.request primitive to the MLME. In this primitive, the ScanChannels parameter shall be equal to *nwkcChannelMask* and the ScanDuration parameter shall be set to *nwkScanDuration*. On receipt of the MLME-SCAN.confirm primitive, the NLME shall choose a PAN identifier that is unique across the PANs detected during the active scan and shall set *macPANId* to that value. The device shall then randomly select (in the permitted range) a network address and shall set *macShortAddress* to that value.

On completion of the two scans, the NLME shall then notify the application of the status of the startup and the node shall enter normal operation.

3.5.5.2 Node warm start procedure

A node shall perform a warm start by performing a warm start reset of the NWK layer.

The application instigates a NWK layer warm start reset by issuing the NLME-RESET.request primitive with the SetDefaultNIB parameter set to FALSE and the NLME responds by issuing the NLME-RESET.confirm primitive.

On receipt of a reset confirmation from the MAC sub-layer, the NLME shall then increment *nwkFrameCounter* by *nwkcFrameCounterWindow*.

The node shall then enter normal operation.

3.5.6 Use of the MAC beacon payload

The NLME of a target node shall additionally configure the MAC beacon payload so that the RC PAN can be identified from other IEEE 802.15.4 PANs. To do this, the NLME shall set *macBeaconPayloadLength* to *nwkcMACBeaconPayloadLength* and *macBeaconPayload* to the byte string specified in Figure 28.

Octets: 1	**1**
Protocol identifier	Protocol version

Figure 28 – Format of the MAC beacon payload

3.5.6.1 Protocol identifier field

The protocol identifier field is 8-bits in length and specifies the identifier of the network protocol in use. This field shall always be set to *nwkcProtocolIdentifier*.

3.5.6.2 Protocol version field

The protocol version field is 8-bits in length and specifies the version of the ZigBee RF4CE protocol. This field shall always be set to *nwkcProtocolVersion*.

3.5.7 Power saving

The ZigBee RF4CE power saving mechanism allows a node to conserve power by sleeping for extended periods while still permitting other nodes to communicate with it. To accomplish this, the power saving node periodically wakes, enables its receiver for a short period, handles any incoming frames and then returns to its sleep state.

A node shall be assumed to be in power save mode when *nwkInPowerSave* is equal to TRUE and the node shall be assumed to not be in power save mode when *nwkInPowerSave* is equal to FALSE. While in power save mode, the node shall not respond to any NWK command frames.

Power saving depends on the values of *nwkDutyCycle* and *nwkActivePeriod*. The *nwkDutyCycle* attribute shall define the length of time between successive periods of activity and the *nwkActivePeriod* attribute shall define the length of time during which the node enables its receiver within each duty cycle. These concepts are illustrated in Figure 29.

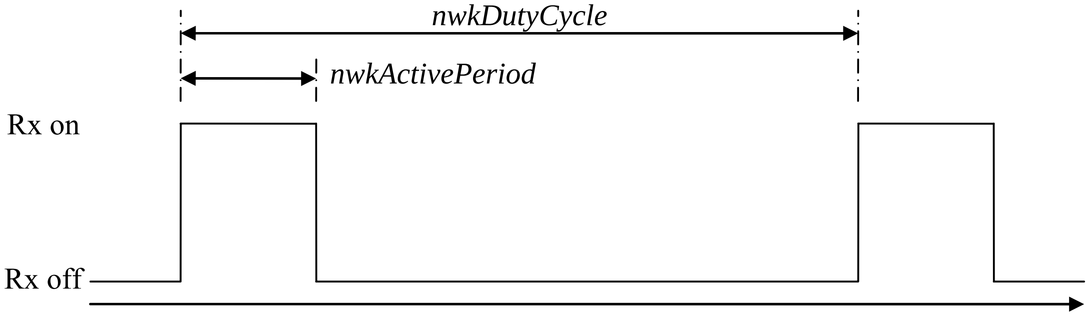


Figure 29 – Power saving mechanism concepts

If *nwkDutyCycle* is set to 0x000000, the NWK layer shall not utilise or stop using power saving and the application must control the operation of the receiver directly by issuing the NLME-RX-ENABLE.request primitive.

If *nwkDutyCycle* is not set to 0x000000, the application may enter or exit power saving by manipulation of the receiver via the NLME-RX-ENABLE.request primitive. A request to enable the receiver shall temporarily disable power saving for the duration of the requested receiver on time. A request to disable the receiver shall enable power saving.

From the point power saving is enabled to the point it is disabled, the NLME shall repeatedly manipulate the receiver, as illustrated in Figure 29, i.e. for each duty cycle, the receiver shall be enabled for *nwkActivePeriod* symbols and disabled for the (*nwkDutyCycle* – *nwkActivePeriod*) symbols.

If any frames are received during the active period, the NLDE shall process the frames accordingly and then return to its regular duty cycling. If the active period expires while processing an incoming frame received during the active period, the node shall fully process the frame before going back to sleep.

3.5.8 Transmission, reception and acknowledgement

Data transmissions from the application are instigated by issuing the NLDE-DATA.request primitive to the NLDE along with the options required for transmission. The NLDE responds by issuing the NLDE-DATA.confirm primitive with an appropriate status value, which may indicate an error.

The application is notified of the reception of data via the NLDE-DATA.indication primitive.

3.5.8.1 Transmission - general

On receipt of a request to transmit a data, network command or vendor specific data frame, the NLDE shall attempt to generate an NPDU to transport to the recipient via the MAC sub-layer data service. The construction of the NPDU is dependent on the type of frame being transmitted.

The ZigBee RF4CE protocol provides the following transmission services:

- Single channel, acknowledged unicast
- Multiple channel, acknowledged unicast
- Single channel, unacknowledged unicast
- Multiple channel, unacknowledged unicast
- Single channel, broadcast
- Multiple channel, broadcast

Each of these services is discussed in the following sub-clauses.

If the originator is a controller and the recipient is a target, the device shall set *macPANId* and *macShortAddress* to the source network address and destination PAN identifier fields from the corresponding pairing table entry before requesting a transmission via the MAC sub-layer. Otherwise, *macPANId* and *macShortAddress* shall be left unchanged.

Each node shall store its current frame counter value in the NIB attribute *nwkFrameCounter* and initialize it to 0x00000001. Each time a data, network command or vendor specific data frame, is generated, the NLDE shall copy the value of *nwkFrameCounter* into the frame counter field of the NHR of the outgoing frame and then increment it by one. Note that the frame counter shall only be incremented when a frame is *generated* and not each time a transmission is made during each transmission service attempt.

