Standard ECMA-398

1st Edition / June 2011

Close Proximity Electric Induction Wireless Communications

Contents

Page

Introduction

Today’s typical consumer uses digital files to store multimedia content such as music, photos, and videos. But these files are quickly becoming larger in number and size. A continual demand for higher quality results in larger file sizes. And proliferation of smaller, portable devices makes it easier to generate more content in less time. But the desire to store, share, and enjoy that content remains strong. And this usually requires transferring the content from one device to another. For example, storing might involve transferring the content from a video camera to an external disk drive. Sharing photos might involve transferring the contents from one mobile phone to another mobile phone. And enjoying content might involve streaming content from a video camera to a TV using a special video cable.

But with today’s available technology, these activities present difficulties to the average consumer. The transfer process may take a long time due to the large file sizes. Or it may involve special cables or complex setup. Therefore, a need exists to make it faster and simpler to transfer large multimedia files. This Standard specifies a technology that addresses this need by using close proximity electric induction to transfer large files quickly and easily.

This Ecma Standard has been adopted by the General Assembly of June 2011.

Close Proximity Electric Induction Wireless Communications

1 Scope

This Standard specifies a connection layer (CNL) and a physical layer (PHY) for transferring data between two close proximity entities using electric induction coupling.

2 Conformance

Implementations conforming to this Standard implement both the CNL and the PHY. All Conforming implementations support a centre frequency of 4,48 GHz and all rate settings specified in Table 2.

3 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 7498-1:1994, *Information technology – Open System Interconnection – Basic Reference Model: The Basic Model*

ITU-T Z.120, *Series Z: Languages and General Software Aspects for Telecommunication Systems, Formal description techniques (FDT) – Message Sequence Chart (MSC)*

4 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

4.1
chip
shortest duration digital unit that is transmitted and used to spread the spectrum

4.2
chip rate
rate at which chips are transmitted

4.3
coupler
antenna used to transmit and receive an electric induction field

4.4
electric induction field
electric field with strength inversely proportional to the distance squared

4.5
initiator
sender of a connection request

4.6
PHY rate
chip rate / spreading factor

4.7
responder
receiver of a connection request

4.8
spreading factor
number of duplications

4.9
symbol
modulation pulse in a single I or Q channel as expressed as a baseband waveform

4.10
symbol rate
rate at which symbols are transmitted in each I or Q channel

4.11
target
peer entity

4.12
unique ID
code uniquely identifying each implemented unit

5 Abbreviations and acronyms

ACK	Acknowledgement
BPSK	Binary Phase Shift Keying
CCF	Convolutional Coding Factor
CNL	CoNnection Layer
CPCI	CNL Protocol Control Information
CPDU	Connection layer Protocol Data Unit
CSDU	Connection layer Service Data Unit
C-Acc	“Connection Accept” message management frame
C-Probe	“Connection Probe” message management frame
C-Req	“Connection Request” message management frame
C-Rls	“Connection Release” message management frame
C-Sleep	“Connection Sleep” message management frame
C-Wake	“Connection Wakeup” message management frame
ECS	Error Check Sequence
EVM	Error Vector Magnitude
FCS	Frame Check Sequence
FEC	Forward Error Collection
HCS	Header Check Sequence
ImACK	Immediate Acknowledgement required
LFSR	Linear Feed-back Shift Register
LiCC	Link Control Command
MSC	Message Sequence Chart
MUX	Multiplexer
NoACK	No Acknowledgement required
PDU	Protocol Data Unit
PHY	Physical Layer
PPCI	PHY Protocol Control Information
PPDU	Phy layer Protocol Data Unit

PSD	Power Spectral Density
PSDU	PHY SDU
SAP	Service Access Point
SDU	Service Data Unit
UID	Unique ID

6 Overview

6.1 Introduction

This Standard specifies the bottom 2 layers of a close proximity wireless transfer technology. By touching (or bringing very close together) two electronic entities, this technology allows high speed exchange of data. The basic concept consists of a touch-activated multi-purpose interface designed for applications requiring high-speed data transfer between two entities in a point-to-point (1:1) mode, without the need for external physical connectors.

The physical layer has a maximum transmission rate of 560 Mbps, adjusting the data rate downward according to the wireless environment to maintain a robust link even when the surrounding wireless condition fluctuates.

The RF transmit power is kept at a very low level to cause negligible interference with other nearby wireless systems, including other close proximity electric induction systems.

Implementations transmit and receive by means of an electric induction field suitable for near field data exchange. This approach is fundamentally different from traditional wireless systems using microwave radiation.

Entities establish a link to enable data transfer and serve as initiator and responder respectively. These two roles have no relation to the actual direction of data transfer as illustrated in Figure 1.

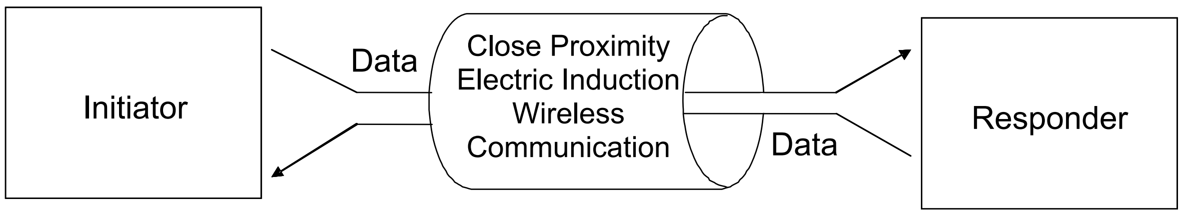


Figure 1 — Connection between Initiator and Responder

The initiator sends a “connection request", and the responder is its peer that receives a “connection request". Entities can assume either of these roles.

As specified in Figure 2, this Standard uses the OSI Basic Reference Model specified in ISO/IEC 7498-1.

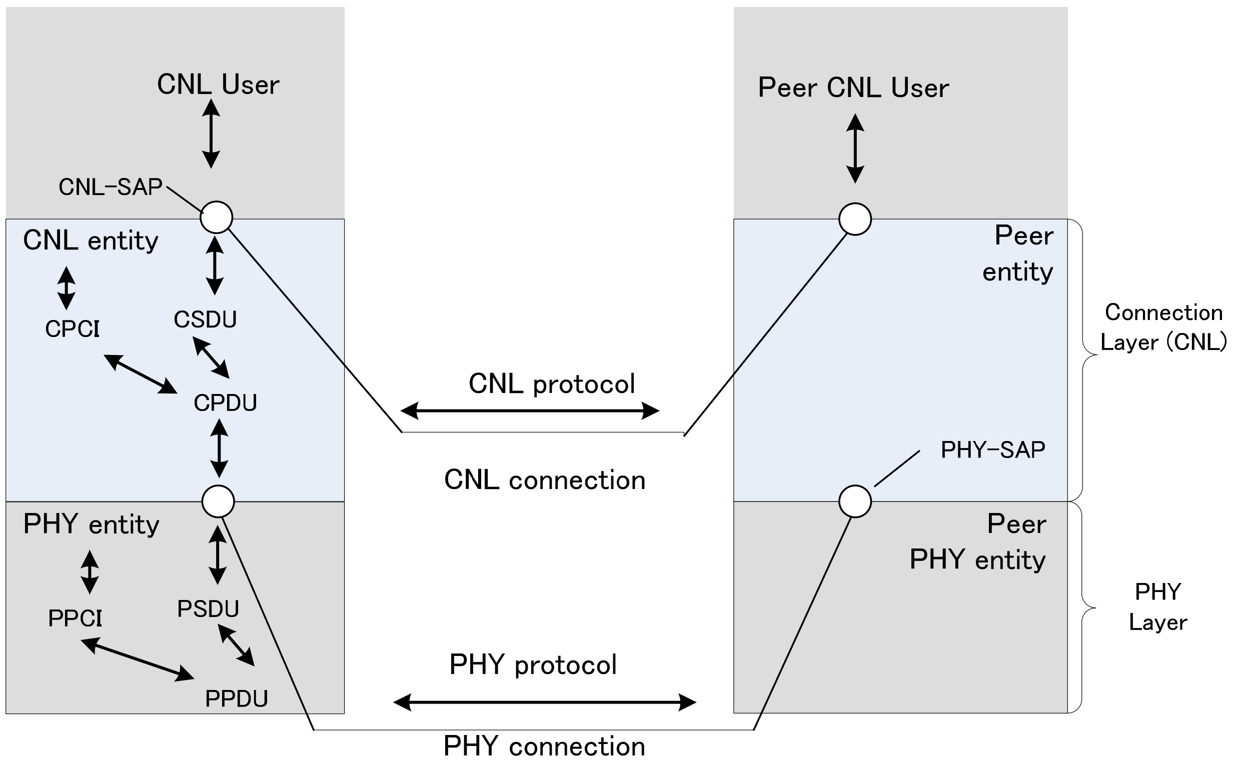


Figure 2 — OSI Basic Reference Model used in this Standard

7 Transmit signal

7.1 Modulation scheme parameters

The modulation scheme uses Pi/2 shift BPSK and a chip rate (R_c) of 560 Mcps, as illustrated in Table 1. Here, the chip rate refers to the shortest duration digital units that are transferred over the air as well as the digital bits that are used to spread the transmitted bandwidth. Since the modulation scheme uses Pi/2 shift BPSK, the reciprocal number of the chip rate ($1/R_c$) represents the interval between samples of an envelope concatenated along the time axis and the symbol rate (R_s) on one channel (Ich or Qch) is half the occupied bandwidth (R_c) of the envelope. Hence, the relationship of $R_s = R_c/2$ is established.

Table 1 — Tx signal parameters

Chip Rate: R_c	560 Mcps
Chip duration: Tc = $1/R_c$	1,786 nsec
Symbol Rate: R_s	280 Msps
Carrier Center Frequency: F_c	4,48 GHz
Modulation	Pi/2 shift BPSK + DSSS
FEC	1/2 Convolutional code + Reed Solomon code

7.2 Transmitter functional block diagram

The transmitter functional block diagram is illustrated in Figure 3. Data from the CNL is first encoded by the Reed-Solomon encoder and the Convolutional encoder. Whether the Convolutional encoder is on or off is determined by the Rate Setting in use, as defined in Table 2.

The spreader spreads the encoded data by duplicating symbols by the spreading factor or process gain G_{SF}.

The spread data is then scrambled by the scrambler. Scrambling is accomplished using a pseudo random sequence generated by the Linear Feedback Shift Register (LFSR) in the Scrambler Sequence Generator. Of the fields of the frame format specified in Figure 14, the Preamble, PHY Header and Payload are scrambled using different random seeds.

The Pi/2 shift BPSK mapper spreads a binary sequence into complex number signals by multiplying the input signal by a rotator whose rotation angle differs by 90 degrees for each sample.

The baseband waveform generator illustrated in Figure 3 is a filter that uses the baseband waveform specified in Figure 12 as the impulse response. The generated baseband signal $S_{BB}(t)$ is then up-converted to centre frequency F_c by the RF module.

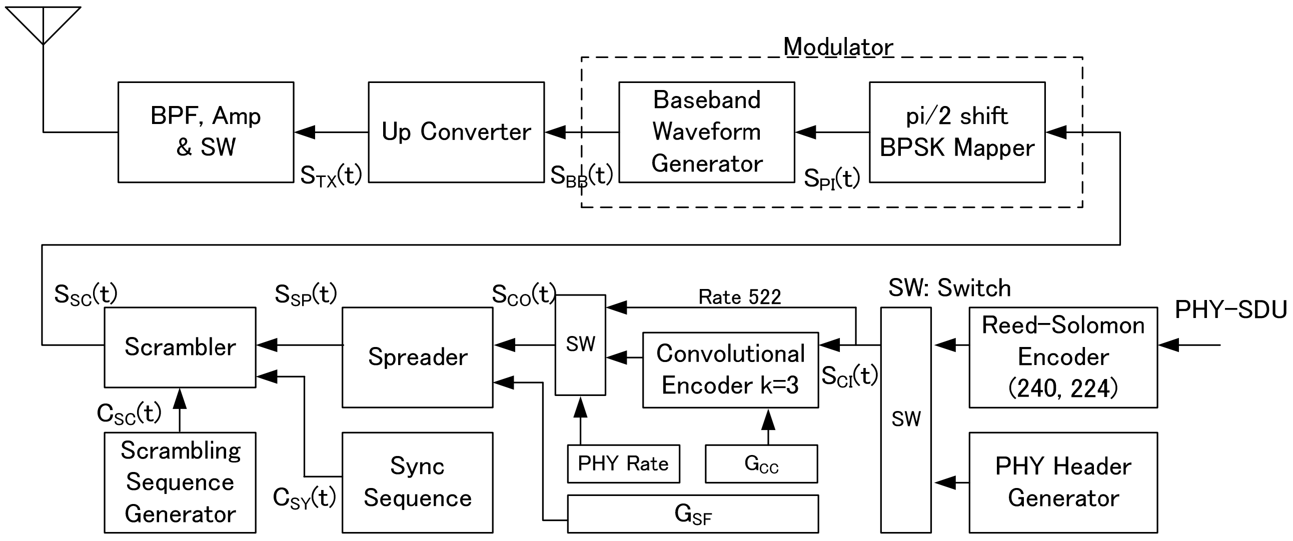


Figure 3 — Transmitter functional block diagram

7.2.1 Supported Rate Settings and rate dependent parameters

Table 2 specifies the rates used by the PHY. For rate control, the PHY manipulates the following parameters:

- Spreading factor (G_{SF}) = 1, 2, 4, 8, or 16
- Convolutional code factor (G_{CC}):
  - G_{CC} = ½ if Convolutional code is used
  - G_{CC} = 1 otherwise
- Reed-Solomon Factor (G_{RS}):
  - G_{RS} = 224/240 if Reed-Solomon coding is used
  - G_{RS} = 1 otherwise

From the above parameters, the rates are calculated as follows.

- Chip Rate = 560 Mcps
- Symbol Rate = 280 Msps

- PHY Rate (Mbps) = Chip Rate (Mcps) / G_{SF}
- Data Rate (Mbps) = PHY Rate (Mbps) x CCF x G_{RS}

Table 2 — Rates (Data Rate is rounded down to the nearest 1 Mbps)

Rate Settings	Chip Rate (Mcps)	Symbol Rate (Msps)	PHY Rate (Mbps)	Data Rate (Mbps)	Spreading Factor: G_{SF}	Convolutional Code Used?	Reed-Solomon Code Used?
Rate 522	560	280	560	522	1	No	Yes
Rate 261	560	280	560	261	1	Yes	Yes
Rate 130	560	280	280	130	2	Yes	Yes
Rate 65	560	280	140	65	4	Yes	Yes
Rate 32	560	280	70	32	8	Yes	Yes
PHY Header	560	280	35	17	16	Yes	No

7.2.2 Reed-Solomon encoder

Table 3 specifies the parameters of the Reed-Solomon encoder. Reed-Solomon code is employed for the CPDU inner coding.

Table 3 — Reed-Solomon encoder parameters

Function	Description
Galois field	GF (2^8)
Primitive polynomial	$p(X) = X^8 + X^4 + X^3 + X^2 + 1$
Primitive element	MSB LSB $\alpha = [00000010]$
Generating polynomial	$g(X) = \prod_{i=0}^{15} (X - \alpha^i)$
Code length	240 Bytes
Data length	224 Bytes

PSDU data shall be Reed-Solomon encoded as follows.

1) Starting with Byte 00 (see Figure 18), each block of 224 bytes shall be transferred to the Reed-Solomon encoder to generate the 16 parity bytes. The bit-ordering of each byte and each Galois field symbol shall be identical. Simultaneously, the same 224 bytes shall be transferred unchanged in the same order to the next stage of processing. These 224 bytes are referred to as message bytes.

2) After each block of message bytes are transferred to the next stage of processing, the 16 bytes of Reed-Solomon parity shall be transferred to the next stage of processing in the order of MSB first of each parity byte. The parity bytes shall be transferred starting from higher order to lower order.

Note that the Reed-Solomon encoder transfers and processes data in byte-by-byte order. See Annex E for examples of Reed-Solomon Encoder data values.

7.2.3 Convolutional encoder

Table 4 specifies the parameters of the Convolutional encoder. Convolutional code is employed for the PHY Header and the CPDU outer coding.

Table 4 — Convolutional encoder parameters

Function	Description
Constraint length	K=3
Polynomial	G0=7oct, G1=5oct

Table 5 specifies the relation between the Rate Setting and the number of convolutional encoders.

Figure 4 and Figure 5 specify the input/output relationship for the convolutional encodings. The input signal in both Figure 4 and Figure 5 is the RS-Encoded data as specified in Figure 14.

Table 5 — Rate Settings and number of convolutional encoders

Rate Setting	Num. of Conv. Enc.	Input/Output bit ordering
Rate 522	No convolutional code	
Rate 261	2	See Figure 4
Rate 130	1	See Figure 5
Rate 65	1	See Figure 5
Rate 32	1	See Figure 5
PHY Header	1	See Figure 5

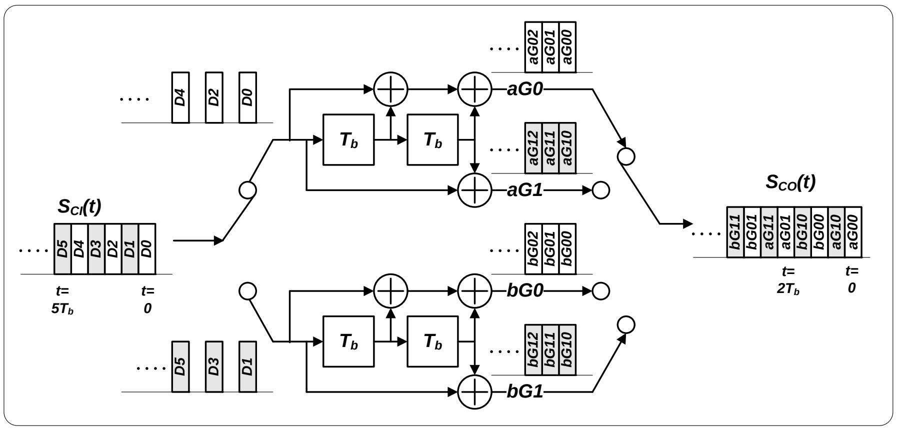


Figure 4 — Convolutional encoder for Rate 261

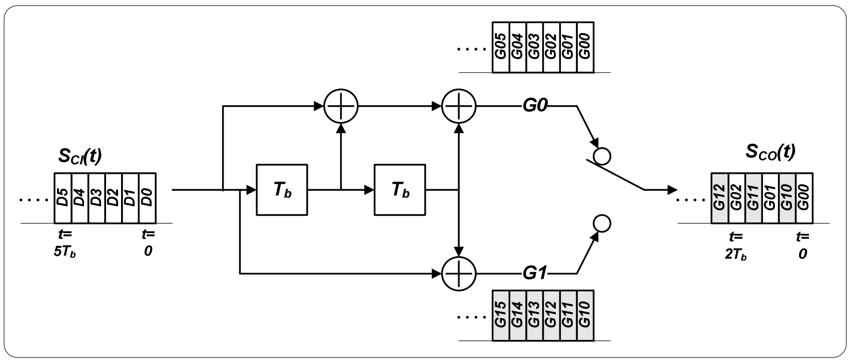


Figure 5 — Convolutional encoder for Rate 130 to Rate 32 and PHY Header

Figure 4 and Figure 5 show each storage element as a box labelled Tb where Tb is the bit duration of the Input data. The storage elements shall have value=0 at t=0. See Annex E for examples of Convolutional Encoder data values.

In Figure 4 and Figure 5, the data is transferred and processed bit by bit starting with the MSB of $S_{CI}(t)$.

7.2.4 ECS

Figure 6 and Equation (1) specify the 16-bit ECS. The notation [n2:n1] means sequence of bits ordered from bit n2 to bit n1 where bit n2 is the most significant bit (MSB) and bit n1 is the least significant bit (LSB).

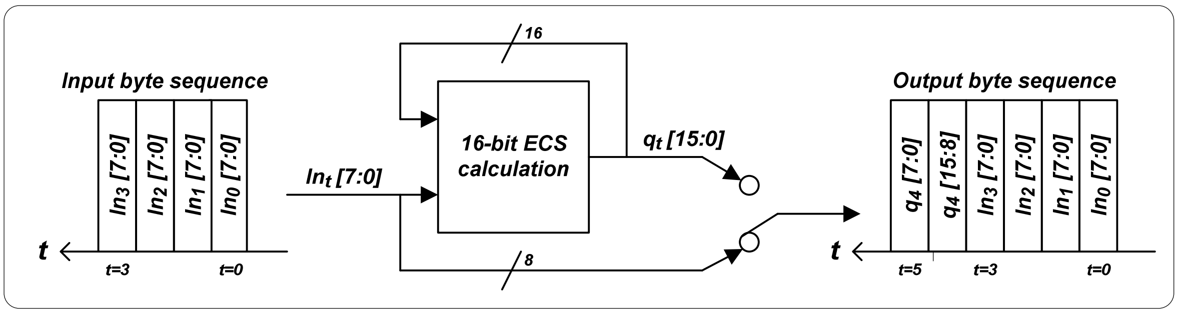


Figure 6 — 16-bit ECS generator

16-bit ECS calculation:

$q_0[15:0]$=0x FFFF

$In_0[7:0]$ is the 1st byte in the ECS target field to be sent.

“^” denotes the XOR (exclusive OR) operator in Equation (1).

$q_{t+1}[15] = q_t [8] \wedge q_t [0] \wedge In_t[0] \wedge q_t [4] \wedge In_t [4]$

$q_{t+1} [14] = q_t [9] \wedge q_t [1] \wedge In_t [1] \wedge q_t [5] \wedge In_t [5]$

$q_{t+1} [13] = q_t [10] \wedge q_t [2] \wedge In_t [2] \wedge q_t [6] \wedge In_t [6]$

$q_{t+1} [12] = q_t [11] \wedge q_t [0] \wedge In_t [0] \wedge q_t [3] \wedge In_t [3] \wedge q_t [7] \wedge In_t [7]$

q_{t+1} [11] = q_t [12] ^ q_t [1] ^ In_t [1]

q_{t+1} [10] = q_t [13] ^ q_t [2] ^ In_t [2]

q_{t+1} [9] = q_t [14] ^ q_t [3] ^ In_t [3]

q_{t+1} [8] = q_t [15] ^ q_t [0] ^ In_t [0] ^ q_t [4] ^ In_t [4]

q_{t+1} [7] = q_t [0] ^ In_t [0] ^ q_t [1] ^ In_t [1] ^ q_t [5] ^ In_t [5]

q_{t+1} [6] = q_t [1] ^ In_t [1] ^ q_t [2] ^ In_t [2] ^ q_t [6] ^ In_t [6]

q_{t+1} [5] = q_t [2] ^ In_t [2] ^ q_t [3] ^ In_t [3] ^ q_t [7] ^ In_t [7]

q_{t+1} [4] = q_t [3] ^ In_t [3]

q_{t+1} [3] = q_t [0] ^ In_t [0] ^ q_t [4] ^ In_t [4]

q_{t+1} [2] = q_t [1] ^ In_t [1] ^ q_t [5] ^ In_t [5]

q_{t+1} [1] = q_t [2] ^ In_t [2] ^ q_t [6] ^ In_t [6]

q_{t+1} [0] = q_t [3] ^ In_t [3] ^ q_t [7] ^ In_t [7] **(1)**

See Annex E for examples of 16-bit ECS data values.

7.2.5 Spreader

Figure 7 and Table 6 specify the function of the spreader.

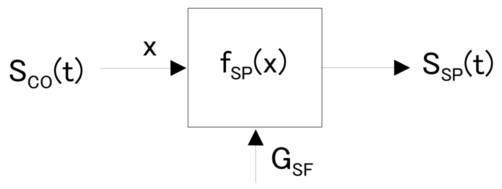


Figure 7 — Spreader

Table 6 — Input and output relationship of the Spreader

Spreading Factor: G_{SF}	Input: x (a = 1 or 0)	Output: $S_{SP}(t)$
1	a	a
2	a	a,a
4	a	a,a,a,a
8	a	a, … ,a : Repeat 8 times
16	a	a, … …,a : Repeat 16 times

7.2.6 Sync sequence

Table 7 specifies the Sync sequence, $C_{SY}(t)$, used for packet synchronization. $C_{SY}(t)$ consists of 128 chips and expects the transmission to start with the 0th index.

Table 7 — Sync sequence $C_{SY}(t)$

0	1	16	1	32	1	48	0	64	0	80	0	96	0	112	1
1	1	17	0	33	0	49	1	65	1	81	1	97	1	113	0
2	0	18	0	34	1	50	0	66	0	82	1	98	1	114	0
3	1	19	0	35	0	51	0	67	0	83	0	99	0	115	1
4	1	20	1	36	0	52	0	68	1	84	0	100	0	116	0
5	1	21	1	37	1	53	0	69	1	85	0	101	0	117	0
6	1	22	1	38	0	54	1	70	1	86	0	102	1	118	0
7	0	23	1	39	1	55	0	71	0	87	0	103	0	119	0
8	1	24	0	40	1	56	0	72	1	88	1	104	0	120	0
9	1	25	0	41	0	57	1	73	1	89	1	105	0	121	0
10	1	26	0	42	1	58	1	74	0	90	1	106	1	122	1
11	0	27	1	43	0	59	1	75	0	91	0	107	0	123	0
12	0	28	1	44	1	60	1	76	1	92	1	108	0	124	1
13	0	29	0	45	1	61	0	77	1	93	0	109	0	125	1
14	0	30	1	46	1	62	1	78	0	94	1	110	1	126	0
15	1	31	1	47	1	63	1	79	1	95	1	111	0	127	0

7.2.7 Scrambler

Figure 8 specifies the configuration of the scrambler. Table 8 specifies the scrambler truth table which uses an inverted XOR of the spread signal $S_{SP}(t)$ and the LFSR-generated signal.

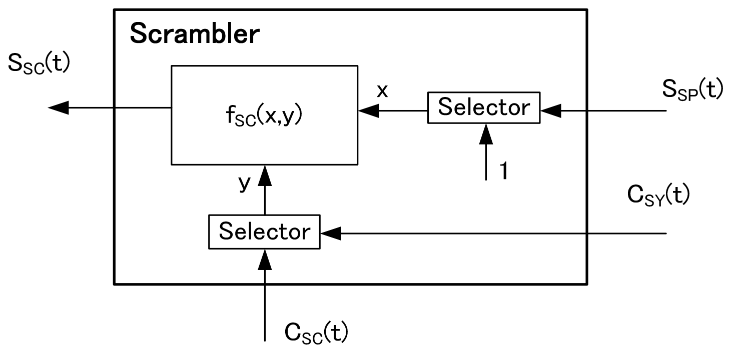


Figure 8 — Scrambler

Table 8 — Truth table of Scrambler

x	Y	$S_{SC}(t)$
0	0	1
0	1	0
1	0	0
1	1	1

As specified in Table 9 the scrambler uses scrambling sequences $C_{PR}(t)$, $C_{HE}(t)$, and $C_{PA}(t)$ for the Preamble, PHY Header, and Payload parts of the transmit packet, respectively. These sequences are generated by the scrambling sequence generator specified in 7.2.8. Note the Preamble and Sync parts use a constant 1 data input. Therefore $C_{PR}(t)$ and $C_{SY}(t)$ pass through the scrambler unmodified.

Table 9 — Input of Scrambler

Part of Packet	x Input of $f_{SC}(x,y)$	y Input of $f_{SC}(x,y)$
Preamble	1	$C_{SC}(t) = C_{PR}(t)$
Sync	1	$C_{SY}(t)$
PHY Header	$S_{SP}(t)$	$C_{SC}(t) = C_{HE}(t)$
Payload	$S_{SP}(t)$	$C_{SC}(t) = C_{PA}(t)$

7.2.8 Scrambling sequence generator

Figure 9 illustrates the configuration of the LFSR used to generate the scrambling sequence. Equation (2) specifies the polynomial equation for LFSR generation. Table 10 specifies the relationship between LFSR seeds and scrambling sequences. The scrambling sequence generator expects the register value to be initialized by the scrambling seed [17:0] at the beginning of each part of the packet (Preamble, PHY Header, and Payload).

$$G(x) = x^{18} + x^{10} + x^7 + x^5 + 1 \qquad \textbf{(2)}$$

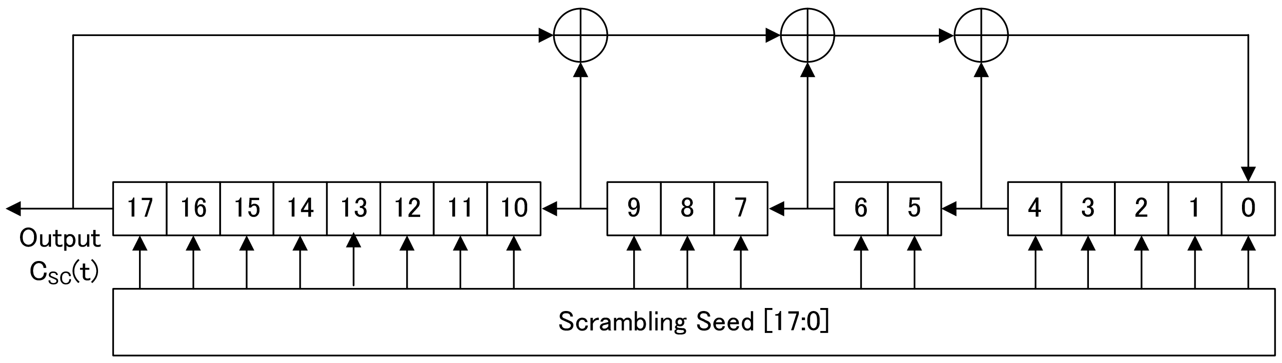


Figure 9 — Block diagram of Scrambling sequence generator LFSR

Table 10 — Scrambling sequence generator seeds and outputs

Part of Packet	Scrambling Seed [17:0]	Output of Scrambling Sequence Generator: $C_{SC}(t)$
Preamble	0x011A0	$C_{PR}(t)$
Header	0x27BFA	$C_{HE}(t)$
Payload	0x3C859	$C_{PA}(t)$

See Annex E for examples of Scrambling sequences.

