TSG-RAN Meeting #23 **RP-040098**
Phoenix, 10-12 March 2004

Title: **CRs on 25.921 R'99 (and linked CRs from later releases)**

Source: **TSG-RAN WG2**

Agenda item: **7.3.3**

Spec	CR	Rev	Phase	Subject	Cat	Version-Current	Version-New	Doc-2nd-Level	Workitem
25.921	049	-	R99	Spare Extension in Data Frame	F	3.9.0	3.10.0	R2-040187	TEI
25.921	050	-	Rel-4	Spare Extension in Data Frame	A	4.6.0	4.7.0	R2-040188	TEI
25.921	051	-	Rel-5	Spare Extension in Data Frame	A	5.3.0	5.4.0	R2-040189	TEI
25.921	52	1	R99	Guideline on release independent ASN.1 updates	F	3.9.0	3.10.0	R2-040319	TEI
25.921	53	2	Rel-4	Guideline on release independent ASN.1 updates	A	4.6.0	4.7.0	R2-040344	TEI
25.921	54	2	Rel-5	Guideline on release independent ASN.1 updates	A	5.3.0	5.4.0	R2-040345	TEI
25.921	55	-	R99	Guideline on the use of variable length containers for late extensions	F	3.9.0	3.10.0	R2-040247	TEI
25.921	56	-	Rel-4	Guideline on the use of variable length containers for late extensions	A	4.6.0	4.7.0	R2-040248	TEI
25.921	57	-	Rel-5	Guideline on the use of variable length containers for late extensions	A	5.3.0	5.4.0	R2-040249	TEI
25.921	58	-	R99	Guideline for the naming of extensions to the RRC ASN.1	F	3.9.0	3.10.0	R2-040300	TEI
25.921	59	-	Rel-4	Guideline for the naming of extensions to the RRC ASN.1	A	4.6.0	4.7.0	R2-040301	TEI
25.921	60	-	Rel-5	Guideline for the naming of extensions to the RRC ASN.1	A	5.3.0	5.4.0	R2-040302	TEI

3GPP TSG-RAN2 Meeting #40 **Tdoc ⌘R2-040187**
Sophia Antipolis, France, 12th – 16th January 2004

CR-Form-v7

CHANGE REQUEST

⌘	**25.921**	**CR**	**049**	⌘**rev**	**-**	⌘	Current version:	**3.9.0**	⌘

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

Proposed change affects:	UICC apps⌘		ME		Radio Access Network	X	Core Network	

Title:	⌘	Spare Extension in Data Frame		
Source:	⌘	RAN WG2		
Work item code:	⌘	TEI	***Date:*** ⌘	12/01/2004
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)* *Rel-6 (Release 6)*

Reason for change:	⌘	It is not clear how to use the Spare Extension in Data Frame of Frame Protocol.
Summary of change:	⌘	- Extension mechanism in Data Frame of Frame Protocol is introduced.
Consequences if not approved:	⌘	In the future, adding new IEs in a backward compatible way is not guaranteed. Impact Analysis: Impact assessment towards the previous version of the specification (same release): This CR has isolated impact on the previous version of the specification because the change only affect on Spare Extension functionality. But since the Spare Extension hasn't been used in this release yet, the impact is very minor. Impact assessment towards the previous release of the specification: Not applicable since this is Rel99 correction.

Clauses affected:	⌘	9a(new)				
		Y	**N**			
Other specs affected:	⌘		X	Other core specifications	⌘	
			X	Test specifications		
			X	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/specs/CR.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.

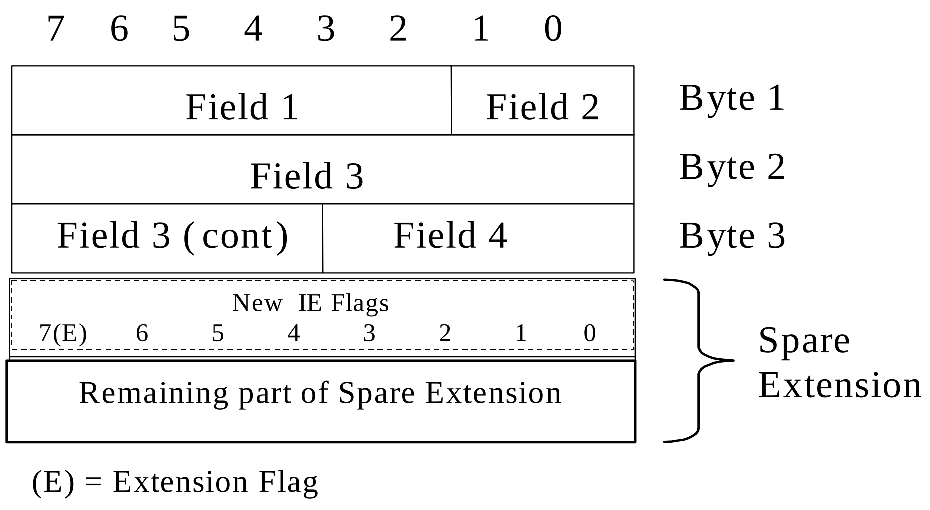
3) With "track changes" disabled, paste the entire CR form (use CTRL-A to select it) into the specification just in front of the clause containing the first piece of changed text. Delete those parts of the specification which are not relevant to the change request.

