



# UBEC ZigBee Platform and IEEE 802.15.4

**UBEC confidential**



**Hugo**  
**kuochang@ubec.com.tw**  
**hugo0210@gmail.com**  
**2006.10.18**



# UBEC's platform



UZ2400

**UBEC confidential**  
Feature and Proprietary Application

- ✚ True single-chip 2.4 GHz IEEE 802.15.4 compliant RF transceiver with baseband modem and MAC support
- ✚ DSSS baseband modem with 2 MChips/s and 250 kbps effective data rate, and 625Kbps turbo mode
- ✚ Low current consumption (RX: 18 mA, TX: 22 mA)
- ✚ Low supply voltage (2.1 – 3.6 V) with integrated voltage regulator
- ✚ Programmable output power
- ✚ No external RF switch / filter needed
- ✚ Very few external components
- ✚ 128(RX) + 128(TX) byte data buffering
- ✚ Digital RSSI / LQI support
- ✚ Hardware MAC encryption (AES-128)
- ✚ Powerful and flexible development tools available

## Interrupt mechanism

**SREG31: ISRSTS**

Offset: 0x31

Bits	Name	Description	Reset Value	R/W
7	SlpAltrq	Sleep alert interrupt	0	RC
6	WakAltrq	Wake-up alert interrupt	0	RC
5	HSymTmrlrq	Half symbol timer interrupt	0	RC
4	SecIrq	Security key request interrupt	0	RC
3	RXOKIrq	RX OK interrupt	0	RC
2	Txg2r	GTS FIFO 2 release interrupt	0	RC
1	Txg1r	GTS FIFO 1 release interrupt	0	RC
0	Txnr	TXFIFO release interrupt	0	RC

**Note:** chances to clear these status when writing this register

# Sending a packet procedure

- ✚ Sending a packet in Normal FIFO

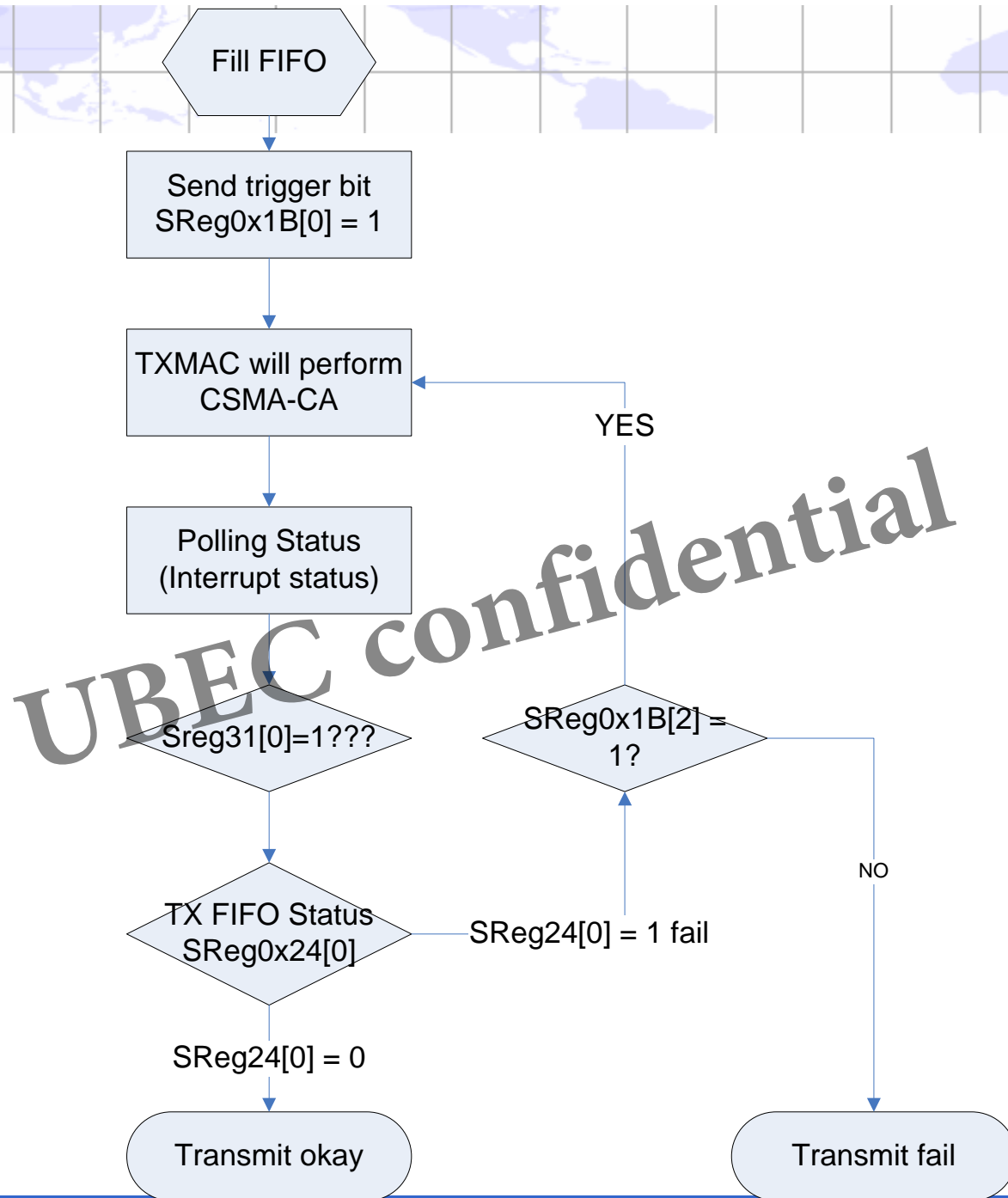
- The format is as follows



**SREG24: TXSR**

Offset: 0x24

Bits	Name	Description	Reset Value	R/W
7-6	<b>txnretryn</b>	Retry times of the most recent TXNFIFO transmission.	0	R
5	<b>ccafail</b>	Channel busy causes CSMA-CA fails.	0	R
4	<b>Txg2fnt</b>	GTSFIFO2 transmission fails due to not enough time before end of GTS slot	0	R
3	<b>Txg1fnt</b>	GTSFIFO1 transmission fails due to not enough time before end of GTS slot	0	R
2	<b>Txg2s</b>	GTSFIFO2 release status. 0: ok, 1: fail(retry count exceed)	0	R
1	<b>Txg1s</b>	GTSFIFO1 release status 0: ok, 1: fail(retry count exceed)	0	R
0	<b>Txns</b>	TXNFIFO release status 0: ok, 1: fail(retry count exceed)	0	R

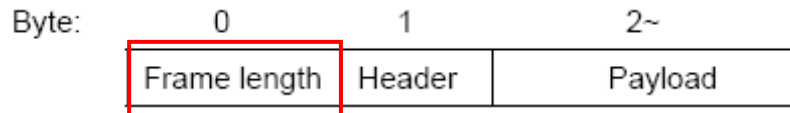


UBEC confidential

# How do receive a packet

## Receiving a packet

- Frame length (in bytes) includes header, payload and FCS(2 bytes).

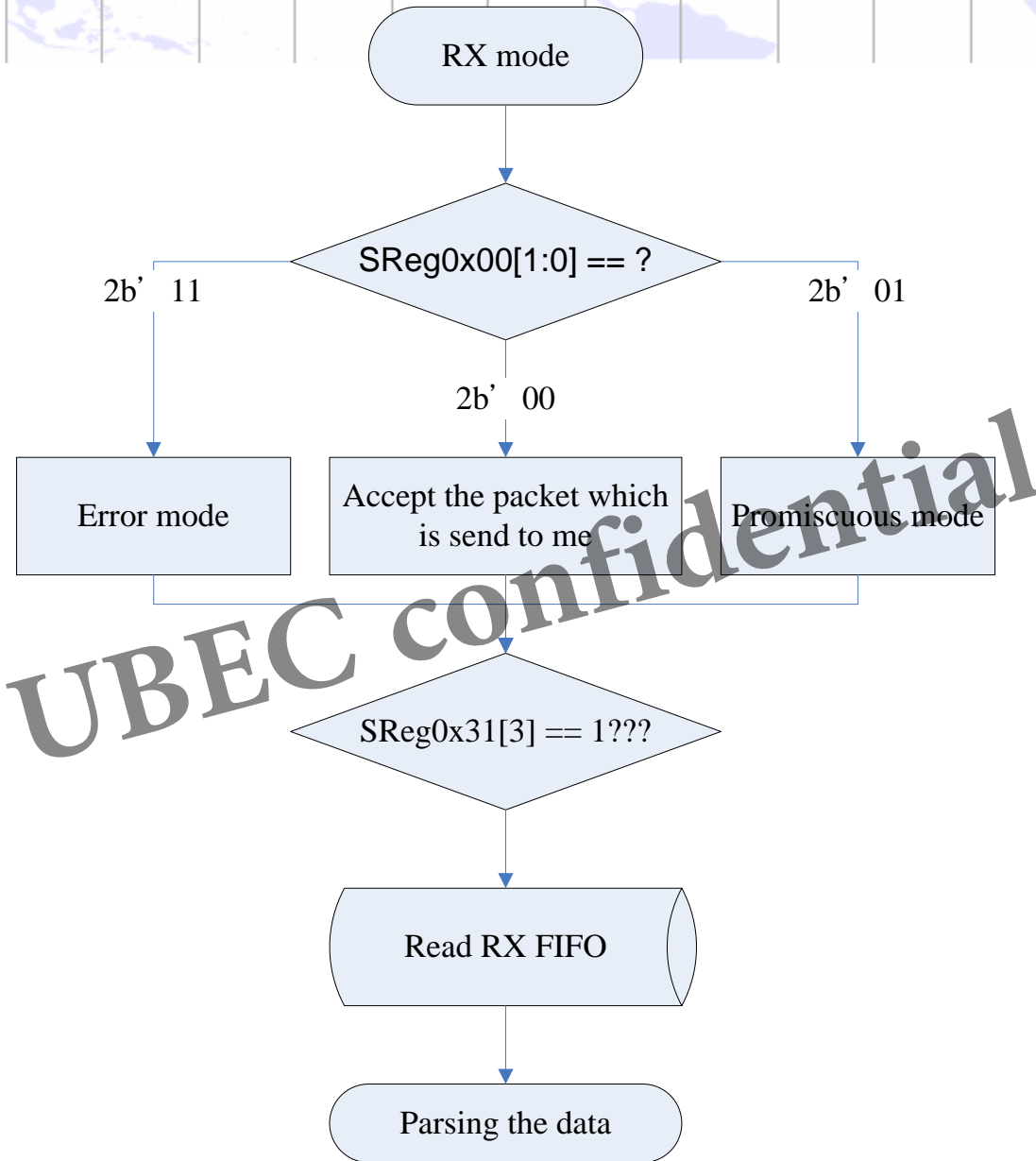


SREG00: RXMCR

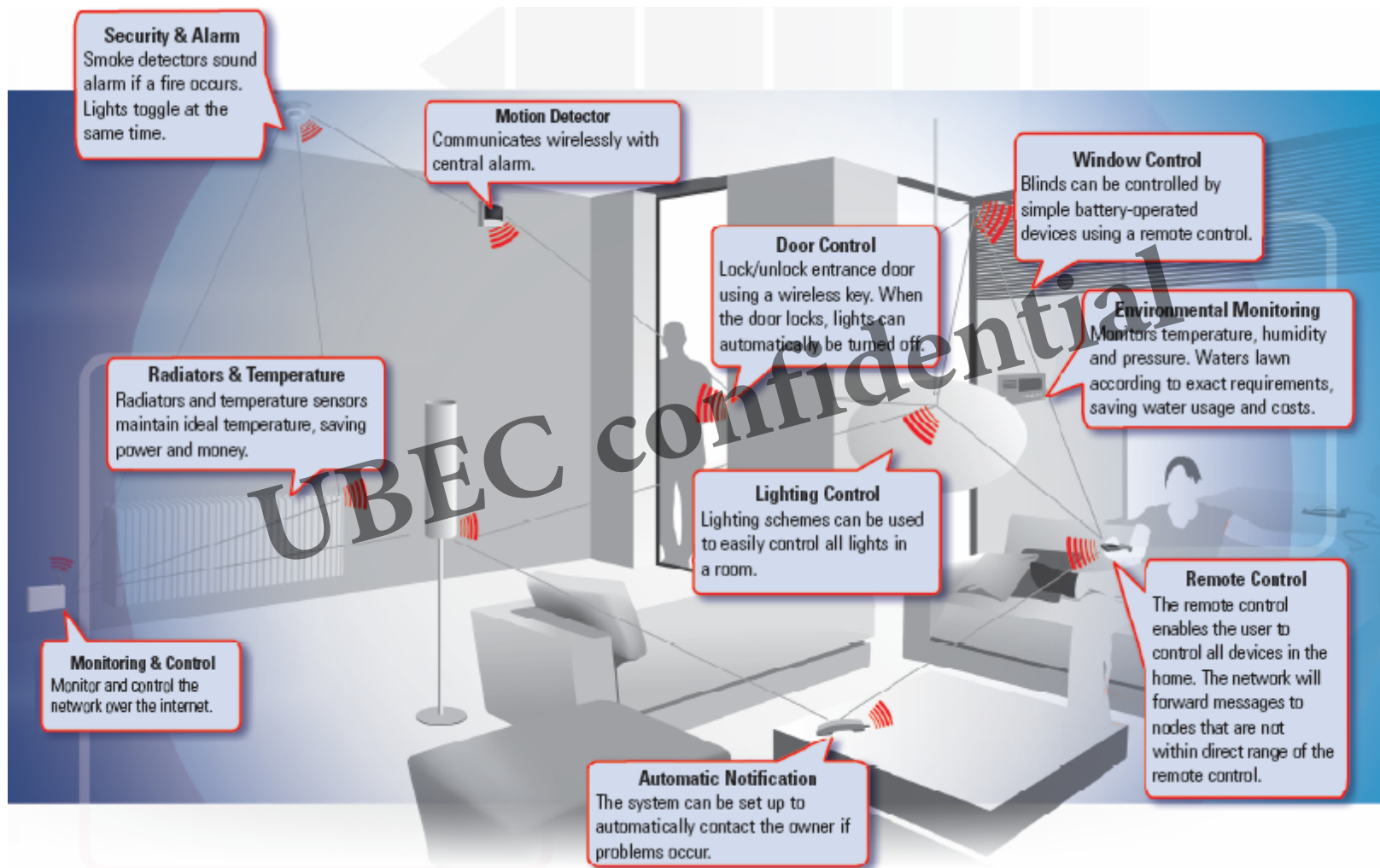
Offset: 0x00

Bits	Name	Description	Reset Value	R/W
7	NoCRC	No CRC data appended with normal fifo	0	R/W
6	BB_Ipbk	Baseband loopback enable	0	R/W
5	noRspACK	No ACK response in any case	0	R/W
4	MAC_Ipbk	MAC loopback function enable	0	R/W
3	isPANCord	This device is a PAN coordinator.	0	R/W
2	isCord	This device is a coordinator.	0	R/W
1	errpkt	Accept all kinds of pkt(including CRC error).	0	R/W
0	Promi_mode	Accept all packets with CRC OK.	0	R/W





# Device Application



# Proprietary Application: Home Security



Sensor  
if anyone open the door, it send sensor  
trigger packet to IP gateway

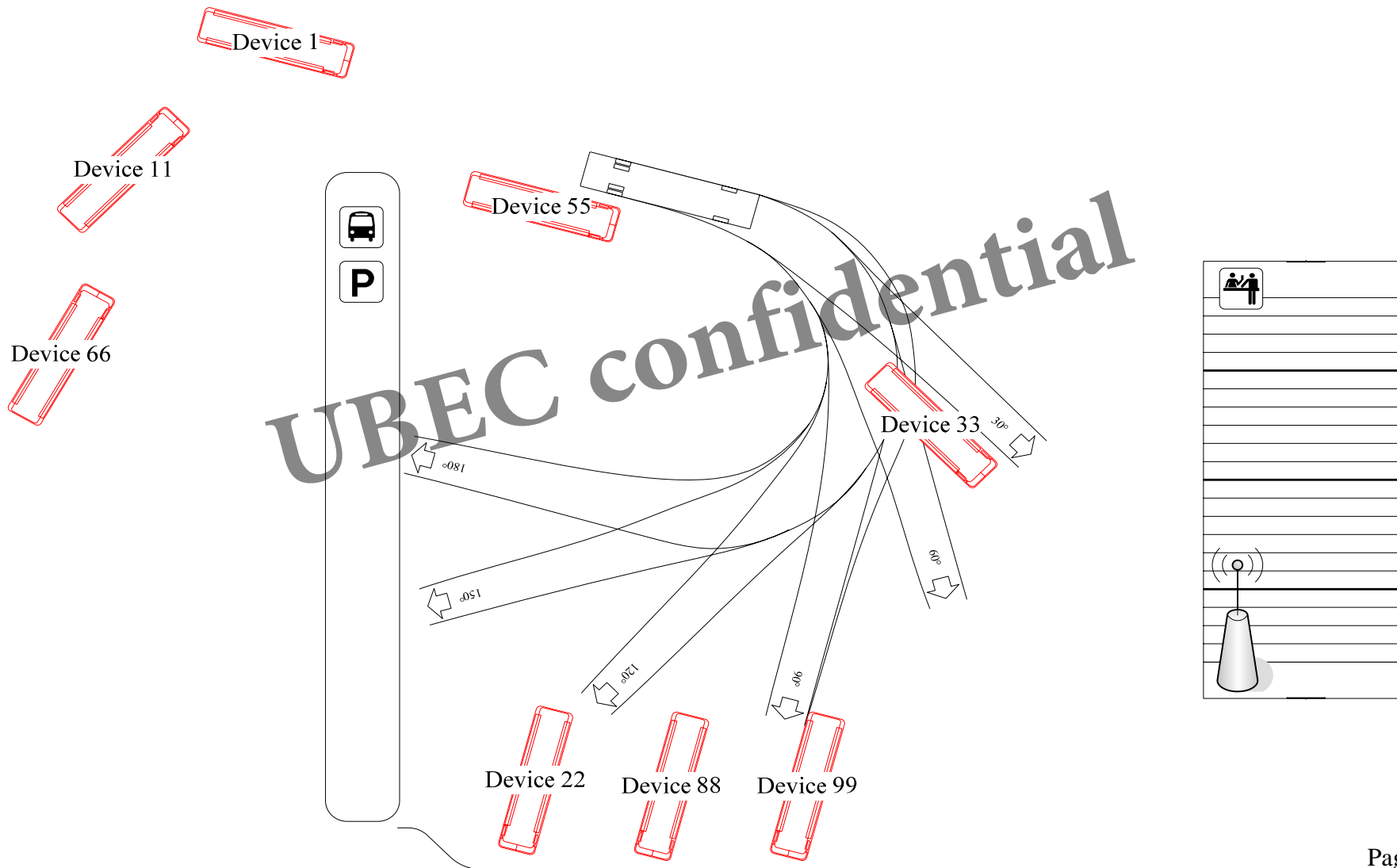
- IP Gateway
1. 3 Sensor triggered output (DO)
  2. Event function output (DO)  
Enable/Disable/Alarm/Lock
  3. 3 Sensor LIVE indication LED

- Remote Controller\_1
1. 4 Event keys. This device is using to remote controller the IPGateway.
  2. Enable/Disable key: Enable/Disable the sensor trigger output or not in IPGateway
  3. LOCK key: enable lock, any other key can't work.
  4. ALARM key: send ALARM to IPGateway.
  - 5 LOCK -> Enable+Disalbe = Association key



UBEC confidential

- Automatic ACK response machine



packet : 21 cc 00 12 34 50 51 52 53 54 55 56 57 12 34 20 21 22 23 24 25 26 27  
0x21 = 2b'0010 0001 bit[2:0]: data packet, bit[5]: ACK request

1. Set PAN ID and Short or Long Address as first
  - PAN ID: SReg0x01 0x02
  - Short Address: SReg0x03 0x04
  - Long Address: SReg0x05-0x0C
2. Trigger this packet SReg0x1B = 5
  - SReg0x1B[2]:1 TX packet in normal FIFO needs ACK response

How do you know you receive a ACK packet?

- Sreg0x24[0] = 0 receive a ACK packet
- Sreg0x24[0] = 1 doesn't receive a ACK packet
- SReg0x24[7:6] indicate the times of retransmitted.

## Security Procedure

**UBEC confidential**  
MAC and Upper layer

# Sending a packet with security encryption:

1. Fill in TXN\_FIFOs you want to send with encryption

Byte: 0	1	2~	
Header length	Frame length	Header	Payload

2. Fill in key into key table memory

- Normal FIFO key Long: 0x280 – 0x28F

3. Fill in cipher mode

- Normal FIFO: SReg2C[2:0]

SREG2C: SECCR0

Offset: 0x2c

Bits	Name	Description	Reset Value	R/W
7	SecIgnore	RX security process ignore	0	WT
6	SecStart	RX security process start	0	WT
5-3	RX_cipher	RX cipher select, as 802.15.4 Sec7.6 Table 75	0	R/W
2-0	TX_cipher_n	TX cipher select for normal fifo, as 802.15.4 Sec7.6 Table 75	0	R/W



# Sending a packet with security encryption:

4. Trigger FIFO start with security
  - Normal FIFO: SReg1B[1:0] = “11”

SREG1B: TXNMTRIG  
Offset: 0x1b

Bits	Name	Description	Reset Value	R/W
7-5	reserved			
4	Txm_ackp	Received ACK frame with pending indication. This bit is cleared at next triggering of TXNFIFO.	0	R
3	indirect	Activate indirect transmission. Only for coordinator use.	0	R/W
2	ackreq	TX packet in TXNFIFO needs ACK response.	0	R/W
1	sech	Security enable for TXNFIFO packet.	0	R/W
0	txstart	Trigger TXMAC to send the packet in TXFIFO.	0	WT

5. Wait for FIFO release interrupt/status as plaintext packet does.

# Sending a packet with security encryption:

Key table @ TX normal FIFO 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25

Package 09CC 00 1234 5051525354555657 1234 2021222324252627 00 01 02 03 04 05 06  
07 08 09

## 1. Non-security SReg0x2C=0x00

Time (us)	Length	Frame control field	Sequence number	Dest. PAN	Dest. Address	Source PAN	Source Address	Encrypted MAC payload	RSSI (dBm)	FCS
+0	35	Type Sec Pnd Ack req Intra PAN	0x00	0x3412	0x5756555453525150	0x3412	0x2726252423222120	00 01 02 03 04	-50	OK
=0		DATA 1 0 0 0						05 06 07 08 09		

## 2. AES-CTR SReg0x2C=0x01

Time (us)	Length	Frame control field	Sequence number	Dest. PAN	Dest. Address	Source PAN	Source Address	Encrypted MAC payload	LQI	FCS
+523255	35	Type Sec Pnd Ack req Intra PAN	0x00	0x3412	0x5756555453525150	0x3412	0x2726252423222120	00 01 02 03 04	152	OK
=523255		DATA 1 0 0 0						05 FC 86 7F 70		

## 3. AES-CCM-128 SReg0x2C=0x02

Length	Frame control field	Sequence number	Dest. PAN	Dest. Address	Source PAN	Source Address	Encrypted MAC payload	LQI	FCS
51	Type Sec Pnd Ack req Intra PAN	0x00	0x3412	0x5756555453525150	0x3412	0x2726252423222120	00 01 02 03 04 05 32 7D 21 A5 B5 00 F7 F4 FB 5A D5 7C C1 27 98 EB 28 B4 80 0D	152	OK

## 4. AES-CCM-64 SReg0x2C=0x03

Time (us)	Length	Frame control field	Sequence number	Dest. PAN	Dest. Address	Source PAN	Source Address	Encrypted MAC payload	LQI	FCS
+51636318	43	Type Sec Pnd Ack req Intra PAN	0x00	0x3412	0x5756555453525150	0x3412	0x2726252423222120	00 01 02 03 04 05 32 7D 21	152	OK
=68594427		DATA 1 0 0 0						A5 72 9E AF 30 2B B1 F3 3E		

# Sending a packet with security encryption:

## 5. AES-CCM-32 SReg0x2C=0x04

Time (us)	Length	Frame control field	Sequence number	Dest. PAN	Dest. Address	Source PAN	Source Address	Encrypted MAC payload	LQI	FCS
+16655054 =85249481	39	Type Sec Pnd Ack req Intra PAN DATA 1 0 0 0	0x00	0x3412	0x5756555453525150	0x3412	0x2726252423222120	00 01 02 03 04 05 32 7D 21 A5 C8 2B 90 15	156	OK

## 6. AES-CBC-MAC-128 SReg0x2C=0x04

Length	Frame control field	Sequence number	Dest. PAN	Dest. Address	Source PAN	Source Address	Encrypted MAC payload	LQI	FCS
39	Type Sec Pnd Ack req Intra PAN DATA 1 0 0 0	0x00	0x3412	0x5756555453525150	0x3412	0x2726252423222120	00 01 02 03 04 05 32 7D 21 A5 C8 2B 90 15	156	OK

## 7. AES-CBC-MAC-64 SReg0x2C=0x06

Time (us)	Length	Frame control field	Sequence number	Dest. PAN	Dest. Address	Source PAN	Source Address	Encrypted MAC payload	LQI	FCS
+12966573 =117729650	43	Type Sec Pnd Ack req Intra PAN DATA 1 0 0 0	0x00	0x3412	0x5756555453525150	0x3412	0x2726252423222120	00 01 02 03 04 05 06 07 08 09 09 84 5A 60 B9 B5 42 4B	168	OK

## 8. AES-CBC-MAC-32 SReg0x2C=0x07

Time (us)	Length	Frame control field	Sequence number	Dest. PAN	Dest. Address	Source PAN	Source Address	Encrypted MAC payload	LQI	FCS
+10641506 =128371156	39	Type Sec Pnd Ack req Intra PAN DATA 1 0 0 0	0x00	0x3412	0x5756555453525150	0x3412	0x2726252423222120	00 01 02 03 04 05 06 07 08 09 B9 B5 42 4B	188	OK

# Receiving a packet with security decryption

## 1. Security interruption

SREG31: ISRSTS

Offset: 0x31

Bits	Name	Description	Reset Value	R/W
7	SlpAltrq	Sleep alert interrupt	0	RC
6	WakAltrq	Wake-up alert interrupt	0	RC
5	HSymTmrIrq	Half symbol timer interrupt	0	RC
4	SecIrq	Security key request interrupt	0	RC
3	RXOKIrq	RX OK interrupt	0	RC
2	Txg2r	GTS FIFO 2 release interrupt	0	RC
1	Txg1r	GTS FIFO 1 release interrupt	0	RC
0	Txnr	TXFIFO release interrupt	0	RC

Note: chances to clear these status when writing this register

## 2. Key and cipher mode:

- RX FIFO key address

RXFIFO key	Long: 0x2B0 – 0x2BF
------------	---------------------

- Cipher mode of RX FIFO: SReg2C[5:3]
- Security start: SReg2C[6] or Security ignore: SReg2C[7]

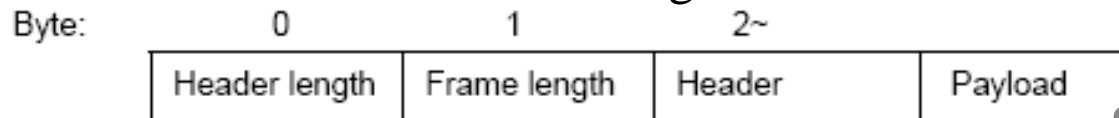
- CTR mode
  - NO MIC (message integrity code), don't make sure the data is right
- CBC-MAC
  - No use Nonce
  - Data the same, Encryption result is the same.