If a transmission request is made with *nwkFrameCounter* equal to its maximum value (0xffffffff), the NLDE shall notify the application with an error code of FRAME_COUNTER_EXPIRED and perform no further processing.

3.5.8.2 Acknowledged unicast transmission service

The acknowledged unicast transmission service allows a node to transmit a data or vendor specific data frame to another node while requesting an acknowledgement. The acknowledged unicast transmission service can only be used to transmit frames to other nodes that have entries in the current pairing table of the originating node. A transmission is attempted either on a single channel or on all channels, and an acknowledgement will be requested. Single channel acknowledged unicast transmissions are attempted only on the expected channel up to a fixed number of retries until an acknowledgement is received from the recipient. Multiple channel acknowledged unicast transmissions are attempted on the expected channel for a fixed number of retries and then on each channel repeatedly for a finite duty cycle until an acknowledgement is received from the recipient.

On receipt of a request for an acknowledged unicast transmission, the NLDE shall attempt to find the entry in the pairing table that corresponds to the recipient node. If the NLDE does not find a corresponding entry in the pairing table or if the corresponding entry is provisional, it shall return a status code of NO_PAIRING. If an entry for the requested recipient exists in the pairing table, the NLDE shall construct the NPDU from the information supplied. The NLDE shall then set *macMaxCSMABackoffs* and *macMaxFrameRetries* to the values of *nwkMaxFirstAttemptCSMABackoffs* and *nwkMaxFirstAttemptFrameRetries*, respectively.

The NLDE shall then request the MAC sub-layer change the value of *phyCurrentChannel* to the destination logical channel field of the corresponding entry in the pairing table. Controller nodes, in addition, shall request the MAC sub-layer change the values of *macPANId* and *macShortAddress* to the destination PAN identifier and source network address fields of the corresponding entry in the pairing table.

The NLDE shall then attempt to transmit the frame to the destination by passing the created NPDU to the MAC sub-layer. To do this, the NLDE issues the MCPS-DATA.request primitive to the MAC sub-layer (for data frames, see 3.2.2.1.4 and for network command frames, see the description of individual frames in 3.3). The MAC sub-layer confirms the status of the transmission attempt by issuing the MCPS-DATA.confirm to the NLDE.

If a single channel acknowledged unicast transmission was requested, when the MAC sub-layer confirms the transmission the NLDE shall pass the status code, indicated by the MAC sub-layer transmission confirmation, to the application by issuing the NLDE-DATA.confirm primitive. The NLDE shall then terminate the transmission procedure.

If a multiple channel acknowledged unicast transmission was requested and the MAC sub-layer confirms an unsuccessful transmission, the NLDE shall set both *macMaxCSMABackoffs* and *macMaxFrameRetries* to zero. The NLDE shall then request the MAC sub-layer change the value of *phyCurrentChannel* to the next channel in the sequence and the acknowledged unicast transmission attempt shall be made again. The acknowledged unicast transmission attempt shall be made on each available channel, repeated either for *nwkcMaxDutyCycle* symbols or until an acknowledgement is received, whichever is sooner.

If an acknowledgement is received from the recipient within *nwkcMaxDutyCycle* symbols, the NLDE shall notify the application with a status code of SUCCESS. The NLDE shall then terminate the transmission procedure.

If an acknowledgement is not received from the recipient after *nwkcMaxDutyCycle* symbols have elapsed, the NLDE shall notify the application with a status code of NO_RESPONSE. The NLDE shall then terminate the transmission procedure.

If the transmission attempt was successful and the logical channel on which the transmission was made is different from that stored in the pairing table, the destination logical channel field of the corresponding entry in the pairing table shall be updated with the channel on which the recipient node responded.

For a target node, once the transmission has been attempted, the NLDE shall request the MAC sub-layer change the value of *phyCurrentChannel* to *nwkBaseChannel*.

3.5.8.3 Unacknowledged unicast transmission service

The unacknowledged unicast transmission service allows a node to transmit a data or vendor specific data frame to another node but without requesting an acknowledgement. The unacknowledged unicast transmission service can only be used to transmit frames to other nodes that have entries in the current pairing table of the originating node. Each transmission is attempted once, either on a single channel or on all channels, and an acknowledgement will not be requested.

On receipt of a request for an unacknowledged unicast transmission from the application, the NLDE shall attempt to find the entry in the pairing table that corresponds to the recipient node. If the NLDE does not find a corresponding entry in the pairing table or if the corresponding entry is provisional, it shall return a status code of NO_PAIRING to the application. If an entry for the requested recipient exists in the pairing table, the NLDE shall construct the NPDU from the information supplied by the application. The NLDE shall then set *macMaxCSMABackoffs* to the value of *nwkMaxFirstAttemptCSMABackoffs*.

The NLDE shall then request the MAC sub-layer change the value of *phyCurrentChannel* to the destination logical channel field of the corresponding entry in the pairing table. Controller nodes, in addition, shall request the MAC sub-layer change the values of *macPANId* and *macShortAddress* to the destination PAN identifier and source network address fields of the corresponding entry in the pairing table.

The NLDE shall then attempt to transmit the frame to the destination by passing the created NPDU to the MAC sub-layer. To do this, the NLDE issues the MCPS-DATA.request primitive to the MAC sub-layer (see 3.2.2.1.4). The MAC sub-layer confirms the status of the transmission attempt by issuing the MCPS-DATA.confirm to the NLDE.

If a single channel unacknowledged unicast transmission was requested, when the MAC sub-layer confirms the transmission, the NLDE shall pass the status code, indicated by the MAC sub-layer transmission confirmation, to the application by issuing the NLDE-DATA.confirm primitive. The NLDE shall then terminate the transmission procedure.

If a multiple channel unacknowledged unicast transmission was requested, when the MAC sub-layer confirms the transmission, the NLDE shall request the MAC sub-layer change the value of *phyCurrentChannel* to the next channel in the sequence and the unacknowledged unicast transmission attempt shall be made again. An unacknowledged unicast transmission attempt shall be repeated once on each channel until the transmission has been attempted on all available channels. The NLDE shall then notify the application with a status code of SUCCESS and the NLDE shall terminate the transmission procedure.