7.2.9 Pi/2 shift BPSK mapper

Figure 10 specifies the configuration of the Pi/2 shift BPSK mapper. The Pi/2 shift BPSK mapper converts an input binary sequence to a complex output sequence.

$$\begin{tikzcd} S_{SC}(t) \arrow[r, "x"] & \boxed{f_{PI}(x)} \arrow[r] & S_{PI}(t) \end{tikzcd}$$

Figure 10 — Pi/2 shift BPSK mapper

Figure 11 illustrates the concept of Pi/2 shift BPSK modulation. Pi/2 shift BPSK expects the modulation axis for BPSK modulation to rotate by 90 degrees for each consecutive symbol. This implementation rotates the phase by 90 degrees for each new bit or chip from the scrambler. For each 4 chips, the modulation axis is rotated by a complete 360 degree cycle.

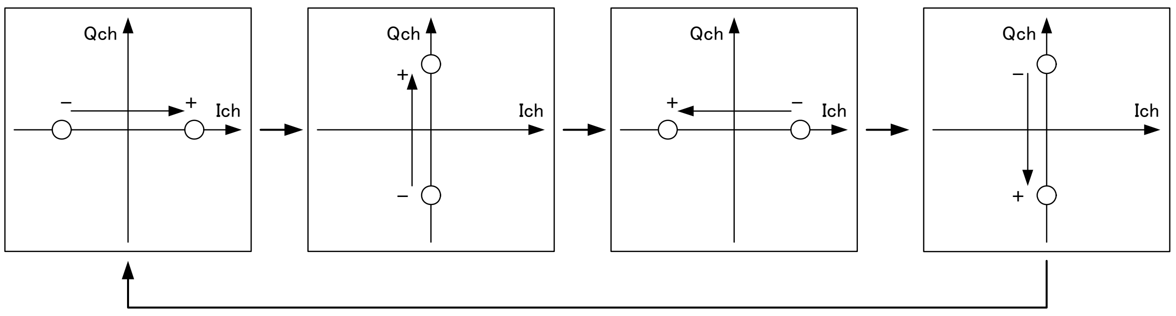


Figure 11 — Concept of Pi/2 shift BPSK

Table 11 specifies the input and output relationship of the Pi/2 shift BPSK mapper. The variable “n” represents a chip number. It can be seen that the values in the "n mod 4" column correspond to a unique phase rotator value.

Input signal $S_{SC}(t_n)$ indicates the scrambler binary output at time $t_n = n*T_C$. The function $(2* S_{SC}(t_n)-1)$ in the Output of Table 11 specifies the binary (0,1) to real (-1,1) transformation.

After the binary value is converted to a real value, the output sequence $S_{PI}(x)$ is multiplied by the rotator value +1, +j, -1, or -j.

Table 11 — Input and output relation of Pi/2 shift BPSK mapper

n mod 4	Rotator	Input:$S_{SC}(t)$	Output:$S_{PI}(x)$
0	1	$S_{SC}(t_0)$	$(2*S_{SC}(t_0)-1)$
1	j	$S_{SC}(t_1)$	$j * (2*S_{SC}(t_1)-1)$
2	-1	$S_{SC}(t_2)$	$- (2*S_{SC}(t_2)-1)$
3	-j	$S_{SC}(t_3)$	$- j * (2*S_{SC}(t_3)-1)$

7.2.10 Mathematical framework of the Up Converter and the Baseband Waveform Generator

$$S_{TX}(t) = \mathrm{Re}\left(S_{BB}(t) \cdot \exp(j2\pi F_c t)\right) \qquad \textbf{(3)}$$

$$S_{BB}(t) = S_{BW}(t) \otimes \sum_{n=0}^{N_{chip}-1} S_{PI}(t - n \cdot T_c) \qquad \textbf{(4)}$$

Equation (3) specifies the Up Converter. Equation (4) specifies the Baseband Waveform Generator.

In Equation (3), $S_{BB}(t)$ represents the transmission baseband sequence, which is illustrated in (Figure 13). F_c is the centre frequency, and Re(x) an operation that calculates the real part of a complex number.

In Equation (4), $S_{BW}(t)$ represents the transmission baseband waveform, $S_{PI}(t)$ a complex number delta function, N_{chip} the number of transmit chips in the packet, and $\otimes$ convolution. Note that $S_{PI}(t)$ has weight +1, +j, -1, or –j depending on the Pi/2 shift BPSK modulation.

7.2.11 Baseband waveform

Figure 12 specifies a transmission baseband waveform $S_{BW}(t)$ using discrete values. In Figure 12, one cycle of the waveform is represented by 8 samples, with the normalized sample numbers shown along the horizontal axis. A cycle of the waveform is $1/R_s$ (R_s = 280 Msps), which is the reciprocal number of the symbol rate. Table 12 specifies the amplitude values that appear in Figure 12.

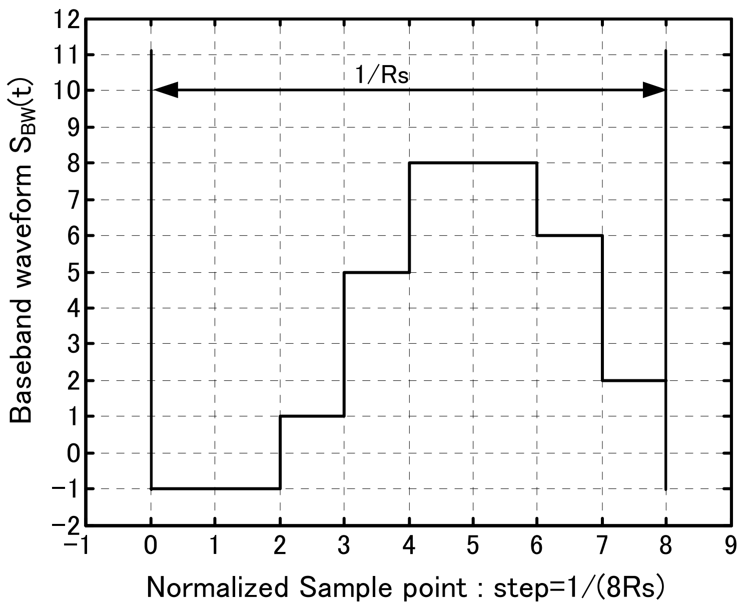


Figure 12 — Baseband waveform S_{BW}(t)

Table 12 — Numerical definition of baseband waveform

Normalized Sample point : step=1/(8Rs)	Amplitude value
0	-1
1	-1
2	1
3	5
4	8
5	8
6	6
7	2

Figure 13 illustrates the transmission baseband sequence S_{BB}(t). The solid line means the real part of S_{BB}(t), and the dashed line is the imaginary part. The imaginary part is delayed with half of symbol duration (1/2R_s) relative to the real part. The real part and imaginary part appear alternately in the complex envelope and the duration between a real pulse and an imaginary pulse is defined as the chip duration. So the mathematical relationship between chip and symbol duration becomes 1/R_S = 2*1/R_C.

In the Pi/2 shift modulation scheme, the modulation axis rotates Pi/2 from chip to chip in the Tx sequence. This means that the modulation axis changes I and Q alternately in chip duration.

Illustrated along the horizontal axis in Figure 13 are sample numbers, with one sample being 1/(8R_s).

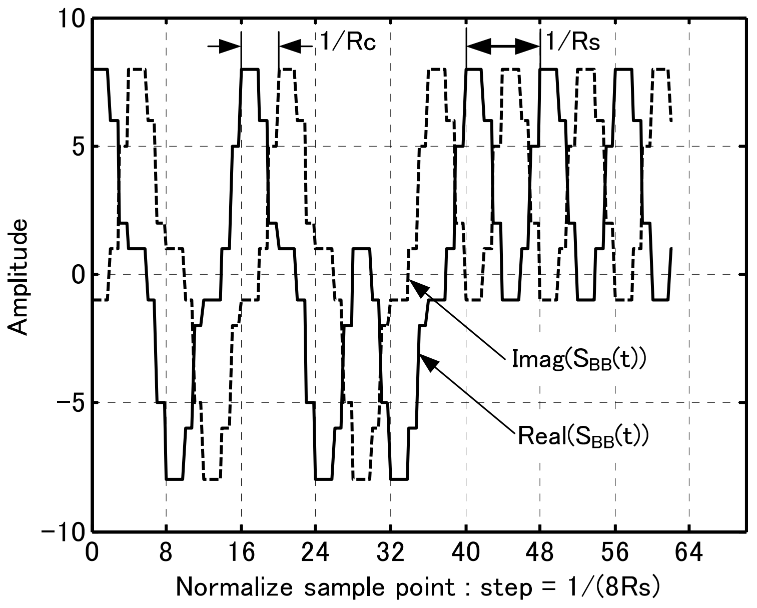


Figure 13 — Tx baseband sequence S_{BB}(t)

7.3 Frame format

7.3.1 PPDU format

Figure 14 specifies how CPDUs are encoded to form PPDUs. This figure expresses the case of 2 CSDU segments in the CPDU.

The PHY shall divide each PSDU into 224-byte message blocks and shall add RS (Reed-Solomon) parity after each 224-byte block. The size of the last message block will be 1 to 224 bytes as specified in Figure 14. RS parity of the last message block is calculated by expanding the message size to 224 bytes with zero padding. When encoding the zero padded message block, the zero pad bytes shall be encoded first. The added zeros of the last block are not transmitted and will be recovered at the receiver. The added zeros are also not transferred to the next stage of processing

For all rates except Rate 522, all bits output from the RS encoder, followed by 4 zero value tail bits, shall be input to the Convolutional encoder. The output length of the Convolutional encoder will be doubled due to the R = 1/2 Convolutional code.

Finally, the transmitter adds spreading, scrambling, and Pi/2 BPSK mapping to form the Transmit packet payload. This payload is combined with the Preamble, Sync word, and PHY Header to complete the Transmit packet (PPDU).

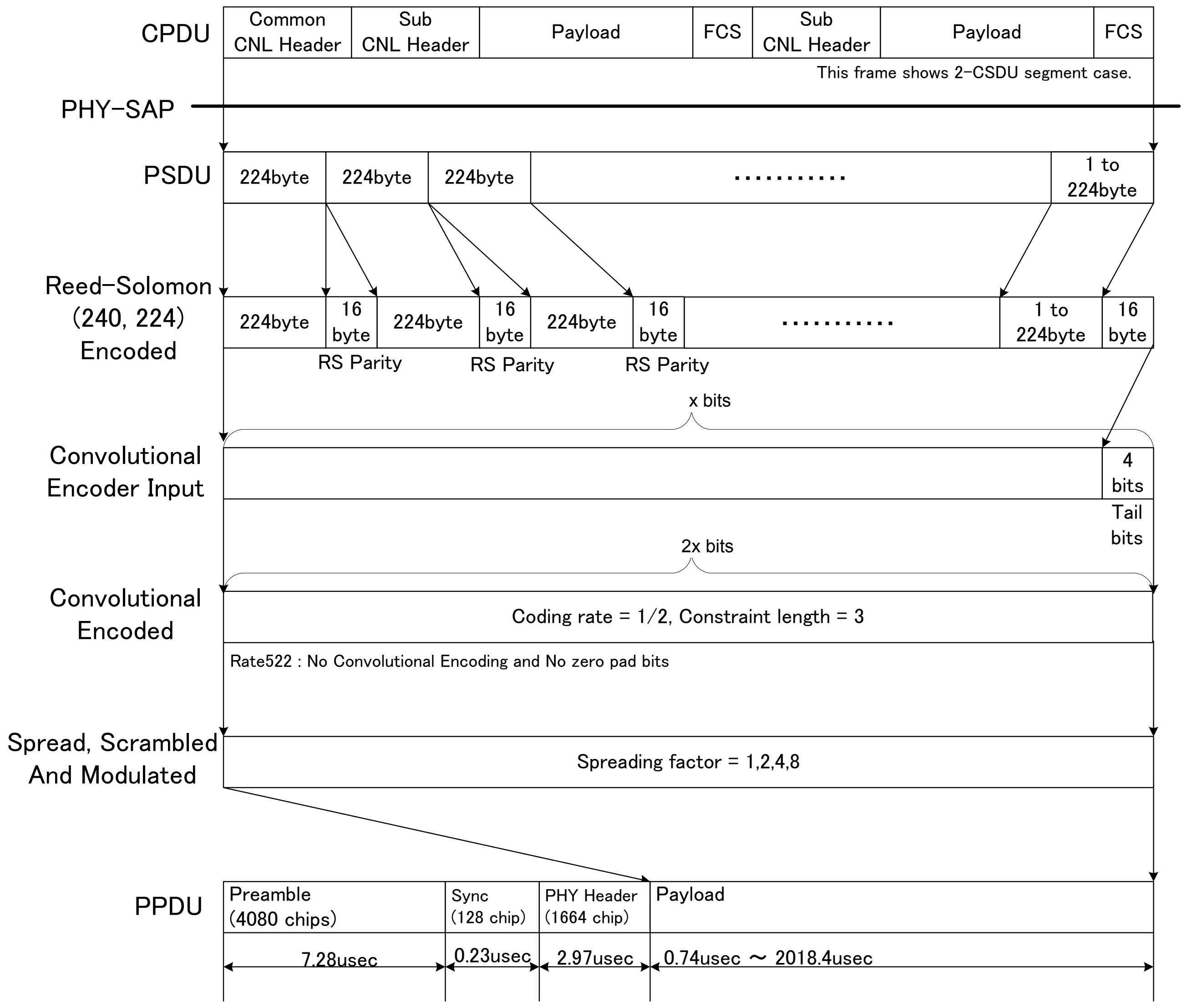


Figure 14 — CPDU encoding process and PPDU format

7.3.2 PHY Header format

Figure 15 specifies the format of the PHY Header. This PHY Header includes 5 fields. Table 13 specifies the details of the PHY Header.

Table 13 — PHY Header format description

Field Name	Size	Value	Description
Ver.	4 bit	0x1 In_0 [7:4]	Frame Format Version
Rate	4 bit	0x1-0x5 In_0[3:0]	Connection Layer Frame Rate 0x0: Reserved 0x1: Rate 32 0x2: Rate 65 0x3: Rate 130 0x4: Rate 261 0x5: Rate 522 0x6-0xF:Reserved
Reserved	8 bit	0x00 In_1[7:0]	Reserved
Length	16 bit	In_2[7:0] and In_3[7:0]	Length after Reed-Solomon encoding [Byte]
HCS	16 bit	q_4[15:0]	16 bit ECS that is calculated over the Version, Rate, Reserved, and Length fields of the PHY Header.

4 zero value tail bits, shall be added to the end of the PHY Header before the Convolutional encoder.

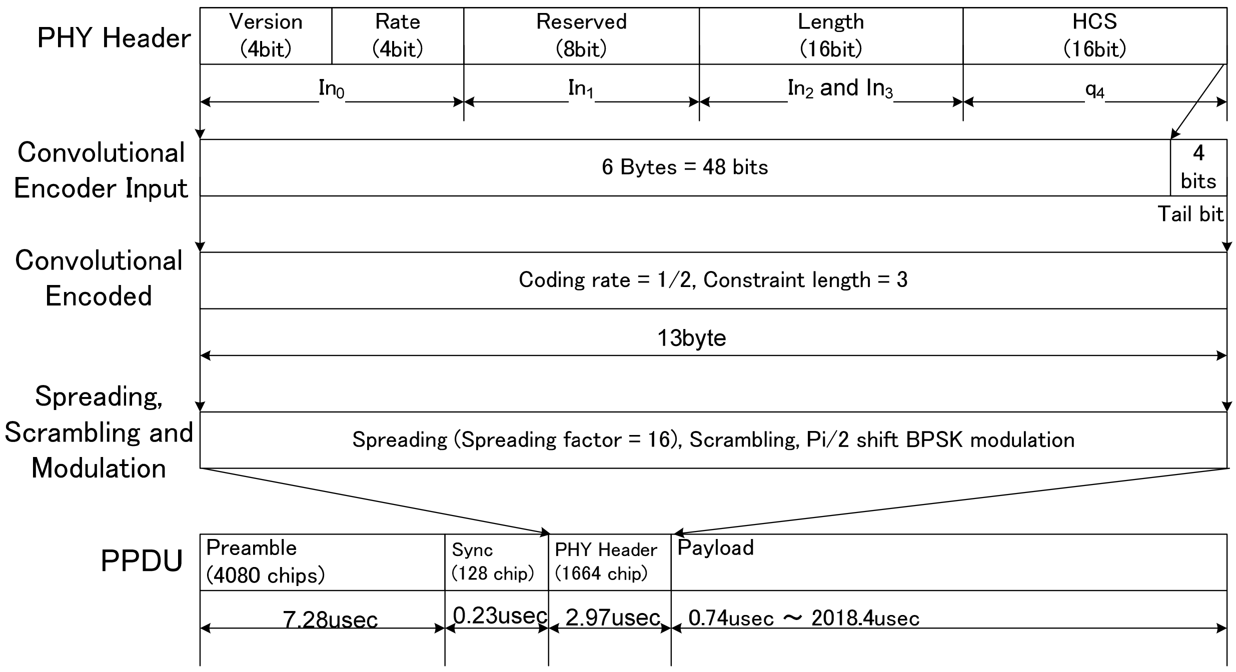


Figure 15 — PHY Header format

7.4 Transmitter

7.4.1 Measurement points

The transmit frequency, transmit symbol rate, and transmit constellation error shall be measured at the output of the transmitter unit (excluding cabling and coupler).

7.4.2 Transmit frequency

The tolerance of Fc is ± 50 x 10^{-6}.

7.4.3 Transmit clock rate requirement

Fc, Rc and Rs shall retain the relationship of Fc=8Rc=16Rs.

7.4.4 Transmit Constellation Error (EVM)

-20dB or less (including the impulse response of the root raised cosine filter α=0.78 used by the measurement equipment).

8 Receiver

8.1 Measurement point

The reference sensitivity and the blocking level shall be measured at the input to the receiver (excluding cabling and coupler).

8.2 Reference sensitivity

Table 14 specifies the reference sensitivity for each rate setting.
Reference sensitivity is defined as follows:

Packet Error Rate (PER) shall be less than 1% at each reference sensitivity point. CSDU size for the PER measurement is 1024 [bytes].

Table 14 — Reference sensitivity point

Rate Setting	Sensitivity[dBm]
Rate 522	-59
Rate 261	-65
Rate 130	-68
Rate 65	-71
Rate 32	-71

8.3 Blocking

Table 15 specifies the blocking levels when the desired signal input level is (Reference sensitivity + 3[dB]) for the data rate in use. The blocking signal is a non-modulated single carrier in each band.

The Packet Error Rate (PER) shall be less than 1% at the blocking level point.

Table 15 — Blocking level

Band	Blocking Level [dBm]
< 2,5 [GHz]	-10
2,5 – 3,8 [GHz]	-15
5,15 – 5,985 [GHz]	-15

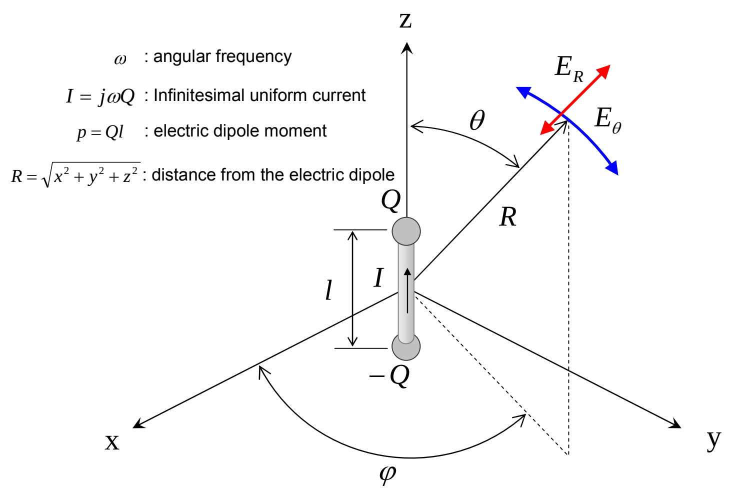
9 Electric Induction Field

The PHY layer shall use an electric induction field for transmitting and receiving data. This differs from a conventional radio that uses an electromagnetic radiation field.

Figure 16 illustrates the ideal electric dipole. Equation (5) illustrates the electric and magnetic field components in each direction generated by this electric dipole.

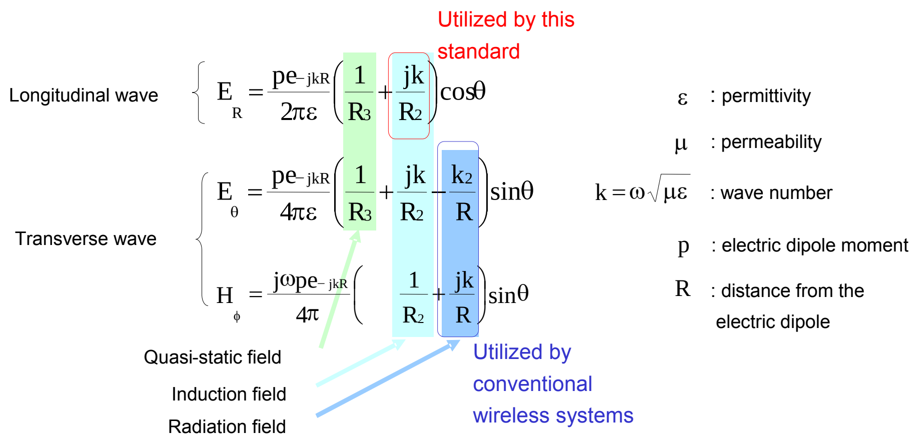
The component proportional to $1/R^3$ is the "quasi-static" field, which exists only on the surface of an antenna and is negligible compared to other fields. The component proportional to $1/R^2$ is the "induction field". The component proportional to 1/R is the "radiation field" which is used in conventional radio systems.

For this close proximity electric induction wireless communications standard, the PHY shall use the induction field component of E_R. This is the longitudinal wave of the electric induction field. The longitudinal wave has by nature no polarization effect compared with the transverse wave in E_θ and H_Φ.



ω : angular frequency
$I = j\omega Q$: Infinitesimal uniform current
$p = Ql$: electric dipole moment
$R = \sqrt{x^2+y^2+z^2}$: distance from the electric dipole
z
E_R
E_θ
θ
Q
R
l
I
$-Q$
x
y
φ

Figure 16 — Electric dipole and field coordinates



$$E_R = \frac{p e^{-jkR}}{2\pi\varepsilon}\left(\frac{1}{R^3} + \frac{jk}{R^2}\right)\cos\theta$$

$$E_\theta = \frac{p e^{-jkR}}{4\pi\varepsilon}\left(\frac{1}{R^3} + \frac{jk}{R^2} - \frac{k^2}{R}\right)\sin\theta$$

$$H_\phi = \frac{j\omega p e^{-jkR}}{4\pi}\left(\frac{1}{R^2} + \frac{jk}{R}\right)\sin\theta$$

ε	: permittivity
μ	: permeability
$k = \omega\sqrt{\mu\varepsilon}$	: wave number
p	: electric dipole moment
R	: distance from the electric dipole

(5)

10 CNL service definition

10.1 Overview of CNL services

10.1.1 Connection control service

The CNL supports the establishment and release of connections.

The CNL supports at most one point-to-point connection at any given time.

10.1.2 Data service

CNL service data units (CSDUs) provide a data transfer service using the PHY. This data transfer service realizes data transfer to/from a peer CNL. The CNL shall guarantee both the CSDU integrity and ordering.

10.1.3 Security service

No service is provided with relation to confidentiality or authentication.

10.2 CNL service access point

This clause specifies the CNL services provided to the CNL User.

Primitives and the details of these primitives are provided in this Standard to illustrate an example of how to access the CNL services. The general flow is illustrated in Figure 17 below.

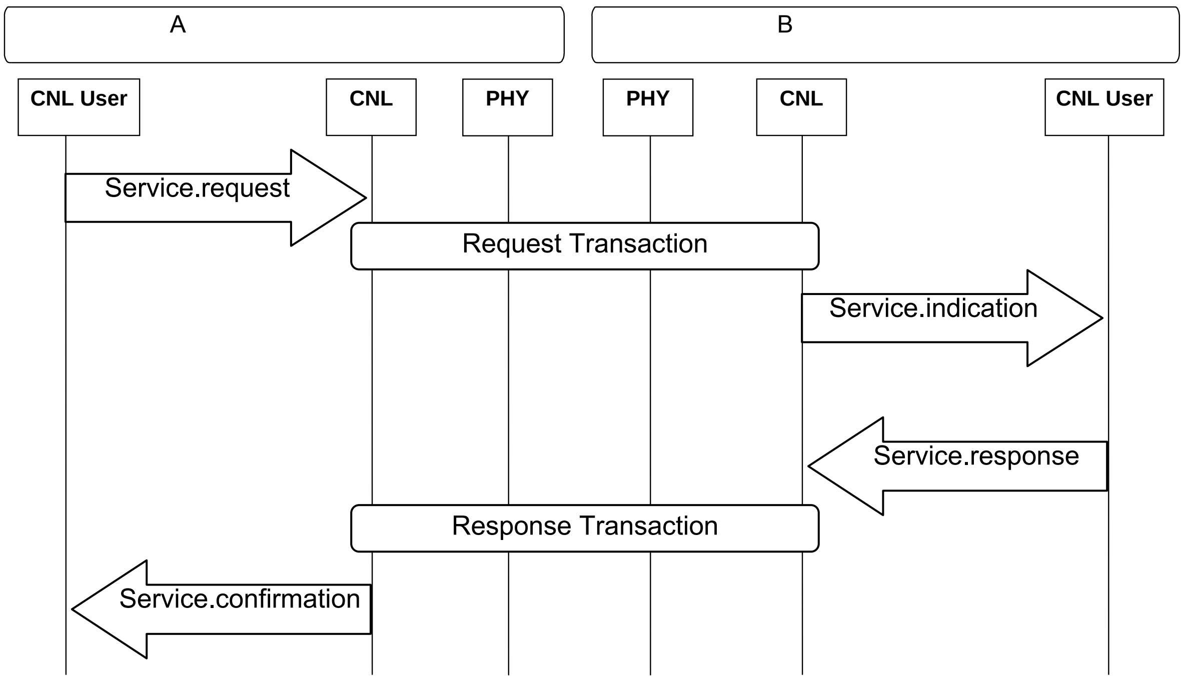


Figure 17 — CNL primitives between the CNL User and the CNL

Not all Services utilize every primitive type.

The details of reference MSCs Request Transaction and Response Transaction depend on the specific Service. See Annex F and G.

The CNL User uses the services provided by CNL through the CNL SAP.

The CNL services and parameters are specified in Table 16. The table also illustrates how each service is accessed by the example service primitives.

Table 16 — List of CNL services and parameters

Services	Primitive Type (informative)				Parameters
	Request	Indication	Response	Confirmation	
CNL_INIT	10.2.1.1				Initialization Parameters
CNL_CLOSE	10.2.2.1				
CNL_CONNECT	10.2.3.1				[Target UID/ Paging UID], CNL User Parameter
CNL_CONNECT		10.2.3.2			Target UID, CNL User Parameter
CNL_ACCEPT	10.2.3.3				Target UID, CNL User Parameter
CNL_ACCEPT		10.2.3.4			Target UID, CNL User Parameter
CNL_ACCEPT			10.2.3.5		Target UID
CNL_ACCEPT				10.2.3.6	Target UID
CNL_RELEASE	10.2.4.1				Reason Code
CNL_RELEASE		10.2.4.2			Release Cause Origin, Release Cause Code
CNL_POWERSAVE	10.2.5.1				T_KeepAlive, Dormant period, Awake period
CNL_POWERSAVE		10.2.5.2			Dormant period, Awake period
CNL_POWERSAVE				10.2.5.3	
CNL_WAKE	10.2.5.4				
CNL_WAKE		10.2.5.5			
CNL_WAKE				10.2.5.6	
CNL_DATA	10.2.6.1				CSDU Profile ID, Payload Length, Data Payload
CNL_DATA		10.2.6.2			CSDU Profile ID, Payload Length, Data Payload

10.2.1 Initialize

This service is used to reset the CNL into the Search State. CNL entities shall have access to the parameters specified in Table 17.

Table 17 — Initialization parameters

Name	Type	Valid range	Description
Own UID	Unique 64 bit ID	Any valid UID	Own entity UID
Max MUX Count	Integer	0x01, 0x02	Other values prohibited
LiCC Version	Integer	0x01	Other values prohibited

10.2.1.1 CNL_INIT.request

This primitive issues a request to initialize the CNL. The semantics of this primitive is as follows:

```
CNL_INIT.request（
    Initialization Parameters
    ）
```

10.2.2 Close

This service is used to close the CNL.

10.2.2.1 CNL_CLOSE.request

This primitive issues a request to close the CNL. The semantics of this primitive is as follows:

```
CNL_CLOSE.request（
    ）
```

10.2.3 Connect and accept

This service supports the target entity connect and accept process. CNL entities shall have access to the parameters specified in Table 18.