UBEC confidential

# Upper cipher encryption

Purpose: Use TXNFIFO to perform upper cipher encryption:

1. Fill in TXNFIFO with following format:

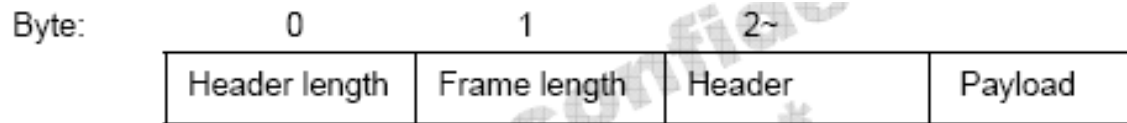


2. Write **NONCE** (13 bytes) at LReg40~4C.
3. Set SReg37[6] = '1'.
4. Set TXNFIFO security key.
5. Set cipher mode of TXNFIFO at SReg2C[2:0].
6. Trigger TXNFIFO to send at SReg1B.
7. Check ISRSTS(SReg31[0]='1') and TXSR(SReg24[0]='0'), meaning ciphering is done.
8. Read back TXNFIFO to get the result.

# Upper cipher decryption

Purpose: TXNFIFO to perform upper cipher decryption:

1. Fill in TXNFIFO with following format:



2. Write **NONCE** (13 bytes) at LReg40~4C.
3. Set SReg37[7] = '1'.
4. Set TXNFIFO security key.
5. Set cipher mode of TXNFIFO at SReg2C[2:0].
6. Trigger TXNFIFO to send at SReg1B.
7. Check ISRSTS(SReg31[0]='1') and TXSR(SReg24[0]='0'), meaning ciphering is done.
8. Read back TXNFIFO to get the result.
9. Check RXSR(SReg30[6]) for MIC check error. If it's '0', then it's ok.

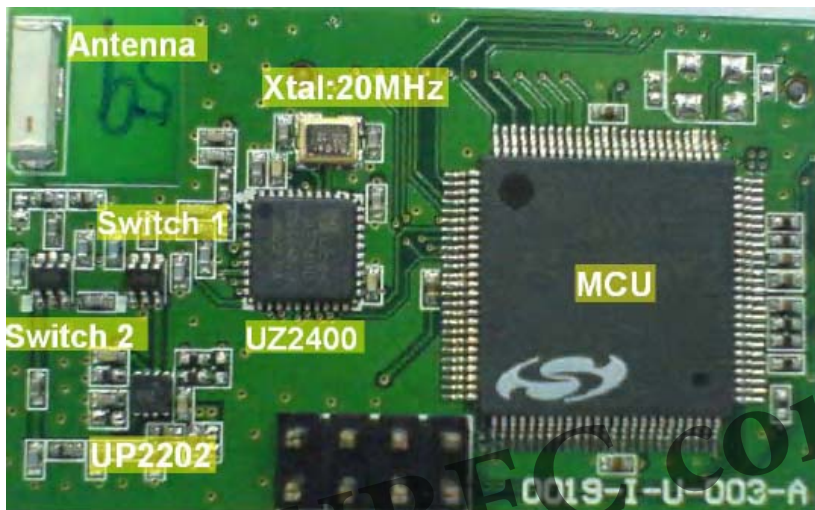
- Avoiding to get the same data output after security processing, the nonce will be implemented by each security operation. in the mean time, this procedure can enhance our data safety also. They included extended address , frame counter and security control
  - Extended address
  - Frame counter
    - This parameter will change very time and let the nonce different
  - Security control

Octets: 8	4	1
Source address	Frame counter	Security control

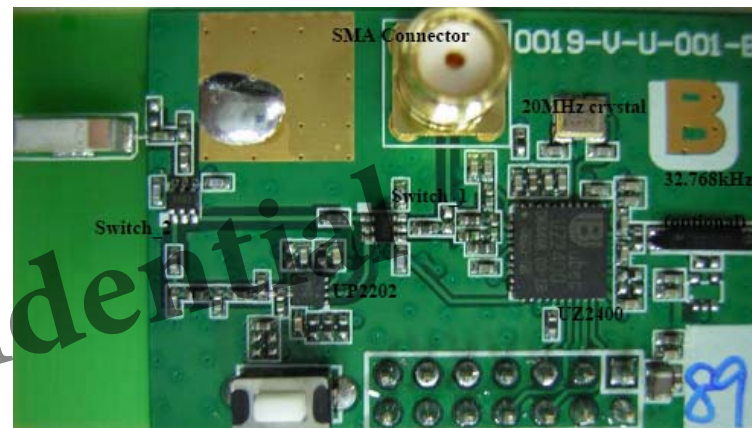


- ✚ Development kit
  - RF module
    - DotForce
    - DotPower
    - COB
    - Stamp type
  - MCU board
    - Silicon Labs C8051F124 mother board
  - Integrate kit
    - U-Zig: PA, UZ2400, MCU
    - U-Zig with diversity antenna
- ✚ Development tool
  - Seeker (debug application, sniffer using)

- U-Zig Power module with Chip antenna



DotPower module



- U-Zig power module module with diversity antenna
- DotForce module



- ✚ Chip on Board module



- ✚ Stamp type RF module



UBEC confidential

IEEE 802.15.4  
(MAC)

**UBEC confidential**

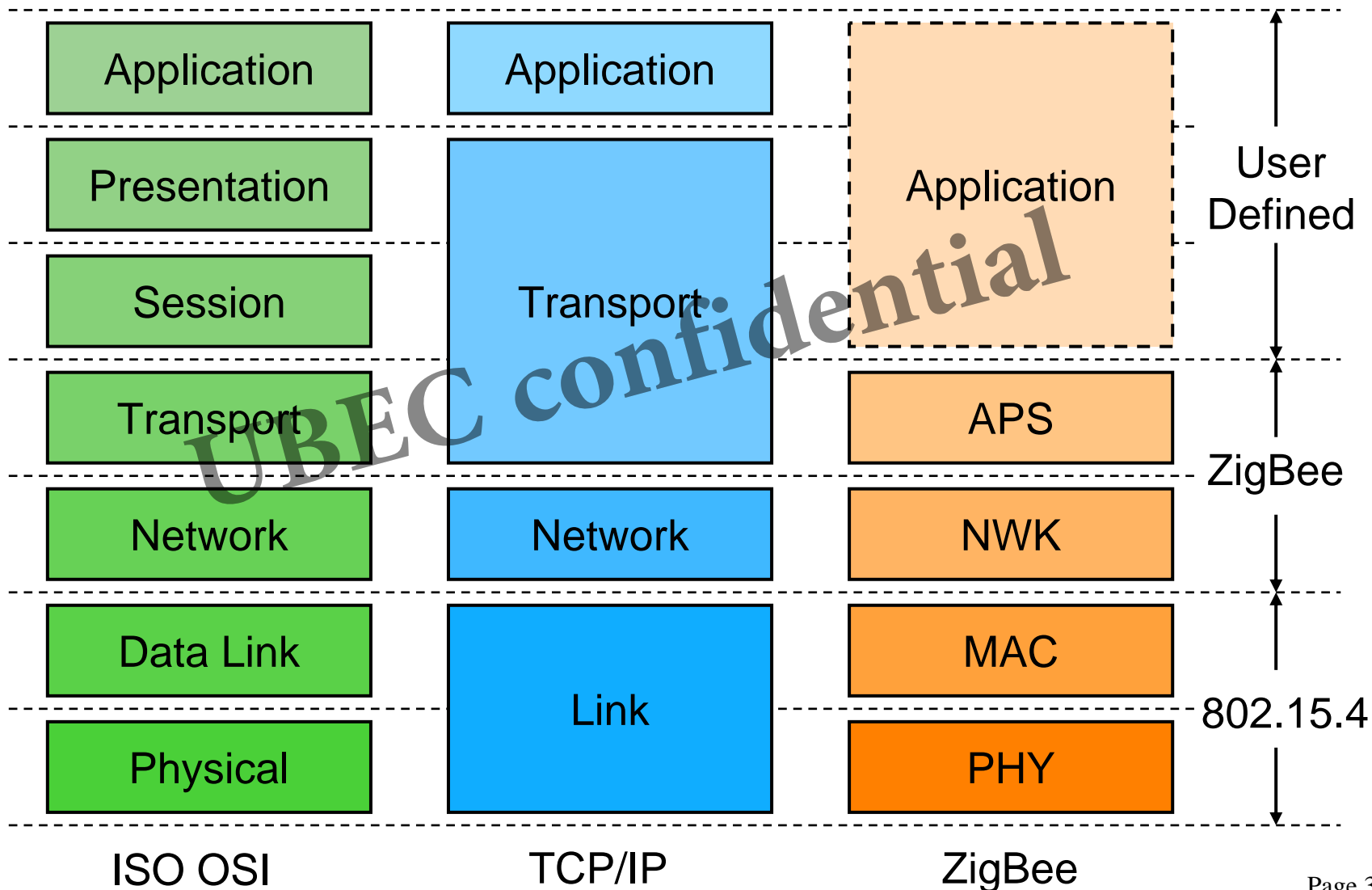
Session 1:  
Brief Introduction

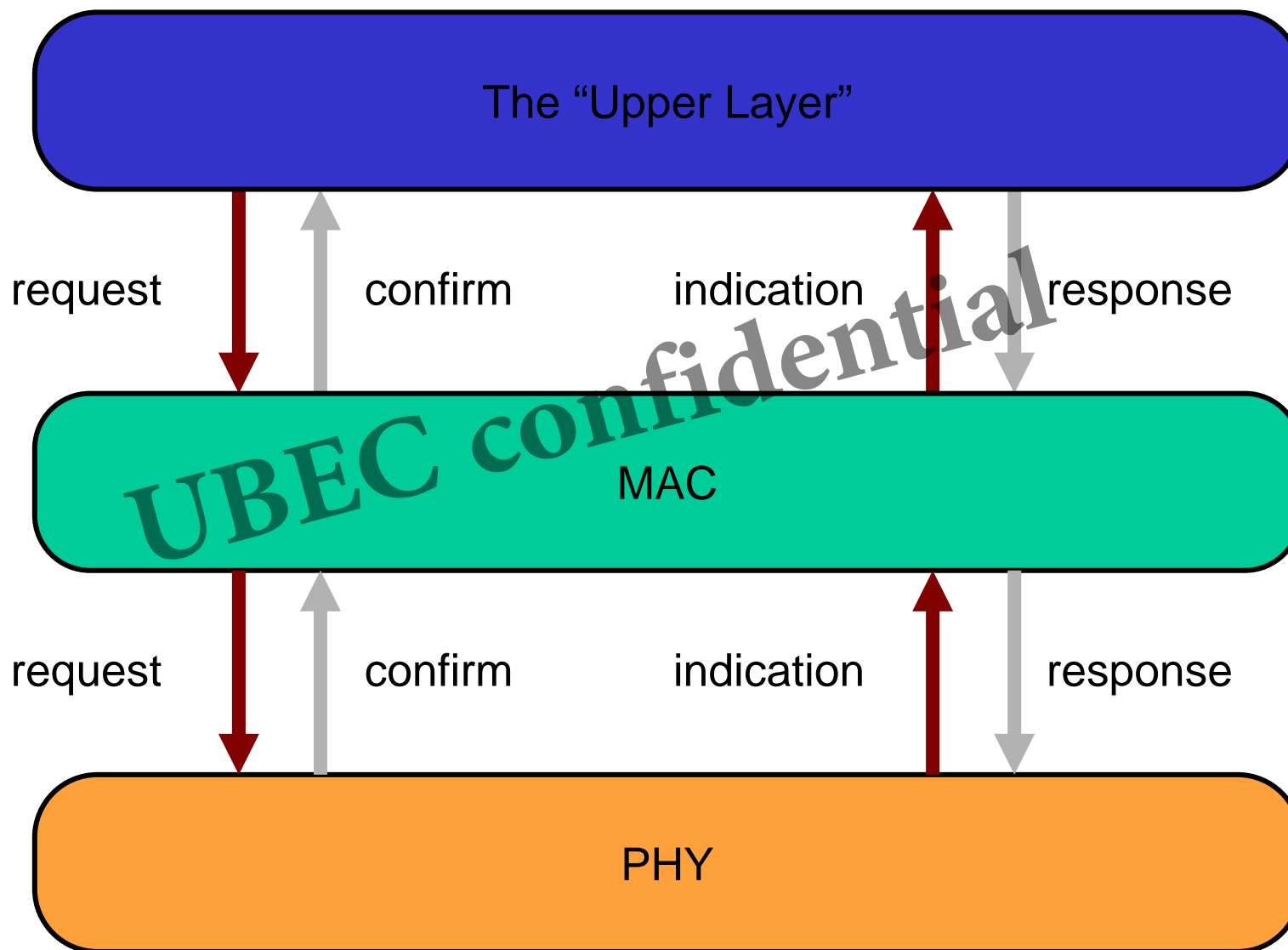
## ✚ Full name:

- *Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)*

## ✚ Keywords

- **PHY** - Physical Layer
- **MAC** - Medium Access Control
- **W** - Wireless





## The PHY Layer

**UBEC confidential**

Radio Frequency, RF, Wireless



- ✚ Frequency band
  - 2.4 GHz, 250kbps, 16 channels
  - 906 MHz, 40kbps, 10 channels
  - 868 MHz, 30kbps, 1 channel
- ✚ Codec
  - DSSS with BPSK/ O-QPSK

UBEC confidential

- ✚ Main function
  - Transmit packets
  - Receive packets and notify upper layer
- ✚ Support function
  - Energy detection
  - Transceiver On/Off
  - Switch channel

UBEC confidential

## The MAC Layer: Overview

What MAC Does

- Peer-to-peer connection

- Packets transmission & receiving control

What's MAC Not

- Topology (Star ? Tree ?)

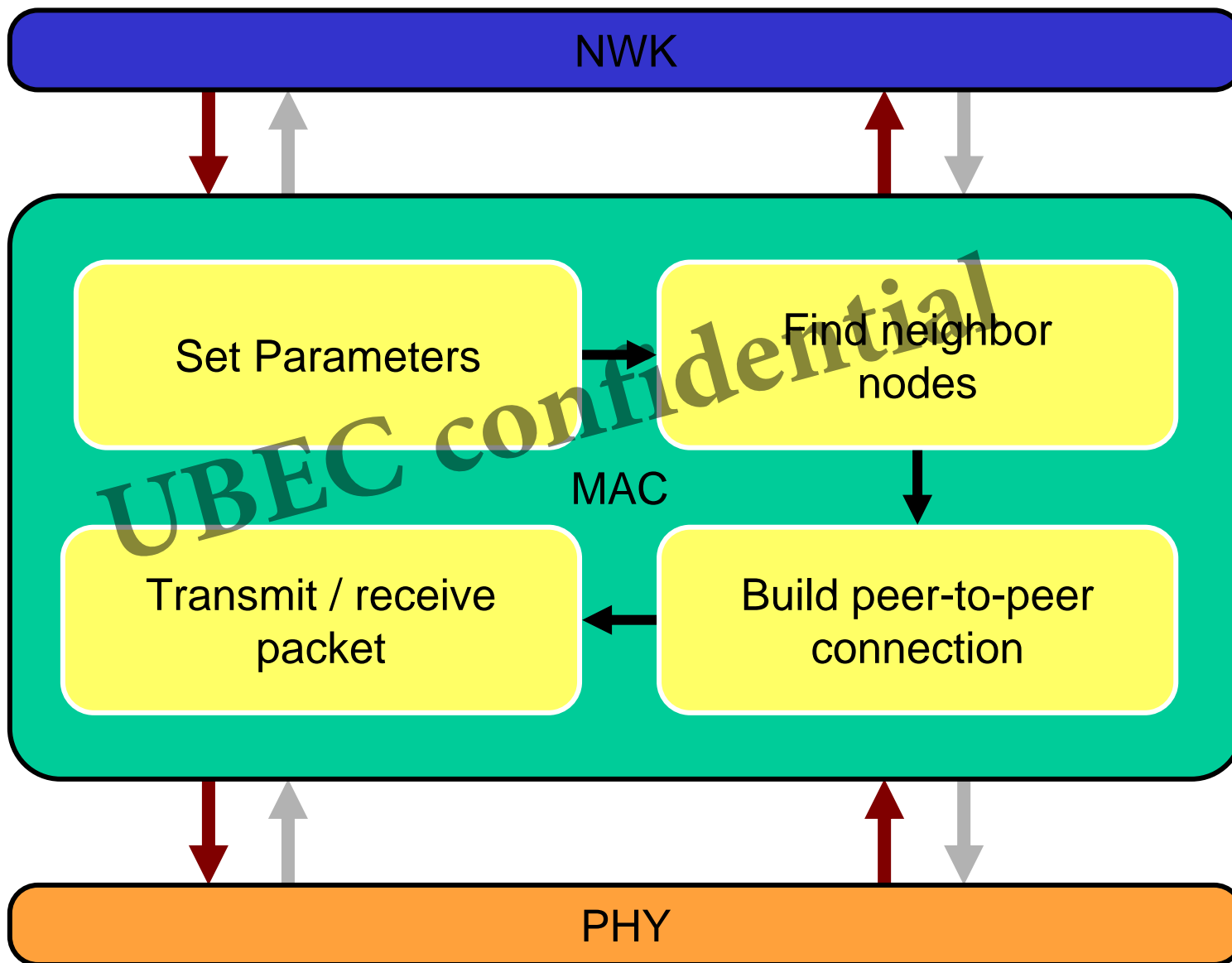
- Routing (What's the target ?)

- Decision (Shall I let this device join ?)