For a target node, once the transmission has been attempted, the NLDE shall request the MAC sub-layer change the value of *phyCurrentChannel* to *nwkBaseChannel*.

3.5.8.4 Broadcast transmission service

The broadcast transmission service allows a node to transmit a data or vendor specific data frame to all other nodes in its POS. Each transmission is attempted once, either on a single channel or on all channels, and an acknowledgement will not be requested.

On receipt of a request for a broadcast transmission from the application or the NLME, the NLDE shall construct the NPDU from the information supplied and appropriate for the individual frame type. The NLDE shall then set *macMaxCSMABackoffs* to the value of *nwkMaxFirstAttemptCSMABackoffs*. In addition, the NLDE shall request the MAC sub-layer change the value of *phyCurrentChannel* to the channel specified by *nwkBaseChannel*.

The NLDE shall then attempt to transmit the frame by passing the created NPDU to the MAC sub-layer. To do this, the NLDE issues the MCPS-DATA.request primitive to the MAC sub-layer (for data frames, see 3.2.2.1.4 and for network command frames, see the description of individual frames in 3.3). The MAC sub-layer confirms the status of the transmission attempt by issuing the MCPS-DATA.confirm to the NLDE.

If a single channel broadcast transmission was requested, when the MAC sub-layer confirms the transmission, the NLDE shall pass the status code, indicated by the MAC sub-layer transmission confirmation, to the application or the NLME by issuing the appropriate primitives. The NLDE shall then terminate the transmission procedure.

If a multiple channel broadcast transmission was requested, when the MAC sub-layer confirms the transmission, the NLDE shall request the MAC sub-layer change the value of *phyCurrentChannel* to the next channel in the sequence and the broadcast transmission attempt shall be made again. A broadcast transmission attempt shall be repeated once on each channel until the transmission has been attempted on all available channels. The NLDE shall then notify the application with a status code of SUCCESS and the NLDE shall terminate the transmission procedure.

3.5.8.5 Reception and rejection

The NLDE receives only those frames passed to it by the MAC sub-layer data service, which applies an incoming frame filter (see [R1]), restricting the number of frames delivered to the NWK layer. Similarly, the NLDE shall also apply an incoming frame filter as follows.

On receipt of a frame from the MAC sub-layer, the NLDE shall accept only frames that satisfy all of the following first-level filtering requirements (as indicated by the MAC):

- If the source PAN identifier is not equal to 0xffff, it shall match the destination PAN identifier field in one of the entries in the pairing table.
- If the source address is indicated as being a network address, it shall match the destination network address field in one of the entries in the pairing table.
- If the source address is indicated as being an IEEE address, it shall match the destination IEEE address field in one of the entries in the pairing table.

If any of the first-level filtering requirements are not satisfied, the NLDE shall discard the frame without processing it further. If all of the first-level filtering requirements are satisfied, the NLDE shall accept only frames that satisfy all of the second-level filtering requirements (NWK header):

- The frame type sub-field shall not contain a reserved frame type.
- If the security enabled sub-field of the frame control field indicates the frame is secured, the security processing (authentication) of the incoming frame shall be successful.
- The frame counter field shall be greater than the recipient frame counter stored in the corresponding entry of the pairing table entry.

If any of the second-level filtering requirements are not satisfied, the NLDE shall discard the frame without processing it further.

If all of the second-level filtering requirements are satisfied and the frame was secured over a secure pairing link or not secured over an insecure pairing link, the NLDE shall replace the recipient frame counter field of the corresponding entry in the pairing table with the value of the frame counter field in the incoming frame. Otherwise, the recipient frame counter field of the corresponding pairing table entry shall be left unchanged.

The NLDE shall then notify the application of the incoming frame.

3.5.9 Discovering services

In order to discover relevant services in the POS of a device, the discovery procedure can be used. Discovery can be used as a pre-requisite to the pairing procedure since it provides the information necessary for creating a pairing link.

The NLME can be configured to indicate to the application, never process or to automatically respond to any received discovery request command frames.

The result of a discovery operation is a list of node descriptors giving information relating to those nodes that responded to the discovery request. A node shall be able to store between *nwkcMinNodeDescListSize* and *nwkcMaxNodeDescListSize* node descriptors.

3.5.9.1 Discovery originator procedure

For the node performing discovery (the originator) the discovery procedure is initiated by the application by issuing the NLME-DISCOVERY.request primitive and its results transferred to the application via the NLME-DISCOVERY.confirm primitive.

A discovery request can be made to all nodes (by setting the destination PAN identifier and destination address both to 0xffff), to nodes on a specific RC PAN (by setting the destination PAN identifier and destination address to the identifier of the desired RC PAN and 0xffff, respectively) or to a specific node (by setting the destination PAN identifier and destination address to the desired values). In addition, a node can discover a specific type of device or any type of device.

On receipt of a request to perform discovery, the NLME shall first generate a discovery request command frame (see 3.3.1) with the information supplied by the application. The node requesting discovery shall specify information about itself in order to give sufficient information to a recipient in order to allow it to decide whether to respond. The originator of the discovery request shall specify its node capabilities, its vendor identifier, a vendor specific identification string, an optional user specific string relating to this node, a list of supported device types and the list of supported profile identifiers. In addition, the originator specifies the device type it is attempting to discover.

If the node performing discovery is a controller, it shall ensure *macPANId* is set to 0xffff before commencing transmission. For each available logical channel, the NLME shall switch to the channel, by setting *phyCurrentChannel*, and then transmit the discovery request command frame via the MAC sub-layer data service; this transmission shall be unicast if the discovery is directed to a specific device or broadcast otherwise. If the MAC sub-layer confirms the successful transmission of the discovery request command frame, the NLME shall request that the MAC sub-layer enable the receiver for the application specified discovery duration (measured in MAC symbols). During this time, the NLME shall reject all command frames except discovery response command frames. The transmission of a discovery request command frame on all available channels is called a discovery trial.

A discovery trial shall be performed either *nwkMaxDiscoveryRepetitions* times, repeated at intervals of *nwkDiscoveryRepetitionInterval*, or until the discovery is complete, whichever is sooner.

During the discovery trial and on receipt of each unique discovery response command frame, the NLME shall check that at least one device type contained in the device type list field matches the search device type supplied by the application and that at least one profile identifier contained in the profile identifier list field matches at least one profile identifier from the discovery profile identifier list supplied by the application. If a match is found, the NLME shall record the information contained in the discovery response command frames in a node descriptor structure (see Table 13). If a match is not found, the NLME shall discard the discovery response command frame.

