# TSG-RAN Meeting #12 *RP-010318*
# Stockholm, Sweden, 12 - 15 June 2001

**Title:** **Agreed CRs (Release '99 and Rel-4 category A) to TR 25.921**

**Source:** **TSG-RAN WG2**

**Agenda item:** **8.2.3**

| Doc-1st- | Status- | Spec | CR | Rev | Phase | Subject | Cat | Version | Versio |
|---|---|---|---|---|---|---|---|---|---|
| R2-011072 | agreed | 25.921 | 014 | | R99 | Clean up | F | 3.3.0 | 3.4.0 |
| R2-011331 | agreed | 25.921 | 015 | | Rel-4 | Clean up | A | 4.0.0 | 4.1.0 |
| R2-011149 | agreed | 25.921 | 016 | | R99 | Usage of spare values in future releases | F | 3.3.0 | 3.4.0 |
| R2-011329 | agreed | 25.921 | 017 | | Rel-4 | Usage of spare values in future releases | A | 4.0.0 | 4.1.0 |
| R2-011457 | agreed | 25.921 | 018 | 1 | R99 | Structure and naming of extensions in ASN.1 | F | 3.3.0 | 3.4.0 |
| R2-011458 | agreed | 25.921 | 019 | | Rel-4 | Structure and naming of extensions in ASN.1 | A | 4.0.0 | 4.1.0 |
| R2-011485 | agreed | 25.921 | 020 | | R99 | Addition of Recommendations for Extensions in RANAP, RNSAP, NBAP, and SABP | F | 3.3.0 | 3.4.0 |
| R2-011486 | agreed | 25.921 | 021 | | Rel-4 | Addition of Recommendations for Extensions in RANAP, RNSAP, NBAP, and SABP | A | 4.0.0 | 4.1.0 |
| R2-011487 | agreed | 25.921 | 022 | | R99 | Clean-up with regard to RAN WG3 Practise of Specifying Control Plane Protocols | F | 3.3.0 | 3.4.0 |
| R2-011488 | agreed | 25.921 | 023 | | Rel-4 | Clean-up with regard to RAN WG3 Practise of Specifying Control Plane Protocols | A | 4.0.0 | 4.1.0 |

**3GPP TSG-RAN WG2 Meeting #21**
**Busan, Korea, May 21st-25th 2001**

***Tdoc R2-011072***

*CR-Form-v3*

# CHANGE REQUEST

⌘ **25.921** **CR** **014** ⌘ rev **-** ⌘ Current version: **3.3.0** ⌘

*For **HELP** on using this form, see bottom of this page or look at the pop-up text over the ⌘ symbols.*

***Proposed change affects:*** ⌘ (U)SIM ☐ ME/UE **X** Radio Access Network **X** Core Network ☐

| | | |
|---|---|---|
| ***Title:*** | ⌘ | Clean-up |
| ***Source:*** | ⌘ | TSG-RAN WG2 |
| ***Work item code:*** | ⌘ | TEI |
| ***Date:*** | ⌘ | 2001-05-15 |
| ***Category:*** | ⌘ | **F** |
| ***Release:*** | ⌘ | R99 |

*Use one of the following categories:*
***F*** *(essential correction)*
***A*** *(corresponds to a correction 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.

*Use one of the following releases:*

| | |
|---|---|
| *2* | *(GSM Phase 2)* |
| *R96* | *(Release 1996)* |
| *R97* | *(Release 1997)* |
| *R98* | *(Release 1998)* |
| *R99* | *(Release 1999)* |
| *REL-4* | *(Release 4)* |
| *REL-5* | *(Release 5)* |

***Reason for change:*** ⌘

Inconsistency between the guidelines and how they are applied
The guidelines are not in-line with the current RAN2 specifications (essentially the RRC specification). Several options and guidelines are out of date, are misleading, and need thus to be corrected or removed.

Differences between RAN2 and RAN3 approaches
In RAN2 and RAN3 sometimes slightly different approaches are made, while the guidelines mainly reflects the RAN2 approach. Therefore there is a need for more generalised guidelines, but still leave room to document how a guideline is specialised in RAN2 and RAN3. Note: A separate CR proposes changes from RAN3 perspective.

Summary of change: ⌘

- Clause 1, "Scope" is updated to reflect the scope of the document, which is control-plane protocols specified in 3GPP TSG-RAN such as RRC, RANAP etc. Some wording changed as well.
- In clause 2, "References", the "3GPP drafting rules" are corrected to be TR 21.801 (Rel-4) with the name "Specification drafting rules".
- Numerous clarifications in Clause 4, "Principles to ensure compatibility":
  Chapter 4.2, Level 1 of principles: Protocol level
  The chapter states that it "shall be possible to discriminate different versions of any protocol". It is changed to indicate the purpose "it shall be possible to inter-work different versions of a protocol specification".
  Also, the chapter seems to suggest a "protocol discriminator" but the purpose can be achieved by other means, which means the sentence can be removed.
  Chapter 4.3.1, New messages
  The principle is rather "the protocols shall specify a mechanism such that new messages can be introduced without causing harm".
  Chapter 4.3.2, Partial decoding
  The chapter is unclear.
  It is changed to describe that "extensions are allowed in a compatible way".

Chapter 4.4.2, Optional IE
To put optional IEs after mandatory ones is a too strong recommendation since it depends on the transfer syntax and the practice is not used. The subclause is removed.
Chapter 4.4.3, Adding mandatory IE
The first paragraph is ambiguous and is clarified. The subclause also mentions UE, which is Uu specific and that sentence can therefore be removed.
Chapter 4.4.4, Missing optional IE
Corrected to be "Absent optional IE" since "missing" is confusing.
Chapter 4.4.5, Comprehension required
The content of the chapter may be interpreted to specify the "ASN.1 comprehension required" mechanism as a principle. Instead the contents is changed to describe the principle of using "criticality" information.
Chapter 4.4.6, Partial Decoding
There is not a requirement to use partial decoding and therefore it is changed to permission ("may").
Chapter 4.5.1, Reserved values and spare fields
The text is inconsistent with the rest of the TR and therefore changed.
Chapter 4.5.3, Missing optional value
This chapter is irrelevant (not used) and is therefore removed.

- Clause 7, "Protocol procedure specification rules" has been updated.
The general part is modified to reflect what really is general for the RAN2 and RAN3 specifications.
The RRC part describes now how these general rules are applied in the RRC case. Protocol specific guidelines that do not need to be visible in the TR should be described in each protocol specification.
A subclause on specification of algorithms and formulas has been added. The operators div and mod are defined.
The DS-41 part is moved to an informative annex since they are out-of-date and no replacement is proposed in this CR.

- Some clarifications to clause 8, "Messages".
Subclause 8.1 "Summary what has been agreed" is removed, since the content is already covered in other parts of the TR.
Chapter 8.2, Definitions
The terms "message contents description" and "message encoding" are clarified by connecting them to the terms and "abstract syntax" and "transfer syntax".
Chapter 8.3, Logical description
The reference to clause 8 should be clause 9
Chapter 8.5, Compilability of the transfer syntax
The reference to CSN.1 is very specific and is changed to indicate that "specialised encoding" is allowed.
Chapter 8.7, Evolvability/Extensibility
The sentence "The transfer syntax shall keep the same level of compactness as the initial design" is ambiguous, formally incorrect and is thus removed.

- Numerous corrections to clause 9 "Usage of tabular format".
Chapter 9.1.1.2, The Information Element table
A heading is missing and is added.
The content of the type and reference column is not using any indentation.
The number of columns of the tabular format shall be 6 (not 65).
Chapter 9.1.1.2.1.1 Mandatory
The handling of Multi column for mandatory IEs is corrected.
Chapter 9.1.1.2.1.2 Optional
The handling of Multi column for optional IEs is corrected.
Chapter 9.1.1.2.1.3 Conditional
The handling of Multi column for conditional IEs is corrected.
It is also clarified that also if a condition is not met may result in a presence or absence requirement for an IE. A sentence of error handling of conditions is added.

Chapter 9.1.1.2.1.4 Choice
The alternative of filling the need column is missing and is added.
Chapter 9.1.1.2.1.5 Set
The text refers to “copies” of IEs, which is unclear and therefore corrected to be “instances”.
The handling of Need column is not correct (should be filled).
Chapter 9.1.1.2.2, Type and reference column
The bullet “A reference to a subclause of this document…”, is removed since the 25.921 predefined types are not used in message descriptions, only in IE type descriptions.
Chapter 9.1.1.2.4, Expressing differences between FDD and TDD modes
The example tabular format is corrected. It should be “mode” not “system type” and the need column for the choice should be filled.

- Chapter 9.1.1.3, Explanatory clauses
  Symbolic names are not part of the explanatory causes in RRC (but in the WG3 specifications) and that recommendation is removed.
- Chapter 9.1.2, IE type description
  It is clarified that explanatory clauses should be used also for the IE type description. It is added that the basic types defined in 25.921 may be used and for these no reference is necessary, since they should be regarded as pre-defined. A recommendation is added saying that multiple definitions of the same type should be avoided.
- The subclauses 10.1-10.3 contain merely an overview of ASN.1 and where any guidelines are given, they are not in-line with how ASN.1 is used in the specifications. Therefore the text in the subclauses is moved to an informative annex. No replacing text is proposed by now.
- A rule of not using user-defined constraints is added to the beginning of clause 10. The references to the sections 10.1-10.3 that are moved to an annex are removed.

| | | |
|---|---|---|
| ***Consequences if not approved:*** | ⌘ | The guidelines can not be applied to the specifications and if they are anyway applied there is a risk that further updates of the specifications will be made in an undesirable way. |

| | | | | |
|---|---|---|---|---|
| ***Clauses affected:*** | ⌘ | 1, 2, 4.2, 4.3.1, 4.3.2, 4.4.2, 4.4.3, 4.4.4, 4.4.5, 4.4.6, 4.5.3, 7.1, 7.1a (new), 7.2, 7.3, 8.1, 8.2, 8.3, 8.5, 8.7, 9.1.1.2, 9.1.1.2.a (new), 9.1.1.2.1, 9.1.1.2.1.1, 9.1.1.2.1.2, 9.1.1.2.1.3, 9.1.1.2.1.4, 9.1.1.2.1.5, 9.1.1.2.2, 9.1.1.2.4, 9.1.1.3, 9.1.2, 10, 10.1.1, 10.1.2, 10.1.3, 10.1.4, 10.2, 10.2.1, 10.2.2, 10.2.3, 10.2.4, 10.2.5, 10.2.6, 10.3, 10.3.1, 10.3.2, 10.3.3, 10.3.4, 10.3.5, 10.3.6, 10.3.7, 10.3.8, 10.3.9, 10.3.10, 10.3.11, 10.3.12, 10.3.13, 10.3.14, Annex A, Annex B (new) | | |
| ***Other specs affected:*** | ⌘ | ☐ Other core specifications<br>☐ Test specifications<br>☐ O&M Specifications | ⌘ | |
| ***Other comments:*** | ⌘ | | | |

**How to create CRs using this form:**
Comprehensive information and tips about how to create CRs can be found at: http://www.3gpp.org/3G_Specs/CRs.htm. Below is a brief summary:

1) Fill out the above form. The symbols above marked ⌘ contain pop-up help information about the field that they are closest to.
2) Obtain the latest version for the release of the specification to which the change is proposed. Use the MS Word "revision marks" feature (also known as "track changes") when making the changes. All 3GPP specifications can be downloaded from the 3GPP server under ftp://www.3gpp.org/specs/ For the latest version, look for the directory name with the latest date e.g. 2000-09 contains the specifications resulting from the September 2000 TSG meetings.

3) With "track changes" disabled, paste the entire CR form (use CTRL-A to select it) into the specification just in front of the clause containing the first piece of changed text. Delete those parts of the specification which are not relevant to the change request.

# 1 Scope

The present document provides a guideline for ~~using formal languages in~~ protocol ~~description~~ specification of UMTS stage 2 and 3 including the usage of formal languages and rules for error handling. This document covers control-plane protocols specified in TSG-RAN such as RRC, RANAP, RNSAP, NBAP and SABP. ~~all interfaces involved in radio access protocols such as Uu, Iu, Iur and Iub.~~

# 2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication, edition number, version number, etc.) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies. In the case of a reference to a 3GPP document (including a GSM document), a non-specific reference implicitly refers to the latest version of that document *in the same Release as the present document.*


[1] ITU-T Recommendation X.680: "Abstract Syntax Notation One (ASN.1): Specification of the basic notation".

[2] ITU-T Recommendation X.681: "Abstract Syntax Notation One (ASN.1): Information object specification".

[3] ITU-T Recommendation X.682: "Abstract Syntax Notation One (ASN.1): Constraint specification".

[4] ITU-T Recommendation X.690: "ASN.1 Encoding Rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)".

[5] ITU-T Recommendation X.691: "ASN.1 Encoding Rules - Specification of Packed Encoding Rules (PER)".

[6] CSN.1: "specification, version 2.0".

[7] ITU-T Recommendation Z.100: "Specification and description language (SDL)".

[8] ITU-T Recommendation Z.105: "SDL Combined with ASN.1 (SDL/ASN.1)".

[9] ITU-T Recommendation Z.120: "Message Sequence Chart (MSC)".

[10] ISO/IEC 9646-3: "The Tree and Tabular Combined Notation".

[11] 3GPP ~~TS 21.200~~TR 21.801 (Rel-4): "~~3GPP D~~Specification drafting rules".


# 3 Void

# 4 Principles to ensure compatibility

## 4.1 Introduction

The rules edicted intend to prevent incompatibilities between several phases of UMTS evolution (in analogy to what happened from GSM phase 1 to GSM phase 2).

## 4.2 Level 1 of principles: Protocol level

It shall be possible to ~~discriminate~~ inter-work different versions of any protocol specification.

An unknown protocol shall not cause problems to any entity that terminates the protocol. ~~The messages using this protocol discriminator shall be discarded by the receiving entity.~~

As a consequence, introduction of new protocol shall not disturb any receiving entity.

## 4.3 Level 2 of principles: Message level

### 4.3.1 New messages

The protocols shall specify a mechanism such that ~~N~~new message types shall be able to be introduced without causing any unexpected behaviour or damage. ~~New messages not understood shall be discarded by the receiving entity.~~

~~As an exception to this principle it can be possible to~~ The protocols may define a mechanism that allows a different behaviour, on received messages that are not understood, when a specific reaction is requested from the receiving entity. This mechanism has to be implemented from the beginning. A special care has to be taken into account when defining broadcast messages and the associated Error handling. Further refinement on this paragraph is needed.

Such a mechanism is not required inside the network part.

### 4.3.2 Partial decoding

PDU extensions are allowed in a compatible way, e.g. by utilising partial decoding or other mechanisms that allows the decoder to skip the extensions. Partial decoding means that a PDU can be decoded in parts. One part forms a complete value that can be separated from other parts. ~~A decoding error in a part does not invalidate previously decoded parts. Subsequent parts are however invalidated because if an error has occurred one cannot be sure whether the trailing values are really valid.~~

~~Example: A multipurpose PDU contains a list of four PDUs. The two first PDUs are valid but the third one is invalid. The two first are decoded but the third and fourth ones are ignored.~~

## 4.4 Level 3 of principles: Information element level

### 4.4.1 New IE

New elements shall generally be discarded when not understood.

In some cases new elements might be taken into account when specific behaviour is requested from the receiving side (e.g. a rejection of the message is expected when the element is not understood: «comprehension required»).

### 4.4.2 Void~~Optional IE~~

~~Optional IE should be located after mandatory ones.~~

### 4.4.3 Adding mandatory IE

For backward compatibility reasons, addition of mandatory IE shall be avoided. ~~In the first stage of UMTS, a set of functionality is available for each class of UE. Mandatory IE may be added only if they are mandatory for further classes of UE.~~

### 4.4.4 ~~Missing~~ Absent optional IE

~~Missing~~ Absent optional element may be understood as having a certain default value hence a defined meaning.

~~See also missing values in Values level.~~

### 4.4.5 Comprehension required

"Comprehension required" requirement can be associated with an IE or a message. It means that the IE or message is tagged with “criticality” information (explicit in the message or implicit based on the type of IE or message). Any action performed by the receiver if the IE or message is not understood (“comprehended”) is based on this “criticality” information.~~It means that after an IE value has been decoded then the value is validated according to some specified criteria. Failure in validation causes rejection of the message.~~

~~Example: A broadcast message contains a list of recipient addresses. If a recipient's address is not included in the list then a recipient ignores the whole message.~~

### 4.4.6 Partial Decoding

The notion of partial decoding ~~shall~~ may also be applied at the IE level.

## 4.5 Level 4 of principles: Values level

### 4.5.1 Reserved values and spare fields

Reserved values shall be forbidden. Otherwise entity receiving such a value shall reject the message. This would create difficulties when provided on broadcast channel.

Spare field shall be forbidden. Otherwise entity receiving such a spare field shall not make any decoding on that field and shall not reject the message.

### 4.5.2 Unspecified values

As far as possible default understanding shall be provided for unspecified values.

### 4.5.3 Void~~Missing optional value~~

~~A default value may be specified for the receiver when the sender did not include a field containing this value.~~

### 4.5.4 Extension of value set

There are cases when a data field may originally contain only a definite set of values. In the future the set of values grows but the number new values can be anticipated. There are two alternative ways to specify extension of a value set:

1) Infinite extension of a value set. Example: The first version of a data field may contain only values 0-3. In the future the field may contain any positive integer value.

2) Finite extension of a value set. Example: The first version of a data field may contain only values 0-3. In the future values 4-15 shall also be used.

# 5 Message Sequence Charts

It is agreed to recommend the use of MSCs as one of the formal methods.

MSCs are adapted for description of normal behaviour of protocol layers between peer entities and/or through SAPs. So it may be used in stage 2 of protocol description.

# 6 Specification and Description Language

The groups are encouraged to use of SDL where appropriate. The SDL code included in the standards should follow the descriptive SDL guidelines from ETSI TC-MTS (DEG MTS-00050) as closely as possible.

The groups themselves should decide how SDL is used.

In some protocol parts, text is more adapted (e.g.: algorithm or multiplexing), in some other parts SDL is better.

SDL is adapted for describing the observable behaviour of a protocol layer.

In TSG-RAN WG2, the specifications shall not use SDL for the normative part of the specifications in the present version.

# 7 Protocol procedure specification rules

## 7.1 General

- A protocol specification shall contain a 'Procedures' clause, which specifies the functional behaviour, using "procedures". A procedure is typically a sequence of events, with a start and an end, which can be observed in the protocol and/or in the interfaces to other layers (upper and/or lower layers).
- The procedure specification shall be made using text and verbal forms.
- The verbal forms, such as ~~Words~~ "shall", "should" and "may" are used in conformance with [11~~10~~] Annex E.
- The procedures should be specified in an asymmetric way, by concentrating on the behaviour on one side of the interface. As guidance, the "controlled" side, rather than the "controlling" side of the interface should be specified.
- The procedures should be specified using the externally observable behaviour, to ease writing of test specifications.
- All normal cases shall be covered. Normal cases are straightforward cases, branches of procedures and combinations of procedures.
- All error cases shall be specified, either explicitly or implicitly. The error cases are all cases that are not considered as normal cases. The error handling should be divided between error handling global to the protocol layer and procedure specific error handling. The procedure specific error cases should be put after the normal cases in each procedure.
- ~~The way to describe procedures is the following:~~

  ~~Protocol errors (global to the protocol layer).~~

  ~~Error handling (global to the protocol layer).~~

  ~~…~~

  ~~1. Procedure <Procedure Name>;~~

  ~~list of normal cases;~~

~~Protocol errors (specific to the procedure);~~

~~Error handling (specific to the procedure).~~

- Redundancy/duplication shall be avoided, in order to avoid problems with later CR, even if this makes the specification initially less readable.

- ~~Mutual cross-referencing shall be introduced: section X that is referred to in section Y should also *say* that it is referred to in section Y.~~

- States and state variables should be used when it provides unambiguity, a way to describe nested procedures and colliding cases.

- Timers, variables and constants and usage of them must be specified.

- Explicit explanation when the action shall be performed is specified in the procedure itself.

- When there are procedural differences between the FDD and TDD modes these should be clearly pointed out using a consequent notation, e.g. “FDD only”, “In TDD, …”.

- ~~The chapter "Default actions upon reception of an IE" is used to avoid duplication of text; this chapter is put at the beginning of the "Message contents to use" text.~~

- When optional ~~(or conditional)~~ IEs are possible in a given message, the meaning of the presence (i.e.: which «function» are activated with the given IE) shall be specified in the procedure for the receiving entity. The requirements on when to include a given IE shall be specified for the transmitting entity. An exception for this rule is when the requirements on the entity is not specified by the protocol.

- ~~The formal values of the IE, e.g., "TRUE" or "FALSE" rather than the coded value, e.g. "1" or "0" shall be used.~~

- Requirements on the content of a message at the sending entity are put before analysis of the message at the receiving entity.

- References to IEs that are parts of another IE is allowed. ~~The notation shall be changed to the so-called "dot-notation" for references to IEs that are parts of another IE.~~

- ~~Names of IEs shall be put between "<IE Name>" quotes, where <IE Name> is the exact name from the tabular format.~~When referring to an IE a formal notation shall be used.

- ~~Square brackets [ ] shall be used for addressing one element of a list.~~

- When referring to a message, a formal notation shall be used ~~"<MESSAGE NAME>" *message* shall be used. Message names are always in upper cases and the word message follows the message name~~.

## 7.1a Specification of algorithms and formulas

When algorithms or formulas are used in the specifications, a formal notation shall be used and mathematical expressions should be used to reduce ambiguity.

- The notation “*OP1* div *OP2*” shall be interpreted as the signed integer result after integer division (truncating any fractional part) of the operand *OP1* with operand *OP2*.

- The notation “*OP1* mod *OP2*” shall be interpreted as the signed remainder after integer division of the operand *OP1* with operand *OP2*.

## 7.2 RRC specific rules

- The specification shall focus on the UE behaviour.

- Only UE timers are normative (when UTRAN timers are present, it is for information).

- The procedure specification text shall specify how the UE shall handle the IEs.

- As much as possible of the UE behaviour shall be tied to reception and non-reception of IEs and included in the subclause "Generic actions upon receipt and absence of an information element", to avoid duplication of text.

- "UTRAN shall" shall be only used when UTRAN behaviour is normative.

- It shall be specified whether timers shall be started when RRC sends the message to lower layers or when the message is effectively sent at the radio interface.

- When referring to messages in the procedure text, the notation “EXAMPLE message” is used (excluding the quotation marks). For example: “The UE shall transmit an RRC CONNECTION REQUEST message”.

- When referring to IEs in the procedure text, the tabular description of IEs should be used as basis. The notation “IE “Example”” is used (including the inner but not the outer quotation marks. Values of IEs are out within quotation marks. For example: “The UE set the IE “Protocol error indicator” to “FALSE” in the RRC CONNECTION REQUEST message”.

- UE performance requirements are considered to be TSG RAN WG2 work. These must be specified only if they are testable.

## 7.3 VoidHandling of DS-41

- Modelling of RRC services is provided by means of primitives.

- RRC CN dependent info:

- In broadcast message, neighbour cells are described the same way as for GSM neighbour cells (i.e.: in the same SystemInformationBlock but with a tag to indicate CN type or RTT).

- In dedicated messages.

  a transparent container as NAS info is used to carry ANSI-41;

  for PLMN Id and Identities used by the RRC, the CN Type info is used;

  NAS binding info is used;

- In Paging messages, a tag to indicate CN type is used.

- Extensions like handover message to Multicarrier is handled the same way as GSM.

# 8 Message specification

## 8.1 VoidSummary of what has been agreed

1) use subset of ASN.1 (compatible with Z.105) for definition of message contents description of protocol messages;

2) there is a need for a default encoding, which can be applied in most cases;

3) there is a need for a special encoding e.g. by means of CSN.1;

4) how to link the message contents description to the different encoding rules needs to be specified;

5) ASN.1 definitions can be used within SDL and TTCN parts of the specifications.

## 8.2 Definitions

Message descriptions are divided into three levels:

- a logical description, which describes messages and relevant information elements in an easily understandable, semi-formal fashion;

- a message contents description, which describes the messages formally and completely in an abstract fashion ("abstract syntax"); and

- a message encoding, which defines the encoded messages (i.e. what is carried as a bit string, "transfer syntax").

## 8.3 Logical description

The logical description of messages shall be done using tabular format specified in clause 98 of this document, Message contents description.

## 8.4 Message contents description

The message contents descriptions shall be written using ASN.1. The message encoding shall be based on the ASN.1 description.

## 8.5 Compilability of the transfer syntax

The transfer syntax should allow as automatic as possible compilers that transform between a sequence of received bits ~~into~~ and a sequence of IEs that can be utilised by the protocol machine. ~~CSN.1~~Specialised encoding may be used. A link between message contents description and transfer syntax needs to be specified.

## 8.6 Efficiency/Compactness

The transfer syntax should allow minimising the size of messages if so necessary. It should allow protocol dependant optimisations.

## 8.7 Evolvability/Extensibility

The message contents description shall allow the evolution of the protocol.

~~The transfer syntax shall keep the same level of compactness as the initial design.~~

## 8.8 Inter IE dependency

The message contents description shall allow that presence of IEs depends on values in previous IEs.

The description of messages should avoid dependency between values in different IE. Indeed, it would mean that values are not independent and that there is a redundancy.

## 8.9 Intra IE dependency

The abstract and transfer syntaxes shall allow that, within an IE, some fields depend on previous ones.

## 8.10 Support of error handling

The syntax used should support optional IEs, default values, partial decoding, "comprehension required" and extensibility as defined above.

# 9 Usage of tabular format

A protocol specification should include a 'Tabular description' subclause, including:

- A message description subclause;

- An IE description subclause.

# 9.1 Tabular description of messages and IEs

## 9.1.1 Message description

A 'Message description' subclause includes one subclause per message.

A message is described with, in this order:

- A general description, including the flow the message belongs to (e.g., SAP, direction, …); this indirectly points to the message header description, which is not described again for each message;
- A table describing a list of information elements;
- Explanatory clauses, mainly for describing textually conditions for presence or absence of some IEs.

### 9.1.1.1 The general description

### 9.1.1.2 The Information Element table

The table is composed of 6~~5~~ columns, labelled and presented as shown below.

| IE/Group Name | Need | Multi | Type and reference | Semantics description | Version |
|---|---|---|---|---|---|
| | | | | | |

NOTE: Indentations are used to visualise the embedding level of an "IE/Group" ~~or "Type and reference"~~.

Indentations are explicitly written with the character ">", one per level of indentation. Indentations of lines can be found in the IE/Group Name ~~and Type and reference~~ column~~s~~.

Each line corresponds either to an IE or to a group. A group includes all the IEs in following lines until, and not including, a line with the same indentation as the group line.

Dummy groups can be used for legibility: the following IE/Group has the same indentation. For such dummy groups, the Need and Multi columns are meaningless and should be left empty.

~~The "Type and reference" column is not filled in the case of a group line and must be filled for "IE/Group Name" column.~~

#### 9.1.1.2.a IE/Group Name column

This column gives the local name of the IE or of a group of IEs. This name is significant only within the scope of the described message, and must appear only once in the column at the same level of indentation. It is a free text, which should be chosen to reflect the meaning of the IE or group of IEs. This text is to be used ~~followed by the key word IE, the whole enclosed between quotes [or in italics]~~ to refer to the IE or the group of IEs in the procedure ~~specificational description~~as described in clause 7..

The first word 'choice' has a particular meaning, and must not be used otherwise.

#### 9.1.1.2.1 Need and multiplicity (Multi) columns

These columns provide most of the information about the presence, absence and number of ~~copy~~instance of the IE (in the message or in the group) or group of IEs. The different possibilities for these columns are described one by one.

The meaning of the 'need' column is summarised below:

MP Mandatorily present.

A value for that information is always needed, and no information is provided about a particular default value. If ever the transfer syntax allows absence (e.g., due to extension), then absence leads to an error diagnosis.

MD Mandatory with default value.

A value for that information is always needed, and a particular default value is mentioned (in the "Semantical information" column). This opens the possibility for the transfer syntax to use absence or a special pattern to encode the default value.

CV Conditional on value.

A value for that information is needed (presence needed) or unacceptable (absence needed) when some conditions are met that can be evaluated on the sole basis of the content of the message.

If conditions for presence needed are specified, the transfer syntax must allow for the presence of the information. If the transfer syntax allows absence, absence when the conditions for presence are met leads to an error diagnosis.

If conditions for absence needed are specified, the transfer syntax must allow encoding the absence. If the information is present and the conditions for absence are met, an error is diagnosed.

When neither conditions for presence or absence are met, the information is treated as optional, as described for "OP".

CH Conditional on history.

A value for that information is needed (presence needed) or unacceptable (absence needed) when some conditions are met that must be evaluated on the basis of information obtained in the past (e.g., from messages received in the past from the other party).

If conditions for presence needed are specified, the transfer syntax must allow for the presence of the information. If the transfer syntax allows absence, absence when the conditions for presence are met leads to an error diagnosis.

If conditions for absence needed are specified, the transfer syntax must allow encoding the absence. If the information is present and the conditions for absence are met, an error is diagnosed.

When neither conditions for presence or absence are met, the information is treated as optional, as described for "OP".

OP Optional.

The presence or absence is significant and modifies the behaviour of the receiver. However whether the information is present or not does not lead to an error diagnosis.

#### 9.1.1.2.1.1 Mandatory

| IE/Group Name | Need | Multi | Type and reference | Semantics description | Version |
|---|---|---|---|---|---|
| Name | MP | | | | |
| Name | MD | | | (default value is indicated) | |

The multiplicity column ~~must~~ may be left empty (see 9.1.1.2.1.5).

For mandatory IEs, the rules are as follows, applied on the number of instances given by the multiplicity column (leaving the multiplicity column empty means one and only one instance):

For an IE not belonging to a group MP indicates that ~~one and only one copy~~the number of instances as given by the multiplicity column of "Name IE" is necessary in the message.

For a group not belonging to another group, MP means that the number of instances as given by the multiplicity column~~one and only one copy~~ of the "Name group" is necessary in the message.

For an IE or a group belonging to another group, MP means that if the parent group is present, then the number of instances as given by the multiplicity column~~one and only one copy~~ of the "Name group" or "Name IE" is necessary in the embedding group.

For an IE not belonging to a group MD indicates that the number of instances as given by the multiplicity column ~~one and only one value~~ for information 'Name IE' is necessary in the message, and that a special value (the default value) exists, for all instances or individual instances, and is mentioned in the 'Semantics description' column.

For a group not belonging to another group, MD means that the number of instances as given by the multiplicity column ~~one and only one value~~ for information structure 'Name group' is necessary in the message, and that a special value (the default value) exists, for all instances or individual instances, and is mentioned in the 'Semantics description' column.

For an IE or a group belonging to another group, MD means that if the parent group is present, then the number of instances as given by the multiplicity column~~one and only one value~~ for information structure 'Name group' or information 'Name IE' is necessary in the embedding group, and that a special value (the default value) exists and is mentioned in the 'Semantics description' column.

The default value might be fixed by the standard, or conditional to the value of some other IE or IEs, or conditional on information obtained in the past.

#### 9.1.1.2.1.2 Optional

| IE/Group Name | Need | Multi | Type and reference | Semantics description | Version |
|---|---|---|---|---|---|
| Name | OP | | | | |

The multiplicity column ~~is~~ may be empty (see 9.1.1.2.1.5).

This indicates that the number of instances as given by the multiplicity column of the 'Name IE' or 'Name group' is not necessary in the message or the embedding group, and that the sender can choose not to include it.

#### 9.1.1.2.1.3 Conditional

| IE/Group Name | Need | Multi | Type and reference | Semantics description | Version |
|---|---|---|---|---|---|
| | CV *cond* | | | | |
| | CH | | | | |

The multiplicity column ~~is~~ may be empty (see 9.1.1.2.1.5).

CV indicates that the requirement for presence o~~f~~ absence of the number of instances as given by the multiplicity column of the IE or group of IE depends on the value of some other IE or IEs, and/or on the message flow (e.g., channel, SAP). In the CV case, the condition is to be described in a textual form in an explanatory clause. *cond* stands for a free text that is used as a reference in the title of the explanatory clause. In the CH case, the condition is described in the procedural section.

~~When condition is met may means~~ The result of evaluating the condition (if the condition is met or not) may mean that the IE is:

- Mandatorily present.
- Mandatorily absent.
- Optional.
- Absent, but optional (this is meaningful only for extension).

The error handling shall be specified in the protocol for the cases when the requirement for presence or absence of an IE indicated by the condition is not followed.

#### 9.1.1.2.1.4 Choice

This is particular group of at least two children.

| IE/Group Name | Need | Multi | Type and reference | Semantics description | Version |
|---|---|---|---|---|---|
| Choice *name* | NOTE 1 | | | | |
| *>Name1* | | | | | |
| *>Name2* | | | | | |

NOTE 1: The Need column shall take one of the values “MP”, “MD”, “OP”, “CV *cond*” or “CH *cond*”.

A 'choice' group is distinguished from standard groups by the use of 'choice' as first word in the name.

The Need column shall and the Multi columns ~~are~~ may be filled ~~normally~~ for the group line. They are not filled for the children lines: the implicit value is conditional, one condition being that one and only one of the children is present if the group is present.

If additional conditions (depending on the value of some other IE or IEs, and/or on the message flow) exist for the choice, they are explained in an explanatory clause.

#### 9.1.1.2.1.5 Sets

In general, this indicates that more than one ~~copy~~instance of an IE/Group might be necessary in the message.