9a Usage of Iub/Iur Frame Protocol

The following clauses contain guidelines for specification of frame protocols.

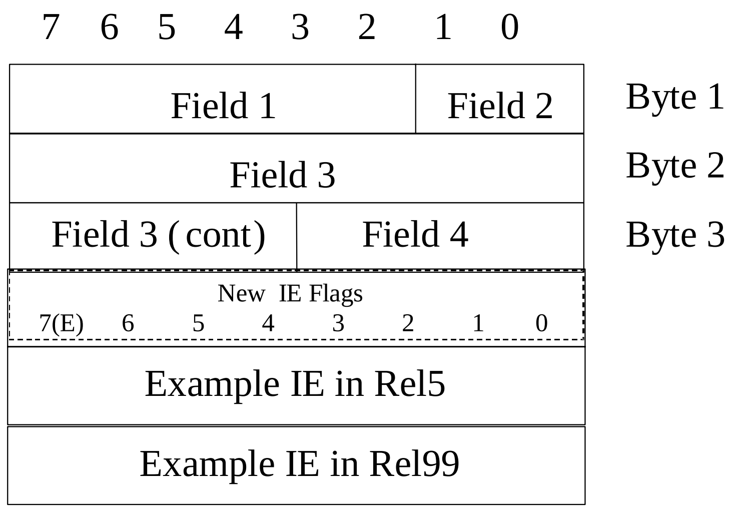
9a.1 Extensions for future releases in Data Frame

Spare Extension is used to define New IEs in the future. When the first IE is added in the Spare Extension in the Data Frame, *New IE Flags* IE shall be added in the first byte of the Spare Extension to indicate the validity of the value of the IEs in the Spare Extension. The last bit position of the *New IE Flags* IE is used as the Extension Flag to allow the extension of the *New IE Flags* IE in the future. The IEs in the Spare Extension will be added in the order in which the IEs are introduced regardless of the release.



In the example below, it is assumed that after *Example IE in Rel5* IE was introduced, *Example IE in Rel99* IE is introduced.

In this example, New IE Flags (0) indicates the validity of *Example IE in Rel5* IE and the New IE Flags (1) indicates the validity of *Example IE in Rel99* IE. The IEs are added in the order of their introduction in the Spare Extension. For the Rel99 and Rel4 nodes, New IE Flags (0) and *Example IE in Rel5* IE will always be seen as Spare Bits while for Rel5 nodes, all the IEs (i.e., New IE Flags (0), New IE Flags(1)), *Example IE in Rel5* IE and *Example IE in Rel99* IE can be used.



Example of Spare Extension Usage

3GPP TSG-RAN2 Meeting #40
Sophia Antipolis, France, 12th – 16th January 2004

Tdoc ⌘R2-040188

CR-Form-v7

CHANGE REQUEST

⌘	**25.921**	**CR**	**050**	⌘**rev**	**-**	⌘	Current version:	**4.6.0**	⌘

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

Proposed change affects:	UICC apps⌘		ME		Radio Access Network	X	Core Network

Title:	⌘	Spare Extension in Data Frame			
Source:	⌘	RAN WG2			
Work item code:	⌘	TEI	***Date:***	⌘	12/01/2004
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)* *Rel-6 (Release 6)*

Reason for change:	⌘	It is not clear how to use the Spare Extension in Data Frame of Frame Protocol.
Summary of change:	⌘	- Extension mechanism in Data Frame of Frame Protocol is introduced.
Consequences if not approved:	⌘	In the future, adding new IEs in a backward compatible way is not guaranteed. Impact Analysis: Impact assessment towards the previous version of the specification (same release): This CR has isolated impact on the previous version of the specification because the change only affect on Spare Extension functionality. But since the Spare Extension hasn't been used in this release yet, the impact is very minor. Impact assessment towards the previous release of the specification: Not applicable since this is Rel99 correction.

Clauses affected:	⌘	9a(new)				
		Y	**N**			
Other specs	⌘		X	Other core specifications	⌘	
affected:			X	Test specifications		
			X	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/specs/CR.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.