Table 18 — Connect and accept parameters

Name	Type	Valid range	Description
Target UID	Unique 64 bit ID	Any valid UID	Connection target entity UID.
CNL User Parameter			CNL User Specified Parameter

10.2.3.1 CNL_CONNECT.request

This primitive is used to request a connection with a Target Entity. The semantics of this primitive is as follows.

```
CNL_CONNECT.request（
    [Target UID/Paging UID]
    CNL User Parameter
    ）
```

10.2.3.2 CNL_CONNECT.indication

This primitive is used to indicate a received C-Req. The semantics of this primitive is as follows.

```
CNL_CONNECT.indication（
    Target UID
    CNL User Parameter
    ）
```

10.2.3.3 CNL_ACCEPT.request

This primitive is used to accept a connection with a Target Entity. The semantics of this primitive is as follows.

```
CNL_ACCEPT.request（
    Target UID
    CNL User Parameter
    ）
```

10.2.3.4 CNL_ACCEPT.indication

This primitive is used to indicate a received C-Acc. The semantics of this primitive is as follows.

```
CNL_ACCEPT.indication（
    Target UID
    CNL User Parameter
    ）
```

10.2.3.5 CNL_ACCEPT.response

This primitive is used to indicate a response to a CNL_ACCEPT.indication. The semantics of this primitive is as follows.

```
    CNL_ACCEPT.response（
        Target UID
）
```

10.2.3.6 CNL_ACCEPT.confirmation

This primitive is used to inform the originating CNL User of a successful connection. The semantics of this primitive is as follows.

```
CNL_ACCEPT.confirmation（
    Target UID
             ）
```

10.2.4 Connection release

This service is used to release the connection with the target entity. CNL entities shall have access to the parameters specified in Table 19.

Table 19 — Connection release parameters

Name	Type	Valid range	Description
Reason Code	Enumeration		This parameter is available for use by the CNL User
Release Cause Origin	Enumeration	CNL, CNL User	Indicates the origin of the connection release event.
Release Cause Code	Enumeration	T_Connect Timeout, T_Accept Timeout, T_Retry Timeout, Reason Code	Indicates the cause for the connection release to CNL User. If the CNL detects a timeout event, then it indicates T_Connect Timeout, T_Accept Timeout or T_Retry Timeout as a Release Cause Code. If the CNL receives a C-RIs, then it indicates the Reason Code which is specified by the target CNL User as a Release Cause Code.

10.2.4.1 CNL_RELEASE.request

This primitive is used to release a connection. The semantics of this primitive is as follows.

```
CNL_RELEASE.request（
  Reason Code ［to be specified by CNL User］
     ）
```

This Reason Code is used to indicate the reason to release the connection to the target entity.

This Reason Code is transmitted to the target entity by a C-RIs.

10.2.4.2 CNL_RELEASE.indication

This primitive is used to indicate the reception of a C-RIs or a timeout. The semantics of this primitive is as follows.

```
CNL_RELEASE.indication（
  Release Cause Origin [CNL Origin, CNL User Origin],
  Release Cause Code ［T_Connect Timeout, T_Accept Timeout, T_Retry Timeout, Reason Code (to be specified by target CNL User)］
     ）
```

In case the release is caused by a timeout event:

Release Cause Origin = CNL Origin

Release Cause Code = T_Connect Timeout, T_Accept Timeout or T_Retry Timeout

In case the release is caused by a C-Rls:

Release Cause Origin = CNL User Origin

Release Cause Code = Reason Code (to be specified by target CNL User)

10.2.5 Power save

This service is used to enter and exit the power save mode. CNL entities shall have access to the parameters specified in Table 20.

Table 20 — Power save parameters

Name	Type	Valid range	Description
T_KeepAlive	Integer	Max 1 s	This value is used to monitor the idle time of the physical medium while two entities are in a connected state. If the T_KeepAlive timer exceeds this value, then C-Probe is transmitted to confirm the availability of the connection.
Dormant period	Integer	1 to 200	Indicates the dormant periods during the Local Hibernate sub-state. Refer to Table 35.
Awake period	Integer	0 to 255	Indicates the awake periods during the Local Hibernate sub-state. Refer to Table 35.

These parameters impact the timing behaviour as specified in clause 10.5.9.2.

10.2.5.1 CNL_POWERSAVE.request

This primitive is used to change the local Entity to the Local Hibernate sub-state. The semantics of this primitive is as follows.

```
CNL_POWERSAVE.request（
    T_KeepAlive
    Dormant period
    Awake period
    ）
```

10.2.5.2 CNL_POWERSAVE.indication

This primitive is used to indicate the reception of a C-Sleep. The semantics of this primitive is as follows.

```
CNL_POWERSAVE.indication（
    Dormant period
    Awake period
    ）
```

10.2.5.3 CNL_POWERSAVE.confirmation

This primitive is used to inform the originating CNL User of a successful C-Sleep transmission.

```
    CNL_POWERSAVE.confirmation （
        ）
```

10.2.5.4 CNL_WAKE.request

This primitive is used to change the local Entity from the Local Hibernate sub-state to the Connected sub-state. This primitive is also used to change the target to the Connected sub-state. The semantics of this primitive is as follows.

```
    CNL_WAKE.request （
        ）
```

10.2.5.5 CNL_WAKE.indication

This primitive is used to indicate the reception of a C-Wake. The semantics of this primitive is as follows.

```
    CNL_WAKE.indication （
        ）
```

10.2.5.6 CNL_WAKE.confirmation

This primitive is used to change the local Entity from the Target Sleep sub-state to the Connected sub-state. This primitive is used to inform the originating CNL User of a successful C-Wake transmission. The semantics of this primitive is as follows.

```
    CNL_WAKE.confirmation （
）
```

10.2.6 Data transfer

This service is used to transfer user data. CNL entities shall have access to the parameters specified in Table 21.

Table 21 — Data transfer parameters

Name	Type	Valid range	Description
CSDU Profile ID	Integer	0/1	These Profile ID values are available for use by the CNL User and therefore out of scope from this specification.
Payload Length	Integer		Specifies the number of bytes of Data Payload.
Data Payload	Octet string		Specifies the data passing the CNL SAP before transmission or after reception.

10.2.6.1 CNL_DATA.request

This primitive is used to initiate the transfer of user data from one entity CNL to another peer CNL. The semantics of this primitive is as follows.

```
CNL_DATA.request（
    CSDU Profile ID［0／1］
    Payload Length
    Data Payload
    ）
```

10.2.6.2 CNL_DATA.indication

This primitive is used to inform the CNL User of a successful reception. The semantics of this primitive is:

```
CNL_DATA.Indication（
    CSDU Profile ID［0／1］
    Payload Length
    Data Payload
    ）
```

10.3 CPDU formats

10.3.1 Conventions

Every CNL protocol data unit (CPDU) shall be transmitted to the PHY layer starting from b7 (MSB) of Byte 00 through b0 (LSB), then from b7 (MSB) of Byte 01 through b0 (LSB), and so on as illustrated in Figure 18.

Reserved fields should be set to 0 upon transmission. And reserved fields shall be ignored upon reception.

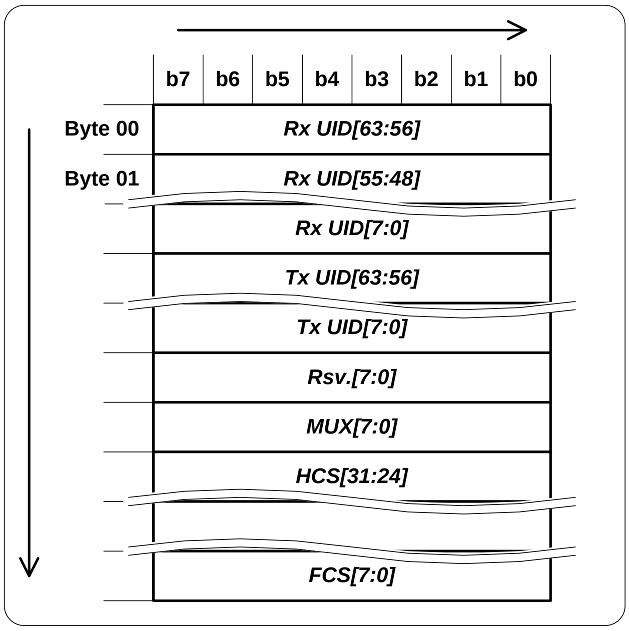


Figure 18 — Conventions

10.3.2 Acknowledgement (ACK) CPDU

The ACK CPDU format is specified in Figure 19 and Table 22 below.

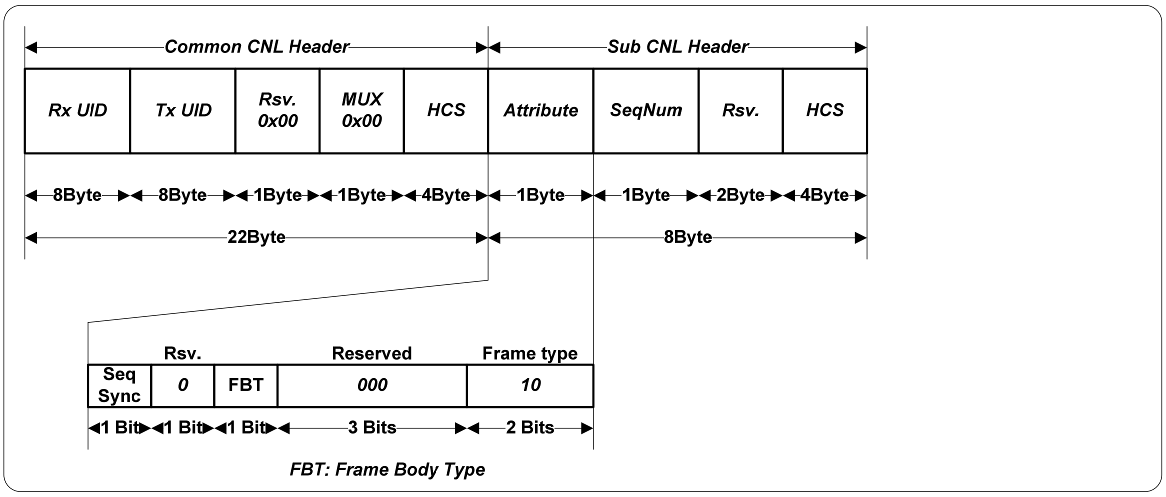


Figure 19 — ACK CPDU

Table 22 — ACK CPDU

Header	Field	Description
Common	Rx UID	This field indicates the UID of the destination entity of the CNL frame. A 64-bit UID is used. The value of the Tx UID field of the last received data frame or management frame shall be copied to the Rx UID field of the ACK frame.
	Tx UID	This field indicates the UID of the source entity of the CNL frame. A 64-bit UID is used. This field shall be set to the sending entity's Own UID.
	Rsv.	This field is reserved.
	MUX	This field indicates the number of multiplexing of the Frame Body. This field shall be set to 0.
	HCS	This field holds the 32-bit ECS that is calculated over the 18 bytes from Rx UID to MUX of the common CNL header. Refer to 10.3.2.1
Sub	Attribute	Bit [7]: Sequence Number Synchronization This bit shall be set to 1 only for the ACK response of the first data Frame Body with the Sequence Number Synchronization bit set to 1, and shall be set to 0 in other cases of an ACK response. Refer to 10.4.3.2. Bit [6]: Reserved This bit shall be set to 0. Bit [5]: Frame Body type This bit shall be set to 0 in case of an ACK for data CPDU, and shall be set to 1 in case of an ACK for Management CPDU. Bits [4:2]: Reserved These bits shall be set to 0. Bits [1:0]: Frame type This field shall be set to 2. See Table 23 for more details.
	SeqNum	This field indicates a sequence number that the CNL uniquely assigns for Frame Body management. Refer to 10.4.3.
	Rsv.	This field is reserved.
	HCS	This field holds the 32-bit ECS that is calculated over the 4 bytes from Attribute to Rsv of the Sub CNL Header. Refer to 10.3.2.1.

The Paging UID is 0xFF:FF:FF:FF:FF:FF:FF:FF.

Other values for the frame type field are specified in Table 23 as follows.

Table 23 — Frame type field encoding

Value	Frame type
0	Reserved
1	Data/Management frame
2	Acknowledge frame
3	Reserved

10.3.2.1 HCS

This field holds the 32-bit ECS.

Figure 20 and Equation (6) specify the 32-bit ECS.

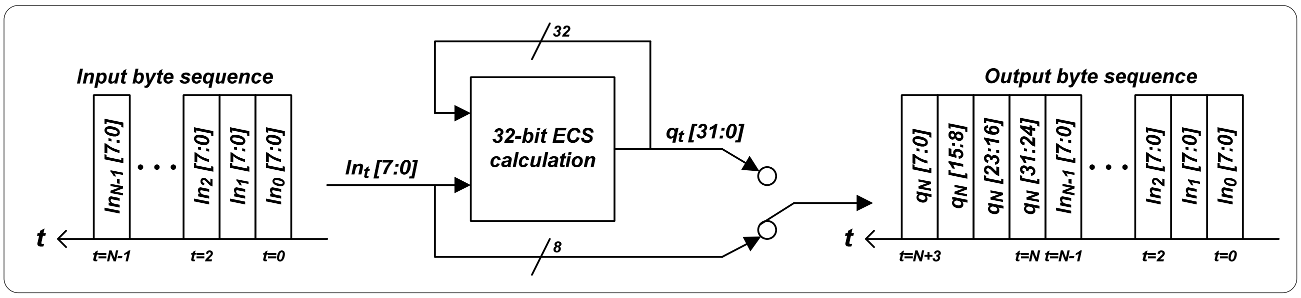


Figure 20 - 32-bit ECS generator

32-bit ECS calculation

$q_0[31:0]$=0x FF,FF,FF,FF

$In_0[7:0]$ is the 1st byte in the ECS target field to be sent.

“^” denotes the XOR (exclusive OR) operator in Equation (6).

q_{t+1} [31] = q_t [23] ^ q_t [29] ^ In_t [5] ^q_t [24] ^ In_t [0] ^ q_t [30] ^ In_t [6];

q_{t+1} [30] = q_t [22] ^ q_t [28] ^ In_t [4] ^ q_t [29] ^ In_t [5];

q_{t+1} [29] = q_t [21] ^ q_t [27] ^ In_t [3] ^ q_t [28] ^ In_t [4] ^ q_t [31] ^ In_t [7] ^ q_t [25] ^ In_t [1];

q_{t+1} [28] = q_t [20] ^ q_t [26] ^ In_t [2] ^ q_t [27] ^ In_t [3] ^ q_t [30] ^In_t [6] ^ q_t [24] ^ In_t [0];

q_{t+1} [27] = q_t [19] ^ q_t [25] ^ In_t [1] ^ q_t [26] ^ In_t [2] ^ q_t [29] ^ In_t [5];

q_{t+1} [26] = q_t [18] ^ q_t [24] ^ In_t [0] ^ q_t [25] ^ In_t [1] ^ q_t [28] ^ In_t [4];

q_{t+1} [25] = q_t [17] ^ q_t [24] ^ In_t [0] ^ q_t [27] ^ In_t [3];

q_{t+1} [24] = q_t [16] ^ q_t [26] ^ In_t [2];

q_{t+1} [23] = q_t [15] ^ q_t [26] ^ In_t [2] ^ q_t [27] ^ In_t [3] ^ q_t [31] ^ In_t [7];

q_{t+1} [22] = q_t [14] ^ q_t [25] ^ In_t [1] ^ q_t [26] ^ In_t [2] ^ q_t [30] ^ In_t [6];

q_{t+1} [21] = q_t [13] ^ q_t [24] ^ In_t [0] ^ q_t [25] ^ In_t [1] ^ q_t [29] ^ In_t [5];

q_{t+1} [20] = q_t [12] ^ q_t [24] ^ In_t [0] ^ q_t [28] ^ In_t [4];

q_{t+1} [19] = q_t [11] ^ q_t [27] ^ In_t [3];

q_{t+1} [18] = q_t [10] ^ q_t [26] ^ In_t [2];

q_{t+1} [17] = q_t [9] ^ q_t [31] ^ In_t [7];

q_{t+1} [16] = q_t [8] ^ q_t [30] ^ In_t [6] ^ q_t [31] ^ In_t [7] ^ q_t [25] ^ In_t [1];

q_{t+1} [15] = q_t [7] ^ q_t [27] ^ In_t [3] ^ q_t [28] ^ In_t [4] ^ q_t [30] ^ In_t [6] ^ q_t [31] ^ In_t [7];

q_{t+1} [14] = q_t [6] ^ q_t [26] ^ In_t [2] ^ q_t [27] ^ In_t [3] ^ q_t [29] ^ In_t [5] ^ q_t [30] ^ In_t [6];

q_{t+1} [13] = q_t [5] ^ q_t [26] ^ In_t [2] ^ q_t [28] ^ In_t [4] ^ q_t [29] ^ In_t [5] ^ q_t [31] ^ In_t [7];

q_{t+1} [12] = q_t [4] ^ q_t [27] ^ In_t [3] ^ q_t [28] ^ In_t [4] ^ q_t [30] ^ In_t [6] ^ q_t [31] ^ In_t [7];

q_{t+1} [11] = q_t [3] ^ q_t [26] ^ In_t [2] ^ q_t [27] ^ In_t [3] ^ q_t [29] ^ In_t [5] ^ q_t [30] ^ In_t [6] ^ q_t [31] ^ In_t [7] ^ q_t [25] ^ In_t [1];

q_{t+1} [10] = q_t [2] ^ q_t [25] ^ In_t [1] ^ q_t [26] ^ In_t [2] ^ q_t [28] ^ In_t [4] ^ q_t [29] ^ In_t [5] ^ q_t [30] ^ In_t [6] ^ q_t [24] ^ In_t [0];

q_{t+1} [9] = q_t [1] ^ q_t [24] ^ In_t [0] ^ q_t [25] ^ In_t [1] ^ q_t [27] ^ In_t [3] ^ q_t [28] ^ In_t [4] ^ q_t [29] ^ In_t [5];

q_{t+1} [8] = q_t [0] ^ q_t [24] ^ In_t [0] ^ q_t [26] ^ In_t [2] ^ q_t [27] ^ In_t [3] ^ q_t [28] ^ In_t [4];

q_{t+1} [7] = q_t [25] ^ In_t [1] ^ q_t [31] ^ In_t [7];

q_{t+1} [6] = q_t [30] ^ In_t [6] ^ q_t [24] ^ In_t [0] ^ q_t [31] ^ In_t [7] ^ q_t [25] ^ In_t [1];

q_{t+1} [5] = q_t [29] ^ In_t [5] ^ q_t [30] ^ In_t [6] ^ q_t [24] ^ In_t [0] ^ q_t [31] ^ In_t [7] ^ q_t [25] ^ In_t [1];

q_{t+1} [4] = q_t [28] ^ In_t [4] ^ q_t [29] ^ In_t [5] ^ q_t [30] ^ In_t [6] ^ q_t [24] ^ In_t [0];

q_{t+1} [3] = q_t [27] ^ In_t [3] ^ q_t [28] ^ In_t [4] ^ q_t [29] ^ In_t [5] ^ q_t [31] ^ In_t [7] ^ q_t [25] ^ In_t [1];

q_{t+1} [2] = q_t [26] ^ In_t [2] ^ q_t [27] ^ In_t [3] ^ q_t [28] ^ In_t [4] ^ q_t [30] ^ In_t [6] ^ q_t [24] ^ In_t [0] ^ q_t [31] ^ In_t [7] ^ q_t [25] ^ In_t [1];

q_{t+1} [1] = q_t [25] ^ In_t [1] ^ q_t [26] ^ In_t [2] ^ q_t [27] ^ In_t [3] ^ q_t [29] ^ In_t [5] ^ q_t [30] ^ In_t [6] ^ q_t [24] ^ In_t [0];

q_{t+1} [0] = q_t [24] ^ In_t [0] ^ q_t [26] ^ In_t [2] ^ q_t [28] ^ In_t [4] ^ q_t [29] ^ In_t [5] ^ q_t [31] ^ In_t [7]; **(6)**

See Annex E for examples of 32-bit ECS data values.

10.3.3 CNL data CPDUs

These are the CPDUs used for the CNL data transfer service. A data CPDU shall contain one or two Frame Bodies. Each Frame Body shall contain a single CSDU, or a portion of a CSDU (called a CSDU segment). CSDUs shall be sent in the order received from the CNL User.

The Single Data CPDU format is specified in Figure 21.

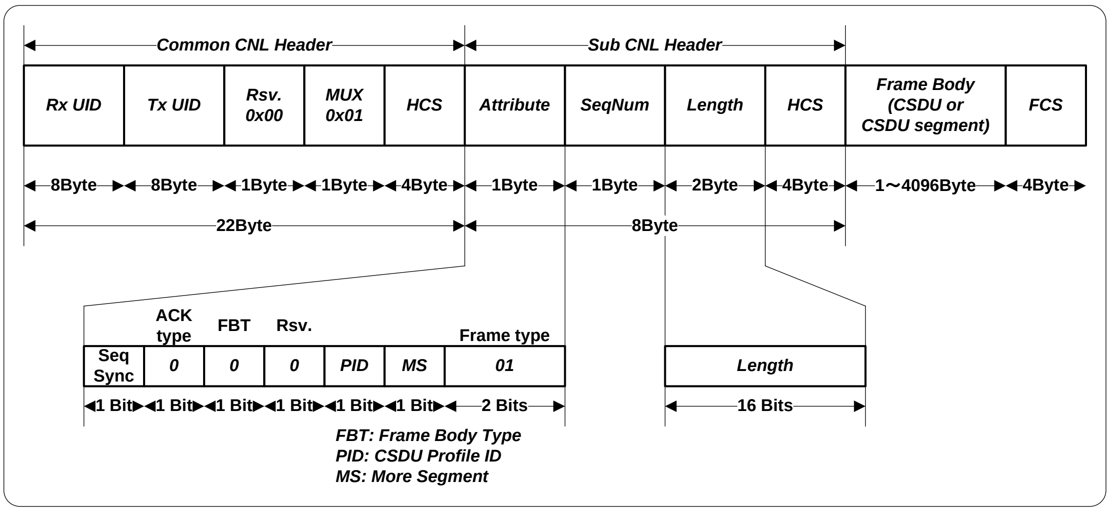


Figure 21 — Single Data CPDU

The Multi-Data CPDU format is specified in Figure 22. This format uses a blocking mechanism to carry multiple Frame Bodies in a single CPDU.

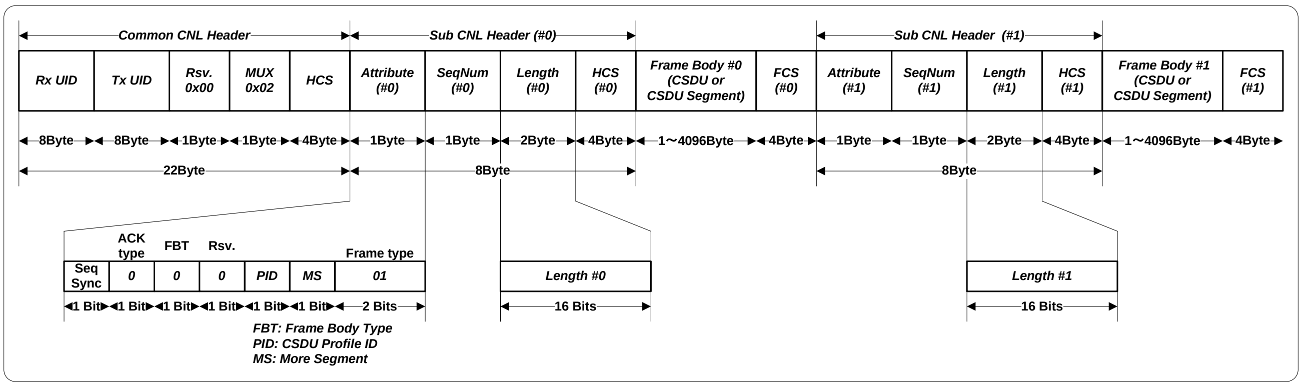


Figure 22 — Multi-Data CPDU

Table 24, Table 25, Table 26, Table 27, and Table 28 specify the header fields in the Data CPDU.

Table 24 —Data CPDU Header Fields

Header	Field	Description
Common	Rx UID	This field indicates the UID of the destination entity of the CNL frame. A 64-bit UID is used. This field shall not be set to the Paging UID.
	Tx UID	This field indicates the UID of the source entity of the CNL frame. A 64-bit UID is used. This field shall be set to the sending entity's Own UID.
	Rsv.	This field is reserved.
	MUX	This field indicates the number of multiplexing of the Frame Body. This field shall be set to 1 for Single Data CPDUs and 2 for Multi-Data CPDUs.
	HCS	This field holds the 32-bit ECS that is calculated based on the 18 bytes from Rx UID to MUX of the Common CNL Header. Refer to 10.3.2.1
Sub	Attribute	Bit [7] : Sequence Number Synchronization This bit shall be set to 1 only for the first data Frame Body, and shall be set to 0 in other cases. Refer to 10.4.3.2. Bit [6] : ACK type This field shall be set to 0. See Table 25 for more details. Bit [5] : Frame Body type This field shall be set to 0. Table 26 lists the frame type values and descriptions Bit [4] : Reserved This bit shall be set to 0. Bit [3] : CSDU Profile ID The CSDU Profile ID bit shall be set to 0 if the CSDU includes Profile ID 0. Otherwise this bit shall be set to 1. Bit [2] : More Segment The More Segment bit shall be set to 0 if the current segment is the sole final segment of the current CSDU. Otherwise this bit shall be set to 1. See 10.4.1. Bits [1:0]: Frame type This field shall be set to 1. Table 23 lists the frame type values and descriptions.
	SeqNum	This field indicates a sequence number that the CNL uniquely assigns for Frame Body management. Refer to 10.4.3.
	Length	This field indicates the Frame Body length (FCS not included). The unit is the byte.
	HCS	This field holds the 32-bit ECS that is calculated over the 4 bytes from Attribute to Length. Refer to 10.3.2.1.

Table 25 — ACK type field encoding

Value	ACK type
0	Immediate ACK
1	No ACK

Table 26 — Frame Body type field encoding

Value	Frame Body type
0	Data frame (CSDU)
1	Management frame (Link Control message)

Table 27 — CSDU Profile ID

CSDU Profile ID value	Description
0	CSDU Profile ID 0
1	CSDU Profile ID 1

Table 28 — More Segment

More segment	Description
0	Final segment
1	More segments

10.3.3.1.1 Frame Body field

The Frame Body is of variable length (1 to 4096 bytes).

The FCS field holds the 32-bit ECS that is calculated over the Frame Body.

Refer to 10.3.2.1.

10.3.4 Management CPDUs (Link control message)

These CPDUs are used for the connection establishment and release of connection, and other control purposes.

The management CPDU format is specified in Figure 23 and Table 29 below.

The length of the Frame Body of the management CPDU is fixed to 32 bytes.

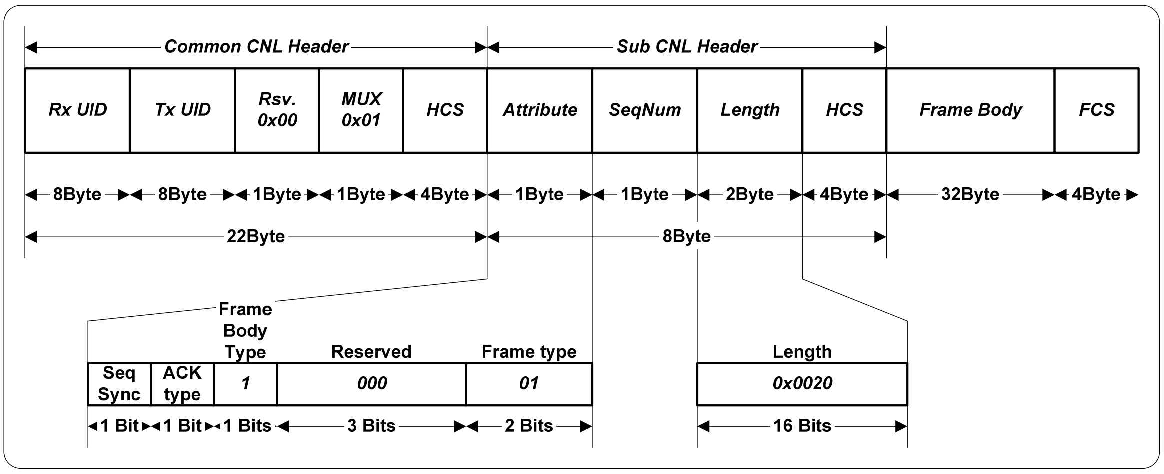


Figure 23 — Management CPDU (Link Control Message)

Table 29 — Management CPDU

Header	Field	Description
Common	Rx UID	This field indicates the UID of the destination entity. A 64-bit UID is used. Refer to Table 30.
	Tx UID	This field indicates the UID of the source entity. A 64-bit UID is used. This field shall be set to the sending entity's Own UID.
	Rsv.	This field is reserved.
	MUX	This field indicates the number of multiplexing of the Frame Body. This field shall be set to 1.
	HCS	This field holds the 32-bit ECS that is calculated over the 18 bytes from Rx UID to MUX of the Common CNL Header. Refer to 10.3.2.1.
Sub	Attribute	Bit [7] : Reserved Bit [6] : ACK type Refer to Table 25 and Table 30. Bit [5] : Frame Body type This field shall be set to 1. See Table 26 for more details. Bits[4:2] : Reserved This bit shall be set to 0.