- Fragmentation / De-fragmentation

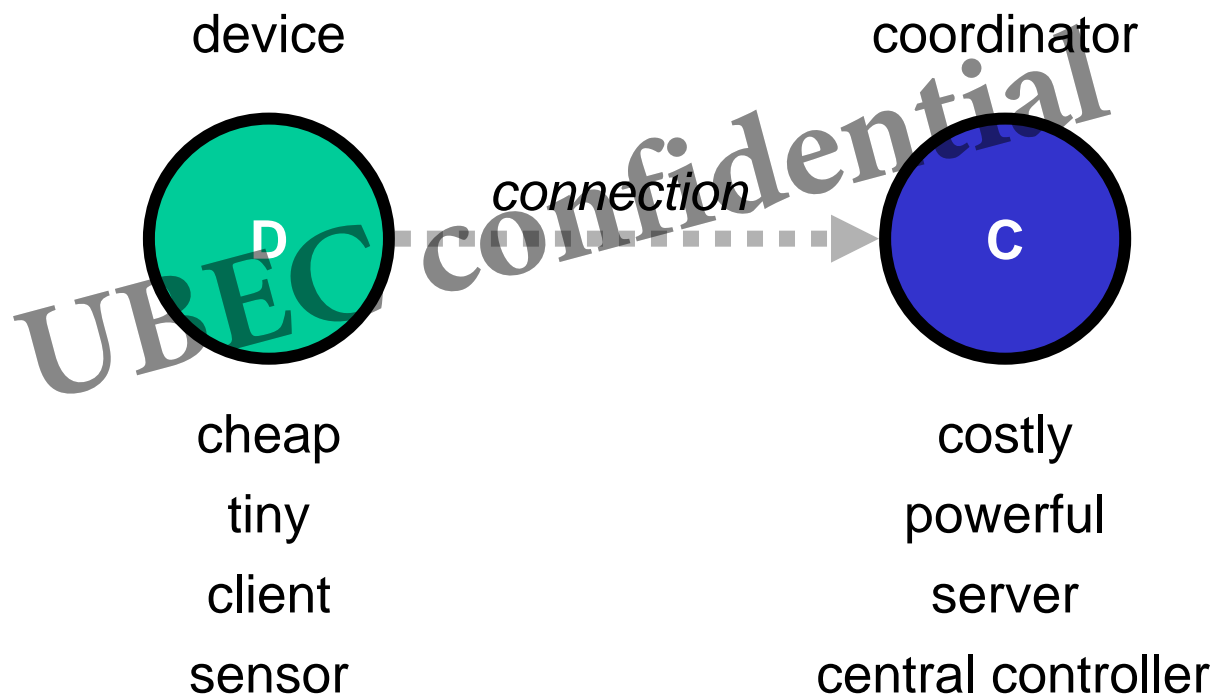
- Roaming

# 4 Signal Processing Steps

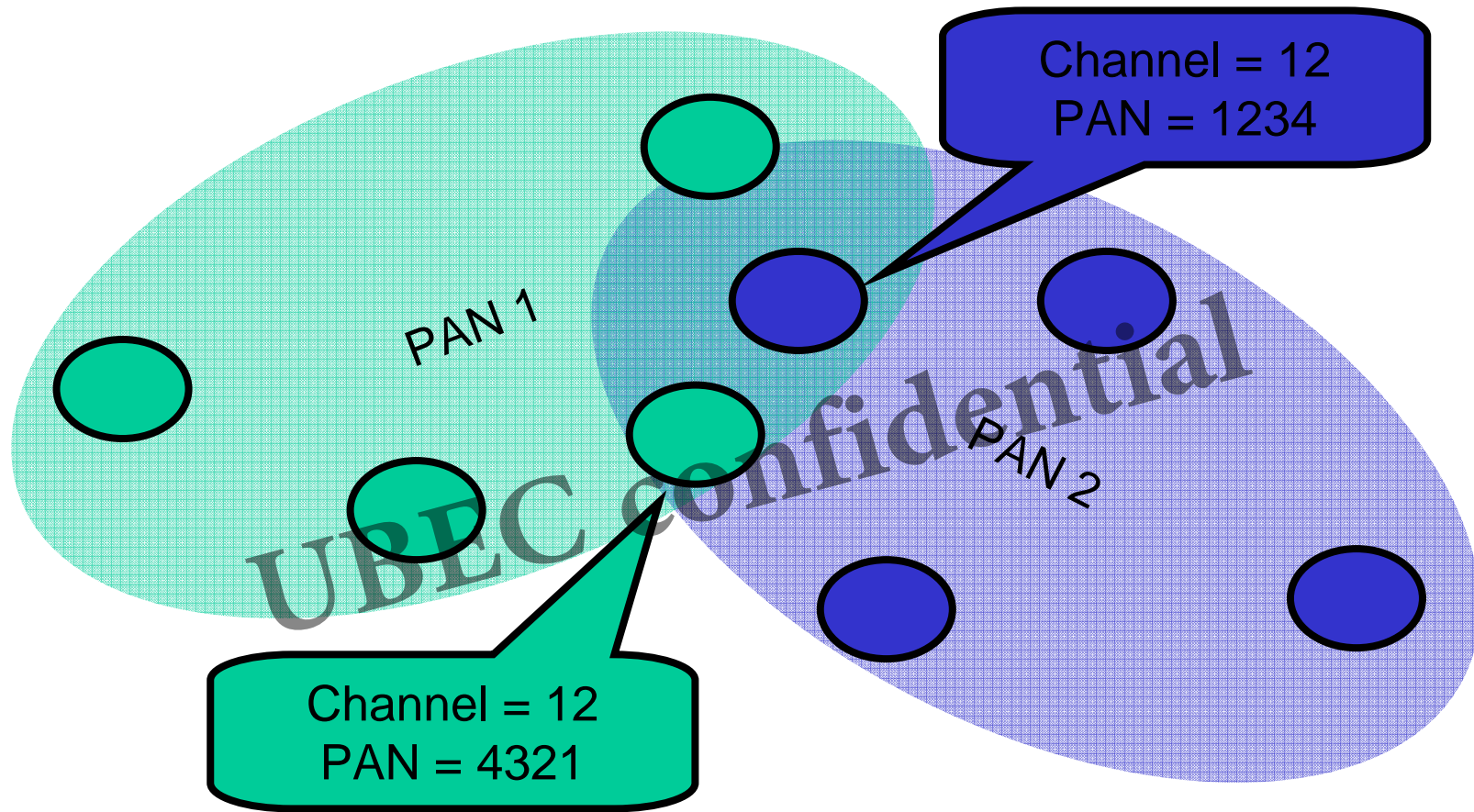


*UBEC confidential*

## MAC Features

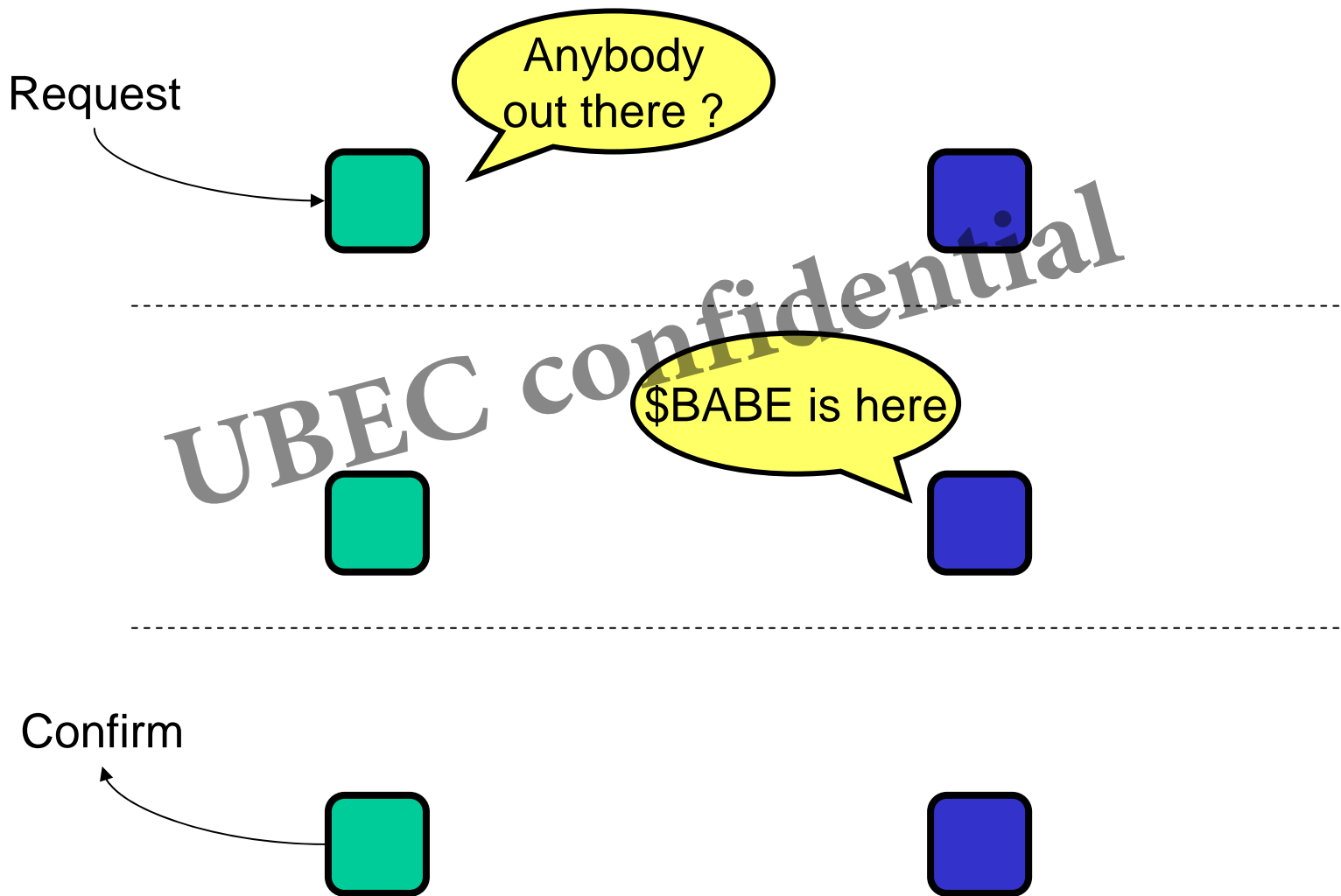


# What is "PAN" ?



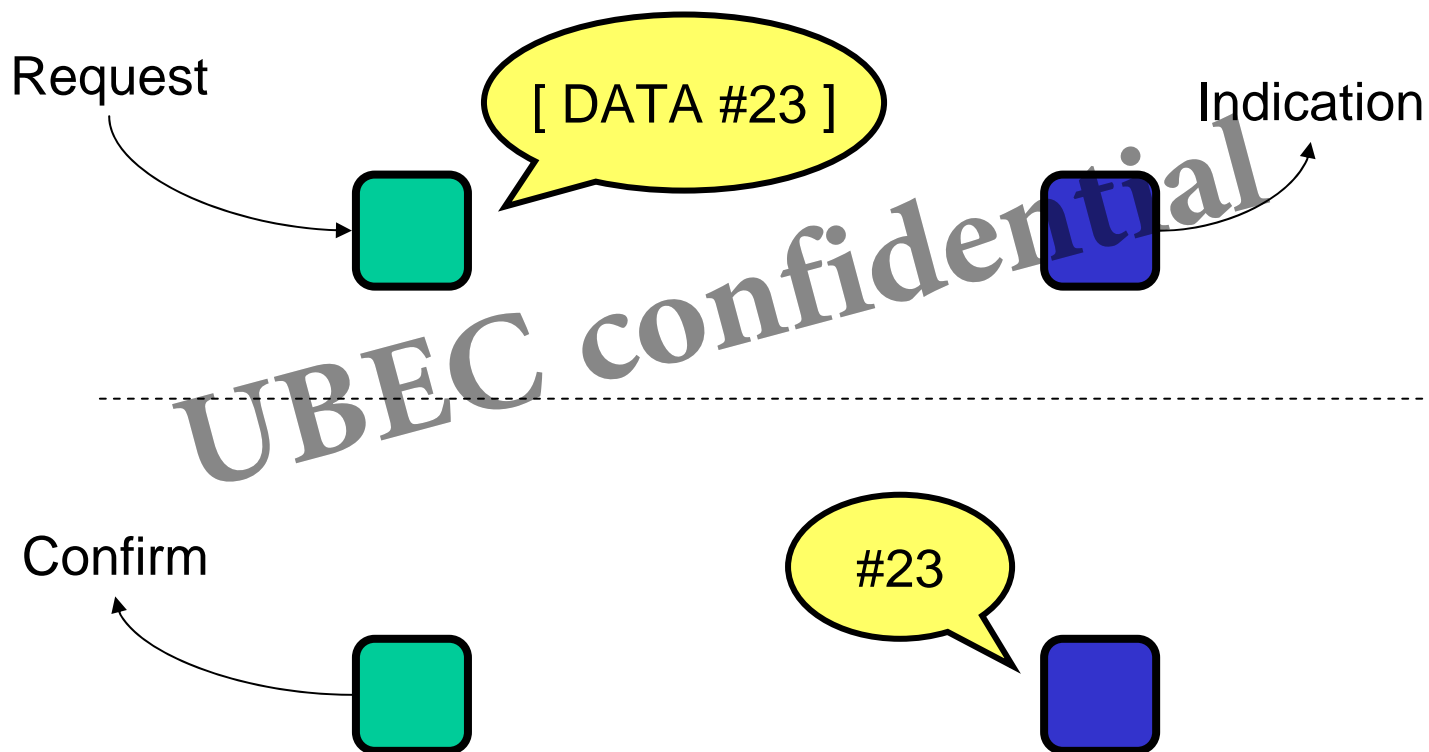
To be continued...

- ✚ Channel
  - 8 bits, 0x0B ~ 0x1A
- ✚ PAN ID (Personal Area Network)
  - 16 bits, 0x0000 ~ 0x3FFF (*Defined by ZigBee*)
- ✚ Long Address
  - 64 bits
  - Fixed, universal unique
- ✚ Short Address
  - 16 bits
  - Runtime, unique in each PAN

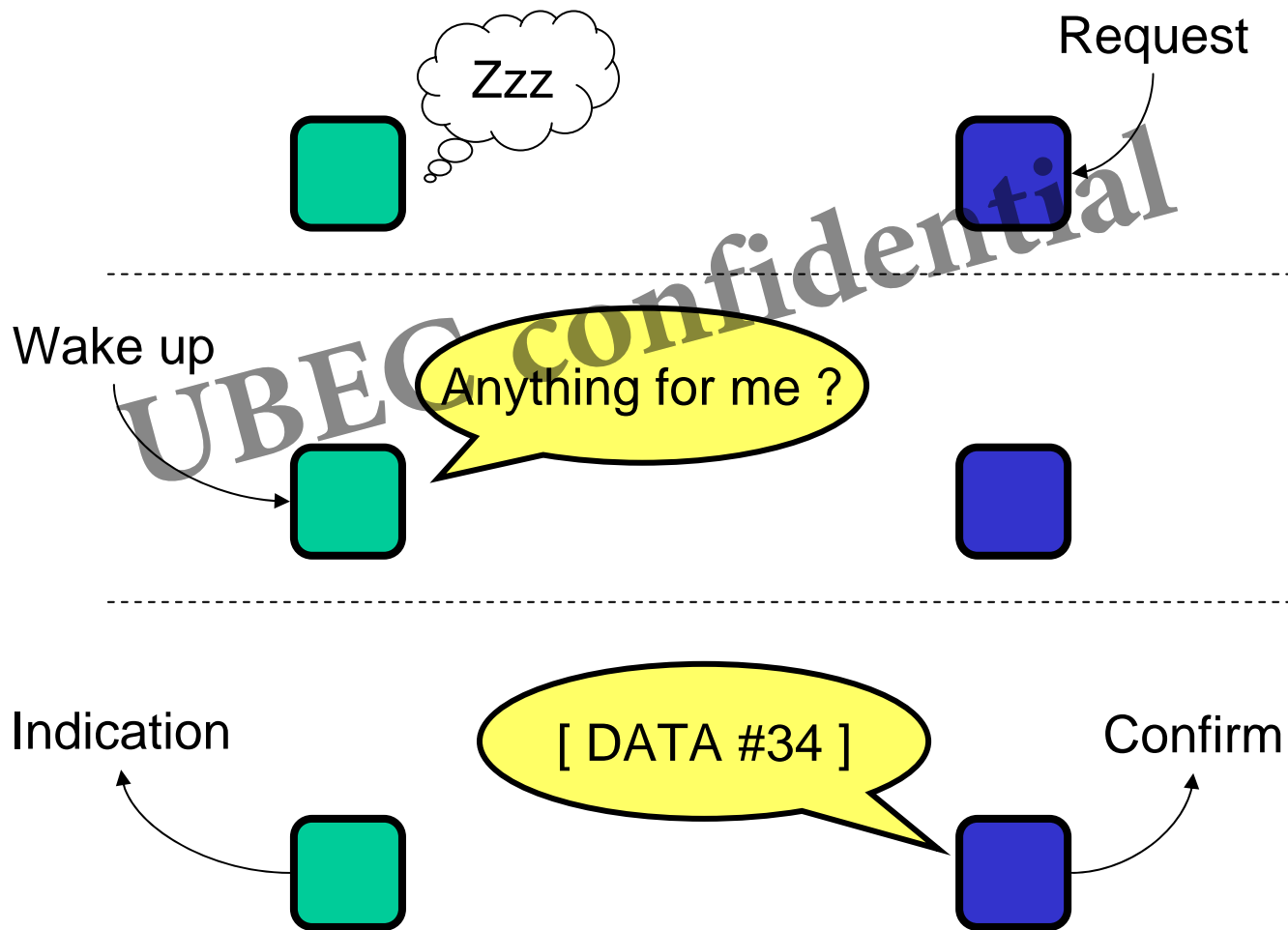




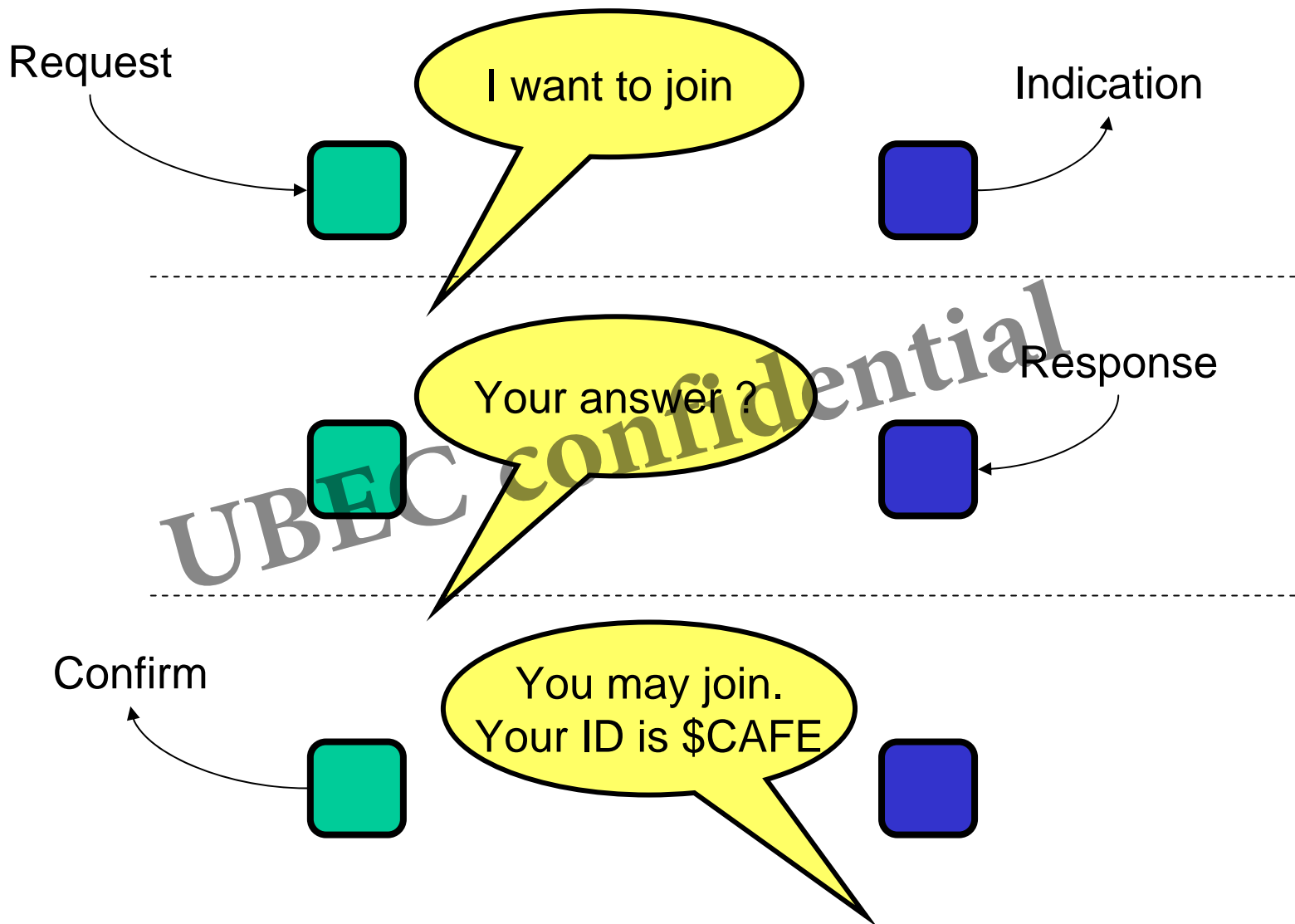
# MAC Function: Transmission (Direct)



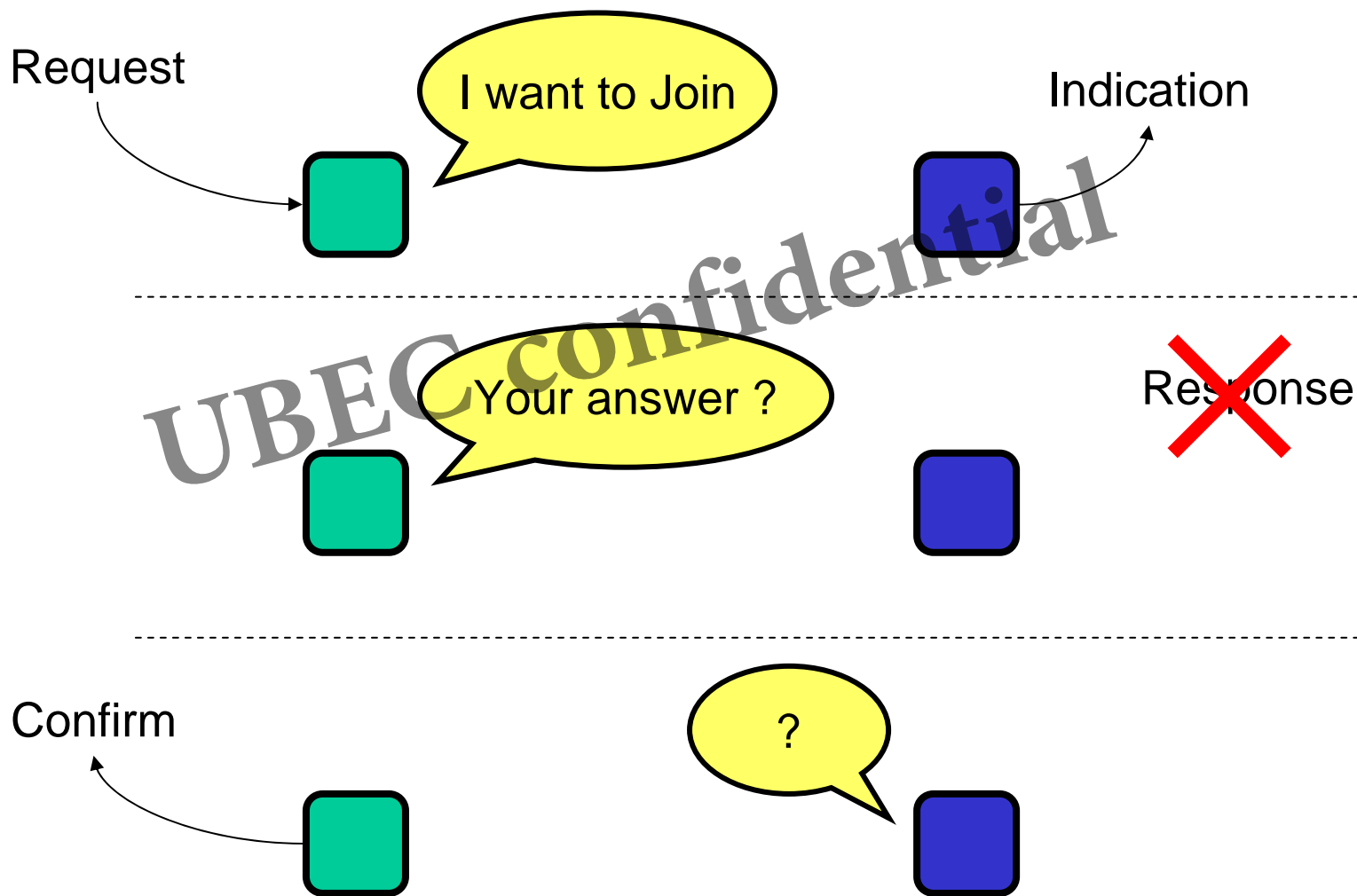
# MAC Function: Transmission (Indirect)



# MAC Function: Associating Process

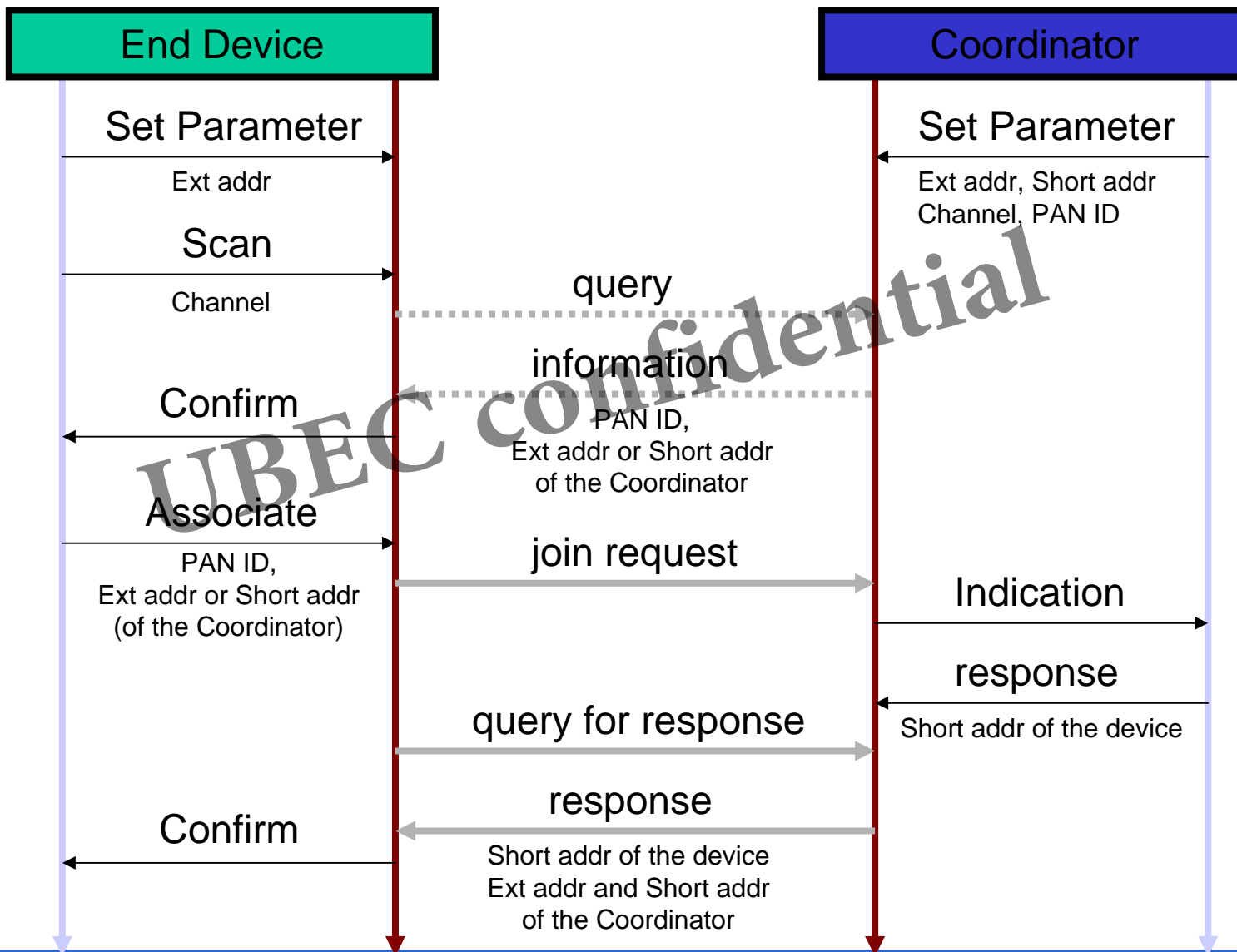


To be continued



- ✚ *Extended address*
- ✚ Channel
- ✚ PAN ID
- ✚ Short address
- ✚ Short / extended address of parent
- ✚ Ability to let other child join
- ✚ *Buffer hold time*
- ✚ *Power saving on / off*

