**3GPP TSG-RAN WG2 Meeting #112-e R2-20xxxxx**

**Online, xx-yy November 2020**

**Agenda Item: x.y**

**Source: MediaTek Inc.**

**Title: Summary of email discussion [Post111-e][901] Extension scenarios for ToAddMod lists (MediaTek)**

**Document for: Discussion, decision**

1 Introduction

This document summarises the following email discussion launched after RAN2#111-e:

* [Post111-e][901][NR16] Extension scenarios for ToAddMod lists (Mediatek)

 Scope: Continue discussion started in AT111-e [013] based on R2-2006915. Converge and settle details.

 Intended outcome: Agreeable CR or Report or both

 Deadline: long

Rapporteur proposes to structure the discussion in two phases:

1. Feedback on the included questions and the original text proposal from [1] (comments due 1 October 2020 0700 UTC)
2. CR updated to reflect the comments from Phase 1 (deadline 15 October 2020 0700 UTC)

2 Discussion

## 2.1 Extension practices

As discussed in [1], there are several different cases of list extension to be considered:

|  |
| --- |
| Case A: The max size of the list is increased, but no new fields are added to the list itemsCase B: The max size of the list remains, but new fields are added to the list item B1: It is possible to add the new fields directly in the list item B2: It is not possible to add the fields directly in the list itemCase C: The max size of the list increases and new fields are added to the list items C1: It is possible to add the new fields directly in the list item C2: It is not possible to add the fields directly in the list item |

It was proposed in [1] to adopt the following practices for handling these cases:

**Table 1: Proposed extension practices**

|  |  |  |
| --- | --- | --- |
| Case | Description | Extension practice |
| A | List size extended, no change to elements | Non-critical extension |
| B1 | Item extension only, with extension markers | Use the extension marker if size is not critical, otherwise follow case B2 |
| B2 | Item extension only, without extension markers | New structure for the new fields, parallel list of the new structure |
| C1 | List size extended, item extended, with extension markers | Non-critical extension and use the extension marker if size is not critical; otherwise follow case C2 |
| C2 | List size extended, item extended, without extension markers | Non-critical extension of the list *without* the new fields, and parallel list (parallel to the combination of the original and extension lists) of new structures for the new fields |

**Q1: Are the proposed extension practices in Table 1 agreeable?**

|  |  |  |
| --- | --- | --- |
| **Company** | **Yes/No** | **Comment** |
| MediaTek | Yes | We understand that the main potential issue here is the use of critical vs. non-critical extension mechanisms. As described in [1], we don’t really see an advantage to using the critical mechanism, and at RAN2#110-e we seem to have implicitly settled on preferring the non-critical mechanism. |

It was further proposed in [1] to deprecate the critical extension mechanism for ToAddMod lists. Based on comments received on this proposal, there may be a preference to use slightly weaker language, e.g. “discouraged” instead of “deprecated”.

**Q2: Is it agreeable to discourage the use of the critical extension mechanism for ToAddMod lists?**

|  |  |  |
| --- | --- | --- |
| **Company** | **Yes/No** | **Comment** |
| MediaTek | Yes |  |

2.2 Documenting list relationships

In the event that the critical extension mechanism is used in the future, [1] suggested using the language “Network does not configure *listX* and *listX-rY* simultaneously to a UE” in the field description, and further clarifying that in case the UE is configured with *listX* initially, and later with *listX-rY*, the network should explicitly release the contents of *listX*. The text proposal suggested the following language in section A.4.2:

If the critical extension mechanism for a list is used, it should be clarified in the field description that the two versions of the list are not configured together, and that the network should release the contents of the original version when configuring the replacement version.

A comment was received in the earlier email discussion indicating that the language could be clarified, e.g. to indicate that the network should use the ToRelease list and state something like "The network does not include xxxToAddModList-rN (respectively xxxToAddModList without suffix) in this <IE name, e..BWP, serving cell, etc> as long as there are Xxx's configured in this <IE name> using xxxToAddModList without suffix (respectively xxxToAddModList-rN)”.

Rapporteur tends to prefer the more succinct form “Network does not configure xxxToAddModList (without suffix) and xxxToAddModList-rN simultaneously to a UE”, which could be captured explicitly in the language of the annex. Additional clarification e.g. regarding the use of the ToRelease list could be added to the TP, but since this is a discouraged example, it may be better not to go into too much detail.