		Bits [1:0]: Frame type This field shall be set to 1. See Table 23 for more details.
	SeqNum	This field indicates a sequence number that the CNL uniquely assigns for Frame Body management. Refer to 10.4.3.
	Length	This field indicates the Frame Body length (FCS not included). The unit is the byte. This field shall be set to 32.
	HCS	This field holds the 32-bit ECS that is calculated using the 4 bytes from Attribute to Length of the Sub CNL Header. Refer to 10.3.2.1.

The relationship between LiCC Type, ACK type and Rx UID is specified below in Table 30.

Table 30 — Frame Body contents, ACK type and allowable Rx UID type

Frame Body contents	LiCC Type	ACK type value	Allowable Rx UID type
Connection request (C-Req)	0x01	1 (NoACK)	Any UID (including Paging UID)
Connection accept (C-Acc)	0x02	0 (ImACK)	Any UID except the Paging UID
Connection release (C-Rls)	0x03	1 (NoACK)	Any UID except the Paging UID
Connection sleep (C-Sleep)	0x08	0 (ImACK)	Any UID except the Paging UID
Connection wake （C-Wake）	0x09	0 (ImACK)	Any UID except the Paging UID
Connection probe (C-Probe)	0x0A	0 (ImACK)	Any UID except the Paging UID

The Frame Body of the management CPDU is specified in Figure 24 below.

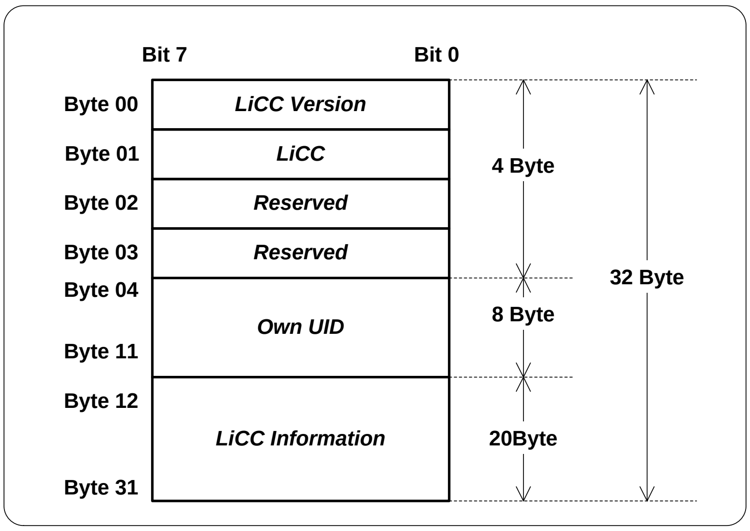


Figure 24 — Management Frame Body components

The Frame Body of each management CPDU is described below.

10.3.4.1 Connection request Frame Body

The message content of the connection request Frame Body is specified in Table 31 below.

Table 31 — Connection request Frame Body components

Component name	Value	Description
LiCC Version	0x01	Link control command version
LiCC	0x01	Connection request
Reserved	0x00	Reserved
Reserved	0x00	Reserved
Own UID		Own UID [Byte 04,Bit 7]: Own UID [63] [Byte 04,Bit 6]: Own UID [62] … [Byte 04,Bit 0]: Own UID [56] [Byte 05,Bit 7]: Own UID [55] … [Byte 11,Bit 0]: Own UID [0]
LiCC Information		For CNL User Parameter field. Refer to 10.2.3.1.

10.3.4.2 Connection accept Frame Body

The message content of the connection accept Frame Body is specified Table 32 below.

Table 32 — Connection accept Frame Body components

Component name	Value	Description
LiCC Version	0x01	Link control command version
LiCC	0x02	Connection accept
Reserved	0x00	Reserved
Reserved	0x00	Reserved
Own UID		Own UID [Byte 04,Bit 7]: Own UID [63] [Byte 04,Bit 6]: Own UID [62] … [Byte 04,Bit 0]: Own UID [56] [Byte 05,Bit 7]: Own UID [55] … [Byte 11,Bit 0]: Own UID [0]
LiCC Information		For CNL User Parameter field. Refer to 10.2.3.3.

10.3.4.3 Connection release Frame Body

The message content of the connection release Frame Body is specified in Table 33 below.

Table 33 — Connection release Frame Body components

Component name	Value	Description
LiCC Version	0x01	Link control command version
LiCC	0x03	Connection release
Reserved	0x00	Reserved
Reserved	0x00	Reserved
Own UID		Own UID [Byte 04,Bit 7]: Own UID [63] [Byte 04,Bit 6]: Own UID [62] … [Byte 04,Bit 0]: Own UID [56] [Byte 05,Bit 7]: Own UID [55] … [Byte 11,Bit 0]: Own UID [0]
LiCC Information		For CNL User Reason code field. Refer to 10.2.4.1.

10.3.4.4 Connection sleep Frame Body

The message content of the connection sleep Frame Body is specified in Table 34 and Table 35 below.

Table 34 — Connection sleep Frame Body components

Component name	Value	Description
LiCC Version	0x01	Link control command version
LiCC	0x08	Connection sleep
Reserved	0x00	Reserved
Reserved	0x00	Reserved
Own UID		Own UID [Byte 04,Bit 7]: Own UID [63] [Byte 04,Bit 6]: Own UID [62] … [Byte 04,Bit 0]: Own UID [56] [Byte 05,Bit 7]: Own UID [55] … [Byte 11,Bit 0]: Own UID [0]
LiCC Information		Byte 12 to 29 are reserved. Byte 30: Dormant period Byte 31: Awake period

Table 35 — Dormant & awake duration

Parameter name	Value	Description
Dormant period	1 to 200	This field indicates the dormant period during the Local Hibernate sub-state. The unit is 5 ms. 0 and 201 to 255 are reserved.
Awake period	0 to 255	This field indicates the awake period during the Local Hibernate sub-state. The unit is 100 μs. 0 is interpreted as 256.

These parameters are processed at the receiver.

The hibernation should be made in the range of the following conditions.

In actual operation “Awake period” of the Local Hibernate sub-state may be more than the received “Awake period” and the “Dormant period” of the Local Hibernate sub-state may be less than the received “Dormant period”.

10.3.4.5 Connection wake Frame Body

The message content of the connection wake Frame Body is specified in Table 36 below.

Table 36 — Connection wake Frame Body contents

Component name	Value	Description
LiCC Version	0x01	Link control command version
LiCC	0x09	Connection wake
Reserved	0x00	Reserved
Reserved	0x00	Reserved
Own UID		Own UID [Byte 04,Bit 7]: Own UID [63] [Byte 04,Bit 6]: Own UID [62] … [Byte 04,Bit 0]: Own UID [56] [Byte 05,Bit 7]: Own UID [55] … [Byte 11,Bit 0]: Own UID [0]
LiCC Information		Reserved

10.3.4.6 Connection probe Frame Body

The message content of the connection probe Frame Body is specified in Table 37 below.

Table 37 — Connection probe Frame Body contents

Component name	Value	Description
LiCC Version	0x01	Link control command version
LiCC	0x0A	Connection probe
Reserved	0x00	Reserved
Reserved	0x00	Reserved
Own UID		Own UID [Byte 04,Bit 7]: Own UID [63] [Byte 04,Bit 6]: Own UID [62] … [Byte 04,Bit 0]: Own UID [56] [Byte 05,Bit 7]: Own UID [55] … [Byte 11,Bit 0]: Own UID [0]
LiCC Information		Reserved

10.4 CNL function description

10.4.1 Segmenting/Reassembling

When the CSDU is larger than 4096 Bytes, the CNL shall perform segmenting and reassembling. In segmenting, the CNL shall map a single CSDU into multiple Frame Bodies. In this case, each Frame Body contains a CSDU segment. A CSDU segment is only a subset of the original CSDU. The CNL shall send these Frame Bodies using Single Data CPDUs or Multi-Data CPDUs.

The More Segment field shall be set to 1 if another segment of the current CSDU is to follow. It shall be set to 0 in the final segment of a segmented frame or in a non-segmented frame. Segmenting is performed upon transmission.

The process of recombining CSDU segments into CSDUs is called reassembling. The CNL shall perform reassembling before delivering received CSDUs to the CNL User.

The maximum size of the CSDU is out of scope of this specification.

An example of the segmented CSDU is illustrated in Figure 25.

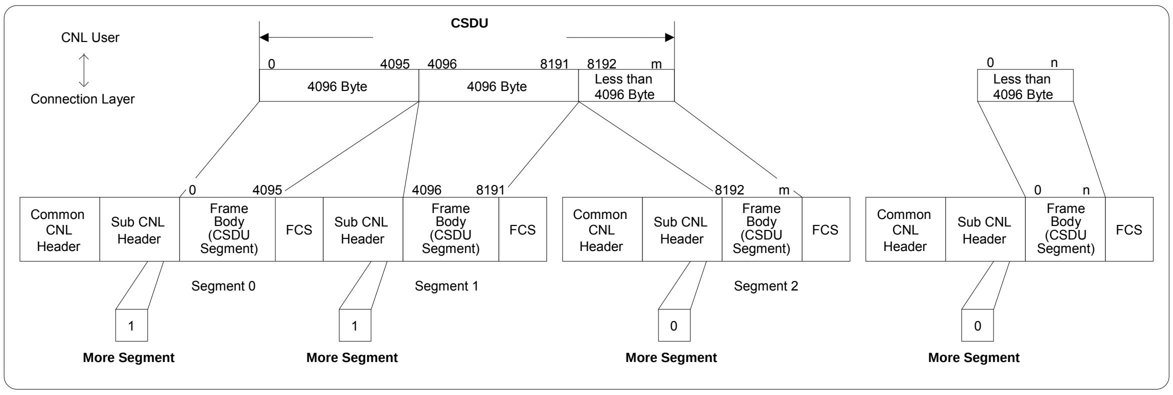


Figure 25 — Segmenting/Reassembling

10.4.2 Medium state sensing

The Sync pattern detection and PHY Header information provided by the PHY layer are used to prevent collisions on the radio medium.

If the Sync pattern and/or PHY Header is detected, then the CNL shall judge the medium state busy for the period of that frame indicated by the PHY Header. Otherwise the CNL shall judge the medium state idle. During the medium busy period, any attempt to transmit a signal shall be held.

10.4.3 CNL-Level acknowledgements

Positive acknowledgement is used. For details, see below.

10.4.3.1 Recovery procedures and retransmit limits

The CNL defines the two acknowledgement type attributes as specified in Table 38 with relation to data frames and management frames. Error recovery and retransmission by the CNL shall be applicable only to those data frames and management frames to which the ImACK attribute is assigned.

Table 38 — Acknowledgement type

ACK Type	Description
ImACK	Requires an ACK as a response.
NoACK	Does not require an ACK as response This type is used only for some of the management frames. Refer to Table 30. This type shall not be applied to a data frame.

The entity shall have at most one unacknowledged CPDU outstanding.

If an error occurs while a data frame or management frame assigned with the ImACK attribute is exchanged, the entity that transmitted the data frame or management frame shall attempt to recover it from the error by retransmitting the frame. Retransmission in the CNL shall be repeated until the current frame is successfully acknowledged or until T_Resend or T_Retry timers expire for data or management frames respectively.

10.4.3.2 ACK procedure

If a CPDU is received and conditions 1 and 2 are satisfied, an ACK response shall be returned.

Condition 1 (Common CNL header part):

- The HCS is correct;
- the RxUID is the same as the receiving entity's Own UID; and
- the TxUID is the same as the Target UID.

Condition 2 (Sub CNL Header part):

- The HCS is correct;
- the ACK type in the attribute field is the ImACK;
- the Frame Body type in the attribute field is Data frame or Management frame; and
- the frame type in the attribute field is Data/Management frame.

ACK SeqNum response for Data frames:

The ACK SeqNum shall be the SeqNum of the last Frame Body with correct FCS of a Data frame which was received in sequence.

The case of a Multi-Data CPDU received in sequence with no error is illustrated in Figure 26.

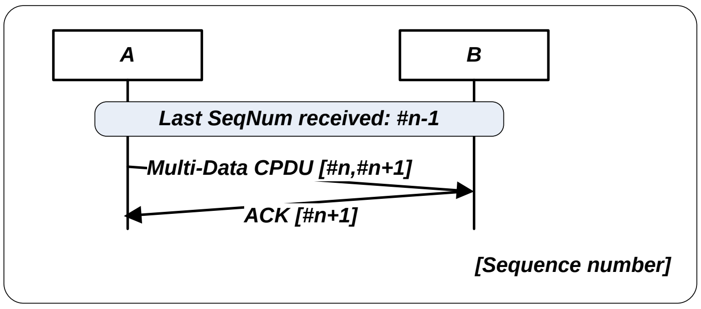


Figure 26 — ACK frame response for a Multi Data CPDU

Figure 27 illustrates several cases of data CPDUs received with incorrect FCS.

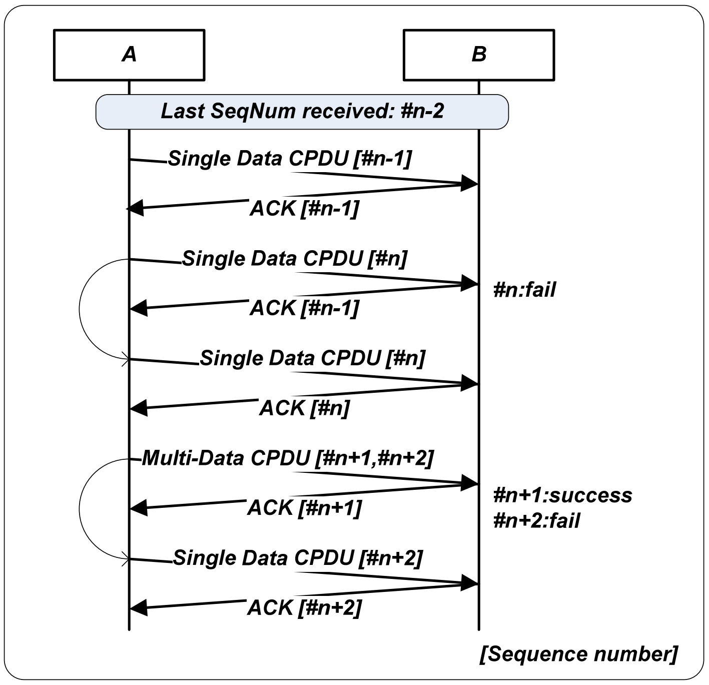


Figure 27 — ACK frame response with FCS error

After the connection establishment, each transmitting entity shall transmit the first data Frame Body with the Sequence Number Synchronization bit of the Sub CNL Header set to 1. If the first data Frame Body with SeqNum #n is received with a correct FCS, then the ACK shall be returned with SeqNum #n. If the first data Frame Body with SeqNum #n has a failed FCS, then the ACK shall be returned with SeqNum #n-1. In both cases, the Sequence Number Synchronization bit of this ACK shall be set to 1.

ACK SeqNum response with FCS error for the Management frame:

If the FCS of the received Management frame is not valid, then the SeqNum of the ACK shall be assigned as the SeqNum of the received Management frame decremented by 1.

10.4.3.3 ACK timeout

The entity that transmitted the data frame or management frame shall judge that the transmission of the frame has failed if it does not detect an ACK within a specified time (ACK Timeout, i.e. SIFS + preamble + sync period).

The transmitting entity shall process the re-transmission when the preamble cannot be detected within ACK timeout from the end of the frame that transmitted the immediate previous frame.

In the case of a data frame, the re-transmission due to ACK Timeout is specified in Figure 28 below.

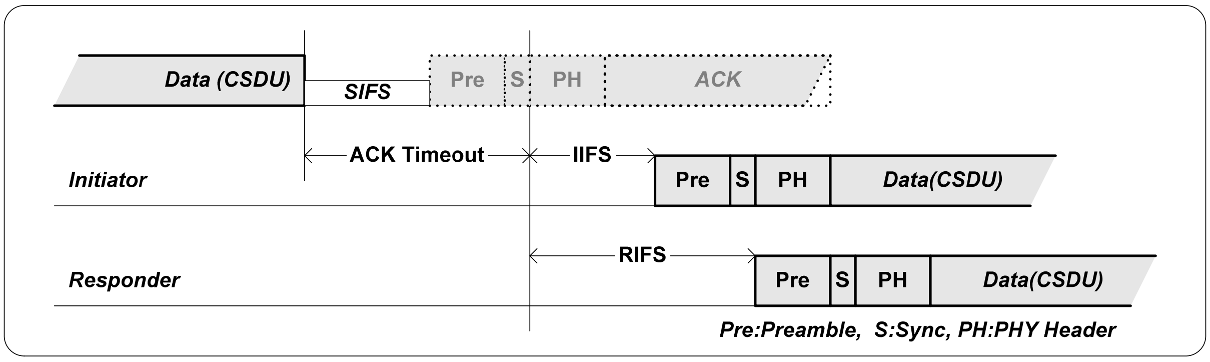


Figure 28 — Data frame ACK timeout & retransmission

In the case of a management frame except C-Acc, the re-transmission due to the ACK timeout is specified in Figure 29 below.

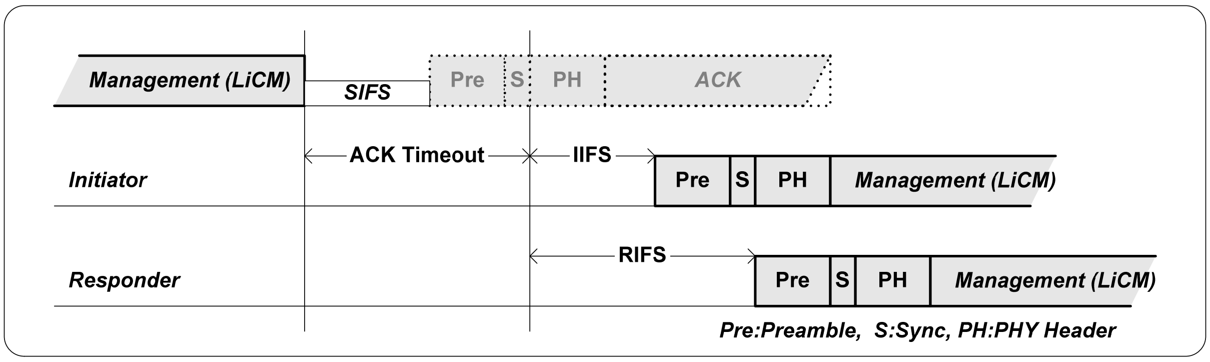


Figure 29 — Management frame (except C-Acc) ACK timeout & retransmission

In the case of C-Acc, the re-transmission due to the ACK timeout is specified in Figure 30 below.

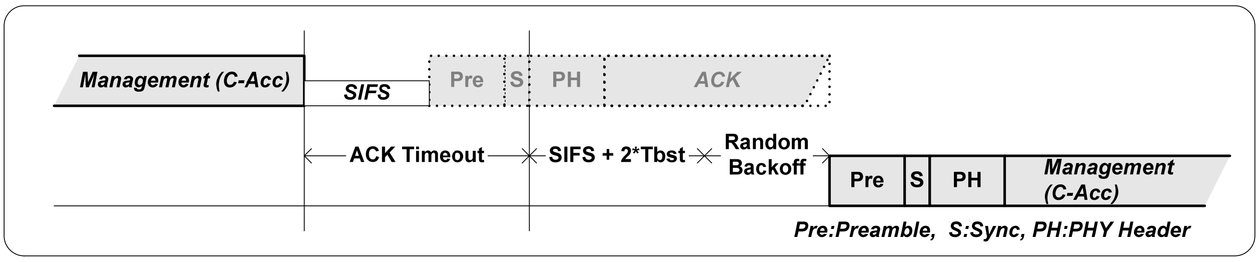


Figure 30 — C-Acc ACK timeout & retransmission

10.4.3.4 Sequence number

The sequence number shall be generated by the transmitting entity.

Sequence numbers are managed independently based on the combination of the Frame Body type and ACK type as specified in Table 39.

Table 39 — Frame Body type, ACK type and SeqNum generation

Frame Body type	ACK type	Allowable number	SeqNum Generation scheme
Data	ImACK	0 to 255	An incrementing sequence of integers. The next value after 255 is 0.
Management	ImACK	0 to 255	The SeqNum shall be different from the SeqNum in the immediate previous frame that was acknowledged.
	NoACK	0 to 255	The SeqNum shall be different from the SeqNum in the immediate previous frame that was transmitted.

The sequence number of the data frame:

The receiving entity shall use SeqNum to detect duplicate or out of sequence frames.

The sequence number of the management frame:

The receiving entity shall use the SeqNum field value to detect duplicate management frames.

Discontinuous detection is not applicable to the management frame, because the management frame is completed with one frame. Also the Sequence Number Synchronization bit shall be set to 0 for management frames.

10.4.3.5 Duplicate CPDU detection and recovery

The CNL re-transmission function may cause the receiving entity to receive duplicate frames. Such duplicate frames shall be detected and discarded by the receiving entity.

Duplicate frames are detected using sequence numbers uniquely assigned by the CNL.

When a frame assigned with the ImACK attribute is received without any errors, the receiving entity shall return an ACK response even if a duplicate frame is detected as illustrated in Figure 31.

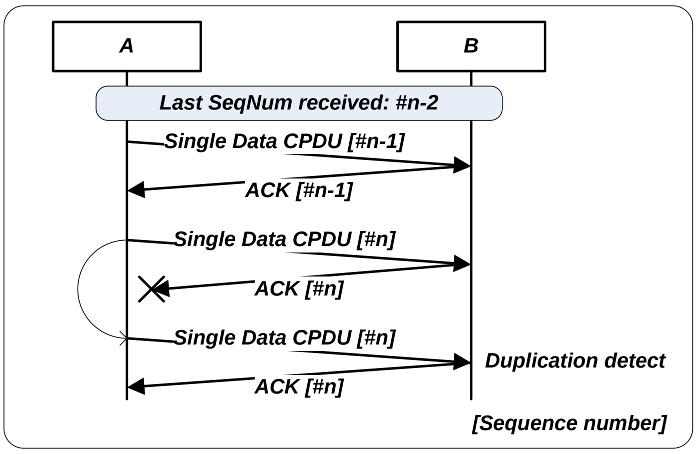


Figure 31 — Duplicate CPDU detection

10.4.3.6 Discontinuous detection

Discontinuous detection is applicable only to data frames.

The receiving entity shall be responsible for detecting the discontinuity of sequence numbers. Frames with discontinuous sequence numbers shall be discarded.

If discontinuous sequence number is detected, the release process shall be performed as specified in 10.5.9.1.

10.4.4 Interframe space (IFS)

The following three types of IFS are specified:

1. SIFS Short interframe space used for an ACK response.
2. IIFS Initiator interframe space used when the initiator transmits a frame.
3. RIFS Responder interframe space used when the responder transmits a frame.

The initiator is an entity that initiates a C-Req. The responder responds to the C-Req from the initiator.

The interframe space parameters are illustrated below in Figure 32 and specified in Table 40.

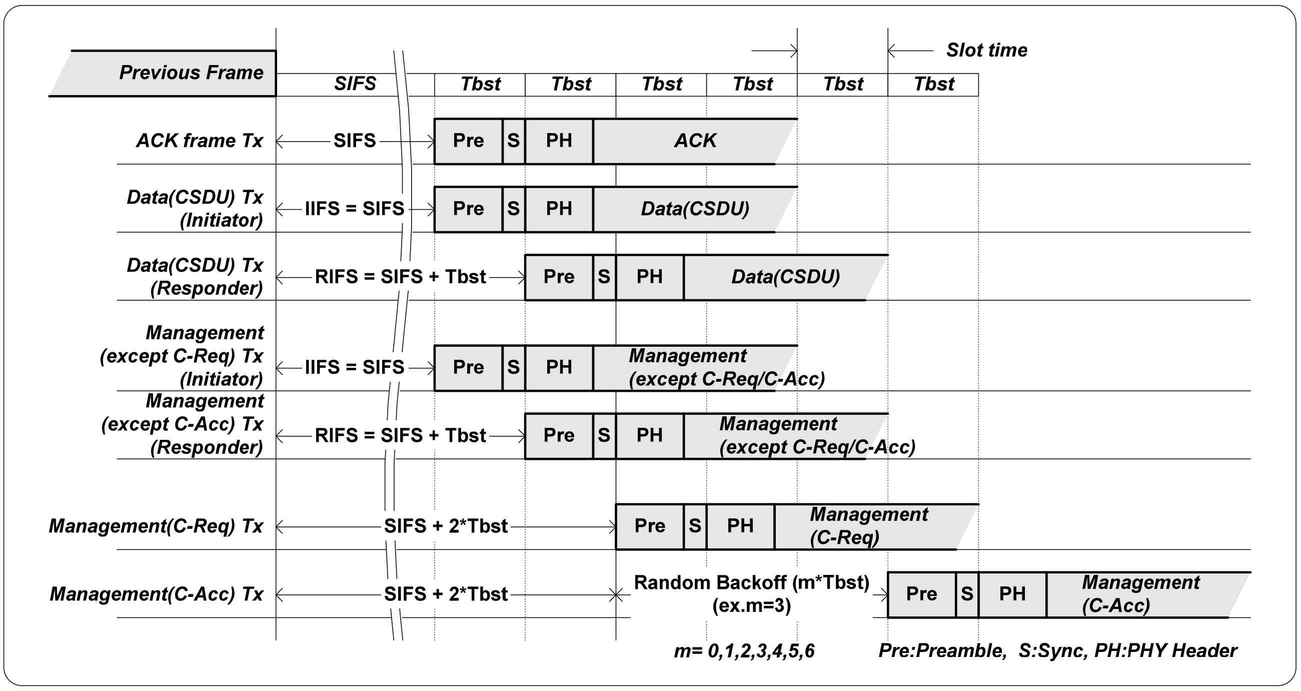

Slot time
Previous Frame
SIFS
Tbst
ACK frame Tx
SIFS
Pre
S
PH
ACK
Data(CSDU) Tx (Initiator)
IIFS = SIFS
Data(CSDU)
Data(CSDU) Tx (Responder)
RIFS = SIFS + Tbst
Management (except C-Req) Tx (Initiator)
Management (except C-Req/C-Acc)
Management (except C-Acc) Tx (Responder)
Management(C-Req) Tx
SIFS + 2*Tbst
Management (C-Req)
Management(C-Acc) Tx
Random Backoff (m*Tbst) (ex.m=3)
Management (C-Acc)
m= 0,1,2,3,4,5,6
Pre:Preamble, S:Sync, PH:PHY Header

Figure 32 — Interframe space example

Table 40 — Interframe space

Parameter name	Value	Description
SIFS	min: 8,5 / max: 11,5 us	Short interframe space
IIFS	min: 8,5 us IIFS shall be greater or equal to SIFS	Initiator minimum interframe space
RIFS	min: 15,7 us RIFS shall be greater or equal to SIFS+Tbst	Responder minimum interframe space
Tbst	8 us ± 10%	Slot time

An entity shall perform medium state sensing as specified in 10.4.2 to check if the state of the medium is idle before sending any frame.

10.4.4.1 Short interframe space (SIFS)

If a data frame or management frame assigned with ImACK attribute is received successfully, an ACK shall be sent only after sensing that the state of the medium remains idle for at least one SIFS period.

10.4.4.2 Initiator interframe space (IIFS)

The initiator shall not transmit a frame until sensing that the state of the medium remains idle for at least one IIFS period.

10.4.4.3 Responder interframe space (RIFS)

The responder shall not transmit a frame until sensing that the state of the medium remains idle for at least one RIFS period.

10.4.4.4 Random backoff time

An entity shall use the random backoff time specified as follows when transmitting the C-Acc.

Random backoff time = Random() $\times$ Slot time

where Random() is an integer between 0 and 6.

An entity shall not transmit C-Acc until sensing that the state of the medium remains idle for at least one SIFS period + 2 Slot times + one random backoff time.

10.4.5 Access procedure

10.4.5.1 Basic access condition

A entity shall transmits a frame after it detects that the state of the medium remains idle for a specified duration or longer.

10.4.5.2 Backoff procedure

The backoff procedure shall be performed if the state of the medium is found to be busy immediately before a frame is transmitted as specified in Figure 33.

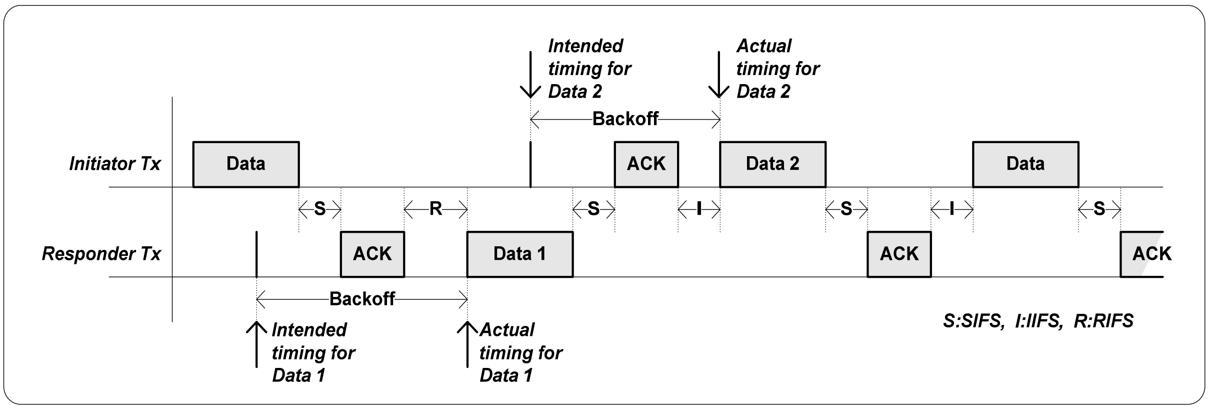


Figure 33 — Backoff procedure

10.4.5.3 Connection request transmission

The C-Req is a management frame indicating a connection request. The entity requesting a link connection shall transmit this frame at regular intervals unless the state of the radio medium is busy or until it receives the C-Acc.