The two lines below indicate different allowed alternatives.

| IE/Group Name | Need | Multi | Type and reference | Semantics description | Version |
|---|---|---|---|---|---|
| Name | MP | nn..pp | | | |
| Name | MP | nn..indefinite | | | |
| Name | MP | nn..sym2 | | | |
| Name | MP | sym1..pp | | | |
| Name | MP | sym1..sym2 | | | |
| Name | MD | nn..pp | | | |
| Name | MD | nn..indefinite | | | |
| Name | MD | nn..sym2 | | | |
| Name | MD | sym1..pp | | | |
| Name | MD | sym1..sym2 | | | |
| Name | OP | nn..pp | | | |
| Name | OP | nn..indefinite | | | |
| Name | OP | nn..sym2 | | | |
| Name | OP | sym1..pp | | | |
| Name | OP | sym1..sym2 | | | |
| Name | Cx cond | nn ..pp | | | |
| Name | Cx cond | nn..indefinite | | | |
| Name | Cx cond | nn..sym2 | | | |
| Name | Cx cond | sym1..pp | | | |
| Name | Cx cond | sym1..sym2 | | | |

Where *nn* and *pp* stand for positive integers, and *sym1* and *sym2* for symbolic names. The Need column can be empty, CV or CH.

The notation '..' can be replaced with the same meaning by 'to'.

This indicates that a number of ~~copies~~instances of the IE/Group are necessary in the message/embedding group. The order is significant. The reference should use the bracket notation (e.g., 'Name[1] IE') to refer to a specific ~~copy~~instance; numbering starts by 1.

The *nn..pp* case indicates that the number of ~~copies~~instances is between nn and pp, inclusively. This means that *nn* ~~copies~~instances are necessary in the message, that additional *pp-nn* ~~copies~~instances are optional and meaningful, and that ~~copies~~instances after the *pp*th are not necessary.

The number *nn* is positive or null. The number *pp* must be equal or greater than *nn*. The 1..1 case should be avoided and ~~,~~ instead the Multi column should be left empty to indicate one and only one instance. ~~and a MP indication used instead~~. ~~Similarly, t~~The 0..1 case combined with MP should be avoided and replaced by 1..1 with an OP indication.

The *nn*..indefinite case indicates that the number of ~~copies~~instances is *nn* or greater. This means that *nn* ~~copies~~instances are necessary in the message, and that additional ~~copies~~instances are optional and meaningful. The number *nn* is positive or null. It is however allowed that the transfer syntax puts some practical limits on the maximum number of ~~copies~~instances.

The use of a symbolic name for one or the other of the range bounds indicates that the value is given in a textual clause. This is necessary the case when the bound depends is conditional to the value of some other IE or IEs.

The 'Need' column is set to MP, MD, CV or CH and interpreted as described in 9.1.1.2.1.1-9.1.1.2.1.3 applied to the whole set. ~~followed by a condition name to indicate that the number of necessary or optional copies is conditional to the value of some other IE or IEs, or on the flow (CV case) or to information obtained in the past (CH case). An explanatory clause describes the condition. Otherwise, the column is left empty.~~

### 9.1.1.2.2 Type and reference column

This column is not filled for groups and must be filled for IEs.

This column includes the reference to a more detailed abstract description of the IE. This includes:

a) A reference to a subclause in the Information Element Description clause in the same document; typically the subclause number and titles are given, and if possible this should be a hypertext link;

b) A reference to another document, and to a subclause in the Information Element Description clause in the indicated document; typically only the subclause title is indicated;

~~c) A reference to a subclause of this document, with possibly additional information as described.~~

### 9.1.1.2.3 Semantics description

Filling this column is optional. It should be use to clarify the meaning of the IE or group of IE, as a summary of their use as described in the procedural part.

### 9.1.1.2.4 Expressing differences between FDD and TDD modes

If a PDU or a structured information element contain information elements whose Need value is different for FDD and TDD modes or if a certain structured information element is completely different for the two modes, a choice group should be used.

| IE/Group Name | Need | Multi | Type and reference | Semantics description | Version |
|---|---|---|---|---|---|
| Choice ~~systemtype~~mode | MP | | | | |
| >FDD | | | | | |
| >>element1 | MP | | | | |
| >>element2 | OP | | | | |
| >TDD | | | | | |
| >>element3 | OP | | | | |
| >>element4 | MP | | | | |

### 9.1.1.2.5 Version column

When an information element is added from one version to a latter one, the version in which the element is added (e.g.: REL-4, REL-5) is included in the version column.

When a new CHOICE group is added from one version to a later one, the version in which the group is added is included in the version column of all new rows. If some of the information elements in the new CHOICE group were included in the older version (but not inside a CHOICE group), the version column is not updated for those information elements (see also the example at the end of this clause).

When an existing CHOICE group is extended from one version to a later one to include more options, the version in which the new options are added is included in the version column of the rows describing the new options.

When the type of an information element is modified from one version to a later one to include more values, the version in which the modification takes place is included in the version column, and the new values are indicated in that column. By convention the version column is left blank for the present release.

The example below shows how the version column is used for the cases described above. The first table shows an example of a Release '99 table:

| IE/Group Name | Need | Multi | Type and reference | Semantics description | Version |
|---|---|---|---|---|---|
| *Element1-99* | MP | | Type1 | | |
| *Element2-99* | MP | | Type2 | | |
| *CHOICE choice1-99* | MP | | | | |
| *>first* | | | | | |
| *>>Element3-99* | MP | | Type3 | | |
| *>second* | | | | | |
| *>>Element4-99* | MP | | Type4 | | |
| *Element5-99* | MP | | Enumerated(a,b) | | |

The second table shows extensions of the above table in Release 4, and where the REL-4 in the version column shall be included:

| IE/Group Name | Need | Multi | Type and reference | Semantics description | Version |
|---|---|---|---|---|---|
| *Element1-99* | MP | | Type1 | | |
| *Element6-r4* | MP | | Type6 | | REL-4 |
| *CHOICE choice2-r4* | MP | | | | REL-4 |
| *>old* | | | | | REL-4 |
| *>>Element2-99* | MP | | Type2 | | |
| *>new* | | | | | REL-4 |
| *>>Element7-r4* | MP | | Type7 | | REL-4 |
| *CHOICE choice1-99* | MP | | | | |
| *>first* | | | | | |
| *>>Element3-99* | MP | | Type3 | | |
| *>second* | | | | | |
| *>>Element4-99* | MP | | Type4 | | |
| *>third* | | | | | REL-4 |
| *>>Element8-r4* | MP | | Type8 | | REL-4 |
| *Element5-99* | MP | | Enumerated(a,b,c) | | Value c is included in REL-4. |

### 9.1.1.3 Explanatory clauses

This includes the subclauses needed to elaborate conditions ~~and symbolic names (e.g., range bounds)~~. There must be one explanatory clause for each named condition~~, and for each symbolic name~~. The text must give the information sufficient to decide whether the IE/group is to be included or not~~, or the value of the symbolic name~~.

## 9.1.2 IE type description

This describes IE types referred elsewhere, either in the description of a message or in the description of another IE type. The description of an IE type must be as generic as possible, i.e., independent of any specific use. A type should as far as possible not be defined in multiple places in a specification.

An 'IE description' subclause includes one subclause per IE type.

The description of an IE type is done as a table similar to that used for the description of messages.

| IE/Group Name | Need | Multi | Type and reference | Semantics description | Version |
|---|---|---|---|---|---|
| | | | | | |

The different columns are filled ~~exactly~~ as message description columns are filled~~.~~ with the addition that in the IE Type and reference column also the use of a basic types defined in subclause 9.2 in this document is allowed. These basic types shall be considered as pre-defined and for those no reference is necessary. For IE type descriptions, explanatory clauses should also be used as described in subclause 9.1.1.3.

# 10 Usage of ASN.1

The following clauses contain guidelines for specification of protocol messages with ASN.1.The purpose of ASN.1 is to make it possible to specify message contents description of a message (i.e. what is the contents of a message) separately from its transfer syntax (i.e. how a message is encoded for transmission). ~~The features that ASN.1 provides include specification of:~~

- ~~Extensibility (both structural and extension of value set).~~
- ~~Optional IEs and values (see the clauses 10.2.2 and 10.3.10).~~
- ~~Default values (see the clauses 10.2.2 and 10.3.10).~~
- ~~Comprehension required (see the clause 10.2.4).~~
- ~~Inter/Intra IE dependency (see the clause 10.3.10).~~
- ~~Specification of partial decoding (see the clause 10.2.5).~~

The clause 11 specifies how message transfer syntax is specified. It should be noted that importance of some transfer syntax properties must be determined early during specification because of their effect on message contents description specification possibilities. The properties are **compactness** and **extensibility.** If extreme compactness is required then extensibility must be restricted. If good extensibility is required then compromises must be done regarding compactness. The sections concerning these issues are marked in the following clauses as **COMPACTNESS** and **EXTENSIBILITY**.

Identifiers that could be keywords of ~~another~~ some language (e.g.: SDL, C, ASN.1, JAVA, C++, …) should be avoided.

In the current version of the ASN.1 specifications, user-defined constraints are not used.

## 10.1 Message level

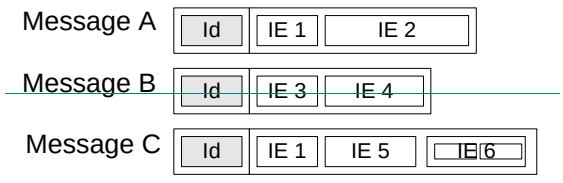
### 10.1.1 Messages

~~It is presumed that messages share the same structure, namely that they contain an identification part and a contents part. An identification part contains an IE that identifies a message among all messages in some context.~~

~~A contents part contains message specific IEs.~~

~~IE is a list of components.~~

~~Example: A protocol layer XYZ contains three messages: A, B and C. The structure of the messages is as presented in the figure 10.1.1.1.~~




**~~Figure 10.1.1.1: Three example messages~~**

~~Messages are specified using ASN.1 [1]. There are three ASN.1 types, *MessageA*, *MessageB* and *MessageC*, which contain definitions for the contents of the above messages. The mapping between the message contents types and message identifiers is as follows:~~

| Message id | Type of message contents |
|---|---|
| 1 | MessageA |
| 2 | MessageB |
| 3 | MessageC |

New message types will be introduced in the future.

In cases where different PDUs have different identification schemes it is possible to apply this categorisation for a set of PDUs that share the same identification scheme.

## 10.1.2 Message definition

In order to capture information in the previous clause the following three things must be defined:

1. A structure for the table.
2. The table itself.
3. A generic message structure that can contain both message identifier IE and message contents IEs (i.e. id 1 + *MessageA*, id 2 + *MessageB*, id 3 + *MessageC*).

The table structure is defined as follows using ASN.1 classes [2]:

```
XYZ-MESSAGE ::= CLASS {
    &id     MessageId UNIQUE,
    &Type
}
WITH SYNTAX {
    &id &Type
}

MessageId ::= INTEGER (0..63)
```

The table is defined as follows:

```
XYZ-Messages XYZ-MESSAGE ::= {
    { messageA-id MessageA } |
    { messageB-id MessageB } |
    { messageC-id MessageC } ,
    ...                 -- Extension marker => additional messages
                        -- can be introduced.
}

messageA-id MessageId ::= 1
messageB-id MessageId ::= 2
messageC-id MessageId ::= 3
```

The following type represents the generic message structure that can carry values of the messages specified in the *XYZ-Messages* table.

```
XYZ-Message ::= SEQUENCE {
    id      XYZ-MESSAGE.&id    ({XYZ-Messages)},
            -- MessageId: 1, 2 or 3

    contents    XYZ-MESSAGE.&Type ({XYZ-Messages}{@id})
            -- id=1 => MessageA, id=2 => MessageB, id=3 => MessageC
}
```

The above definition means that if *id* is 1 then the *Message* type could be interpreted as the following type:

```
XYZ-Message ::= SEQUENCE {
    id       MessageId,   -- 1
    contents    SEQUENCE {
        ie1     IE1,
        ie2     IE2
    }
}
```

If *id* is 2 then the type could be interpreted as the following type:

```
XYZ-Message ::= SEQUENCE {
    id       MessageId,   -- 2
    contents    SEQUENCE {
        ie3     IE3,
        ie4     IE4
    }
}
```

## 10.1.3 Messages and ASN.1 modules

ASN.1 definitions shall be placed in ASN.1 modules such that definitions in a module form a logical unit. For example PDUs definitions for one protocol layer could be in one ASN.1 module and IE definitions in another.

The tagging mode for the modules shall be "AUTOMATIC TAGS". Note that "AUTOMATIC TAGS" is not relevant for PER.

Example: A message definition module for the XYZ protocol layer.

```
XYZ-Messages DEFINITIONS AUTOMATIC TAGS ::=

BEGIN

XYZ-Messages XYZ-MESSAGE ::= {
    { messageA-id MessageA } |
    { messageB-id MessageB } |
    { messageC-id MessageC } ,
    ...                 -- Additional messages can be introduced.
}

MessageA ::= SEQUENCE {
    -- Message contents
}

messageA-id MessageId ::= 1

MessageB ::= SEQUENCE {
    -- Message contents
}

messageB-id MessageId ::= 2

MessageC ::= SEQUENCE {
    -- Message contents
}

messageC-id MessageId ::= 3

END
```

## 10.1.4 Messages and SDL

The identifiers *messageA-id*, *MessageA*, *messageB-id*, etc. can be used in descriptive SDL when protocol behaviour is specified. Note that classes and objects cannot be referenced in SDL96 but are allowed in SDL2000. Types and values

however can be imported to SDL definitions. The figures below contain some examples about usage of ASN.1 in SDL specifications.

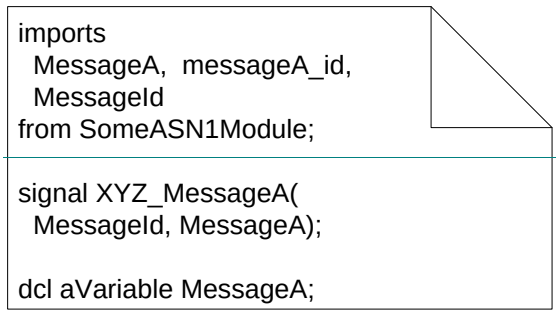



**Figure 10.1.4.1: Import and use of ASN.1 definitions in SDL**

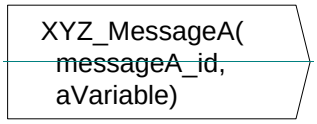



**Figure 10.1.4.2: Sending of a message id and contents**

## 10.2 Information element level

Messages consist of information elements.

The following ASN.1 message types are used in the following clauses.

```
MessageA ::= SEQUENCE {
    ie1 IE1,                -- A mandatory IE.

    ie2 IE2 OPTIONAL,       -- An optional IE.

-- Extensions from there
    ExtensionMarker SEQUENCE {} OPTIONAL

}

MessageB ::= SEQUENCE {
    ie3 IE3
        (CONSTRAINED BY {-- ComprehensionRequired(is for receiver) --}
        !comprehensionRequiredFailure) ,

    ie4 IE4 DEFAULT 0,      -- An optional IE with a default value.

-- Extensions from there
    ExtensionMarker SEQUENCE {} OPTIONAL
}

MessageC ::= SEQUENCE {
    ie1 IE1
        (CONSTRAINED BY {-- PartialDecoding(OnErrorIgnoreRest) --}
        !partialDecodingFailure)
        OPTIONAL,

    ie5 IE5
        (CONSTRAINED BY {-- PartialDecoding(OnErrorIgnoreRest) --}
        !partialDecodingFailure)
        OPTIONAL,

-- Extensions from there
    ExtensionMarker SEQUENCE {
        ie6 IE6
            (CONSTRAINED BY {-- PartialDecoding(OnErrorIgnoreRest) --}
            !partialDecodingFailure)
            OPTIONAL        -- A new IE
  } OPTIONAL
```

```
}

-- Error codes
comprehensionRequiredFailure INTEGER ::= 1
partialDecodingFailure       INTEGER ::= 2
```

## 10.2.1 Message contents

A message contents structure is defined using a sequence type (10.3.10).

Example: *MessageA*, *MessageB* and *MessageC* are message contents structures.

## 10.2.2 Optional IEs and default values

An IE can be marked as optional.

**COMPACTNESS:** Optional IEs shall be after mandatory ones. When ASN.1 is used with PER, this requirement is not relevant.

When the extension "SEQUENCE {} OPTIONAL" is used, the sender shall never indicate that the field is present.

Example: *MessageA.ie2* is an optional IE.

```
ie2 IE2 OPTIONAL
```

An IE can be marked as being optional and having a default value. In those cases a missing optional IE may be understood as having a certain value hence a defined meaning.

Example: *MessageB.ie4* is an optional IE with a default value.

```
ie4 IE4     DEFAULT 0
```

## 10.2.3 New IEs

**EXTENSIBILITY:** If new IEs will be added to a message then the message contents structure must be specified as extensible using the ellipsis notation (...). New IEs shall be added after the extension marker. New IEs shall be optional or shall have default values.

Example: *MessageC.ie6* is an additional optional IE.

```
    ...,
    ie6 IE6     OPTIONAL
```

## 10.2.4 Comprehension required

"Comprehension required" requirement can be associated with an IE. It means that after an IE value has been decoded then the value is validated. Failure in validation causes rejection of the message.

The requirement is specified as an extension to ASN.1 by using user defined constraints [3]. The comment part of the constraint shall be of the form:

*ComprehensionRequired(<additional constraint>)*

where <additional constraint> specifies the rule that the IE must satisfy.

Example: The *MessageB* is a broadcast message. The *ie3* IE contains recipient addresses. It is not until the addresses have been decoded when a receiver can decide whether it should decode the rest of the message or not.

```
    ie3 IE3
```

```
        (CONSTRAINED BY {-- ComprehensionRequired(is for receiver) --}
        !comprehensionRequiredFailure) ,
```

## 10.2.5 Partial decoding

"Partial decoding" means that a PDU can be decoded in parts. One part forms a complete value that can be separated from other parts. A decoding error in a part does not invalidate previously decoded parts. Subsequent parts are however invalidated.

"Partial decoding" is specified as an extension to ASN.1 using user defined constraints. The comment of constraint shall be of the form:

*PartialDecoding(<OnErrorClause>)*

where <OnErrorClause> specifies action in case of a decoding error. The possible alternatives are:

- OnErrorIgnoreRest: End decoding, ignore rest of the message.

Example: The *MessageC* is a multipurpose message. The IEs *ie1*, *ie5* and *ie6* are independent of each other.

```
    ie1 IE1
        (CONSTRAINED BY {-- PartialDecoding(OnErrorIgnoreRest) --}
        !partialDecodingFailure)
```

## 10.2.6 Error specification

An error specification can be associated with user-defined constraints.

A simple integer value can be associated with an exception specification or as elaborate structured value as needed.

Example: If decoding of *ie1* fails then decoder returns the error code *partialDecodingFailure*.

```
    ie1 IE1
        (CONSTRAINED BY {-- PartialDecoding(OnErrorIgnoreRest) --}
        !partialDecodingFailure)
```

# 10.3 Component level

Information elements consist of components.

The following ASN.1 types shall be used at the component level:

- Boolean (10.3.4)
- Integer (10.3.5)
- Enumerated (10.3.6)
- Bit string (10.3.7)
- Octet string (10.3.8)
- Null (10.3.9)
- Sequence (10.3.10)
- Sequence-of (10.3.11)
- Choice (10.3.12)
- Character string types (10.3.13)

### 10.3.1 Extensibility

**COMPACTNESS:** In the component level use of ASN.1 extensibility is forbidden unless otherwise stated in the following clauses.

### 10.3.2 Comprehension required

"Comprehension required" can be applied to components of sequence types, alternatives of choice types and elements of sequence-of types. See 10.2.4.

### 10.3.3 Partial decoding

"Partial decoding" can be applied to components of sequence types, alternatives of choice types and elements of sequence-of types. See 10.2.5.

### 10.3.4 Boolean

Example: A simple boolean type.

```
Flag ::= BOOLEAN

setFlag Flag ::= TRUE
```

### 10.3.5 Integer

An integer type should be constrained.

**COMPACTNESS:** An integer type shall be constrained to have a finite value set. The value set can be either continuous or non-continuous.

Named numbers can be associated with an integer type.

**COMPACTNESS, EXTENSIBILITY:** If an integer type needs to be extended in the future then two value sets must be defined:

- A value set that specifies the values that can be sent in the current protocol version.
- A value set that specifies all the possible values that can be received now and in the future.

The former value set is specified in a user-defined constraint. The comment part shall be of the form:

*Send(<value set>)*

The latter form is specified using a normal constraint, e.g. a value range constraint.

Examples: Integer types and values.

```
Counter          ::= INTEGER (0..255)      -- 0 <= Counter value <= 255

SparseValueSet ::= INTEGER (0|3|5|6|8|11)

SignedInteger   ::= INTEGER (-10..10)

-- idle stands for value 0.
Status           ::= INTEGER { idle(0), veryBusy(3) } (0..3)

-- Send values 0..3 but be prepared to receive values 0..15.
Extensible      ::= INTEGER (0..15)(CONSTRAINED BY {-- Send(0..3) --})

initialCounter Counter          ::= 0

zero             SparseValueSet ::= 0
```

```
initialStatus  Status          ::= idle
```

## 10.3.6 Enumerated

The EnumerationItem shall not contain a named number (e.g.: foo (3)).

The list of enumerated values specifies the value set for an enumerated type.

**COMPACTNESS, EXTENSIBILITY:** If an enumerated type needs to be extended in the future then two value sets must be defined as in case of integer types.

NOTE: An integer type with named numbers can be used as on alternative to an enumerated type.

Example: Enumerated types and value.

```
Enum          ::= ENUMERATED { a, b, c, d }

-- Send values a, b, c or d but be prepared to receive values
-- a, b, c, d, spare4, spare5, spare6 and spare7.
ExtendedEnum ::= ENUMERATED { a, b, c, d, spare4, spare5, spare6, spare7 }
    (CONSTRAINED BY {-- Send(a|b|c|d) --})

aEnum Enum ::= a
```

## 10.3.7 Bit string

A size constraint shall be specified. It shall be finite.

Named bits can be associated with a bit string type.

Example: Bit string types and values.

```
FixedLengthBitStr     ::= BIT STRING (SIZE (10))

VariableLengthBitStr ::= BIT STRING (SIZE (0..10))

BitFlags              ::= BIT STRING { a(0), b(1), c(2), d(3)} (SIZE (4))

fix FixedLengthBitStr     ::= '0001101100'B

var VariableLengthBitStr ::= '0'B

flg BitFlags              ::= { a, c, d } -- '1011'B
```

## 10.3.8 Octet string

A size constraint shall be specified. It shall be finite.

Example: Octet string types and values.

```
FixedLengthOctetStr     ::= OCTET STRING (SIZE (10))

VariableLengthOctetStr ::= OCTET STRING (SIZE (0..10))

UpperLayerPDUSegment    ::= OCTET STRING (SIZE (1..512))

fix FixedLengthOctetStr     ::= '0102030405060708090A'H

var VariableLengthOctetStr ::= 'FF'H
```

## 10.3.9 Null

A null type has only one value, NULL.

Example: Null type as an alternative type of a choice type.

```
IE ::= CHOICE {
    doThis    ThisArg,
    doThat    ThatArg,
    doNothing    NULL
}
```

## 10.3.10 Sequence

A sequence type is a record. Components of a sequence type can be optional or they can have default values. Optional components and components with default values should be after mandatory components.

Inner subtyping can be used to force an optional component to be present or absent in a derived type.

If an optional component is conditionally present or absent then the condition shall be specified in a user defined constraint of the form:

*Condition(<condition expression>)*

<condition expression> shall be such that both sender and receiver are able to evaluate it before a conditional component is encoded or decoded.

"Comprehension required" can be associated with a component of a sequence type.

"Partial decoding" can be associated with a component of a sequence type.

**EXTENSIBILITY:** A sequence type can be marked as extensible. Example: Sequence types and values.

```
Record ::= SEQUENCE {
    flag        Flag,
    counter Counter,
    bitFlags    BitFlags        OPTIONAL,
    extEnum ExtendedEnum    DEFAULT a
}

DerivedRecord ::= Record (WITH COMPONENTS {
                ...,
                bitFlags    PRESENT
                 })

RecordWithConditionalComponent ::= SEQUENCE {
    mand        INTEGER (0..7),
    opt     BOOLEAN OPTIONAL,
    cond        BOOLEAN OPTIONAL
} ( WITH COMPONENT {mand(7), cond PRESENT} | WITH COMPONENT {cond ABSENT} )

aRecord Record ::= {
    flag        TRUE,
    counter 100
}

anotherRecord DerivedRecord ::= {
    flag        TRUE,
    counter 100,
    bitFlags    '0101'B     -- bitFlags must be present
}
```

### 10.3.11 Sequence-of

A sequence-of type is a list of some element type. A size constraint shall be specified. It shall be finite.

"Comprehension required" can be associated with an element of a sequence-of type.

"Partial decoding" can be associated with an element of a sequence-of type.

Example: Sequence-of types and values.

```
FixedLengthList        ::= SEQUENCE (SIZE (10)) OF Record

VariableLengthList     ::= SEQUENCE (SIZE (0..10)) OF Status

UpperLayerPDUSegments ::= SEQUENCE (SIZE (1..10)) OF UpperLayerPDUSegment

aList VariableLengthList ::= { idle, 1, 2, veryBusy, 2, 1, idle }
```

### 10.3.12 Choice

A choice type is a variant record. Only one alternative component can be selected.

Inner subtyping can be used to force an alternative to be selected in a derived type.

"Comprehension required" can be associated with an alternative component of a choice type.

"Partial decoding" can be associated with an alternative component of a choice type.

**EXTENSIBILITY:** A choice type can be marked as extensible.

Example: Choice type and value.

```
VariantRecord ::= CHOICE {
    flag          Flag,
    counter Counter,
    extEnum ExtendedEnum
}

aVariantRecord VariantRecord ::= flag : FALSE
```

### 10.3.13 Restricted character string types

A size constraint shall be specified. It shall be finite.

It should specified the permitted alphabet for compactness reasons (see examples in PER [5]).

Example: Character string types.

```
FixedStr   ::= IA5String (SIZE (10))

VarStr     ::= IA5String (SIZE (1..10))

FixedWStr ::= BMPString (SIZE (10))

VarWStr    ::= BMPString (SIZE (1..10))
```

## 10.3.14 IEs and ASN.1 modules

If an IE or a component field within an IE is a parameter from another protocol layer then that type for such a field should be defined in another module. In this way there is a clear separation of definitions that are specific to different protocol layers.

Example: The XYZ protocol message *MessageC* contains an IE, which contains an OPQ protocol layer specific field *parameter1*. Type for the field is imported from that OPQ specific module.

```
XYZ-Messages DEFINITIONS AUTOMATIC TAGS ::=

BEGIN

IMPORTS
    OPQParameter      -- OPQParameter is not defined within XYZ-Messages
                      -- module.
FROM OPQ-DataTypes;

MessageC ::= SEQUENCE {
    -- Other IEs.
    ie6 IE6 OPTIONAL
}
-- Other definitions ...

IE6 ::= SEQUENCE {
    parameter1  OPQParameter,     -- Imported definitions can be
                                  -- referred to.
    parameter2  XYZParameter
}

XYZParameter ::= INTEGER (0..255)

END
```

Example: The OPQ protocol layer specific module exports *OPQParameter* type so that other modules can refer it.

```
OPQ-DataTypes DEFINITIONS AUTOMATIC TAGS ::=

BEGIN


OPQParameter ::= INTEGER (0..7)

END
```

# Annex AB (informative): Usage of ASN.1

NOTE: The text in this annex should be seen as illustration of how ASN.1 can be used and do not necessarily reflect how ASN.1 is used in the RAN specifications.

The following clauses contain guidelines for specification of protocol messages with ASN.1.The purpose of ASN.1 is to make it possible to specify message contents description of a message (i.e. what is the contents of a message) separately from its transfer syntax (i.e. how a message is encoded for transmission). The features that ASN.1 provides include specification of:

- Extensibility (both structural and extension of value set).
- Optional IEs and values (see the clauses B10.2.2 and B10.3.10).
- Default values (see the clauses 10BA.2.2 and B10.3.10).
- Comprehension required (see the clause 10BA.2.4).
- Inter/Intra IE dependency (see the clause 10BA.3.10).
- Specification of partial decoding (see the clause 10BA.2.5).

The clause 11 specifies how message transfer syntax is specified. It should be noted that importance of some transfer syntax properties must be determined early during specification because of their effect on message contents description specification possibilities. The properties are **compactness** and **extensibility.** If extreme compactness is required then extensibility must be restricted. If good extensibility is required then compromises must be done regarding compactness. The sections concerning these issues are marked in the following clauses as **COMPACTNESS** and **EXTENSIBILITY**.

Identifiers that could be keywords of another language (e.g.: SDL, C, ASN.1, JAVA, C++, …) should be avoided.

## 10BA.1 Message level

### 10BA.1.1 Messages

It is presumed that messages share the same structure, namely that they contain an identification part and a contents part. An identification part contains an IE that identifies a message among all messages in some context.

A contents part contains message specific IEs.

IE is a list of components.

Example: A protocol layer XYZ contains three messages: A, B and C. The structure of the messages is as presented in the figure 10BA.1.1.1.

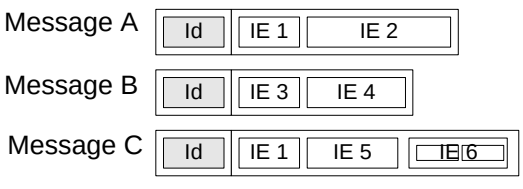



**Figure 10BA.1.1.1: Three example messages**

Messages are specified using ASN.1 [1]. There are three ASN.1 types, *MessageA*, *MessageB* and *MessageC*, which contain definitions for the contents of the above messages. The mapping between the message contents types and message identifiers is as follows:

| Message id | Type of message contents |
|---|---|
| 1 | MessageA |
| 2 | MessageB |
| 3 | MessageC |

New message types will be introduced in the future.

In cases where different PDUs have different identification schemes it is possible to apply this categorisation for a set of PDUs that share the same identification scheme.

## ~~10~~BA.1.2 Message definition

In order to capture information in the previous clause the following three things must be defined:

1. A structure for the table.
2. The table itself.
3. A generic message structure that can contain both message identifier IE and message contents IEs (i.e. id 1 + *MessageA*, id 2 + *MessageB*, id 3 + *MessageC*).

The table structure is defined as follows using ASN.1 classes [2]:

```
XYZ-MESSAGE ::= CLASS {
    &id     MessageId UNIQUE,
    &Type
}
WITH SYNTAX {
    &id &Type
}

MessageId ::= INTEGER (0..63)
```

The table is defined as follows:

```
XYZ-Messages XYZ-MESSAGE ::= {
    { messageA-id MessageA } |
    { messageB-id MessageB } |
    { messageC-id MessageC } ,
    ...                 -- Extension marker => additional messages
                        -- can be introduced.
}

messageA-id MessageId ::= 1
messageB-id MessageId ::= 2
messageC-id MessageId ::= 3
```

The following type represents the generic message structure that can carry values of the messages specified in the *XYZ-Messages* table.

```
XYZ-Message ::= SEQUENCE {
    id      XYZ-MESSAGE.&id   ({XYZ-Messages)},
            -- MessageId: 1, 2 or 3

    contents    XYZ-MESSAGE.&Type ({XYZ-Messages}{@id})
            -- id=1 => MessageA, id=2 => MessageB, id=3 => MessageC
}
```

The above definition means that if *id* is 1 then the *Message* type could be interpreted as the following type:

```
XYZ-Message ::= SEQUENCE {
    id      MessageId,  -- 1
    contents    SEQUENCE {
        ie1     IE1,
        ie2     IE2
    }
}
```

If *id* is 2 then the type could be interpreted as the following type:

```
XYZ-Message ::= SEQUENCE {
    id      MessageId,  -- 2
    contents    SEQUENCE {
        ie3     IE3,
        ie4     IE4
    }
}
```

## ~~10~~BA.1.3 Messages and ASN.1 modules

ASN.1 definitions shall be placed in ASN.1 modules such that definitions in a module form a logical unit. For example PDUs definitions for one protocol layer could be in one ASN.1 module and IE definitions in another.

The tagging mode for the modules shall be "AUTOMATIC TAGS". Note that "AUTOMATIC TAGS" is not relevant for PER.

Example: A message definition module for the XYZ protocol layer.

```
XYZ-Messages DEFINITIONS AUTOMATIC TAGS ::=

BEGIN

XYZ-Messages XYZ-MESSAGE ::= {
    { messageA-id MessageA } |
    { messageB-id MessageB } |
    { messageC-id MessageC } ,
    ...             -- Additional messages can be introduced.
}

MessageA ::= SEQUENCE {
    -- Message contents
}

messageA-id MessageId ::= 1

MessageB ::= SEQUENCE {
    -- Message contents
}

messageB-id MessageId ::= 2

MessageC ::= SEQUENCE {
    -- Message contents
}

messageC-id MessageId ::= 3

END
```

## ~~10~~BA.1.4 Messages and SDL

The identifiers *messageA-id*, *MessageA*, *messageB-id*, etc. can be used in descriptive SDL when protocol behaviour is specified. Note that classes and objects cannot be referenced in SDL96 but are allowed in SDL2000. Types and values

however can be imported to SDL definitions. The figures below contain some examples about usage of ASN.1 in SDL specifications.

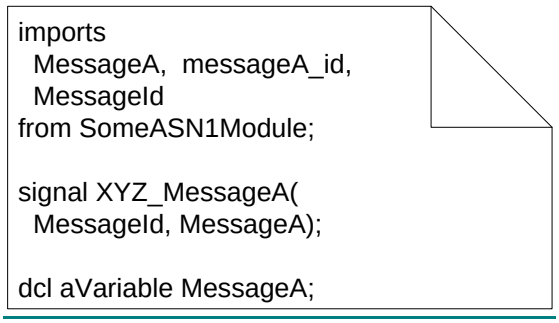



**Figure ~~10~~BA.1.4.1: Import and use of ASN.1 definitions in SDL**

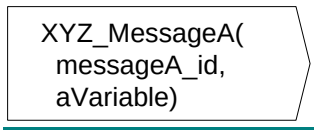



**Figure ~~10~~BA.1.4.2: Sending of a message id and contents**

## ~~10~~BA.2 Information element level

Messages consist of information elements.