UBEC confidential



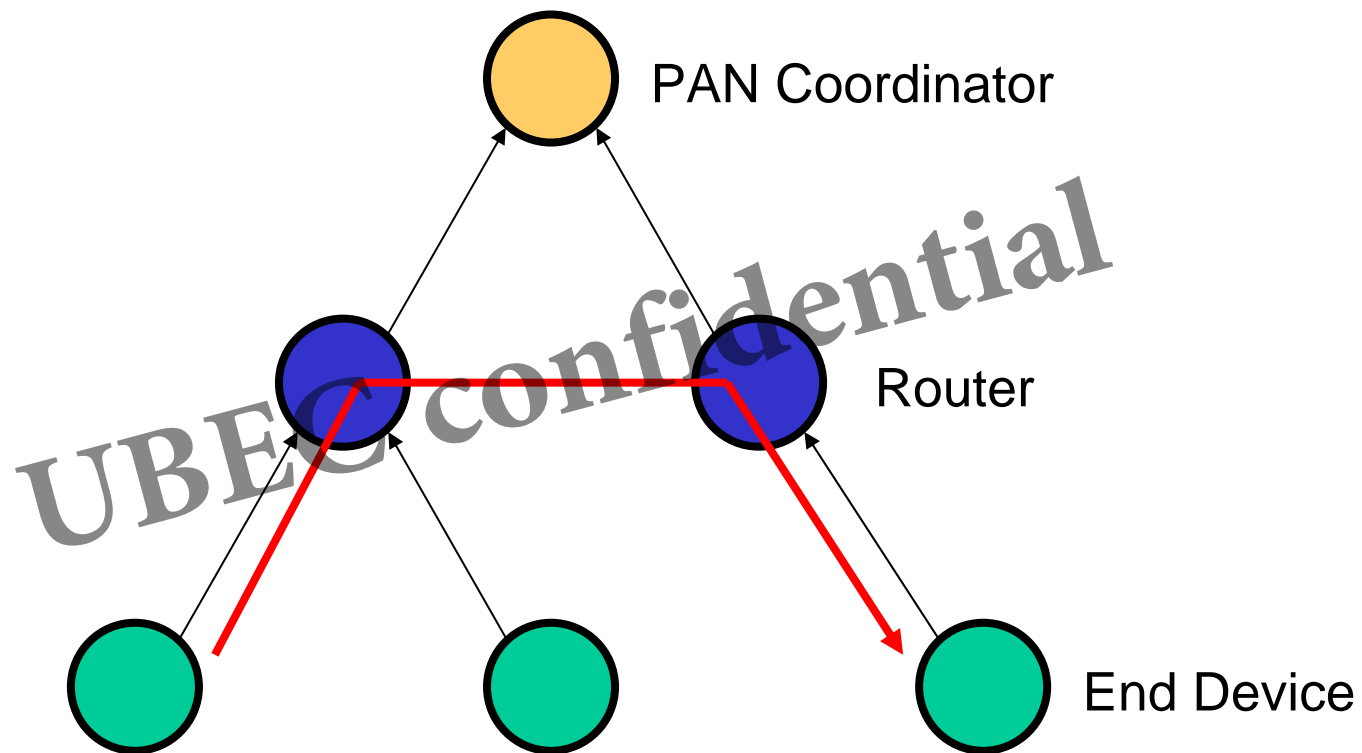
## MAC Network

**UBEC confidential**

Myth and Truth

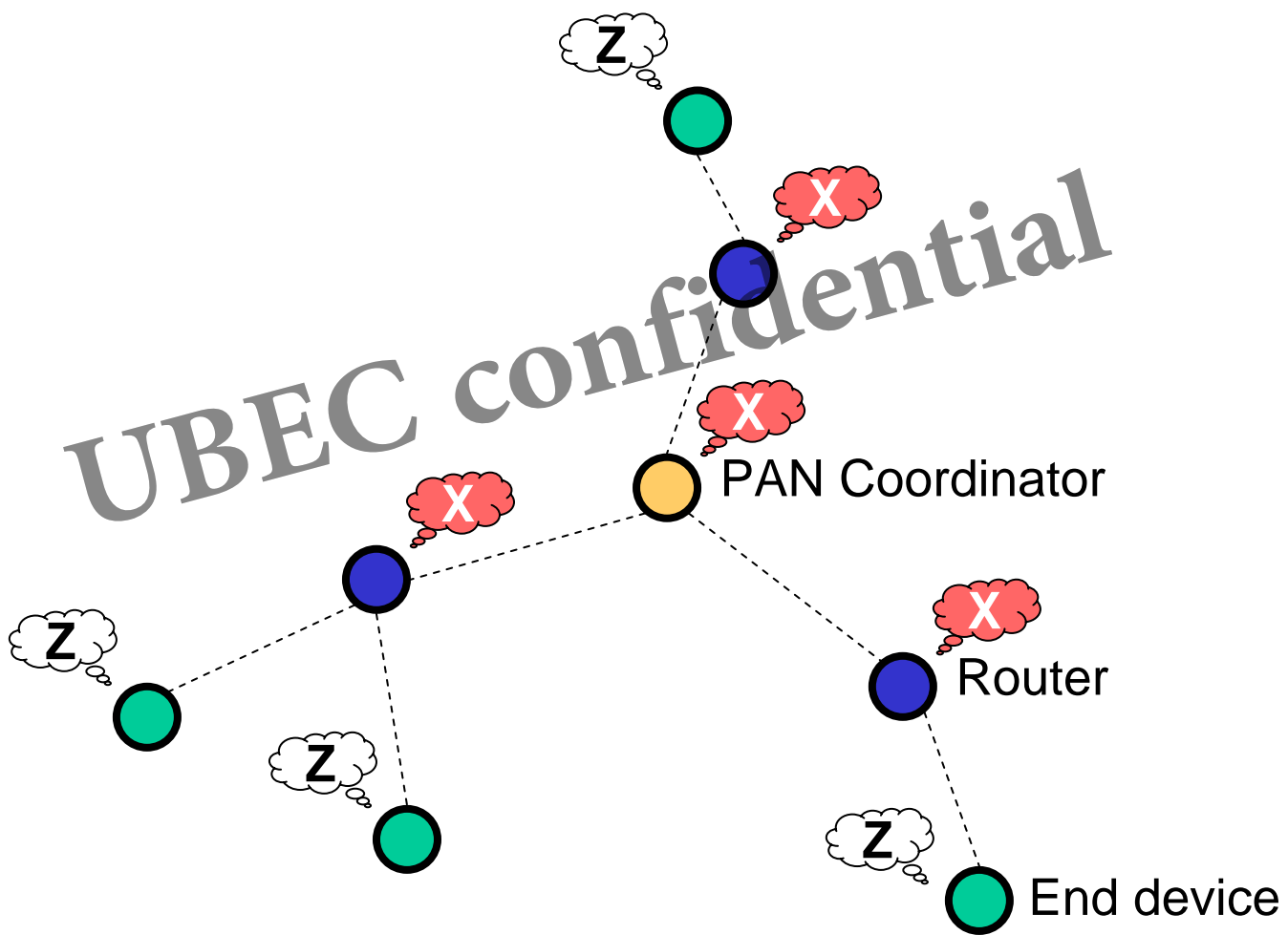
- ✚ Myth: TX rate = 250 kbps
  - Truth: about 1xx kbps for peer-to-peer connection
- ✚ Myth: Power saving
  - Truth: performance is a trade-off by your application
- ✚ Myth: The Protocol is simple
  - Truth: ...., and so is the MCU.
- ✚ Myth: ZigBee should be a cheap solution
  - Truth: Yeah, but just for the end-devices.





- ✚ Active Power Consumption
  - MCU = 25mA
  - UZ2400 = 20mA
- ✚ Sleep Power Consumption
  - MCU = 1.5uA
  - UZ2400 = 2uA
- ✚ Sleep-to-Active time
  - About 10ms

UBEC confidential



UBEC confidential

## ZigBee does NOT cover...

- ✦ Building a completely battery-powered network.
- ✦ Battery-powered device that send packets every second.
- ✦ Ad-hoc network without a coordinator.
- ✦ Roaming and hand-over.
- ✦ Data fragment and recovery.
- ✦ Full duplex heavy load communication.

UBEC confidential

IEEE 802.15.4

**UBEC confidential**

Session 2:

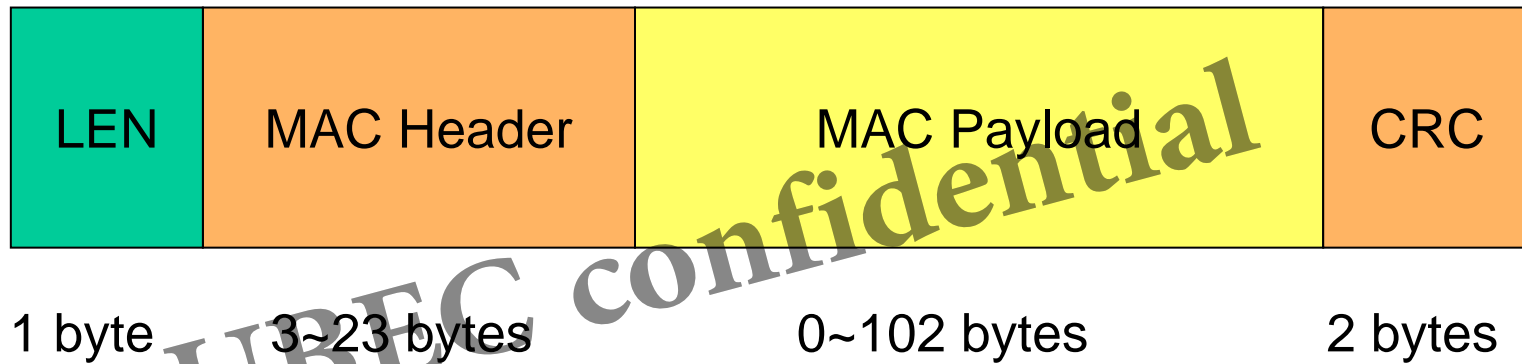
MAC Packets

## MAC Frame Format

**UBEC confidential**

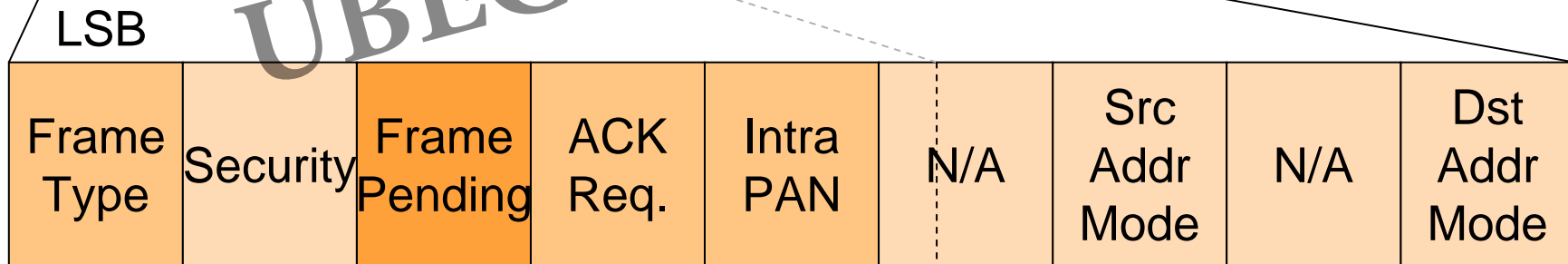
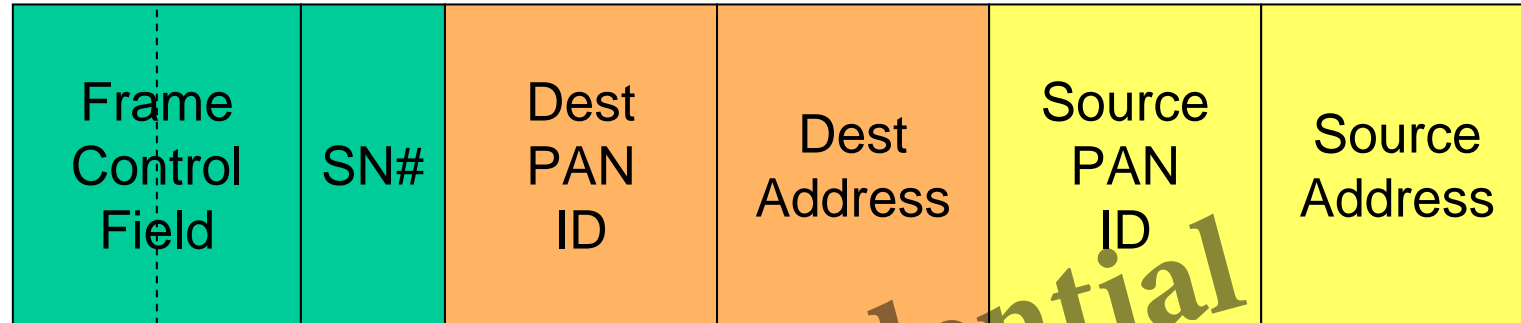
Overview

# MAC Frame Format



# MAC Header

Byte:            2            1            0/2            0/2/8            0/2            0/2/8



bit: 3            1            1            1            1            3            2            2            2

- 00: skip address field
- 10: short address
- 11: long address



# 4 Types of MAC Packets



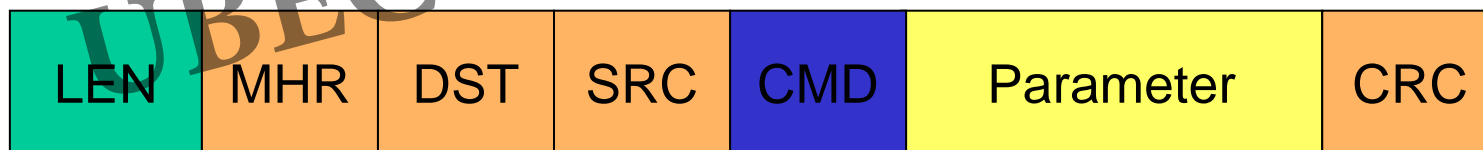
3



7~23

0~102

Command



7~23

1

0~9

Beacon



7 / 13

4~83

0~52

## Medium Access Control

**UBEC confidential**

CSMA/CA

- ✦ Carrier **S**ense **M**ultiple **A**ccess with **C**ollision **A**voidance
  - RF device can not listen while transmitting.
  - Collision avoidance: listen before transmit.
  - Every packet must follow CSMA/CA except for some exceptions:
    - ACKs
    - *Data packet responses the polling*
    - *Beacons in Beacon Network*
    - *Data in GTS*

# CSMA/CA Time Line (2.4G)

1	2 3	4	5	6	7
RX	R→T	TX	T→R	RX	No Act
320us~2.56ms	192us	1byte = 32us	192us	672us	192us / 640us

1. Wait random duration 320us ~ 2.56ms. Basic unit is 320us, called **Backoff**.
2. Check the channel status. If channel is occupied, repeat waiting with maximum duration doubled. Repeat 3 times at most, or report **Channel Access Failure**.
3. Tell the RF circuit to prepare transmission.
4. Transmitting data. 1 byte = 32us.
5. RF circuit automatically switch to receiving mode for ACK.
6. Wait 672us for ACK. If ACK is not received, repeat the whole procedure. At most 3 times, or report **No ACK**.
7. Pause before sending next packet. If the packet length is greater than 18 bytes (including header and CRC), it shall pause for 640us.
8. If ACK is not required, step 5 & 6 can be skipped.

- ✦ **Destination PAN** must match or equal to \$FFFF (broadcast)
- ✦ **Destination address** must match with the device's short address or extended address or equal to \$FFFF (broadcast)
- ✦ If the packet is a beacon packet, the **Source PAN** must match to or equal to the device PAN and \$FFFF
- ✦ If the packet is a data packet, when the destination field is missing, only **PAN Coordinator** can receive it.

- ✦ \$FFFF, reserved value for the short address, means broadcast.
- ✦ \$FFFE, another reserved value for short address, means “short address will not be used”.
- ✦ The addresses for the destination and the source of data packet are determined by the upper layer.
- ✦ Some MAC command only allow extended address in the source address field.
- ✦ Besides, only the short address is allowed as source address when the address of a device is shorter than \$FFFE,

## MAC Frame Format

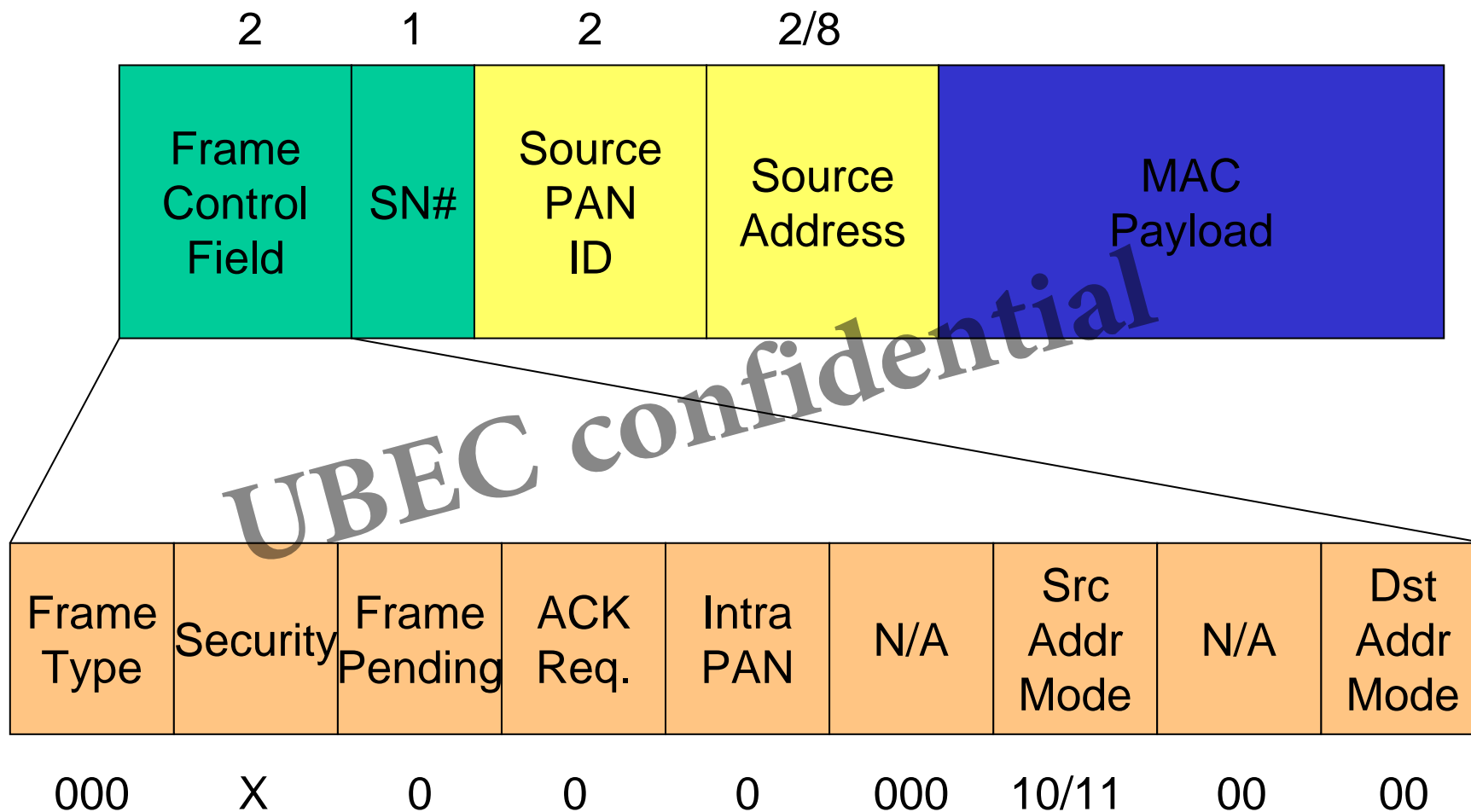
**UBEC confidential**

Beacon Packets

- ✚ A special packet for node announcement
  - PAN ID & address
  - Ability to let other devices join
  - Buffered data
  - Others
- ✚ Beacon is used to
  - Inquire node information during Scan
  - Notify the end devices that packets are in pending mode
  - Synchronization (beacon network)

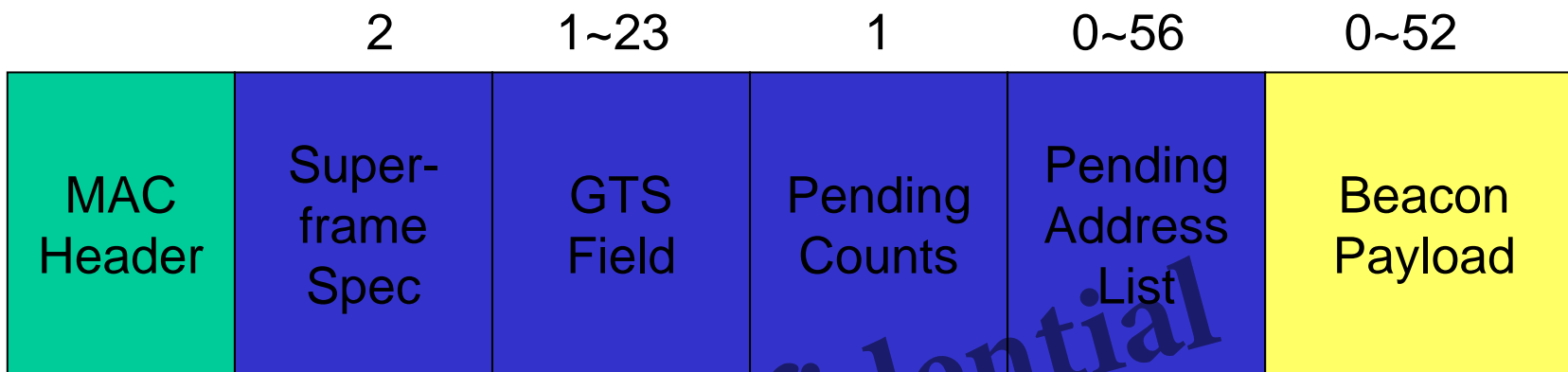


# MAC Beacon: MAC Header



Typical values of FCF: \$00, \$08/0C

# MAC Beacon: MAC Payload



UBEC confidential

Beacon Order	Super-frame Order	Final CAP Slots	Battery Life Ext.	N/A	PAN Coordinator	Permit Associate
--------------	-------------------	-----------------	-------------------	-----	-----------------	------------------

bits: 4	4	4	1	1	1	1
\$F	\$F	\$F	0	0	X	X

## MAC Frame Format

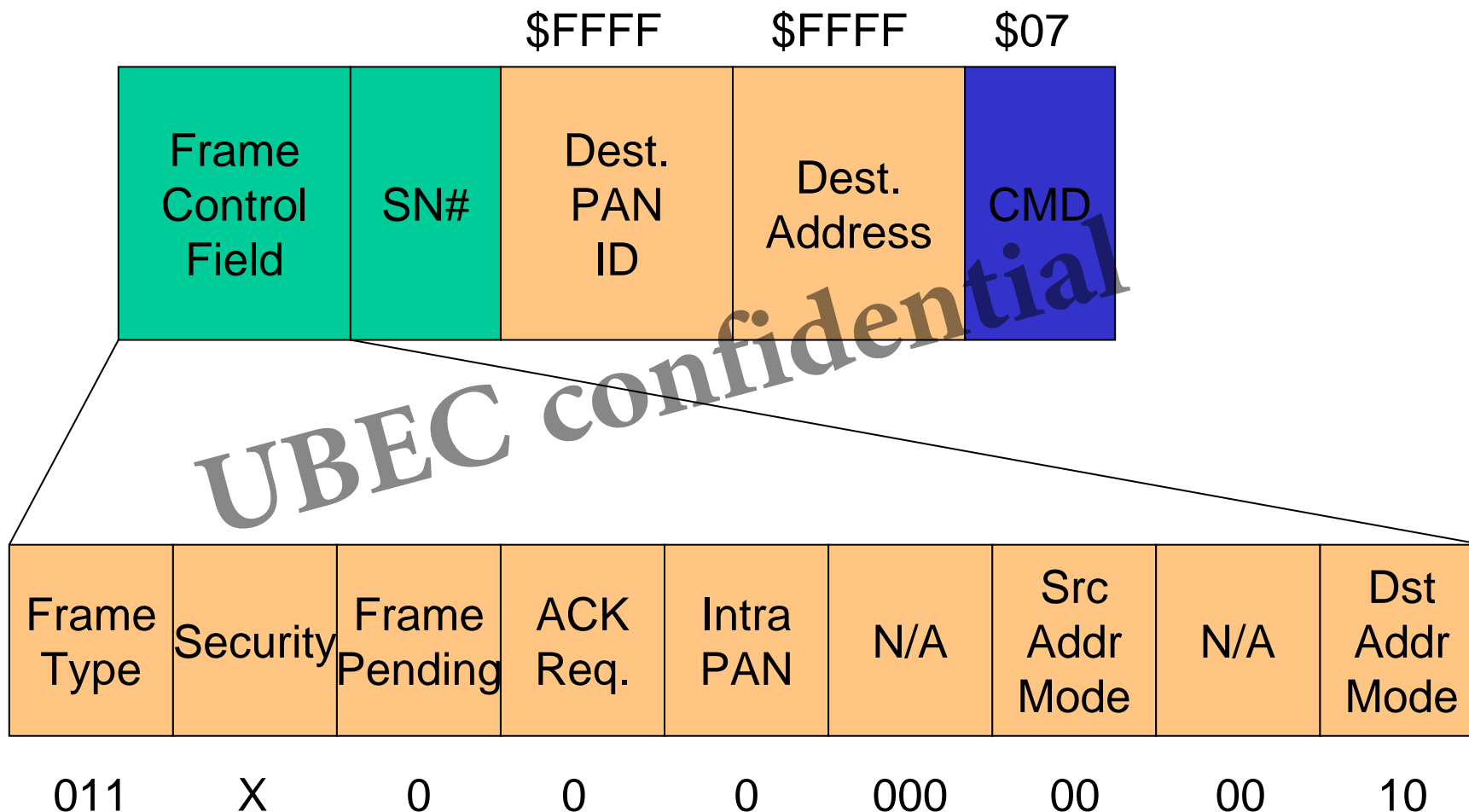
**UBEC confidential**

## MAC Commands

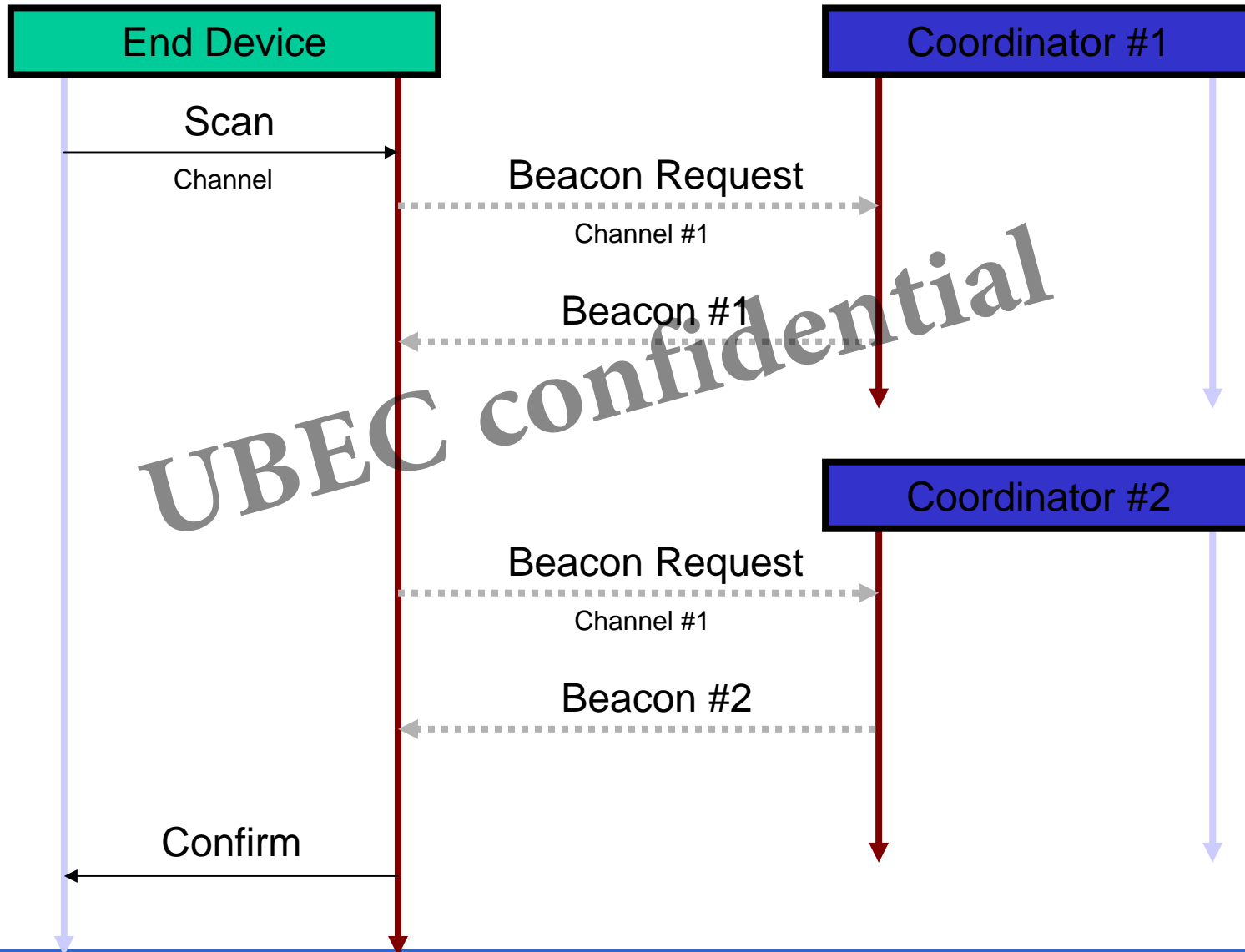
- ✚ Control all MAC behaviors
  - Beacon Request
  - Association Request
  - Association Response
  - Data Request
  - Orphan Notification
  - Coordinator Realignment
  - Disassociation Notification
  - PAN ID Conflict Notification
  - GTS Request