The C-Req transmission shall be stopped when C-Acc is received as illustrated in Figure 34 and specified in Table 41.

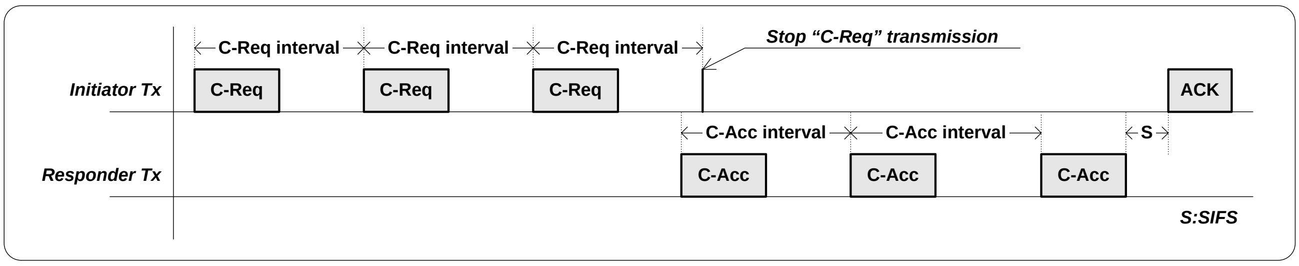


Figure 34 — C-Req transmission

Table 41 — C-Req interval and C-Acc interval

Parameter name	Value	Description
C-Req interval	100 [us] ±10%	C-Req frame interval
C-Acc interval	ACK Timeout + SIFS + 2*Tbst + Random Backoff Refer to Figure 30 and Figure 32 for more detail.	C-Acc frame interval

10.4.6 Multirate support

Each frame shall be sent using the rate settings specified in Table 42.

Table 42 — Frame type/Frame Body type vs. transfer rate

Frame type	Frame Body type	Required transfer rate setting
Acknowledgement	n/a	Rate 32
Data/Management	Data	Rate 522 to Rate 32
	Management	Rate 32

10.4.7 UID filter

Received frames shall be filtered using the Rx UID and Tx UID fields of the common CNL header. The CNL sends Indications when it receives frames depending on the CNL states and the TX/RX UID field values as specified in Table 43. The CNL Indication is Conditional in some situations. In that case, the CNL filters the UID (meaning it does not send the Indication) if the Tx UID does not meet the filter conditions. The details of the conditions are up to the implementation and are thus out of the scope of this Standard.

Table 43 — UID filtering

CNL State	Common CNL Header Rx UID field value	Common CNL Header Tx UID field value	CNL Indication to CNL User (Conditional or Unconditional)
Search, Connection Request, Response Waiting, Accept Waiting	Own UID Paging UID	Any UID (except Paging UID)	Conditional It is possible to give a limitation to Tx UID field value to restrict the target entity.
Responder Response, Initiator Connected, Responder Connected	Own UID	Target UID	Unconditional

10.5 CNL state

The CNL uses the states specified in Table 44 to manage the connection.

Table 44 — CNL states and corresponding roles

State Name	Entity Role
Close	None
Search	Responder
Connection Request	Initiator
Accept Waiting	Responder
Response Waiting	Initiator
Responder Response	Responder
Initiator Connected	Initiator
Responder Connected	Responder

The exclusivity of the CNL_ACCEPT request and CNL_ACCEPT response is arbitrated by the CNL User.

10.5.1 Close state

In this state, the CNL is not being activated.

When CNL_CLOSE.request is received from the CNL User, the CNL entity shall change to this state.

When a CNL_INIT.request is received from the CNL User, the CNL entity shall change to the Search state.

10.5.2 Search state

This state is for waiting for the C-Req.

When CNL_CONNECT.request is received from the CNL User, the CNL entity shall change to the Connection Request state.

When C-Req is received, the CNL entity shall change to the Accept Waiting state after sending a connect indication to the CNL User.

When CNL_CLOSE.request is received from the CNL User, the CNL entity shall change to the Close State.

In the Search state, the hibernation procedure may be performed to reduce power consumption.

The hibernation period is composed of a dormant period (Tds) during which it is not possible to receive any frame and an awake period (Tas) during which it is possible to receive the C-Req frame as specified in Figure 35. The Awake period shall be set in consideration of the C-Req interval tolerance and the C-Req frame duration. See Table 47 for the required limits on Tds and Tas.

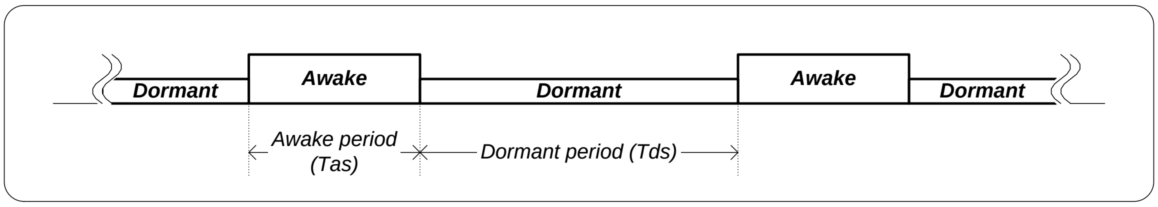


Figure 35 — Dormant & awake period in Search state

10.5.3 Connection request state

This state is for requesting a link establishment. In this state, the entity shall transmit the C-Req and wait for the C-Acc.

When C-Acc is received from the responder candidate, the CNL entity shall change to the Response Waiting state after sending an accept indication to the CNL User.

When C-Req is received in this state, a connect indication shall be sent to the CNL User. After the indication, if CNL_ACCEPT.request is received from the CNL User, the CNL entity shall change to the Responder Response state after stopping the T_Connect timer. Or, after the indication, if CNL_RELEASE.request is received from the CNL User, the CNL shall stop the T_Connect timer, then transmit a C-Rls, and then the CNL entity shall change to the Search state.

When the T_Connect timer expires, the CNL entity shall change to the Search state after indicating a released connection to the CNL User.

- o The T_Connect timer is used for watching the duration of C-Req transmission. This timer shall start on the attempt to transmit the first C-Req and shall stop when C-Acc is received.

When a C-Rls is received, the CNL shall stop the T_Connect timer, then indicate a released connection to the CNL User, and then the CNL entity shall change to the Search state.

10.5.4 Accept waiting state

This state is for waiting for the CNL User response, after receiving the C-Req from the initiator candidate.

If the CNL User permits to establish the connection with the initiator candidate (as indicated by a CNL_ACCEPT.request), the CNL entity shall change to the Responder Response state.

If the CNL User does not permit to establish the connection with the initiator candidate (as indicated by CNL_RELEASE.request), the CNL entity shall change to the Search state after transmitting the C-Rls.

10.5.5 Response waiting state

This state is for waiting for the CNL User accept response, after receiving the C-Acc from the responder candidate.

If the CNL User permits to establish the connection with the responder candidate (as indicated by a CNL_ACCEPT.response), the CNL entity shall change to the Initiator Connected state, then start the T_KeepAlive timer from zero, and send an ACK to respond to the corresponding C-Acc.

If the CNL User does not permit to establish the connection with the responder candidate (as indicated by CNL_RELEASE.request), the CNL entity shall change to the Search state after transmitting the C-Rls.

When the C-Rls from the responder candidate is received, a release indication shall be sent to the CNL User, and then the CNL entity shall change to the Search state.

10.5.6 Responder response state

This state is for accepting a link connection. In this state, the entity shall transmit the C-Acc and wait for an ACK response.

When an ACK for the C-Acc is received, the CNL shall start the T_KeepAlive timer from zero, then send a confirmation to the CNL User, and then the CNL entity shall change to the Responder Connected state.

When the C-Rls is received or T_Accept timer expires, the CNL entity shall change to the Search state after sending a release indication to the CNL User.

- o The T_Accept timer is used for watching the duration of a C-Acc transmission. This timer shall start on the attempt to transmit the first C-Acc and shall stop when the ACK for C-Acc is received.

If a data CPDU or a management CPDU(C-Sleep,C-Wake,C-Probe) is received after an ACK timeout of the C-Acc, the CNL entity may send an ACK to respond to the corresponding CPDU.

10.5.7 Initiator connected state

When the CNL_RELEASE.request is received from the CNL User, the CNL entity shall change to the Search state after transmitting the C-Rls.

When the C-Rls is received or the T_Retry timer expires, the CNL entity shall change to the Search state after sending a connection release indication to the CNL User.

- The T_Retry timer is used for watching the duration of management frame transmissions except C-Req, C-Acc and C-Rls. This timer shall start on the attempt to transmit the first management frame and shall stop when the ACK is received.

When the T_KeepAlive timer expires, the CNL shall start to transmit the C-Probe. If the ACK is received for the C-Probe, the CNL shall restart the T_KeepAlive timer from zero.

If C-Probe is received, the CNL shall stop the T_KeepAlive timer, then send an ACK for the C-Probe, and then restart the T_KeepAlive timer from zero.

If C-Acc is received, the CNL shall stop the T_KeepAlive timer, then send an ACK for the C-Acc, and then restart the T_KeepAlive timer from zero.

10.5.8 Responder connected state

When CNL_RELEASE.request is received from the CNL User, the CNL entity shall change to Search state after transmitting the C-Rls.

When the C-Rls is received or the T_Retry timer expires, the CNL entity shall change to the Search state after sending a connection release indication to the CNL User.

- The T_Retry timer is used for watching the duration of management frame transmissions except C-Req, C-Acc and C-Rls. This timer shall start on the attempt to transmit the first management frame and shall stop when the ACK is received.

When the T_KeepAlive timer expires, the CNL shall start to transmit the C-Probe. If the ACK is received for the C-Probe, the CNL shall restart the T_KeepAlive timer from zero.

If C-Probe is received, the CNL shall stop the T_KeepAlive timer, then send an ACK for the C-Probe, and then restart the T_KeepAlive timer from zero.

10.5.9 Sub-states within the Initiator connected state or Responder connected state

The connected state has the following sub-states.

A. Connected Sub-State

B. Local Hibernate Sub-State

C. Target Sleep Sub-State

On entry to the Initiator connected state, the initiator shall enter the connected sub-state. On entry to the Responder connected state, the responder shall enter the connected sub-state. In case of the initiator, the initiator's Local Hibernate sub-state shows the hibernation of the initiator, and the Target Sleep sub-state shows the hibernation of the responder.

In case of the responder, the responder's Local Hibernate sub-state shows the hibernation of the responder, and the Target Sleep sub-state shows the hibernation of the initiator.

10.5.9.1 Connected sub-state

Data frames shall be exchanged in this sub-state.

When CNL_DATA.request is received from the CNL User, the CNL shall transmit data frames after stopping the T_KeepAlive timer. After the ACK is received for each CPDU, the CNL shall start the T_KeepAlive timer from zero.
When a data CPDU with a More Segment field = 0 is received from the target, the CNL shall stop the T_KeepAlive timer, then send an ACK to the target, and then send a data indication to the CNL User. After sending the data indication, the CNL shall start the T_KeepAlive timer from zero.
When a data CPDU with all More Segment fields = 1 is received from the target, the CNL shall send an ACK to the target after stopping and starting the T_KeepAlive timer from zero.

When CNL_POWERSAVE.request is received from the CNL User, the CNL entity shall perform the following steps in sequence:

1. Transmit the C-Sleep and wait to receive an Ack.
2. After receiving the Ack, confirm the CNL_POWERSAVE service to the CNL User and then change to the Local Hibernate sub-state if the CNL entity supports the Power Save service.

When the C-Sleep is received from the other entity, the CNL entity shall perform the following steps in sequence:

1. Stop the T_KeepAlive timer,
2. send an ACK,
3. send an indication to the CNL User (with a CNL_POWERSAVE.Indication for example),
4. start the T_KeepAliver timer from zero, and
5. change to the Target Sleep sub-state.

When T_Resend timer expires, the CNL entity shall change to the Target Sleep sub-state and start to transmit the C-Wake.

o The T_Resend timer is used for watching the duration of data frame transmissions. This timer shall start on the first attempt to transmit each data frame and shall stop when the ACK is received.

When the T_Retry timer expires in this sub-state, the CNL entity shall change to the Search state after sending a connection release indication to the CNL User.

If a discontinuous sequence number is detected, or a repeated Sequence Number Synchronization bit is detected,the CNL entity shall transmit the C-RIs followed by changing to the Search state. See 10.4.3.6 for more details on discontinuous sequence number detection. A repeated Sequence Number Synchronization bit means that a data Frame Body, except the first data Frame Body, has the Sequence Number Synchronization bit of the Sub CNL Header set to 1.

10.5.9.2 Local hibernate sub-state

This sub-state is for performing hibernation while being connected.

When the C-Wake is received from the target, the CNL entity shall send an indication to the CNL User, then transmit an ACK, then start the T_KeepAlive timer from zero, and then change to the Connected sub-state.

When CNL_DATA.request or CNL_WAKE.request is received, the CNL entity shall change to the Target Sleep sub-state and shall start to transmit a C-Wake. When the T_Retry timer expires in this sub-state, the CNL entity shall change to the Search state.

The Local Hibernate Sub-state consists of dormant and awake periods as specified in Figure 36.

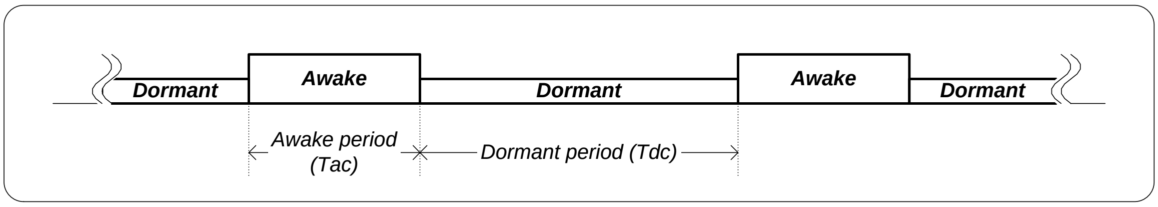


Figure 36 — Dormant & awake period in Local Hibernate sub-state

Refer to 10.3.4.4 for handling of awake period and dormant period in the hibernation.

The hibernation period is comprised of a dormant period (Tdc) during which it is not possible to receive any frame and an awake period (Tac) during which it is possible to receive the C-Wake and C-Probe frame.

See Table 47 for the required limits on Tdc and Tac.

10.5.9.3 Target sleep sub-state

This state is used when it is assumed that the target entity is hibernating and the local entity is not hibernating.

When a C-Sleep is received from the target, the CNL shall stop the T_KeepAlive timer, then send an ACK, then send an indication to the CNL User, and then start the T_KeepAlive timer from zero.

When CNL_POWERSAVE.request is received from the CNL User, the CNL entity shall change to the entity's Local Hibernate sub-state if the CNL supports the Power Save service.

When the T_Retry timer expires in this sub-state, the CNL entity shall change to the Search state after sending a connection release indication to the CNL User.

When CNL_WAKE.request is received from the CNL User, the CNL shall start to transmit the C-Wake to the target.

If an ACK for the C-Wake is received, the CNL shall send a wake confirmation to the CNL User, then start the T_KeepAlive timer from zero, and then the CNL entity shall change to the Connected sub-state.

When CNL_DATA.request is received from the CNL User, the CNL shall stop the T_KeepAlive timer, and then start to transmit the C-Wake. If an ACK for the C-Wake is received, the CNL shall send a wake confirmation to the CNL User, then start the T_KeepAlive timer from zero, and then the CNL entity shall change to the Connected sub-state and start to transmit the data.

When the C-Wake is received from the target, the CNL shall send an ACK response, then send an indication to the CNL User, then start the T_KeepAlive timer from zero, and then the CNL entity shall change to the Connected sub-state.

10.5.9.4 Permitted frame types for each state

The following Table 45 and Table 46 specify which frame types are permitted to be transmitted from within each CNL entity state.

"X" in the table specifies that transmission of that frame type is permitted. A blank indicates that transmission of that frame type is not permitted.

Table 45 — Permitted frame transmissions from within each state

	C-Req	C-Acc	C-Rls	C-Wake	C-Sleep	C-Probe	ACK	CSDU
Close								
Search								
Connection Request	X		X					
Accept Waiting			X					
Response Waiting			X				X	
Responder Response		X					X	

Table 46 — Permitted frame transmissions from within each sub-state

	C-Req	C-Acc	C-Rls	C-Wake	C-Sleep	C-Probe	ACK	CSDU
Connected Sub-State			X		X	X	X	X
Local Hibernate Sub-State						X	X	
Target Sleep Sub-State				X		X	X	

10.6 Numerical parameters

Numerical parameters are specified in Table 47.

Table 47 — Numeric parameters

Parameter name	Min.	Max
T_Connect		10 s
T_Accept	500 ms	10 s
T_Retry	1,25 s	10 s
T_Resend		200 ms
Tas	140 us	
Tds		1 s
Tac	100 us	
Tdc		1 s

Annex A
(normative)

UID Specification

A.1 UID Composition

The 64-bit UID is specified in Figure A.1 (with the exception of the Paging UID).

Specifier ID 20 bits	Reserved 16 bits	Extension Identifier 28 bits

Figure A.1 — Composition of UID

Each implemented entity unit shall be assigned a unique UID value.

A.1.1 Specifier ID

The Specifier ID identifies the organization that is responsible for the individual implemented entity. All Specifier IDs shall be assigned using the Procedure at http://www.ecma-international.org/publications/standards/Ecma-399.htm.

A.1.2 Reserved

This component shall be set to 0x0000.

A.1.3 Extension Identifier

Usage of this component is up to each Specifier, including the allocation methods and policies. The combination of Specifier ID and Extension Identifier shall be unique for each implemented entity unit.

Annex B
(informative)

Coupler

The antenna used to achieve close proximity electric induction wireless communications is called a “Coupler”. The coupler is optimized to transmit and receive the longitudinal wave ER. The transverse wave should be suppressed. Therefore the coupler is not compatible with a normal microwave transceiver antenna.

The measurement method and reference coupler drawings are provided in Annex C and Annex D.

Annex C
(informative)

Coupler measurement

The following considerations should be observed when measuring a coupler.

- Use a reference coupler in measurement.
- Let CUT (coupler under test) face a reference coupler.
- Set z-axis along the line between the centres of the two couplers, then set x/y-axis vertical to z-axis.
- Fix the position of CUT and define the nearest position of reference coupler from CUT as the origin (0,0,0).
- Connect CUT and reference coupler to network analyzer.
- Measure S_{21} between the two couplers when the reference coupler is at the position as follows.
- Measurement should be done using 50 ohm impedance.

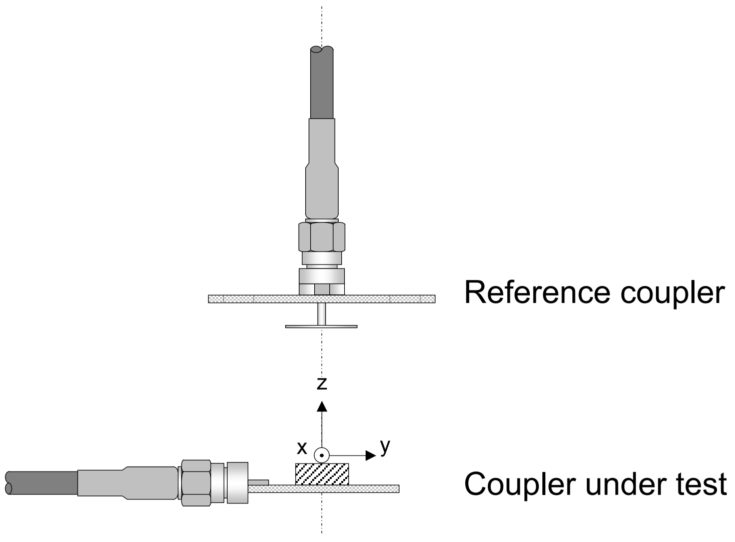


Figure C.1 — Coupler measurement setup

The measurement setup is illustrated in Figure C.1. The recommended coupler characteristics are illustrated in Table C.1.

Table C.1 — Recommended coupler characteristics

Coupling Conditions	
Reference Coupler (x,y,z) Positions	(0,0,15), (10,0,10), (-10,0,10), (0,10,10), (0,-10,10) mm
Coupling Strength	S_{21} (4,48 GHz) > -25 dB
Flatness	Maximum S_{21} – Minimum S_{21} from 4,20 GHz to 4,76 GHz < 3 dB
Un-coupling Conditions	
Position of Reference Coupler (x,y,z)	(0,0,100) mm
Coupling Strength	S_{21} (4,48 GHz) < -40 dB

Annex D
(informative)

Reference Coupler

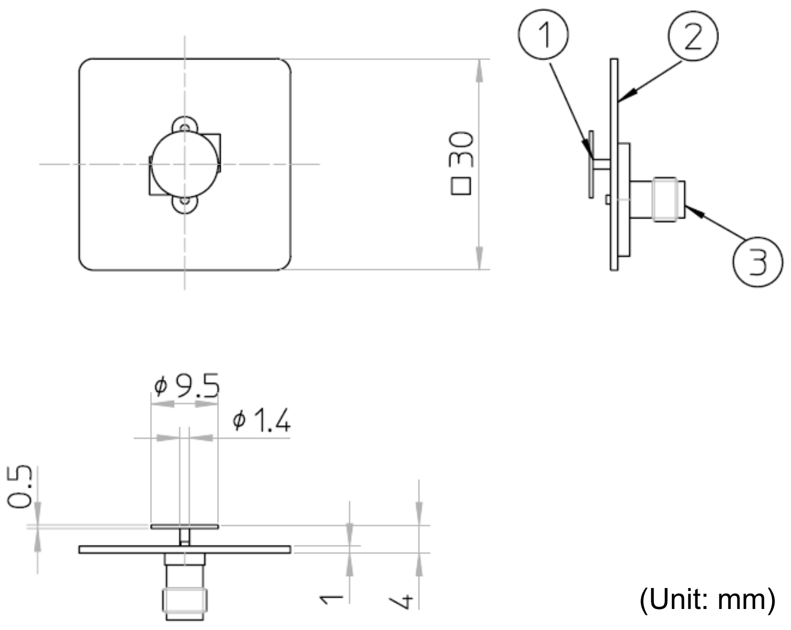
The following items comprise the reference coupler.

1: Coupling Electrode (brass)

2: Printed Wiring Board (PPE: Polyphenyleneether, ε=3.3)

3: Coaxial Connector

Figure D.1 specifies the structure of a reference coupler.



(Unit: mm)

Figure D.1 — Reference coupler

The pattern for the corresponding printed wiring board is specified in Figure D.2.

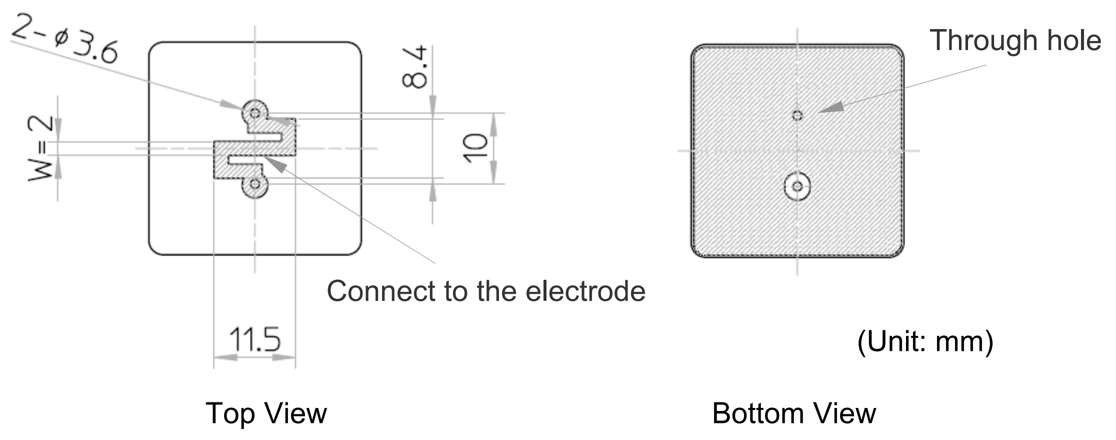


Figure D.2 — Printed wiring board pattern

Table D.1 illustrates the typical characteristics based on the measurement method described in Annex C.

Table D.1 — Typical coupler characteristics

Coupling Condition	
Positions of Reference Coupler (x,y,z)	(0,0,15), (10,0,10), (-10,0,10), (0,10,10), (0,-10,10) mm
Coupling Strength	S_{21} (4,48 GHz) = -17 dB
Flatness	Maximum S_{21} – Minimum S_{21} from 4,20 GHz to 4,76 GHz = 1 dB
Un-coupling Condition	
Position of Reference Coupler (x,y,z)	(0,0,100) mm
Coupling Strength	S_{21} (4,48GHz) = -45 dB

Annex E
(informative)

Sample Data Sequences

E.1 Reed-Solomon Encoder

Table E.1 and Table E.2 illustrate sample values calculated by the Reed-Solomon Encoder. In Table E.1, the input data is a simple counting sequence starting with Byte 00 = 0x00, incremented by one for each subsequent byte, and ending with Byte 223 = 0xDF. In both Table E.1 and Table E.2, the RS Parity Bytes are listed from left to right from higher order to lower order such that the leftmost byte is the highest order byte from the Reed-Solomon encoder and the first parity byte transferred to the next stage of processing.

Table E.1 — Reed-Solomon Encoder Sample data for 224 input bytes

Input Data 224 bytes	0x00 01 02 03 04 05 06 07 08, …, DA DB DC DD DE DF (Increment data)
RS Parity 16 bytes	0xA1 5D 0E E4 0B 5F 8B AE E4 68 87 AA 1B 97 11 5B

Table E.2 — Reed-Solomon Encoder Sample data for 16 input bytes

Input Data 16 bytes	0x00 01 02 03 04 05 06 07 08 09 0A 0B 78 CC CA DC
RS Parity 16 bytes	0xCF C3 47 06 36 82 7B DA FA 47 4E 5C 3E 8F F4 10

E.2 Convolutional Encoder

Table E.3, Table E.4 and Table E.5 illustrate sample data for the Convolutional encoder.

Table E.3 — Convolutional Encoder Sample data for Rate 261

t (T_b)	Input Data		Output Data	
0	D0	1	aG00	1
			aG10	1
1	D1	0	bG00	0
			bG10	0
2	D2	1	aG01	0
			aG11	1
3	D3	1	bG01	1
			bG11	1
4	D4	0	aG02	0
			aG12	1
5	D5	0	bG02	1
			bG12	0
6	D6	1	aG03	0
			aG13	0
7	D7	1	bG03	0
			bG13	0
…	…	…	…	…

Table E.4 — Convolutional Encoder Sample data for Rate 130 to Rate 32 and Phy Header

t (T_b)	Input Data		Output Data	
0	D0	1	G00	1
			G10	1
1	D1	0	G01	1
			G11	0
2	D2	1	G02	0
			G12	0
3	D3	1	G03	0
			G13	1
4	D4	0	G04	0
			G14	1
5	D5	0	G05	1
			G15	1
6	D6	1	G06	1
			G16	1
7	D7	1	G07	0
			G17	1
…	…	…	…	…

Table E.5 — Convolutional Encoder Sample data for Phy Header

Input Data(hex)	{Ver=0x1, Rate=0x2, Reserved=0x00, Length=0x00 52, HCS=0xB5 22, Tail bits=0x0} = { 0x12 00 00 52 B5 22 0}
Output Data(hex)	0x03 BE C0 00 00 00 38 BE 21 48 BE CE C0

E.3 PHY Header HCS

Table E.6 illustrates sample values for the 16-bit ECS used for the PHY Header HCS.

Table E.6 — PHY Header HCS Sample data

Field Name	Ver[3:0]	Rate[3:0]	Reserved[7:0]	Length[15:0]	HCS[15:0]
Value (hex)	0x1	0x1	0x00	0x0052	0x2003
Value (binary)	0001	0001	0000 0000	0000 0000 0101 0010	0010 0000 0000 0011

E.4 Common CNL Header HCS

Table E.7 illustrates sample values for the 32-bit ECS used for the Common CNL Header HCS.

Table E.7 — Common CNL Header HCS Sample data

Field Name	Rx UID[63:0]	Tx UID[63:0]	Rsv[7:0]	MUX[7:0]	HCS[31:0]
Value (hex)	0x00 01 02 03 04 05 06 07	0x08 09 0A 0B 0C 0D 0E 0F	0x00	0x01	0x4F 82 7F 74

E.5 Sub CNL Header HCS

Table E.8 illustrates sample values for the 32-bit ECS used for the Common CNL Header HCS.

Table E.8 — Sub CNL Header HCS Sample data

Field Name	Attribute[7:0]	SeqNum[7:0]	Length[15:0]	HCS[31:0]
Value (hex)	0x61	0x01	0x00 20	0x7A FA 8D 67

NOTE: Attribute[7:0] = {Reserved, ACK Type, Frame Body type, Reserved, Frame type} = {0,1,1,000,01} (binary).

E.6 Scrambling sequence generator

Table E.9 illustrates sample values for the Scrambling sequence generator.