The following ASN.1 message types are used in the following clauses.

```
MessageA ::= SEQUENCE {
    ie1 IE1,             -- A mandatory IE.

    ie2 IE2 OPTIONAL,        -- An optional IE.

-- Extensions from there
    ExtensionMarker SEQUENCE {} OPTIONAL

}

MessageB ::= SEQUENCE {
    ie3 IE3
        (CONSTRAINED BY {-- ComprehensionRequired(is for receiver) --}
        !comprehensionRequiredFailure)  ,

    ie4 IE4 DEFAULT 0,       -- An optional IE with a default value.

-- Extensions from there
    ExtensionMarker SEQUENCE {} OPTIONAL
}

MessageC ::= SEQUENCE {
    ie1 IE1
        (CONSTRAINED BY {-- PartialDecoding(OnErrorIgnoreRest) --}
        !partialDecodingFailure)
        OPTIONAL,

    ie5 IE5
        (CONSTRAINED BY {-- PartialDecoding(OnErrorIgnoreRest) --}
        !partialDecodingFailure)
        OPTIONAL,

-- Extensions from there
    ExtensionMarker SEQUENCE {
        ie6 IE6
            (CONSTRAINED BY {-- PartialDecoding(OnErrorIgnoreRest) --}
            !partialDecodingFailure)
            OPTIONAL        -- A new IE
  } OPTIONAL
```

```
}

-- Error codes
comprehensionRequiredFailure INTEGER ::= 1
partialDecodingFailure       INTEGER ::= 2
```

## 10BA.2.1 Message contents

A message contents structure is defined using a sequence type (10BA.3.10).

Example: *MessageA*, *MessageB* and *MessageC* are message contents structures.

## 10BA.2.2 Optional IEs and default values

An IE can be marked as optional.

**COMPACTNESS:** Optional IEs shall be after mandatory ones. When ASN.1 is used with PER, this requirement is not relevant.

When the extension "SEQUENCE {} OPTIONAL" is used, the sender shall never indicate that the field is present.

Example: *MessageA.ie2* is an optional IE.

```
ie2 IE2 OPTIONAL
```

An IE can be marked as being optional and having a default value. In those cases a missing optional IE may be understood as having a certain value hence a defined meaning.

Example: *MessageB.ie4* is an optional IE with a default value.

```
ie4 IE4     DEFAULT 0
```

## 10BA.2.3 New IEs

**EXTENSIBILITY:** If new IEs will be added to a message then the message contents structure must be specified as extensible using the ellipsis notation (...). New IEs shall be added after the extension marker. New IEs shall be optional or shall have default values.

Example: *MessageC.ie6* is an additional optional IE.

```
    ... ,
    ie6 IE6     OPTIONAL
```

## 10BA.2.4 Comprehension required

"Comprehension required" requirement can be associated with an IE. It means that after an IE value has been decoded then the value is validated. Failure in validation causes rejection of the message.

The requirement is specified as an extension to ASN.1 by using user defined constraints [3]. The comment part of the constraint shall be of the form:

*ComprehensionRequired(<additional constraint>)*

where <additional constraint> specifies the rule that the IE must satisfy.

Example: The *MessageB* is a broadcast message. The *ie3* IE contains recipient addresses. It is not until the addresses have been decoded when a receiver can decide whether it should decode the rest of the message or not.

```
    ie3 IE3
```

```
        (CONSTRAINED BY {-- ComprehensionRequired(is for receiver) --}
        !comprehensionRequiredFailure)  ,
```

## ~~10~~BA.2.5 Partial decoding

"Partial decoding" means that a PDU can be decoded in parts. One part forms a complete value that can be separated from other parts. A decoding error in a part does not invalidate previously decoded parts. Subsequent parts are however invalidated.

"Partial decoding" is specified as an extension to ASN.1 using user defined constraints. The comment of constraint shall be of the form:

*PartialDecoding(<OnErrorClause>)*

where <OnErrorClause> specifies action in case of a decoding error. The possible alternatives are:

- OnErrorIgnoreRest: End decoding, ignore rest of the message.

Example: The *MessageC* is a multipurpose message. The IEs *ie1*, *ie5* and *ie6* are independent of each other.

```
    ie1 IE1
        (CONSTRAINED BY {-- PartialDecoding(OnErrorIgnoreRest) --}
        !partialDecodingFailure)
```

## ~~10~~BA.2.6 Error specification

An error specification can be associated with user-defined constraints.

A simple integer value can be associated with an exception specification or as elaborate structured value as needed.

Example: If decoding of *ie1* fails then decoder returns the error code *partialDecodingFailure*.

```
    ie1 IE1
        (CONSTRAINED BY {-- PartialDecoding(OnErrorIgnoreRest) --}
        !partialDecodingFailure)
```

# ~~10~~BA.3 Component level

Information elements consist of components.

The following ASN.1 types shall be used at the component level:

- Boolean (~~10~~BA.3.4)
- Integer (~~10~~BA.3.5)
- Enumerated (~~10~~BA.3.6)
- Bit string (~~10~~BA.3.7)
- Octet string (~~10~~BA.3.8)
- Null (~~10~~BA.3.9)
- Sequence (~~10~~BA.3.10)
- Sequence-of (~~10~~BA.3.11)
- Choice (~~10~~BA.3.12)
- Character string types (~~10~~BA.3.13)

## ~~10~~BA.3.1 Extensibility

**COMPACTNESS:** In the component level use of ASN.1 extensibility is forbidden unless otherwise stated in the following clauses.

## ~~10~~BA.3.2 Comprehension required

"Comprehension required" can be applied to components of sequence types, alternatives of choice types and elements of sequence-of types. See ~~10~~BA.2.4.

## ~~10~~BA.3.3 Partial decoding

"Partial decoding" can be applied to components of sequence types, alternatives of choice types and elements of sequence-of types. See ~~10~~BA.2.5.

## ~~10~~BA.3.4 Boolean

Example: A simple boolean type.

```
Flag ::= BOOLEAN

setFlag Flag ::= TRUE
```

## ~~10~~BA.3.5 Integer

An integer type should be constrained.

**COMPACTNESS:** An integer type shall be constrained to have a finite value set. The value set can be either continuous or non-continuous.

Named numbers can be associated with an integer type.

**COMPACTNESS, EXTENSIBILITY:** If an integer type needs to be extended in the future then two value sets must be defined:

- A value set that specifies the values that can be sent in the current protocol version.
- A value set that specifies all the possible values that can be received now and in the future.

The former value set is specified in a user-defined constraint. The comment part shall be of the form:

*Send(<value set>)*

The latter form is specified using a normal constraint, e.g. a value range constraint.

Examples: Integer types and values.

```
Counter        ::= INTEGER (0..255)     -- 0 <= Counter value <= 255

SparseValueSet ::= INTEGER (0|3|5|6|8|11)

SignedInteger  ::= INTEGER (-10..10)

-- idle stands for value 0.
Status         ::= INTEGER { idle(0), veryBusy(3) } (0..3)

-- Send values 0..3 but be prepared to receive values 0..15.
Extensible     ::= INTEGER (0..15)(CONSTRAINED BY {-- Send(0..3) --})

initialCounter Counter        ::= 0

zero           SparseValueSet ::= 0
```

```
initialStatus  Status         ::= idle
```

## 10BA.3.6 Enumerated

The EnumerationItem shall not contain a named number (e.g.: foo (3)).

The list of enumerated values specifies the value set for an enumerated type.

**COMPACTNESS, EXTENSIBILITY:** If an enumerated type needs to be extended in the future then two value sets must be defined as in case of integer types.

NOTE: An integer type with named numbers can be used as on alternative to an enumerated type.

Example: Enumerated types and value.

```
Enum          ::= ENUMERATED { a, b, c, d }

-- Send values a, b, c or d but be prepared to receive values
-- a, b, c, d, spare4, spare5, spare6 and spare7.
ExtendedEnum ::= ENUMERATED { a, b, c, d, spare4, spare5, spare6, spare7 }
    (CONSTRAINED BY {-- Send(a|b|c|d) --})

aEnum Enum ::= a
```

## 10BA.3.7 Bit string

A size constraint shall be specified. It shall be finite.

Named bits can be associated with a bit string type.

Example: Bit string types and values.

```
FixedLengthBitStr     ::= BIT STRING (SIZE (10))

VariableLengthBitStr ::= BIT STRING (SIZE (0..10))

BitFlags              ::= BIT STRING { a(0), b(1), c(2), d(3)} (SIZE (4))

fix FixedLengthBitStr     ::= '0001101100'B

var VariableLengthBitStr ::= '0'B

flg BitFlags              ::= { a, c, d } -- '1011'B
```

## 10BA.3.8 Octet string

A size constraint shall be specified. It shall be finite.

Example: Octet string types and values.

```
FixedLengthOctetStr     ::= OCTET STRING (SIZE (10))

VariableLengthOctetStr ::= OCTET STRING (SIZE (0..10))

UpperLayerPDUSegment    ::= OCTET STRING (SIZE (1..512))

fix FixedLengthOctetStr     ::= '0102030405060708090A'H

var VariableLengthOctetStr ::= 'FF'H
```

## 10BA.3.9 Null

A null type has only one value, NULL.

Example: Null type as an alternative type of a choice type.

```
IE ::= CHOICE {
    doThis    ThisArg,
    doThat    ThatArg,
    doNothing     NULL
}
```

## 10BA.3.10 Sequence

A sequence type is a record. Components of a sequence type can be optional or they can have default values. Optional components and components with default values should be after mandatory components.

Inner subtyping can be used to force an optional component to be present or absent in a derived type.

If an optional component is conditionally present or absent then the condition shall be specified in a user defined constraint of the form:

*Condition(<condition expression>)*

<condition expression> shall be such that both sender and receiver are able to evaluate it before a conditional component is encoded or decoded.

"Comprehension required" can be associated with a component of a sequence type.

"Partial decoding" can be associated with a component of a sequence type.

**EXTENSIBILITY:** A sequence type can be marked as extensible. Example: Sequence types and values.

```
Record ::= SEQUENCE {
    flag        Flag,
    counter Counter,
    bitFlags    BitFlags        OPTIONAL,
    extEnum ExtendedEnum    DEFAULT a
}

DerivedRecord ::= Record (WITH COMPONENTS {
                ...,
                bitFlags    PRESENT
                  })

RecordWithConditionalComponent ::= SEQUENCE {
    mand        INTEGER (0..7),
    opt     BOOLEAN OPTIONAL,
    cond        BOOLEAN OPTIONAL
} ( WITH COMPONENT {mand(7), cond PRESENT} | WITH COMPONENT {cond ABSENT} )

aRecord Record ::= {
    flag        TRUE,
    counter 100
}

anotherRecord DerivedRecord ::= {
    flag        TRUE,
    counter 100,
    bitFlags    '0101'B     -- bitFlags must be present
}
```

## ~~10~~BA.3.11 Sequence-of

A sequence-of type is a list of some element type. A size constraint shall be specified. It shall be finite.

"Comprehension required" can be associated with an element of a sequence-of type.

"Partial decoding" can be associated with an element of a sequence-of type.

Example: Sequence-of types and values.

```
FixedLengthList       ::= SEQUENCE (SIZE (10)) OF Record

VariableLengthList    ::= SEQUENCE (SIZE (0..10)) OF Status

UpperLayerPDUSegments ::= SEQUENCE (SIZE (1..10)) OF UpperLayerPDUSegment

aList VariableLengthList ::= { idle, 1, 2, veryBusy, 2, 1, idle }
```

## ~~10~~BA.3.12 Choice

A choice type is a variant record. Only one alternative component can be selected.

Inner subtyping can be used to force an alternative to be selected in a derived type.

"Comprehension required" can be associated with an alternative component of a choice type.

"Partial decoding" can be associated with an alternative component of a choice type.

**EXTENSIBILITY:** A choice type can be marked as extensible.

Example: Choice type and value.

```
VariantRecord ::= CHOICE {
    flag        Flag,
    counter Counter,
    extEnum ExtendedEnum
}

aVariantRecord VariantRecord ::= flag : FALSE
```

## ~~10~~BA.3.13 Restricted character string types

A size constraint shall be specified. It shall be finite.

It should specified the permitted alphabet for compactness reasons (see examples in PER [5]).

Example: Character string types.

```
FixedStr  ::= IA5String (SIZE (10))

VarStr    ::= IA5String (SIZE (1..10))

FixedWStr ::= BMPString (SIZE (10))

VarWStr   ::= BMPString (SIZE (1..10))
```

## ~~10~~BA.3.14 IEs and ASN.1 modules

If an IE or a component field within an IE is a parameter from another protocol layer then that type for such a field should be defined in another module. In this way there is a clear separation of definitions that are specific to different protocol layers.

Example: The XYZ protocol message *MessageC* contains an IE, which contains an OPQ protocol layer specific field *parameter1*. Type for the field is imported from that OPQ specific module.

```
XYZ-Messages DEFINITIONS AUTOMATIC TAGS ::=

BEGIN

IMPORTS
    OPQParameter    -- OPQParameter is not defined within XYZ-Messages
                    -- module.
FROM OPQ-DataTypes;

MessageC ::= SEQUENCE {
    -- Other IEs.
    ie6 IE6 OPTIONAL
}
-- Other definitions ...

IE6 ::= SEQUENCE {
    parameter1  OPQParameter,   -- Imported definitions can be
                                -- referred to.
    parameter2  XYZParameter
}

XYZParameter ::= INTEGER (0..255)

END
```

Example: The OPQ protocol layer specific module exports *OPQParameter* type so that other modules can refer it.

```
OPQ-DataTypes DEFINITIONS AUTOMATIC TAGS ::=

BEGIN


OPQParameter ::= INTEGER (0..7)

END
```

# Annex BC (informative): Handling of DS-41

- Modelling of RRC services is provided by means of primitives.
- RRC CN dependent info:
- In broadcast message, neighbour cells are described the same way as for GSM neighbour cells (i.e.: in the same SystemInformationBlock but with a tag to indicate CN type or RTT).
- In dedicated messages.

  a transparent container as NAS info is used to carry ANSI-41;

  for PLMN Id and Identities used by the RRC, the CN Type info is used;

  NAS binding info is used;
- In Paging messages, a tag to indicate CN type is used.
- Extensions like handover message to Multicarrier is handled the same way as GSM.

# Annex CA: Change history

| Change history | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Date** | **TSG #** | **TSG Doc.** | **CR** | **Rev** | **Subject/Comment** | **Old** | **New** |
| 12/1999 | RP-06 | RP-99659 | - | | Approved at TSG-RAN #6 and placed under Change Control | - | 3.0.0 |
| 03/2000 | RP-07 | RP-000048 | 001 | 2 | Further clarifications on specialised encoding | 3.0.0 | 3.1.0 |
| | RP-07 | RP-000048 | 003 | 1 | Modification of the 'presence' column specification in tabular format, and other editorial modifications | 3.0.0 | 3.1.0 |
| | RP-07 | RP-000048 | 005 | | Editorial corrections on subclause 11.2 | 3.0.0 | 3.1.0 |
| | RP-07 | RP-000048 | 006 | | Improvement of integers and enumerated, and introduction of reals and octet strings | 3.0.0 | 3.1.0 |
| 12/2000 | RP-10 | RP-000575 | 007 | | Extension rules for supporting future releases | 3.1.0 | 3.2.0 |
| 03/2001 | RP-11 | RP-010033 | 008 | | Description of backward compatibility consideration rule for RANAP, SABP, RNSAP and NBAP ASN.1 | 3.2.0 | 3.3.0 |
| | RP-11 | RP-010033 | 009 | | Usage of the Version column | 3.2.0 | 3.3.0 |
| | RP-11 | RP-010033 | 010 | 1 | Clean-up | 3.2.0 | 3.3.0 |
| | RP-11 | RP-010033 | 011 | | Recommendations on the use of the extension mechanism | 3.2.0 | 3.3.0 |

**3GPP TSG-RAN WG2 Meeting #21**
**Busan, Korea, May 21st-25th 2001**

***Tdoc R2-011331***

*CR-Form-v3*

# CHANGE REQUEST

⌘ **25.921** **CR** **015** ⌘ rev **-** ⌘ Current version: **4.0.0** ⌘

*For **HELP** on using this form, see bottom of this page or look at the pop-up text over the ⌘ symbols.*

***Proposed change affects:*** ⌘ (U)SIM ☐ ME/UE **X** Radio Access Network **X** Core Network ☐

| | | |
|---|---|---|
| ***Title:*** | ⌘ | Clean-up |
| ***Source:*** | ⌘ | TSG-RAN WG2 |
| ***Work item code:*** | ⌘ | TEI |
| ***Date:*** | ⌘ | 2001-05-28 |
| ***Category:*** | ⌘ | **A** |
| ***Release:*** | ⌘ | REL-4 |

*Use one of the following categories:*
***F*** *(essential correction)*
***A*** *(corresponds to a correction 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.

*Use one of the following releases:*

| | |
|---|---|
| *2* | *(GSM Phase 2)* |
| *R96* | *(Release 1996)* |
| *R97* | *(Release 1997)* |
| *R98* | *(Release 1998)* |
| *R99* | *(Release 1999)* |
| *REL-4* | *(Release 4)* |
| *REL-5* | *(Release 5)* |

***Reason for change:*** ⌘

Inconsistency between the guidelines and how they are applied
The guidelines are not in-line with the current RAN2 specifications (essentially the RRC specification). Several options and guidelines are out of date, are misleading, and need thus to be corrected or removed.

Differences between RAN2 and RAN3 approaches
In RAN2 and RAN3 sometimes slightly different approaches are made, while the guidelines mainly reflects the RAN2 approach. Therefore there is a need for more generalised guidelines, but still leave room to document how a guideline is specialised in RAN2 and RAN3. Note: A separate CR proposes changes from RAN3 perspective.

Summary of change: ⌘

- Clause 1, "Scope" is updated to reflect the scope of the document, which is control-plane protocols specified in 3GPP TSG-RAN such as RRC, RANAP etc. Some wording changed as well.
- In clause 2, "References", the "3GPP drafting rules" are corrected to be TR 21.801 (Rel-4) with the name "Specification drafting rules".
- Numerous clarifications in Clause 4, "Principles to ensure compatibility":
  Chapter 4.2, Level 1 of principles: Protocol level
  The chapter states that it "shall be possible to discriminate different versions of any protocol". It is changed to indicate the purpose "it shall be possible to inter-work different versions of a protocol specification".
  Also, the chapter seems to suggest a "protocol discriminator" but the purpose can be achieved by other means, which means the sentence can be removed.
  Chapter 4.3.1, New messages
  The principle is rather "the protocols shall specify a mechanism such that new messages can be introduced without causing harm".
  Chapter 4.3.2, Partial decoding
  The chapter is unclear.
  It is changed to describe that "extensions are allowed in a compatible way".

Chapter 4.4.2, Optional IE
To put optional IEs after mandatory ones is a too strong recommendation since it depends on the transfer syntax and the practice is not used. The subclause is removed.
Chapter 4.4.3, Adding mandatory IE
The first paragraph is ambiguous and is clarified. The subclause also mentions UE, which is Uu specific and that sentence can therefore be removed.
Chapter 4.4.4, Missing optional IE
Corrected to be "Absent optional IE" since "missing" is confusing.
Chapter 4.4.5, Comprehension required
The content of the chapter may be interpreted to specify the "ASN.1 comprehension required" mechanism as a principle. Instead the contents is changed to describe the principle of using "criticality" information.
Chapter 4.4.6, Partial Decoding
There is not a requirement to use partial decoding and therefore it is changed to permission ("may").
Chapter 4.5.1, Reserved values and spare fields
The text is inconsistent with the rest of the TR and therefore changed.
Chapter 4.5.3, Missing optional value
This chapter is irrelevant (not used) and is therefore removed.

- Clause 7, "Protocol procedure specification rules" has been updated.
The general part is modified to reflect what really is general for the RAN2 and RAN3 specifications.
The RRC part describes now how these general rules are applied in the RRC case. Protocol specific guidelines that do not need to be visible in the TR should be described in each protocol specification.
A subclause on specification of algorithms and formulas has been added. The operators div and mod are defined.
The DS-41 part is moved to an informative annex since they are out-of-date and no replacement is proposed in this CR.

- Some clarifications to clause 8, "Messages".
Subclause 8.1 "Summary what has been agreed" is removed, since the content is already covered in other parts of the TR.
Chapter 8.2, Definitions
The terms "message contents description" and "message encoding" are clarified by connecting them to the terms and "abstract syntax" and "transfer syntax".
Chapter 8.3, Logical description
The reference to clause 8 should be clause 9
Chapter 8.5, Compilability of the transfer syntax
The reference to CSN.1 is very specific and is changed to indicate that "specialised encoding" is allowed.
Chapter 8.7, Evolvability/Extensibility
The sentence "The transfer syntax shall keep the same level of compactness as the initial design" is ambiguous, formally incorrect and is thus removed.

- Numerous corrections to clause 9 "Usage of tabular format".
Chapter 9.1.1.2, The Information Element table
A heading is missing and is added.
The content of the type and reference column is not using any indentation.
The number of columns of the tabular format shall be 6 (not 65).
Chapter 9.1.1.2.1.1 Mandatory
The handling of Multi column for mandatory IEs is corrected.
Chapter 9.1.1.2.1.2 Optional
The handling of Multi column for optional IEs is corrected.
Chapter 9.1.1.2.1.3 Conditional
The handling of Multi column for conditional IEs is corrected.
It is also clarified that also if a condition is not met may result in a presence or absence requirement for an IE. A sentence of error handling of conditions is added.

Chapter 9.1.1.2.1.4 Choice
The alternative of filling the need column is missing and is added.
Chapter 9.1.1.2.1.5 Set
The text refers to “copies” of IEs, which is unclear and therefore corrected to be “instances”.
The handling of Need column is not correct (should be filled).
Chapter 9.1.1.2.2, Type and reference column
The bullet “A reference to a subclause of this document…”, is removed since the 25.921 predefined types are not used in message descriptions, only in IE type descriptions.
Chapter 9.1.1.2.4, Expressing differences between FDD and TDD modes
The example tabular format is corrected. It should be “mode” not “system type” and the need column for the choice should be filled.

- Chapter 9.1.1.3, Explanatory clauses
  Symbolic names are not part of the explanatory causes in RRC (but in the WG3 specifications) and that recommendation is removed.
- Chapter 9.1.2, IE type description
  It is clarified that explanatory clauses should be used also for the IE type description. It is added that the basic types defined in 25.921 may be used and for these no reference is necessary, since they should be regarded as pre-defined. A recommendation is added saying that multiple definitions of the same type should be avoided.
- The subclauses 10.1-10.3 contain merely an overview of ASN.1 and where any guidelines are given, they are not in-line with how ASN.1 is used in the specifications. Therefore the text in the subclauses is moved to an informative annex. No replacing text is proposed by now.
- A rule of not using user-defined constraints is added to the beginning of clause 10. The references to the sections 10.1-10.3 that are moved to an annex are removed.

| | | |
|---|---|---|
| ***Consequences if not approved:*** | ⌘ | The guidelines can not be applied to the specifications and if they are anyway applied there is a risk that further updates of the specifications will be made in an undesirable way. |

| | | | | |
|---|---|---|---|---|
| ***Clauses affected:*** | ⌘ | 1, 2, 4.2, 4.3.1, 4.3.2, 4.4.2, 4.4.3, 4.4.4, 4.4.5, 4.4.6, 4.5.3, 7.1, 7.1a (new), 7.2, 7.3, 8.1, 8.2, 8.3, 8.5, 8.7, 9.1.1.2, 9.1.1.2.a (new), 9.1.1.2.1, 9.1.1.2.1.1, 9.1.1.2.1.2, 9.1.1.2.1.3, 9.1.1.2.1.4, 9.1.1.2.1.5, 9.1.1.2.2, 9.1.1.2.4, 9.1.1.3, 9.1.2, 10, 10.1.1, 10.1.2, 10.1.3, 10.1.4, 10.2, 10.2.1, 10.2.2, 10.2.3, 10.2.4, 10.2.5, 10.2.6, 10.3, 10.3.1, 10.3.2, 10.3.3, 10.3.4, 10.3.5, 10.3.6, 10.3.7, 10.3.8, 10.3.9, 10.3.10, 10.3.11, 10.3.12, 10.3.13, 10.3.14, Annex A, Annex B (new) | | |
| ***Other specs affected:*** | ⌘ | ☐ Other core specifications<br>☐ Test specifications<br>☐ O&M Specifications | ⌘ | |
| ***Other comments:*** | ⌘ | | | |

**How to create CRs using this form:**
Comprehensive information and tips about how to create CRs can be found at: http://www.3gpp.org/3G_Specs/CRs.htm. Below is a brief summary:

1) Fill out the above form. The symbols above marked ⌘ contain pop-up help information about the field that they are closest to.
2) Obtain the latest version for the release of the specification to which the change is proposed. Use the MS Word "revision marks" feature (also known as "track changes") when making the changes. All 3GPP specifications can be downloaded from the 3GPP server under ftp://www.3gpp.org/specs/ For the latest version, look for the directory name with the latest date e.g. 2000-09 contains the specifications resulting from the September 2000 TSG meetings.

3) With "track changes" disabled, paste the entire CR form (use CTRL-A to select it) into the specification just in front of the clause containing the first piece of changed text. Delete those parts of the specification which are not relevant to the change request.

# 1 Scope

The present document provides a guideline for ~~using formal languages in~~ protocol ~~description~~ specification of UMTS stage 2 and 3 including the usage of formal languages and rules for error handling. This document covers control-plane protocols specified in TSG-RAN such as RRC, RANAP, RNSAP, NBAP and SABP. ~~all interfaces involved in radio access protocols such as Uu, Iu, Iur and Iub.~~

# 2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication, edition number, version number, etc.) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies. In the case of a reference to a 3GPP document (including a GSM document), a non-specific reference implicitly refers to the latest version of that document *in the same Release as the present document.*


[1] ITU-T Recommendation X.680: "Abstract Syntax Notation One (ASN.1): Specification of the basic notation".

[2] ITU-T Recommendation X.681: "Abstract Syntax Notation One (ASN.1): Information object specification".

[3] ITU-T Recommendation X.682: "Abstract Syntax Notation One (ASN.1): Constraint specification".

[4] ITU-T Recommendation X.690: "ASN.1 Encoding Rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)".

[5] ITU-T Recommendation X.691: "ASN.1 Encoding Rules - Specification of Packed Encoding Rules (PER)".

[6] CSN.1: "specification, version 2.0".

[7] ITU-T Recommendation Z.100: "Specification and description language (SDL)".

[8] ITU-T Recommendation Z.105: "SDL Combined with ASN.1 (SDL/ASN.1)".

[9] ITU-T Recommendation Z.120: "Message Sequence Chart (MSC)".

[10] ISO/IEC 9646-3: "The Tree and Tabular Combined Notation".

[11] 3GPP ~~TS 21.200~~TR 21.801 (Rel-4): "~~3GPP D~~Specification drafting rules".


# 3 Void

# 4 Principles to ensure compatibility

## 4.1 Introduction

The rules edicted intend to prevent incompatibilities between several phases of UMTS evolution (in analogy to what happened from GSM phase 1 to GSM phase 2).

## 4.2 Level 1 of principles: Protocol level

It shall be possible to ~~discriminate~~ inter-work different versions of any protocol specification.

An unknown protocol shall not cause problems to any entity that terminates the protocol. ~~The messages using this protocol discriminator shall be discarded by the receiving entity.~~

As a consequence, introduction of new protocol shall not disturb any receiving entity.

## 4.3 Level 2 of principles: Message level

### 4.3.1 New messages

The protocols shall specify a mechanism such that ~~N~~new message types shall be able to be introduced without causing any unexpected behaviour or damage. ~~New messages not understood shall be discarded by the receiving entity.~~

~~As an exception to this principle it can be possible to~~ The protocols may define a mechanism that allows a different behaviour, on received messages that are not understood, when a specific reaction is requested from the receiving entity. This mechanism has to be implemented from the beginning. A special care has to be taken into account when defining broadcast messages and the associated Error handling. Further refinement on this paragraph is needed.

Such a mechanism is not required inside the network part.

### 4.3.2 Partial decoding

PDU extensions are allowed in a compatible way, e.g. by utilising partial decoding or other mechanisms that allows the decoder to skip the extensions. Partial decoding means that a PDU can be decoded in parts. One part forms a complete value that can be separated from other parts. ~~A decoding error in a part does not invalidate previously decoded parts. Subsequent parts are however invalidated because if an error has occurred one cannot be sure whether the trailing values are really valid.~~

~~Example: A multipurpose PDU contains a list of four PDUs. The two first PDUs are valid but the third one is invalid. The two first are decoded but the third and fourth ones are ignored.~~

## 4.4 Level 3 of principles: Information element level

### 4.4.1 New IE

New elements shall generally be discarded when not understood.

In some cases new elements might be taken into account when specific behaviour is requested from the receiving side (e.g. a rejection of the message is expected when the element is not understood: «comprehension required»).

### 4.4.2 Void~~Optional IE~~

~~Optional IE should be located after mandatory ones.~~

### 4.4.3 Adding mandatory IE

For backward compatibility reasons, addition of mandatory IE shall be avoided. ~~In the first stage of UMTS, a set of functionality is available for each class of UE. Mandatory IE may be added only if they are mandatory for further classes of UE.~~

### 4.4.4 ~~Missing~~ Absent optional IE

~~Missing~~ Absent optional element may be understood as having a certain default value hence a defined meaning.

~~See also missing values in Values level.~~

### 4.4.5 Comprehension required

"Comprehension required" requirement can be associated with an IE or a message. It means that the IE or message is tagged with “criticality” information (explicit in the message or implicit based on the type of IE or message). Any action performed by the receiver if the IE or message is not understood (“comprehended”) is based on this “criticality” information.~~It means that after an IE value has been decoded then the value is validated according to some specified criteria. Failure in validation causes rejection of the message.~~

~~Example: A broadcast message contains a list of recipient addresses. If a recipient's address is not included in the list then a recipient ignores the whole message.~~

### 4.4.6 Partial Decoding

The notion of partial decoding ~~shall~~ may also be applied at the IE level.

## 4.5 Level 4 of principles: Values level

### 4.5.1 Reserved values and spare fields

Reserved values shall be forbidden. Otherwise entity receiving such a value shall reject the message. This would create difficulties when provided on broadcast channel.

Spare field shall be forbidden. Otherwise entity receiving such a spare field shall not make any decoding on that field and shall not reject the message.

### 4.5.2 Unspecified values

As far as possible default understanding shall be provided for unspecified values.

### 4.5.3 Void~~Missing optional value~~

~~A default value may be specified for the receiver when the sender did not include a field containing this value.~~

### 4.5.4 Extension of value set

There are cases when a data field may originally contain only a definite set of values. In the future the set of values grows but the number new values can be anticipated. There are two alternative ways to specify extension of a value set:

1) Infinite extension of a value set. Example: The first version of a data field may contain only values 0-3. In the future the field may contain any positive integer value.

2) Finite extension of a value set. Example: The first version of a data field may contain only values 0-3. In the future values 4-15 shall also be used.

# 5 Message Sequence Charts

It is agreed to recommend the use of MSCs as one of the formal methods.

MSCs are adapted for description of normal behaviour of protocol layers between peer entities and/or through SAPs. So it may be used in stage 2 of protocol description.

# 6 Specification and Description Language

The groups are encouraged to use of SDL where appropriate. The SDL code included in the standards should follow the descriptive SDL guidelines from ETSI TC-MTS (DEG MTS-00050) as closely as possible.

The groups themselves should decide how SDL is used.

In some protocol parts, text is more adapted (e.g.: algorithm or multiplexing), in some other parts SDL is better.

SDL is adapted for describing the observable behaviour of a protocol layer.

In TSG-RAN WG2, the specifications shall not use SDL for the normative part of the specifications in the present version.

# 7 Protocol procedure specification rules

## 7.1 General

- A protocol specification shall contain a 'Procedures' clause, which specifies the functional behaviour, using "procedures". A procedure is typically a sequence of events, with a start and an end, which can be observed in the protocol and/or in the interfaces to other layers (upper and/or lower layers).
- The procedure specification shall be made using text and verbal forms.
- The verbal forms, such as ~~Words~~ "shall", "should" and "may" are used in conformance with [11~~10~~] Annex E.
- The procedures should be specified in an asymmetric way, by concentrating on the behaviour on one side of the interface. As guidance, the "controlled" side, rather than the "controlling" side of the interface should be specified.
- The procedures should be specified using the externally observable behaviour, to ease writing of test specifications.
- All normal cases shall be covered. Normal cases are straightforward cases, branches of procedures and combinations of procedures.
- All error cases shall be specified, either explicitly or implicitly. The error cases are all cases that are not considered as normal cases. The error handling should be divided between error handling global to the protocol layer and procedure specific error handling. The procedure specific error cases should be put after the normal cases in each procedure.
- ~~The way to describe procedures is the following:~~

  ~~Protocol errors (global to the protocol layer).~~

  ~~Error handling (global to the protocol layer).~~

  ~~…~~

  ~~1. Procedure <Procedure Name>;~~

  ~~list of normal cases;~~

~~Protocol errors (specific to the procedure);~~

~~Error handling (specific to the procedure).~~

- Redundancy/duplication shall be avoided, in order to avoid problems with later CR, even if this makes the specification initially less readable.

- ~~Mutual cross-referencing shall be introduced: section X that is referred to in section Y should also *say* that it is referred to in section Y.~~

- States and state variables should be used when it provides unambiguity, a way to describe nested procedures and colliding cases.

- Timers, variables and constants and usage of them must be specified.

- Explicit explanation when the action shall be performed is specified in the procedure itself.