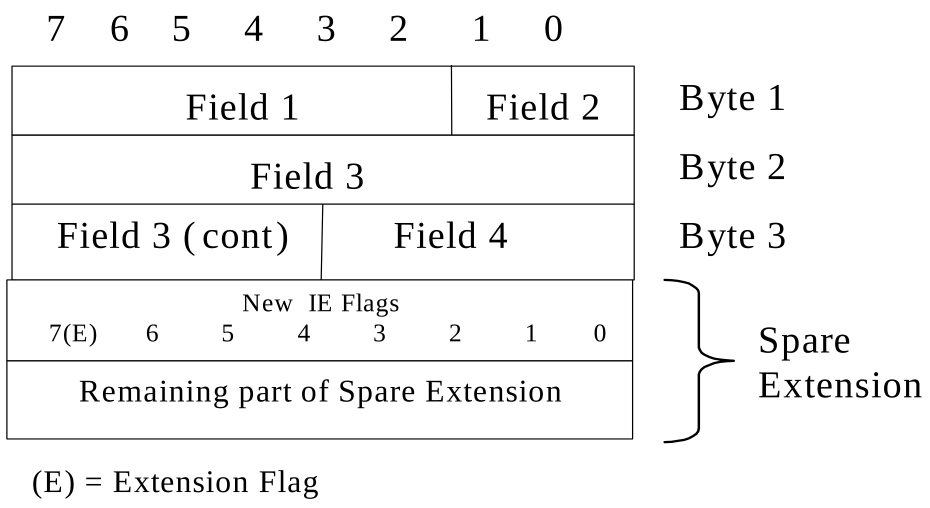
3) With "track changes" disabled, paste the entire CR form (use CTRL-A to select it) into the specification just in front of the clause containing the first piece of changed text. Delete those parts of the specification which are not relevant to the change request.

9a Usage of Iub/Iur Frame Protocol

The following clauses contain guidelines for specification of frame protocols.

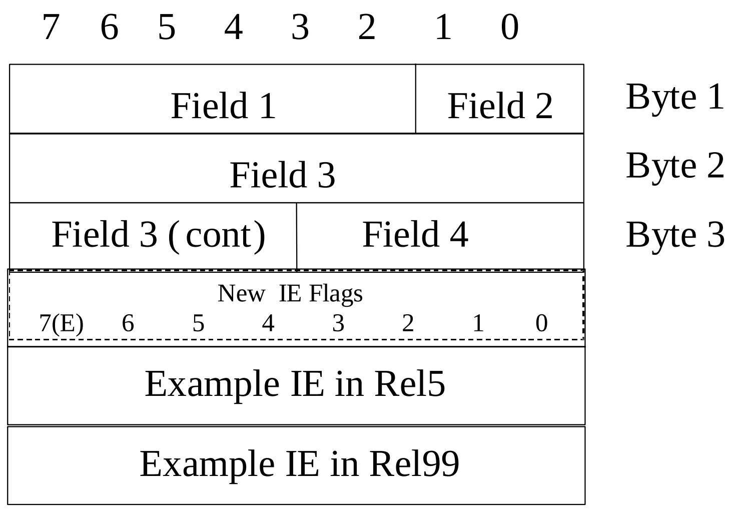
9a.1 Extensions for future releases in Data Frame

Spare Extension is used to define New IEs in the future. When the first IE is added in the Spare Extension in the Data Frame, *New IE Flags* IE shall be added in the first byte of the Spare Extension to indicate the validity of the value of the IEs in the Spare Extension. The last bit position of the *New IE Flags* IE is used as the Extension Flag to allow the extension of the *New IE Flags* IE in the future. The IEs in the Spare Extension will be added in the order in which the IEs are introduced regardless of the release.



In the example below, it is assumed that after *Example IE in Rel5* IE was introduced, *Example IE in Rel99* IE is introduced.

In this example, New IE Flags (0) indicates the validity of *Example IE in Rel5* IE and the New IE Flags (1) indicates the validity of *Example IE in Rel99* IE. The IEs are added in the order of their introduction in the Spare Extension. For the Rel99 and Rel4 nodes, New IE Flags (0) and *Example IE in Rel5* IE will always be seen as Spare Bits while for Rel5 nodes, all the IEs (i.e., New IE Flags (0), New IE Flags(1)), *Example IE in Rel5* IE and *Example IE in Rel99* IE can be used.