UBEC confidential

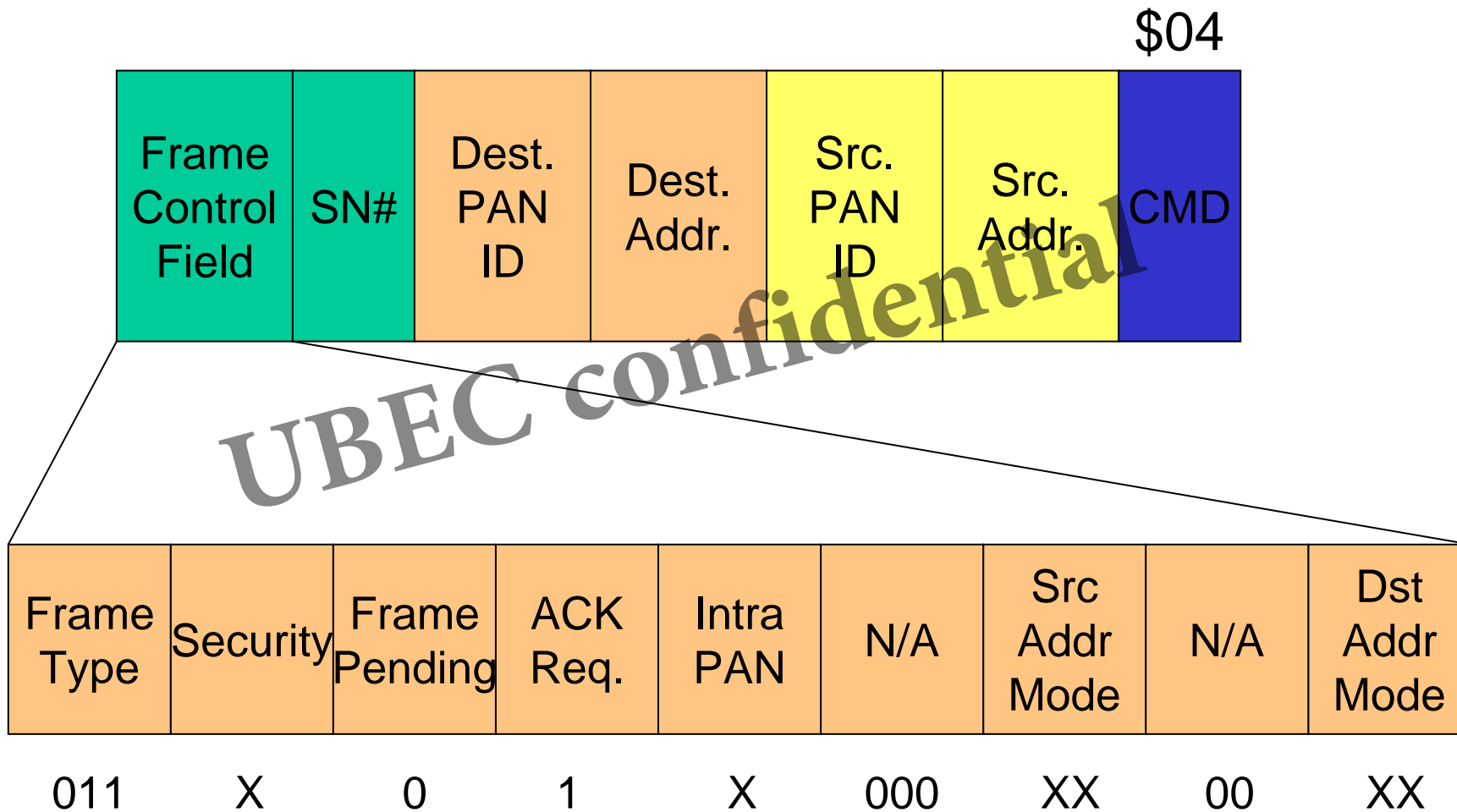
# Beacon Request



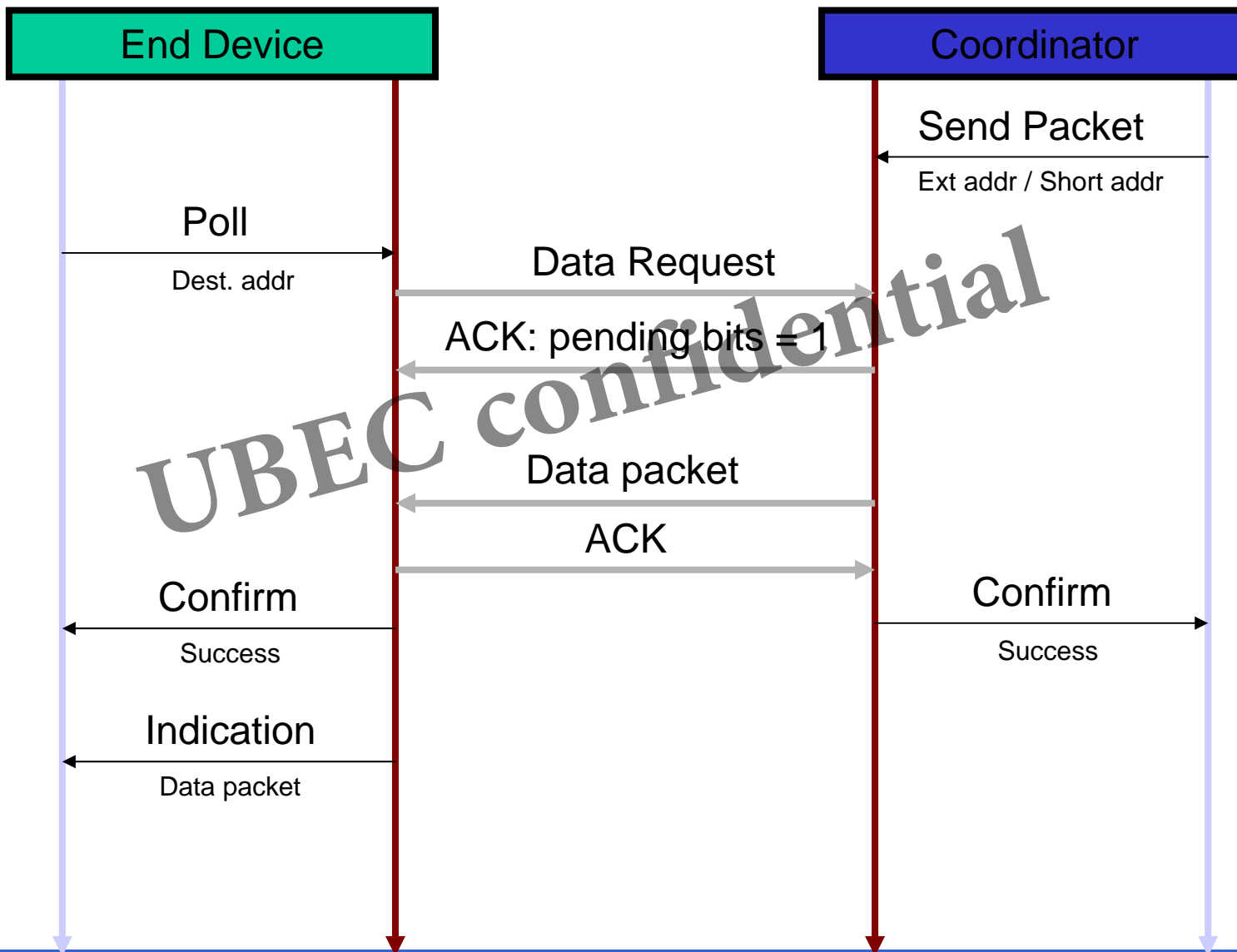
Typical values of FCF: \$03, \$08



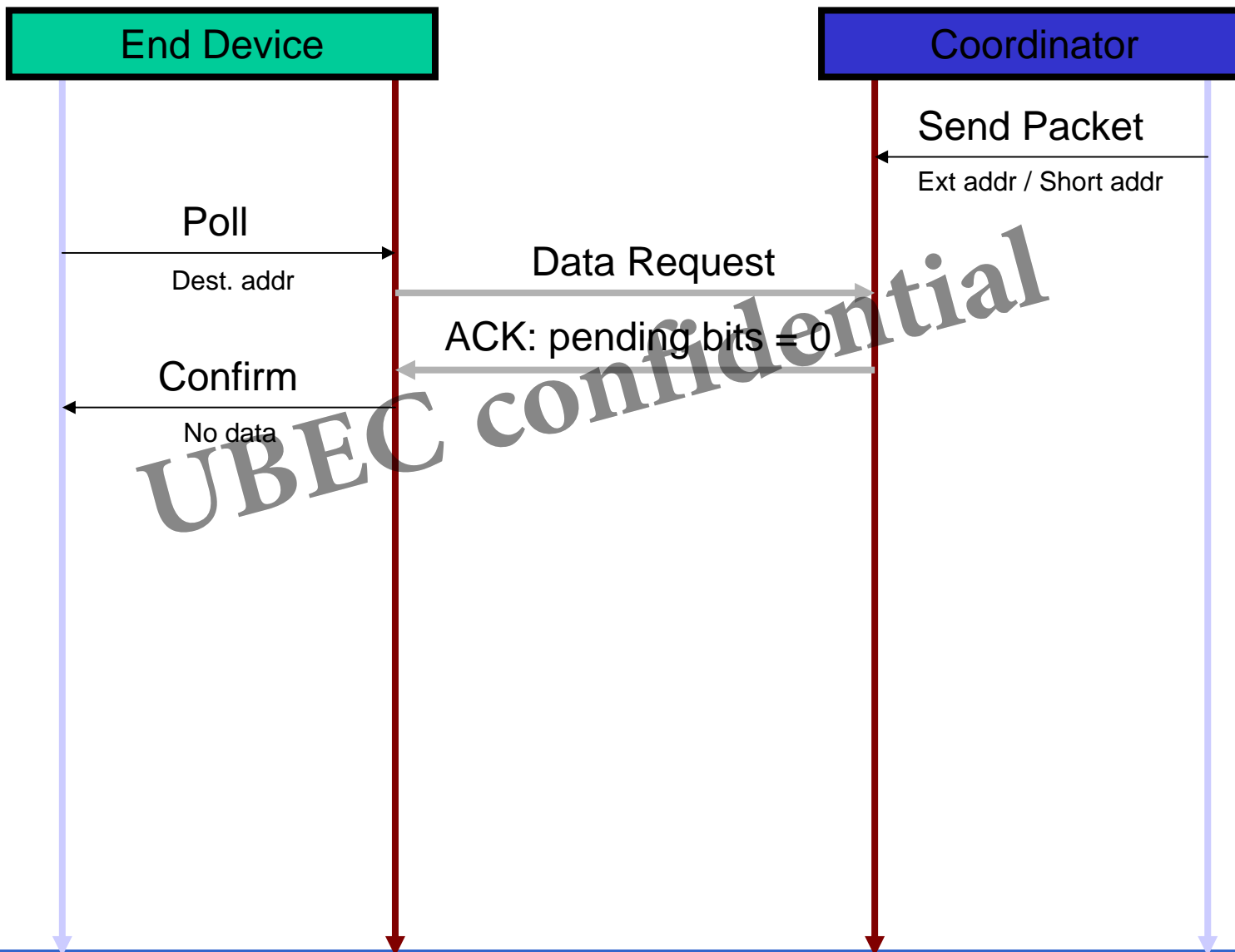
UBEC confidential



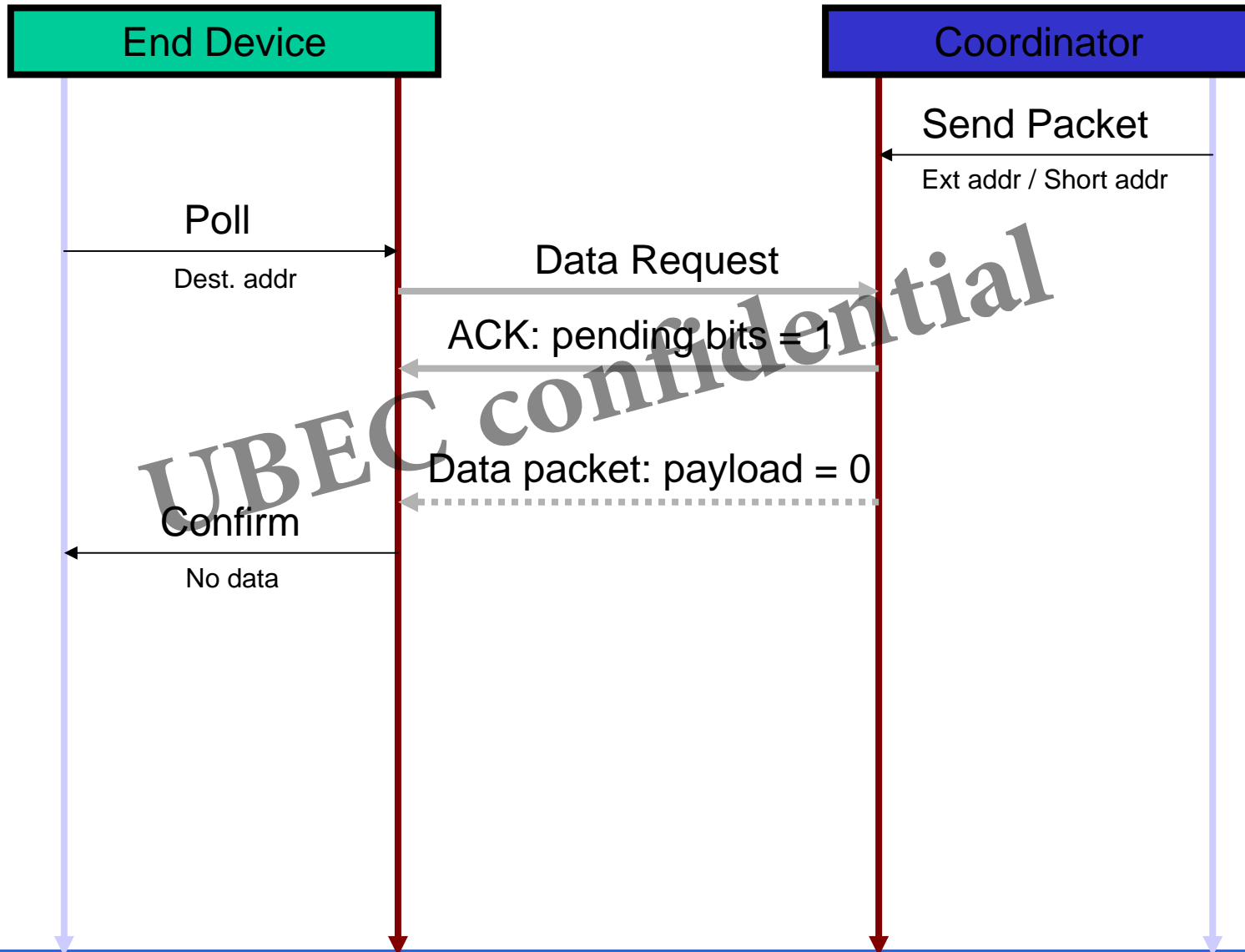
UBEC confidential



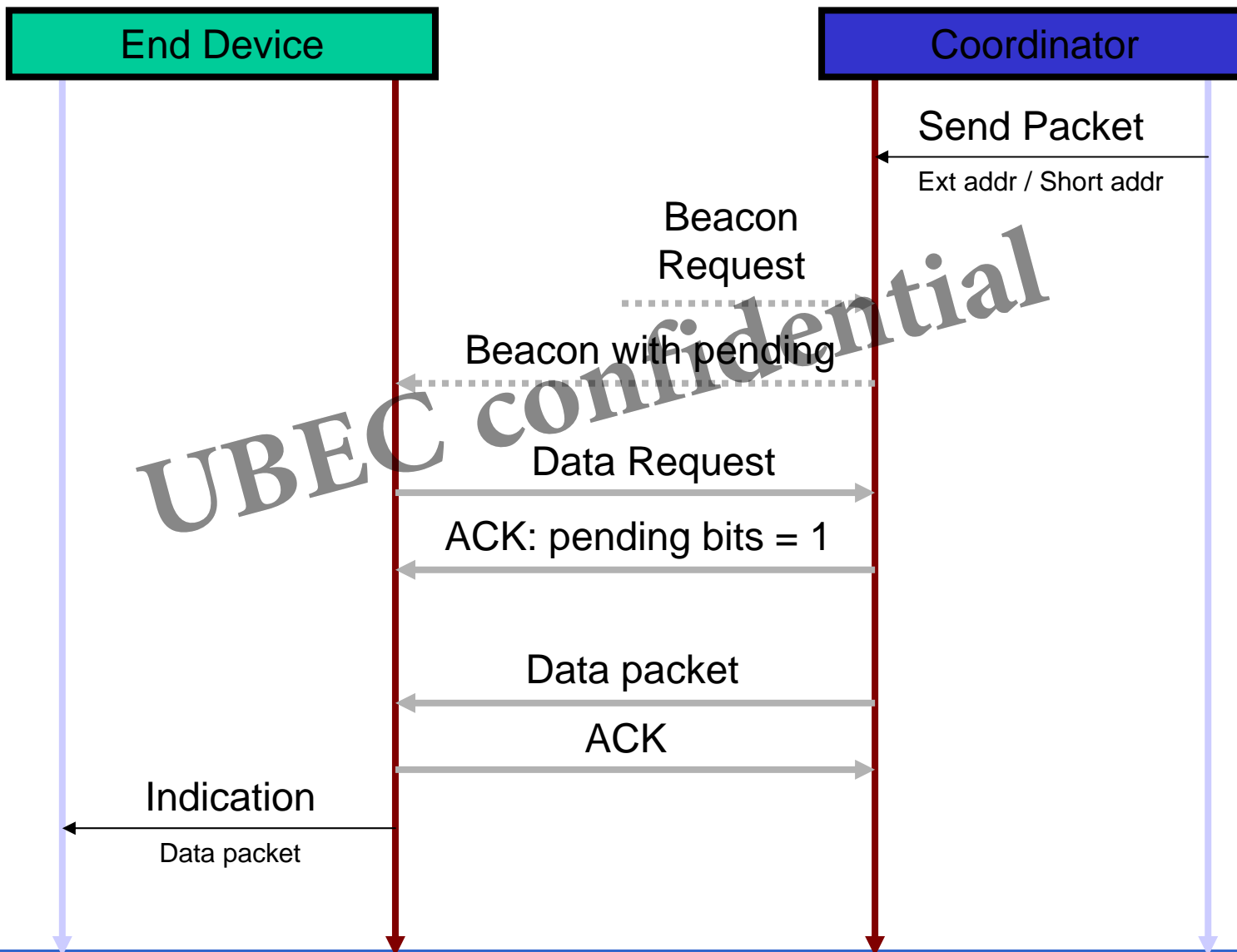




# Polling, no Data

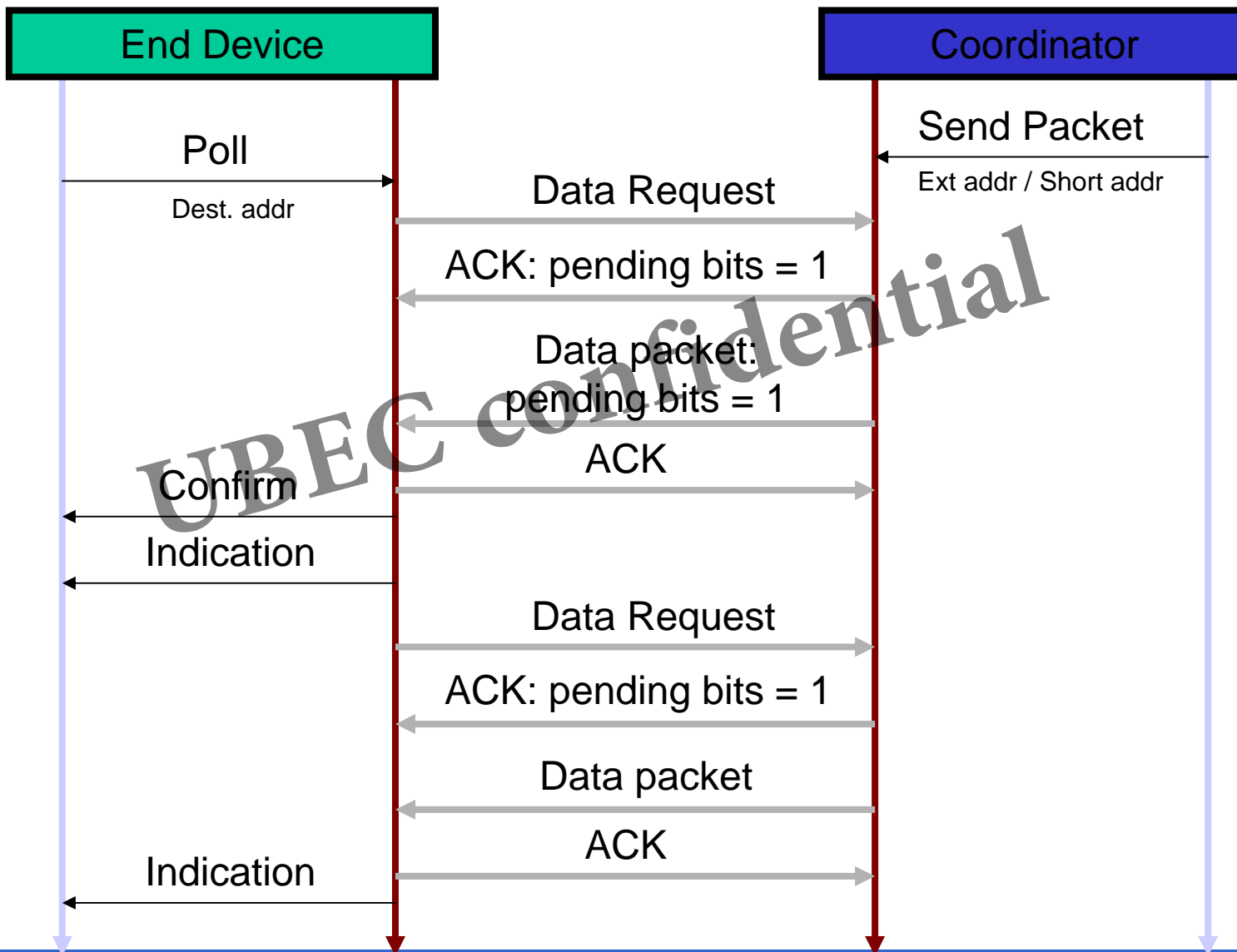


# Polling, Automatic

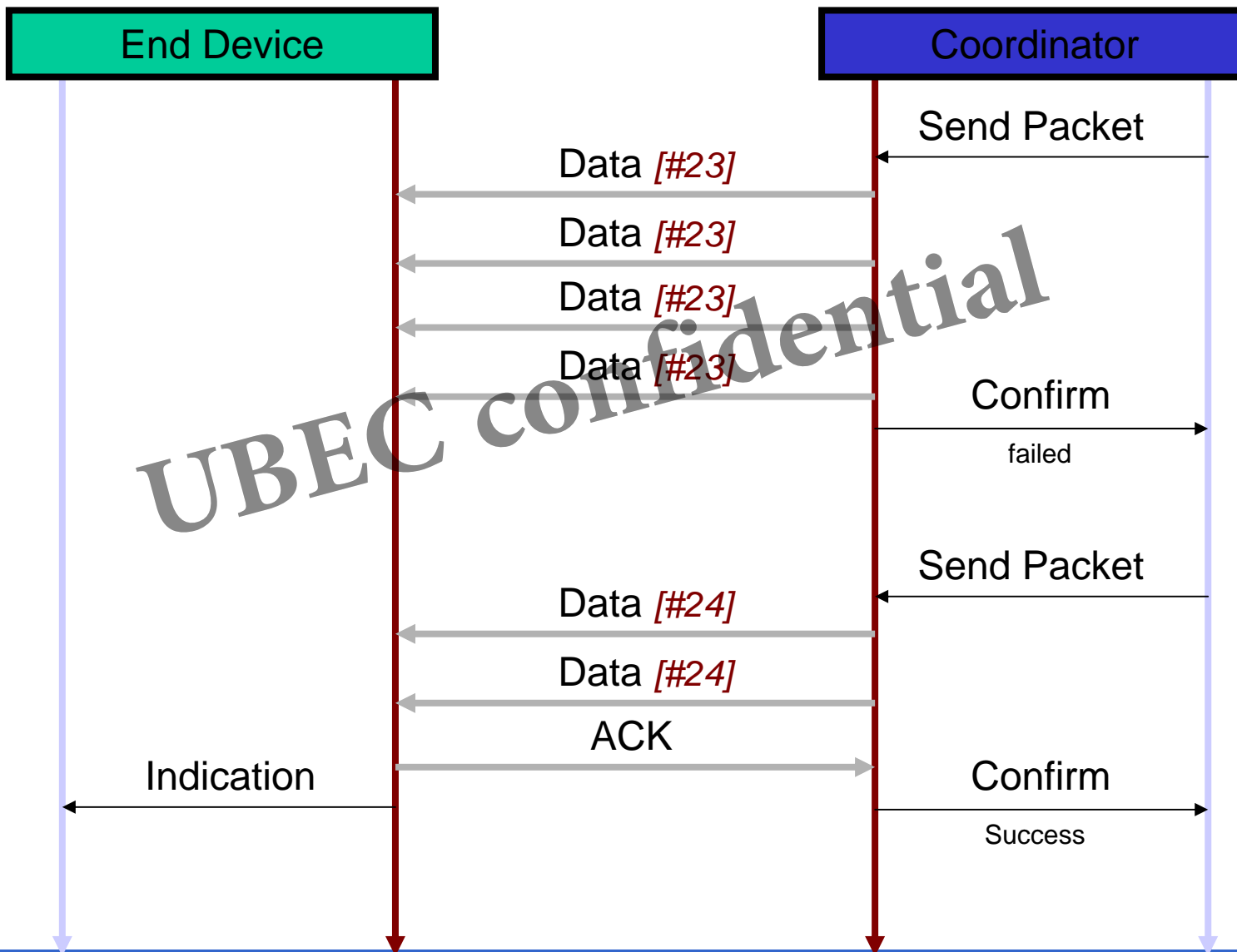


UBEC confidential

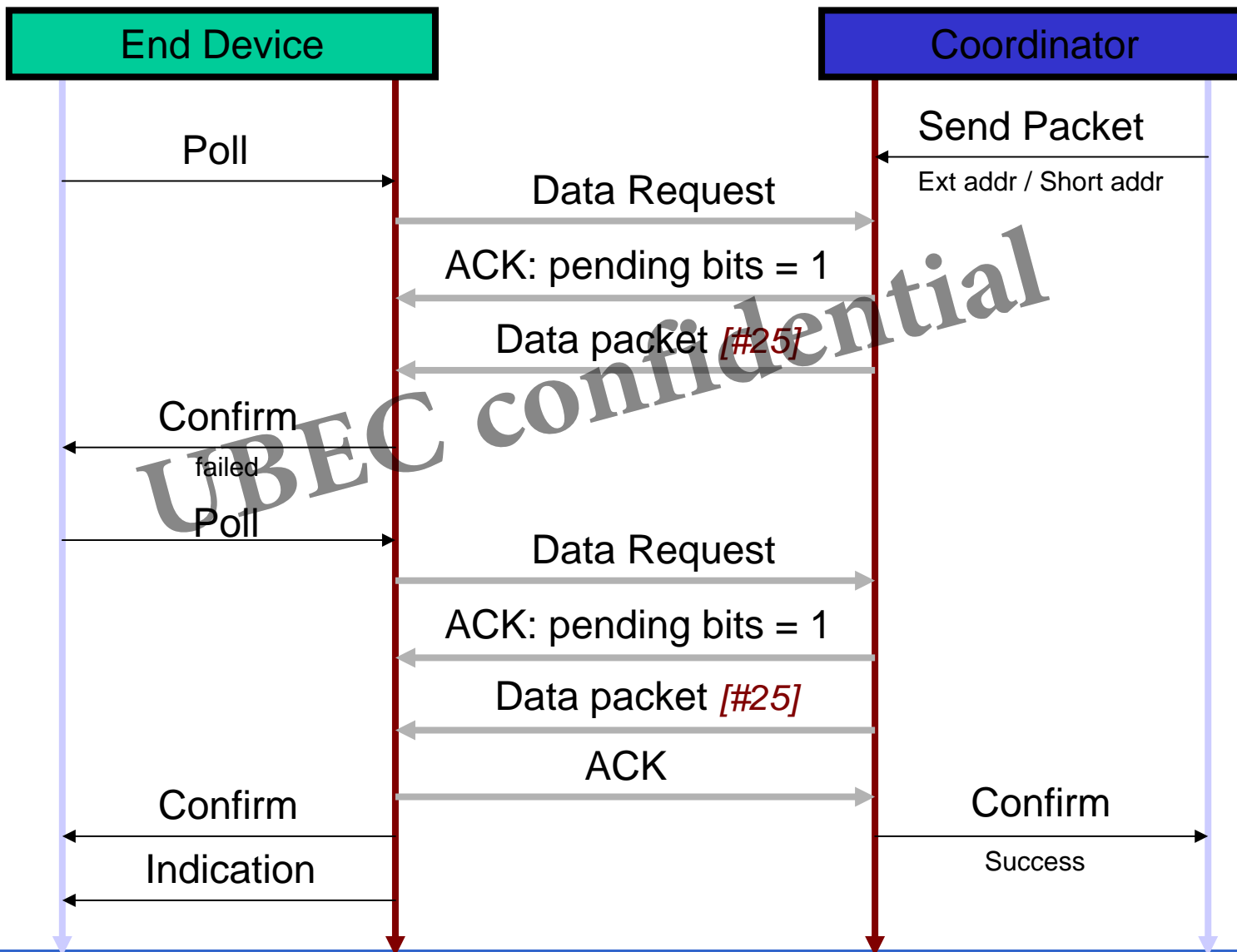
# Polling, Continuous



# Direct v.s. Indirect: retransmit



# Direct v.s. Indirect: retransmit



# Associate Request

011	X	0	1	0	000	11	00	11/10
Frame Type	Security	Frame Pending	ACK Req.	Intra PAN	N/A	Src Addr Mode	N/A	Dst Addr Mode

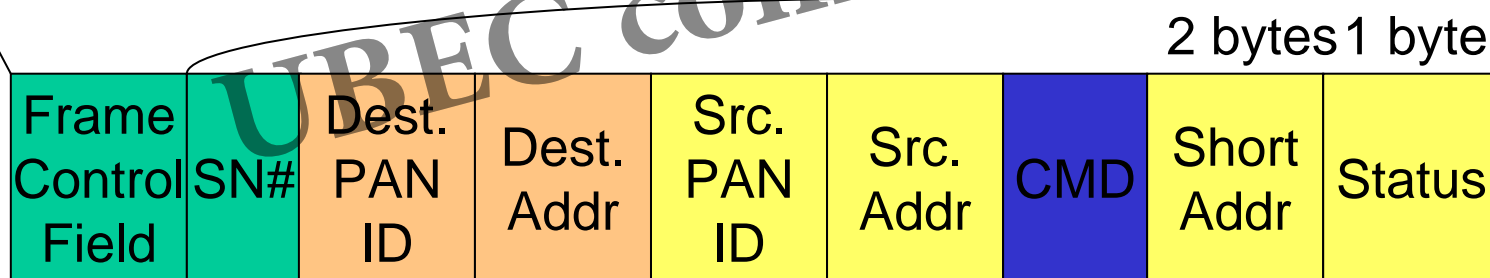
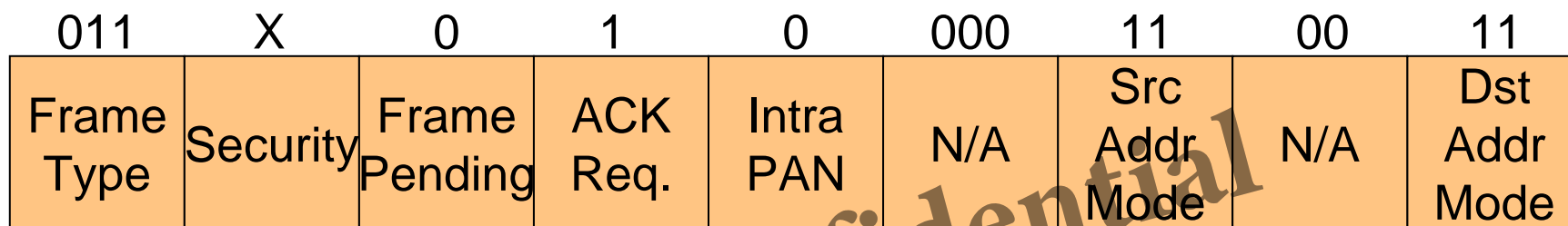
Frame Control Field	SN#	Dest. PAN ID	Dest. Addr	Src. PAN ID	Src. Addr	CMD	Cap-ability
---------------------	-----	--------------	------------	-------------	-----------	-----	-------------

Parent PAN    Parent MAC    \$FFFF    Child MAC

Alt. PAN Coord	Device Type	Power Source	RX ON When Idle	N/A	N/A	Security	Allocate Addr
----------------	-------------	--------------	-----------------	-----	-----	----------	---------------

Typical values of FCF: \$23, \$CC/\$C8

# Associate Response



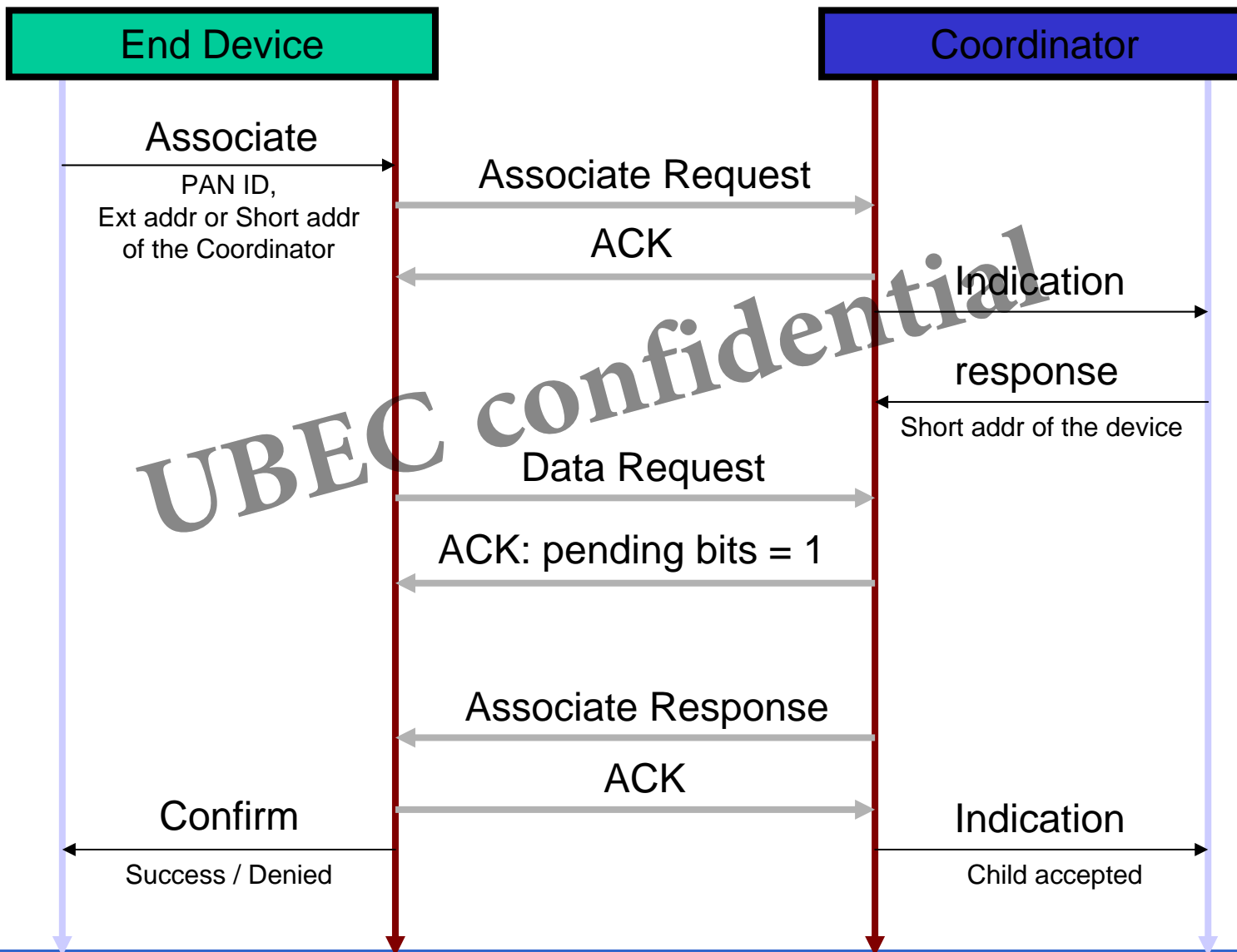
Target PAN    Child MAC    Target PAN    Parent MAC    \$02