- When there are procedural differences between the FDD and TDD modes these should be clearly pointed out using a consequent notation, e.g. “FDD only”, “In TDD, …”.

- ~~The chapter "Default actions upon reception of an IE" is used to avoid duplication of text; this chapter is put at the beginning of the "Message contents to use" text.~~

- When optional ~~(or conditional)~~ IEs are possible in a given message, the meaning of the presence (i.e.: which «function» are activated with the given IE) shall be specified in the procedure for the receiving entity. The requirements on when to include a given IE shall be specified for the transmitting entity. An exception for this rule is when the requirements on the entity is not specified by the protocol.

- ~~The formal values of the IE, e.g., "TRUE" or "FALSE" rather than the coded value, e.g. "1" or "0" shall be used.~~

- Requirements on the content of a message at the sending entity are put before analysis of the message at the receiving entity.

- References to IEs that are parts of another IE is allowed. ~~The notation shall be changed to the so-called "dot-notation" for references to IEs that are parts of another IE.~~

- ~~Names of IEs shall be put between "<IE Name>" quotes, where <IE Name> is the exact name from the tabular format.~~When referring to an IE a formal notation shall be used.

- ~~Square brackets [ ] shall be used for addressing one element of a list.~~

- When referring to a message, a formal notation shall be used ~~"<MESSAGE NAME>" *message* shall be used. Message names are always in upper cases and the word message follows the message name~~.

## 7.1a Specification of algorithms and formulas

When algorithms or formulas are used in the specifications, a formal notation shall be used and mathematical expressions should be used to reduce ambiguity.

- The notation “*OP1* div *OP2*” shall be interpreted as the signed integer result after integer division (truncating any fractional part) of the operand *OP1* with operand *OP2*.

- The notation “*OP1* mod *OP2*” shall be interpreted as the signed remainder after integer division of the operand *OP1* with operand *OP2*.

## 7.2 RRC specific rules

- The specification shall focus on the UE behaviour.

- Only UE timers are normative (when UTRAN timers are present, it is for information).

- The procedure specification text shall specify how the UE shall handle the IEs.

- As much as possible of the UE behaviour shall be tied to reception and non-reception of IEs and included in the subclause "Generic actions upon receipt and absence of an information element", to avoid duplication of text.

- "UTRAN shall" shall be only used when UTRAN behaviour is normative.

- It shall be specified whether timers shall be started when RRC sends the message to lower layers or when the message is effectively sent at the radio interface.

- When referring to messages in the procedure text, the notation “EXAMPLE message” is used (excluding the quotation marks). For example: “The UE shall transmit an RRC CONNECTION REQUEST message”.

- When referring to IEs in the procedure text, the tabular description of IEs should be used as basis. The notation “IE “Example”” is used (including the inner but not the outer quotation marks. Values of IEs are out within quotation marks. For example: “The UE set the IE “Protocol error indicator” to “FALSE” in the RRC CONNECTION REQUEST message”.

- UE performance requirements are considered to be TSG RAN WG2 work. These must be specified only if they are testable.

## 7.3 VoidHandling of DS-41

- ~~Modelling of RRC services is provided by means of primitives.~~

- ~~RRC CN dependent info:~~

- ~~In broadcast message, neighbour cells are described the same way as for GSM neighbour cells (i.e.: in the same SystemInformationBlock but with a tag to indicate CN type or RTT).~~

- ~~In dedicated messages.~~

  ~~a transparent container as NAS info is used to carry ANSI-41;~~

  ~~for PLMN Id and Identities used by the RRC, the CN Type info is used;~~

  ~~NAS binding info is used;~~

- ~~In Paging messages, a tag to indicate CN type is used.~~

- ~~Extensions like handover message to Multicarrier is handled the same way as GSM.~~

# 8 Message specification

## 8.1 VoidSummary of what has been agreed

1) ~~use subset of ASN.1 (compatible with Z.105) for definition of message contents description of protocol messages;~~

2) ~~there is a need for a default encoding, which can be applied in most cases;~~

3) ~~there is a need for a special encoding e.g. by means of CSN.1;~~

4) ~~how to link the message contents description to the different encoding rules needs to be specified;~~

5) ~~ASN.1 definitions can be used within SDL and TTCN parts of the specifications.~~

## 8.2 Definitions

Message descriptions are divided into three levels:

- a logical description, which describes messages and relevant information elements in an easily understandable, semi-formal fashion;

- a message contents description, which describes the messages formally and completely in an abstract fashion ("abstract syntax"); and

- a message encoding, which defines the encoded messages (i.e. what is carried as a bit string, "transfer syntax").

## 8.3 Logical description

The logical description of messages shall be done using tabular format specified in clause 98 of this document, Message contents description.

## 8.4 Message contents description

The message contents descriptions shall be written using ASN.1. The message encoding shall be based on the ASN.1 description.

## 8.5 Compilability of the transfer syntax

The transfer syntax should allow as automatic as possible compilers that transform between a sequence of received bits ~~into~~ and a sequence of IEs that can be utilised by the protocol machine. ~~CSN.1~~Specialised encoding may be used. A link between message contents description and transfer syntax needs to be specified.

## 8.6 Efficiency/Compactness

The transfer syntax should allow minimising the size of messages if so necessary. It should allow protocol dependant optimisations.

## 8.7 Evolvability/Extensibility

The message contents description shall allow the evolution of the protocol.

~~The transfer syntax shall keep the same level of compactness as the initial design.~~

## 8.8 Inter IE dependency

The message contents description shall allow that presence of IEs depends on values in previous IEs.

The description of messages should avoid dependency between values in different IE. Indeed, it would mean that values are not independent and that there is a redundancy.

## 8.9 Intra IE dependency

The abstract and transfer syntaxes shall allow that, within an IE, some fields depend on previous ones.

## 8.10 Support of error handling

The syntax used should support optional IEs, default values, partial decoding, "comprehension required" and extensibility as defined above.

# 9 Usage of tabular format

A protocol specification should include a 'Tabular description' subclause, including:

- A message description subclause;

- An IE description subclause.

# 9.1 Tabular description of messages and IEs

## 9.1.1 Message description

A 'Message description' subclause includes one subclause per message.

A message is described with, in this order:

- A general description, including the flow the message belongs to (e.g., SAP, direction, …); this indirectly points to the message header description, which is not described again for each message;
- A table describing a list of information elements;
- Explanatory clauses, mainly for describing textually conditions for presence or absence of some IEs.

### 9.1.1.1 The general description

### 9.1.1.2 The Information Element table

The table is composed of 6~~5~~ columns, labelled and presented as shown below.

| IE/Group Name | Need | Multi | Type and reference | Semantics description | Version |
|---|---|---|---|---|---|
| | | | | | |

NOTE: Indentations are used to visualise the embedding level of an "IE/Group" ~~or "Type and reference"~~.

Indentations are explicitly written with the character ">", one per level of indentation. Indentations of lines can be found in the IE/Group Name ~~and Type and reference~~ column~~s~~.

Each line corresponds either to an IE or to a group. A group includes all the IEs in following lines until, and not including, a line with the same indentation as the group line.

Dummy groups can be used for legibility: the following IE/Group has the same indentation. For such dummy groups, the Need and Multi columns are meaningless and should be left empty.

~~The "Type and reference" column is not filled in the case of a group line and must be filled for "IE/Group Name" column.~~

#### 9.1.1.2.a IE/Group Name column

This column gives the local name of the IE or of a group of IEs. This name is significant only within the scope of the described message, and must appear only once in the column at the same level of indentation. It is a free text, which should be chosen to reflect the meaning of the IE or group of IEs. This text is to be used ~~followed by the key word IE, the whole enclosed between quotes [or in italics]~~ to refer to the IE or the group of IEs in the procedure ~~specificational description~~as described in clause 7..

The first word 'choice' has a particular meaning, and must not be used otherwise.

#### 9.1.1.2.1 Need and multiplicity (Multi) columns

These columns provide most of the information about the presence, absence and number of ~~copy~~instance of the IE (in the message or in the group) or group of IEs. The different possibilities for these columns are described one by one.

The meaning of the 'need' column is summarised below:

MP Mandatorily present.

A value for that information is always needed, and no information is provided about a particular default value. If ever the transfer syntax allows absence (e.g., due to extension), then absence leads to an error diagnosis.

MD Mandatory with default value.

A value for that information is always needed, and a particular default value is mentioned (in the "Semantical information" column). This opens the possibility for the transfer syntax to use absence or a special pattern to encode the default value.

CV Conditional on value.

A value for that information is needed (presence needed) or unacceptable (absence needed) when some conditions are met that can be evaluated on the sole basis of the content of the message.

If conditions for presence needed are specified, the transfer syntax must allow for the presence of the information. If the transfer syntax allows absence, absence when the conditions for presence are met leads to an error diagnosis.

If conditions for absence needed are specified, the transfer syntax must allow encoding the absence. If the information is present and the conditions for absence are met, an error is diagnosed.

When neither conditions for presence or absence are met, the information is treated as optional, as described for "OP".

CH Conditional on history.

A value for that information is needed (presence needed) or unacceptable (absence needed) when some conditions are met that must be evaluated on the basis of information obtained in the past (e.g., from messages received in the past from the other party).

If conditions for presence needed are specified, the transfer syntax must allow for the presence of the information. If the transfer syntax allows absence, absence when the conditions for presence are met leads to an error diagnosis.

If conditions for absence needed are specified, the transfer syntax must allow encoding the absence. If the information is present and the conditions for absence are met, an error is diagnosed.

When neither conditions for presence or absence are met, the information is treated as optional, as described for "OP".

OP Optional.

The presence or absence is significant and modifies the behaviour of the receiver. However whether the information is present or not does not lead to an error diagnosis.

#### 9.1.1.2.1.1 Mandatory

| IE/Group Name | Need | Multi | Type and reference | Semantics description | Version |
|---|---|---|---|---|---|
| Name | MP | | | | |
| Name | MD | | | (default value is indicated) | |

The multiplicity column ~~must~~ may be left empty (see 9.1.1.2.1.5).

For mandatory IEs, the rules are as follows, applied on the number of instances given by the multiplicity column (leaving the multiplicity column empty means one and only one instance):

For an IE not belonging to a group MP indicates that ~~one and only one copy~~the number of instances as given by the multiplicity column of "Name IE" is necessary in the message.

For a group not belonging to another group, MP means that the number of instances as given by the multiplicity column~~one and only one copy~~ of the "Name group" is necessary in the message.

For an IE or a group belonging to another group, MP means that if the parent group is present, then the number of instances as given by the multiplicity column~~one and only one copy~~ of the "Name group" or "Name IE" is necessary in the embedding group.

For an IE not belonging to a group MD indicates that the number of instances as given by the multiplicity column ~~one and only one value~~ for information 'Name IE' is necessary in the message, and that a special value (the default value) exists, for all instances or individual instances, and is mentioned in the 'Semantics description' column.

For a group not belonging to another group, MD means that the number of instances as given by the multiplicity column ~~one and only one value~~ for information structure 'Name group' is necessary in the message, and that a special value (the default value) exists, for all instances or individual instances, and is mentioned in the 'Semantics description' column.

For an IE or a group belonging to another group, MD means that if the parent group is present, then the number of instances as given by the multiplicity column~~one and only one value~~ for information structure 'Name group' or information 'Name IE' is necessary in the embedding group, and that a special value (the default value) exists and is mentioned in the 'Semantics description' column.

The default value might be fixed by the standard, or conditional to the value of some other IE or IEs, or conditional on information obtained in the past.

#### 9.1.1.2.1.2 Optional

| IE/Group Name | Need | Multi | Type and reference | Semantics description | Version |
|---|---|---|---|---|---|
| Name | OP | | | | |

The multiplicity column ~~is~~ may be empty (see 9.1.1.2.1.5).

This indicates that the number of instances as given by the multiplicity column of the 'Name IE' or 'Name group' is not necessary in the message or the embedding group, and that the sender can choose not to include it.

#### 9.1.1.2.1.3 Conditional

| IE/Group Name | Need | Multi | Type and reference | Semantics description | Version |
|---|---|---|---|---|---|
| | CV *cond* | | | | |
| | CH | | | | |

The multiplicity column ~~is~~ may be empty (see 9.1.1.2.1.5).

CV indicates that the requirement for presence o~~f~~ absence of the number of instances as given by the multiplicity column of the IE or group of IE depends on the value of some other IE or IEs, and/or on the message flow (e.g., channel, SAP). In the CV case, the condition is to be described in a textual form in an explanatory clause. *cond* stands for a free text that is used as a reference in the title of the explanatory clause. In the CH case, the condition is described in the procedural section.

~~When condition is met may means~~ The result of evaluating the condition (if the condition is met or not) may mean that the IE is:

- Mandatorily present.
- Mandatorily absent.
- Optional.
- Absent, but optional (this is meaningful only for extension).

The error handling shall be specified in the protocol for the cases when the requirement for presence or absence of an IE indicated by the condition is not followed.

#### 9.1.1.2.1.4 Choice

This is particular group of at least two children.

| IE/Group Name | Need | Multi | Type and reference | Semantics description | Version |
|---|---|---|---|---|---|
| Choice *name* | NOTE 1 | | | | |
| *>Name1* | | | | | |
| *>Name2* | | | | | |

NOTE 1: The Need column shall take one of the values “MP”, “MD”, “OP”, “CV *cond*” or “CH *cond*”.

A 'choice' group is distinguished from standard groups by the use of 'choice' as first word in the name.

The Need column shall and the Multi columns ~~are~~ may be filled ~~normally~~ for the group line. They are not filled for the children lines: the implicit value is conditional, one condition being that one and only one of the children is present if the group is present.

If additional conditions (depending on the value of some other IE or IEs, and/or on the message flow) exist for the choice, they are explained in an explanatory clause.

#### 9.1.1.2.1.5 Sets

In general, this indicates that more than one ~~copy~~instance of an IE/Group might be necessary in the message.

The two lines below indicate different allowed alternatives.

| IE/Group Name | Need | Multi | Type and reference | Semantics description | Version |
|---|---|---|---|---|---|
| Name | MP | nn..pp | | | |
| Name | MP | nn..indefinite | | | |
| Name | MP | nn..sym2 | | | |
| Name | MP | sym1..pp | | | |
| Name | MP | sym1..sym2 | | | |
| Name | MD | nn..pp | | | |
| Name | MD | nn..indefinite | | | |
| Name | MD | nn..sym2 | | | |
| Name | MD | sym1..pp | | | |
| Name | MD | sym1..sym2 | | | |
| Name | OP | nn..pp | | | |
| Name | OP | nn..indefinite | | | |
| Name | OP | nn..sym2 | | | |
| Name | OP | sym1..pp | | | |
| Name | OP | sym1..sym2 | | | |
| Name | Cx cond | nn ..pp | | | |
| Name | Cx cond | nn..indefinite | | | |
| Name | Cx cond | nn..sym2 | | | |
| Name | Cx cond | sym1..pp | | | |
| Name | Cx cond | sym1..sym2 | | | |

Where *nn* and *pp* stand for positive integers, and *sym1* and *sym2* for symbolic names. The Need column can be empty, CV or CH.

The notation '..' can be replaced with the same meaning by 'to'.

This indicates that a number of ~~copies~~instances of the IE/Group are necessary in the message/embedding group. The order is significant. The reference should use the bracket notation (e.g., 'Name[1] IE') to refer to a specific ~~copy~~instance; numbering starts by 1.

The *nn..pp* case indicates that the number of ~~copies~~instances is between nn and pp, inclusively. This means that *nn* ~~copies~~instances are necessary in the message, that additional *pp-nn* ~~copies~~instances are optional and meaningful, and that ~~copies~~instances after the *pp*th are not necessary.

The number *nn* is positive or null. The number *pp* must be equal or greater than *nn*. The 1..1 case should be avoided and ~~,~~ instead the Multi column should be left empty to indicate one and only one instance. ~~and a MP indication used instead~~. ~~Similarly, t~~The 0..1 case combined with MP should be avoided and replaced by 1..1 with an OP indication.

The *nn*..indefinite case indicates that the number of ~~copies~~instances is *nn* or greater. This means that *nn* ~~copies~~instances are necessary in the message, and that additional ~~copies~~instances are optional and meaningful. The number *nn* is positive or null. It is however allowed that the transfer syntax puts some practical limits on the maximum number of ~~copies~~instances.

The use of a symbolic name for one or the other of the range bounds indicates that the value is given in a textual clause. This is necessary the case when the bound depends is conditional to the value of some other IE or IEs.

The 'Need' column is set to MP, MD, CV or CH and interpreted as described in 9.1.1.2.1.1-9.1.1.2.1.3 applied to the whole set. ~~followed by a condition name to indicate that the number of necessary or optional copies is conditional to the value of some other IE or IEs, or on the flow (CV case) or to information obtained in the past (CH case). An explanatory clause describes the condition. Otherwise, the column is left empty.~~

### 9.1.1.2.2 Type and reference column

This column is not filled for groups and must be filled for IEs.

This column includes the reference to a more detailed abstract description of the IE. This includes:

a) A reference to a subclause in the Information Element Description clause in the same document; typically the subclause number and titles are given, and if possible this should be a hypertext link;

b) A reference to another document, and to a subclause in the Information Element Description clause in the indicated document; typically only the subclause title is indicated;

~~c) A reference to a subclause of this document, with possibly additional information as described.~~

### 9.1.1.2.3 Semantics description

Filling this column is optional. It should be use to clarify the meaning of the IE or group of IE, as a summary of their use as described in the procedural part.

### 9.1.1.2.4 Expressing differences between FDD and TDD modes

If a PDU or a structured information element contain information elements whose Need value is different for FDD and TDD modes or if a certain structured information element is completely different for the two modes, a choice group should be used.

| IE/Group Name | Need | Multi | Type and reference | Semantics description | Version |
|---|---|---|---|---|---|
| Choice ~~systemtype~~mode | MP | | | | |
| >FDD | | | | | |
| >>element1 | MP | | | | |
| >>element2 | OP | | | | |
| >TDD | | | | | |
| >>element3 | OP | | | | |
| >>element4 | MP | | | | |

### 9.1.1.2.5 Version column

When an information element is added from one version to a latter one, the version in which the element is added (e.g.: REL-4, REL-5) is included in the version column.

When a new CHOICE group is added from one version to a later one, the version in which the group is added is included in the version column of all new rows. If some of the information elements in the new CHOICE group were included in the older version (but not inside a CHOICE group), the version column is not updated for those information elements (see also the example at the end of this clause).

When an existing CHOICE group is extended from one version to a later one to include more options, the version in which the new options are added is included in the version column of the rows describing the new options.

When the type of an information element is modified from one version to a later one to include more values, the version in which the modification takes place is included in the version column, and the new values are indicated in that column. By convention the version column is left blank for the present release.

The example below shows how the version column is used for the cases described above. The first table shows an example of a Release '99 table:

| IE/Group Name | Need | Multi | Type and reference | Semantics description | Version |
|---|---|---|---|---|---|
| *Element1-99* | MP | | Type1 | | |
| *Element2-99* | MP | | Type2 | | |
| *CHOICE choice1-99* | MP | | | | |
| *>first* | | | | | |
| *>>Element3-99* | MP | | Type3 | | |
| *>second* | | | | | |
| *>>Element4-99* | MP | | Type4 | | |
| *Element5-99* | MP | | Enumerated(a,b) | | |

The second table shows extensions of the above table in Release 4, and where the REL-4 in the version column shall be included:

| IE/Group Name | Need | Multi | Type and reference | Semantics description | Version |
|---|---|---|---|---|---|
| *Element1-99* | MP | | Type1 | | |
| *Element6-r4* | MP | | Type6 | | REL-4 |
| *CHOICE choice2-r4* | MP | | | | REL-4 |
| *>old* | | | | | REL-4 |
| *>>Element2-99* | MP | | Type2 | | |
| *>new* | | | | | REL-4 |
| *>>Element7-r4* | MP | | Type7 | | REL-4 |
| *CHOICE choice1-99* | MP | | | | |
| *>first* | | | | | |
| *>>Element3-99* | MP | | Type3 | | |
| *>second* | | | | | |
| *>>Element4-99* | MP | | Type4 | | |
| *>third* | | | | | REL-4 |
| *>>Element8-r4* | MP | | Type8 | | REL-4 |
| *Element5-99* | MP | | Enumerated(a,b,c) | | Value c is included in REL-4. |

### 9.1.1.3 Explanatory clauses

This includes the subclauses needed to elaborate conditions ~~and symbolic names (e.g., range bounds)~~. There must be one explanatory clause for each named condition~~, and for each symbolic name~~. The text must give the information sufficient to decide whether the IE/group is to be included or not~~, or the value of the symbolic name~~.

## 9.1.2 IE type description

This describes IE types referred elsewhere, either in the description of a message or in the description of another IE type. The description of an IE type must be as generic as possible, i.e., independent of any specific use. A type should as far as possible not be defined in multiple places in a specification.

An 'IE description' subclause includes one subclause per IE type.

The description of an IE type is done as a table similar to that used for the description of messages.

| IE/Group Name | Need | Multi | Type and reference | Semantics description | Version |
|---|---|---|---|---|---|
| | | | | | |

The different columns are filled ~~exactly~~ as message description columns are filled~~.~~ with the addition that in the IE Type and reference column also the use of a basic types defined in subclause 9.2 in this document is allowed. These basic types shall be considered as pre-defined and for those no reference is necessary. For IE type descriptions, explanatory clauses should also be used as described in subclause 9.1.1.3.

# 10 Usage of ASN.1

The following clauses contain guidelines for specification of protocol messages with ASN.1.The purpose of ASN.1 is to make it possible to specify message contents description of a message (i.e. what is the contents of a message) separately from its transfer syntax (i.e. how a message is encoded for transmission). ~~The features that ASN.1 provides include specification of:~~

- ~~Extensibility (both structural and extension of value set).~~
- ~~Optional IEs and values (see the clauses 10.2.2 and 10.3.10).~~
- ~~Default values (see the clauses 10.2.2 and 10.3.10).~~
- ~~Comprehension required (see the clause 10.2.4).~~
- ~~Inter/Intra IE dependency (see the clause 10.3.10).~~
- ~~Specification of partial decoding (see the clause 10.2.5).~~

The clause 11 specifies how message transfer syntax is specified. It should be noted that importance of some transfer syntax properties must be determined early during specification because of their effect on message contents description specification possibilities. The properties are **compactness** and **extensibility.** If extreme compactness is required then extensibility must be restricted. If good extensibility is required then compromises must be done regarding compactness. The sections concerning these issues are marked in the following clauses as **COMPACTNESS** and **EXTENSIBILITY**.

Identifiers that could be keywords of ~~another~~ some language (e.g.: SDL, C, ASN.1, JAVA, C++, …) should be avoided.

In the current version of the ASN.1 specifications, user-defined constraints are not used.

## 10.1 Message level

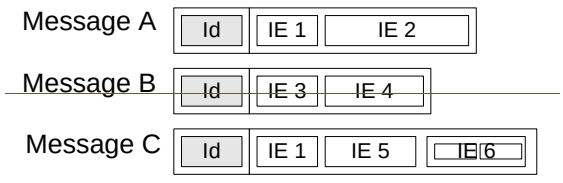
### 10.1.1 Messages

~~It is presumed that messages share the same structure, namely that they contain an identification part and a contents part. An identification part contains an IE that identifies a message among all messages in some context.~~

~~A contents part contains message specific IEs.~~

~~IE is a list of components.~~

~~Example: A protocol layer XYZ contains three messages: A, B and C. The structure of the messages is as presented in the figure 10.1.1.1.~~




**~~Figure 10.1.1.1: Three example messages~~**

~~Messages are specified using ASN.1 [1]. There are three ASN.1 types, *MessageA*, *MessageB* and *MessageC*, which contain definitions for the contents of the above messages. The mapping between the message contents types and message identifiers is as follows:~~

| Message id | Type of message contents |
|---|---|
| 1 | MessageA |
| 2 | MessageB |
| 3 | MessageC |

New message types will be introduced in the future.

In cases where different PDUs have different identification schemes it is possible to apply this categorisation for a set of PDUs that share the same identification scheme.

## 10.1.2 Message definition

In order to capture information in the previous clause the following three things must be defined:

1. A structure for the table.
2. The table itself.
3. A generic message structure that can contain both message identifier IE and message contents IEs (i.e. id 1 + *MessageA*, id 2 + *MessageB*, id 3 + *MessageC*).

The table structure is defined as follows using ASN.1 classes [2]:

```
XYZ-MESSAGE ::= CLASS {
    &id      MessageId UNIQUE,
    &Type
}
WITH SYNTAX {
    &id &Type
}

MessageId ::= INTEGER (0..63)
```

The table is defined as follows:

```
XYZ-Messages XYZ-MESSAGE ::= {
    { messageA-id MessageA } |
    { messageB-id MessageB } |
    { messageC-id MessageC } ,
    ...                 -- Extension marker => additional messages
                        -- can be introduced.
}

messageA-id MessageId ::= 1
messageB-id MessageId ::= 2
messageC-id MessageId ::= 3
```

The following type represents the generic message structure that can carry values of the messages specified in the *XYZ-Messages* table.

```
XYZ-Message ::= SEQUENCE {
    id        XYZ-MESSAGE.&id    ({XYZ-Messages)},
              -- MessageId: 1, 2 or 3

    contents     XYZ-MESSAGE.&Type ({XYZ-Messages}{@id})
              -- id=1 => MessageA, id=2 => MessageB, id=3 => MessageC
}
```

The above definition means that if *id* is 1 then the *Message* type could be interpreted as the following type:

```
XYZ-Message ::= SEQUENCE {
    id       MessageId,    -- 1
    contents    SEQUENCE {
        ie1     IE1,
        ie2     IE2
    }
}
```

If *id* is 2 then the type could be interpreted as the following type:

```
XYZ-Message ::= SEQUENCE {
    id       MessageId,    -- 2
    contents    SEQUENCE {
        ie3     IE3,
        ie4     IE4
    }
}
```

## 10.1.3 Messages and ASN.1 modules

ASN.1 definitions shall be placed in ASN.1 modules such that definitions in a module form a logical unit. For example PDUs definitions for one protocol layer could be in one ASN.1 module and IE definitions in another.

The tagging mode for the modules shall be "AUTOMATIC TAGS". Note that "AUTOMATIC TAGS" is not relevant for PER.

Example: A message definition module for the XYZ protocol layer.

```
XYZ-Messages DEFINITIONS AUTOMATIC TAGS ::=

BEGIN

XYZ-Messages XYZ-MESSAGE ::= {
    { messageA-id MessageA } |
    { messageB-id MessageB } |
    { messageC-id MessageC } ,
    ...                 -- Additional messages can be introduced.
}

MessageA ::= SEQUENCE {
    -- Message contents
}

messageA-id MessageId ::= 1

MessageB ::= SEQUENCE {
    -- Message contents
}

messageB-id MessageId ::= 2

MessageC ::= SEQUENCE {
    -- Message contents
}

messageC-id MessageId ::= 3

END
```

## 10.1.4 Messages and SDL

The identifiers *messageA-id*, *MessageA*, *messageB-id*, etc. can be used in descriptive SDL when protocol behaviour is specified. Note that classes and objects cannot be referenced in SDL96 but are allowed in SDL2000. Types and values

however can be imported to SDL definitions. The figures below contain some examples about usage of ASN.1 in SDL specifications.

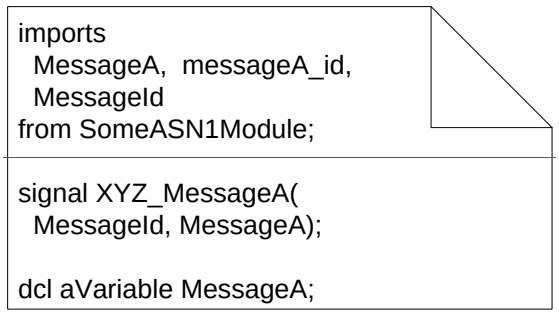



**Figure 10.1.4.1: Import and use of ASN.1 definitions in SDL**

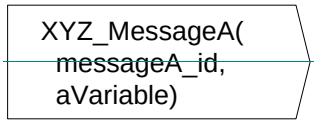



**Figure 10.1.4.2: Sending of a message id and contents**

## 10.2 Information element level

Messages consist of information elements.

The following ASN.1 message types are used in the following clauses.

```
MessageA ::= SEQUENCE {
    ie1 IE1,                -- A mandatory IE.

    ie2 IE2 OPTIONAL,       -- An optional IE.

-- Extensions from there
    ExtensionMarker SEQUENCE {} OPTIONAL

}

MessageB ::= SEQUENCE {
    ie3 IE3
        (CONSTRAINED BY {-- ComprehensionRequired(is for receiver) --}
        !comprehensionRequiredFailure) ,

    ie4 IE4 DEFAULT 0,      -- An optional IE with a default value.

-- Extensions from there
    ExtensionMarker SEQUENCE {} OPTIONAL
}

MessageC ::= SEQUENCE {
    ie1 IE1
        (CONSTRAINED BY {-- PartialDecoding(OnErrorIgnoreRest) --}
        !partialDecodingFailure)
        OPTIONAL,

    ie5 IE5
        (CONSTRAINED BY {-- PartialDecoding(OnErrorIgnoreRest) --}
        !partialDecodingFailure)
        OPTIONAL,

-- Extensions from there
    ExtensionMarker SEQUENCE {
        ie6 IE6
            (CONSTRAINED BY {-- PartialDecoding(OnErrorIgnoreRest) --}
            !partialDecodingFailure)
            OPTIONAL        -- A new IE
  } OPTIONAL
```

```
}

-- Error codes
comprehensionRequiredFailure INTEGER ::= 1
partialDecodingFailure       INTEGER ::= 2
```

## 10.2.1 Message contents

A message contents structure is defined using a sequence type (10.3.10).

Example: *MessageA*, *MessageB* and *MessageC* are message contents structures.

## 10.2.2 Optional IEs and default values

An IE can be marked as optional.

**COMPACTNESS:** Optional IEs shall be after mandatory ones. When ASN.1 is used with PER, this requirement is not relevant.

When the extension "SEQUENCE {} OPTIONAL" is used, the sender shall never indicate that the field is present.

Example: *MessageA.ie2* is an optional IE.

```
ie2 IE2 OPTIONAL
```

An IE can be marked as being optional and having a default value. In those cases a missing optional IE may be understood as having a certain value hence a defined meaning.

Example: *MessageB.ie4* is an optional IE with a default value.

```
ie4 IE4     DEFAULT 0
```

## 10.2.3 New IEs

**EXTENSIBILITY:** If new IEs will be added to a message then the message contents structure must be specified as extensible using the ellipsis notation (...). New IEs shall be added after the extension marker. New IEs shall be optional or shall have default values.

Example: *MessageC.ie6* is an additional optional IE.

```
    ...,
    ie6 IE6     OPTIONAL
```

## 10.2.4 Comprehension required

"Comprehension required" requirement can be associated with an IE. It means that after an IE value has been decoded then the value is validated. Failure in validation causes rejection of the message.

The requirement is specified as an extension to ASN.1 by using user defined constraints [3]. The comment part of the constraint shall be of the form:

*ComprehensionRequired(<additional constraint>)*

where <additional constraint> specifies the rule that the IE must satisfy.

Example: The *MessageB* is a broadcast message. The *ie3* IE contains recipient addresses. It is not until the addresses have been decoded when a receiver can decide whether it should decode the rest of the message or not.

```
    ie3 IE3
```

```
        (CONSTRAINED BY {-- ComprehensionRequired(is for receiver) --}
        !comprehensionRequiredFailure) ,
```

## 10.2.5 Partial decoding

"Partial decoding" means that a PDU can be decoded in parts. One part forms a complete value that can be separated from other parts. A decoding error in a part does not invalidate previously decoded parts. Subsequent parts are however invalidated.

"Partial decoding" is specified as an extension to ASN.1 using user defined constraints. The comment of constraint shall be of the form:

*PartialDecoding(<OnErrorClause>)*

where <OnErrorClause> specifies action in case of a decoding error. The possible alternatives are:

- OnErrorIgnoreRest: End decoding, ignore rest of the message.

Example: The *MessageC* is a multipurpose message. The IEs *ie1*, *ie5* and *ie6* are independent of each other.

```
    ie1 IE1
        (CONSTRAINED BY {-- PartialDecoding(OnErrorIgnoreRest) --}
        !partialDecodingFailure)
```

## 10.2.6 Error specification

An error specification can be associated with user-defined constraints.

A simple integer value can be associated with an exception specification or as elaborate structured value as needed.

Example: If decoding of *ie1* fails then decoder returns the error code *partialDecodingFailure*.

```
    ie1 IE1
        (CONSTRAINED BY {-- PartialDecoding(OnErrorIgnoreRest) --}
        !partialDecodingFailure)
```

# 10.3 Component level

Information elements consist of components.

The following ASN.1 types shall be used at the component level:

- Boolean (10.3.4)
- Integer (10.3.5)
- Enumerated (10.3.6)
- Bit string (10.3.7)
- Octet string (10.3.8)
- Null (10.3.9)
- Sequence (10.3.10)
- Sequence-of (10.3.11)
- Choice (10.3.12)
- Character string types (10.3.13)

### 10.3.1 Extensibility

**COMPACTNESS:** In the component level use of ASN.1 extensibility is forbidden unless otherwise stated in the following clauses.

### 10.3.2 Comprehension required

"Comprehension required" can be applied to components of sequence types, alternatives of choice types and elements of sequence-of types. See 10.2.4.

### 10.3.3 Partial decoding

"Partial decoding" can be applied to components of sequence types, alternatives of choice types and elements of sequence-of types. See 10.2.5.

### 10.3.4 Boolean

Example: A simple boolean type.

```
Flag ::= BOOLEAN

setFlag Flag ::= TRUE
```

### 10.3.5 Integer

An integer type should be constrained.

**COMPACTNESS:** An integer type shall be constrained to have a finite value set. The value set can be either continuous or non-continuous.

