**3GPP TSG-RAN WG2 Meeting #131bis R2-xxxx**

**Prague, Czech Republic, 13th – 17th Oct., 2025**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *CR-Form-v12.3* | | | | | | | | |
| **CHANGE REQUEST** | | | | | | | | |
|  | | | | | | | | |
|  | **38.322** | **CR** | **0066** | **rev** | **1** | **Current version:** | **19.0.0** |  |
|  | | | | | | | | |
| *For* [***HELP***](http://www.3gpp.org/3G_Specs/CRs.htm#_blank)*on using this form: comprehensive instructions can be found at* [*http://www.3gpp.org/Change-Requests*](http://www.3gpp.org/Change-Requests)*.* | | | | | | | | |
|  | | | | | | | | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***Proposed change affects:*** | UICC apps |  | ME | **X** | Radio Access Network | **X** | Core Network |  |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | | | | |
| ***Title:*** | Miscellaneous corrections on RLC for R19 XR | | | | | | | | | |
|  |  | | | | | | | | | |
| ***Source to WG:*** | vivo | | | | | | | | | |
| ***Source to TSG:*** | R2 | | | | | | | | | |
|  |  | | | | | | | | | |
| ***Work item code:*** | NR\_XR\_Ph3-Core | | | | |  | ***Date:*** | | | 2025-10-17 |
|  |  | | | |  | |  | | |  |
| ***Category:*** | **F** |  | | | | | ***Release:*** | | | Rel-19 |
|  | *Use one of the following categories:* ***F*** *(correction)* ***A*** *(mirror corresponding to a change in an earlier release)* ***B*** *(addition of feature),* ***C*** *(functional modification of feature)* ***D*** *(editorial modification)*  Detailed explanations of the above categories can be found in 3GPP [TR 21.900](http://www.3gpp.org/ftp/Specs/html-info/21900.htm). | | | | | | | | *Use one of the following releases: Rel-8 (Release 8) Rel-9 (Release 9) Rel-10 (Release 10) Rel-11 (Release 11) … Rel-17 (Release 17) Rel-18 (Release 18) Rel-19 (Release 19)  Rel-20 (Release 20)* | |
|  |  | | | | | | | | | |
| ***Reason for change:*** | | TBD | | | | | | | | |
|  | |  | | | | | | | | |
| ***Summary of change:*** | | 1. TBD | | | | | | | | |
|  | |  | | | | | | | | |
| ***Consequences if not approved:*** | | TBD | | | | | | | | |
|  | |  | | | | | | | | |
| ***Clauses affected:*** | | 5.2.3.1.1, 5.2.3.2.5, 5.3.3.3, 7.1 | | | | | | | | |
|  | |  | | | | | | | | |
|  | | **Y** | **N** |  | | | |  | | |
| ***Other specs*** | |  | **X** | Other core specifications | | | | TS/TR ... CR ... | | |
| ***affected:*** | |  | **X** | Test specifications | | | | TS/TR ... CR ... | | |
| ***(show related CRs)*** | |  | **X** | O&M Specifications | | | | TS/TR ... CR ... | | |
|  | |  | | | | | | | | |
| ***Other comments:*** | |  | | | | | | | | |
|  | |  | | | | | | | | |
| ***This CR's revision history:*** | |  | | | | | | | | |

Start of change

5.2.3.1.1 General

The transmitting side of an AM RLC entity shall prioritize transmission of RLC control PDUs over AMD PDUs. The transmitting side of an AM RLC entity shall prioritize transmission of AMD PDUs containing previously transmitted RLC SDUs or RLC SDU segments over transmission of AMD PDUs containing not previously transmitted RLC SDUs or RLC SDU segments.

The transmitting side of an AM RLC entity shall maintain a transmitting window according to the state variable TX\_Next\_Ack as follows:

- a SN falls within the transmitting window if TX\_Next\_Ack <= SN < TX\_Next\_Ack + AM\_Window\_Size;

- a SN falls outside of the transmitting window otherwise.

The transmitting side of an AM RLC entity shall not submit to lower layer any AMD PDU whose SN falls outside of the transmitting window.

For each RLC SDU received from the upper layer, the AM RLC entity shall:

- associate a SN with the RLC SDU equal to TX\_Next and construct an AMD PDU by setting the SN of the AMD PDU to TX\_Next;

- increment TX\_Next by one.

When submitting an AMD PDU that contains a segment of an RLC SDU, to lower layer, the transmitting side of an AM RLC entity shall:

- set the SN of the AMD PDU to the SN of the corresponding RLC SDU.

The transmitting side of an AM RLC entity can receive a positive acknowledgement (confirmation of successful reception by its peer AM RLC entity) for an RLC SDU by the following:

- STATUS PDU from its peer AM RLC entity.