If the number of node descriptors stored at the end of any single discovery trial is equal to *nwkMaxReportedNodeDescriptors*, the NLME shall notify the application of the list of node descriptors discovered for the devices from which discovery response commands were received and a status code of SUCCESS.

If the number of node descriptors stored at the end of any single discovery trial exceeds *nwkMaxReportedNodeDescriptors*, the NLME shall notify the application with a status code of DISCOVERY_ERROR.

If, at any time during the discovery process, the number of node descriptors stored is equal to *nwkcMaxNodeDescListSize*, the NLME shall notify the application of the list of node descriptors discovered for the devices from which discovery response commands were received and a status code of SUCCESS.

If *nwkMaxDiscoveryRepetitions* discovery trials have been made and the procedure has not yet been terminated as described above, the NLME shall notify the application of the list of node descriptors discovered for the devices from which discovery response commands were received and a status code of SUCCESS.

If, at the end of *nwkMaxDiscoveryRepetitions* discovery trials, no node descriptors are stored, the NLME shall notify the application with a status code of DISCOVERY_TIMEOUT.

When notifying the application of the completion of the discovery procedure, the NLME shall also indicate to the application whether all of the discovery trials were complete.

3.5.9.2 Discovery recipient procedure

For the node receiving a discovery request (the recipient), the application is notified of the reception of a discovery request command frame via the NLME-DISCOVERY.indication primitive, provided *nwkIndicateDiscoveryRequests* is set to TRUE, and the application responds by issuing the NLME-DISCOVERY.response primitive to the NLME. The application is notified of the status of the transmission of the subsequent discovery response command frame via the NLME-COMM-STATUS.indication primitive.

On receipt of a discovery request command frame (see 3.3.1), if *nwkInPowerSave* is equal to TRUE or if *nwkIndicateDiscoveryRequests* is equal to FALSE, the NLME shall discard the frame and perform no further processing.

On receipt of a discovery request command frame when *nwkInPowerSave* of the node is equal to FALSE and *nwkIndicateDiscoveryRequests* is equal to TRUE, the NLME shall first check the LQI of the received frame, as reported by the MAC sub-layer data service. If this value is less than *nwkDiscoveryLQIThreshold*, the NLME shall reject the discovery request command frame and perform no further processing. If the LQI of the received discovery request command frame is greater than or equal to *nwkDiscoveryLQIThreshold*, the NLME shall notify the application with the information contained in the frame along with a status of SUCCESS if capacity is available in its pairing table for another pairing link or NO_REC_CAPACITY otherwise.

If the application chooses to accept the discovery request, the NLME generates a discovery response command frame (see 3.3.2) with the information supplied from the application. The node responding to the discovery request shall specify information about itself in order to give sufficient information to the originator in order to allow it to decide whether to attempt to pair with this node. The recipient of the discovery request shall specify its node capabilities, its vendor identifier, a vendor specific identification string, an optional user specific string relating to this node, a list of supported device types and the list of supported profile identifiers.

The NLME shall then set *macMaxCSMABackoffs* and *macMaxFrameRetries* to the values of *nwkMaxFirstAttemptCSMABackoffs* and *nwkMaxFirstAttemptFrameRetries*, respectively.

The NLME shall then transmit this frame to the originator of the discovery request via the MAC sub-layer data service; this transmission shall be unicast back to the originator. The NLME shall then notify the application of the status of the transmission.

3.5.9.3 Automatic discovery response

Automatic discovery response mode is initiated by the application by issuing the NLME-AUTO-DISCOVERY.request primitive and its results transferred to the application via the NLME-AUTO-DISCOVERY.confirm primitive. During automatic discovery response mode, *nwkIndicateDiscoveryRequests* shall be ignored.

On receipt of a request to enter automatic discovery response mode, the NLME waits to receive discovery request command frames for at most the specified automatic discovery response mode duration. During this time, the NLME shall discard any non discovery request command frames that are received.

On receipt of a discovery request command frame, the NLME shall first check the LQI of the received frame, as reported by the MAC sub-layer data service. If this value is less than *nwkDiscoveryLQIThreshold*, the NLME shall reject the discovery request command frame. If the LQI of the received discovery request command frame is greater than or equal to *nwkDiscoveryLQIThreshold*, the NLME shall check that the requested device type field matches one of the device types supplied by the application and that at least one of the profile identifiers listed in the profile identifier list field matches one of the profile identifiers supplied by the application. If a match is found, the NLME shall wait for the reception of a second identical discovery request command frame from the same node and shall again check for a match.

If the second discovery request command frame matches the information supplied by the application and is identical to the first discovery request command frame, the NLME shall generate a discovery response command frame (see 3.3.2) with the information supplied from the application. The node responding to the discovery request shall specify information about itself in order to give sufficient information to the originator in order to allow it to decide whether to attempt to pair with this node. The recipient of the discovery request shall specify its node capabilities, its vendor identifier, a vendor specific identification string, an optional user specific string relating to this node, a list of supported device types and the list of supported profile identifiers.

The NLME shall then transmit this frame to the originator of the discovery request via the MAC sub-layer data service; this transmission shall be unicast back to the originator. The NLME shall then disable automatic discovery response mode and notify the application with a status of SUCCESS.

If a second discovery request command frame matches the information supplied by the application but is received from a different node, the NLME shall disable automatic discovery response mode and notify the application with a status of DISCOVERY_ERROR.

If a discovery request command frame is received that does not match the information supplied by the application, the NLME shall discard the frame and remains in automatic discovery mode.

If less than two matching discovery request command frames are received after the requested duration, the NLME shall disable automatic discovery response mode and notify the application with a status of DISCOVERY_TIMEOUT.

3.5.10 Pairing devices

Each node shall maintain a pairing table to track the nodes with which it may communicate. Pairing links shall be bi-directional so an entry shall be created on both the originator (to the recipient) and the recipient (back to the originator). This allows the NLDE to filter incoming frames from nodes that do not appear in the pairing table.

A target node shall be able to store at least *nwkcMinTargetPairingTableSize* entries and a controller node shall be able to store at least *nwkcMinControllerPairing-TableSize* entries. The pairing table shall be ordered according to a pairing reference that is communicated to the application during pairing. This allows the application to transmit its data using a simple reference rather than having to supply the addressing information required by the MAC sub-layer data service.