Named numbers can be associated with an integer type.

**COMPACTNESS, EXTENSIBILITY:** If an integer type needs to be extended in the future then two value sets must be defined:

- A value set that specifies the values that can be sent in the current protocol version.
- A value set that specifies all the possible values that can be received now and in the future.

The former value set is specified in a user-defined constraint. The comment part shall be of the form:

*Send(<value set>)*

The latter form is specified using a normal constraint, e.g. a value range constraint.

Examples: Integer types and values.

```
Counter          ::= INTEGER (0..255)      -- 0 <= Counter value <= 255

SparseValueSet ::= INTEGER (0|3|5|6|8|11)

SignedInteger   ::= INTEGER (-10..10)

-- idle stands for value 0.
Status           ::= INTEGER { idle(0), veryBusy(3) } (0..3)

-- Send values 0..3 but be prepared to receive values 0..15.
Extensible      ::= INTEGER (0..15)(CONSTRAINED BY {-- Send(0..3) --})

initialCounter Counter          ::= 0

zero             SparseValueSet ::= 0
```

```
initialStatus  Status          ::= idle
```

## 10.3.6 Enumerated

The EnumerationItem shall not contain a named number (e.g.: foo (3)).

The list of enumerated values specifies the value set for an enumerated type.

**COMPACTNESS, EXTENSIBILITY:** If an enumerated type needs to be extended in the future then two value sets must be defined as in case of integer types.

NOTE: An integer type with named numbers can be used as on alternative to an enumerated type.

Example: Enumerated types and value.

```
Enum          ::= ENUMERATED { a, b, c, d }

-- Send values a, b, c or d but be prepared to receive values
-- a, b, c, d, spare4, spare5, spare6 and spare7.
ExtendedEnum ::= ENUMERATED { a, b, c, d, spare4, spare5, spare6, spare7 }
    (CONSTRAINED BY {-- Send(a|b|c|d) --})

aEnum Enum ::= a
```

## 10.3.7 Bit string

A size constraint shall be specified. It shall be finite.

Named bits can be associated with a bit string type.

Example: Bit string types and values.

```
FixedLengthBitStr     ::= BIT STRING (SIZE (10))

VariableLengthBitStr ::= BIT STRING (SIZE (0..10))

BitFlags              ::= BIT STRING { a(0), b(1), c(2), d(3)} (SIZE (4))

fix FixedLengthBitStr     ::= '0001101100'B

var VariableLengthBitStr ::= '0'B

flg BitFlags              ::= { a, c, d } -- '1011'B
```

## 10.3.8 Octet string

A size constraint shall be specified. It shall be finite.

Example: Octet string types and values.

```
FixedLengthOctetStr     ::= OCTET STRING (SIZE (10))

VariableLengthOctetStr ::= OCTET STRING (SIZE (0..10))

UpperLayerPDUSegment    ::= OCTET STRING (SIZE (1..512))

fix FixedLengthOctetStr     ::= '0102030405060708090A'H

var VariableLengthOctetStr ::= 'FF'H
```

## 10.3.9 Null

A null type has only one value, NULL.

Example: Null type as an alternative type of a choice type.

```
IE ::= CHOICE {
    doThis    ThisArg,
    doThat    ThatArg,
    doNothing    NULL
}
```

## 10.3.10 Sequence

A sequence type is a record. Components of a sequence type can be optional or they can have default values. Optional components and components with default values should be after mandatory components.

Inner subtyping can be used to force an optional component to be present or absent in a derived type.

If an optional component is conditionally present or absent then the condition shall be specified in a user defined constraint of the form:

*Condition(<condition expression>)*

<condition expression> shall be such that both sender and receiver are able to evaluate it before a conditional component is encoded or decoded.

"Comprehension required" can be associated with a component of a sequence type.

"Partial decoding" can be associated with a component of a sequence type.

**EXTENSIBILITY:** A sequence type can be marked as extensible. Example: Sequence types and values.

```
Record ::= SEQUENCE {
    flag        Flag,
    counter Counter,
    bitFlags    BitFlags        OPTIONAL,
    extEnum ExtendedEnum    DEFAULT a
}

DerivedRecord ::= Record (WITH COMPONENTS {
                ...,
                bitFlags    PRESENT
                })

RecordWithConditionalComponent ::= SEQUENCE {
    mand        INTEGER (0..7),
    opt     BOOLEAN OPTIONAL,
    cond        BOOLEAN OPTIONAL
} ( WITH COMPONENT {mand(7), cond PRESENT} | WITH COMPONENT {cond ABSENT} )

aRecord Record ::= {
    flag        TRUE,
    counter 100
}

anotherRecord DerivedRecord ::= {
    flag        TRUE,
    counter 100,
    bitFlags    '0101'B     -- bitFlags must be present
}
```

### 10.3.11 Sequence-of

A sequence-of type is a list of some element type. A size constraint shall be specified. It shall be finite.

"Comprehension required" can be associated with an element of a sequence-of type.

"Partial decoding" can be associated with an element of a sequence-of type.

Example: Sequence-of types and values.

```
FixedLengthList       ::= SEQUENCE (SIZE (10)) OF Record

VariableLengthList    ::= SEQUENCE (SIZE (0..10)) OF Status

UpperLayerPDUSegments ::= SEQUENCE (SIZE (1..10)) OF UpperLayerPDUSegment

aList VariableLengthList ::= { idle, 1, 2, veryBusy, 2, 1, idle }
```

### 10.3.12 Choice

A choice type is a variant record. Only one alternative component can be selected.

Inner subtyping can be used to force an alternative to be selected in a derived type.

"Comprehension required" can be associated with an alternative component of a choice type.

"Partial decoding" can be associated with an alternative component of a choice type.

**EXTENSIBILITY:** A choice type can be marked as extensible.

Example: Choice type and value.

```
VariantRecord ::= CHOICE {
    flag        Flag,
    counter Counter,
    extEnum ExtendedEnum
}

aVariantRecord VariantRecord ::= flag : FALSE
```

### 10.3.13 Restricted character string types

A size constraint shall be specified. It shall be finite.

It should specified the permitted alphabet for compactness reasons (see examples in PER [5]).

Example: Character string types.

```
FixedStr  ::= IA5String (SIZE (10))

VarStr    ::= IA5String (SIZE (1..10))

FixedWStr ::= BMPString (SIZE (10))

VarWStr   ::= BMPString (SIZE (1..10))
```

## 10.3.14 IEs and ASN.1 modules

If an IE or a component field within an IE is a parameter from another protocol layer then that type for such a field should be defined in another module. In this way there is a clear separation of definitions that are specific to different protocol layers.

Example: The XYZ protocol message *MessageC* contains an IE, which contains an OPQ protocol layer specific field *parameter1*. Type for the field is imported from that OPQ specific module.

```
XYZ-Messages DEFINITIONS AUTOMATIC TAGS ::=

BEGIN

IMPORTS
    OPQParameter      -- OPQParameter is not defined within XYZ-Messages
                      -- module.
FROM OPQ-DataTypes;

MessageC ::= SEQUENCE {
    -- Other IEs.
    ie6 IE6 OPTIONAL
}
-- Other definitions ...

IE6 ::= SEQUENCE {
    parameter1  OPQParameter,     -- Imported definitions can be
                                  -- referred to.
    parameter2  XYZParameter
}

XYZParameter ::= INTEGER (0..255)

END
```

Example: The OPQ protocol layer specific module exports *OPQParameter* type so that other modules can refer it.

```
OPQ-DataTypes DEFINITIONS AUTOMATIC TAGS ::=

BEGIN


OPQParameter ::= INTEGER (0..7)

END
```

# Annex A:
# Change history

| Change history | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Date** | **TSG #** | **TSG Doc.** | **CR** | **Rev** | **Subject/Comment** | **Old** | **New** |
| 12/1999 | RP-06 | RP-99659 | - | | Approved at TSG-RAN #6 and placed under Change Control | - | 3.0.0 |
| 03/2000 | RP-07 | RP-000048 | 001 | 2 | Further clarifications on specialised encoding | 3.0.0 | 3.1.0 |
| | RP-07 | RP-000048 | 003 | 1 | Modification of the 'presence' column specification in tabular format, and other editorial modifications | 3.0.0 | 3.1.0 |
| | RP-07 | RP-000048 | 005 | | Editorial corrections on subclause 11.2 | 3.0.0 | 3.1.0 |
| | RP-07 | RP-000048 | 006 | | Improvement of integers and enumerated, and introduction of reals and octet strings | 3.0.0 | 3.1.0 |
| 12/2000 | RP-10 | RP-000575 | 007 | | Extension rules for supporting future releases | 3.1.0 | 3.2.0 |
| 03/2001 | RP-11 | RP-010033 | 008 | | Description of backward compatibility consideration rule for RANAP, SABP, RNSAP and NBAP ASN.1 | 3.2.0 | 3.3.0 |
| | RP-11 | RP-010033 | 009 | | Usage of the Version column | 3.2.0 | 3.3.0 |
| | RP-11 | RP-010033 | 010 | 1 | Clean-up | 3.2.0 | 3.3.0 |
| | RP-11 | RP-010033 | 011 | | Recommendations on the use of the extension mechanism | 3.2.0 | 3.3.0 |
| | RP-11 | - | - | | Upgrade to Release 4 - no technical change | 3.3.0 | 4.0.0 |

# Annex B (informative): Usage of ASN.1

NOTE: The text in this annex should be seen as illustration of how ASN.1 can be used and do not necessarily reflect how ASN.1 is used in the RAN specifications.

The following clauses contain guidelines for specification of protocol messages with ASN.1.The purpose of ASN.1 is to make it possible to specify message contents description of a message (i.e. what is the contents of a message) separately from its transfer syntax (i.e. how a message is encoded for transmission). The features that ASN.1 provides include specification of:

- Extensibility (both structural and extension of value set).
- Optional IEs and values (see the clauses ~~B10~~.2.2 and ~~B10~~.3.10).
- Default values (see the clauses ~~10~~B.2.2 and ~~B10~~.3.10).
- Comprehension required (see the clause ~~10~~B.2.4).
- Inter/Intra IE dependency (see the clause ~~10~~B.3.10).
- Specification of partial decoding (see the clause ~~10~~B.2.5).

The clause 11 specifies how message transfer syntax is specified. It should be noted that importance of some transfer syntax properties must be determined early during specification because of their effect on message contents description specification possibilities. The properties are **compactness** and **extensibility.** If extreme compactness is required then extensibility must be restricted. If good extensibility is required then compromises must be done regarding compactness. The sections concerning these issues are marked in the following clauses as **COMPACTNESS** and **EXTENSIBILITY**.

Identifiers that could be keywords of another language (e.g.: SDL, C, ASN.1, JAVA, C++, …) should be avoided.

## ~~10~~B.1 Message level

### ~~10~~B.1.1 Messages

It is presumed that messages share the same structure, namely that they contain an identification part and a contents part. An identification part contains an IE that identifies a message among all messages in some context.

A contents part contains message specific IEs.

IE is a list of components.

Example: A protocol layer XYZ contains three messages: A, B and C. The structure of the messages is as presented in the figure ~~10~~B.1.1.1.

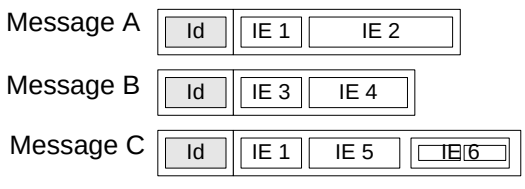



**Figure ~~10~~B.1.1.1: Three example messages**

Messages are specified using ASN.1 [1]. There are three ASN.1 types, *MessageA*, *MessageB* and *MessageC*, which contain definitions for the contents of the above messages. The mapping between the message contents types and message identifiers is as follows:

| Message id | Type of message contents |
|---|---|
| 1 | MessageA |
| 2 | MessageB |
| 3 | MessageC |

New message types will be introduced in the future.

In cases where different PDUs have different identification schemes it is possible to apply this categorisation for a set of PDUs that share the same identification scheme.

## ~~10~~B.1.2 Message definition

In order to capture information in the previous clause the following three things must be defined:

1. A structure for the table.
2. The table itself.
3. A generic message structure that can contain both message identifier IE and message contents IEs (i.e. id 1 + *MessageA*, id 2 + *MessageB*, id 3 + *MessageC*).

The table structure is defined as follows using ASN.1 classes [2]:

```
XYZ-MESSAGE ::= CLASS {
    &id     MessageId UNIQUE,
    &Type
}
WITH SYNTAX {
    &id &Type
}

MessageId ::= INTEGER (0..63)
```

The table is defined as follows:

```
XYZ-Messages XYZ-MESSAGE ::= {
    { messageA-id MessageA } |
    { messageB-id MessageB } |
    { messageC-id MessageC } ,
    ...                 -- Extension marker => additional messages
                        -- can be introduced.
}

messageA-id MessageId ::= 1
messageB-id MessageId ::= 2
messageC-id MessageId ::= 3
```

The following type represents the generic message structure that can carry values of the messages specified in the *XYZ-Messages* table.

```
XYZ-Message ::= SEQUENCE {
    id      XYZ-MESSAGE.&id   ({XYZ-Messages)},
            -- MessageId: 1, 2 or 3

    contents    XYZ-MESSAGE.&Type ({XYZ-Messages}{@id})
            -- id=1 => MessageA, id=2 => MessageB, id=3 => MessageC
}
```

The above definition means that if *id* is 1 then the *Message* type could be interpreted as the following type:

```
XYZ-Message ::= SEQUENCE {
    id      MessageId,  -- 1
    contents    SEQUENCE {
        ie1     IE1,
        ie2     IE2
    }
}
```

If *id* is 2 then the type could be interpreted as the following type:

```
XYZ-Message ::= SEQUENCE {
    id      MessageId,  -- 2
    contents    SEQUENCE {
        ie3     IE3,
        ie4     IE4
    }
}
```

## ~~10~~B.1.3 Messages and ASN.1 modules

ASN.1 definitions shall be placed in ASN.1 modules such that definitions in a module form a logical unit. For example PDUs definitions for one protocol layer could be in one ASN.1 module and IE definitions in another.

The tagging mode for the modules shall be "AUTOMATIC TAGS". Note that "AUTOMATIC TAGS" is not relevant for PER.

Example: A message definition module for the XYZ protocol layer.

```
XYZ-Messages DEFINITIONS AUTOMATIC TAGS ::=

BEGIN

XYZ-Messages XYZ-MESSAGE ::= {
    { messageA-id MessageA } |
    { messageB-id MessageB } |
    { messageC-id MessageC } ,
    ...             -- Additional messages can be introduced.
}

MessageA ::= SEQUENCE {
    -- Message contents
}

messageA-id MessageId ::= 1

MessageB ::= SEQUENCE {
    -- Message contents
}

messageB-id MessageId ::= 2

MessageC ::= SEQUENCE {
    -- Message contents
}

messageC-id MessageId ::= 3

END
```

## ~~10~~B.1.4 Messages and SDL

The identifiers *messageA-id*, *MessageA*, *messageB-id*, etc. can be used in descriptive SDL when protocol behaviour is specified. Note that classes and objects cannot be referenced in SDL96 but are allowed in SDL2000. Types and values

however can be imported to SDL definitions. The figures below contain some examples about usage of ASN.1 in SDL specifications.

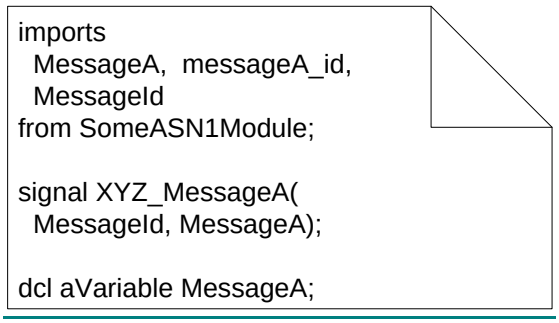



**Figure ~~10~~B.1.4.1: Import and use of ASN.1 definitions in SDL**

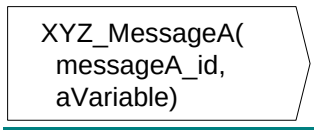



**Figure ~~10~~B.1.4.2: Sending of a message id and contents**

## ~~10~~B.2 Information element level

Messages consist of information elements.

The following ASN.1 message types are used in the following clauses.

```
MessageA ::= SEQUENCE {
    ie1 IE1,             -- A mandatory IE.

    ie2 IE2 OPTIONAL,        -- An optional IE.

-- Extensions from there
    ExtensionMarker SEQUENCE {} OPTIONAL

}

MessageB ::= SEQUENCE {
    ie3 IE3
        (CONSTRAINED BY {-- ComprehensionRequired(is for receiver) --}
        !comprehensionRequiredFailure)  ,

    ie4 IE4 DEFAULT 0,       -- An optional IE with a default value.

-- Extensions from there
    ExtensionMarker SEQUENCE {} OPTIONAL
}

MessageC ::= SEQUENCE {
    ie1 IE1
        (CONSTRAINED BY {-- PartialDecoding(OnErrorIgnoreRest) --}
        !partialDecodingFailure)
        OPTIONAL,

    ie5 IE5
        (CONSTRAINED BY {-- PartialDecoding(OnErrorIgnoreRest) --}
        !partialDecodingFailure)
        OPTIONAL,

-- Extensions from there
    ExtensionMarker SEQUENCE {
        ie6 IE6
            (CONSTRAINED BY {-- PartialDecoding(OnErrorIgnoreRest) --}
            !partialDecodingFailure)
            OPTIONAL        -- A new IE
  } OPTIONAL
```

```
}

-- Error codes
comprehensionRequiredFailure INTEGER ::= 1
partialDecodingFailure       INTEGER ::= 2
```

## ~~10~~B.2.1 Message contents

A message contents structure is defined using a sequence type (~~10~~B.3.10).

Example: *MessageA*, *MessageB* and *MessageC* are message contents structures.

## ~~10~~B.2.2 Optional IEs and default values

An IE can be marked as optional.

**COMPACTNESS:** Optional IEs shall be after mandatory ones. When ASN.1 is used with PER, this requirement is not relevant.

When the extension "SEQUENCE {} OPTIONAL" is used, the sender shall never indicate that the field is present.

Example: *MessageA.ie2* is an optional IE.

```
ie2 IE2 OPTIONAL
```

An IE can be marked as being optional and having a default value. In those cases a missing optional IE may be understood as having a certain value hence a defined meaning.

Example: *MessageB.ie4* is an optional IE with a default value.

```
ie4 IE4     DEFAULT 0
```

## ~~10~~B.2.3 New IEs

**EXTENSIBILITY:** If new IEs will be added to a message then the message contents structure must be specified as extensible using the ellipsis notation (...). New IEs shall be added after the extension marker. New IEs shall be optional or shall have default values.

Example: *MessageC.ie6* is an additional optional IE.

```
    ... ,
    ie6 IE6     OPTIONAL
```

## ~~10~~B.2.4 Comprehension required

"Comprehension required" requirement can be associated with an IE. It means that after an IE value has been decoded then the value is validated. Failure in validation causes rejection of the message.

The requirement is specified as an extension to ASN.1 by using user defined constraints [3]. The comment part of the constraint shall be of the form:

*ComprehensionRequired(<additional constraint>)*

where <additional constraint> specifies the rule that the IE must satisfy.

Example: The *MessageB* is a broadcast message. The *ie3* IE contains recipient addresses. It is not until the addresses have been decoded when a receiver can decide whether it should decode the rest of the message or not.

```
    ie3 IE3
```

```
        (CONSTRAINED BY {-- ComprehensionRequired(is for receiver) --}
        !comprehensionRequiredFailure)  ,
```

## ~~10~~B.2.5 Partial decoding

"Partial decoding" means that a PDU can be decoded in parts. One part forms a complete value that can be separated from other parts. A decoding error in a part does not invalidate previously decoded parts. Subsequent parts are however invalidated.

"Partial decoding" is specified as an extension to ASN.1 using user defined constraints. The comment of constraint shall be of the form:

*PartialDecoding(<OnErrorClause>)*

where <OnErrorClause> specifies action in case of a decoding error. The possible alternatives are:

- OnErrorIgnoreRest: End decoding, ignore rest of the message.

Example: The *MessageC* is a multipurpose message. The IEs *ie1*, *ie5* and *ie6* are independent of each other.

```
    ie1 IE1
        (CONSTRAINED BY {-- PartialDecoding(OnErrorIgnoreRest) --}
        !partialDecodingFailure)
```

## ~~10~~B.2.6 Error specification

An error specification can be associated with user-defined constraints.

A simple integer value can be associated with an exception specification or as elaborate structured value as needed.

Example: If decoding of *ie1* fails then decoder returns the error code *partialDecodingFailure*.

```
    ie1 IE1
        (CONSTRAINED BY {-- PartialDecoding(OnErrorIgnoreRest) --}
        !partialDecodingFailure)
```

# ~~10~~B.3 Component level

Information elements consist of components.

The following ASN.1 types shall be used at the component level:

- Boolean (~~10~~B.3.4)
- Integer (~~10~~B.3.5)
- Enumerated (~~10~~B.3.6)
- Bit string (~~10~~B.3.7)
- Octet string (~~10~~B.3.8)
- Null (~~10~~B.3.9)
- Sequence (~~10~~B.3.10)
- Sequence-of (~~10~~B.3.11)
- Choice (~~10~~B.3.12)
- Character string types (~~10~~B.3.13)

## ~~10~~B.3.1 Extensibility

**COMPACTNESS:** In the component level use of ASN.1 extensibility is forbidden unless otherwise stated in the following clauses.

## ~~10~~B.3.2 Comprehension required

"Comprehension required" can be applied to components of sequence types, alternatives of choice types and elements of sequence-of types. See ~~10~~B.2.4.

## ~~10~~B.3.3 Partial decoding

"Partial decoding" can be applied to components of sequence types, alternatives of choice types and elements of sequence-of types. See ~~10~~B.2.5.

## ~~10~~B.3.4 Boolean

Example: A simple boolean type.

```
Flag ::= BOOLEAN

setFlag Flag ::= TRUE
```

## ~~10~~B.3.5 Integer

An integer type should be constrained.

**COMPACTNESS:** An integer type shall be constrained to have a finite value set. The value set can be either continuous or non-continuous.

Named numbers can be associated with an integer type.

**COMPACTNESS, EXTENSIBILITY:** If an integer type needs to be extended in the future then two value sets must be defined:

- A value set that specifies the values that can be sent in the current protocol version.
- A value set that specifies all the possible values that can be received now and in the future.

The former value set is specified in a user-defined constraint. The comment part shall be of the form:

*Send(<value set>)*

The latter form is specified using a normal constraint, e.g. a value range constraint.

Examples: Integer types and values.

```
Counter        ::= INTEGER (0..255)     -- 0 <= Counter value <= 255

SparseValueSet ::= INTEGER (0|3|5|6|8|11)

SignedInteger  ::= INTEGER (-10..10)

-- idle stands for value 0.
Status         ::= INTEGER { idle(0), veryBusy(3) } (0..3)

-- Send values 0..3 but be prepared to receive values 0..15.
Extensible     ::= INTEGER (0..15)(CONSTRAINED BY {-- Send(0..3) --})

initialCounter Counter        ::= 0

zero           SparseValueSet ::= 0
```

```
initialStatus  Status         ::= idle
```

## ~~10~~B.3.6 Enumerated

The EnumerationItem shall not contain a named number (e.g.: foo (3)).

The list of enumerated values specifies the value set for an enumerated type.

**COMPACTNESS, EXTENSIBILITY:** If an enumerated type needs to be extended in the future then two value sets must be defined as in case of integer types.

NOTE: An integer type with named numbers can be used as on alternative to an enumerated type.

Example: Enumerated types and value.

```
Enum          ::= ENUMERATED { a, b, c, d }

-- Send values a, b, c or d but be prepared to receive values
-- a, b, c, d, spare4, spare5, spare6 and spare7.
ExtendedEnum ::= ENUMERATED { a, b, c, d, spare4, spare5, spare6, spare7 }
    (CONSTRAINED BY {-- Send(a|b|c|d) --})

aEnum Enum ::= a
```

## ~~10~~B.3.7 Bit string

A size constraint shall be specified. It shall be finite.

Named bits can be associated with a bit string type.

Example: Bit string types and values.

```
FixedLengthBitStr    ::= BIT STRING (SIZE (10))

VariableLengthBitStr ::= BIT STRING (SIZE (0..10))

BitFlags             ::= BIT STRING { a(0), b(1), c(2), d(3)} (SIZE (4))

fix FixedLengthBitStr    ::= '0001101100'B

var VariableLengthBitStr ::= '0'B

flg BitFlags             ::= { a, c, d } -- '1011'B
```

## ~~10~~B.3.8 Octet string

A size constraint shall be specified. It shall be finite.

Example: Octet string types and values.

```
FixedLengthOctetStr    ::= OCTET STRING (SIZE (10))

VariableLengthOctetStr ::= OCTET STRING (SIZE (0..10))

UpperLayerPDUSegment   ::= OCTET STRING (SIZE (1..512))

fix FixedLengthOctetStr    ::= '0102030405060708090A'H

var VariableLengthOctetStr ::= 'FF'H
```

## ~~10~~B.3.9 Null

A null type has only one value, NULL.

Example: Null type as an alternative type of a choice type.

```
IE ::= CHOICE {
    doThis    ThisArg,
    doThat    ThatArg,
    doNothing     NULL
}
```

## ~~10~~B.3.10Sequence

A sequence type is a record. Components of a sequence type can be optional or they can have default values. Optional components and components with default values should be after mandatory components.

Inner subtyping can be used to force an optional component to be present or absent in a derived type.

If an optional component is conditionally present or absent then the condition shall be specified in a user defined constraint of the form:

*Condition(<condition expression>)*

<condition expression> shall be such that both sender and receiver are able to evaluate it before a conditional component is encoded or decoded.

"Comprehension required" can be associated with a component of a sequence type.

"Partial decoding" can be associated with a component of a sequence type.

**EXTENSIBILITY:** A sequence type can be marked as extensible. Example: Sequence types and values.

```
Record ::= SEQUENCE {
    flag        Flag,
    counter Counter,
    bitFlags    BitFlags        OPTIONAL,
    extEnum ExtendedEnum    DEFAULT a
}

DerivedRecord ::= Record (WITH COMPONENTS {
                ...,
                bitFlags    PRESENT
                  })

RecordWithConditionalComponent ::= SEQUENCE {
    mand        INTEGER (0..7),
    opt     BOOLEAN OPTIONAL,
    cond        BOOLEAN OPTIONAL
} ( WITH COMPONENT {mand(7), cond PRESENT} | WITH COMPONENT {cond ABSENT} )

aRecord Record ::= {
    flag        TRUE,
    counter 100
}

anotherRecord DerivedRecord ::= {
    flag        TRUE,
    counter 100,
    bitFlags    '0101'B     -- bitFlags must be present
}
```

## ~~10~~B.3.11Sequence-of

A sequence-of type is a list of some element type. A size constraint shall be specified. It shall be finite.

"Comprehension required" can be associated with an element of a sequence-of type.

"Partial decoding" can be associated with an element of a sequence-of type.

Example: Sequence-of types and values.

```
FixedLengthList       ::= SEQUENCE (SIZE (10)) OF Record

VariableLengthList    ::= SEQUENCE (SIZE (0..10)) OF Status

UpperLayerPDUSegments ::= SEQUENCE (SIZE (1..10)) OF UpperLayerPDUSegment

aList VariableLengthList ::= { idle, 1, 2, veryBusy, 2, 1, idle }
```

## ~~10~~B.3.12Choice

A choice type is a variant record. Only one alternative component can be selected.

Inner subtyping can be used to force an alternative to be selected in a derived type.

"Comprehension required" can be associated with an alternative component of a choice type.

"Partial decoding" can be associated with an alternative component of a choice type.

**EXTENSIBILITY:** A choice type can be marked as extensible.

Example: Choice type and value.

```
VariantRecord ::= CHOICE {
    flag        Flag,
    counter Counter,
    extEnum ExtendedEnum
}

aVariantRecord VariantRecord ::= flag : FALSE
```

## ~~10~~B.3.13Restricted character string types

A size constraint shall be specified. It shall be finite.

It should specified the permitted alphabet for compactness reasons (see examples in PER [5]).

Example: Character string types.

```
FixedStr  ::= IA5String (SIZE (10))

VarStr    ::= IA5String (SIZE (1..10))

FixedWStr ::= BMPString (SIZE (10))

VarWStr   ::= BMPString (SIZE (1..10))
```

## ~~10~~B.3.14IEs and ASN.1 modules

If an IE or a component field within an IE is a parameter from another protocol layer then that type for such a field should be defined in another module. In this way there is a clear separation of definitions that are specific to different protocol layers.

Example: The XYZ protocol message *MessageC* contains an IE, which contains an OPQ protocol layer specific field *parameter1*. Type for the field is imported from that OPQ specific module.

```
XYZ-Messages DEFINITIONS AUTOMATIC TAGS ::=

BEGIN

IMPORTS
    OPQParameter    -- OPQParameter is not defined within XYZ-Messages
                    -- module.
FROM OPQ-DataTypes;

MessageC ::= SEQUENCE {
    -- Other IEs.
    ie6 IE6 OPTIONAL
}
-- Other definitions ...

IE6 ::= SEQUENCE {
    parameter1  OPQParameter,   -- Imported definitions can be
                                -- referred to.
    parameter2  XYZParameter
}

XYZParameter ::= INTEGER (0..255)

END
```

Example: The OPQ protocol layer specific module exports *OPQParameter* type so that other modules can refer it.

```
OPQ-DataTypes DEFINITIONS AUTOMATIC TAGS ::=

BEGIN


OPQParameter ::= INTEGER (0..7)

END
```

# Annex C (informative): Handling of DS-41

- Modelling of RRC services is provided by means of primitives.
- RRC CN dependent info:
- In broadcast message, neighbour cells are described the same way as for GSM neighbour cells (i.e.: in the same SystemInformationBlock but with a tag to indicate CN type or RTT).
- In dedicated messages.

  a transparent container as NAS info is used to carry ANSI-41;

  for PLMN Id and Identities used by the RRC, the CN Type info is used;

  NAS binding info is used;
- In Paging messages, a tag to indicate CN type is used.
- Extensions like handover message to Multicarrier is handled the same way as GSM.

**3GPP TSG-RAN WG2 Meeting #21**
**Busan, Korea, 21-25 May 2001**

***R2-011149***

*CR-Form-v3*

# CHANGE REQUEST

⌘ **25.921** **CR** **016** ⌘ rev **-** ⌘ Current version: **3.3.0** ⌘

*For **HELP** on using this form, see bottom of this page or look at the pop-up text over the ⌘ symbols.*

***Proposed change affects:*** ⌘ (U)SIM ☐ ME/UE **X** Radio Access Network **X** Core Network ☐

| | | |
|---|---|---|
| ***Title:*** | ⌘ | Usage of spare values in future releases |
| ***Source:*** | ⌘ | TSG-RAN WG2 |
| ***Work item code:*** | ⌘ | TEI |
| ***Date:*** | ⌘ | 2001-05-15 |
| ***Category:*** | ⌘ | **F** |
| ***Release:*** | ⌘ | R99 |

*Use one of the following categories:*
***F*** *(essential correction)*
***A*** *(corresponds to a correction 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.

*Use one of the following releases:*

| | |
|---|---|
| *2* | *(GSM Phase 2)* |
| *R96* | *(Release 1996)* |
| *R97* | *(Release 1997)* |
| *R98* | *(Release 1998)* |
| *R99* | *(Release 1999)* |
| *REL-4* | *(Release 4)* |
| *REL-5* | *(Release 5)* |

| | | |
|---|---|---|
| ***Reason for change:*** | ⌘ | A description is needed to clarify how the spare values are to be used in future releases. |
| ***Summary of change:*** | ⌘ | A new sub-clause is included in 10.4.3.4 providing guidelines for the usage of spare values to define extensions. |
| ***Consequences if not approved:*** | ⌘ | Possibility of wrong usage of spare values, leading to backward incompatibility. |

| | | |
|---|---|---|
| ***Clauses affected:*** | ⌘ | 10.4.3.4.4a (new) |
| ***Other specs affected:*** | ⌘ | ☐ Other core specifications ⌘<br>☐ Test specifications<br>☐ O&M Specifications |
| ***Other comments:*** | ⌘ | Backwards compatibility: This CR presents only some further examples of the usage of the extensions, and makes no actual changes in methodology, and is thus backwards compatible. |

**How to create CRs using this form:**
Comprehensive information and tips about how to create CRs can be found at: http://www.3gpp.org/3G_Specs/CRs.htm. Below is a brief summary:

1) Fill out the above form. The symbols above marked ⌘ contain pop-up help information about the field that they are closest to.
2) Obtain the latest version for the release of the specification to which the change is proposed. Use the MS Word "revision marks" feature (also known as "track changes") when making the changes. All 3GPP specifications can be downloaded from the 3GPP server under ftp://www.3gpp.org/specs/ For the latest version, look for the directory name with the latest date e.g. 2000-09 contains the specifications resulting from the September 2000 TSG meetings.
3) With "track changes" disabled, paste the entire CR form (use CTRL-A to select it) into the specification just in front of the clause containing the first piece of changed text. Delete those parts of the specification which are not relevant to the change request.

### 10.4.3.4.4a Replacement of a spare value with a new element

If a new value is to be included in an IE of type ENUMERATED, for which spare values were defined in the previous version, those spare values can be replaced with the new values.