Example of Spare Extension Usage

3GPP TSG-RAN2 Meeting #40 ***Tdoc ⌘R2-040189***
Sophia Antipolis, France, 12th – 16th January 2004

CR-Form-v7

CHANGE REQUEST

⌘	**25.921**	**CR**	**051**	⌘ **rev**	**-**	⌘	Current version: **5.3.0**	⌘

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

Proposed change affects:	UICC apps⌘ [ ]	ME [ ]	Radio Access Network [X]	Core Network [ ]

Title:	⌘	Spare Extension in Data Frame		
Source:	⌘	RAN WG2		
Work item code:	⌘	TEI	***Date:*** ⌘	12/01/2004
Category:	⌘	**A**	***Release:*** ⌘	Rel-5
		Use <u>one</u> 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 <u>one</u> 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)* *Rel-6 (Release 6)*

Reason for change:	⌘	It is not clear how to use the Spare Extension in Data Frame of Frame Protocol.
Summary of change:	⌘	- Extension mechanism in Data Frame of Frame Protocol is introduced.
Consequences if not approved:	⌘	In the future, adding new IEs in a backward compatible way is not guaranteed. <u>Impact Analysis:</u> Impact assessment towards the previous version of the specification (same release): This CR has isolated impact on the previous version of the specification because the change only affect on Spare Extension functionality. But since the Spare Extension hasn't been used in this release yet, the impact is very minor. Impact assessment towards the previous release of the specification: Not applicable since this is Rel99 correction.

Clauses affected:	⌘	9a(new)				
		Y	**N**			
Other specs affected:	⌘		X	Other core specifications	⌘	
			X	Test specifications		
			X	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/specs/CR.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.

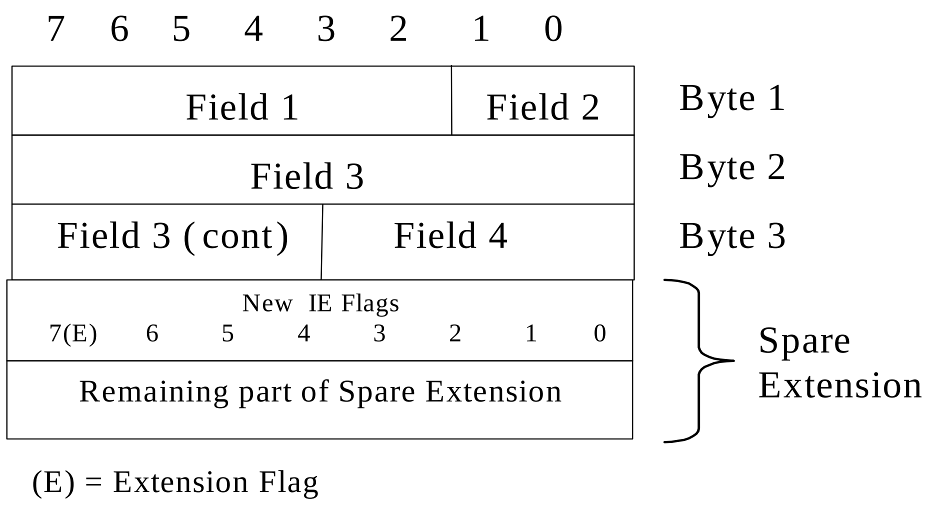
3) With "track changes" disabled, paste the entire CR form (use CTRL-A to select it) into the specification just in front of the clause containing the first piece of changed text. Delete those parts of the specification which are not relevant to the change request.

9a Usage of Iub/Iur Frame Protocol

The following clauses contain guidelines for specification of frame protocols.

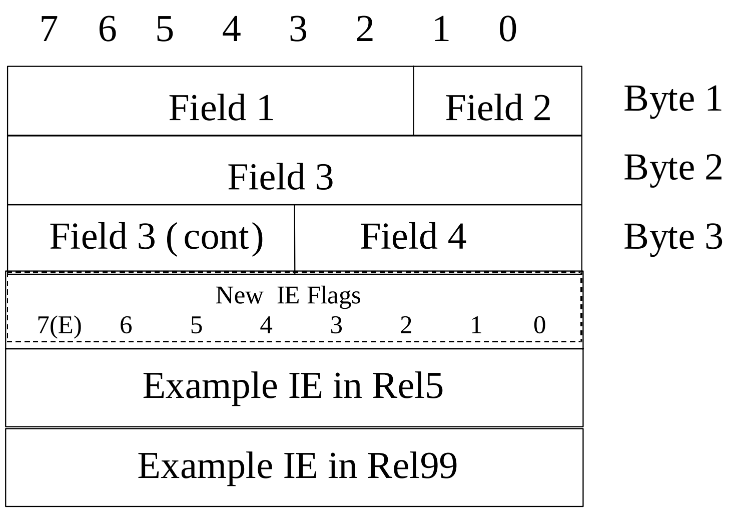
9a.1 Extensions for future releases in Data Frame

Spare Extension is used to define New IEs in the future. When the first IE is added in the Spare Extension in the Data Frame, *New IE Flags* IE shall be added in the first byte of the Spare Extension to indicate the validity of the value of the IEs in the Spare Extension. The last bit position of the *New IE Flags* IE is used as the Extension Flag to allow the extension of the *New IE Flags* IE in the future. The IEs in the Spare Extension will be added in the order in which the IEs are introduced regardless of the release.



In the example below, it is assumed that after *Example IE in Rel5* IE was introduced, *Example IE in Rel99* IE is introduced.