- 00: successful
- 01: full
- 02: denial

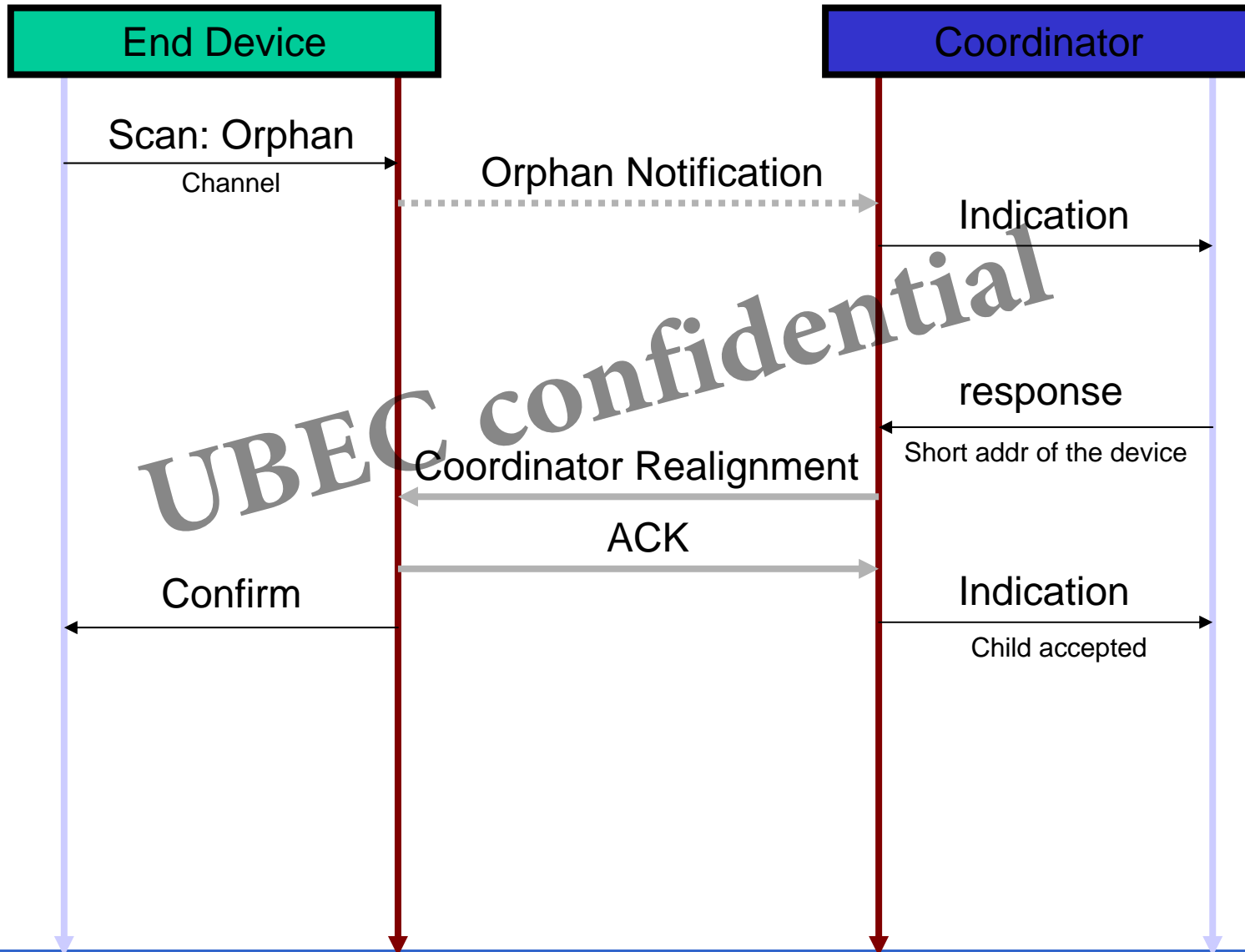
Typical values of FCF: \$23, \$CC



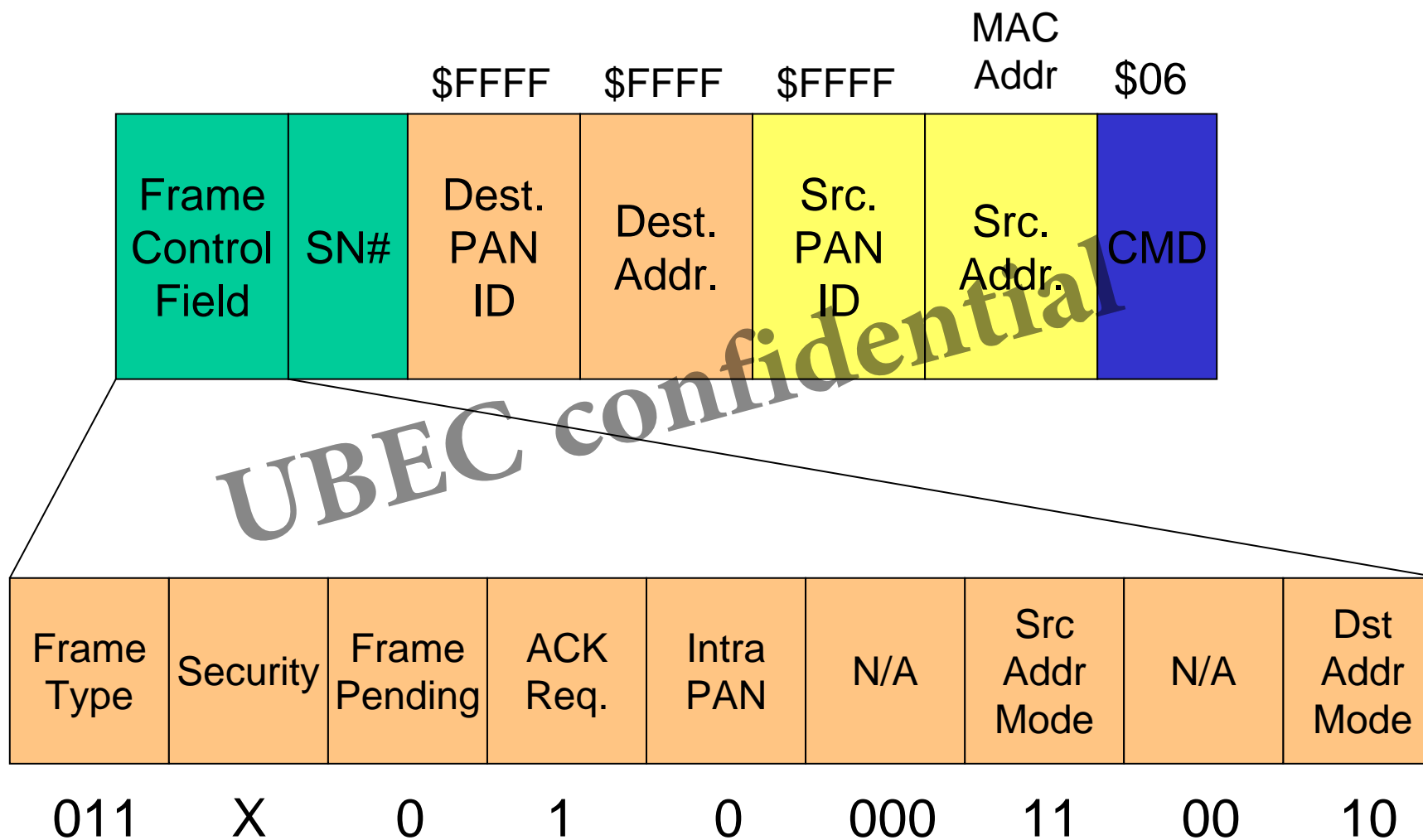
# Review: Association



# Another way to Associate

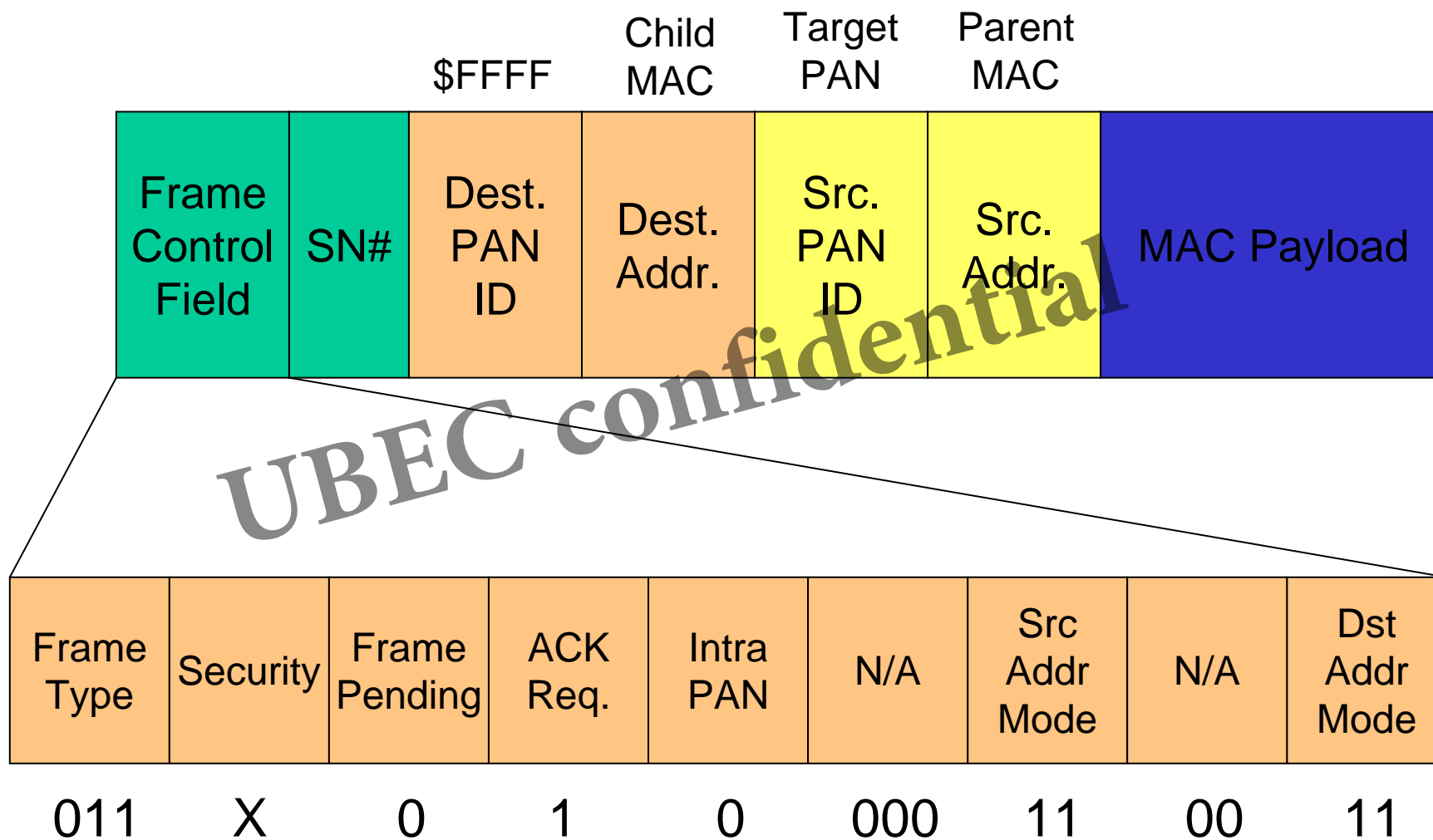


# Orphan Notification



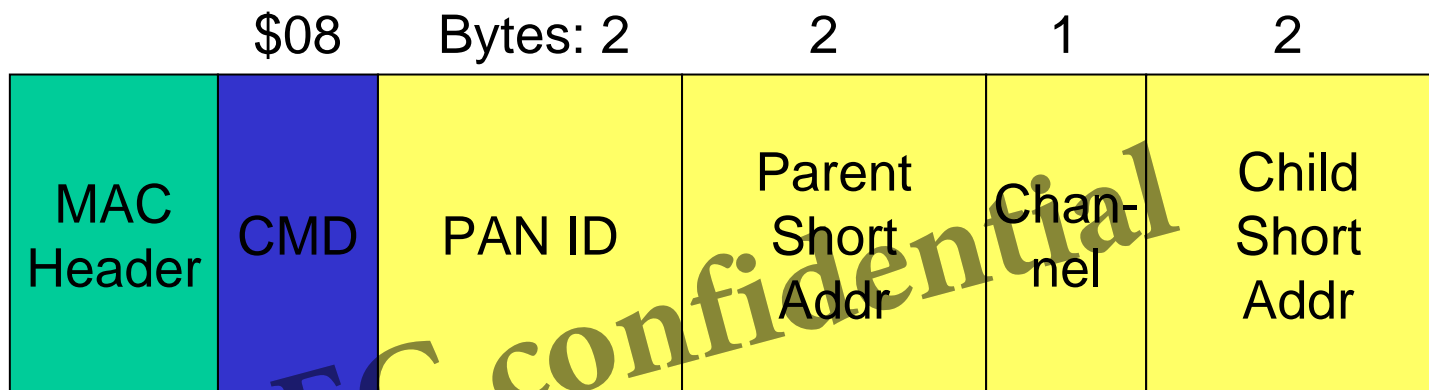
Typical values of FCF: \$23, \$8C

# Coordinator Realignment: Header



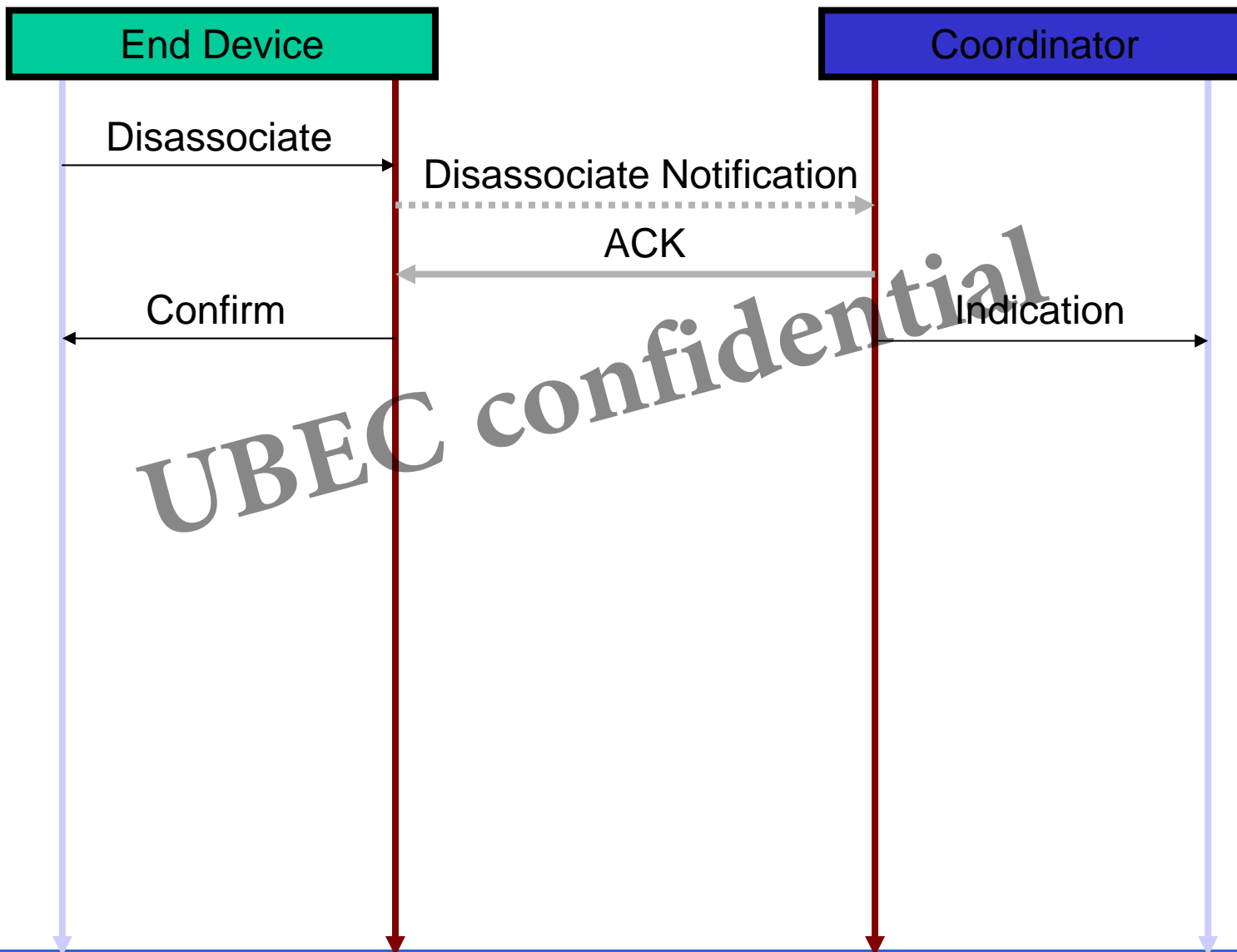
Typical values of FCF: \$23, \$CC

# Coordinator Realignment: Payload

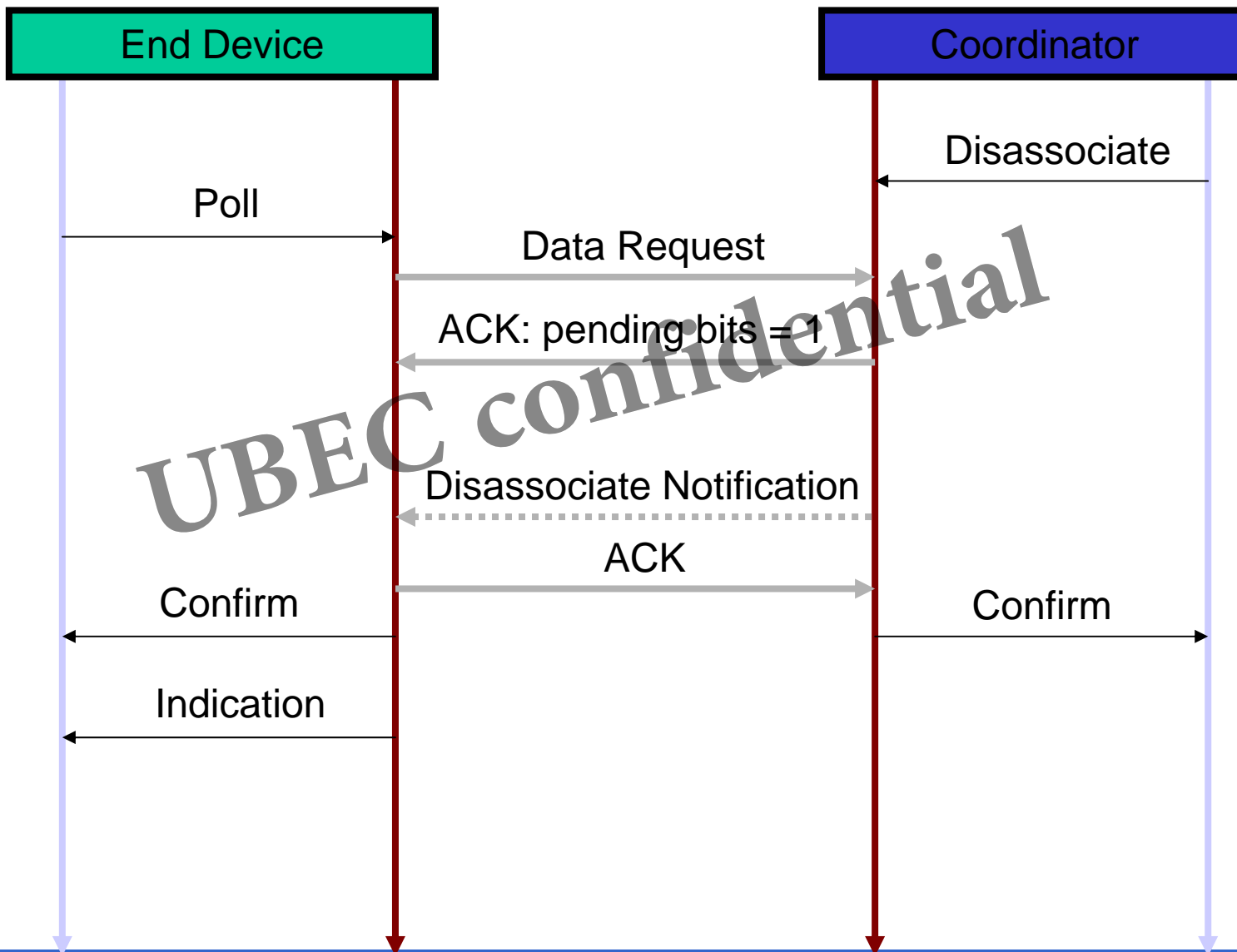


UBEC confidential

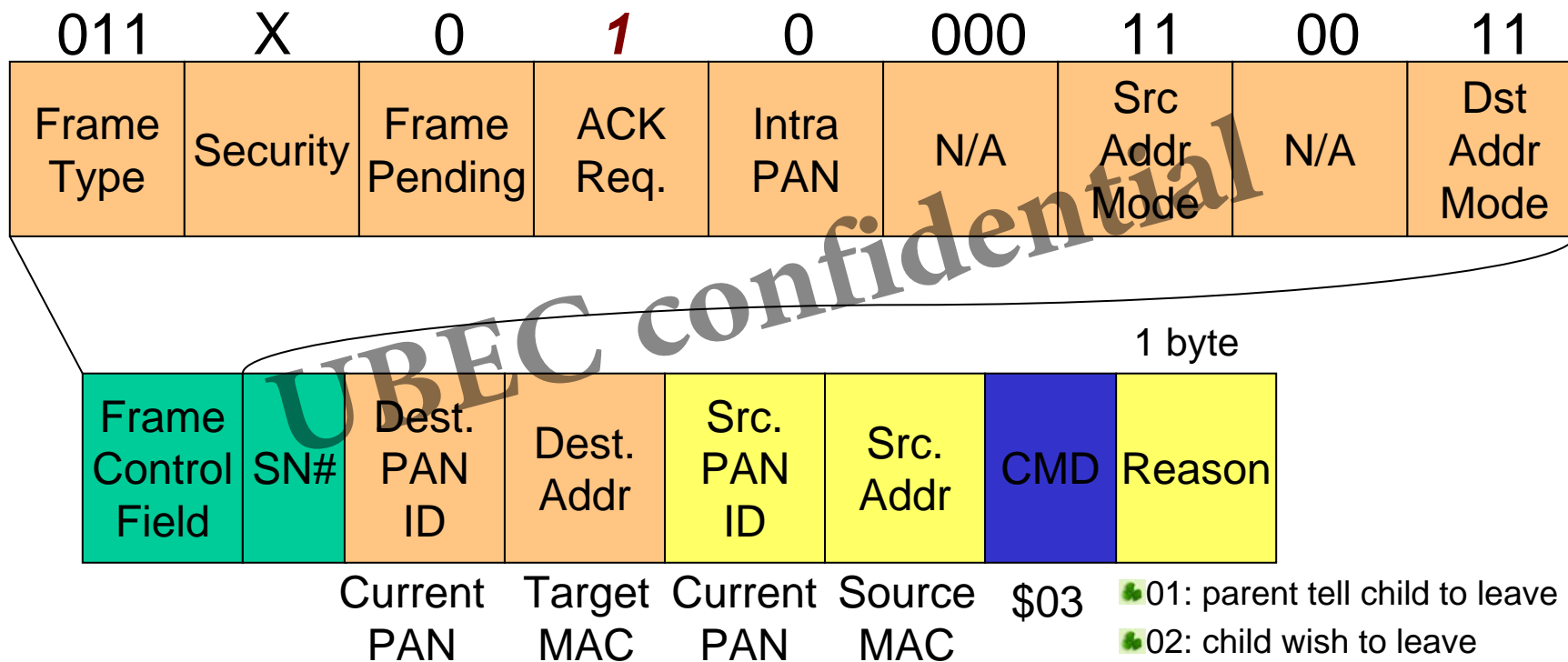
# Disconnect: from Child



# Disconnect: from Coordinator



# Disassociate Notification



Typical values of FCF: \$23, \$CC



- ✚ You've learned:
  - CSMA/CA
  - MAC Beacon Format
  - MAC Command Format
  - More detailed MAC behavior
    - Scan
    - Polling
    - Associate
    - Orphan
    - Disassociate

UBEC confidential

IEEE 802.15.4

**UBEC confidential**

Session 3:

API

# API naming rule



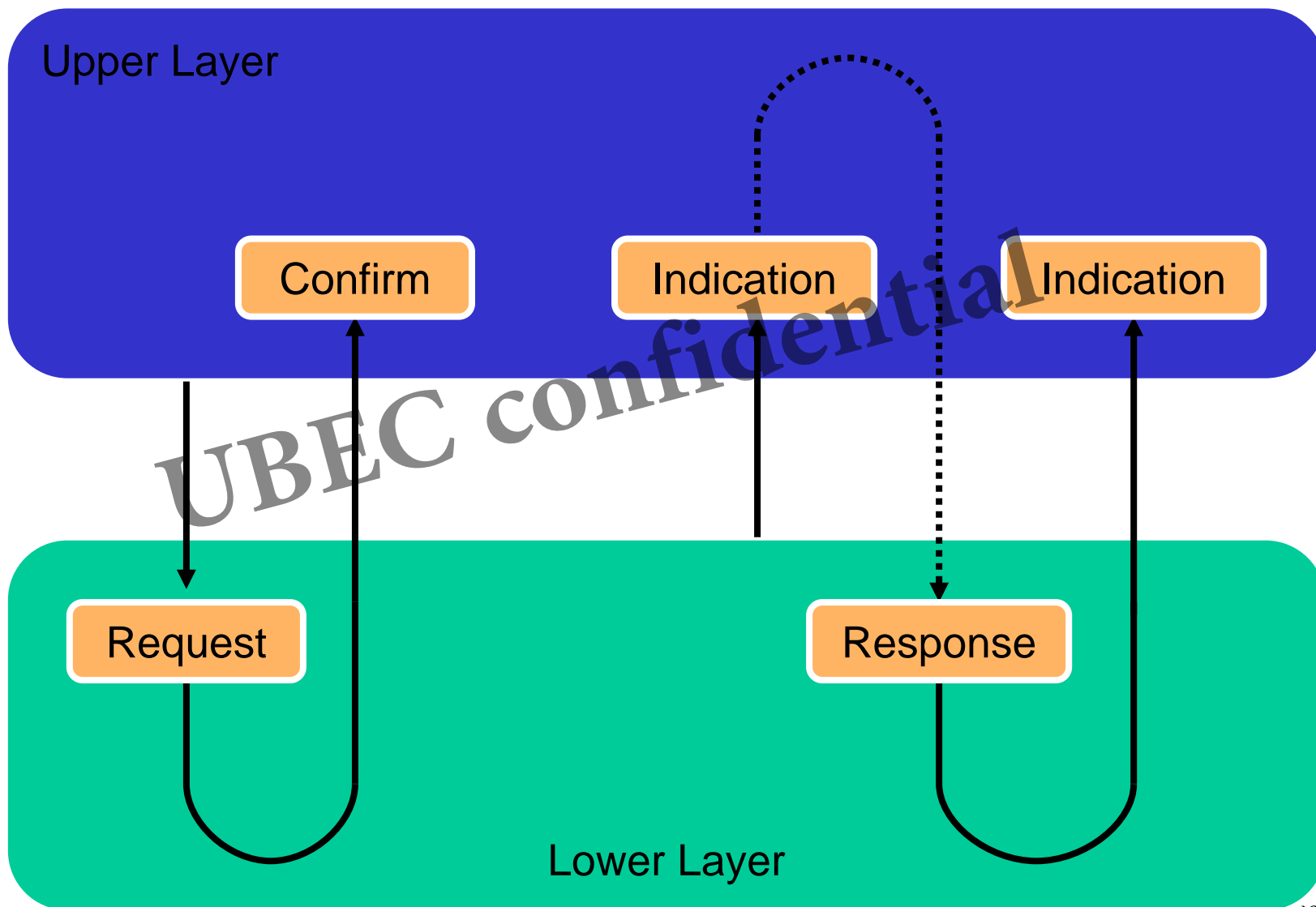
Command set.  
MCPS = data  
MLME = control

What this event does

Which layer generate this event

UBEC confidential

# API naming rule



## API for Data Transmission

**UBEC confidential**

MCPS-DATA

MLME-POLL

# MCPS-DATA.request

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

\$00 = no address  
\$02 = short address  
\$03 = extended address

Identifier, for status report.  
MCPS-DATA.request will  
not be executed in sequence.  
Thus an ID is required.

Bitmask:  
\$01 = ACK required  
\$02 = GTS (session 5)  
\$04 = Indirect transmission  
\$08 = Security

# MCPS-DATA.confirm

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

Identifier for status report

\$00 = success  
\$E8 = invalid parameter  
\$E9 = no ack  
\$E1 = channel access fail  
\$F0 = transaction overflow  
\$F1 = transaction expired