If more new values are needed, than spare values included in the previous version, one spare value can be replaced by a special extension value (called e-new in example 8). If that value is used, a new element in the nonCriticalExtension part (element1-new) will define the new values, as shown in Example 8 below:

```
-- In the previous version, MessageA-r3-IEs was defined:
MessageA-r3-IEs ::=                 SEQUENCE {
    element1                            ENUMERATED { e1, e2, spare1, spare2 }
}

-- Now three new values are needed for element1: e3, e4 and e5. MessageA-r3-IEs is redefined:
MessageA-r3-IEs ::=                 SEQUENCE {
-- If the following has the value e-new, the actual value of element1 is defined in
-- element1-new included in MessageA-r4-ext-IEs
    element1                            ENUMERATED { e1, e2, e3, e-new }
}

MessageA-r4-ext-IEs ::=             SEQUENCE {
-- the following shall be present, if element1 in MessageA-r3-IEs has the value e-new.
    element1-new                        ENUMERATED { e4, e5, spare1, spare2 }   OPTIONAL
}
```

**Example 8**

If a spare value is included in a CHOICE, and that has to be replaced with a new information element and an apropriate type in the new version, the name of the element replaces the spare name in the CHOICE, but the type can not be replaced, because that would lead to incompatibilities. Instead, the new type is included in the nonCriticalExtension part of the message, as shown in Example 9 below:

```
-- In the previous version, MessageA-r3-IEs was defined:
MessageA-r3-IEs ::=                 SEQUENCE {
    element1                            CHOICE {
        e1                                  E1,
        e2                                  E2,
        spare1                              NULL,
        spare2                              NULL
    }
}

-- Now a new option is needed for the element1 CHOICE: e3 with type E3.
-- MessageA-r3-IEs is redefined:
MessageA-r3-IEs ::=                 SEQUENCE {
-- If element1 has the value e3, the value of e3 is specified in the element e3
-- included in MessageA-r4-ext-IEs.
    element1                            CHOICE {
        e1                                  E1,
        e2                                  E2,
        e3                                  NULL,
        spare2                              NULL
    }
}

MessageA-r4-ext-IEs ::=             SEQUENCE {
-- the following shall be present, if element1 in MessageA-r3-IEs has the value e3.
    e3                                  E3          OPTIONAL
}
```

**Example 9**

**3GPP TSG-RAN WG2 Meeting #21**
**Busan, Korea, 21-25 May 2001**

***R2-011329***

*CR-Form-v3*

# CHANGE REQUEST

⌘ **25.921** **CR** **017** ⌘ rev **-** ⌘ Current version: **4.0.0** ⌘

*For **HELP** on using this form, see bottom of this page or look at the pop-up text over the ⌘ symbols.*

***Proposed change affects:*** ⌘ (U)SIM ☐ ME/UE **X** Radio Access Network **X** Core Network ☐

| | | |
|---|---|---|
| ***Title:*** | ⌘ | Usage of spare values in future releases |
| ***Source:*** | ⌘ | TSG-RAN WG2 |
| ***Work item code:*** | ⌘ | TEI |
| ***Date:*** | ⌘ | 2001-05-25 |
| ***Category:*** | ⌘ | **A** |
| ***Release:*** | ⌘ | REL-4 |

*Use one of the following categories:*
***F*** *(essential correction)*
***A*** *(corresponds to a correction 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.

*Use one of the following releases:*

| | |
|---|---|
| *2* | *(GSM Phase 2)* |
| *R96* | *(Release 1996)* |
| *R97* | *(Release 1997)* |
| *R98* | *(Release 1998)* |
| *R99* | *(Release 1999)* |
| *REL-4* | *(Release 4)* |
| *REL-5* | *(Release 5)* |

| | | |
|---|---|---|
| ***Reason for change:*** | ⌘ | A description is needed to clarify how the spare values are to be used in future releases. |
| ***Summary of change:*** | ⌘ | A new sub-clause is included in 10.4.3.4 providing guidelines for the usage of spare values to define extensions. |
| ***Consequences if not approved:*** | ⌘ | Possibility of wrong usage of spare values, leading to backward incompatibility. |

| | | |
|---|---|---|
| ***Clauses affected:*** | ⌘ | 10.4.3.4.4a (new) |
| ***Other specs affected:*** | ⌘ | ☐ Other core specifications ⌘<br>☐ Test specifications<br>☐ O&M Specifications |
| ***Other comments:*** | ⌘ | Corresponds to CR016 to 25.921 (rel-99) |

**How to create CRs using this form:**
Comprehensive information and tips about how to create CRs can be found at: http://www.3gpp.org/3G_Specs/CRs.htm. Below is a brief summary:

1) Fill out the above form. The symbols above marked ⌘ contain pop-up help information about the field that they are closest to.
2) Obtain the latest version for the release of the specification to which the change is proposed. Use the MS Word "revision marks" feature (also known as "track changes") when making the changes. All 3GPP specifications can be downloaded from the 3GPP server under ftp://www.3gpp.org/specs/ For the latest version, look for the directory name with the latest date e.g. 2000-09 contains the specifications resulting from the September 2000 TSG meetings.
3) With "track changes" disabled, paste the entire CR form (use CTRL-A to select it) into the specification just in front of the clause containing the first piece of changed text. Delete those parts of the specification which are not relevant to the change request.

### 10.4.3.4.4a Replacement of a spare value with a new element

If a new value is to be included in an IE of type ENUMERATED, for which spare values were defined in the previous version, those spare values can be replaced with the new values.

If more new values are needed, than spare values included in the previous version, one spare value can be replaced by a special extension value (called e-new in example 8). If that value is used, a new element in the nonCriticalExtension part (element1-new) will define the new values, as shown in Example 8 below:

```
-- In the previous version, MessageA-r3-IEs was defined:
MessageA-r3-IEs ::=                 SEQUENCE {
    element1                            ENUMERATED { e1, e2, spare1, spare2 }
}

-- Now three new values are needed for element1: e3, e4 and e5. MessageA-r3-IEs is redefined:
MessageA-r3-IEs ::=                 SEQUENCE {
-- If the following has the value e-new, the actual value of element1 is defined in
-- element1-new included in MessageA-r4-ext-IEs
    element1                            ENUMERATED { e1, e2, e3, e-new }
}

MessageA-r4-ext-IEs ::=             SEQUENCE {
-- the following shall be present, if element1 in MessageA-r3-IEs has the value e-new.
    element1-new                        ENUMERATED { e4, e5, spare1, spare2 }   OPTIONAL
}
```

**Example 8**

If a spare value is included in a CHOICE, and that has to be replaced with a new information element and an apropriate type in the new version, the name of the element replaces the spare name in the CHOICE, but the type can not be replaced, because that would lead to incompatibilities. Instead, the new type is included in the nonCriticalExtension part of the message, as shown in Example 9 below:

```
-- In the previous version, MessageA-r3-IEs was defined:
MessageA-r3-IEs ::=                 SEQUENCE {
    element1                            CHOICE {
        e1                                  E1,
        e2                                  E2,
        spare1                              NULL,
        spare2                              NULL
    }
}

-- Now a new option is needed for the element1 CHOICE: e3 with type E3.
-- MessageA-r3-IEs is redefined:
MessageA-r3-IEs ::=                 SEQUENCE {
-- If element1 has the value e3, the value of e3 is specified in the element e3
-- included in MessageA-r4-ext-IEs.
    element1                            CHOICE {
        e1                                  E1,
        e2                                  E2,
        e3                                  NULL,
        spare2                              NULL
    }
}

MessageA-r4-ext-IEs ::=             SEQUENCE {
-- the following shall be present, if element1 in MessageA-r3-IEs has the value e3.
    e3                                  E3          OPTIONAL
}
```

**Example 9**

**3GPP TSG-RAN WG2 Meeting #21**
**Busan, Korea, 21-25 May 2001**

***R2-011457***

*CR-Form-v4*

# CHANGE REQUEST

⌘ **25.921** **CR** **018** ⌘ ev **r1** ⌘ Current version: **3.3.0** ⌘

*For **HELP** on using this form, see bottom of this page or look at the pop-up text over the ⌘ symbols.*

***Proposed change affects:*** ⌘ (U)SIM ☐ ME/UE **X** Radio Access Network **X** Core Network ☐

| | | |
|---|---|---|
| ***Title:*** | ⌘ | Structure and naming of extensions in ASN.1 |
| ***Source:*** | ⌘ | TSG-RAN WG2 |
| ***Work item code:*** | ⌘ | TEI |
| ***Date:*** | ⌘ | 2001-05-25 |
| ***Category:*** | ⌘ | **F** |
| ***Release:*** | ⌘ | R99 |

*Use one of the following categories:*
***F*** *(correction)*
***A*** *(corresponds to a correction 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.

*Use one of the following releases:*
*2 (GSM Phase 2)*
*R96 (Release 1996)*
*R97 (Release 1997)*
*R98 (Release 1998)*
*R99 (Release 1999)*
*REL-4 (Release 4)*
*REL-5 (Release 5)*

| | | |
|---|---|---|
| ***Reason for change:*** | ⌘ | The description of how to use the extension mechanism is not very clear. |
| ***Summary of change:*** | ⌘ | It is clarified how to structure the message top-level type in case of extensions.<br>It is clarified how to name the new elements in case of extensions to avoid ambiguities.<br>The examples in 10.4 are corrected to conform to these rules.<br>Changes in revision 1 of this CR: Disclaimer added, that further possible structures and naming conventions are FFS. |
| ***Consequences if not approved:*** | ⌘ | Different change requests could use the extension mechanism in different ways, causing a complicated structure in ASN.1, and in some cases introducing ambiguities. |

| | | |
|---|---|---|
| ***Clauses affected:*** | ⌘ | 10.4.2, 10.4.3.1, 10.4.3.2, 10.4.3.3, 10.4.3.4.1, 10.4.3.4.2, 10.4.3.4.3, 10.4.3.4.4 |
| ***Other specs affected:*** | ⌘ | ☐ Other core specifications<br>☐ Test specifications<br>☐ O&M Specifications |
| ***Other comments:*** | ⌘ | CR900 to 25.331 (R99, shadow CR901 to 25.331,REL-4) and CR902r1 to 25.331 (REL-4) apply the above rules to the current ASN.1. |

**How to create CRs using this form:**
Comprehensive information and tips about how to create CRs can be found at: http://www.3gpp.org/3G_Specs/CRs.htm. Below is a brief summary:

1) Fill out the above form. The symbols above marked ⌘ contain pop-up help information about the field that they are closest to.

2) Obtain the latest version for the release of the specification to which the change is proposed. Use the MS Word "revision marks" feature (also known as "track changes") when making the changes. All 3GPP specifications can be downloaded from the 3GPP server under ftp://ftp.3gpp.org/specs/ For the latest version, look for the directory name with the latest date e.g. 2001-03 contains the specifications resulting from the March 2001 TSG meetings.

3) With "track changes" disabled, paste the entire CR form (use CTRL-A to select it) into the specification just in front of the clause containing the first piece of changed text. Delete those parts of the specification which are not relevant to the change request.

## 10.4 Extensions for future releases in RRC

### 10.4.1 Basic principles

All non-critical extensions are shown even if empty as it costs no bits.

### 10.4.2 Naming convention

The abstract type defining a message provides mechanisms to allow for extending the message in future releases:

- For critical extensions, this is done by defining the message as a CHOICE of two alternatives, one being the intended message structure, and the other being an empty SEQUENCE named "criticalExtensions".
- For non-critical extensions, this is done by defining an OPTIONAL element named "nonCriticalExtensions" of type "SEQUENCE {}" at the end of the message definition.

When extensions are introduced, this is done by replacing one of the empty SEQUENCEs by a new structure, that includes a new type containing the message extensions, and the same extension mechanism recursively for further extensions.

The new elements introduced to specify the extensions should be grouped together in an element with a name showing the release, in which the extension was made, and the release of the message root, on which the extension was made (the second one applies only for non-critical extensions). For this naming, "r3" is used for Release-99, "r4" for Release 4, "r5" for Release 5 and so on. The suffix "ext" is used to indicate non-critical extensions.

If non-critical extensions for two different roots happen to be identical in contents, their types are still named differently, possibly with the second being declared as synonymous to the first.

~~The suffixes "-r3" for the present release, "-r4" for Release 4 and so on are used to differentiate further releases.~~

An example is given below to illustrate these principles, on the message named ~~"test-msg"~~"Test-msg"

```
-- In Release-99, the Test-msg is defined as following:
tTest-msg-r3 ::= CHOICE {
    r3                              SEQUENCE {
        test-msg-r3                     tTest-msg-r3-IEs,
        nonCriticalExtensions           SEQUENCE {} OPTIONAL
    },
    criticalExtensions              SEQUENCE {}
}

-- In Release 4, the Test-msg gets the following structure, if critical and non-critical
-- extensions are introduced.
Test-msg ::= CHOICE {
    r3                              SEQUENCE {
        test-msg-r3                     Test-msg-r3-IEs,
        nonCriticalExtensions           SEQUENCE {
            test-msg-r3-r4-ext              Test-msg-r3-r4-ext-IEs,
            nonCriticalExtensions           SEQUENCE {} OPTIONAL
        } OPTIONAL
    },
    criticalExtensions              CHOICE {
        r4                              SEQUENCE {
            test-msg-r4                     Test-msg-r4-IEs,
            nonCriticalExtensions           SEQUENCE {} OPTIONAL
        },
        criticalExtensions              SEQUENCE {}
    }
}

-- In Release 5, the Test-msg gets the following structure, if more critical and non-critical
-- extensions are introduced.
-- Here, non-critical extensions are introduced in both the r3 and r4 root of Test-msg.
Test-msg ::= CHOICE {
    r3                              SEQUENCE {
        test-msg-r3                     Test-msg-r3-IEs,
        nonCriticalExtensions           SEQUENCE {
            test-msg-r3-r4-ext              Test-msg-r3-r4-ext-IEs,
```

```
            nonCriticalExtensions           SEQUENCE {
                test-msg-r3-r5-ext              Test-msg-r3-r5-ext-IEs,
                nonCriticalExtensions           SEQUENCE {} OPTIONAL
            }   OPTIONAL
        } OPTIONAL
    },
    criticalExtensions              CHOICE {
        r4                              SEQUENCE {
            test-msg-r4                     Test-msg-r4-IEs,
            nonCriticalExtensions           SEQUENCE {
                test-msg-r4-r5-ext              Test-msg-r4-r5-ext-IEs,
                nonCriticalExtensions           SEQUENCE {} OPTIONAL
            }   OPTIONAL
        },
        criticalExtensions              CHOICE {
            r5                              SEQUENCE {
                test-msg-r5                     Test-msg-r5-IEs,
                nonCriticalExtensions           SEQUENCE {} OPTIONAL
            },
            criticalExtensions              SEQUENCE {}
        }
    }
}

test-msg-r4 ::= CHOICE {
    r3                              SEQUENCE {
        test-msg-r3                     test-msg-r3-IEs,
        test-msg-r3-r4ext               test-msg-r3-r4ext-IEs,
        nonCriticalExtensions           SEQUENCE {} OPTIONAL
    },
    r4                              SEQUENCE {
        test-msg-r4                     test-msg-r4-IEs,
        nonCriticalExtensions           SEQUENCE {} OPTIONAL
    },
    criticalExtensions              SEQUENCE {}
}

test-msg-r5 ::= CHOICE {
    r3                              SEQUENCE {
        test-msg-r3                     test-msg-r3-IEs,
        test-msg-r3-r4ext               test-msg-r3-r4ext-IEs,
        test-msg-r3-r5ext               test-msg-r3-r5ext-IEs,
        nonCriticalExtensions           SEQUENCE {} OPTIONAL
    },
    r4                              SEQUENCE {
        test-msg-r4                     test-msg-r4-IEs,
        test-msg-r4-r5ext               test-msg-r4-r5ext-IEs,
        nonCriticalExtensions           SEQUENCE {} OPTIONAL
    },
    r5                              SEQUENCE {
        test-msg-r5                     test-msg-r5-IEs,
        nonCriticalExtensions           SEQUENCE {} OPTIONAL
    },
    criticalExtensions              SEQUENCE {}
}
```

Critical extensions in release-*N* in message "`Test-msg`" should be included in the type "`Test-msg-r`*N*`-IEs`" (*N*=3 is used for release-99).

Non-critical extensions in release-*N* included in the release-*M* branch of the top-level CHOICE should be included in a type "`Test-msg-r`*M*`-r`*N*`-ext-IEs`".

If an abstract type is introduced in Release-*N* when new elements are included in an extension, it should have a suffix "`-r`*N*". For Release-99 types, no such suffix is used.

If an abstract type is introduced in Release-*N* to extend an already existing type "`TypeX`", it should get the same name with a suffix "`-r`*N*`-ext`", i.e. "`TypeX-r`*N*`-ext`".

Using the above naming rules, when changes are done in release-*N*, only changes in types with a suffix "`-r`*N*" or "`-r`*N*`-ext`" are allowed, in order to avoid conflicts with previous releases. An exception is the Message type itself, which can be changed by replacing the empty SEQUENCEs with extensions as shown above, and elements having spare values defined, where the spare value can be replaced with a newly introduced value.

An exception to the above structure can be needed, if there are some elements to be used in a message, which need to be comprehended even in case of critical extensions (e.g. for error handling procedures). In this case, the elements can be placed before one of the criticalExtensions CHOICEs, as shown in the example below:

```
Test-msg ::= CHOICE {
    r3                              SEQUENCE {
        test-msg-r3                     Test-msg-r3-IEs,
        nonCriticalExtensions           SEQUENCE {
            test-msg-r3-r4-ext              Test-msg-r3-r4-ext-IEs,
            nonCriticalExtensions           SEQUENCE {} OPTIONAL
        } OPTIONAL
    },
    criticalExtensions              SEQUENCE {
        importantElements               ImportantElements,
        rest-of-message                 CHOICE {
            r4                              SEQUENCE {
                test-msg-r4                     Test-msg-r4-IEs,
                nonCriticalExtensions           SEQUENCE {} OPTIONAL
            },
            criticalExtensions              SEQUENCE {}
        }
    }
}
```

In the above example, the elements in "importantElements" can be comprehended from a UE implementing this structure, even if a future version of the message including critical extensions is transmitted (i.e. the criticalExtension branch of the second CHOICE is used).

NOTE:- The structure presented in this clause and the proposed naming rules are one possibility. Further possibilities are FFS.

NOTE:- When non-critical extensions are introduced in a message that does not have yet a criticalExtension branch, they are introduced in the "Test-msg-r*M*-r*N*-ext-IEs" type as described above. It is possible, that after this change, another change introduces a critical extension for the same message, thus defining a critical extension branch. In this case, the whole message is redefined in the type "Test-msg-r*N*-IEs", and care is to be taken to include in this new type also all non-critical extensions that were introduced previously, in a way that best fits the new structure of the message.

For being prepared for such cases, it could be beneficial to define in advance the "Test-msg-r*N*-IEs" whenever a non-critical extension is introduced, which would be an unused type mirroring the actual structure of the message, as long as no critical extensions are introduced, and would be used as the basis of the message if a critical extension is introduced. It is FFS if this concept is feasible, and if it should be introduced in the future.

## 10.4.3 Recommendations for extensions for further releases in RRC

### 10.4.3.1 General

When in RRC an information element group is to be extended, the extension cannot be done directly in that IE, but only in the top level of the message, in the extension IEs of the message structure shown in Example 1. For implementing the extension, it has therefore to be investigated, in which messages the element to be extended is included.

Depending on criticality of the extension, this will be done by using the criticalExtension CHOICE branch, or the nonCriticalExtension information element.

The following subclauses provide some recommendations on how to use these elements.

```
MessageA-~~r3~~ ::=                 CHOICE {
    r3                                  SEQUENCE {
        messageA-r3                         MessageA-r3-IEs,
        nonCriticalExtensions               SEQUENCE {} OPTIONAL
    },
    criticalExtensions                  SEQUENCE {}
}
```

```
MessageA-r3-IEs ::=                 SEQUENCE {
    -- All messageA related information elements are included here.
}
```

**Example 1**

### 10.4.3.2 Critical Extensions

When the extension is a critical one (i.e. the receiver has to reject the whole message, and handle according to the error procedures of the protocol), the criticalExtension branch of the top-level CHOICE in the message is used. In this case the message information elements can be updated similar to the tabular, providing a message structure for the new release's information elements, similar to the updated structure in the tabular description.

Example 2 shows the structure of MessageA presented above, how it would become after a critical extension in Release 4.

In this example, in the criticalExtensions branch a new information element is defined (MessageA-r4-IEs) which will contain all messageA specific elements for Release 4, including the extensions in the place they fit naturally according to the semantics.

Note that in the new structure additional nonCriticalExtensions and criticalExtensions information elements are defined to allow for further extensions in future releases.

```
MessageA-r4 ::= CHOICE {
    r3                              SEQUENCE {
        messageA-r3                     MessageA-r3-IEs,
        nonCriticalExtensions           SEQUENCE {} OPTIONAL
    },
    criticalExtensions              SEQUENCECHOICE {
        messageA                        CHOICE {
        r4                                  SEQUENCE {
            messageA-r4                         MessageA-r4-IEs,
            nonCriticalExtensions               SEQUENCE {} OPTIONAL
        },
        criticalExtensions          SEQUENCE {}
    }
}

MessageA-r3-IEs ::=                 SEQUENCE {
    -- This is not changed compared to the above example. It includes all information
    -- elements used in Release-99 for messageA.
}

MessageA-r4-IEs ::=                 SEQUENCE {
    -- Here, the updated information elements used for MessageA in Release 4 are included.
}
```

**Example 2**

### 10.4.3.3 Non-critical Extensions

For non-critical extensions (i.e. the receiver shall just ignore the extensions, and use the rest of the message as if the extensions were not present), the approach is to use the nonCriticalExtensions information element, which is encoded at the end of the message, allowing backward compatibility.

The structure of the message of the example above is shown in Example 3 for the Release 4 message

Examples for special non-critical extensions and MessageA-r3-r4-ext-IEs are given in the following subclauses.

```
MessageA-r4 ::=                 CHOICE {
    r3                              SEQUENCE {
        messageA-r3                     MessageA-r3-IEs,
        nonCriticalExtensions           SEQUENCE {
            messageA-r3-r4-ext                  MessageA-r3-r4-ext-IEs,
            nonCriticalExtensions           SEQUENCE {} OPTIONAL
        } OPTIONAL
```

```
    },
    criticalExtensions              SEQUENCE {}
}

MessageA-r3-IEs ::=                 SEQUENCE {
    -- This is not changed compared to the same IE in Release-99. It includes all information
    -- elements used in Release-99 for messageA.
}

MessageA-r3-r4-ext-IEs ::=              SEQUENCE {
    -- Here are additional information elements needed to describe the extensions compared to
    -- the information included in MessageA-r3-IEs.
}
```

**Example 3**

### 10.4.3.4 Examples of non-critical extensions

#### 10.4.3.4.1 Addition of a separate IE

If the extension is the addition of an information element (not inside a CHOICE, SEQUENCE OF, SET OF etc.), this new element can be directly included in MessageA-r3-r4-ext-IEs.

Example4 shows how the MessageA is extended to include a new element, "element3".

```
MessageA-r3-IEs ::=                 SEQUENCE {
    element1                            Element1,
    element2                            Element2
}

MessageA-r3-r4-ext-IEs ::=              SEQUENCE {
    element3                            Element3-r4
}
```

**Example 4**

#### 10.4.3.4.2 Addition of an IE to a structured group

If the extension is the addition of an information element inside a CHOICE, SEQUENCE OF, etc. (meaning that the information element can be absent or present more than once, depending on some condition), the structure of the original message should be duplicated in MessageA-r3-r4-ext-IEs using only the elements relevant to the extension (usually the CHOICEs, SEQUENCE OFs, etc.), and a comment should be included to indicate that the two structures should be used consistently (e.g. when a CHOICE is duplicated, the same branch should be followed in both places, when a SEQUENCE OF is duplicated, the number of occurrences should be the same etc.).

This is illustrated in Example5, where a new element, "element1a-3", has to be included inside the "choice1b" branch of the "choice1" CHOICE. Here "choice1" is included again in MessageA-r3-r4-ext-IEs, and "element1a-3" is included there in the appropriate branch.

```
MessageA-r3-IEs ::=                 SEQUENCE {
-- For the "choice1b" branch of "choice1", an additional information element is
-- defined in MessageA-r3-r4-ext-IEs ("element1a-3").
    choice1                             CHOICE {
        choice1a                            SEQUENCE {
            element1a-1                         Element1a-1
        },
        choice1b                            SEQUENCE {
            element1a-2                         Element1a-2
        }
    }
}

MessageA-r3-r4-ext-IEs ::=              SEQUENCE {
-- In the following CHOICE the same branch shall be used as in choice1 in MessageA-r3-IEs.
    choice1                             CHOICE {
        choice1a                            NULL,
        choice1b                            SEQUENCE {
            element1a-3                         Element1a-3-r4
```

```
        }
    }
}
```

**Example 5**

### 10.4.3.4.3 Addition of a new CHOICE group

If the extension consists of moving some existing information elements inside a newly created CHOICE, the new branches of the created CHOICE should be included in MessageA-r3-r4-ext-IEs, and the CHOICE marked OPTIONAL, where absence means that the old elements are used. If the CHOICE is present, the old elements should be set to some default values, in order for older equipment to be understood, and new equipment should ignore the information therein.

This is illustrated in Example 6, where "element1" is to be moved inside the branch "choice1a" of a new CHOICE ("choice1").

```
MessageA-r3-IEs ::=                 SEQUENCE {
-- The contents of "element1" shall be ignored, if in "MessageA-r3-r4-ext-IEs" the branch
-- "choice1b" of the CHOICE "choice1" is used.
    element1                            Element1,
    element2                            Element2
}

MessageA-r3-r4-ext-IEs ::=              SEQUENCE {
    choice1                             CHOICE {
        choice1a                            SEQUENCE {},
        choice1b                            SEQUENCE {
            element3                            Element3-r4
        }
    }
}
```

**Example 6**

### 10.4.3.4.4 Extension of value range

If the value range of an element is to be extended, an element including the new values should be defined in MessageA-r3-r4-ext-IEs. If one of the new values is to be used, the already existing element from Release '99 should be set to some defined value (or be absent if it was OPTIONAL), in order for older equipment to work properly, and the new value should be signalled in the new information element.

In Example 7, "element1" is extended to have a range (0..15).

```
MessageA-r3-IEs ::=                 SEQUENCE {
-- "element1" shall be ignored if "element1" in MessageA-r3-r4-ext-IEs is present, and the
-- value of that element used instead.
    element1                            INTEGER (0..7)
    element2                            Element2
}

MessageA-r3-r4-ext-IEs ::=              SEQUENCE {
    element1                            INTEGER (0..15)         OPTIONAL
}
```

**Example 7**

**3GPP TSG-RAN WG2 Meeting #21**
**Busan, Korea, 21-25 May 2001**

***R2-011458***

*CR-Form-v4*

# CHANGE REQUEST

⌘ **25.921** **CR** **019** ⌘ ev **-** ⌘ Current version: **4.0.0** ⌘

*For **HELP** on using this form, see bottom of this page or look at the pop-up text over the ⌘ symbols.*

***Proposed change affects:*** ⌘ (U)SIM ☐ ME/UE **X** Radio Access Network **X** Core Network ☐

| | | |
|---|---|---|
| ***Title:*** | ⌘ | Structure and naming of extensions in ASN.1 |
| ***Source:*** | ⌘ | TSG-RAN WG2 |
| ***Work item code:*** | ⌘ | TEI |
| ***Date:*** | ⌘ | 2001-05-25 |
| ***Category:*** | ⌘ | **A** |
| ***Release:*** | ⌘ | REL-4 |

*Use one of the following categories:*
***F*** *(correction)*
***A*** *(corresponds to a correction 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.

*Use one of the following releases:*
*2 (GSM Phase 2)*
*R96 (Release 1996)*
*R97 (Release 1997)*
*R98 (Release 1998)*
*R99 (Release 1999)*
*REL-4 (Release 4)*
*REL-5 (Release 5)*

| | | |
|---|---|---|
| ***Reason for change:*** | ⌘ | The description of how to use the extension mechanism is not very clear. |
| ***Summary of change:*** | ⌘ | It is clarified how to structure the message top-level type in case of extensions.<br>It is clarified how to name the new elements in case of extensions to avoid ambiguities.<br>The examples in 10.4 are corrected to conform to these rules. |
| ***Consequences if not approved:*** | ⌘ | Different change requests could use the extension mechanism in different ways, causing a complicated structure in ASN.1, and in some cases introducing ambiguities. |

| | | |
|---|---|---|
| ***Clauses affected:*** | ⌘ | 10.4.2, 10.4.3.1, 10.4.3.2, 10.4.3.3, 10.4.3.4.1, 10.4.3.4.2, 10.4.3.4.3, 10.4.3.4.4 |
| ***Other specs affected:*** | ⌘ | ☐ Other core specifications<br>☐ Test specifications<br>☐ O&M Specifications |
| | ⌘ | |
| ***Other comments:*** | ⌘ | Corresponds to CR018r1 to 25.921 (rel-99) |

**How to create CRs using this form:**
Comprehensive information and tips about how to create CRs can be found at: http://www.3gpp.org/3G_Specs/CRs.htm. Below is a brief summary:

1) Fill out the above form. The symbols above marked ⌘ contain pop-up help information about the field that they are closest to.
2) Obtain the latest version for the release of the specification to which the change is proposed. Use the MS Word "revision marks" feature (also known as "track changes") when making the changes. All 3GPP specifications can be downloaded from the 3GPP server under ftp://ftp.3gpp.org/specs/ For the latest version, look for the directory name with the latest date e.g. 2001-03 contains the specifications resulting from the March 2001 TSG meetings.
3) With "track changes" disabled, paste the entire CR form (use CTRL-A to select it) into the specification just in front of the clause containing the first piece of changed text. Delete those parts of the specification which are not relevant to the change request.

# 10.4 Extensions for future releases in RRC

## 10.4.1 Basic principles

All non-critical extensions are shown even if empty as it costs no bits.

## 10.4.2 Naming convention

The abstract type defining a message provides mechanisms to allow for extending the message in future releases:

- For critical extensions, this is done by defining the message as a CHOICE of two alternatives, one being the intended message structure, and the other being an empty SEQUENCE named "criticalExtensions".
- For non-critical extensions, this is done by defining an OPTIONAL element named "nonCriticalExtensions" of type "SEQUENCE {}" at the end of the message definition.

When extensions are introduced, this is done by replacing one of the empty SEQUENCEs by a new structure, that includes a new type containing the message extensions, and the same extension mechanism recursively for further extensions.

The new elements introduced to specify the extensions should be grouped together in an element with a name showing the release, in which the extension was made, and the release of the message root, on which the extension was made (the second one applies only for non-critical extensions). For this naming, "r3" is used for Release-99, "r4" for Release 4, "r5" for Release 5 and so on. The suffix "ext" is used to indicate non-critical extensions.

If non-critical extensions for two different roots happen to be identical in contents, their types are still named differently, possibly with the second being declared as synonymous to the first.

~~The suffixes "-r3" for the present release, "-r4" for Release 4 and so on are used to differentiate further releases.~~

An example is given below to illustrate these principles, on the message named ~~"test-msg"~~"Test-msg"

```
-- In Release-99, the Test-msg is defined as following:
tTest-msg-r3 ::= CHOICE {
    r3                              SEQUENCE {
        test-msg-r3                     tTest-msg-r3-IEs,
        nonCriticalExtensions           SEQUENCE {} OPTIONAL
    },
    criticalExtensions              SEQUENCE {}
}

-- In Release 4, the Test-msg gets the following structure, if critical and non-critical
-- extensions are introduced.
Test-msg ::= CHOICE {
    r3                              SEQUENCE {
        test-msg-r3                     Test-msg-r3-IEs,
        nonCriticalExtensions           SEQUENCE {
            test-msg-r3-r4-ext              Test-msg-r3-r4-ext-IEs,
            nonCriticalExtensions           SEQUENCE {} OPTIONAL
        } OPTIONAL
    },
    criticalExtensions              CHOICE {
        r4                              SEQUENCE {
            test-msg-r4                     Test-msg-r4-IEs,
            nonCriticalExtensions           SEQUENCE {} OPTIONAL
        },
        criticalExtensions              SEQUENCE {}
    }
}

-- In Release 5, the Test-msg gets the following structure, if more critical and non-critical
-- extensions are introduced.
-- Here, non-critical extensions are introduced in both the r3 and r4 root of Test-msg.
Test-msg ::= CHOICE {
    r3                              SEQUENCE {
        test-msg-r3                     Test-msg-r3-IEs,
        nonCriticalExtensions           SEQUENCE {
            test-msg-r3-r4-ext              Test-msg-r3-r4-ext-IEs,
```

```
            nonCriticalExtensions           SEQUENCE {
                test-msg-r3-r5-ext              Test-msg-r3-r5-ext-IEs,
                nonCriticalExtensions           SEQUENCE {} OPTIONAL
            }   OPTIONAL
        } OPTIONAL
    },
    criticalExtensions              CHOICE {
        r4                              SEQUENCE {
            test-msg-r4                     Test-msg-r4-IEs,
            nonCriticalExtensions           SEQUENCE {
                test-msg-r4-r5-ext              Test-msg-r4-r5-ext-IEs,
                nonCriticalExtensions           SEQUENCE {} OPTIONAL
            }   OPTIONAL
        },
        criticalExtensions              CHOICE {
            r5                              SEQUENCE {
                test-msg-r5                     Test-msg-r5-IEs,
                nonCriticalExtensions           SEQUENCE {} OPTIONAL
            },
            criticalExtensions              SEQUENCE {}
        }
    }
}

test-msg-r4 ::= CHOICE {
    r3                              SEQUENCE {
        test-msg-r3                     test-msg-r3-IEs,
        test-msg-r3-r4ext               test-msg-r3-r4ext-IEs,
        nonCriticalExtensions           SEQUENCE {} OPTIONAL
    },
    r4                              SEQUENCE {
        test-msg-r4                     test-msg-r4-IEs,
        nonCriticalExtensions           SEQUENCE {} OPTIONAL
    },
    criticalExtensions              SEQUENCE {}
}

test-msg-r5 ::= CHOICE {
    r3                              SEQUENCE {
        test-msg-r3                     test-msg-r3-IEs,
        test-msg-r3-r4ext               test-msg-r3-r4ext-IEs,
        test-msg-r3-r5ext               test-msg-r3-r5ext-IEs,
        nonCriticalExtensions           SEQUENCE {} OPTIONAL
    },
    r4                              SEQUENCE {
        test-msg-r4                     test-msg-r4-IEs,
        test-msg-r4-r5ext               test-msg-r4-r5ext-IEs,
        nonCriticalExtensions           SEQUENCE {} OPTIONAL
    },
    r5                              SEQUENCE {
        test-msg-r5                     test-msg-r5-IEs,
        nonCriticalExtensions           SEQUENCE {} OPTIONAL
    },
    criticalExtensions              SEQUENCE {}
}
```

Critical extensions in release-*N* in message "`Test-msg`" should be included in the type "`Test-msg-r`*N*`-IEs`" (*N*=3 is used for release-99).