In this example, New IE Flags (0) indicates the validity of *Example IE in Rel5* IE and the New IE Flags (1) indicates the validity of *Example IE in Rel99* IE. The IEs are added in the order of their introduction in the Spare Extension. For the Rel99 and Rel4 nodes, New IE Flags (0) and *Example IE in Rel5* IE will always be seen as Spare Bits while for Rel5 nodes, all the IEs (i.e., New IE Flags (0), New IE Flags(1)), *Example IE in Rel5* IE and *Example IE in Rel99* IE can be used.



Example of Spare Extension Usage

3GPP TSG RAN WG2#40 ***Tdoc ⌘R2-040319***
Sophia Antipolis, France, 6th – 10th September 2003

CR-Form-v7

CHANGE REQUEST

⌘	**25.921**	**CR**	**52**	⌘**rev**	**1**	⌘	Current version:	**3.9.0** ⌘

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

Proposed change affects: UICC apps⌘ ☐ ME ☐ Radio Access Network ☐ Core Network ☐

Title:	⌘	Guideline on release independent ASN.1 updates		
Source:	⌘	RAN WG2		
Work item code:	⌘	TEI	***Date:*** ⌘	14/01/2004
Category:	⌘	**F**	***Release:*** ⌘	R99
		Use <u>one</u> 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 <u>one</u> 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)* *Rel-6 (Release 6)*

Reason for change:	⌘	Guidelines for introducing release independent features within the ASN.1 are currently not provided. The existing practice is to introduce such features in the latest available version of the specification. This may result in decoding problems when extensions for earlier non frozen releases are agreed. As a result of this, it may not be possible to use the release independent feauture until the ASN.1 of the (late) release in which they were introduced is frozen. A guideline is needed to avoid this problem from now on
Summary of change:	⌘	This CR introduces a recommendation to be used when introducing release independent features within the ASN.1. The recommendation is to introduce modifications in the latest release for which the ASN.1 has been frozen. If this approach is unsuitable for other reasons (eg. due to alignment with other specifications/ groups) then the recommendation is to create provisions in latest release for which the ASN.1 has been frozen (by introducing dummy parameters that are encoding compatible with the additional release independent information). In this case, the actual definitions are introduced in the later available release of RRC
Consequences if not approved:	⌘	It may not be possible to use release independent features until the ASN.1 of the (late) release in which they were introduced is frozen

Clauses affected:	⌘	10.4.3.6 (New)				
		Y	**N**			
Other specs	⌘		**X**	Other core specifications	⌘	

affected:	X	Test specifications	
	X	O&M Specifications	
Other comments:	⌘	Note that CR 55 introduces subclause 10.4.3.5	

10.4.3.6 Use of non critical extensions for release independent features

The objective of release independent features is that they do not require the UE to implement a specific release of the specification. Some release independent features may require the introduction of a non critical extension in one or more messages. In this case the recommendation is to introduce the extensions in the latest release of the specification for which the ASN.1 is frozen. This applies both to the tabular and the ASN.1.

In case, for some reasons eg. alignment with other specifications/ groups, the release independent feature is introduced in release X, which is later than the release recommended according to the above, the following recommendations apply:

1. Within the tabular notation, the non critical extension is introduced in release X.

2. Within the ASN.1, the information element is introduced as a non critical extension in release Y, which is the latest release of the specification for which the ASN.1 is frozen.

3. Within the ASN.1 of releases z, with $Y \leq z < X$, the non critical extension is provisioned for by introducing a dummy information element that is encoding compatible with the actual information element introduced in release X

The reason for the provisioning of the extension in the ASN.1 of specifications earlier than release X is that for releases z, with $Y \leq z < X$, it is still possible to modify the ASN.1. As a result, if no provisions would be created in such releases, it would not be possible to use the release independent feature until release X is frozen.

An alternative approach would be to include the non critical extension within the variable length extension container (VLEC). However, considering the overhead associated with the introduction of such a container this approach is not recommended.

Note For release Y it is allowed to introduce non- critical extensions to release Y. Furthermore, backwards incompatible changes are allowed for later releases. If the release independent feature is introduced in release X, which is later than Y, it will be affected by such changes.