Table E.9 — Scrambling Sequence Generator Sample data

chip index	Seed[17:0] = [011A0]		Seed[17:0] = [27BFA]		Seed[17:0] = [3C859]	
	chip	hex	chip	hex	chip	hex
0	0	0	1	9	1	F
1	0		0		1	
2	0		0		1	
3	0		1		1	
4	0	4	1	E	0	2
5	1		1		0	
6	0		1		1	
7	0		0		0	
8	0	6	1	F	0	1
9	1		1		0	
10	1		1		0	
11	0		1		1	
12	1	8	1	E	0	6
13	0		1		1	
14	0		1		1	
15	0		0		0	
16	0	0	1	9	0	6
17	0		0		1	
18	0		0		1	
19	0		1		0	
20	1	B	0	1	0	6
21	0		0		1	
22	1		0		1	
23	1		1		0	
24	0	5	1	B	1	E
25	1		0		1	
26	0		1		1	
27	1		1		0	
28	0	4	0	5	0	6
29	1		1		1	
30	0		0		1	
31	0		1		0	
32	1	D	0	0	1	D
33	1		0		1	
34	0		0		0	
35	1		0		1	
36	1	8	1	B	1	B
37	0		0		0	
38	0		1		1	
39	0		1		1	
40	1	9	0	6	0	6
41	0		1		1	
42	0		1		1	
43	1		0		0	
44	0	6	0	2	0	2
45	1		0		0	
46	1		1		1	
47	0		0		0	
48	1	8	0	4	1	B
49	0		1		0	
50	0		0		1	
51	0		0		1	
52	1	A	1	C	1	E

Table E.9 — Scrambling Sequence Generator Sample data (concluded)

chip index	Seed[17:0] = [011A0]		Seed[17:0] = [27BFA]		Seed[17:0] = [3C859]	
	chip	hex	chip	hex	chip	hex
53	0		1		1	
54	1		0		1	
55	0		0		0	
56	1	C	1	B	1	F
57	1		0		1	
58	0		1		1	
59	0		1		1	
60	0	4	0	7	1	D
61	1		1		1	
62	0		1		0	
63	0		1		1	
64	1	B	0	6	0	0
65	0		1		0	
66	1		1		0	
67	1		0		0	
68	1	F	1	B	0	2
69	1		0		0	
70	1		1		1	
71	1		1		0	
72	0	1	0	7	1	E
73	0		1		1	
74	0		1		1	
75	1		1		0	
76	1	8	1	A	1	D
77	0		0		1	
78	0		1		0	
79	0		0		1	

Annex F
(informative)

CNL frame exchange sequences

F.1CNL frame exchange sequences

F.1.1 Connection setup procedure

Entities set up a link connection by exchanging their respective UIDs by means of the C-Req and C-Acc. The setup procedure begins when the entity (A) requesting a connection transmits the C-Req. The procedure completes when the peer (B) that has received the C-Req from peer A accepts the connection and responds with C-Acc as illustrated in Figure F.1.

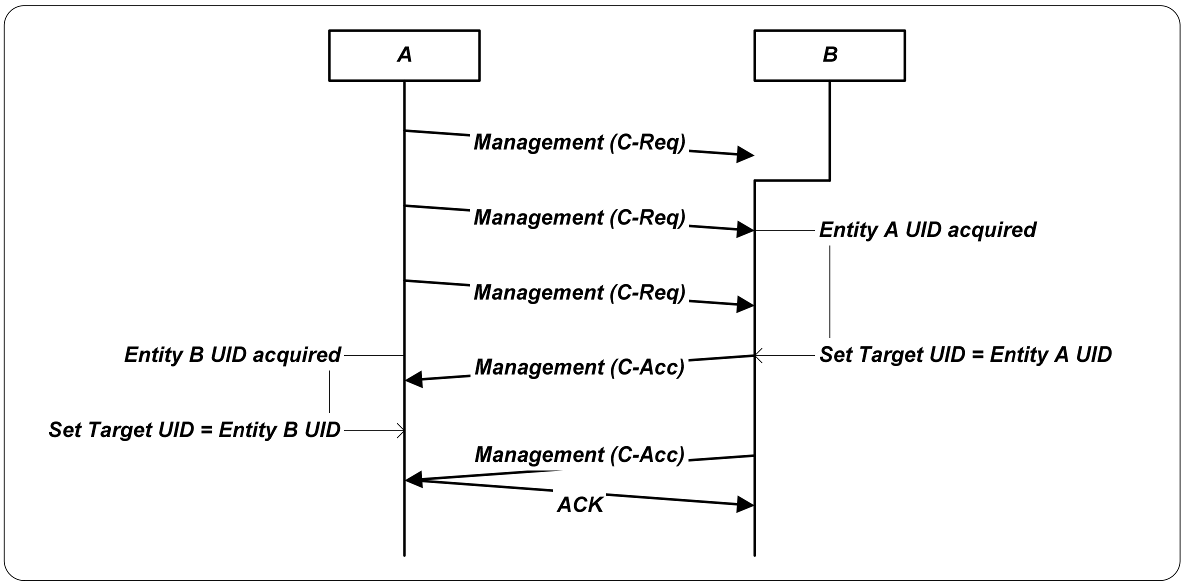


Figure F.1 — Connection setup procedure

F.1.2 CSDU exchange procedure

CSDUs are transmitted using Data Frame CPDUs. The entity that successfully receives the data frame returns an ACK response as illustrated in Figure F.2.

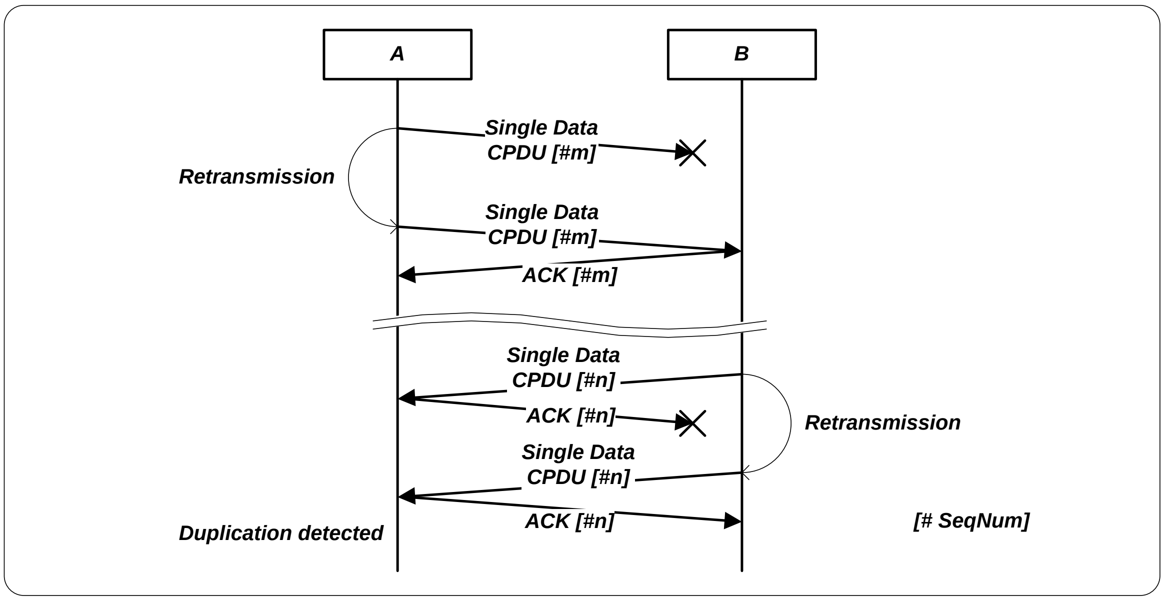


Figure F.2 — CSDU exchange procedure

F.1.3 Connection sleep procedure

The entity transmits the C-Sleep when it hibernates or when it permits the other entity to go into hibernation. The performing of the hibernation is optional for both entities.

Figure F.3 illustrates the case when both entities perform the hibernation. There can be three more cases.

1. Only the Entity A performs hibernation.

2. Only the Entity B performs hibernation.

3. Both Entities do not perform the hibernation.

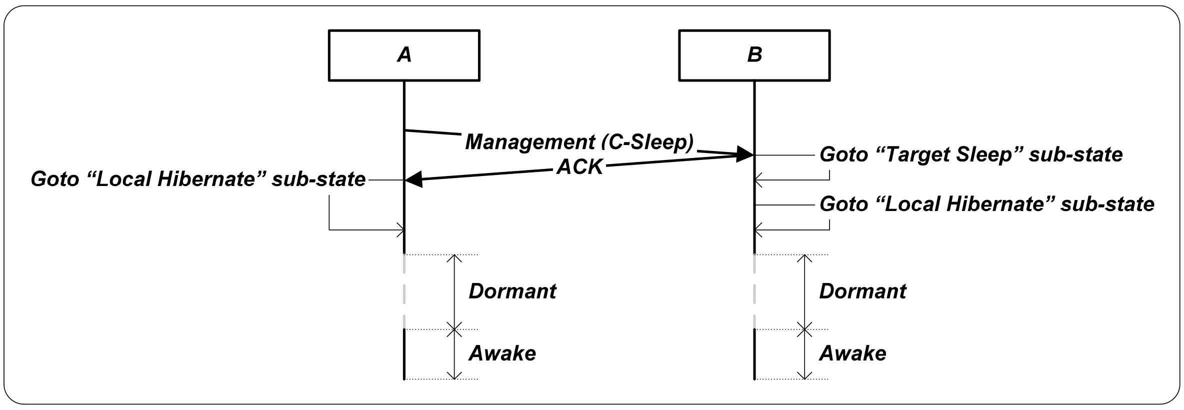

A
B
Management (C-Sleep)
ACK
Goto "Target Sleep" sub-state
Goto "Local Hibernate" sub-state
Goto "Local Hibernate" sub-state
Dormant
Awake
Dormant
Awake

Figure F.3 — Connection sleep procedure (both entities perform the hibernation)

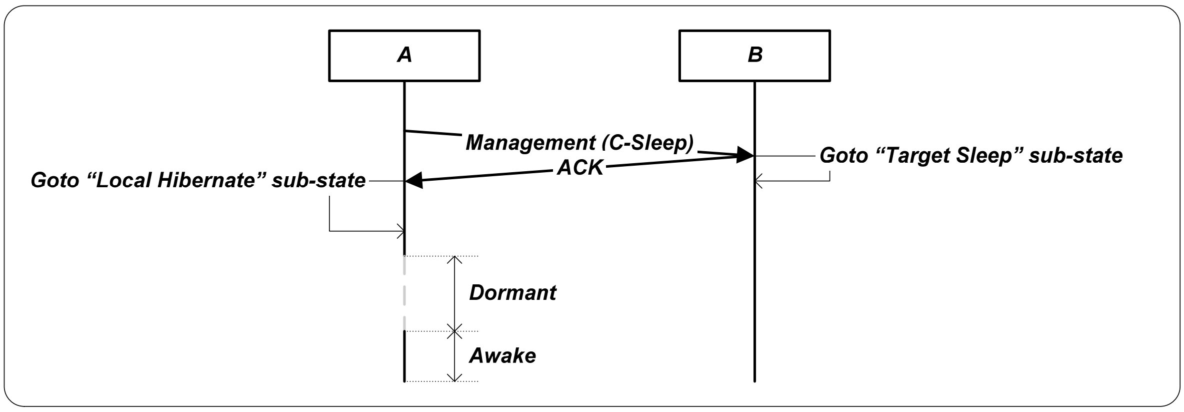

A
B
Management (C-Sleep)
ACK
Goto "Target Sleep" sub-state
Goto "Local Hibernate" sub-state
Dormant
Awake

Figure F.4 — Connection sleep procedure (only the entity A performs the hibernation)

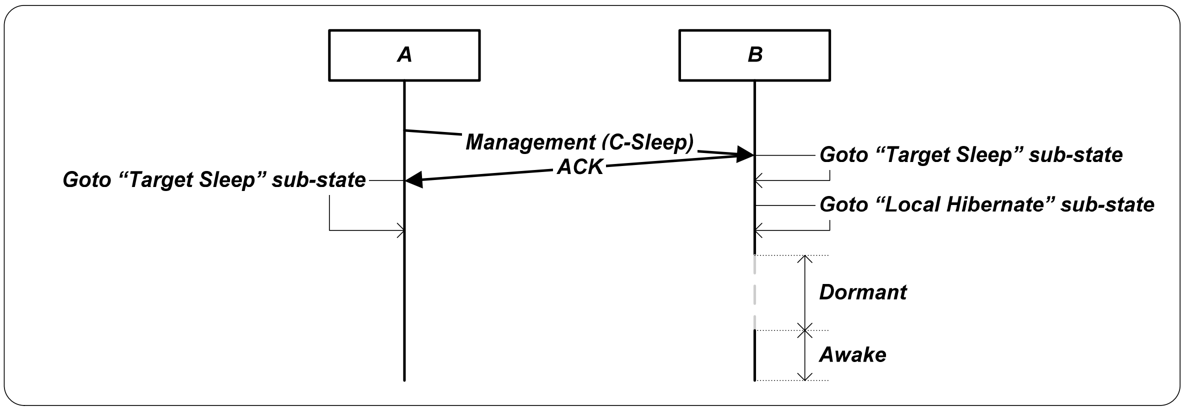

A
B
Management (C-Sleep)
ACK
Goto "Target Sleep" sub-state
Goto "Target Sleep" sub-state
Goto "Local Hibernate" sub-state
Dormant
Awake

Figure F.5 — Connection sleep procedure (only the entity B performs the hibernation)

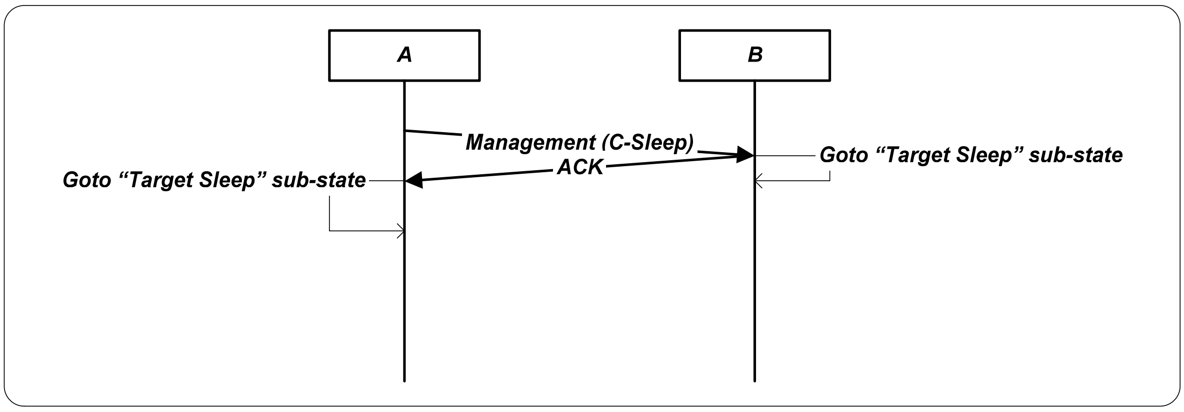


Figure F.6 — Connection sleep procedure (both entities do not perform the hibernation)

F.1.4 Connection wakeup procedure

The entity cancels hibernation when the C-Wake is received.

The entity transmits C-Wake to cancel hibernation of the other entity before transmitting any data frames or management frames assigned with a NoACK attribute as illustrated in Figure F.7.

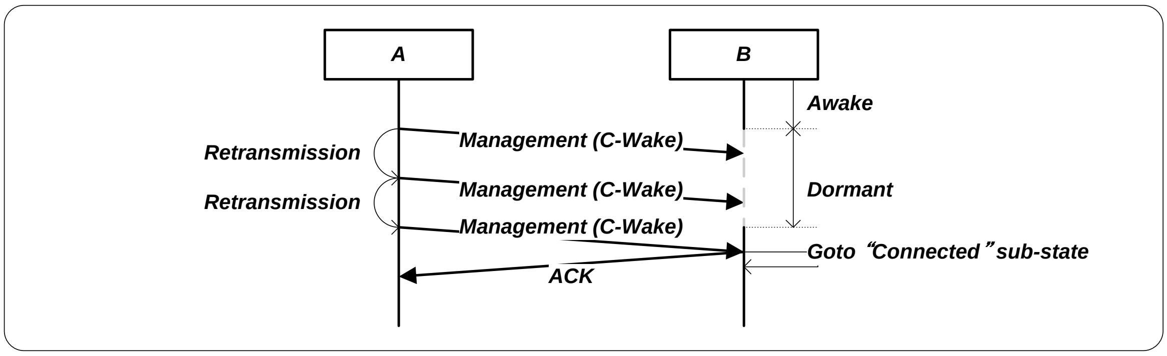


Figure F.7 — Connection wakeup procedure

F.1.5 Connection confirmation procedure

This procedure is intended to confirm whether the radio connection is alive. The entity (A) confirming the connection transmits the C-Probe. If the C-Probe is successfully received, the receiving peer (B) returns an ACK response. If Entity A cannot receive an ACK, entity A retransmits the C-Probe.

After an ACK response for the C-Probe, Entity B can continue hibernation as illustrated in Figure F.8.

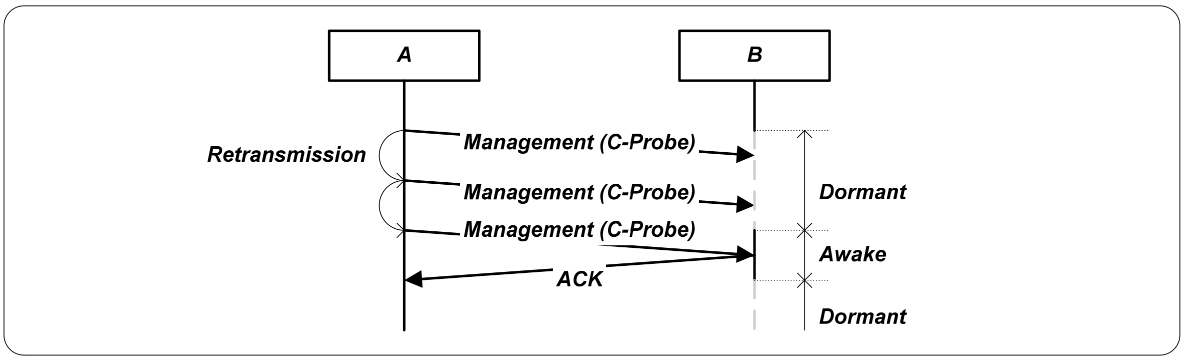


Figure F.8 — Connection confirmation procedure

F.1.6 Connection release procedure

An entity requests the other entity to release the connection by transmitting the C-Rls. The entity (A) that transmits the C-Rls discards the target UID that was set when the connection was established and changes to the Search state. The peer (B) that has received the C-Rls discards the target UID and changes to the Search state, as does Entity A as illustrated in Figure F.9.

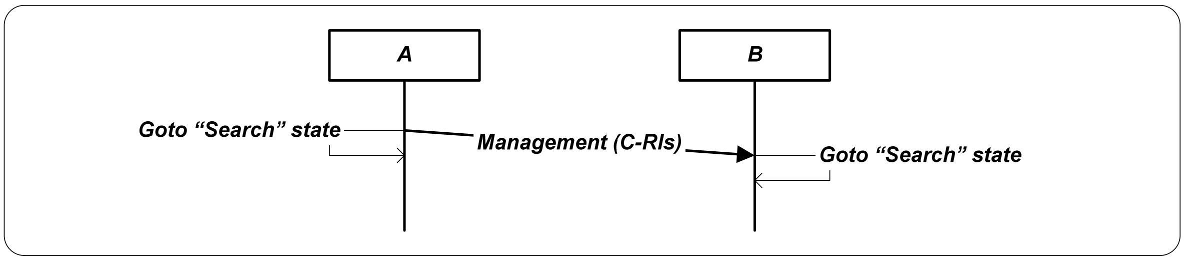


Figure F.9 — Connection release procedure

Annex G
(informative)

CNL service operation

G.1 Initialize operation

When the CNL User requests CNL_INIT service, the CNL initialises as illustrated in Figure G.1.

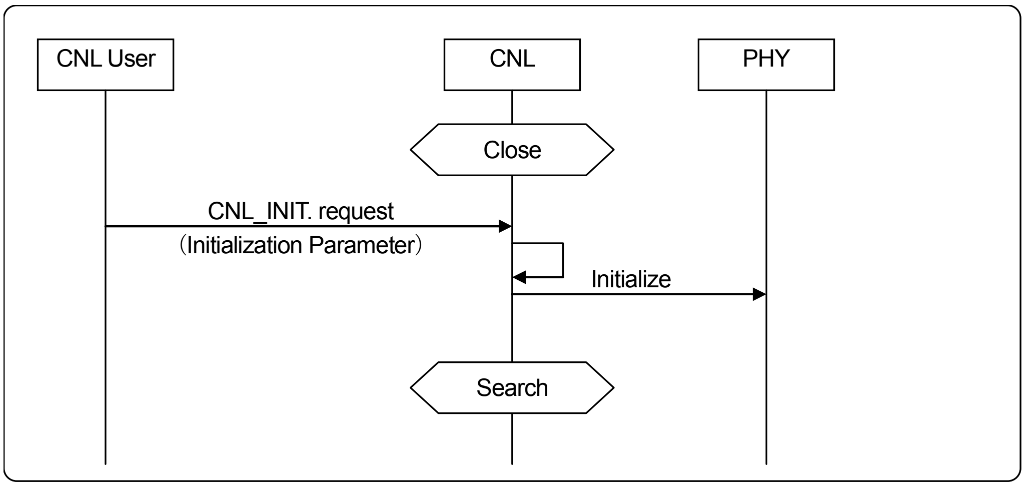


Figure G.1 — Initialize operation

G.2 Close operation

When the CNL User requests CNL_CLOSE service, the CNL finalises as illustrated in Figure G.2.

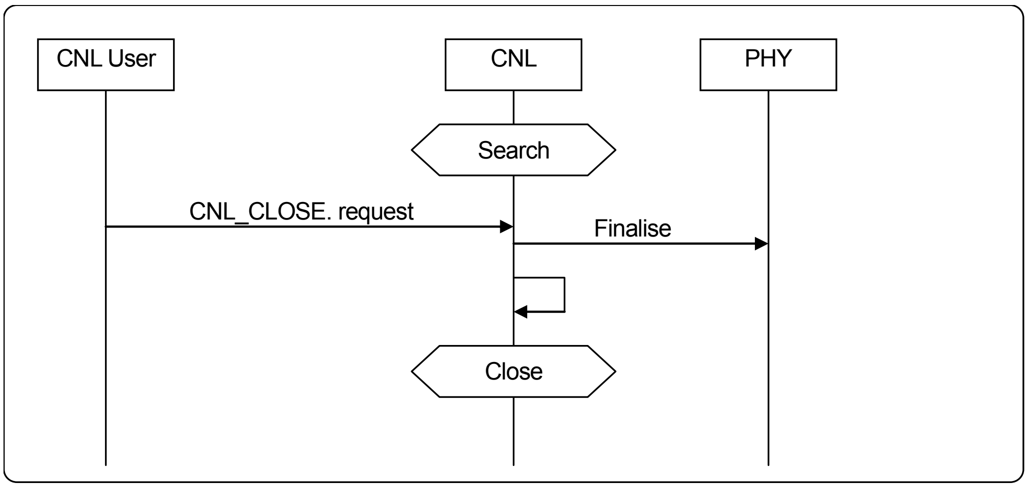


Figure G.2 — Close operation

G.3 Connect request

The procedure for establishing a connection as an initiator is illustrated below.

G.3.1 Connect request operation

When the CNL User requests CNL_CONNECT service, the CNL starts transmitting the C-Req as illustrated in Figure G.3.

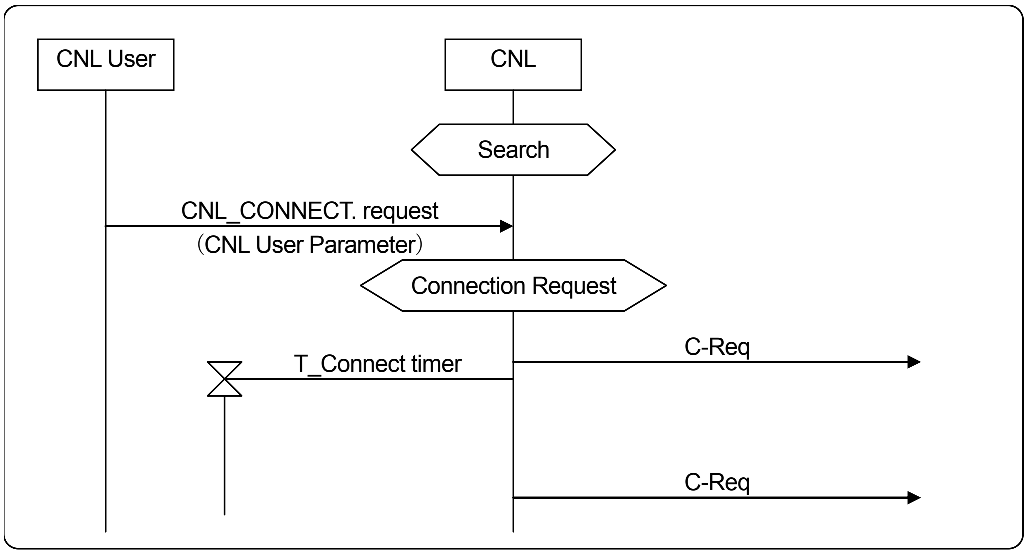


Figure G.3 — Connection request operation

G.3.2 Accept receive operation

When C-Acc is received from the target, the CNL notifies the CNL User as illustrated in Figure G.4.

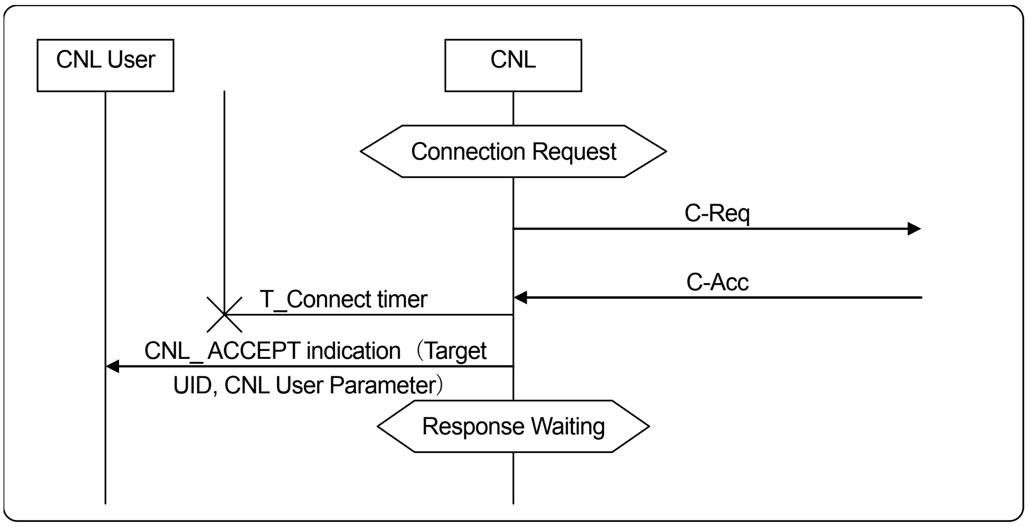


Figure G.4 — Accept receive operation

G.3.3 Accept response operation

When the CNL User requests CNL_ACCEPT service, the CNL starts transmitting an ACK to respond to the corresponding C-Acc as illustrated in Figure G.5.

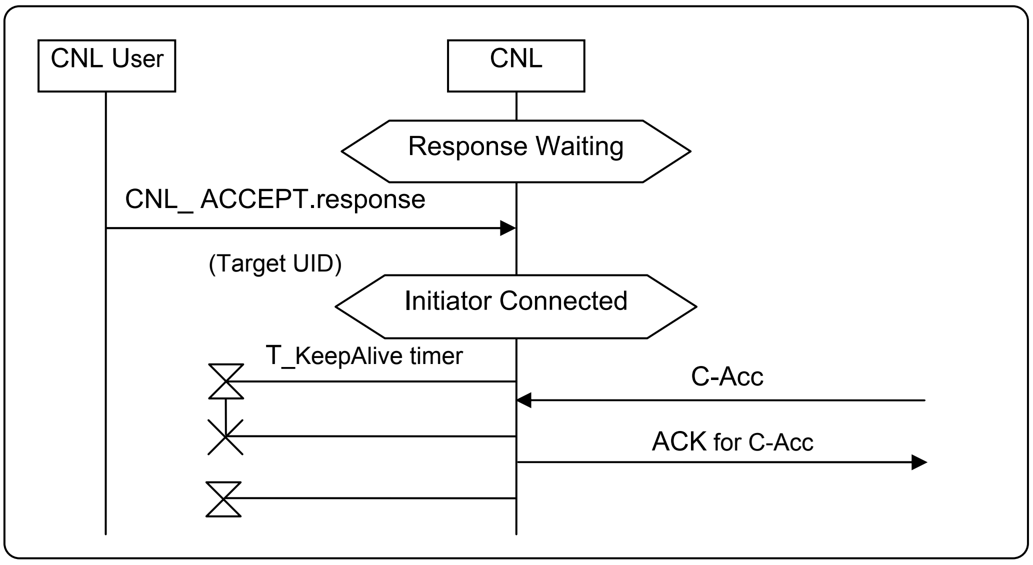


Figure G.5 — Accept response operation

G.3.4 Connect release operation

When T_Connect timer expires, the CNL notifies the CNL User as illustrated in Figure G.6.