Non-critical extensions in release-*N* included in the release-*M* branch of the top-level CHOICE should be included in a type "`Test-msg-r`*M*`-r`*N*`-ext-IEs`".

If an abstract type is introduced in Release-*N* when new elements are included in an extension, it should have a suffix "`-r`*N*". For Release-99 types, no such suffix is used.

If an abstract type is introduced in Release-*N* to extend an already existing type "`TypeX`", it should get the same name with a suffix "`-r`*N*`-ext`", i.e. "`TypeX-r`*N*`-ext`".

Using the above naming rules, when changes are done in release-*N*, only changes in types with a suffix "`-r`*N*" or "`-r`*N*`-ext`" are allowed, in order to avoid conflicts with previous releases. An exception is the Message type itself, which can be changed by replacing the empty SEQUENCEs with extensions as shown above, and elements having spare values defined, where the spare value can be replaced with a newly introduced value.

An exception to the above structure can be needed, if there are some elements to be used in a message, which need to be comprehended even in case of critical extensions (e.g. for error handling procedures). In this case, the elements can be placed before one of the criticalExtensions CHOICEs, as shown in the example below:

```
Test-msg ::= CHOICE {
    r3                              SEQUENCE {
        test-msg-r3                     Test-msg-r3-IEs,
        nonCriticalExtensions           SEQUENCE {
            test-msg-r3-r4-ext              Test-msg-r3-r4-ext-IEs,
            nonCriticalExtensions           SEQUENCE {} OPTIONAL
        } OPTIONAL
    },
    criticalExtensions              SEQUENCE {
        importantElements               ImportantElements,
        rest-of-message                 CHOICE {
            r4                              SEQUENCE {
                test-msg-r4                     Test-msg-r4-IEs,
                nonCriticalExtensions           SEQUENCE {} OPTIONAL
            },
            criticalExtensions              SEQUENCE {}
        }
    }
}
```

In the above example, the elements in "importantElements" can be comprehended from a UE implementing this structure, even if a future version of the message including critical extensions is transmitted (i.e. the criticalExtension branch of the second CHOICE is used).

NOTE:- The structure presented in this clause and the proposed naming rules are one possibility. Further possibilities are FFS.

NOTE:- When non-critical extensions are introduced in a message that does not have yet a criticalExtension branch, they are introduced in the "Test-msg-r*M*-r*N*-ext-IEs" type as described above. It is possible, that after this change, another change introduces a critical extension for the same message, thus defining a critical extension branch. In this case, the whole message is redefined in the type "Test-msg-r*N*-IEs", and care is to be taken to include in this new type also all non-critical extensions that were introduced previously, in a way that best fits the new structure of the message.

For being prepared for such cases, it could be beneficial to define in advance the "Test-msg-r*N*-IEs" whenever a non-critical extension is introduced, which would be an unused type mirroring the actual structure of the message, as long as no critical extensions are introduced, and would be used as the basis of the message if a critical extension is introduced. It is FFS if this concept is feasible, and if it should be introduced in the future.

## 10.4.3 Recommendations for extensions for further releases in RRC

### 10.4.3.1 General

When in RRC an information element group is to be extended, the extension cannot be done directly in that IE, but only in the top level of the message, in the extension IEs of the message structure shown in Example 1. For implementing the extension, it has therefore to be investigated, in which messages the element to be extended is included.

Depending on criticality of the extension, this will be done by using the criticalExtension CHOICE branch, or the nonCriticalExtension information element.

The following subclauses provide some recommendations on how to use these elements.

```
MessageA-~~r3~~ ::=                 CHOICE {
    r3                                  SEQUENCE {
        messageA-r3                         MessageA-r3-IEs,
        nonCriticalExtensions               SEQUENCE {} OPTIONAL
    },
    criticalExtensions              SEQUENCE {}
}
```

```
MessageA-r3-IEs ::=                 SEQUENCE {
    -- All messageA related information elements are included here.
}
```

**Example 1**

### 10.4.3.2 Critical Extensions

When the extension is a critical one (i.e. the receiver has to reject the whole message, and handle according to the error procedures of the protocol), the criticalExtension branch of the top-level CHOICE in the message is used. In this case the message information elements can be updated similar to the tabular, providing a message structure for the new release's information elements, similar to the updated structure in the tabular description.

Example 2 shows the structure of MessageA presented above, how it would become after a critical extension in Release 4.

In this example, in the criticalExtensions branch a new information element is defined (MessageA-r4-IEs) which will contain all messageA specific elements for Release 4, including the extensions in the place they fit naturally according to the semantics.

Note that in the new structure additional nonCriticalExtensions and criticalExtensions information elements are defined to allow for further extensions in future releases.

```
MessageA~~-r4~~ ::= CHOICE {
    r3                              SEQUENCE {
        messageA-r3                     MessageA-r3-IEs,
        nonCriticalExtensions           SEQUENCE {} OPTIONAL
    },
    criticalExtensions              ~~SEQUENCE~~CHOICE {
~~        messageA                        CHOICE {~~
~~            r4                              SEQUENCE {~~
~~                messageA-r4                     MessageA-r4-IEs,~~
~~                nonCriticalExtensions           SEQUENCE {} OPTIONAL~~
~~            },~~
~~            criticalExtensions          SEQUENCE {}~~
~~        }~~
    }
}

MessageA-r3-IEs ::=                 SEQUENCE {
    -- This is not changed compared to the above example. It includes all information
    -- elements used in Release-99 for messageA.
}

MessageA-r4-IEs ::=                 SEQUENCE {
    -- Here, the updated information elements used for MessageA in Release 4 are included.
}
```

**Example 2**

### 10.4.3.3 Non-critical Extensions

For non-critical extensions (i.e. the receiver shall just ignore the extensions, and use the rest of the message as if the extensions were not present), the approach is to use the nonCriticalExtensions information element, which is encoded at the end of the message, allowing backward compatibility.

The structure of the message of the example above is shown in Example 3 for the Release 4 message

Examples for special non-critical extensions and MessageA-r3-r4-ext-IEs are given in the following subclauses.

```
MessageA-r4 ::=                 CHOICE {
    r3                              SEQUENCE {
        messageA-r3                     MessageA-r3-IEs,
        nonCriticalExtensions           SEQUENCE {
            messageA-r3-r4-ext                  MessageA-r3-r4-ext-IEs,
            nonCriticalExtensions           SEQUENCE {} OPTIONAL
        } OPTIONAL
```

```
    },
    criticalExtensions              SEQUENCE {}
}

MessageA-r3-IEs ::=                 SEQUENCE {
    -- This is not changed compared to the same IE in Release-99. It includes all information
    -- elements used in Release-99 for messageA.
}

MessageA-r3-r4-ext-IEs ::=              SEQUENCE {
    -- Here are additional information elements needed to describe the extensions compared to
    -- the information included in MessageA-r3-IEs.
}
```

**Example 3**

### 10.4.3.4 Examples of non-critical extensions

#### 10.4.3.4.1 Addition of a separate IE

If the extension is the addition of an information element (not inside a CHOICE, SEQUENCE OF, SET OF etc.), this new element can be directly included in MessageA-r3-r4-ext-IEs.

Example4 shows how the MessageA is extended to include a new element, "element3".

```
MessageA-r3-IEs ::=                 SEQUENCE {
    element1                            Element1,
    element2                            Element2
}

MessageA-r3-r4-ext-IEs ::=              SEQUENCE {
    element3                            Element3-r4
}
```

**Example 4**

#### 10.4.3.4.2 Addition of an IE to a structured group

If the extension is the addition of an information element inside a CHOICE, SEQUENCE OF, etc. (meaning that the information element can be absent or present more than once, depending on some condition), the structure of the original message should be duplicated in MessageA-r3-r4-ext-IEs using only the elements relevant to the extension (usually the CHOICEs, SEQUENCE OFs, etc.), and a comment should be included to indicate that the two structures should be used consistently (e.g. when a CHOICE is duplicated, the same branch should be followed in both places, when a SEQUENCE OF is duplicated, the number of occurrences should be the same etc.).

This is illustrated in Example5, where a new element, "element1a-3", has to be included inside the "choice1b" branch of the "choice1" CHOICE. Here "choice1" is included again in MessageA-r3-r4-ext-IEs, and "element1a-3" is included there in the appropriate branch.

```
MessageA-r3-IEs ::=                 SEQUENCE {
-- For the "choice1b" branch of "choice1", an additional information element is
-- defined in MessageA-r3-r4-ext-IEs ("element1a-3").
    choice1                             CHOICE {
        choice1a                            SEQUENCE {
            element1a-1                         Element1a-1
        },
        choice1b                            SEQUENCE {
            element1a-2                         Element1a-2
        }
    }
}

MessageA-r3-r4-ext-IEs ::=              SEQUENCE {
-- In the following CHOICE the same branch shall be used as in choice1 in MessageA-r3-IEs.
    choice1                             CHOICE {
        choice1a                            NULL,
        choice1b                            SEQUENCE {
            element1a-3                         Element1a-3-r4
```

```
        }
    }
}
```

**Example 5**

### 10.4.3.4.3 Addition of a new CHOICE group

If the extension consists of moving some existing information elements inside a newly created CHOICE, the new branches of the created CHOICE should be included in MessageA-r3-r4-ext-IEs, and the CHOICE marked OPTIONAL, where absence means that the old elements are used. If the CHOICE is present, the old elements should be set to some default values, in order for older equipment to be understood, and new equipment should ignore the information therein.

This is illustrated in Example 6, where "element1" is to be moved inside the branch "choice1a" of a new CHOICE ("choice1").

```
MessageA-r3-IEs ::=                 SEQUENCE {
-- The contents of "element1" shall be ignored, if in "MessageA-r3-r4-ext-IEs" the branch
-- "choice1b" of the CHOICE "choice1" is used.
    element1                            Element1,
    element2                            Element2
}

MessageA-r3-r4-ext-IEs ::=              SEQUENCE {
    choice1                             CHOICE {
        choice1a                            SEQUENCE {},
        choice1b                            SEQUENCE {
            element3                            Element3-r4
        }
    }
}
```

**Example 6**

### 10.4.3.4.4 Extension of value range

If the value range of an element is to be extended, an element including the new values should be defined in MessageA-r3-r4-ext-IEs. If one of the new values is to be used, the already existing element from Release '99 should be set to some defined value (or be absent if it was OPTIONAL), in order for older equipment to work properly, and the new value should be signalled in the new information element.

In Example 7, "element1" is extended to have a range (0..15).

```
MessageA-r3-IEs ::=                 SEQUENCE {
-- "element1" shall be ignored if "element1" in MessageA-r3-r4-ext-IEs is present, and the
-- value of that element used instead.
    element1                            INTEGER (0..7)
    element2                            Element2
}

MessageA-r3-r4-ext-IEs ::=              SEQUENCE {
    element1                            INTEGER (0..15)         OPTIONAL
}
```

**Example 7**

**3GPP TSG-RAN WG2 Meeting #21**
**Busan, Korea, May 21st – 25th, 2001**

***R2-011485***

*CR-Form-v3*

# CHANGE REQUEST

⌘ **25.921** **CR** **020** ⌘ rev **-** ⌘ Current version: **3.3.0** ⌘

*For **HELP** on using this form, see bottom of this page or look at the pop-up text over the ⌘ symbols.*

***Proposed change affects:*** ⌘ (U)SIM ☐ ME/UE **X** Radio Access Network **X** Core Network ☐

| | | |
|---|---|---|
| ***Title:*** | ⌘ | Addition of Recommendations for Extensions in RANAP, RNSAP, NBAP, and SABP |
| ***Source:*** | ⌘ | TSG-RAN WG3 |
| ***Work item code:*** | ⌘ | TEI |
| ***Date:*** | ⌘ | 25 May, 2001 |
| ***Category:*** | ⌘ | **F** |
| ***Release:*** | ⌘ | R99 |

*Use one of the following categories:*
***F*** *(essential correction)*
***A*** *(corresponds to a correction 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.

*Use one of the following releases:*
*2* *(GSM Phase 2)*
*R96* *(Release 1996)*
*R97* *(Release 1997)*
*R98* *(Release 1998)*
*R99* *(Release 1999)*
*REL-4* *(Release 4)*
*REL-5* *(Release 5)*

| | | |
|---|---|---|
| ***Reason for change:*** | ⌘ | Xxxxxxxxx |
| ***Summary of change:*** | ⌘ | Recommendations for extensions in future versions of specifications (future releases) has been added with respect to:<br>1. New procedures<br>2. New IEs<br>3. Changing the Presence of an existing IE.<br>4. Changing the Assigned Criticality of an existing IE.<br>5. Removing an existing IE.<br><br>Further more, the title of chapter 10.5 is aligned with the title of chapter 10.4 (Extensions for future releases in RRC). |
| ***Consequences if not approved:*** | ⌘ | The guidelines in this CR will not be maintained once more and more companies change to new delegates. This gives a risk that further updates of the specifications will be made in an undesirable way. |

| | | | | |
|---|---|---|---|---|
| ***Clauses affected:*** | ⌘ | 10.5, 10.5.3 (new) | | |
| ***Other specs affected:*** | ⌘ | **X** Other core specifications<br>☐ Test specifications<br>☐ O&M Specifications | ⌘ | TR 25.921 CR021 (Rel. 4) |
| ***Other comments:*** | ⌘ | | | |

**How to create CRs using this form:**
Comprehensive information and tips about how to create CRs can be found at: http://www.3gpp.org/3G_Specs/CRs.htm. Below is a brief summary:

1) Fill out the above form. The symbols above marked ⌘ contain pop-up help information about the field that they are closest to.

2) Obtain the latest version for the release of the specification to which the change is proposed. Use the MS Word "revision marks" feature (also known as "track changes") when making the changes. All 3GPP specifications can be downloaded from the 3GPP server under ftp://www.3gpp.org/specs/ For the latest version, look for the directory name with the latest date e.g. 2000-09 contains the specifications resulting from the September 2000 TSG meetings.

3) With "track changes" disabled, paste the entire CR form (use CTRL-A to select it) into the specification just in front of the clause containing the first piece of changed text. Delete those parts of the specification which are not relevant to the change request.

## 10.5 Extensions for future releases in RANAP, RNSAP, NBAP; and SABP ~~RANAP/SABP/RNSAP/NBAP specific extension rules with Backward Compatibility consideration~~

The following clauses contain rules for extension mechanisums of ASN.1 for RANAP, SABP, RNSAP and NBAP. The purpose of these rules is to guarantee backward compatibility for ASN.1.

### 10.5.1 Allowed Extension

The allowed extension for ASN.1 description in RANAP, SABP, RNSAP and NBAP are:

1) adding New IEs or IE groups which should be achieved by using the protocol extension container (extension by using of ellipsis notation (...) should be avoided) for:
   - adding at the top level of message; and
   - adding in the SEQUENCE type,
2) extending the range of already define IEs which has ellipsis notation(...);
3) changing the assigned criticality information of already defined IEs; and
4) adding new IEs of IE groups after ellipsis notation (...) in the CHOICE type if the ellipsis notation (...) is present.

### 10.5.2 Not Allowed Extension

The not allowed extension for ASN.1 description in RANAP, SABP, RNSAP and NBAP are:

1) deleting the already defined IEs or IE groups when no individual criticality information is defined.
2) adding or deleting the criticality information of existing IEs;
3) deleting the already defined values in the ASN.1 type. Instead, a semantic description is added in order to clarify the behaviour; and
4) changing the presence of already defined IEs with no assigned criticality.

This is because above changes do not guarantee the backward compatibility.

### 10.5.3 Recommendations for extensions for further releases

#### 10.5.3.1 General

This sub-clause gives recommendations for future extensions in versions of the RANAP, RNSAP, NBAP, and SABP where non-backward compatible changes are not acceptable.

#### 10.5.3.2~~10.4.3.2~~ Usage of Presence and Assigned Criticality in Future Releases

##### 10.5.3.2.1 New Procedures

For procedures introduced when the backward compatibility mechanisms are taken into use the following recommendation applies to the Assigned Criticality of the procedure (in the tabular description of messages visible as the Assigned Criticality of the IE Message Type):

| Assigned Criticality | Recommendation | Typical usage |
|---|---|---|
| Ignore | Should be used if<br>• the sender does not care whether or not the procedure is supported<br>• or if the sender "already knows" that the procedure is supported | Typically used for procedures where<br>• the sender do not care whether or not the procedure is supported<br>• or where the usage is dependent on previously exchanged information |
| Ignore and Notify | Should be used if<br>• the sender does not care whether or not the procedure is supported<br>• or if the sender "already knows" that the procedure is supported<br>but need to know whether or not the procedure was understood. | Typically not used. |
| Reject | Should be used if<br>• the procedure shall be rejected when not supported | Typically used for new procedures where the sender has no prior knowledge on whether or not the procedure will be understood |

#### 10.54.3.2.2 New IEs

For new IEs introduced when the backward compatibility mechanisms are taken into use the following recommendation applies to the Assigned Criticality of the IE:

| Presence | Assigned Criticality | Recommendation | Typical usage |
|---|---|---|---|
| Optional | Ignore | Should be used if the sender does not care whether or not the function related to the IE is supported. | Typically used for "non core" features (specification text; "... shall, if supported, ..."). |
| | Ignore and Notify | Should be used if<br>• the sender does not care whether or not the function related to the IE is supported<br>but need to know whether or not the IE was understood. | Typically used for "non core" features (specification text; "... shall, if supported, ..."). |
| | Reject | Should be used if<br>• the alternative to executing the feature related to the IE is rejecting the procedure. | Typically used for "core" features (specification text; "... shall ..."). |
| Mandatory / Conditional | Ignore | Should be used for "core" features where<br>• it is essential that all implementations of future releases support the feature related to the IE<br>• it is possible to inter-work with nodes implementing older releases (not understanding the IE related to the feature and consequently not supporting the feature) | Typically not used.<br>Note 1. |
| | Ignore and Notify | Should be used for "core" features where<br>• it is essential that all implementations of future releases support the feature related to the IE<br>• it is possible to inter-work with nodes implementing older releases (not understanding the IE related to the feature and consequently not supporting the feature)<br>but the sending node need to know whether or not the IE was understood. | Typically not used.<br>Note 1. |
| | Reject | Should be used for "core" features where<br>• it is essential that all implementations of future releases support the feature related to the IE<br>• it is not possible to inter-work with nodes implementing older releases (not supporting the feature) | Typically not used.<br>Note 2. |

NOTE 1: This combination (presence + assigned criticality) could be used as an intermediate state, i.e. when the Assigned Criticality is expected/planned to be changed to "Reject" in the future.

NOTE 2: This combination (presence + assigned criticality) should be avoided since it prevents inter-working with older version of a specification.

### 10.54.3.2.3 Changing the Presence of an IE

The Presence can always be changed in future version of a specification

NOTE: Mandatory and Conditional IEs with Assigned Criticality "Reject" will still cause rejection when missing in a node based on a previous version of the specification (even though changed to Optional).

Recommendation:

The Presence of Mandatory IEs with Assigned Criticality "Reject" should not be changed in future versions of a specification.

The Presence of Conditional IEs with Assigned Criticality "Reject" should not be changed in future versions of a specification, unless it is also ensured that the condition will not result in a requirement to include the IE.

#### 10.54.3.2.4 Changing the Assigned Criticality of an IE

The Assigned Criticality can always be changed in future version of a specification.

NOTE: The behaviour for missing IEs will remain unchanged when inter-working with a node based on a previous version of the specification.

Recommendation:

When changing the Assigned Criticality of Mandatory and Conditional IEs with Assigned Criticality "Reject" in future versions of a specification special attention should be paid to inter-working between different versions of the specification.

#### 10.54.3.2.5 Removing IEs

Any IE (with Assigned Criticality) can be removed in future version of a specification.

NOTE: Mandatory and Conditional IEs with Assigned Criticality "Reject" will still cause rejection when missing in a node based on a previous version of the specification (even though changed to Optional).

Recommendation:

Mandatory IEs with Assigned Criticality "Reject" should not be removed in future versions of a specification.

Conditional IEs with Assigned Criticality "Reject" should not be removed in future versions of a specification, unless it is also ensured that the condition, if evaluated for the message where the IE is removed by a node based on a previous version of the specification, will not result in a requirement to include the IE.

**3GPP TSG-RAN WG2 Meeting #21**
**Busan, Korea, May 21st – 25th, 2001**

***R2-011486***

*CR-Form-v3*

# CHANGE REQUEST

⌘ **25.921** **CR** **021** ⌘ rev **-** ⌘ Current version: **4.0.0** ⌘

*For **HELP** on using this form, see bottom of this page or look at the pop-up text over the ⌘ symbols.*

***Proposed change affects:*** ⌘ (U)SIM ☐ ME/UE ☒ Radio Access Network ☒ Core Network ☐

| | | |
|---|---|---|
| ***Title:*** | ⌘ | Addition of Recommendations for Extensions in RANAP, RNSAP, NBAP, and SABP |
| ***Source:*** | ⌘ | TSG-RAN WG3 |
| ***Work item code:*** | ⌘ | TEI |
| ***Date:*** | ⌘ | 25 May, 2001 |
| ***Category:*** | ⌘ | **A** |
| ***Release:*** | ⌘ | REL-4 |

*Use one of the following categories:*
***F*** *(essential correction)*
***A*** *(corresponds to a correction 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.

*Use one of the following releases:*
*2 (GSM Phase 2)*
*R96 (Release 1996)*
*R97 (Release 1997)*
*R98 (Release 1998)*
*R99 (Release 1999)*
*REL-4 (Release 4)*
*REL-5 (Release 5)*

| | | |
|---|---|---|
| ***Reason for change:*** | ⌘ | Xxxxxxxxx |
| ***Summary of change:*** | ⌘ | Recommendations for extensions in future versions of specifications (future releases) has been added with respect to:<br>1. New procedures<br>2. New IEs<br>3. Changing the Presence of an existing IE.<br>4. Changing the Assigned Criticality of an existing IE.<br>5. Removing an existing IE.<br><br>Further more, the title of chapter 10.5 is aligned with the title of chapter 10.4 (Extensions for future releases in RRC). |
| ***Consequences if not approved:*** | ⌘ | The guidelines in this CR will not be maintained once more and more companies change to new delegates. This gives a risk that further updates of the specifications will be made in an undesirable way. |

| | | | | |
|---|---|---|---|---|
| ***Clauses affected:*** | ⌘ | 10.5, 10.5.3 | | |
| ***Other specs affected:*** | ⌘ | ☒ Other core specifications<br>☐ Test specifications<br>☐ O&M Specifications | ⌘ | TR 25.921 CRxx (Rel. '99) |
| ***Other comments:*** | ⌘ | | | |

**How to create CRs using this form:**
Comprehensive information and tips about how to create CRs can be found at: http://www.3gpp.org/3G_Specs/CRs.htm. Below is a brief summary:

1) Fill out the above form. The symbols above marked ⌘ contain pop-up help information about the field that they are closest to.

2) Obtain the latest version for the release of the specification to which the change is proposed. Use the MS Word "revision marks" feature (also known as "track changes") when making the changes. All 3GPP specifications can be downloaded from the 3GPP server under ftp://www.3gpp.org/specs/ For the latest version, look for the directory name with the latest date e.g. 2000-09 contains the specifications resulting from the September 2000 TSG meetings.

3) With "track changes" disabled, paste the entire CR form (use CTRL-A to select it) into the specification just in front of the clause containing the first piece of changed text. Delete those parts of the specification which are not relevant to the change request.

## 10.5 Extensions for future releases in RANAP, RNSAP, NBAP; and SABP ~~RANAP/SABP/RNSAP/NBAP specific extension rules with Backward Compatibility consideration~~

The following clauses contain rules for extension mechanisums of ASN.1 for RANAP, SABP, RNSAP and NBAP. The purpose of these rules is to guarantee backward compatibility for ASN.1.

### 10.5.1 Allowed Extension

The allowed extension for ASN.1 description in RANAP, SABP, RNSAP and NBAP are:

1) adding New IEs or IE groups which should be achieved by using the protocol extension container (extension by using of ellipsis notation (...) should be avoided) for:
   - adding at the top level of message; and
   - adding in the SEQUENCE type,
2) extending the range of already define IEs which has ellipsis notation(...);
3) changing the assigned criticality information of already defined IEs; and
4) adding new IEs of IE groups after ellipsis notation (...) in the CHOICE type if the ellipsis notation (...) is present.

### 10.5.2 Not Allowed Extension

The not allowed extension for ASN.1 description in RANAP, SABP, RNSAP and NBAP are:

1) deleting the already defined IEs or IE groups when no individual criticality information is defined.
2) adding or deleting the criticality information of existing IEs;
3) deleting the already defined values in the ASN.1 type. Instead, a semantic description is added in order to clarify the behaviour; and
4) changing the presence of already defined IEs with no assigned criticality.

This is because above changes do not guarantee the backward compatibility.

### 10.5.3 Recommendations for extensions for further releases

#### 10.5.3.1 General

This sub-clause gives recommendations for future extensions in versions of the RANAP, RNSAP, NBAP, and SABP where non-backward compatible changes are not acceptable.

#### 10.54.3.2 Usage of Presence and Assigned Criticality in Future Releases

##### 10.5.3.2.1 New Procedures

For procedures introduced when the backward compatibility mechanisms are taken into use the following recommendation applies to the Assigned Criticality of the procedure (in the tabular description of messages visible as the Assigned Criticality of the IE Message Type):

| Assigned Criticality | Recommendation | Typical usage |
|---|---|---|
| Ignore | Should be used if<br>• the sender does not care whether or not the procedure is supported<br>• or if the sender "already knows" that the procedure is supported | Typically used for procedures where<br>• the sender do not care whether or not the procedure is supported<br>• or where the usage is dependent on previously exchanged information |
| Ignore and Notify | Should be used if<br>• the sender does not care whether or not the procedure is supported<br>• or if the sender "already knows" that the procedure is supported<br>but need to know whether or not the procedure was understood. | Typically not used. |
| Reject | Should be used if<br>• the procedure shall be rejected when not supported | Typically used for new procedures where the sender has no prior knowledge on whether or not the procedure will be understood |

#### 10.54.3.2.2 New IEs

For new IEs introduced when the backward compatibility mechanisms are taken into use the following recommendation applies to the Assigned Criticality of the IE:

| Presence | Assigned Criticality | Recommendation | Typical usage |
|---|---|---|---|
| Optional | Ignore | Should be used if the sender does not care whether or not the function related to the IE is supported. | Typically used for "non core" features (specification text; "... shall, if supported, ..."). |
| | Ignore and Notify | Should be used if<br>• the sender does not care whether or not the function related to the IE is supported<br>but need to know whether or not the IE was understood. | Typically used for "non core" features (specification text; "... shall, if supported, ..."). |
| | Reject | Should be used if<br>• the alternative to executing the feature related to the IE is rejecting the procedure. | Typically used for "core" features (specification text; "... shall ..."). |
| Mandatory / Conditional | Ignore | Should be used for "core" features where<br>• it is essential that all implementations of future releases support the feature related to the IE<br>• it is possible to inter-work with nodes implementing older releases (not understanding the IE related to the feature and consequently not supporting the feature) | Typically not used.<br>Note 1. |
| | Ignore and Notify | Should be used for "core" features where<br>• it is essential that all implementations of future releases support the feature related to the IE<br>• it is possible to inter-work with nodes implementing older releases (not understanding the IE related to the feature and consequently not supporting the feature)<br>but the sending node need to know whether or not the IE was understood. | Typically not used.<br>Note 1. |
| | Reject | Should be used for "core" features where<br>• it is essential that all implementations of future releases support the feature related to the IE<br>• it is not possible to inter-work with nodes implementing older releases (not supporting the feature) | Typically not used.<br>Note 2. |

NOTE 1:. This combination (presence + assigned criticality) could be used as an intermediate state, i.e. when the Assigned Criticality is expected/planned to be changed to "Reject" in the future.

NOTE 2:. This combination (presence + assigned criticality) should be avoided since it prevents inter-working with older version of a specification.

### 10.54.3.2.3 Changing the Presence of an IE

The Presence can always be changed in future version of a specification

NOTE:. Mandatory and Conditional IEs with Assigned Criticality "Reject" will still cause rejection when missing in a node based on a previous version of the specification (even though changed to Optional).

Recommendation:

The Presence of Mandatory IEs with Assigned Criticality "Reject" should not be changed in future versions of a specification.

The Presence of Conditional IEs with Assigned Criticality "Reject" should not be changed in future versions of a specification, unless it is also ensured that the condition will not result in a requirement to include the IE.

#### 10.54.3.2.4 Changing the Assigned Criticality of an IE

The Assigned Criticality can always be changed in future version of a specification.

NOTE: The behaviour for missing IEs will remain unchanged when inter-working with a node based on a previous version of the specification.

Recommendation:

When changing the Assigned Criticality of Mandatory and Conditional IEs with Assigned Criticality "Reject" in future versions of a specification special attention should be paid to inter-working between different versions of the specification.

#### 10.54.3.2.5 Removing IEs

Any IE (with Assigned Criticality) can be removed in future version of a specification.

NOTE: Mandatory and Conditional IEs with Assigned Criticality "Reject" will still cause rejection when missing in a node based on a previous version of the specification (even though changed to Optional).

Recommendation:

Mandatory IEs with Assigned Criticality "Reject" should not be removed in future versions of a specification.

Conditional IEs with Assigned Criticality "Reject" should not be removed in future versions of a specification, unless it is also ensured that the condition, if evaluated for the message where the IE is removed by a node based on a previous version of the specification, will not result in a requirement to include the IE.

**3GPP TSG-RAN WG2 Meeting #21**
**Busan, Korea, May 21st-25th 2001**

***R2-011487***

*CR-Form-v3*

# CHANGE REQUEST

⌘ **25.921** **CR** **022** ⌘ rev **-** ⌘ Current version: **3.3.0** ⌘

*For **HELP** on using this form, see bottom of this page or look at the pop-up text over the ⌘ symbols.*

***Proposed change affects:*** ⌘ (U)SIM ☐ ME/UE **X** Radio Access Network **X** Core Network ☐

| | | |
|---|---|---|
| ***Title:*** | ⌘ | Clean-up with regard to RAN WG3 Practise of Specifying Control Plane Protocols |
| ***Source:*** | ⌘ | TSG-RAN WG3 |
| ***Work item code:*** | ⌘ | TEI |
| ***Date:*** | ⌘ | 25 May, 2001 |
| ***Category:*** | ⌘ | **F** |
| ***Release:*** | ⌘ | R99 |

*Use one of the following categories:*
***F*** *(essential correction)*
***A*** *(corresponds to a correction 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.

*Use one of the following releases:*
*2* *(GSM Phase 2)*
*R96* *(Release 1996)*
*R97* *(Release 1997)*
*R98* *(Release 1998)*
*R99* *(Release 1999)*
*REL-4* *(Release 4)*
*REL-5* *(Release 5)*

***Reason for change:*** ⌘

Inconsistency between the guidelines and how they are applied
The guidelines are not in-line with the current RAN3 control plane specifications (RANAP, RNSAP, NBAP; and SABP). Several options and guidelines are out of date, are misleading, and need thus to be corrected or removed.

Differences between RAN2 and RAN3 approaches
In RAN2 and RAN3 sometimes different approaches are made, while the guidelines mainly reflects the RAN2 approach. Therefore there is a need for more generalised guidelines, but still leave room to document how a guideline is specialised in RAN2 and RAN3. Note: A separate CR proposes changes that are either RAN2 specific or common to RAN2 and RAN3.

Summary of change: ⌘

The following changes have been made:

- Subclause 4.3.1, New messages:
  The paragraph related to the "network part" indicates that it is not required that new (not understood) messages shall be possible to introduce without causing any damage. However, this is incorrect and the paragraph stating this exception is removed.
- Clause 6, Specification and Description Language:
  The Subclause gives the impression that WG3 uses SDL. It is clarified that SDLs are not used in WG3 either.
- Subclause 9.1, Tabular description of messages and IEs:
  The current subclause is RRC specific. A new subclause 9.1A, based on subclause 9.1, is created to reflect the usage of the tabular description in RAN WG3.