3GPP TSG RAN WG2#40 **Tdoc ⌘R2-040344**
Sophia Antipolis, France, 12th – 16th January 2004

CR-Form-v7

CHANGE REQUEST

⌘	**25.921**	**CR**	**53**	⌘**rev**	**2**	⌘	Current version:	**4.6.0** ⌘

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

Proposed change affects: UICC apps⌘ ☐ ME ☐ Radio Access Network ☐ Core Network ☐

Title: ⌘	Guideline on release independent ASN.1 updates			
Source: ⌘	RAN WG2			
Work item code: ⌘	TEI		***Date:*** ⌘	14/01/2004
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)* *Rel-6* *(Release 6)*	

Reason for change: ⌘	Guidelines for introducing release independent features within the ASN.1 are currently not provided. The existing practice is to introduce such features in the latest available version of the specification. This may result in decoding problems when extensions for earlier non frozen releases are agreed. As a result of this, it may not be possible to use the release independent feauture until the ASN.1 of the (late) release in which they were introduced is frozen. A guideline is needed to avoid this problem from now on
Summary of change: ⌘	This CR introduces a recommendation to be used when introducing release independent features within the ASN.1. The recommendation is to introduce modifications in the latest release for which the ASN.1 has been frozen. If this approach is unsuitable for other reasons (eg. due to alignment with other specifications/ groups) then the recommendation is to create provisions in latest release for which the ASN.1 has been frozen (by introducing dummy parameters that are encoding compatible with the additional release independent information). In this case, the actual definitions are introduced in the later available release of RRC
Consequences if not approved: ⌘	It may not be possible to use release independent features until the ASN.1 of the (late) release in which they were introduced is frozen

Clauses affected: ⌘	10.4.3.6 (New)				
	Y	**N**			
Other specs ⌘		**X**	Other core specifications	⌘	

affected:	X	Test specifications	
	X	O&M Specifications	
Other comments:	⌘	Note that CR 56 introduces subclause 10.4.3.5	

10.4.3.6 Use of non critical extensions for release independent features

The objective of release independent features is that they do not require the UE to implement a specific release of the specification. Some release independent features may require the introduction of a non critical extension in one or more messages. In this case the recommendation is to introduce the extensions in the latest release of the specification for which the ASN.1 is frozen. This applies both to the tabular and the ASN.1.

In case, for some reasons eg. alignment with other specifications/ groups, the release independent feature is introduced in release X, which is later than the release recommended according to the above, the following recommendations apply:

1. Within the tabular notation, the non critical extension is introduced in release X.
2. Within the ASN.1, the information element is introduced as a non critical extension in release Y, which is the latest release of the specification for which the ASN.1 is frozen.
3. Within the ASN.1 of releases z, with Y≤ z < X, the non critical extension is provisioned for by introducing a dummy information element that is encoding compatible with the actual information element introduced in release X

The reason for the provisioning of the extension in the ASN.1 of specifications earlier than release X is that for releases z, with Y≤ z < X, it is still possible to modify the ASN.1. As a result, if no provisions would be created in such releases, it would not be possible to use the release independent feature until release X is frozen.

An alternative approach would be to include the non critical extension within the variable length extension container (VLEC). However, considering the overhead associated with the introduction of such a container this approach is not recommended.

Note For release Y it is allowed to introduce non- critical extensions to release Y. Furthermore, backwards incompatible changes are allowed for later releases. If the release independent feature is introduced in release X, which is later than Y, it will be affected by such changes.

3GPP TSG RAN WG2#40
Sophia Antipolis, France, 12th – 16th January 2004

Tdoc ⌘R2-040345

CR-Form-v7

CHANGE REQUEST

⌘	**25.921**	**CR**	**54**	⌘**rev**	**2**	⌘	Current version:	**5.3.0**	⌘

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

Proposed change affects: UICC apps⌘ ☐ ME ☐ Radio Access Network ☐ Core Network ☐

Title:	⌘	Guideline on release independent ASN.1 updates		
Source:	⌘	RAN WG2		
Work item code:	⌘	TEI	***Date:*** ⌘	14/01/2004
Category:	⌘	**A**	***Release:*** ⌘	REL-5

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)*
Rel-6	*(Release 6)*

Reason for change: ⌘ Guidelines for introducing release independent features within the ASN.1 are currently not provided. The existing practice is to introduce such features in the latest available version of the specification. This may result in decoding problems when extensions for earlier non frozen releases are agreed. As a result of this, it may not be possible to use the release independent feauture until the ASN.1 of the (late) release in which they were introduced is frozen. A guideline is needed to avoid this problem from now on

Summary of change: ⌘ This CR introduces a recommendation to be used when introducing release independent features within the ASN.1. The recommendation is to introduce modifications in the latest release for which the ASN.1 has been frozen.
If this approach is unsuitable for other reasons (eg. due to alignment with other specifications/ groups) then the recommendation is to create provisions in latest release for which the ASN.1 has been frozen (by introducing dummy parameters that are encoding compatible with the additional release independent information). In this case, the actual definitions are introduced in the later available release of RRC

Consequences if not approved: ⌘ It may not be possible to use release independent features until the ASN.1 of the (late) release in which they were introduced is frozen

Clauses affected: ⌘ 10.4.3.6 (New)

		Y	**N**		
Other specs	⌘		**X**	Other core specifications	⌘

affected:	X	Test specifications	
	X	O&M Specifications	
Other comments:	⌘	Note that CR 57 introduces subclause 10.4.3.5	

10.4.3.6 Use of non critical extensions for release independent features

The objective of release independent features is that they do not require the UE to implement a specific release of the specification. Some release independent features may require the introduction of a non critical extension in one or more messages. In this case the recommendation is to introduce the extensions in the latest release of the specification for which the ASN.1 is frozen. This applies both to the tabular and the ASN.1.