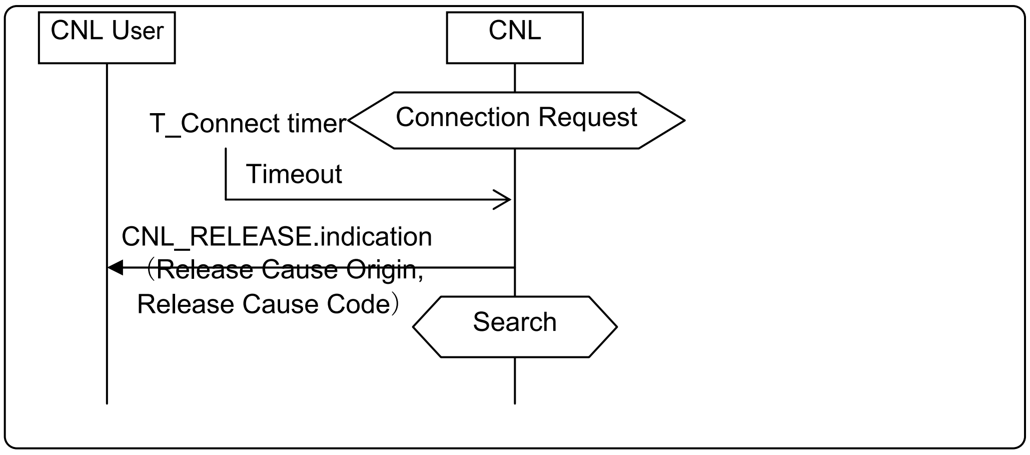


Figure G.6 — Connection release operation

G.3.5 Accept release operation

When the CNL User requests CNL_RELEASE service, the CNL transmits a C-Rls as illustrated in Figure G.7.

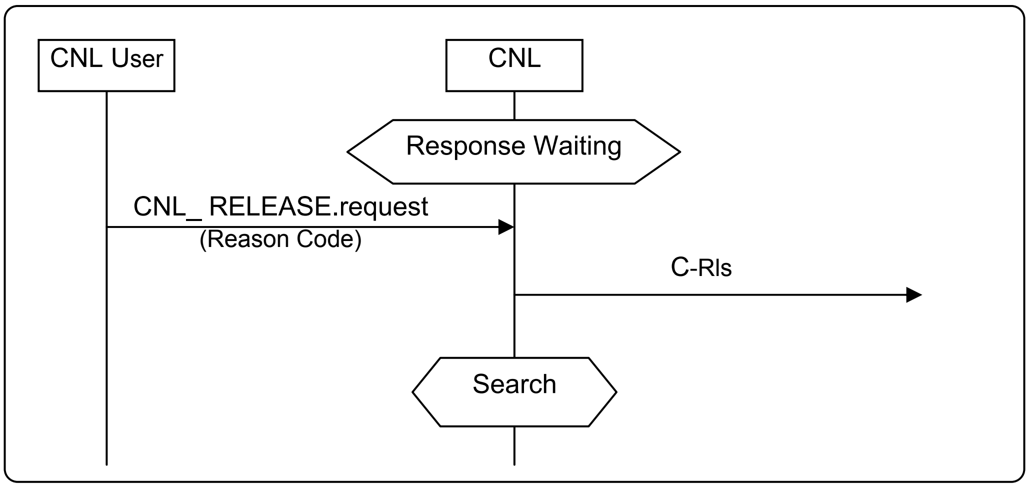


Figure G.7 — Accept release operation

G.4 Connect accept

The procedure that the responder uses to establish a connection is illustrated below.

G.4.1 Request receive operation

When the C-Req is received from the initiator, the CNL notifies the CNL User as illustrated in Figure G.8.

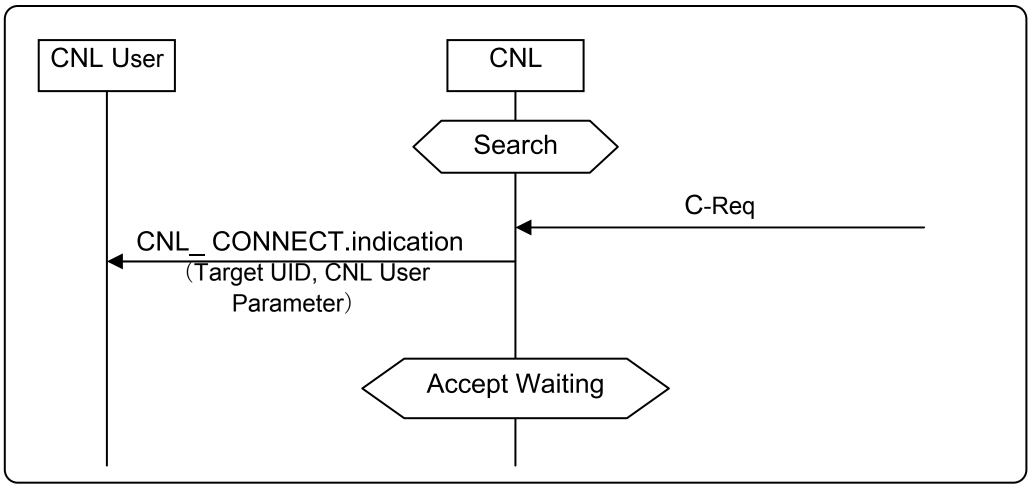


Figure G.8 — Request receive operation

G.4.2 Accept request operation

When the CNL User requests CNL_ACCEPT service, the CNL starts transmitting a C-Acc as illustrated in Figure G.9.

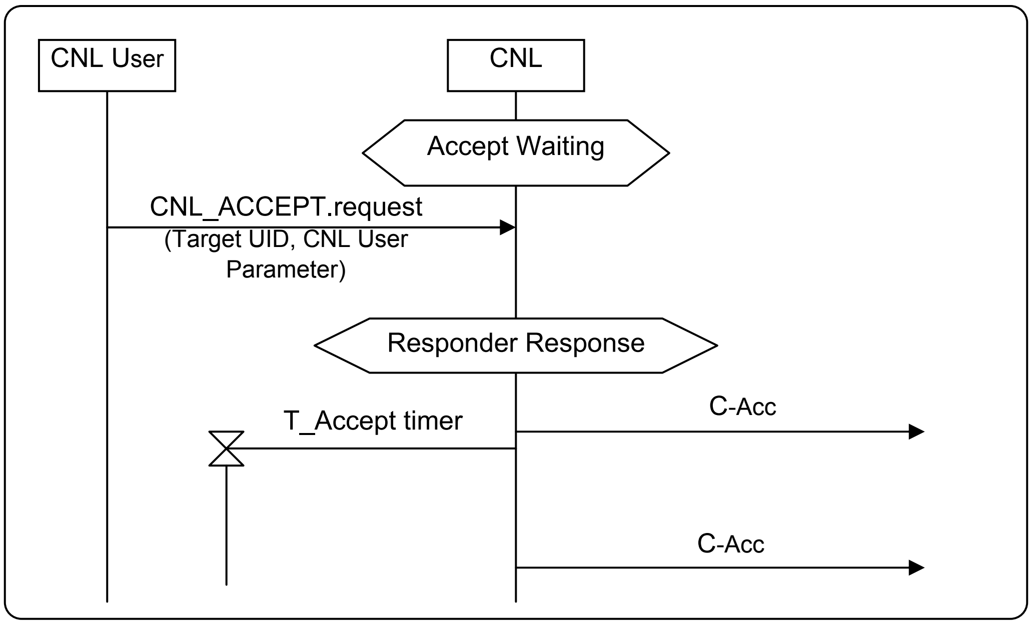


Figure G.9 — Accept request operation

G.4.3 Accept acknowledge operation

When an ACK for the C-Acc is received from the target, the CNL notifies the CNL User and starts operating as a responder as illustrated in Figure G.10.

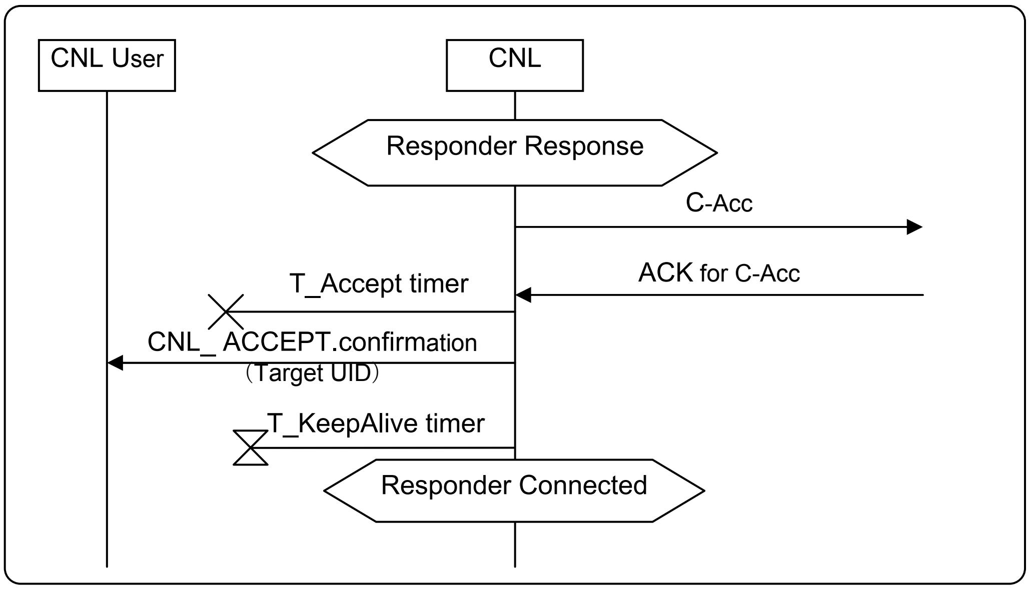


Figure G.10 — Accept acknowledge

G.4.4 Accept release operation

When T_Accept timer expires, the CNL notifies the CNL User as illustrated in Figure G.11.

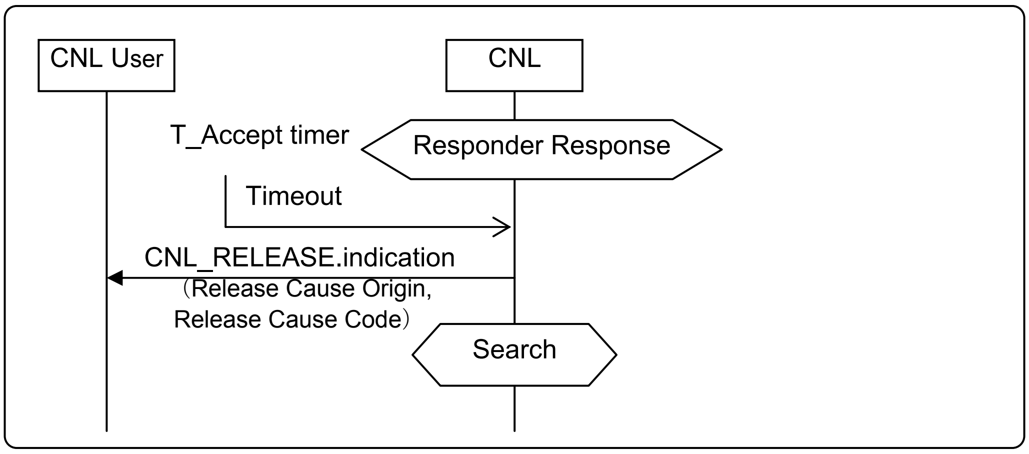


Figure G.11 — Accept release operation

G.4.5 Connect release operation

When the CNL User requests a CNL_RELEASE service, the CNL transmits a C-Rls as illustrated in Figure G.12.

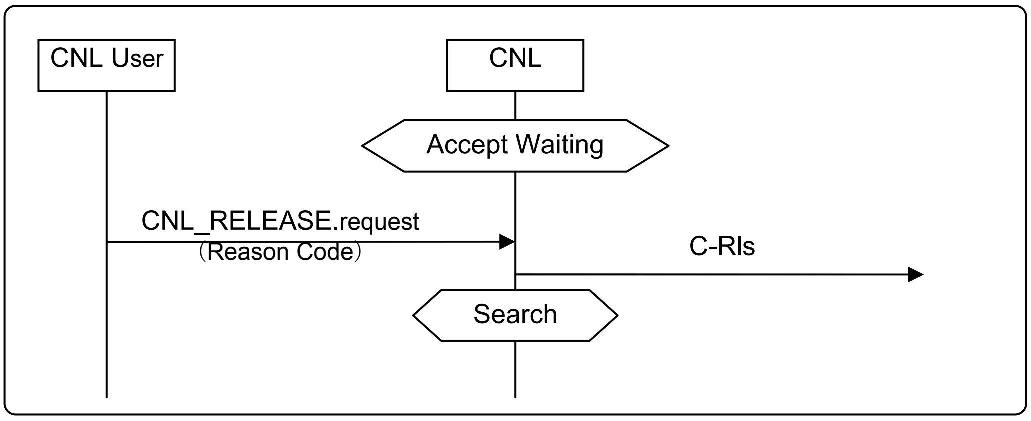


Figure G.12 — Connect release operation

G.4.6 Request crossover operation

When the C-Req is received during the C-Req transmission, the CNL sends a CNL_CONNECT.indication to notify the CNL User as illustrated in Figure G.13.

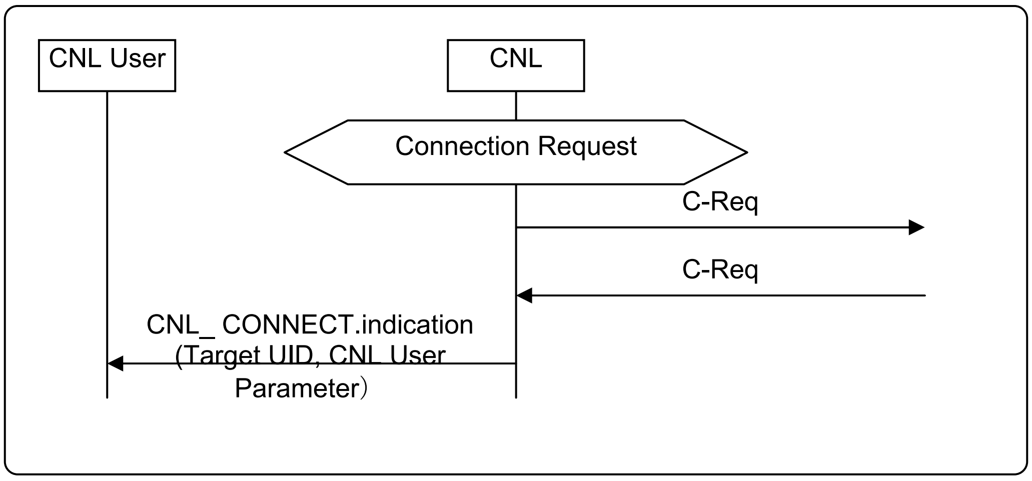


Figure G.13 — Request crossover operation

G.4.7 Accept request operation

When the CNL User requests CNL_ACCEPT service, the CNL starts transmitting the C-Acc as illustrated in Figure G.14.

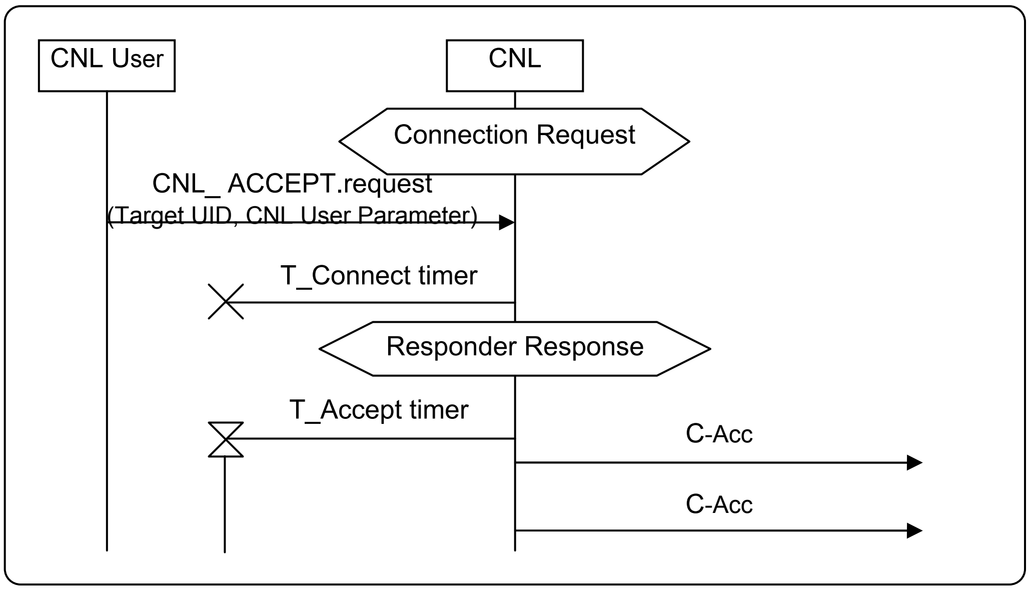


Figure G.14 — Accept request operation

G.4.8 Accept release operation

When the CNL User requests CNL_RELEASE service, the CNL transmits the C-Rls as illustrated in Figure G.15.

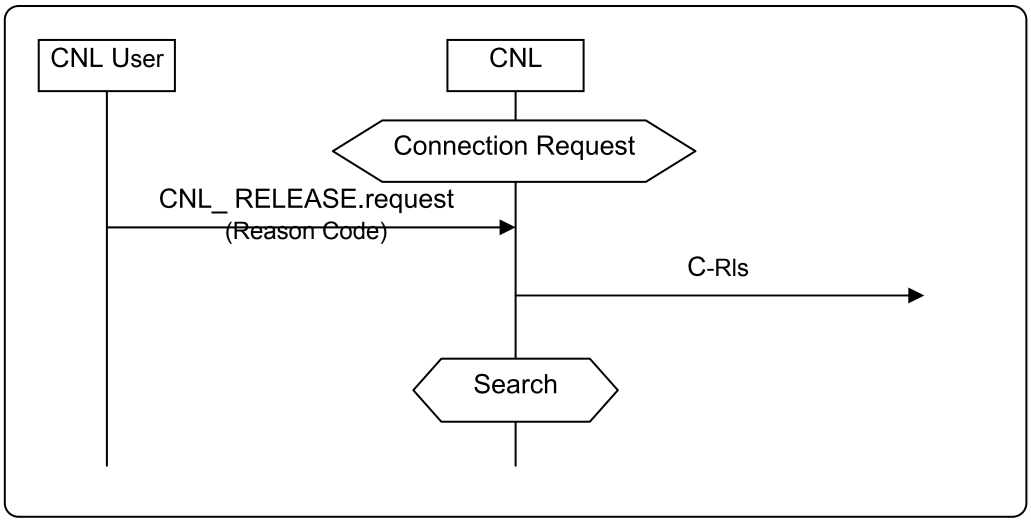


Figure G.15 — Accept release operation

G.5 Release

The connection release procedure is ordered from the CNL User or the target.

G.5.1 Release request receive operation

When the CNL User requests CNL_RELEASE service, the CNL transmits the C-RIs as illustrated in Figure G.16.

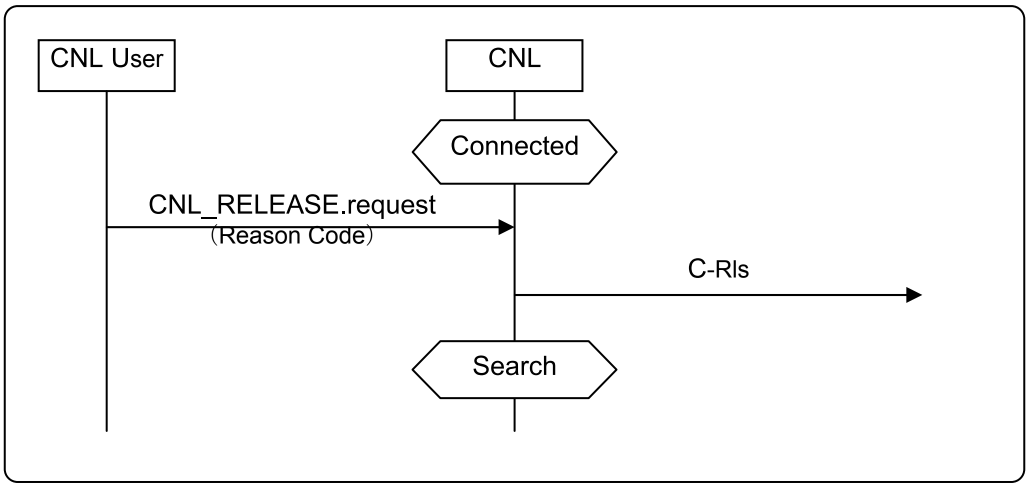


Figure G.16 — Request receive operation

G.5.2 Release receive operation

When the C_RIs is received from the target, the CNL notifies the CNL User as shown in Figure G.17.

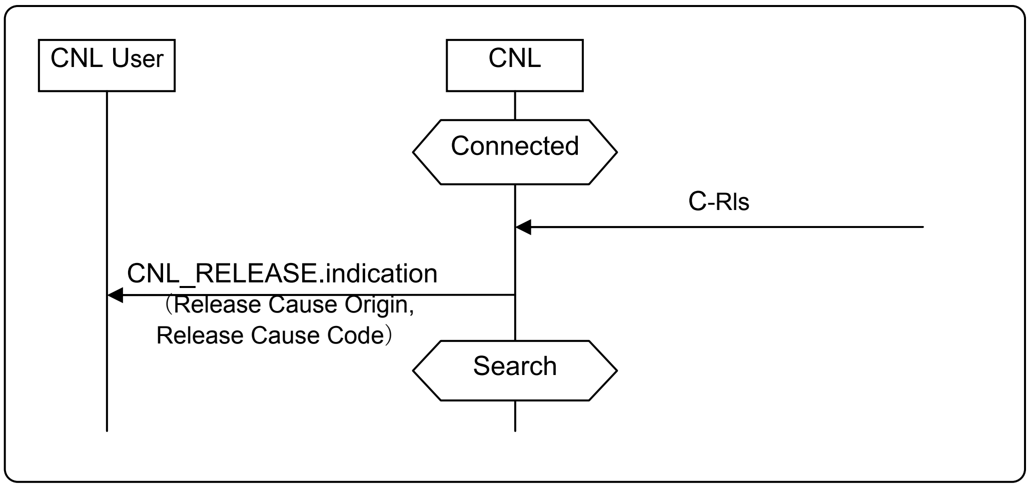


Figure G.17 — Release receive operation

G.6 Transfer data

The data transfer procedure is illustrated below.

G.6.1 Data send operation

When the CNL User requests CNL_DATA service, the CNL starts the data transmission process.

Data send operation is performed until an ACK response for the last segmented data frame is received as illustrated in Figure G.18.

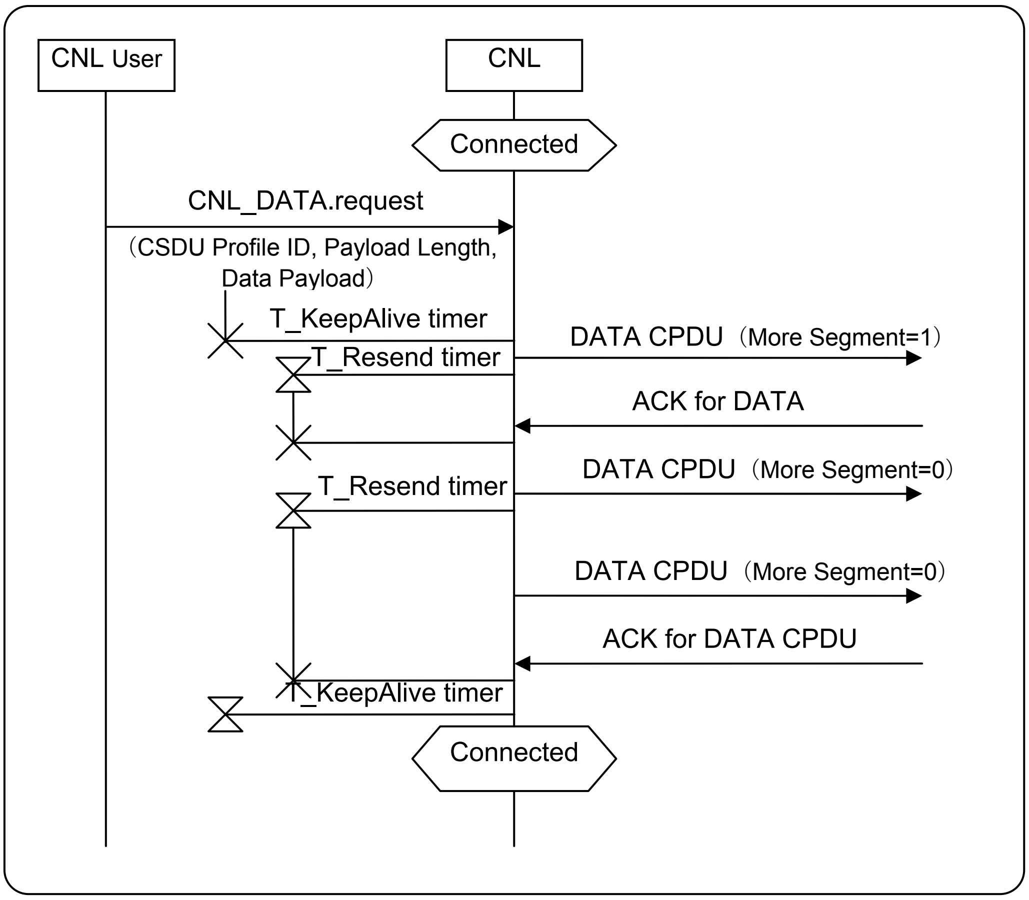


Figure G.18 — Data send operation

G.6.2 Data receive operation

When data is received from the target, the CNL notifies the CNL User.

The CNL notifies the CNL User after the last fragmented data frame is received as illustrated in Figure G.19.

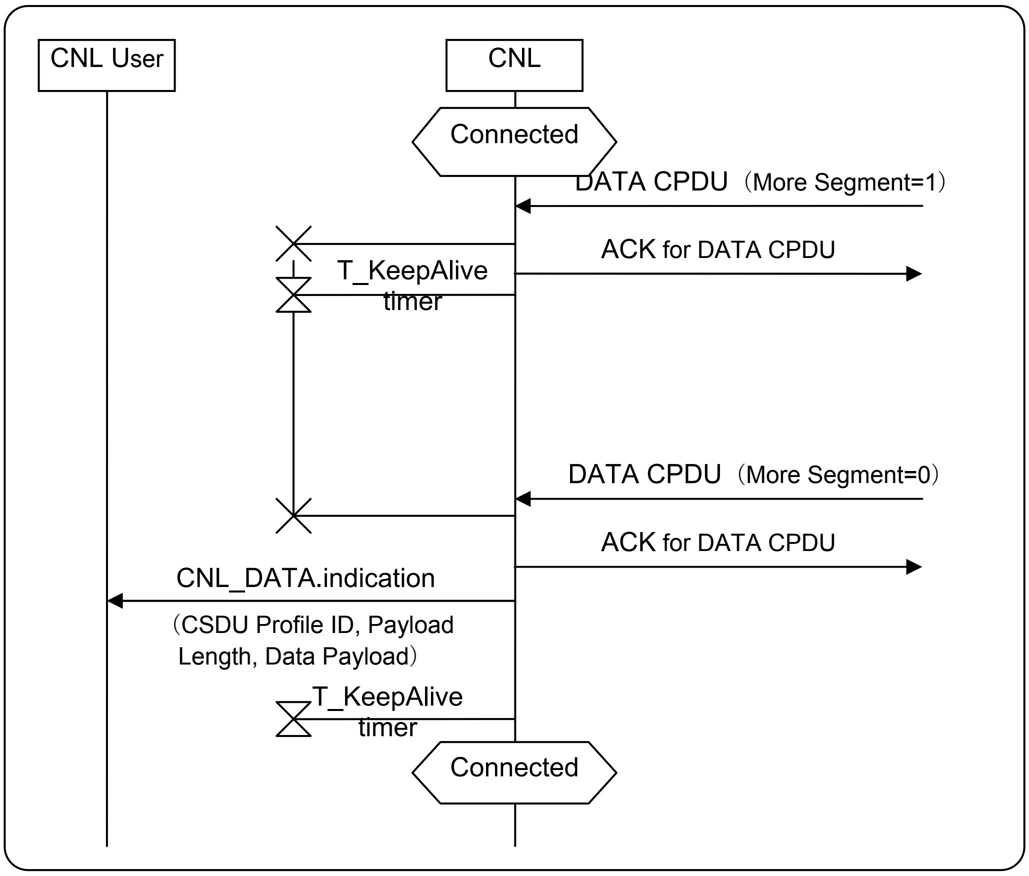


Figure G.19 — Data receive operation

G.6.3 Resend timeout operation

When the T_Resend timer expires, the CNL starts transmitting the C-Wake as illustrated in Figure G.20.