Pairing shall only be permitted if instigated from a controller to a target or from a target to a target.

3.5.10.1 Pairing originator procedure

For the node performing pairing (the originator) the pairing procedure is initiated by the application by issuing the NLME-PAIR.request primitive to the NLME and its results and status are transferred to the application via the NLME-PAIR.confirm primitive.

On receipt of a pairing request, the NLME shall first check whether it already has an entry in its pairing table corresponding to the pairing recipient. If an entry already exists in the pairing table, the NLME shall store the pairing reference of this entry and update it rather than creating a new entry. If an entry does not already exist in the pairing table, the NLME shall check whether it has capacity in its pairing table for a new entry. If the NLME does not have capacity, it shall notify the application with a status of NO_ORG_CAPACITY and perform no further processing. If the NLME has capacity in its pairing table, it shall store the pairing reference of the next free entry.

The NLME shall then generate a pair request command frame as described in 3.3.3 and with the information supplied by the application.

The NLME shall then set *macMaxCSMABackoffs* and *macMaxFrameRetries* to the values of *nwkMaxFirstAttemptCSMABackoffs* and *nwkMaxFirstAttemptFrameRetries*, respectively. In addition, if the node performing pairing is a controller, it shall ensure *macPANId* is set to 0xffff before commencing transmission. The NLME shall then switch to the requested channel by changing *phyCurrentChannel*.

The NLME shall then transmit the pair request command frame using the multiple channel, acknowledged data transmission service via the MAC sub-layer data service. If the transmission was not successful, the NLME shall request the MAC sub-layer change the value of *phyCurrentChannel* to the next channel in the sequence and the transmission attempt shall be made again. The transmission attempt shall be made on each available channel until it is successful or until all channels have been attempted. If the transmission is still not successful, the NLME shall notify the application with the status returned from the MAC sub-layer and perform no further processing.

If the transmission was successful, the NLME shall request that the MAC sub-layer enable the receiver and wait either for *nwkResponseWaitTime* symbols or until the corresponding pair response command frame is received from the recipient, whichever is sooner. If the expected pair response command frame is not received within *nwkResponseWaitTime*, the NLME shall notify the application with a status of NO_RESPONSE and perform no further processing.

If the expected pair response command frame is received with a status other than SUCCESS, the NLME shall notify the application of the status returned in the pair response command frame and perform no further processing.

If the expected pair response command frame is received with a status of SUCCESS, the NLME shall use the information contained in the pair response command frame to create a provisional pairing table entry at the position identified by the pairing reference, stored above. The pairing table entry shall be created as described in Table 51.

Table 51 – Creation of the originator pairing table entry

Pairing table entry field	Description
Source network address	The value of *macShortAddress* if the node is a target OR The value of the allocated network address field from the pair response command frame.
Destination logical channel	The value of the channel designator sub-field of the frame control field of the pair response command frame if the recipient is a target OR The channel on which the pair response command frame was received otherwise.
Destination IEEE address	The source address from the pair response command frame.
Destination PAN identifier	The source PAN identifier from the pair response command frame.
Destination network address	The value of the network address field from the pair response command frame.
Recipient capabilities	The value of the node capabilities field from the pair response command frame.
Recipient frame counter	Reset to 0x00000000.
Security link key	Generated later, as necessary.

The NLME shall then check whether security is required for this pairing link. If the security enabled bit of both the node capabilities field of the incoming pair response command frame and *nwkcNodeCapabilities* are not set to one, the NLME shall move the pairing table entry from provisional to active and notify the application with the status code of SUCCESS and the pairing reference at which the pairing table entry was created or updated.

If the security enabled bit of both the node capabilities field of the incoming pair response command frame and *nwkcNodeCapabilities* are set to one, the recipient will generate a security link key for the pairing link and exchange it with the originator. The originator shall recover the key using the procedure described in 3.5.11.2. If the security link key recovery procedure was not successful, the NLME shall remove the provisional pairing table entry and perform no further processing. If the security link key recovery procedure was successful, the NLME shall move the created or updated pairing table entry from provisional to active.

The originator shall use the created pairing table entry as described in 3.5.8.

3.5.10.2 Pairing recipient procedure

For the node receiving a pairing request (the recipient), the application is notified via the NLME-PAIR.indication primitive and the application responds by issuing the NLME-PAIR.response primitive to the NLME. The application is notified of the status of the transmission of the pair response command frame via the NLME-COMM-STATUS.indication primitive.

If a pairing request command frame (see 3.3.3) is received by a controller, the NLME shall discard the frame, not notify the application and perform no further processing.

If a pairing request command frame is received by a target when *nwkInPowerSave* of the node is equal to TRUE, the NLME shall discard the frame and perform no further processing.

If a pairing request command frame is received by a target when *nwkInPowerSave* of the node is equal to FALSE, the NLME shall first check whether it already has an entry in its pairing table corresponding to the pairing originator. If an entry already exists in the pairing table, the NLME shall store the pairing reference of this entry and update it rather than creating a new entry.

If an entry corresponding to the pairing originator does not already exist in the pairing table, the NLME checks whether it has capacity in its pairing table for a new entry. If the NLME does not have capacity, it shall notify the application with a status of NO_REC_CAPACITY. If the NLME has capacity in its pairing table for a new entry, the NLME shall store the pairing reference of the next free entry.

The NLME shall then use the information contained in the pair request command frame to create or update the pairing table entry at the position identified by the pairing reference, stored above. The pairing table entry shall be created as described in Table 52 and it shall be marked as provisional.

Table 52 – Creation of the recipient pairing table entry

Pairing table entry field	**Description**
Source network address	The value of *macShortAddress*.
Destination logical channel	The value of the channel designator sub-field of the frame control field of the pair request command frame if the originator is a target OR The value of *nwkBaseChannel* otherwise.
Destination IEEE address	The source address from the pair request command frame.
Destination PAN identifier	The source PAN identifier from the pair request command frame if the originator is a target OR The value of *macPANId* otherwise.
Destination network address	Allocated if the originator is a controller and the recipient is a target OR The value of the network address field from the pair request command frame otherwise.
Recipient capabilities	The value of the node capabilities field from the pair request command frame.
Recipient frame counter	Reset to 0x00000000.
Security link key	Generated as necessary on acceptance of the pairing link by the application.