When receiving a positive acknowledgement for an RLC SDU with SN = x, the transmitting side of an AM RLC entity shall:

- send an indication to the upper layers of successful delivery of the RLC SDU;

- set TX\_Next\_Ack equal to the SN of the RLC SDU with the smallest SN, whose SN falls within the range TX\_Next\_Ack <= SN <= TX\_Next and for which a positive acknowledgment has not been received yet.

If *stopReTxDiscardedSDU* is configured, when indicated from upper layer to discard a particular RLC SDU (see TS 38.323 [4]), the transmitting side of an AM RLC entity shall not consider the corresponding RLC SDU for retransmission, or the corresponding RLC SDU segment(s) for (re)transmission, if the RLC SDU or the RLC SDU segment(s) thereof has been submitted to lower layers.

NOTE Y: The corresponding RLC SDU or RLC SDU segment(s) above includes RLC SDU or RLC SDU segment(s) which have been already considered or pending for transmission or retransmission.

Next change

##### 5.2.3.2.5 Actions when *t-RxDiscard* expires

When *t-RxDiscard* expires, the receiving side of an AM RLC entity shall:

- discard the AMD PDU(s) in the reception buffer with SN < RX\_Next\_Discard\_Trigger, if any;

- consider the RLC SDU(s) with SN < RX\_Next\_Discard\_Trigger, if any, as completely received;

- update RX\_Next to the SN of the first RLC SDU with SN >= RX\_Next\_Discard\_Trigger for which not all bytes have been received;

- if RX\_Next\_Highest > RX\_Next +1; or

- if RX\_Next\_Highest = RX\_Next + 1 and there is at least one missing byte segment of the SDU associated with SN = RX\_Next before the last byte of all received segments of this SDU:

- start *t-RxDiscard*;

- set RX\_Next\_Discard\_Trigger to RX\_Next\_Highest.

Next change

5.3.3.3 Reception of a STATUS report

Upon reception of a STATUS report from the receiving RLC AM entity the transmitting side of an AM RLC entity shall:

- if the STATUS report comprises a positive or negative acknowledgement for the RLC SDU with sequence number equal to POLL\_SN:

- if *t-PollRetransmit* is running:

- stop and reset *t-PollRetransmit*.

NOTE: If *stopRetxDiscardedSDU* is configured, when all RLC SDUs with SNs up to and including POLL\_SN are already positively or negatively acknowledged or indicated as discarded from upper layer (e.g., PDCP), the transmitting side of an AM RLC entity may stop and reset the running *t-PollRetransmit*.

Next change

7 Variables, constants and timers

7.1 State variables

This clause describes the state variables used in AM and UM entities in order to specify the RLC protocol. The state variables defined in this clause are normative.

All state variables and all counters are non-negative integers.

All state variables related to AM data transfer can take values from 0 to 4095 for 12 bit SN or from 0 to 262143 for 18 bit SN. All arithmetic operations contained in the present document on state variables related to AM data transfer are affected by the AM modulus (i.e. final value = [value from arithmetic operation] modulo 4096 for 12 bit SN and 262144 for 18 bit SN).

All state variables related to UM data transfer can take values from 0 to 63 for 6 bit SN or from 0 to 4095 for 12 bit SN. All arithmetic operations contained in the present document on state variables related to UM data transfer are affected by the UM modulus (i.e. final value = [value from arithmetic operation] modulo 64 for 6 bit SN and 4096 for 12 bit SN).

When performing arithmetic comparisons of state variables or SN values, a modulus base shall be used.

TX\_Next\_Ack and RX\_Next shall be assumed as the modulus base at the transmitting side and receiving side of an AM RLC entity, respectively. This modulus base is subtracted from all the values involved, and then an absolute comparison is performed (e.g. RX\_Next <= SN < RX\_Next + AM\_Window\_Size is evaluated as [RX\_Next – RX\_Next] modulo 2[*sn-FieldLength*] <= [SN – RX\_Next] modulo 2[*sn-FieldLength*] < [RX\_Next + AM\_Window\_Size – RX\_Next] modulo 2[*sn-FieldLength*]), where *sn-FieldLength* is 12 or 18 for 12 bit SN and 18 bit SN, respectively.

RX\_Next\_Highest– UM\_Window\_Size shall be assumed as the modulus base at the receiving UM RLC entity. This modulus base is subtracted from all the values involved, and then an absolute comparison is performed (e.g. (RX\_Next\_Highest– UM\_Window\_Size) <= SN < RX\_Next\_Highest is evaluated as [(RX\_Next\_Highest– UM\_Window\_Size) – (RX\_Next\_Highest– UM\_Window\_Size)] modulo 2[*sn-FieldLength*] <= [SN – (RX\_Next\_Highest– UM\_Window\_Size)] modulo 2[*sn-FieldLength*] < [RX\_Next\_Highest– (RX\_Next\_Highest– UM\_Window\_Size)] modulo 2[*sn-FieldLength*]), where *sn-FieldLength* is 6 or 12 for 6 bit SN and 12 bit SN, respectively.