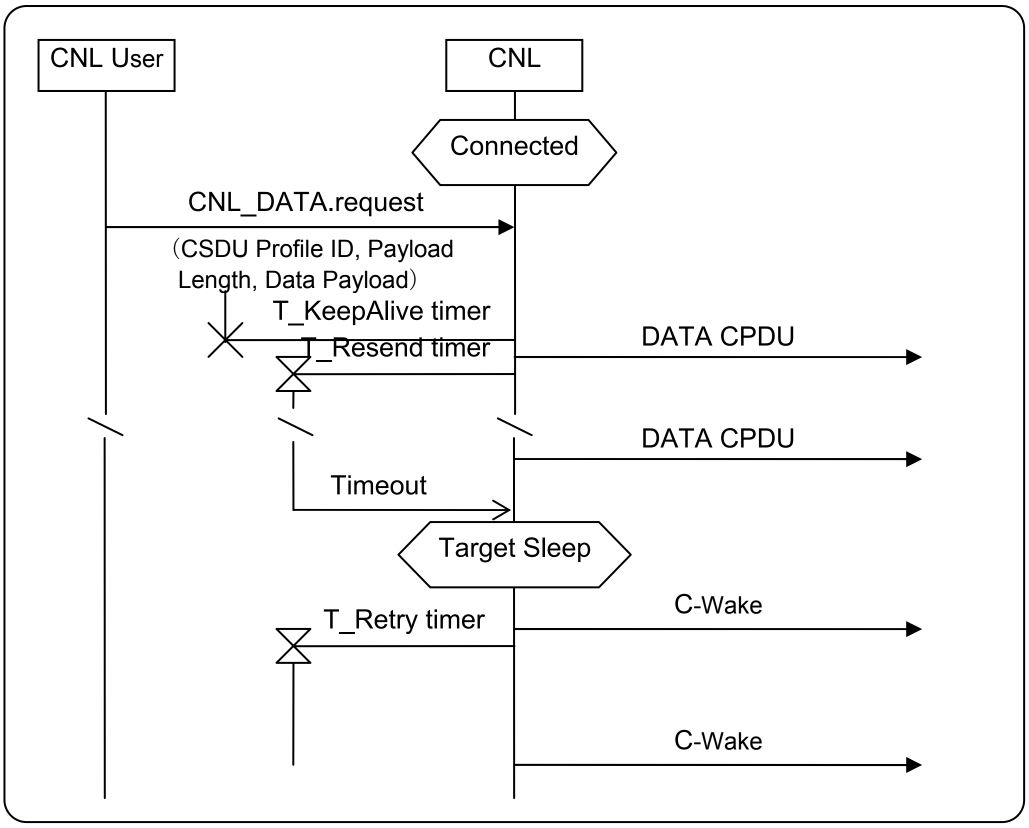


Figure G.20 — Retry timeout operation

G.6.4 Target wake operation

When an ACK response for C-Wake is received, the CNL starts to transmit the data as illustrated in Figure G.21.

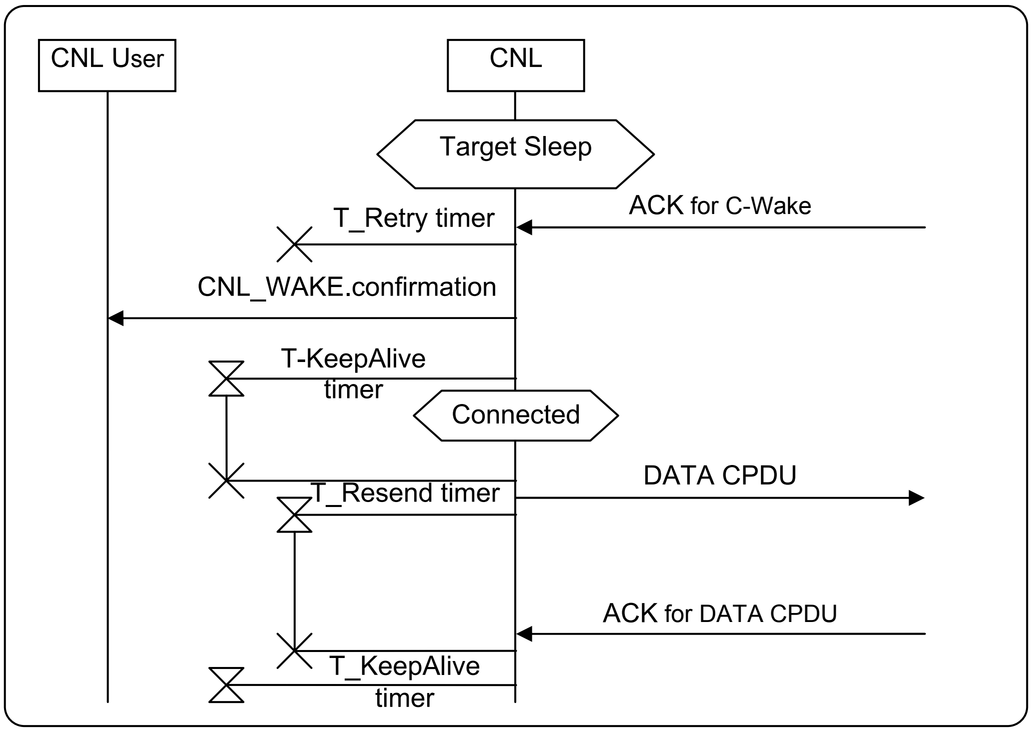


Figure G.21 — Target wake operation

G.7 Power save

The power saving procedure (optional) is illustrated below.

G.7.1 Power save request operation

When the CNL User requests CNL_POWERSAVE service, the CNL transmits a C-Sleep as illustrated in Figure G.22.

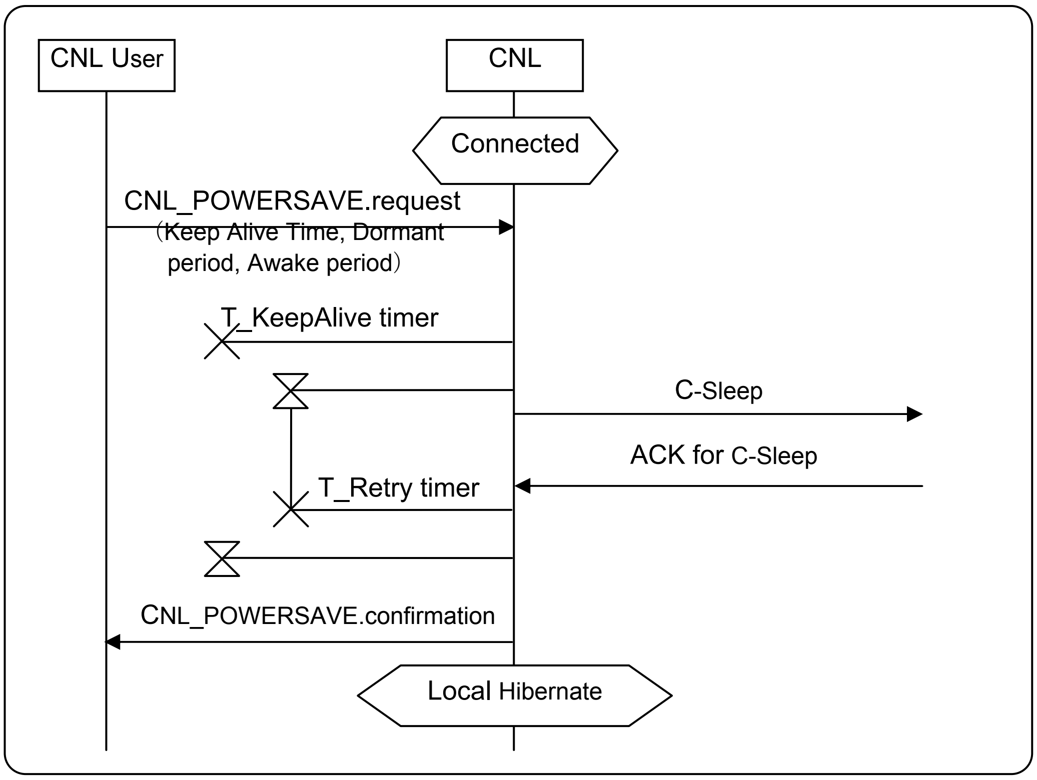


Figure G.22 — Power save request operation

G.7.2 Sleep receive operation

When the C-Sleep is received from the target, after an ACK response, the CNL changes to the Target Sleep sub-state as illustrated in Figure G.23.

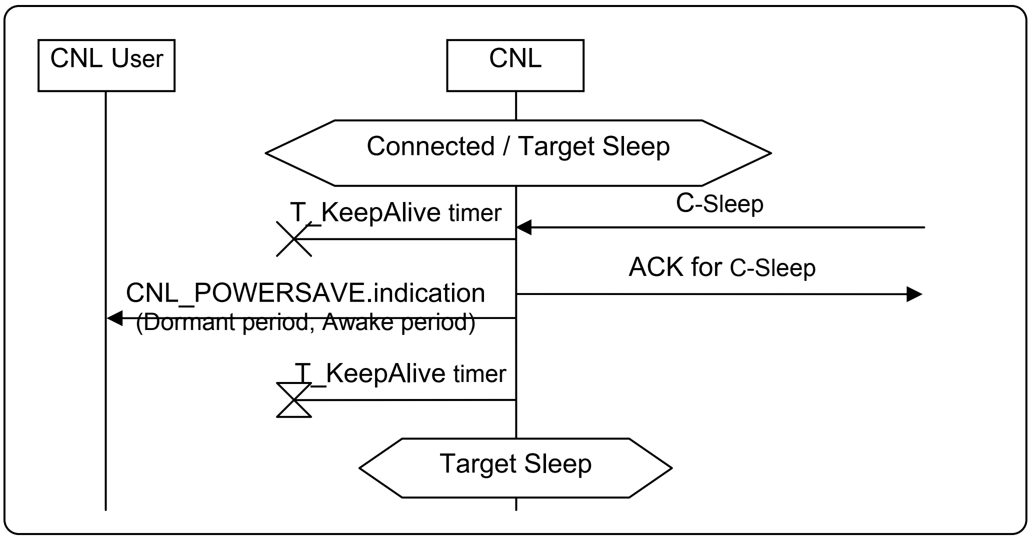


Figure G.23 — Sleep receive operation

G.8 Wakeup

Power saving cancellation procedure is illustrated.

G.8.1 Wakeup request operation

When the CNL User requests CNL_WAKE service, the CNL transmits a C-Wake to the target as illustrated in Figure G.24.

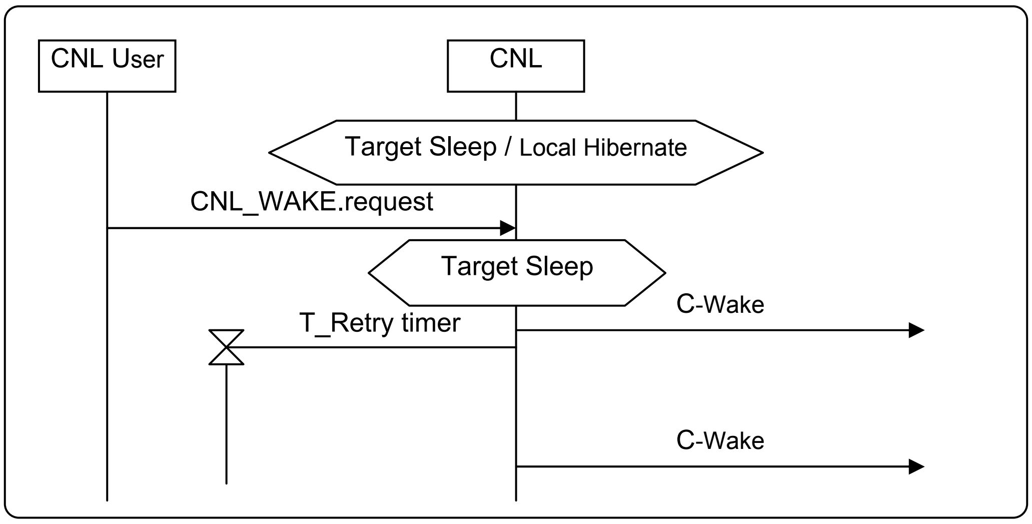


Figure G.24 — Wakeup request operation

G.8.2 Wakeup acknowledge operation

After an ACK response for the C-Wake is received, the CNL changes to the Connected sub-state as illustrated in Figure G.25.

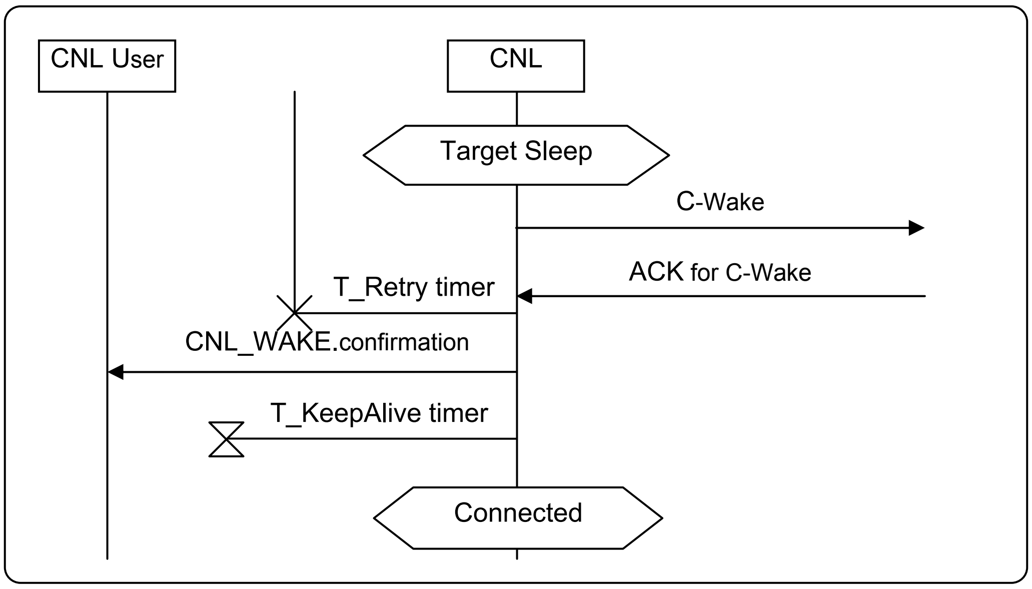


Figure G.25 — Wakeup acknowledge operation

G.8.3 Wakeup receive operation

When the C-Wake is received from the target, The CNL changes to the Connected sub-state after transmitting an ACK response for the C-Wake as illustrated in Figure G.26.

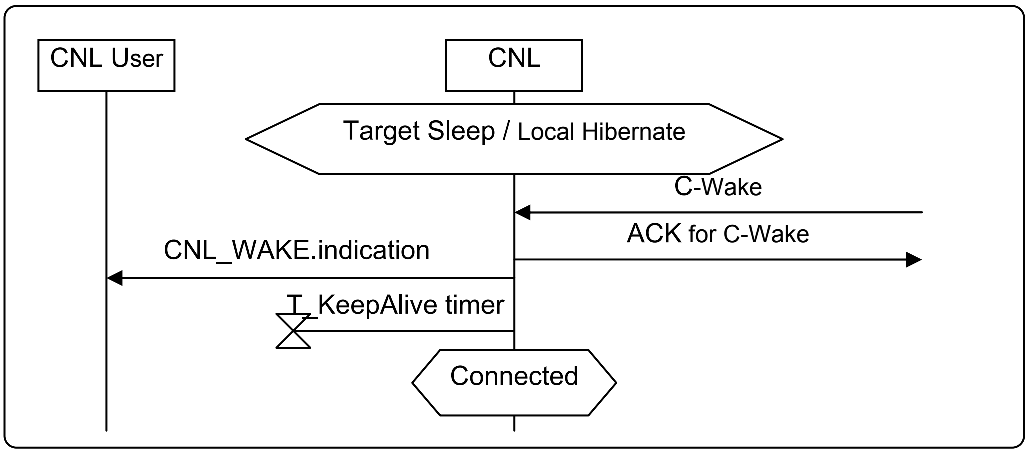


Figure G.26 — Wakeup receive operation

G.8.4 Data send request operation

When the CNL User requests CNL_DATA service while the CNL is in the Target Sleep sub-state, the CNL starts to transmit the C-Wake as illustrated in Figure G.27.

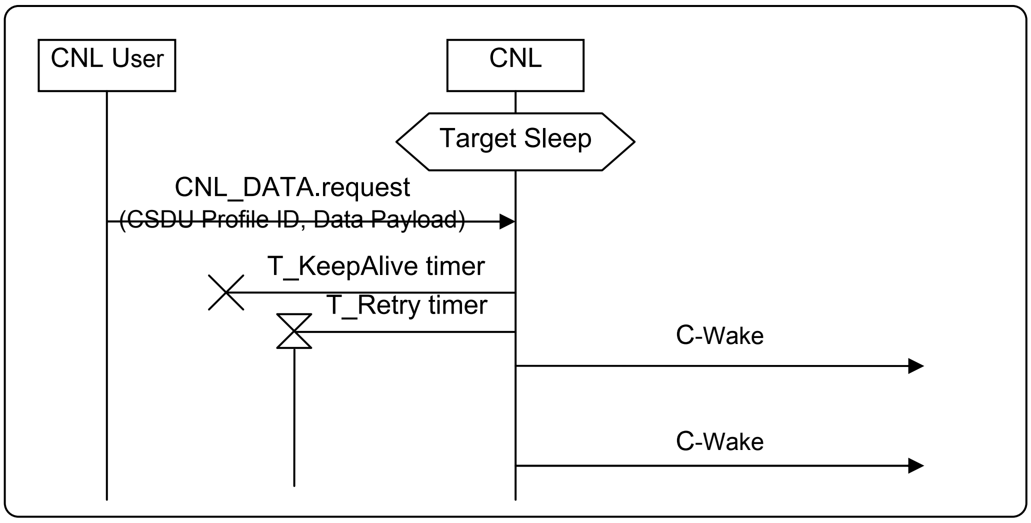


Figure G.27 — Data send request operation

G.8.5 Wakeup data send operation

After receiving an ACK response for the C-Wake, the CNL shall start transmitting data CPDUs as illustrated in Figure G.28.

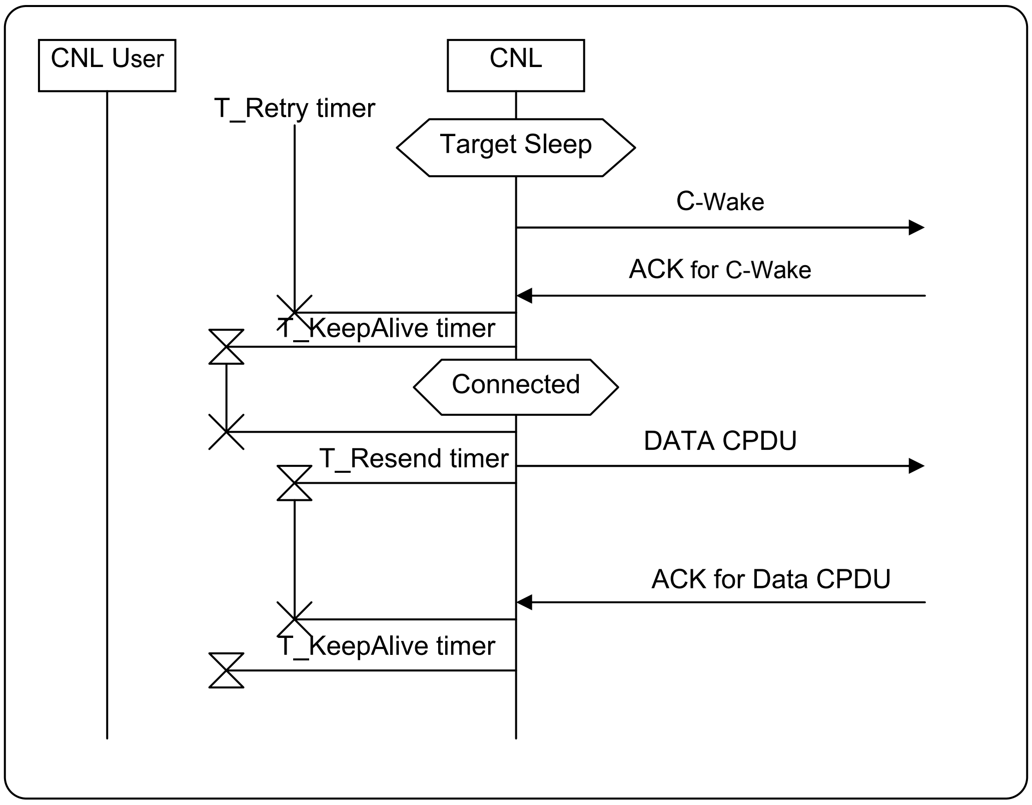


Figure G.28 — Wakeup data send operation

G.8.6 Wakeup timeout operation

When T_Retry timer expires, the CNL notifies the CNL User as illustrated in Figure G.29.

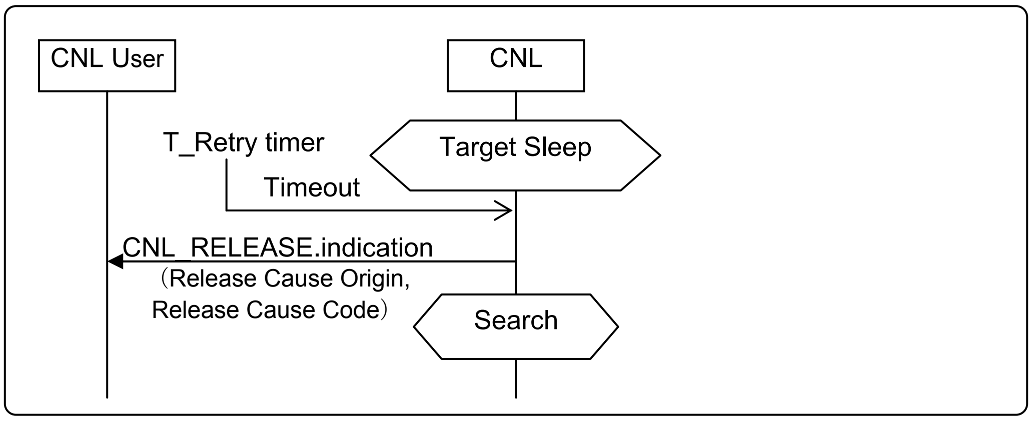


Figure G.29 — Wakeup timeout operation

G.9 Probe

This procedure is intended to confirm whether the connection is alive.

G.9.1 Probe send operation

When the T_KeepAlive timer expires, the CNL starts to transmit the C-Probe as illustrated in Figure G.30.

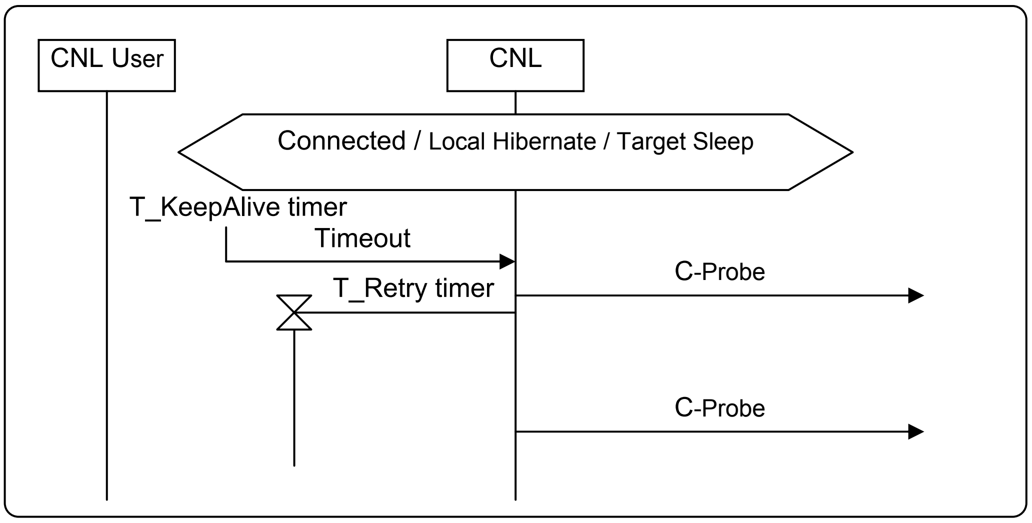


Figure G.30 — Probe send operation

G.10 Probe ACK receive operation

When an ACK response for the C-Probe is received from the target, CNL restarts the T_KeepAlive timer from zero as illustrated in Figure G.31.

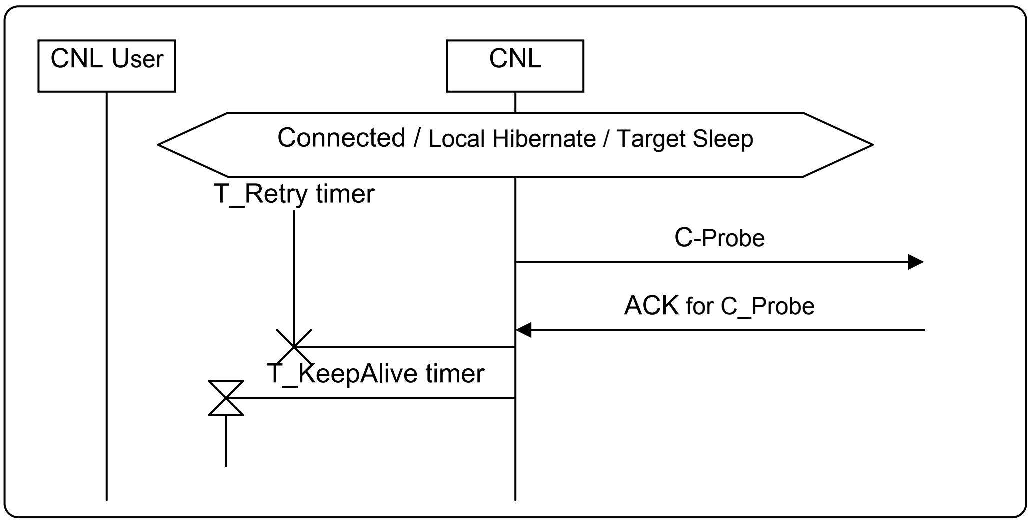


Figure G.31 — Probe ACK receive operation

G.10.1 Probe receive operation

When the C-Probe is received from the target, the CNL restarts the T_KeepAlive timer from zero after transmitting the ACK response for the C-Probe as illustrated in Figure G.32.

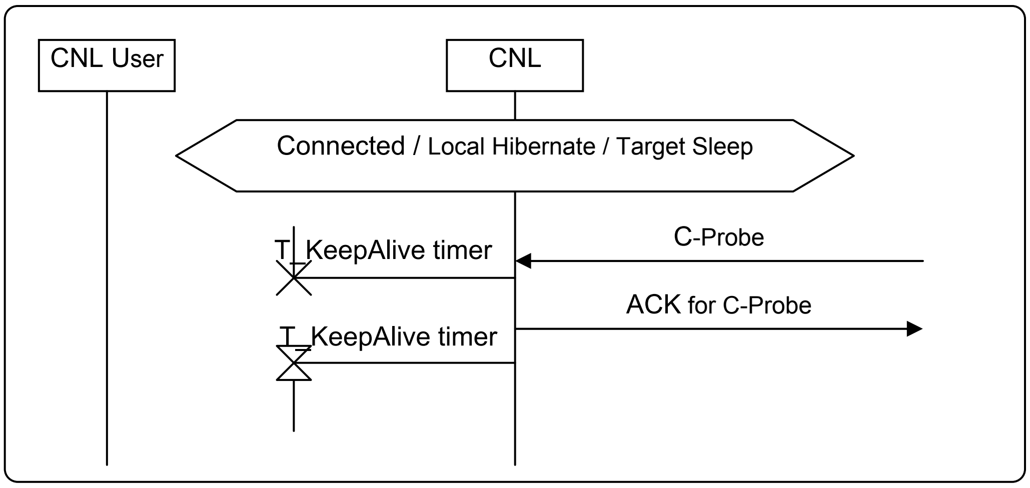


Figure G.32 — Probe receive operation

G.10.2 Probe timeout operation

When the T_Retry timer expires during the C-Probe transmission, the CNL notifies the CNL User as illustrated in Figure G.33 below.

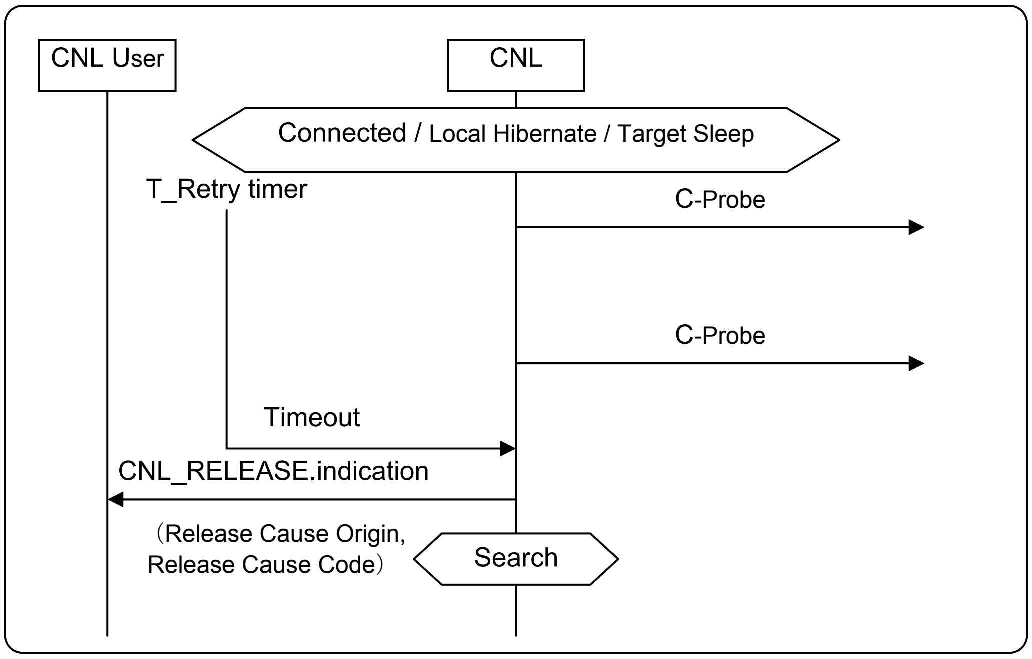


Figure G.33 — Probe timeout operation