**Q3: Is the language above (with the addition of “Network does not configure…”) agreeable as a guideline for the case that the critical extension mechanism is used for a ToAddMod list?**

|  |  |  |
| --- | --- | --- |
| **Company** | **Yes/No** | **Comment** |
| MediaTek | Yes | As noted above, we tend to prefer the shorter description, but OK to follow the majority view on the wording. |

For the non-critical case used to extend the length of a list, [1] suggested using the language “The UE shall consider entries in *listX* and *listX2-rY* as a single list”, along with an indication that this means entries added by one list can be modified by the other, or removed by any extension of the corresponding ToRelease list. (The exact nomenclature of the fields can be discussed separately below.) The text proposal suggested the following language to be captured in Annex A.4 of TS 38.331:

The field description table should indicate that the UE considers the original list and the extension list as a single list; thus entries added with the original list can be modified by the extension list (or removed by the extension of the ToRelease list), or vice versa.

A comment was received during the earlier email discussion to the effect that it would be better to include the full UE requirement in each field description (i.e. including the indication that an entry created with one list can be modified with the other, or deleted with the secondary ToRelease list). This could be captured in the guideline (e.g. by replacing “; thus” with “, and that”), but it results in more verbose field descriptions and CR authors may not be perfectly reliable in following the guideline.

**Q4: Is the language above (potentially with the change mentioned in the last paragraph) agreeable as a guideline for the case that the non-critical extension mechanism is used to extend the length of a ToAddMod list?**

|  |  |  |
| --- | --- | --- |
| **Company** | **Yes/No** | **Comment** |
| MediaTek | Yes | Slight preference for the current form with “thus”, because it clarifies that this principle is applicable for all such cases of list extensions, even if the field description does not capture it explicitly. If we make the change to say “and that” as suggested, then we depend on CR authors to remember to include this language always—if someone forgets to do it, it could create uncertainty about whether the affected list has a special behaviour. |

2.3 Field nomenclature

The current status of 38.331 generally uses the following conventions for naming ToAddMod list extensions (note that these are not the same as used in 36.331):

* When a list is non-critically extended, the new list has a “2” at the end of the name (before the -r16 suffix), e.g. *spatialRelationInfoToAddModList2-r16*
	+ The numbering could be continued, e.g. if we need *spatialRelationInfoToAddModList3-r17*
* When a new structure is created to hold the new fields of a list item, the new structure has “Ext” at the end of the name (before the -r16 suffix), e.g. *SearchSpaceExt-r16*
	+ The corresponding parallel list similarly has “Ext” at the end, e.g. *commonSearchSpaceListExt-r16*

In [1], it was proposed to codify these naming practices. Some comments received offline suggested that it would be preferable to align with 36.331; however, to do this consistently would require alignment of existing field names in 38.331. (Since name changes are backward compatible, this could be done without breaking the ASN.1 freeze.)

**Q5: Are the naming conventions above agreeable as a guideline for extensions in 38.331? If not, please indicate a preferred alternative approach.**

|  |  |  |
| --- | --- | --- |
| **Company** | **Yes/No** | **Comment** |
| MediaTek | Yes | We already have some divergence within 38.331. In most places the “Ext” suffix is used as suggested here, but in the case of candidateBeamRSListExt-v1610, it is used for extending the number of entries of the list (like the LTE convention). We could consider changing the name of this outlier field for consistency within 38.331. It seems not as critical to have alignment between 38.331 and 36.331, and it would be a bit disruptive to go through and change multiple field names in 38.331. |

2.4 Additional comments

**Q6: Any further comments on the general issues of ToAddMod list extensions?**

|  |  |
| --- | --- |
| **Company** | **Comment** |
|  |  |

The text proposal from [1] is included in section 5 below. Comments on the detailed language are invited. Some comments were received during the offline discussion of [1] as follows:

1. In the critical extension case, a single ToRelease list could apply to entries configured with either of the ToAddMod lists, which may cause confusion.
	* Rapporteur agrees with the principle, but this may be difficult to clarify in the text. Comments are invited.
2. The proposed guidelines suggest that in general, ToAddMod list entries should be extensible, and this principle could be captured in the annex (while acknowledging that entries need to be considered case by case, e.g. because overhead may be a concern).
	* Rapporteur agrees and this could be captured in a revision of the TP.
3. Examples could be numbered.
	* Rapporteur thinks this is reasonable, and it should be enough to include a number in the ASN.1 comments in the examples.
4. A question was raised on whether the third example is currently used anywhere in 38.331.
	* Rapporteur understands that it is used for the extension of the *spatialRelationInfoToAddModList* in *PUCCH-Config*.
5. A question was raised on whether the *originalToReleaseListExt-rN* and the *originalToReleaseList2-rN* are both necessary in the third example, and whether the elements of the *originalToReleaseList2-rN* should be of type *ListElementId-rN* rather than *ListElementId*.
	* Rapporteur thinks further discussion may be needed here.
	* The *originalToReleaseList2-rN* cannot contain the full number of entries of the combined list, so in case the list was populated with the *originalToAddModListExt-rN*, it seems necessary to have the *originalToReleaseListExt-rN* to release these entries. However, it also seems possible in the ASN.1 to populate the list with a combination of the *originalToAddModList* and the *originalToAddModList2-rN* (i.e. using the old *ListElementId* and not using the *originalToAddModListExt-rN*), and in this case the *originalToReleaseList2-rN* (with elements of type *ListElementId*) would be needed. Whether this scenario would be practical depends on the semantics of the particular extended structure, and we may need to discuss what to capture in the example.
	* In the case of the *spatialRelationInfoToAddModList*, we followed the same structure that is currently in the TP.

**Q7: Any comments on the above points?**

|  |  |
| --- | --- |
| **Company** | **Comment** |
|  |  |

Any further comments on the body of the TP are invited.

3 Conclusion

[To be populated]

4 References

[1] R2-2006915, “Extension scenarios for ToAddMod lists”, MediaTek Inc., RAN2#111-e

5 Text proposal from [1]

## A.4.2 Critical extension of messages and fields

The mechanisms to critically extend a message are defined in A.3.3. There are both "outer branch" and "inner branch" mechanisms available. The "outer branch" consists of a CHOICE having the name *criticalExtensions*, with two values, *c1* and *criticalExtensionsFuture*. The *criticalExtensionsFuture* branch consists of an empty SEQUENCE, while the c1 branch contains the "inner branch" mechanism.

The "inner branch" structure is a CHOICE with values of the form "*MessageName-rX-IEs*" (e.g., "*RRCConnectionReconfiguration-r8-IEs*") or "*spareX*", with the spare values having type NULL. The "-rX-IEs" structures contain the *complete* structure of the message IEs for the appropriate release; i.e., the critical extension branch for the Rel-10 version of a message includes all Rel-8 and Rel-9 fields (that are not obviated in the later version), rather than containing only the additional Rel-10 fields.

The following guidelines may be used when deciding which mechanism to introduce for a particular message, i.e. only an 'outer branch', or an 'outer branch' in combination with an 'inner branch' including a certain number of spares:

- For certain messages, e.g. initial uplink messages, messages transmitted on a broadcast channel, critical extension may not be applicable.

- An outer branch may be sufficient for messages not including any fields.

- The number of spares within inner branch should reflect the likelihood that the message will be critically extended in future releases (since each release with a critical extension for the message consumes one of the spare values). The estimation of the critical extension likelihood may be based on the number, size and changeability of the fields included in the message.

- In messages where an inner branch extension mechanism is available, all spare values of the inner branch should be used before any critical extensions are added using the outer branch.

The following example illustrates the use of the critical extension mechanism by showing the ASN.1 of the original and of a later release