The transmitting side of each AM RLC entity shall maintain the following state variables:

a) TX\_Next\_Ack – Acknowledgement state variable

This state variable holds the value of the SN of the next RLC SDU for which a positive acknowledgment is to be received in-sequence, and it serves as the lower edge of the transmitting window. It is initially set to 0, and is updated whenever the AM RLC entity receives a positive acknowledgment for an RLC SDU with SN = TX\_Next\_Ack.

b) TX\_Next – Send state variable

This state variable holds the value of the SN to be assigned for the next newly generated AMD PDU. It is initially set to 0, and is updated whenever the AM RLC entity constructs an AMD PDU with SN = TX\_Next and contains an RLC SDU or the last segment of a RLC SDU.

c) POLL\_SN – Poll send state variable

This state variable holds the value of the highest SN of the AMD PDU among the AMD PDUs submitted to lower layer when POLL\_SN is set according to clause 5.3.3.2. It is initially set to 0.

The transmitting side of each AM RLC entity shall maintain the following counters:

a) PDU\_WITHOUT\_POLL – Counter

This counter is initially set to 0. It counts the number of AMD PDUs sent since the most recent poll bit was transmitted.

b) BYTE\_WITHOUT\_POLL – Counter

This counter is initially set to 0. It counts the number of data bytes sent since the most recent poll bit was transmitted.

c) RETX\_COUNT – Counter

This counter counts the number of retransmissions of an RLC SDU or RLC SDU segment (see clause 5.3.2). There is one RETX\_COUNT counter maintained per RLC SDU. This counter is reset to zero for each RLC SDU when indicated by upper layer.

The receiving side of each AM RLC entity shall maintain the following state variables:

a) RX\_Next – Receive state variable

This state variable holds the value of the SN following the last in-sequence completely received RLC SDU, and it serves as the lower edge of the receiving window. It is initially set to 0, and is updated whenever the AM RLC entity receives an RLC SDU with SN = RX\_Next or *t-RxDiscard* expires.

b) RX\_Next\_Status\_Trigger – *t-Reassembly* state variable

This state variable holds the value of the SN following the SN of the RLC SDU which triggered *t-Reassembly*.

c) RX\_Highest\_Status – Maximum STATUS transmit state variable

This state variable holds the highest possible value of the SN which can be indicated by "ACK\_SN" when a STATUS PDU needs to be constructed. It is initially set to 0.

d) RX\_Next\_Highest – Highest received state variable

This state variable holds the value of the SN following the SN of the RLC SDU with the highest SN among received RLC SDUs. It is initially set to 0.

e) RX\_Next\_Discard\_Trigger – *t-RxDiscard* state variable

This state variable holds the value of the SN following the SN of the RLC SDU which triggered *t-RxDiscard*. This state variable shall be maintained only when the AM RLC entity is configured with *t-RxDiscard*.

Each transmitting UM RLC entity shall maintain the following state variables:

a) TX\_Next – UM send state variable

This state variable holds the value of the SN to be assigned for the next newly generated UMD PDU with segment. It is initially set to 0, and is updated after the UM RLC entity submits a UMD PDU including the last segment of an RLC SDU to lower layers.

Each receiving UM RLC entity shall maintain the following state variables:

a) RX\_Next\_Reassembly – UM receive state variable

This state variable holds the value of the earliest SN that is still considered for reassembly. It is initially set to 0. For groupcast and broadcast of NR sidelink communication or for SL-SRB4 of NR sidelink discovery, it is initially set to the SN of the first received UMD PDU containing an SN. For the receiving UM RLC entity configured for MCCH or MTCH, it is up to UE implementation to set the initial value of RX\_Next\_Reassembly to a value before RX\_Next\_Highest.

b) RX\_Timer\_Trigger – UM *t-Reassembly* state variable

This state variable holds the value of the SN following the SN which triggered *t-Reassembly*.

c) RX\_Next\_Highest– UM receive state variable

This state variable holds the value of the SN following the SN of the UMD PDU with the highest SN among received UMD PDUs. It serves as the higher edge of the reassembly window. It is initially set to 0. For groupcast and broadcast of NR sidelink communication or for SL-SRB4 of NR sidelink discovery, it is initially set to the SN of the first received UMD PDU containing an SN. For the receiving UM RLC entity configured for MCCH or MTCH, it is initially set to the SN of the first received UMD PDU containing an SN.

End of change