  The heading of subclause 9.1 are also changed to reflect the split into subclauses 9.1 and 9.1a.
- Subclause 9.2, Basic types:

| | | |
|---|---|---|
| | | The tabular layout used in this subclause is RRC specific. A note is added indicating that the tabular layout used in the subclause is from RRC but to be considered an example and that the basic types as such anyhow are applicable to the RAN WG3 specifications.<br><br>Further more, a few notations used in RAN WG3 have been added to the basic types Enumerated, Bit string, and Octet string.<br>• Subclause 11.1, Selection of transfer syntax specification method:<br>It is clarified that the RAN WG3 specifications uses octet aligned PER.<br>• Subclause 11.2, Specialised encoding:<br>The heading of this subclause is changed to reflect that the subclause is not applicable to the RAN WG3 specifications.<br>Based on comments received during RAN3#21, the following additional changes have been made:<br>- 9.1a.1.1.2: Rephrase Conditional presence to express that if condition is true, IE shall be present;<br>- 9.1.a.1.1.7: Add: "If an IE/IE group is not understood or missing"<br>- 9.1a.2: Add: "The inclusion of the criticality and assigned criticality columns is optional, but shall be included if separate criticality needs to be indicated". |
| ***Consequences if not approved:*** | ⌘ | The guidelines can not be applied to the specifications and if they are anyway applied there is a risk that further updates of the specifications will be made in an undesirable way. |

| | | | | | |
|---|---|---|---|---|---|
| ***Clauses affected:*** | ⌘ | 4.3.1, 6, 9.1, 9.1a (new), 9.2 (hanging text before 9.2.1), 9.2.1, 9.2.4, 9.2.5, 11.1, and 11.2 | | | |
| ***Other specs affected:*** | ⌘ | X | Other core specifications | ⌘ | TR 25.921 CR023 (Rel. 4) |
| | | | Test specifications | | |
| | | | O&M Specifications | | |
| ***Other comments:*** | ⌘ | | | | |

**How to create CRs using this form:**

Comprehensive information and tips about how to create CRs can be found at: http://www.3gpp.org/3G_Specs/CRs.htm. Below is a brief summary:

1) Fill out the above form. The symbols above marked ⌘ contain pop-up help information about the field that they are closest to.

2) Obtain the latest version for the release of the specification to which the change is proposed. Use the MS Word "revision marks" feature (also known as "track changes") when making the changes. All 3GPP specifications can be downloaded from the 3GPP server under ftp://www.3gpp.org/specs/ For the latest version, look for the directory name with the latest date e.g. 2000-09 contains the specifications resulting from the September 2000 TSG meetings.

3) With "track changes" disabled, paste the entire CR form (use CTRL-A to select it) into the specification just in front of the clause containing the first piece of changed text. Delete those parts of the specification which are not relevant to the change request.

### 4.3.1 New messages

New message types shall be able to be introduced without causing any damage. New messages not understood shall be discarded by the receiving entity.

As an exception to this principle it can be possible to define a mechanism that allows a different behaviour when a specific reaction is requested from the receiving entity. This mechanism has to be implemented from the beginning. A special care has to be taken into account when defining broadcast messages and the associated Error handling. Further refinement on this paragraph is needed.

~~Such a mechanism is not required inside the network part.~~

# 6 Specification and Description Language

The groups are encouraged to use of SDL where appropriate. The SDL code included in the standards should follow the descriptive SDL guidelines from ETSI TC-MTS (DEG MTS-00050) as closely as possible.

The groups themselves should decide how SDL is used.

In some protocol parts, text is more adapted (e.g.: algorithm or multiplexing), in some other parts SDL is better.

SDL is adapted for describing the observable behaviour of a protocol layer.

In ~~TSG-RAN WG2,~~this release the specifications shall not use SDL for the normative part of the specifications ~~in the present version~~.

## 9.1 Tabular description of messages and IEs in RRC

**<Editor's note: The rest of the subclause is unchanged and has been removed.>**

# 9.1a Tabular description of messages and IEs in RANAP, RNSAP, NBAP, and SABP

## 9.1a.1 Message description

A 'Message description' subclause includes one subclause per message.

A message is described with, in this order:

- A table describing a list of information elements;
- Explanatory clauses, mainly for describing textually conditions for presence or absence and range bounds for some IEs/IE groups.

### 9.1a.1.1 The Information Element table

The table used in RANAP, RNSAP, NBAP, and SABP is composed of 7 columns, labelled and presented as shown below.

| IE/Group Name | Presence | Range | IE type and reference | Semantics description | Criticality | Assigned Criticality |
|---|---|---|---|---|---|---|
| | | | | | | |

NOTE: Indentations are used to visualise the embedding level of an "IE/Group".

Indentations are explicitly written with the character ">" as well as by use of ruler indentations, one per level of indentation. Indentations of lines can be found in the IE/Group Name column.

Each line corresponds either to an IE or to a IE group. An IE group includes all the IEs in following lines until, and not including, a line with the same indentation as the group line.

#### 9.1a.1.1.1 IE/Group Name column

This column gives the local name of the IE or of a group of IEs. This name is significant only within the scope of the described message, and must appear only once in the column at the same level of indentation. It is a free text, which should be chosen to reflect the meaning of the IE or group of IEs. This text is to be used to refer to the IE or the group of IEs in the procedure specification as described in clause 7.

The name of an IE group shall be given in bold font.

The first word 'choice' has a particular meaning, and must not be used otherwise.

#### 9.1a.1.1.2 Presence and Range columns

These columns provide most of the information about the presence, absence and number of instance of the IE (in the message or in the group) or group of IEs. The different possibilities for these columns are described one by one.

At least one of the Presence and Range columns shall be filled.

The meaning of the Presence column is summarised below:

M Mandatorily present.

A value for that information is always needed, and no information is provided about a particular default value.

C Conditional.

The IE/IE group is required to be present when a condition is met that can be evaluated on the sole basis of the content of the message. If the condition is not met, the IE/IE group shall not be included.

O Optional.

The presence or absence is significant and modifies the behaviour of the receiver.

#### 9.1a.1.1.2.1 Mandatory

| IE/Group Name | Presence | Range | IE type and reference | Semantics description | Criticality | Assigned Criticality |
|---|---|---|---|---|---|---|
| Name | M | | | | | |
| **Name** | | 1 | | | | |

For an IE M indicates that one and only one instance of *Name* IE shall be present in that part of the message.

For an IE group 1 in the Range column indicates that one and only one instance of *Name* IE group shall be present in that part of the message.

#### 9.1a.1.1.2.2 Optional

| IE/Group Name | Presence | Range | IE type and reference | Semantics description | Criticality | Assigned Criticality |
|---|---|---|---|---|---|---|
| Name | O | | | | | |
| **Name** | | 0..1 | | | | |

For an IE O indicates that one and only one instance of *Name* IE may be present in that part of the message and that the sender can choose not to include it.

For an IE group 0..1 in the Range column indicates that one and only one instance of *Name* IE group may be present in that part of the message and that the sender can choose not to include it.

#### 9.1a.1.1.2.3 Conditional

| IE/Group Name | Presence | Range | IE type and reference | Semantics description | Criticality | Assigned Criticality |
|---|---|---|---|---|---|---|
| Name | C - *cond* | | | | | |
| **Name** | C - *cond* | nn..<sym> | | | | |

For an IE/IE group C indicates that the requirement for presence or absence of the IE/IE group depends on a condition described in a textual form in an explanatory clause. "*cond*" stands for a free text that is used as a reference in the title of the explanatory clause.

The result of evaluating the condition (if the condition is met or not) may mean that the IE is:

- Mandatorily present, where nn is giving the minimum number of instances that shall be present and "sym" is a symbolic name giving the maximum number of instances than may be present.
- Mandatorily absent.
- Optional, where nn is giving the minimum number of instances and "sym" is a symbolic name giving the maximum number of instances than may be present..

#### 9.1a.1.1.2.4 Choice

| IE/Group Name | Presence | Range | IE type and reference | Semantics description | Criticality | Assigned Criticality |
|---|---|---|---|---|---|---|
| Choice *name* | | | | | | |
| *>Name1* | | | | | | |
| *>Name2* | | | | | | |

A 'choice' is distinguished from IEs/IE groups by the use of 'choice' as first word in the name.

The Presence columns are filled normally for the group line (Choice *name*). They are not filled for the choice tags, e.g. "*Name1*".

#### 9.1a.1.1.2.5 Sets

In general, this indicates that more than one instance of an IE/IE group may be present in the message.

The two lines below indicate different allowed alternatives.

| IE/Group Name | Presence | Range | IE type and reference | Semantics description | Criticality | Assigned Criticality |
|---|---|---|---|---|---|---|
| **Name** | | nn..pp | | | | |
| **Name** | | nn..sym | | | | |

Where *nn* and *pp* stand for positive integers (non zero) and *sym* for a symbolic name. The number pp must be greater than nn.

The use of a symbolic name for the upper range bound indicates that the value is given in a textual clause.

The notation '..' can be replaced with the same meaning by 'to'.

This indicates that a number of instances of the IE/IE group shall be present in the message/embedding IE group. The order is significant.

The *nn..pp* case indicates that the number of instances is between nn and pp, inclusively. This means that *nn* instances shall be present in the message, that up to *pp* instances may be present.

The *nn..sym* case indicates that the number of instances is between nn and the value represented by "sym", inclusively. This means that *nn* instances shall be present in the message, that up to *sym* instances may be present.

### 9.1a.1.1.3 IE Type and reference column

This column is not filled for IE groups and must be filled for IEs.

This column includes the reference to a more detailed abstract description of the IE. This includes:

a) A reference to a subclause in the "Information Element Description" clause in the same document; typically the subclause number and titles are given, and if possible this should be a hypertext link. Titles need only be given if the name of the type is different from the name of the IE;

b) A reference to another document, and to a subclause in the Information Element Description clause in the indicated document; typically only the subclause title is indicated;

### 9.1a.1.1.4 Semantics description column

Filling this column is optional. It should be use to clarify the meaning of the IE/IE group.

### 9.1a.1.1.5 Expressing differences between FDD and TDD modes

Differences between FDD and TDD can be expressed either by separate tabular description of the messages or by comments in the semantics description column. The former alternative should be used for messages with major differences and the latter for messages with minor differences between FDD and TDD.

### 9.1a.1.1.6 Criticality column

Each IE or IE group may have criticality information applied to it.
The following cases are possible:

| | |
|---|---|
| – | No criticality information is applied explicitly. |
| YES | Criticality information is applied. 'YES' is usable only for non-repeatable information elements. |
| GLOBAL | The information element and all its repetitions together have one common criticality information. 'GLOBAL' is usable only for repeatable information elements. |
| EACH | Each repetition of the information element has its own criticality information. It is not allowed to assign different criticality values to the repetitions. 'EACH' is usable only for repeatable information elements. |

#### 9.1a.1.1.7 Assigned Criticality column

This column provides the actual criticality information as defined in subclause 10.3.2 in RANAP, RNSAP, NBAP, and SABP.

If an IE/IE group is not understood or missing , the receiving node shall take different actions depending on the value of the Criticality Information. The three possible values of the Criticality Information for an IE/IE group are:

1. Reject IE;
2. Ignore IE and Notify Sender;
3. Ignore IE.

### 9.1a.1.2 Explanatory clauses

This includes the subclauses needed to elaborate conditions and symbolic names (e.g., range bounds). There must be one explanatory clause for each named condition, and for each symbolic name. The text must give the information sufficient to decide whether the IE/IE group is to be included or not, or the value of the symbolic name. The text shall be given in separate tables for the conditions and range bounds.

## 9.1a.2 IE type description

This describes IE types referred elsewhere, either in the description of a message or in the description of another IE type. The description of an IE type must be as generic as possible, i.e., independent of any specific use. A type should as far as possible not be defined in multiple places in a specification.

An 'IE description' subclause includes one subclause per IE type.

The description of an IE type is done as a table similar to that used for the description of messages. In RANAP, RNSAP, NBAP, and SABP this table has the layout as shown below.

| IE/Group Name | Presence | Range | IE type and reference | Semantics description | Criticality | Assigned Criticality |
|---|---|---|---|---|---|---|
| | | | | | | |

The different columns are filled as message description columns are filled with the addition that in the IE Type and reference column also the use of a basic types defined in subclause 9.2 in this document is allowed. These basic types shall be considered as pre-defined and for those no reference is necessary. For IE type descriptions, explanatory clauses should also be used as described in subclause 9.1a.1.2. The inclusion of the criticality and assigned criticality columns is optional, but shall be included if separate criticality needs to be indicated.

## 9.1a.3 Extension for further releases

### 9.1a.3.1 Basic principle

Added elements or choice branches are included where they fit most naturally according to their semantics if the ASN.1 allows (e.g. there exist an extension container in this place). For further information on handling of extensions in RANAP, RNSAP, NBAP, and SABP see subclause 10.5.

## 9.2 Basic types

To reduce the text in tabular descriptions, some basic abstract types of IE are defined in this document.

NOTE: The tabular description in this subclause used to describe different formats of the basic types follow the layout applicable to RRC. However, the basic types as such are applicable also to RANAP, RNSAP, NBAP, and SABP.

### 9.2.1 Enumerated

| IE/Group Name | Need | Multi | Type and reference | Semantics description | Version |
|---|---|---|---|---|---|
| | | | Enumerated (*c1*, *c2*, *c3*) | | |
| | | | Enumerated (x1..*xn*) | | |
| | | | Enumerated (*c1*, *c2*, *c3*, ...) | | |
| | | | Enumerated (*c1*, *c2*, *c3*, ..., *c4*, *c5*) | | |

In the first format, *c1*, *c2*, *c3* stands for a list of 2 or more symbolic names separated by commas.

In the second format, *x* is some character string, possibly empty, *n* is an integer, and indicates a list of *n* different values, with no particular property except for being distinct.

In the third format the IE value range is c1, c2, and c3 with an infinite extension possibility.

In the fourth format the IE value range is c1, c2, c3, c4, and c5 with an infinite extension possibility. The values c4 and c5 have been added in a backward compatible way, typically in a release later than release '99.

This indicates that the value of the IE when present takes one and only one of the values indicated in the list.

### 9.2.4 Bit string

| IE/Group Name | Need | Multi | Type and reference | Semantics description | Version |
|---|---|---|---|---|---|
| | | | Bit string | | |
| | | | Bit string (*nn*) | | |
| | | | Bit string (*mm..pp*) | | |
| | | | Bit string (*mm..pp*, …) | | |

Where *nn*, *mm*, and *pp* ~~is a~~are positive non-null numbers indicating the fixed size, lower bound, and upper bound of the number of bits in the string respectively. In the fourth format the number of bits in the string is extensible beyond the upper bound. If no size is given the bit string can have any number of bits.

Bit strings are unstructured as seen by the protocol. They are typically transparent fields, used by other protocols (other layers or others systems), or as containers on which bit-per-bit boolean operations are done (e.g., ciphered containers).

### 9.2.5 Octet string

| IE/Group Name | Need | Multi | Type and reference | Semantics description | Version |
|---|---|---|---|---|---|
| | | | Octet string | | |
| | | | Octet string (*nn*) | | |
| | | | Octet string (*mm..pp*) | | |
| | | | Octet string (*mm..pp, …*) | | |

Where *nn*, *mm*, and, *pp* ~~is a~~are positive non-null numbers indicating the fixed size, lower bound, and upper bound of the number of octets in the string respectively. In the fourth format the number of octets in the string is extensible beyond the upper bound. If no size is given the octet string can have any number of bits.

This is just a shortcut for bit strings with a length a multiple of 8, and the same comments as on bit strings apply.

It should be noted that this does not indicate that the information is 'octet aligned', which is an encoding notion (and hence foreign to the tabular format) according to which in the transfer syntax a field starts at an octet boundary relatively to the beginning of the message (or other container).

## 11.1 Selection of transfer syntax specification method

For RRC Basic Packed Encoding Rules (BASIC-PER) ~~unaligned~~ PER Unaligned Variant and possible use of specialised encoding is chosen.

For RANAP, RNSAP, NBAP, and SABP Basic Packed Encoding Rules (BASIC-PER) Aligned Variant is chosen.

## 11.2 Specialised encoding (only RRC)

**<Editor's note: The rest of the subclause is unchanged and has been removed.>**

**3GPP TSG-RAN WG2 Meeting #21**
**Busan, Korea, May 21st-25th 2001**

***R2-011488***

*CR-Form-v3*

# CHANGE REQUEST

⌘ **25.921** **CR** **023** ⌘ rev **-** ⌘ Current version: **4.0.0** ⌘

*For **HELP** on using this form, see bottom of this page or look at the pop-up text over the ⌘ symbols.*

***Proposed change affects:*** ⌘ (U)SIM ☐ ME/UE **X** Radio Access Network **X** Core Network ☐

| | | |
|---|---|---|
| ***Title:*** | ⌘ | Clean-up with regard to RAN WG3 Practise of Specifying Control Plane Protocols |
| ***Source:*** | ⌘ | TSG-RAN WG3 |
| ***Work item code:*** | ⌘ | TEI |
| ***Date:*** | ⌘ | 25 May, 2001 |
| ***Category:*** | ⌘ | **A** |
| ***Release:*** | ⌘ | REL-4 |

*Use one of the following categories:*
***F*** *(essential correction)*
***A*** *(corresponds to a correction 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.

*Use one of the following releases:*
*2* *(GSM Phase 2)*
*R96* *(Release 1996)*
*R97* *(Release 1997)*
*R98* *(Release 1998)*
*R99* *(Release 1999)*
*REL-4* *(Release 4)*
*REL-5* *(Release 5)*

***Reason for change:*** ⌘

Inconsistency between the guidelines and how they are applied
The guidelines are not in-line with the current RAN3 control plane specifications (RANAP, RNSAP, NBAP; and SABP). Several options and guidelines are out of date, are misleading, and need thus to be corrected or removed.

Differences between RAN2 and RAN3 approaches
In RAN2 and RAN3 sometimes different approaches are made, while the guidelines mainly reflects the RAN2 approach. Therefore there is a need for more generalised guidelines, but still leave room to document how a guideline is specialised in RAN2 and RAN3. Note: A separate CR proposes changes that are either RAN2 specific or common to RAN2 and RAN3.

Summary of change: ⌘

The following changes have been made:

- Subclause 4.3.1, New messages:
  The paragraph related to the "network part" indicates that it is not required that new (not understood) messages shall be possible to introduce without causing any damage. However, this is incorrect and the paragraph stating this exception is removed.
- Clause 6, Specification and Description Language:
  The Subclause gives the impression that WG3 uses SDL. It is clarified that SDLs are not used in WG3 either.
- Subclause 9.1, Tabular description of messages and IEs:
  The current subclause is RRC specific. A new subclause 9.1A, based on subclause 9.1, is created to reflect the usage of the tabular description in RAN WG3.

  The heading of subclause 9.1 are also changed to reflect the split into subclauses 9.1 and 9.1a.

| | | |
|---|---|---|
| | | • Subclause 9.2, Basic types:<br>The tabular layout used in this subclause is RRC specific. A note is added indicating that the tabular layout used in the subclause is from RRC but to be considered an example and that the basic types as such anyhow are applicable to the RAN WG3 specifications.<br><br>Further more, a few notations used in RAN WG3 have been added to the basic types Enumerated, Bit string, and Octet string.<br>• Subclause 11.1, Selection of transfer syntax specification method:<br>It is clarified that the RAN WG3 specifications uses octet aligned PER.<br>• Subclause 11.2, Specialised encoding:<br>The heading of this subclause is changed to reflect that the subclause is not applicable to the RAN WG3 specifications.<br><br>Based on comments received during RAN3#21, the following additional changes have been made:<br>- 9.1a.1.1.2: Rephrase Conditional presence to express that if condition is true, IE shall be present;<br><br>- 9.1.a.1.1.7: Add: "If an IE/IE group is not understood or missing"<br><br>- 9.1a.2: Add: "The inclusion of the criticality and assigned criticality columns is optional, but shall be included if separate criticality needs to be indicated". |
| ***Consequences if not approved:*** | ⌘ | The guidelines can not be applied to the specifications and if they are anyway applied there is a risk that further updates of the specifications will be made in an undesirable way. |

| | | | | | |
|---|---|---|---|---|---|
| ***Clauses affected:*** | ⌘ | 4.3.1, 6, 9.1, 9.1a (new), 9.2 (hanging text before 9.2.1), 9.2.1, 9.2.4, 9.2.5, 11.1, and 11.2 | | | |
| ***Other specs affected:*** | ⌘ | **X** | Other core specifications | ⌘ | TR 25.921 CR022 (Rel. '99) |
| | | | Test specifications | | |
| | | | O&M Specifications | | |
| ***Other comments:*** | ⌘ | | | | |

**How to create CRs using this form:**

Comprehensive information and tips about how to create CRs can be found at: http://www.3gpp.org/3G_Specs/CRs.htm. Below is a brief summary:

1) Fill out the above form. The symbols above marked ⌘ contain pop-up help information about the field that they are closest to.

2) Obtain the latest version for the release of the specification to which the change is proposed. Use the MS Word "revision marks" feature (also known as "track changes") when making the changes. All 3GPP specifications can be downloaded from the 3GPP server under ftp://www.3gpp.org/specs/ For the latest version, look for the directory name with the latest date e.g. 2000-09 contains the specifications resulting from the September 2000 TSG meetings.

3) With "track changes" disabled, paste the entire CR form (use CTRL-A to select it) into the specification just in front of the clause containing the first piece of changed text. Delete those parts of the specification which are not relevant to the change request.

### 4.3.1 New messages

New message types shall be able to be introduced without causing any damage. New messages not understood shall be discarded by the receiving entity.

As an exception to this principle it can be possible to define a mechanism that allows a different behaviour when a specific reaction is requested from the receiving entity. This mechanism has to be implemented from the beginning. A special care has to be taken into account when defining broadcast messages and the associated Error handling. Further refinement on this paragraph is needed.

~~Such a mechanism is not required inside the network part.~~

# 6 Specification and Description Language

The groups are encouraged to use of SDL where appropriate. The SDL code included in the standards should follow the descriptive SDL guidelines from ETSI TC-MTS (DEG MTS-00050) as closely as possible.

The groups themselves should decide how SDL is used.

In some protocol parts, text is more adapted (e.g.: algorithm or multiplexing), in some other parts SDL is better.

SDL is adapted for describing the observable behaviour of a protocol layer.

In ~~TSG-RAN WG2,~~ this release the specifications shall not use SDL for the normative part of the specifications ~~in the present version~~.

## 9.1 Tabular description of messages and IEs in RRC

**<Editor's note: The rest of the subclause is unchanged and has been removed.>**

# 9.1a Tabular description of messages and IEs in RANAP, RNSAP, NBAP, and SABP

## 9.1a.1 Message description

A 'Message description' subclause includes one subclause per message.

A message is described with, in this order:

- A table describing a list of information elements;
- Explanatory clauses, mainly for describing textually conditions for presence or absence and range bounds for some IEs/IE groups.

### 9.1a.1.1 The Information Element table

The table used in RANAP, RNSAP, NBAP, and SABP is composed of 7 columns, labelled and presented as shown below.

| IE/Group Name | Presence | Range | IE type and reference | Semantics description | Criticality | Assigned Criticality |
|---|---|---|---|---|---|---|
| | | | | | | |

NOTE: Indentations are used to visualise the embedding level of an "IE/Group".

Indentations are explicitly written with the character ">" as well as by use of ruler indentations, one per level of indentation. Indentations of lines can be found in the IE/Group Name column.

Each line corresponds either to an IE or to a IE group. An IE group includes all the IEs in following lines until, and not including, a line with the same indentation as the group line.

#### 9.1a.1.1.1 IE/Group Name column

This column gives the local name of the IE or of a group of IEs. This name is significant only within the scope of the described message, and must appear only once in the column at the same level of indentation. It is a free text, which should be chosen to reflect the meaning of the IE or group of IEs. This text is to be used to refer to the IE or the group of IEs in the procedure specification as described in clause 7.

The name of an IE group shall be given in bold font.

The first word 'choice' has a particular meaning, and must not be used otherwise.

#### 9.1a.1.1.2 Presence and Range columns

These columns provide most of the information about the presence, absence and number of instance of the IE (in the message or in the group) or group of IEs. The different possibilities for these columns are described one by one.

At least one of the Presence and Range columns shall be filled.

The meaning of the Presence column is summarised below:

M Mandatorily present.

A value for that information is always needed, and no information is provided about a particular default value.

C Conditional.

The IE/IE group is required to be present when a condition is met that can be evaluated on the sole basis of the content of the message. If the condition is not met, the IE/IE group shall not be included.

O Optional.

The presence or absence is significant and modifies the behaviour of the receiver.

#### 9.1a.1.1.2.1 Mandatory

| IE/Group Name | Presence | Range | IE type and reference | Semantics description | Criticality | Assigned Criticality |
|---|---|---|---|---|---|---|
| Name | M | | | | | |
| **Name** | | 1 | | | | |

For an IE M indicates that one and only one instance of *Name* IE shall be present in that part of the message.

For an IE group 1 in the Range column indicates that one and only one instance of *Name* IE group shall be present in that part of the message.

#### 9.1a.1.1.2.2 Optional

| IE/Group Name | Presence | Range | IE type and reference | Semantics description | Criticality | Assigned Criticality |
|---|---|---|---|---|---|---|
| Name | O | | | | | |
| **Name** | | 0..1 | | | | |

For an IE O indicates that one and only one instance of *Name* IE may be present in that part of the message and that the sender can choose not to include it.

For an IE group 0..1 in the Range column indicates that one and only one instance of *Name* IE group may be present in that part of the message and that the sender can choose not to include it.

#### 9.1a.1.1.2.3 Conditional

| IE/Group Name | Presence | Range | IE type and reference | Semantics description | Criticality | Assigned Criticality |
|---|---|---|---|---|---|---|
| Name | C - *cond* | | | | | |
| **Name** | C - *cond* | nn..<sym> | | | | |

For an IE/IE group C indicates that the requirement for presence or absence of the IE/IE group depends on a condition described in a textual form in an explanatory clause. "*cond*" stands for a free text that is used as a reference in the title of the explanatory clause.

The result of evaluating the condition (if the condition is met or not) may mean that the IE is:

- Mandatorily present, where nn is giving the minimum number of instances that shall be present and "sym" is a symbolic name giving the maximum number of instances than may be present.
- Mandatorily absent.
- Optional, where nn is giving the minimum number of instances and "sym" is a symbolic name giving the maximum number of instances than may be present..

#### 9.1a.1.1.2.4 Choice

| IE/Group Name | Presence | Range | IE type and reference | Semantics description | Criticality | Assigned Criticality |
|---|---|---|---|---|---|---|
| Choice *name* | | | | | | |
| *>Name1* | | | | | | |
| *>Name2* | | | | | | |

A 'choice' is distinguished from IEs/IE groups by the use of 'choice' as first word in the name.

The Presence columns are filled normally for the group line (Choice *name*). They are not filled for the choice tags, e.g. "*Name1*".

#### 9.1a.1.1.2.5 Sets

In general, this indicates that more than one instance of an IE/IE group may be present in the message.

The two lines below indicate different allowed alternatives.

| IE/Group Name | Presence | Range | IE type and reference | Semantics description | Criticality | Assigned Criticality |
|---|---|---|---|---|---|---|
| **Name** | | nn..pp | | | | |
| **Name** | | nn..sym | | | | |

Where *nn* and *pp* stand for positive integers (non zero) and *sym* for a symbolic name. The number pp must be greater than nn.

The use of a symbolic name for the upper range bound indicates that the value is given in a textual clause.

The notation '..' can be replaced with the same meaning by 'to'.

This indicates that a number of instances of the IE/IE group shall be present in the message/embedding IE group. The order is significant.

The *nn..pp* case indicates that the number of instances is between nn and pp, inclusively. This means that *nn* instances shall be present in the message, that up to *pp* instances may be present.

The *nn..sym* case indicates that the number of instances is between nn and the value represented by "sym", inclusively. This means that *nn* instances shall be present in the message, that up to *sym* instances may be present.

### 9.1a.1.1.3 IE Type and reference column

This column is not filled for IE groups and must be filled for IEs.

This column includes the reference to a more detailed abstract description of the IE. This includes:

a) A reference to a subclause in the "Information Element Description" clause in the same document; typically the subclause number and titles are given, and if possible this should be a hypertext link. Titles need only be given if the name of the type is different from the name of the IE;

b) A reference to another document, and to a subclause in the Information Element Description clause in the indicated document; typically only the subclause title is indicated;

### 9.1a.1.1.4 Semantics description column

Filling this column is optional. It should be use to clarify the meaning of the IE/IE group.

### 9.1a.1.1.5 Expressing differences between FDD and TDD modes

Differences between FDD and TDD can be expressed either by separate tabular description of the messages or by comments in the semantics description column. The former alternative should be used for messages with major differences and the latter for messages with minor differences between FDD and TDD.

### 9.1a.1.1.6 Criticality column

Each IE or IE group may have criticality information applied to it.
The following cases are possible:

| | |
|---|---|
| **–** | No criticality information is applied explicitly. |
| **YES** | Criticality information is applied. 'YES' is usable only for non-repeatable information elements. |
| **GLOBAL** | The information element and all its repetitions together have one common criticality information. 'GLOBAL' is usable only for repeatable information elements. |
| **EACH** | Each repetition of the information element has its own criticality information. It is not allowed to assign different criticality values to the repetitions. 'EACH' is usable only for repeatable information elements. |

#### 9.1a.1.1.7 Assigned Criticality column

This column provides the actual criticality information as defined in subclause 10.3.2 in RANAP, RNSAP, NBAP, and SABP.

If an IE/IE group is not understood or missing, the receiving node shall take different actions depending on the value of the Criticality Information. The three possible values of the Criticality Information for an IE/IE group are:

1. Reject IE;
2. Ignore IE and Notify Sender;
3. Ignore IE.

### 9.1a.1.2 Explanatory clauses

This includes the subclauses needed to elaborate conditions and symbolic names (e.g., range bounds). There must be one explanatory clause for each named condition, and for each symbolic name. The text must give the information sufficient to decide whether the IE/IE group is to be included or not, or the value of the symbolic name. The text shall be given in separate tables for the conditions and range bounds.

## 9.1a.2 IE type description

This describes IE types referred elsewhere, either in the description of a message or in the description of another IE type. The description of an IE type must be as generic as possible, i.e., independent of any specific use. A type should as far as possible not be defined in multiple places in a specification.

An 'IE description' subclause includes one subclause per IE type.

The description of an IE type is done as a table similar to that used for the description of messages. In RANAP, RNSAP, NBAP, and SABP this table has the layout as shown below.

| **IE/Group Name** | **Presence** | **Range** | **IE type and reference** | **Semantics description** | **Criticality** | **Assigned Criticality** |
|---|---|---|---|---|---|---|
| | | | | | | |

The different columns are filled as message description columns are filled with the addition that in the IE Type and reference column also the use of a basic types defined in subclause 9.2 in this document is allowed. These basic types shall be considered as pre-defined and for those no reference is necessary. For IE type descriptions, explanatory clauses should also be used as described in subclause 9.1a.1.2. The inclusion of the criticality and assigned criticality columns is optional, but shall be included if separate criticality needs to be indicated.

## 9.1a.3 Extension for further releases

### 9.1a.3.1 Basic principle

Added elements or choice branches are included where they fit most naturally according to their semantics if the ASN.1 allows (e.g. there exist an extension container in this place). For further information on handling of extensions in RANAP, RNSAP, NBAP, and SABP see subclause 10.5.

## 9.2 Basic types

To reduce the text in tabular descriptions, some basic abstract types of IE are defined in this document.

NOTE: The tabular description in this subclause used to describe different formats of the basic types follow the layout applicable to RRC. However, the basic types as such are applicable also to RANAP, RNSAP, NBAP, and SABP.

### 9.2.1 Enumerated

| IE/Group Name | Need | Multi | Type and reference | Semantics description | Version |
|---|---|---|---|---|---|
| | | | Enumerated (*c1*, *c2*, *c3*) | | |
| | | | Enumerated (x1..*xn*) | | |
| | | | Enumerated (*c1*, *c2*, *c3*, ...) | | |
| | | | Enumerated (*c1*, *c2*, *c3*, ..., *c4*, *c5*) | | |

In the first format, *c1*, *c2*, *c3* stands for a list of 2 or more symbolic names separated by commas.

In the second format, *x* is some character string, possibly empty, *n* is an integer, and indicates a list of *n* different values, with no particular property except for being distinct.

In the third format the IE value range is c1, c2, and c3 with an infinite extension possibility.

In the fourth format the IE value range is c1, c2, c3, c4, and c5 with an infinite extension possibility. The values c4 and c5 have been added in a backward compatible way, typically in a release later than release '99.

This indicates that the value of the IE when present takes one and only one of the values indicated in the list.

### 9.2.4 Bit string

| IE/Group Name | Need | Multi | Type and reference | Semantics description | Version |
|---|---|---|---|---|---|
| | | | Bit string | | |
| | | | Bit string (*nn*) | | |
| | | | Bit string (*mm..pp*) | | |
| | | | Bit string (*mm..pp, …*) | | |

Where *nn*, *mm*, and, *pp* ~~is a~~are positive non-null numbers indicating the fixed size, lower bound, and upper bound of the number of bits in the string respectively. In the fourth format the number of bits in the string is extensible beyond the upper bound. If no size is given the bit string can have any number of bits.

Bit strings are unstructured as seen by the protocol. They are typically transparent fields, used by other protocols (other layers or others systems), or as containers on which bit-per-bit boolean operations are done (e.g., ciphered containers).

### 9.2.5 Octet string

| IE/Group Name | Need | Multi | Type and reference | Semantics description | Version |
|---|---|---|---|---|---|
| | | | Octet string | | |
| | | | Octet string (*nn*) | | |
| | | | Octet string (*mm..pp*) | | |
| | | | Octet string (*mm..pp, …*) | | |

Where *nn*, *mm*, and, *pp* ~~is a~~are positive non-null numbers indicating the lower fixed size, lower bound, and upper bound and upper bound of the number of octets in the string respectively. In the fourth format the number of octets in the string is extensible beyond the upper bound. If no size is given the octet string can have any number of bits.

This is just a shortcut for bit strings with a length a multiple of 8, and the same comments as on bit strings apply.

It should be noted that this does not indicate that the information is 'octet aligned', which is an encoding notion (and hence foreign to the tabular format) according to which in the transfer syntax a field starts at an octet boundary relatively to the beginning of the message (or other container).

## 11.1 Selection of transfer syntax specification method

For RRC Basic Packed Encoding Rules (BASIC-PER) ~~unaligned~~ PER Unaligned Variant and possible use of specialised encoding is chosen.

For RANAP, RNSAP, NBAP, and SABP Basic Packed Encoding Rules (BASIC-PER) Aligned Variant is chosen.

## 11.2 Specialised encoding (only RRC)

**<Editor's note: The rest of the subclause is unchanged and has been removed.>**