-- /example/ ASN1START -- Original release

RRCMessage ::= SEQUENCE {

 rrc-TransactionIdentifier RRC-TransactionIdentifier,

 criticalExtensions CHOICE {

 c1 CHOICE{

 rrcMessage-r8 RRCMessage-r8-IEs,

 spare3 NULL, spare2 NULL, spare1 NULL

 },

 criticalExtensionsFuture SEQUENCE {}

 }

}

-- ASN1STOP

-- /example/ ASN1START -- Later release

RRCMessage ::= SEQUENCE {

 rrc-TransactionIdentifier RRC-TransactionIdentifier,

 criticalExtensions CHOICE {

 c1 CHOICE{

 rrcMessage-r8 RRCMessage-r8-IEs,

 rrcMessage-r10 RRCMessage-r10-IEs,

 rrcMessage-r11 RRCMessage-r11-IEs,

 rrcMessage-r14 RRCMessage-r14-IEs

 },

 later CHOICE {

 c2 CHOICE{

 rrcMessage-r16 RRCMessage-r16-IEs,

 spare7 NULL, spare6 NULL, spare5 NULL, spare4 NULL,

 spare3 NULL, spare2 NULL, spare1 NULL

 },

 criticalExtensionsFuture SEQUENCE {}

 }

 }

}

-- ASN1STOP

It is important to note that critical extensions may also be used at the level of individual fields i.e. a field may be replaced by a critically extended version. When sending the extended version, the original version may also be included (e.g. original field is mandatory, E-UTRAN is unaware if UE supports the extended version). In such cases, a UE supporting both versions may be required to ignore the original field. The following example illustrates the use of the critical extension mechanism by showing the ASN.1 of the original and of a later release.

-- /example/ ASN1START -- Original release

RRCMessage ::= SEQUENCE {

 rrc-TransactionIdentifier RRC-TransactionIdentifier,

 criticalExtensions CHOICE {

 c1 CHOICE{

 rrcMessage-r8 RRCMessage-r8-IEs,

 spare3 NULL, spare2 NULL, spare1 NULL

 },

 criticalExtensionsFuture SEQUENCE {}

 }

}

RRCMessage-rN-IEs ::= SEQUENCE {

 field1-rN ENUMERATED {

 value1, value2, value3, value4} OPTIONAL, -- Need N

 field2-rN InformationElement2-rN OPTIONAL, -- Need N

 nonCriticalExtension RRCConnectionReconfiguration-vMxy-IEs OPTIONAL

}

RRCConnectionReconfiguration-vMxy-IEs ::= SEQUENCE {

 field2-rM InformationElement2-rM OPTIONAL, -- Cond NoField2rN

 nonCriticalExtension SEQUENCE {} OPTIONAL

}

-- ASN1STOP

| Conditional presence | Explanation |
| --- | --- |
| *NoField2rN* | The field is optionally present, need N, if field2-rN is absent. Otherwise the field is absent |

Finally, it is noted that a critical extension may be introduced in the same release as the one in which the original field was introduced e.g. to correct an essential ASN.1 error. In such cases a UE capability may be introduced, to assist the network in deciding whether or not to use the critical extension.

In the case of list fields (SEQUENCE OF types in ASN.1) using the ToAddMod/ToRelease construction, the use of critical extensions to increase the size of a list should be avoided; that is, the following example is not recommended:

-- /example/ ASN1START -- Deprecated example

ContainingStructure ::= SEQUENCE {

 originalToAddModList SEQUENCE (SIZE (1..originalMaxSize)) OF ListElementType OPTIONAL, -- Need N

 ...

 [[

 replacementToAddModList-rN SEQUENCE (SIZE (1..newMaxSize)) OF ListElementType OPTIONAL -- Need N

 ]]

}

-- ASN1STOP

Preferentially, a non-critical list extension mechanism should be used instead, such that only the new entries of the list are added as a new field. This approach is further discussed in section A.4.3.x.

If the critical extension mechanism for a list is used, it should be clarified in the field description that the two versions of the list are not configured together, and that the network should release the contents of the original version when configuring the replacement version.

## A.4.3 Non-critical extension of messages

### […]

### A.4.3.x Non-critical extensions of lists with ToAddMod/ToRelease

When the length of a list using the ToAddMod/ToRelease construction is extended and/or fields are added to the list element structure, the list should be non-critically extended, i.e. by adding only the new entries to the list, coupled with the use of a parallel list structure to contain any fields added to the list elements. The following general principles apply:

– When the length of the list is extended, this is reflected in a non-critical extension of the list, with a number added sequentially to the end of the field name (before any -rN suffix). A new ToRelease list is generally needed. The field description table should indicate that the UE considers the original list and the extension list as a single list; thus entries added with the original list can be modified by the extension list (or removed by the extension of the ToRelease list), or vice versa. The result is as shown in the following example:

-- /example/ ASN1START

ContainingStructure ::= SEQUENCE {

 originalToAddModList SEQUENCE (SIZE (1..originalMaxSize)) OF ListElementType OPTIONAL, -- Need N

 originalToReleaseList SEQUENCE (SIZE (1..originalMaxSize)) OF ListElementId OPTIONAL, -- Need N

 ...,

 [[

 -- Non-critical extension lists

 originalToAddModList2-rN SEQUENCE (SIZE (1..numAdditionalElements-rN)) OF ListElementType OPTIONAL, -- Need N

 originalToReleaseList2-rN SEQUENCE (SIZE (1..numAdditionalElements-rN)) OF ListElementId OPTIONAL -- Need N

 ]]

}

-- ASN1STOP

– When fields are added to the list element structure, an extension marker should normally be used if available. If no extension marker is available or if overhead or other considerations prevent using the extension marker, an extension structure should be created for the new fields, with the suffix “Ext” added to the end of the field name (before any -rN suffix), and a parallel list introduced to hold the new structures, also with the “Ext” suffix. The field description table should indicate that the parallel list contains the same number of entries, and in the same order, as the original list. No new ToRelease list is typically needed (unless the list element ID type changes). The result is as shown in the following example:

-- /example/ ASN1START

ContainingStructure ::= SEQUENCE {

 originalToAddModList SEQUENCE (SIZE (1..originalMaxSize)) OF ListElementType OPTIONAL, -- Need N

 originalToReleaseList SEQUENCE (SIZE (1..originalMaxSize)) OF ListElementId OPTIONAL, -- Need N

 ...,

 [[

 -- Parallel list

 originalToAddModListExt-rN SEQUENCE (SIZE (1..originalMaxSize)) OF ListElementTypeExt-rN OPTIONAL -- Need N

 ]]

}

ListElementType ::= SEQUENCE {

 elementId ListElementId,

 field1 INTEGER (0..3),

 field2 ENUMERATED { value1, value2, value3 }

}

ListElementTypeExt-rN ::= SEQUENCE {

 field3 BIT STRING (SIZE(8))

}

-- ASN1STOP

– When the length of a list is extended *and* fields are added to the list element structure, an extension marker should normally be used for the added fields if available, and the list extended with the non-critical mechanism as described above. If no extension marker is available or if overhead or other considerations prevent using the extension marker, the list should be non-critically extended to hold the new entries, and a second list parallel to the concatenation of the original and extended lists should be introduced to hold the new entries (similar to the second example above). Finally, an extension structure should be created for the new fields (as in the second example above). The field description table should indicate that the parallel list contains the same number of entries, and in the same order, as the concatenation of the original list and the extension list. An extended ToRelease list is generally needed; in addition, if the element ID type changes, a second, parallel ToRelease list would be needed. The result is as shown in the following example:

-- /example/ ASN1START

ContainingStructure ::= SEQUENCE {

 originalToAddModList SEQUENCE (SIZE (1..originalMaxSize)) OF ListElementType OPTIONAL, -- Need N

 originalToReleaseList SEQUENCE (SIZE (1..originalMaxSize)) OF ListElementId OPTIONAL, -- Need N

 ...,

 [[

 -- Non-critical extension lists

 originalToAddModList2-rN SEQUENCE (SIZE (1..numAdditionalElements-rN)) OF ListElementType OPTIONAL, -- Need N

 originalToReleaseList2-rN SEQUENCE (SIZE (1..numAdditionalElements-rN)) OF ListElementId OPTIONAL, -- Need N

 -- Parallel lists with newMaxSize = originalMaxSize + numAdditionalElements

 originalToAddModListExt-rN SEQUENCE (SIZE (1..newMaxSize-rN)) OF ListElementTypeExt-rN OPTIONAL, -- Need N

 originalToReleaseListExt-rN SEQUENCE (SIZE (1..newMaxSize-rN)) OF ListElementId-rN OPTIONAL -- Need N

 ]]

}

ListElementType ::= SEQUENCE {

 elementId ListElementId,

 field1 INTEGER (0..3),

 field2 ENUMERATED { value1, value2, value3 }

}

ListElementTypeExt-rN ::= SEQUENCE {

 elementId-rN ListElementId-rN,

 field3-rN BIT STRING (SIZE(8))

}

-- ASN1STOP