If an existing pairing table entry was updated, the NLME shall notify the application with a status of DUPLICATE_PAIRING and the pairing reference of the updated entry. If a new pairing table entry was created, the NLME shall notify the application with a status of SUCCESS and the pairing reference of the new entry.

If the application accepts the pair, it shall respond with a status of SUCCESS. If the application does not wish to accept the pair or if the NLME indicated it did not have capacity for a new entry in its pairing table, it shall respond with a status of NOT_PERMITTED or NO_REC_CAPACITY, respectively.

The NLME shall then set *macMaxCSMABackoffs* and *macMaxFrameRetries* to the values of *nwkMaxFirstAttemptCSMABackoffs* and *nwkMaxFirstAttemptFrameRetries*, respectively.

The NLME shall then generate a pair response command frame as described in 3.3.4 with the information supplied from the application. The NLME shall then transmit this frame to the originator of the pair request using the single channel, acknowledged data transmission service via the MAC sub-layer data service. If the transmission status was not equal to SUCCESS or if the transmission status was equal to SUCCESS but the application did not wish to accept the pair, the NLME shall remove the provisional pairing table entry and perform no further processing.

The NLME shall then check whether security is required for this pairing link. If the security enabled bit of both the node capabilities field of the incoming pair request command frame and *nwkcNodeCapabilities* are not set to one, the NLME shall move the created pairing table entry from provisional to active.

If the security enabled bit of both the node capabilities field of the incoming pair request command frame and *nwkcNodeCapabilities* are set to one, the NLME shall generate a security link key for the pairing link and exchange it with the originator using the procedure described in 3.5.11.1. If the security link key exchange procedure was not successful, the NLME shall remove the provisional pairing table entry and perform no further processing. If the security link key exchange procedure was successful, the NLME shall move the created pairing table entry from provisional to active.

The recipient shall use the created pairing table entry as described in 3.5.8.

3.5.10.3 Removing a pairing link

The procedure for removing a pairing link is initiated by the local application by issuing the NLME-UNPAIR.request primitive to the NLME and indicated to the remote application via the NLME-UNPAIR.indication primitive. Once the remote application has extracted any information, necessary to inform the user, from the pairing table it issues the NLME-UNPAIR.response primitive to the NLME. The status of the unpair are transferred to the local application via the NLME-UNPAIR.confirm primitive.

On receipt of a request to remove a pairing link, the NLME shall first check whether the requested pairing reference exists in the pairing table. If a corresponding entry does not exist in the pairing table, the NLME shall notify the application with an error code of NO_PAIRING. If a corresponding entry exists in the pairing table, the NLME shall generate an unpair request command frame as described in 3.3.5.

If a link key exists for the corresponding entry in the pairing table, the NLME shall securely process the unpair request frame before transmission using the procedure described in 3.5.11 and with the link key from the pairing table entry. Otherwise, the unpair request command frame shall be sent without security.

The NLME shall then transmit the unpair request command frame to the device listed in the pairing table entry via the MAC sub-layer data service. On receipt of the transmission confirmation from the MAC sub-layer (regardless of its success), the NLME shall delete the requested pairing table entry and the application shall be notified of the status of the transmission.

On receipt of an unpair request command frame (see 3.3.5), the NLME shall verify the security on the frame. If the unpair request command frame was sent without security but the pairing link is secure, the NLME shall discard the frame and the NLME shall perform no further processing. If the unpair request command frame was sent with security, the NLME shall securely process the incoming frame as described in 3.5.11. If the secure processing of the incoming frame fails, the NLME shall discard the frame and the NLME shall perform no further processing.

Once the security of the incoming unpair request command frame has been verified, the NLME shall locate the entry in its pairing table that corresponds to the device that transmitted the frame. If an entry is found, the NLME shall notify the application, indicating the reference of the pairing table entry to be deleted. The application may then extract any information from the pairing table that is necessary to inform the user. When the application has all the information it requires from the pairing table entry, it shall indicate to the NLME that the entry can be deleted. The NLME shall then remove the entry from the pairing table.

3.5.11 Security

A node that is capable of applying security shall set the security capable sub-field of *nwkcNodeCapabilities* to one.

A unique and random security key shall be established for each communication link during the pairing procedure. If both nodes are capable of applying security, as indicated by the node capabilities field that is exchanged by the devices, a key shall be generated and exchanged as described in sub-clause 3.5.11.1.

Each node shall maintain a 4 octet network frame counter, *nwkFrameCounter*, in its NIB table. This counter is initialized to 0x00000001 when the node is first initialized and is then incremented by one each time a packet is generated for transmission.

Additionally, a node shall store the network frame counter of each of the nodes it is paired with in the recipient frame counter field of the corresponding entries in the pairing table. This value is initialized to zero when the pairing table entry is created and is updated each time it successfully receives a secured packet from the corresponding node.

The ZigBee RF4CE frame security shall use the CCM* mode of operation as described in Annex B of [R1] . This operation uses the AES-128 block cipher algorithm. The parameter *M* shall have a value of 4.

In the following sub-sections, the concatenated fields are represented in the order where the leftmost field occupies the lower order bytes.

3.5.11.1 Target link key exchange procedure

The target link key exchange procedure shall be instigated following the successful transmission of the pair response command frame back to the originator.

The NLME shall generate a 128-bit security link key for the pairing link and attempt to transfer it to the originator of the pair request command frame. To transfer the key, the NLME shall send (n + 1) key seed command frames to the pairing originator, where n shall be equal to the value of the key exchange transfer count field of the pair request command frame. During the transfer the NLME shall set *macMaxFrame-Retries* to zero and each transmission shall use the single channel, acknowledged data transmission service.

The transfer of all key seeds shall take no longer than ((n + 1) * *nwkcMaxKeySeed-WaitTime*) symbols. If all the transfers do not complete within this time, the security link key exchange procedure shall be considered unsuccessful and the NLME shall notify the application with a communication status of SECURITY_TIMEOUT and return to the pairing procedure.

For each transmission of the key seed command frame, the NLME shall increment the seed sequence number field, starting from 0x00. For the first n seed transmissions, the seed data field shall contain an 80-octet random number. For the $(n + 1)^{th}$ seed transmission, the seed data field is a special case and shall be computed in two phases:

1. Compute the XOR of four 128-bit random numbers and the 128-bit security link key generated above.
2. Compute the XOR of all of the previous random numbers sent in the first n key seed transmissions and the result of phase 1.