UBEC confidential

# MCPS-DATA.indication

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

Link quality, not signal strength. Low strength in less interference environment still gets high quality.

MAC security: access control list



## MLME-POLL.request

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

**UBEC confidential**

Notice: there is no source address field.

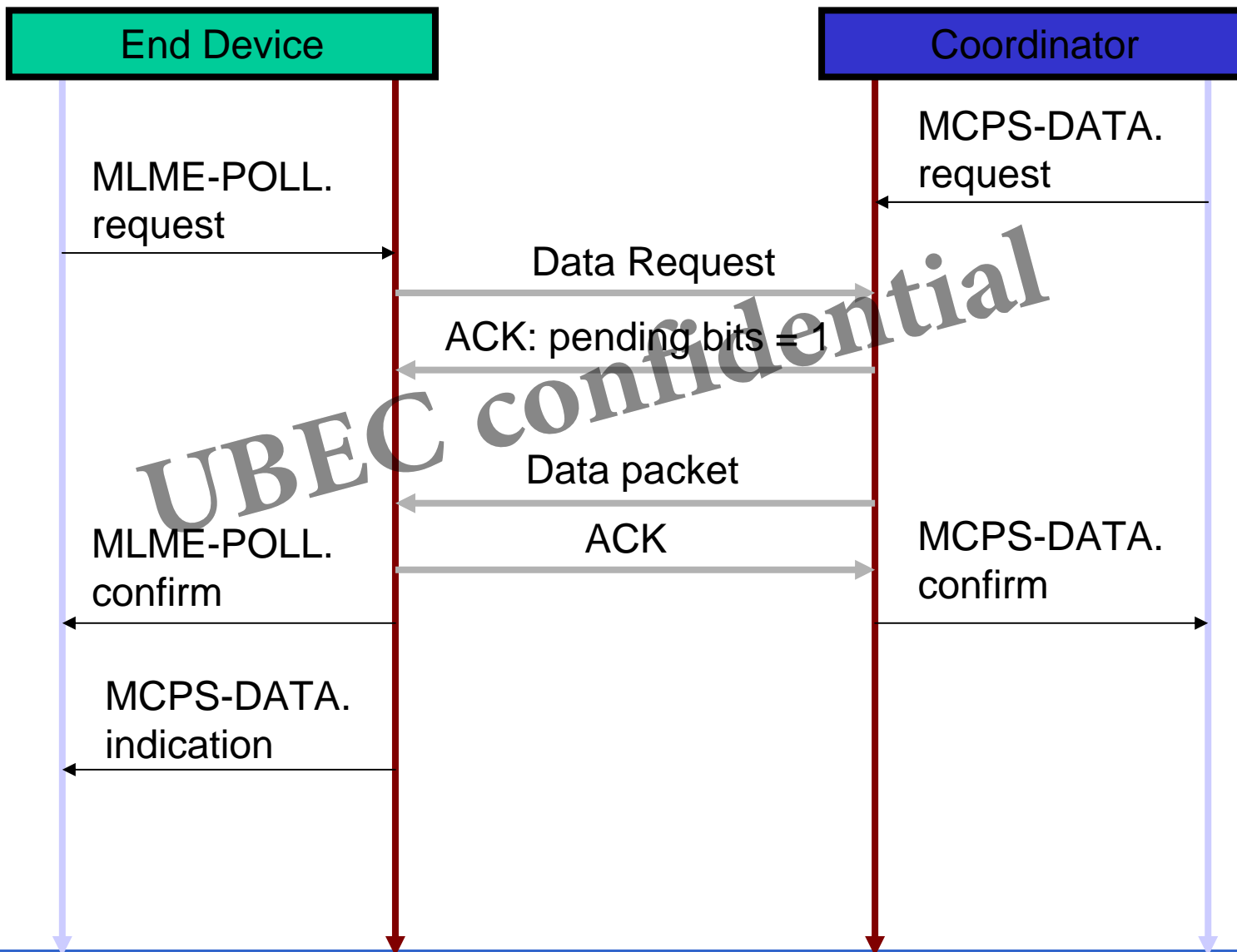
```
MLME-POLL.confirm (  
    status  
)
```

A yellow rectangular callout box with a black border and a pointer pointing to the 'status' parameter in the code above. It contains four lines of text defining status codes.

\$00 = success  
\$E8 = invalid parameter  
\$E9 = no ack  
\$EB = no data

UBEC confidential

# Review: indirect transmission



APIs for scan

**UBEC confidential**

MLME-SCAN

MLME-BEACON-NOTIFICATION

# MLME-SCAN.request

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

\$00 = Energy scan  
\$01 = Active Scan  
\$02 = Passive Scan  
\$03 = Orphan Scan

32 bits bitmask,  
Bit N = Channel N,  
1 = Scan

Duration =  $15.36 \text{ ms} * (1 + 2^{\text{ScanDuration}})$

UBEC confidential

# MLME-SCAN.confirm

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

\$00 = no address  
\$E8 = invalid parameter  
\$EA = no beacon

Array of RSSI values,  
one for each channel

Array of beacon information

UBEC confidential

```
PANDescriptor {  
    CoordAddrMode  
    CoordPANId  
    CoordAddress  
    LogicalChannel  
    SuperframeSpec  
    GTSPermit  
    LinkQuality  
    TimeStamp  
    SecurityUse  
    ACLEntry  
    SecurityFailure  
}
```

**UBEC confidential**

# MLME-BEACON-NOTIFY.indication

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

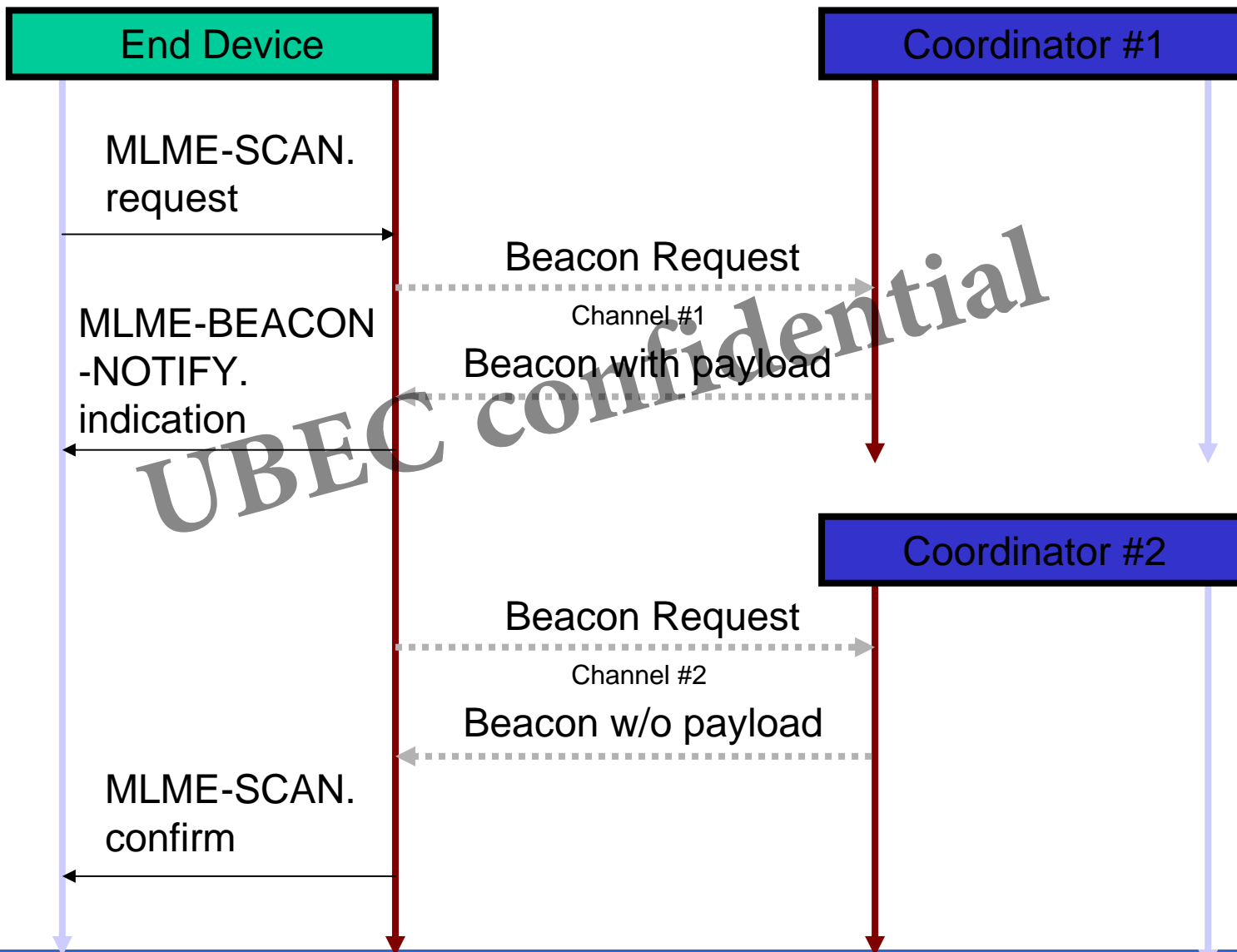
Bit 0-2 = number of short  
Bit 4-6 = number of extended

String of address,  
short address first.