In case, for some reasons eg. alignment with other specifications/ groups, the release independent feature is introduced in release X, which is later than the release recommended according to the above, the following recommendations apply:

1. Within the tabular notation, the non critical extension is introduced in release X.
2. Within the ASN.1, the information element is introduced as a non critical extension in release Y, which is the latest release of the specification for which the ASN.1 is frozen.
3. Within the ASN.1 of releases z, with $Y \leq z < X$, the non critical extension is provisioned for by introducing a dummy information element that is encoding compatible with the actual information element introduced in release X

The reason for the provisioning of the extension in the ASN.1 of specifications earlier than release X is that for releases z, with $Y \leq z < X$, it is still possible to modify the ASN.1. As a result, if no provisions would be created in such releases, it would not be possible to use the release independent feature until release X is frozen.

An alternative approach would be to include the non critical extension within the variable length extension container (VLEC). However, considering the overhead associated with the introduction of such a container this approach is not recommended.

Note For release Y it is allowed to introduce non- critical extensions to release Y. Furthermore, backwards incompatible changes are allowed for later releases. If the release independent feature is introduced in release X, which is later than Y, it will be affected by such changes.

3GPP TSG RAN WG2#40 | ***Tdoc ⌘R2-040247***

Sophia Antipolis, France, 6th – 10th September 2003

CR-Form-v7

CHANGE REQUEST

⌘	**25.921**	**CR**	**55**	⌘**rev**	**-**	⌘	Current version:	**3.9.0** ⌘

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

Proposed change affects: UICC apps⌘ ☐ ME ☐ Radio Access Network ☐ Core Network ☐

Title:	⌘	Guideline on the use of variable length containers for late extensions		
Source:	⌘	RAN WG2		
Work item code:	⌘	TEI	***Date:*** ⌘	14/01/2004
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)* *Rel-6 (Release 6)*

Reason for change:	⌘	Guidelines for the use of extension containers are currently not provided
Summary of change:	⌘	This CR introduces a recommendation to be used when performing late corrections introducing non critical extensions in the RRC messages
Consequences if not approved:	⌘	Extension containers may be used in a sub- optimal manner, resulting in additional signalling overhead

Clauses affected:	⌘	10.4.3.5				
		Y	**N**			
Other specs affected:	⌘		**X**	Other core specifications	⌘	
			X	Test specifications		
			X	O&M Specifications		
Other comments:	⌘					

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.

10.4.3.3 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.

Before that Backward Compatibility is started for the following Release *N+1*, the non-critical extension information elements of the current Release *N* are added at the end of the message. At the point when Backward Compatibility is started for the following Release *N+1*, an optional BIT STRING container should be added before the information elements of the new release. In the case that further non-critical extension information elements need to be added to Release *N* they shall be placed within the BIT STRING container.

For example: As long as Backward Compatibility is not being enforced for Release 4, Release '99 extensions are added "normally" at the end of a message within a nonCriticalExtensions sequence. Once Backward Compatibility is started for Release 4, then new Release '99 specific extensions are introduced within an extension container. An extension container is a "normal" bit string field that encapsulates an extension structure. As a result:

- New extensions can be added **both** in Release '99 and Release 4 in a backward compatible way; and
- Release 4 systems are able to skip over unknown Release '99 extensions.

The extension container can be viewed as a specific type of non-critical extension and it is included in the same way. If the extension container is added to Release *N* before that Backward Compatibility has started for Release *N+1*, further non-critical extensions to Release *N* should not be included in the container, but should be placed after it, using the usual mechanism. In this way the extension container is not used until necessary, and therefore the corresponding length field overhead is not incurred unnecessarily. Additional guidelines concerning the use of extension containers are provided in 10.4.3.5.

The structure of the message of the example above is shown in Example 3 for Release '99 and 4 messages.