The seed data field of the final key seed transmission shall then be set to the value computed in phase 2. These computations are illustrated in Figure 30.

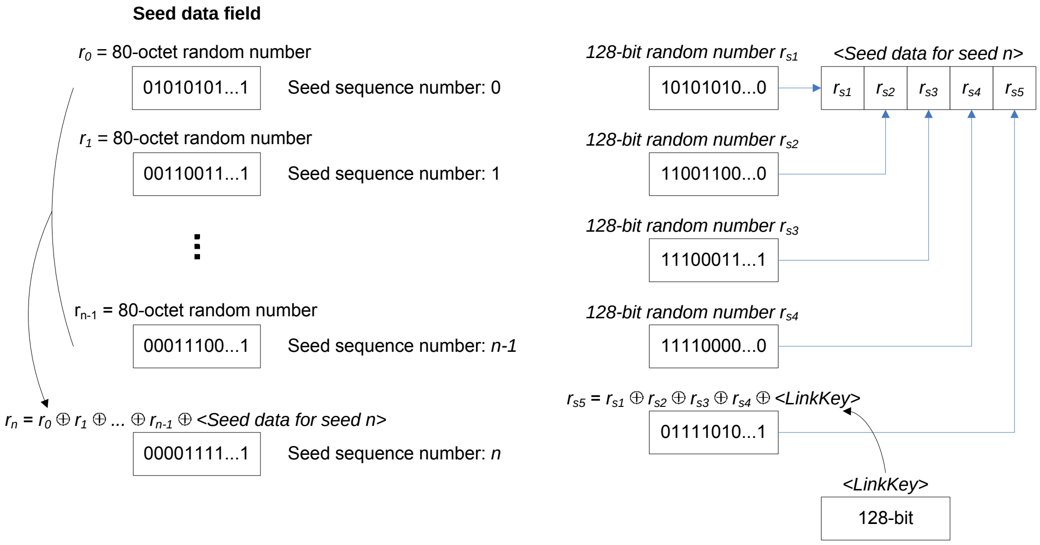
If an acknowledgement is not received for any of the key seed command frames, the NLME shall regenerate the appropriate random parts of the seed data field, while keeping the seed sequence number field the same and re-transmit the key seed command frame to the pairing originator.

If all transfers are received within ((n + 1) * *nwkcMaxKeySeedWaitTime*) symbols, the NLME shall wait either for *nwkResponseWaitTime* symbols or until a ping request command frame is received from the originator of the pair request command frame, whichever is sooner. If a ping request command frame is not received within *nwkResponseWaitTime*, the security link key exchange procedure shall be considered unsuccessful and the NLME shall notify the application with a communication status of SECURITY_FAILURE and return to the pairing procedure.

If a ping request command frame is received within *nwkResponseWaitTime*, the NLME shall verify that the ping request was received as a secured transmission, that the ping options field is equal to 0x00 and that the ping payload field was 4 octets in length. If the ping request command frame cannot be verified in this way, the security link key exchange procedure shall be considered unsuccessful and the NLME shall notify the application with a communication status of SECURITY_FAILURE and return to the pairing procedure.

If the ping request command frame was verified as described above, the NLME shall generate a ping response command frame and transmit it back to the originator of the ping request command frame using the secure data transmission service via the MAC sub-layer data service. The ping options and ping payload fields shall contain the same values of the corresponding fields of the ping request command frame.

The NLME shall then notify the application of the status of the transmission. If the transmission was unsuccessful, the security link key exchange procedure shall also be considered unsuccessful. Conversely, if the transmission was successful, the security link key exchange procedure shall also be considered successful. The NLME shall then return to the pairing procedure.


Seed data field
r_0 = 80-octet random number
01010101...1
Seed sequence number: 0
r_1 = 80-octet random number
00110011...1
Seed sequence number: 1
r_{n-1} = 80-octet random number
00011100...1
Seed sequence number: *n-1*
$r_n = r_0 \oplus r_1 \oplus \ldots \oplus r_{n-1} \oplus$ <Seed data for seed n>
00001111...1
Seed sequence number: *n*
128-bit random number r_{s1}
10101010...0
<Seed data for seed n>
r_{s1} r_{s2} r_{s3} r_{s4} r_{s5}
128-bit random number r_{s2}
11001100...0
128-bit random number r_{s3}
11100011...1
128-bit random number r_{s4}
11110000...0
$r_{s5} = r_{s1} \oplus r_{s2} \oplus r_{s3} \oplus r_{s4} \oplus$ <LinkKey>
01111010...1
<LinkKey>
128-bit

a) Seed data for the key seed transmissions **b) Computing the seed data for seed n**

Figure 30 – Target side link key exchange

3.5.11.2 Pairing originator link key recovery procedure

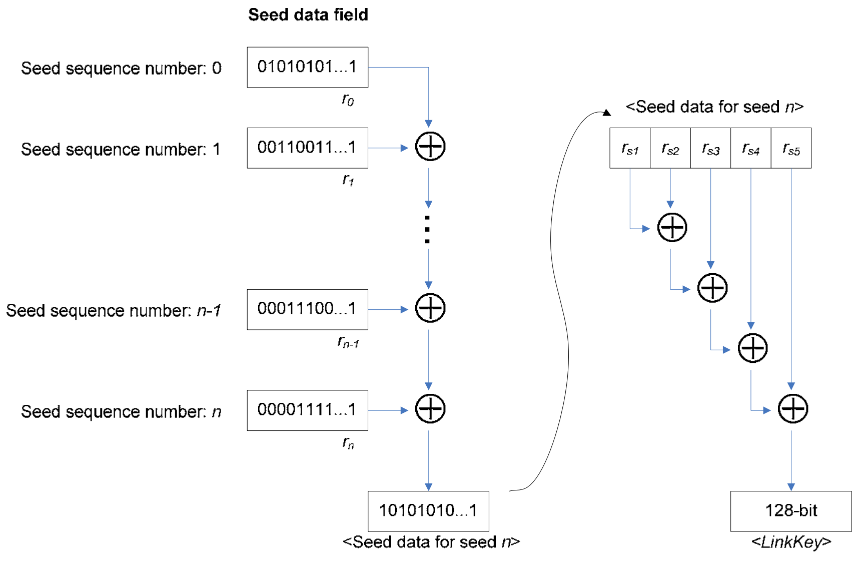
The originator link key recovery procedure shall be instigated following the successful reception of the pair response command frame from the pairing recipient.