Beacon payload raw data

UBEC confidential





APIs for starting network

**UBEC confidential**

MLME-RESET

MLME-START

MLME-SET

# MLME-RESET.request


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

A yellow rectangular callout box with a black border and a pointer pointing to the 'SetDefaultPIB' parameter in the code above. It contains the text 'TRUE = Reset all settings'.

TRUE = Reset all settings

UBEC confidential

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

A yellow rectangular callout box with a black border and a pointer pointing to the 'status' parameter in the code above. It contains two lines of text: '\$00 = success' and '\$E3 = TRX busy'.

\$00 = success  
\$E3 = TRX busy

UBEC confidential

## MLME-SET.request

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

PIB: PAN Information Base, including

- macAssociatePermit (\$41)
- macBeaconPayload (\$45)
- macBeaconPayloadLength (\$46)
- macCoordExtendedAddress (\$4A)
- macCoordShortAddress (\$4B)
- macPANId (\$50)
- macShortAddress (\$53)
- macRxOnWhenIdle (\$52)
- macTransactionPresistaceTime (\$55)

In real world implementation, we'll use a void pointer (void \*) to pass values.

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

\$00 = success  
\$E8 = invalid parameter  
\$F4 = unsupported

**UBEC confidential**

# MLME-START.request

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

See section 5

Whether the device  
is a PAN coordinator

Tell all child device  
to rejoin network

UBEC confidential

# MLME-START.confirm

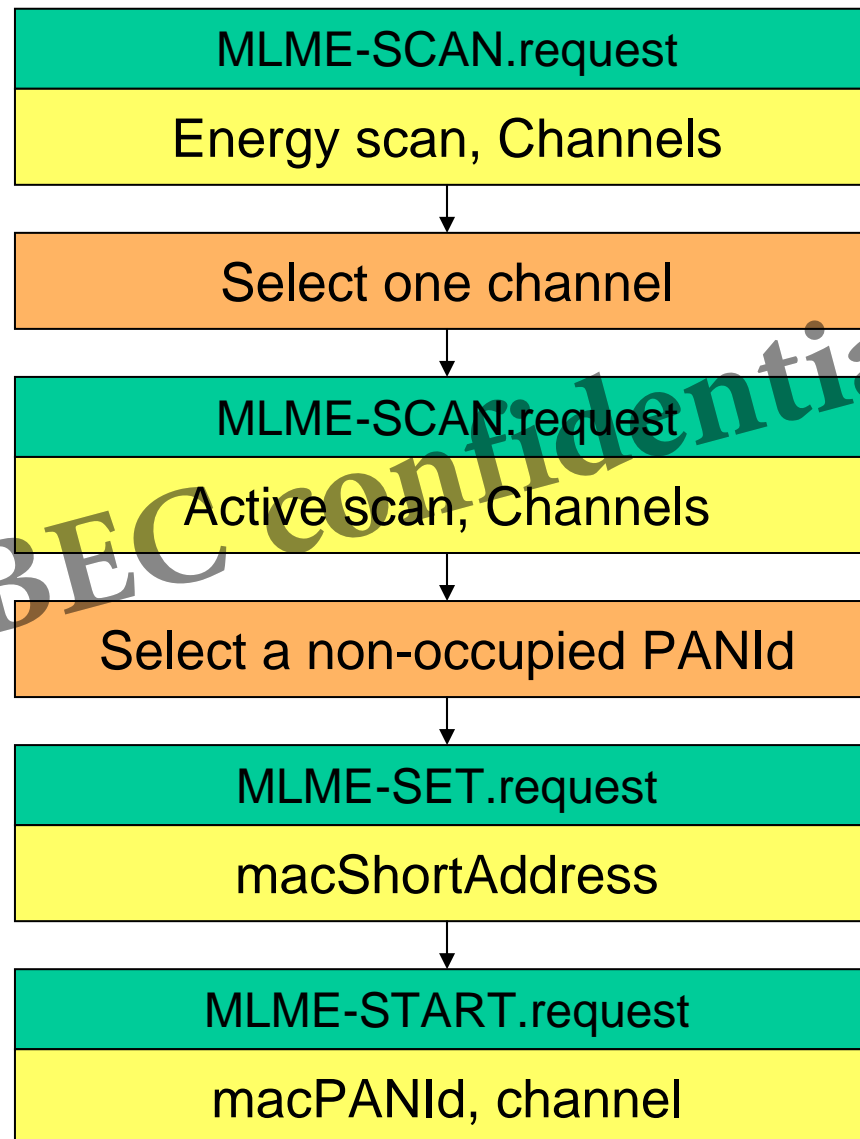
```
MLME-START.confirm (  
    status  
)
```

A yellow rectangular callout box with a black border, containing three lines of text. A yellow arrow points from the 'status' parameter in the code above to the first line of the callout.

\$00 = success  
\$E8 = invalid parameter  
\$EC = **no short address**

UBEC confidential





## APIs for Association

**UBEC confidential**

MLME-ASSOCIATE


MLME-DISASSOCIATE

MLME-ORPHAN

MLME-COMM-STATUS

# MLME-ASSOCIATE.request

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

A yellow rectangular callout box with a black border and a pointer pointing to the 'CoordAddrMode' parameter in the code above.

Value, from \$0B to \$1A

UBEC confidential

# MLME-ASSOCIATE.indication

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

A yellow rectangular callout box with a black border and a pointer pointing to the 'DeviceAddress' parameter in the code above.

Extended Address

**UBEC confidential**

# MLME-ASSOCIATE.response

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

Extended Address

Short Address

\$00 = Success  
\$01 = Overflow  
\$02 = Denial

UBEC confidential

# MLME-ASSOCIATE.confirm

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

Acquired short address \$FFFE means no short address; extended-address will be used for all packet.

**UBEC confidential**

\$00 = success  
\$E1 = channel access fail  
\$E8 = invalid parameter  
\$E9 = no ack  
\$EB = no data

# MLME-DISASSOCIATE.request

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

Extended Address

\$01: Coordinator wishes device to leave  
\$02: Devices wishes to leave

UBEC confidential

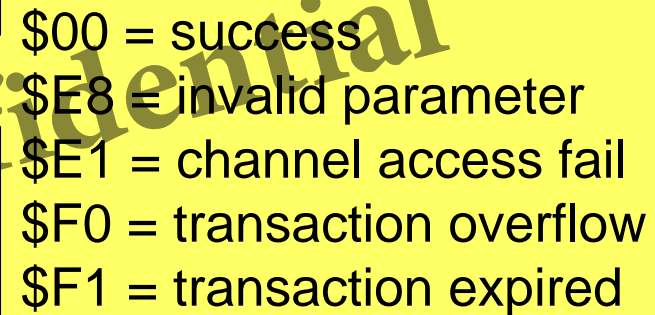
```
MLME-DISASSOCIATE.indication (  
    DeviceAddress,  
    DisassociateReason,  
    SecurityUse,  
    ACLEntry  
)
```

**UBEC confidential**



# MLME-DISASSOCIATE.confirm

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

A yellow rectangular callout box with a black border, containing a list of status codes and their meanings. A yellow arrow points from the word 'status' in the code block above to this box.

- \$00 = success
- \$E8 = invalid parameter
- \$E1 = channel access fail
- \$F0 = transaction overflow
- \$F1 = transaction expired

UBEC confidential

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

A yellow rectangular callout box with a black border and a pointer pointing to the 'OrphanAddress' parameter in the code above.

Extended Address

**UBEC confidential**

# MLME-ORPHAN.response

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

Extended Address

Short Address

If true, this event will be ignored.

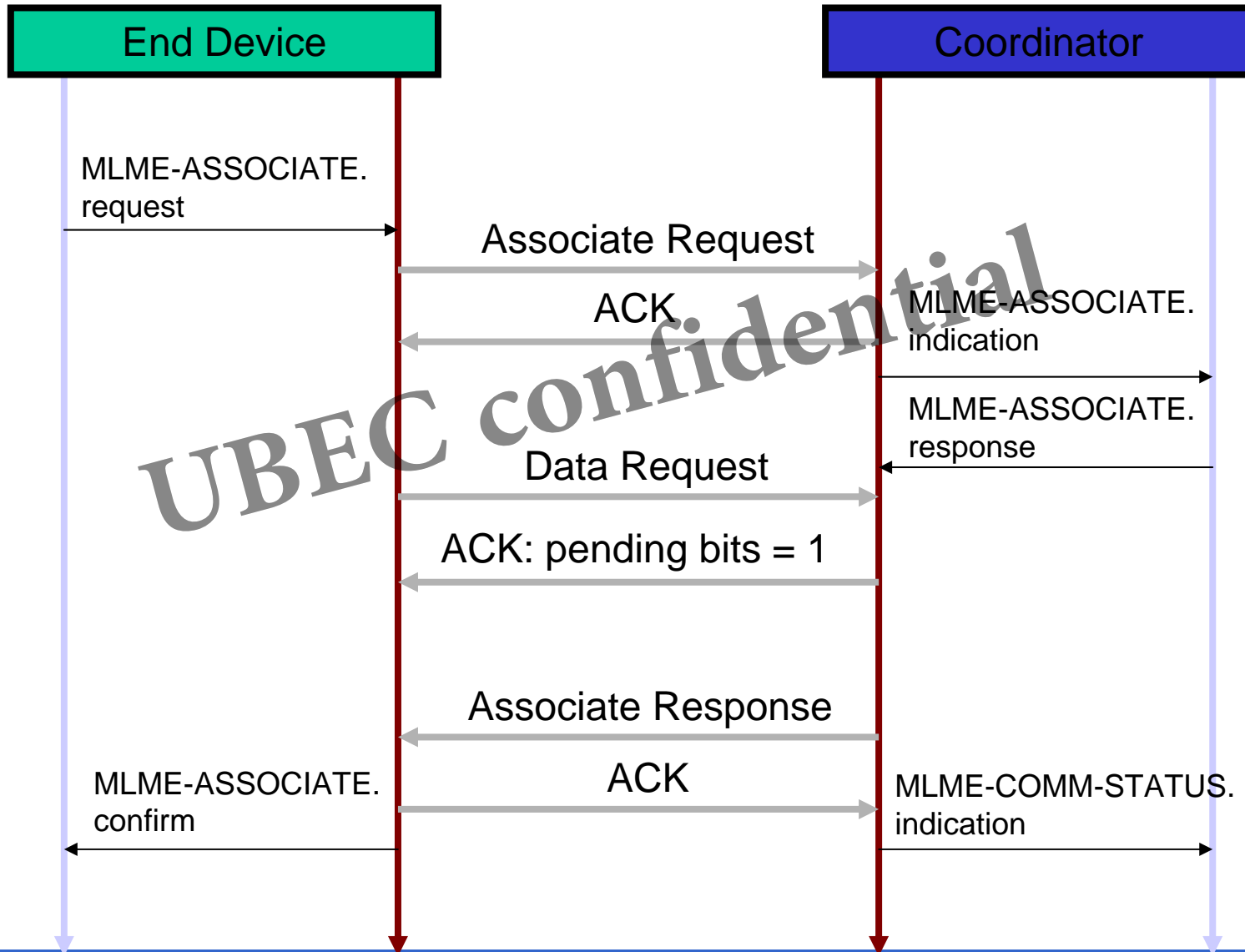
UBEC confidential

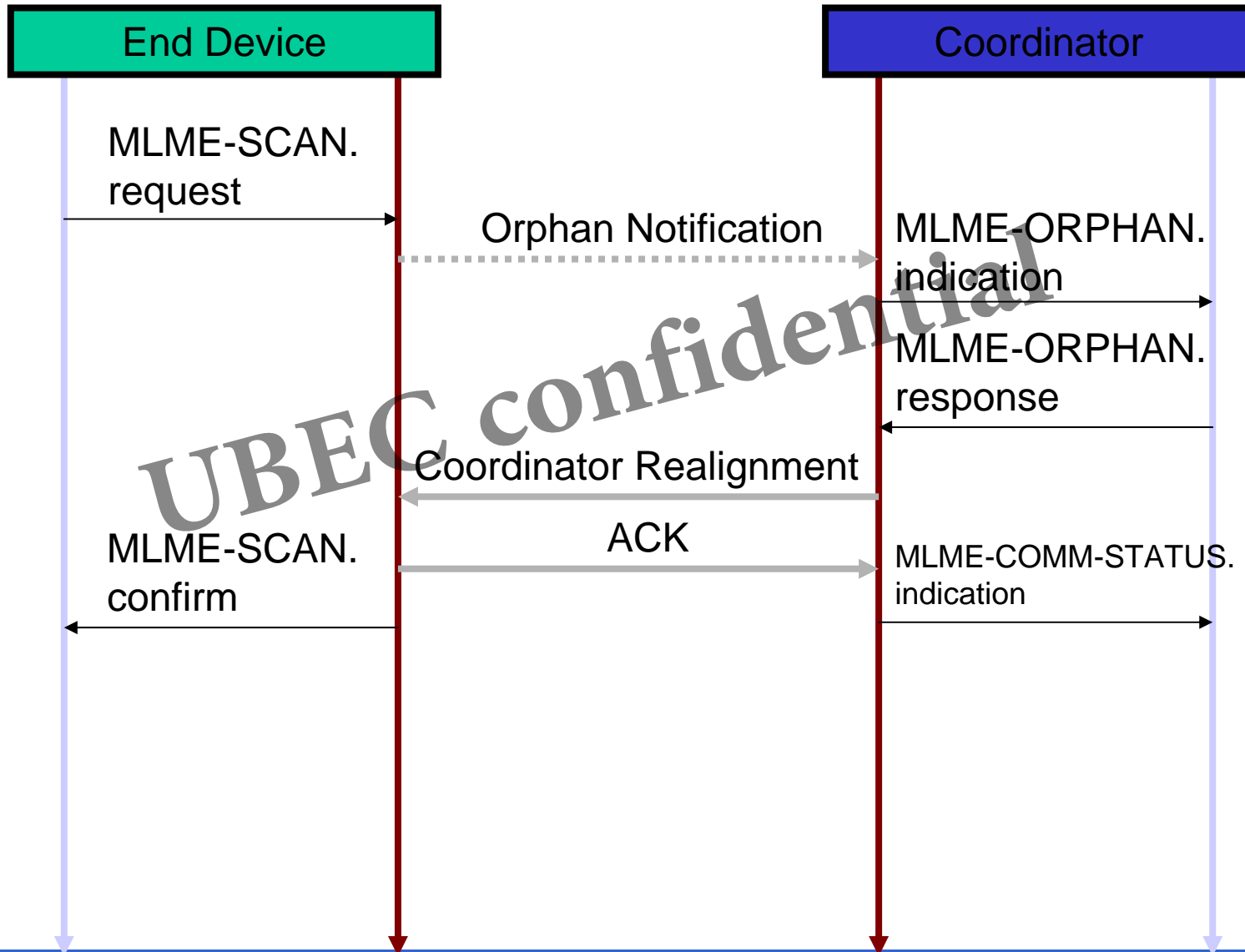
# MLME-COMM-STATUS.indication

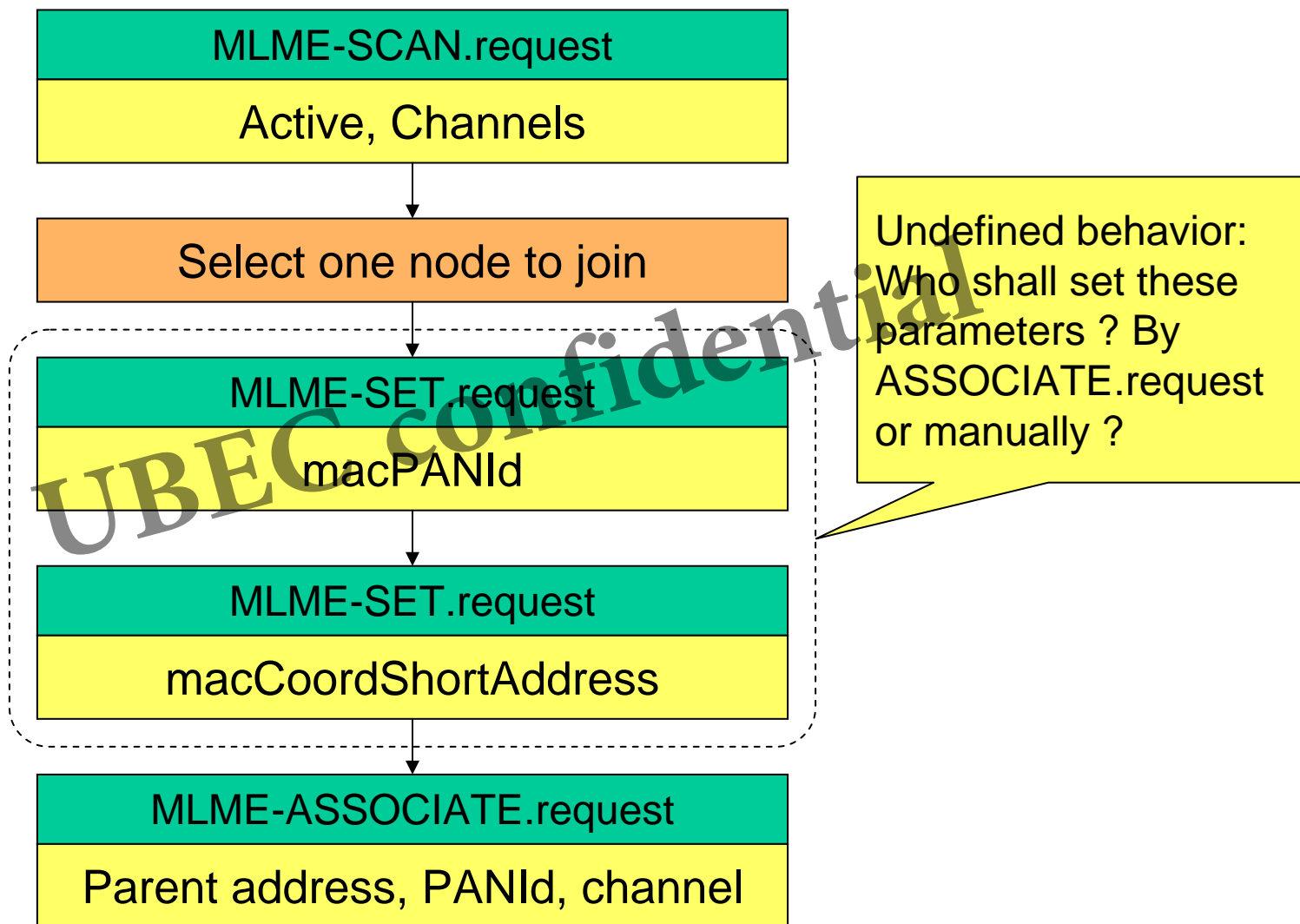
```
MLME-COMM-STATUS.indication (  
    PANid,  
    SrcAddrMode,  
    SrcAddr,  
    DstAddrMode,  
    DstAddr,  
    status  
)
```

Undefined

\$00 = success  
\$E8 = invalid parameter  
\$E9 = no ack  
\$E1 = channel access fail  
\$F0 = transaction overflow  
\$F1 = transaction expired







Other APIs

**UBEC confidential**

MLME-GET

MLME-RX-ENABLE



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

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

In real world implementation, we'll use a void pointer (void \*) to pass values.

# MLME-RX-ENABLE

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

Only works in beacon network.

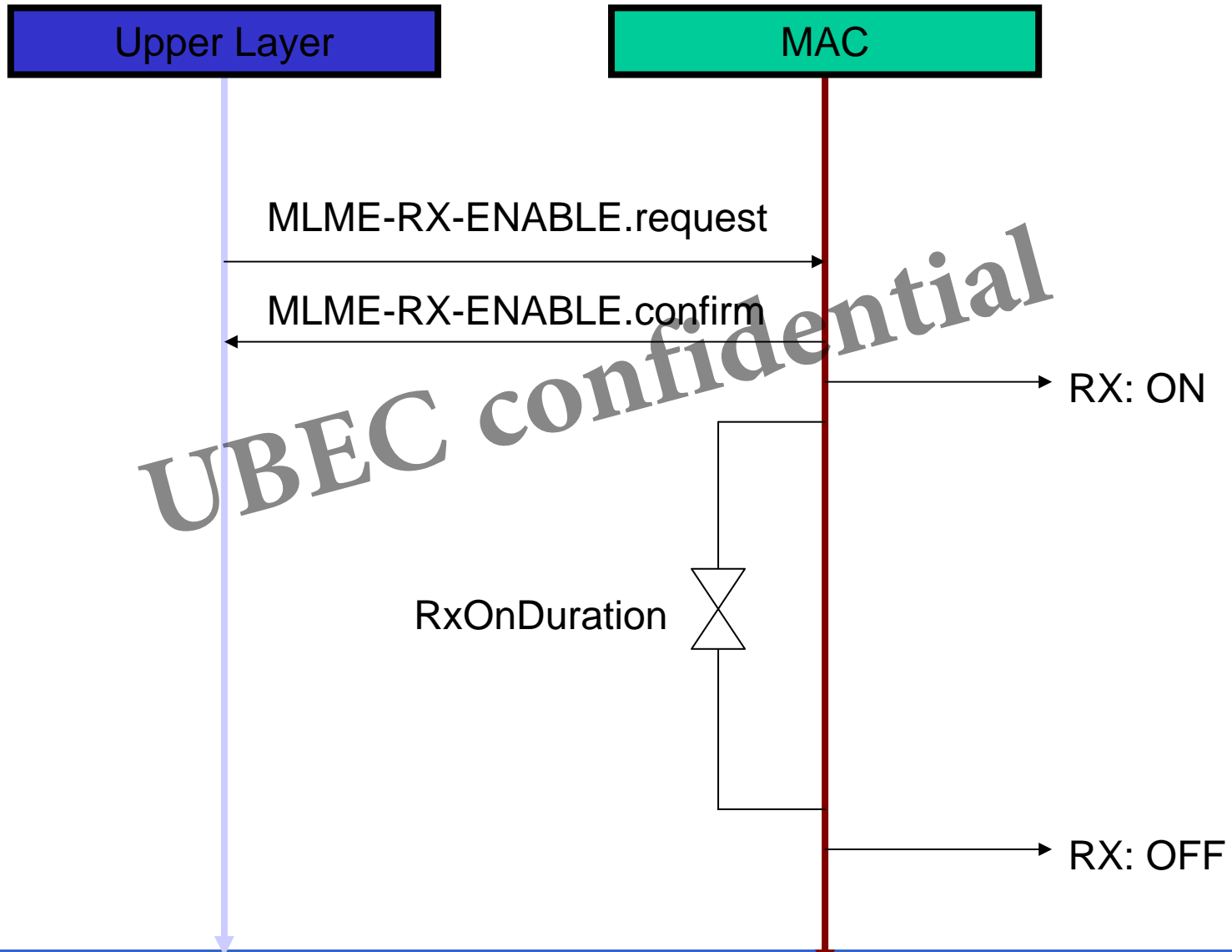
Only works in beacon network.

\$000000~\$FFFFFF  
Unit = 16us  
Max = 268.435 sec

\$00 = success  
\$F2 = TX active

UBEC confidential

# MLME-RX-ENABLE



UBEC confidential

- ✚ In this session, you've learned
  - MAC API call convention.
  - MAP APIs for scan, data transmission, association, and other functions.
  - More detailed MAC behaviors.
  - Standard startup procedure for coordinator and end device.

UBEC confidential