The NLME shall wait for the reception of (n + 1) key seed command frames from the target, where n shall be equal to the value of the key exchange transfer count field of the pair request command frame sent by this node. All key seeds shall be received within ((n + 1) * *nwkcMaxKeySeedWaitTime*) symbols. If all the transfers are not received within this time, the security link key recovery procedure shall be considered unsuccessful and the NLME shall notify the application with a status of SECURITY_TIMEOUT and return to the pairing procedure.

If all transfers are successfully received within ((n + 1) * *nwkcMaxKeySeedWaitTime*) symbols, the NLME shall compute the link key in two phases:

1. Compute the XOR of the seed data fields from each of the (n + 1) key seed command frames.
2. Divide the result of phase 1 into five 128-bit blocks and compute their XOR.

The link key shall then be set to the result of phase 2. These computations are illustrated in Figure 31.



a) Processing the received seed data

b) Computing the link key

Figure 31 – Pairing originator side link key exchange

The NLME shall then generate a ping request command frame and transmit it to the recipient of the pair request command frame using the secure data transmission service via the MAC sub-layer data service. The ping options field shall contain 0x00 and the ping payload field shall contain a 4 octet random value.

If the transmission was not successful, the security link key recovery procedure shall be considered unsuccessful and the NLME shall notify the application with the status returned from the MAC sub-layer and return to the pairing procedure.

If the transmission was successful, the NLME shall request that the MAC sub-layer enable the receiver and wait either for *nwkResponseWaitTime* symbols or until a ping response command frame is received from the recipient of the pair request command frame, whichever is sooner. If a ping response command frame is not received within *nwkResponseWaitTime*, the security link key recovery procedure shall be considered unsuccessful and the NLME shall notify the application with a status of NO_RESPONSE and return to the pairing procedure.

If a ping response command frame is received within *nwkResponseWaitTime*, the NLME shall verify that the ping response was received as a secured transmission and that the ping options and the ping payload fields contain the same values as was sent in the ping request command frame. If the ping response command frame cannot not verified in this way, the security link key recovery procedure shall be considered unsuccessful and the NLME shall notify the application with a status of SECURITY_FAILURE and return to the pairing procedure.

If the ping response command frame can be verified in this way, the security link key recovery procedure shall be considered successful and the NLME shall notify the application with a status of SUCCESS and return to the pairing procedure.

3.5.11.3 Outgoing frame security

An outgoing ZigBee RF4CE network frame (NPDU) shall be secured according to the procedure described in section B.4.1 in [R1] .

The inputs to the operation are as follows

- The Key *key* shall be set to the security link key field from the pairing table entry corresponding to the recipient.
- The nonce *N* shall be formed by the concatenation of the following fields:

  SrcIEEEAddr || FrameCounter || SecurityLevel

SrcIEEEAddr shall be the 8-byte IEEE address of the node performing this security operation, i.e. *aExtendedAddress*. FrameCounter shall be set to the 4-byte frame counter field from the NHR of the outgoing frame. SecurityLevel shall be set to the 1-byte value of 0x05.

- The octet string *a* shall be formed by the concatenation of the following fields:

  FrameControl || FrameCounter || DstIEEEAddr

  FrameControl shall be set to the 1-byte frame control field in the NHR of the outgoing frame. FrameCounter shall be set to the 4-byte Frame counter field in the NHR of the outgoing frame. DstIEEEAddr shall be set to the 8-byte destination IEEE address field from the pairing table entry corresponding to the recipient.
- The octet string *m* shall be set to the ZigBee RF4CE network payload (NSDU).

Upon completion of the security operation, the encrypted message ciphertext *C* and the encrypted authentication tag *U* are generated.

The ciphertext *C* shall replace the ZigBee RF4CE network payload (NSDU). The tag *U* shall replace the Message integrity code (NFR) in the ZigBee RF4CE frame.

3.5.11.4 Incoming frame security

A secured incoming ZigBee RF4CE network frame shall be un-secured according to the procedure described in section B.4.2 in [R1] .

The inputs to the operation are as follows:

- The Key *key* shall be set to the security link key field from the pairing table entry corresponding to the originator.
- The nonce *N* shall be formed by the concatenation of the following fields:

  SrcIEEEAddr || FrameCounter || SecurityLevel

  SrcIEEEAddr shall be set to the 8-byte Destination IEEE address field from the pairing table entry corresponding to the MAC source address of the incoming frame. FrameCounter shall be set to the 4-byte frame counter field from the NHR of the incoming frame. SecurityLevel shall be set to the 1-byte value of 0x05.
- The octet string *c* shall be formed by the concatenation of the following fields from the incoming frame:

  NWK payload || NFR
- The octet string *a* shall be formed by the concatenation of the following fields:

  FrameControl || FrameCounter || DstIEEEAddr

  FrameControl shall be set to the 1-byte frame control field in the NHR of the incoming frame. FrameCounter shall be set to the 4-byte Frame counter field in the NHR of the incoming frame. DstIEEEAddr shall be set to the 8-byte IEEE address of the node performing this operation, i.e. *aExtendedAddress*.

Upon completion of the security operation, if the verification of the authentication tag fails, the incoming frame shall be discarded.

Otherwise, the NWK payload of the incoming frame shall be replaced by the output string *m* and the recipient frame counter field in the pairing table entry corresponding to the originator of the incoming frame shall be replaced by the network frame counter field in the NHR of the incoming frame.

4 Revision History

Version	Date	Details	Editor
0.5	April 2008	Initial draft.	Phil Jamieson
0.75	August 2008	Update following technical working group informal review.	Phil Jamieson
0.85	September 2008	Further updates to add security and application profile protocols, plus comments from the membership.	Phil Jamieson
0.90	October 2008	Updates following technical working group review and network pre-test event. CERC profile split into separate document (080007).	Phil Jamieson
0.91	November 2008	Updates following 0.90 review and CERC profile test event.	Phil Jamieson
0.93	December 2008	Updates following 0.91 and evaluators review.	Phil Jamieson
0.95	December 2008	Frame control reserved sub-field change. Specification approved by the TWG as a v1.00 release candidate.	Phil Jamieson
1.00	December 2008	Specification approved by the RF4CE founders.	Phil Jamieson