Examples for special non-critical extensions and MessageA-v440ext-IEs are given in the following subclauses.

```
-- This shows the message structure in Release '99 (including one non-critical extension)
-- before backward compatibility is started for Release 4.
MessageA ::=               CHOICE {
    r3                             SEQUENCE {
        messageA-r3                    MessageA-r3-IEs,
        v380nonCriticalExtensions          SEQUENCE {
            messageA-v380ext                   MessageA-v380ext-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-v380ext-IEs :: =          SEQUENCE {
    -- Here are information elements added to Release '99 as extensions to the information
    -- contained in MessageA-r3-IEs.
}



-- This shows the Release '99 message structure once backward compatibility
-- has been started for Release 4.
MessageA ::=               CHOICE {
    r3                             SEQUENCE {
        messageA-r3                    MessageA-r3-IEs,
        v380nonCriticalExtensions          SEQUENCE {
            messageA-v380ext                   MessageA-v380ext-IEs,
            laterNonCriticalExtensions         SEQUENCE {
                -- Container for additional Release '99 extensions
                messageA-r3-add-ext                BIT STRING
                    (CONTAINING MessageA-r3-add-ext-IEs)            OPTIONAL,
                nonCriticalExtensions          SEQUENCE {} OPTIONAL
            }   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-v380ext-IEs :: =           SEQUENCE {
    -- Here are information elements added to Release '99 as extensions to the information
    -- contained in MessageA-r3-IEs.
}

MessageA-r3-add-ext-IEs :: =        SEQUENCE {
    -- Here are information elements added to Release '99 as extensions to the information
    -- contained in MessageA-r3-IEs after backward compatibility was started for Release 4.
}



-- This shows the structure of the Release 4 message
-- (including one Release 4 non-critical extension).
MessageA ::=                CHOICE {
    r3                              SEQUENCE {
        messageA-r3                     MessageA-r3-IEs,
        v380nonCriticalExtensions           SEQUENCE {
            messageA-v380ext                    MessageA-v380ext-IEs,
            laterNonCriticalExtensions          SEQUENCE {
                -- Container for additional Release '99 extensions
                messageA-r3-add-ext                 BIT STRING
                    (CONTAINING MessageA-r3-add-ext-IEs)            OPTIONAL,
                v440nonCriticalExtensions           SEQUENCE {
                    messageA-v440ext                    MessageA-v440ext-IEs,
                    nonCriticalExtensions           SEQUENCE {} OPTIONAL
                }   OPTIONAL
            }   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-v380ext-IEs :: =           SEQUENCE {
    -- Here are information elements added to Release '99 as extensions to the information
    -- contained in MessageA-r3-IEs.
}

MessageA-r3-add-ext-IEs :: =        SEQUENCE {
    -- Here are information elements added to Release '99 as extensions to the information
    -- contained in MessageA-r3-IEs after backward compatibility was started for Release 4.
}

MessageA-v440ext-IEs ::=            SEQUENCE {
    -- Here are information elements added to Release 4 as extensions to the information
    -- contained in MessageA-r3-IEs and MessageA-v380ext-IEs.
}
```

Example 3

10.4.3.5 Additional guidelines on the use of variable length extension containers

"Variable length extension containers" (i.e. non critical extension containers that have their abstract syntax defined using the ASN.1 type "BIT STRING") have been defined to support the introduction of extensions to a release after the subsequent release is frozen (and UEs based on that subsequent release may appear).

Extension containers should be introduced in each message unless the size of the message is critical and the likelihood of late corrections is low. For downlink messages for which different versions have been defined, an extensions container should be introduced for each message version (branch).

In case a variable length extension container (VLEC) includes an extension, the PER encoder will include an additional length determinant. In case a separate container is introduced for each release, this would result in a significant signalling overhead. In order to avoid this signalling overhead, the extensions container should not be dedicated to late corrections of one specific release. If the extensions container is required to support the introduction of late corrections in an order of release, one or more release specific extensions container(s) may be nested within the original extension container.

The above guidelines are illustrated by means of an example. Suppose a message includes a single VLEC and one late R99 correction has been defined using this extension container, When the need for a late REL-5 correction arises, this correction may be added to the extensions container in two different ways:

1. Just by adding the extension after the late R99 correction

2. By introducing a REL-5 extensions container, that is nested within the existing extensions container

In case the first option is used, the addition of another late R99 correction (after the REL-5 correction) would require the R99 receiver to comprehend the transfer syntax of the REL-5 extension. This would require the introduction of the late REL-5 correction (or a type with an equivalent encoding) in the R99 transfer syntax. The second option avoids this problem, although this comes at the cost of additional signalling overhead.

Which option to use should be decided when introducing an extension for a release later than R99. This can be decided on a case by case basis eg. depending on whether for the concerned message further late R99 corrections need to be accommodated.

3GPP TSG RAN WG2#40 ***Tdoc ⌘R2-040248***
Sophia Antipolis, France, 6th – 10th September 2003

CR-Form-v7

CHANGE REQUEST

⌘	**25.921**	**CR**	**56**	⌘**rev**	**-**	⌘ Current version:	**4.6.0** ⌘

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

Proposed change affects: UICC apps⌘ ☐ ME ☐ Radio Access Network ☐ Core Network ☐

Title:	⌘	Guideline on the use of variable length containers for late extensions	
Source:	⌘	RAN WG2	
Work item code:	⌘	TEI	***Date:*** ⌘ 14/01/2004
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)* *Rel-6 (Release 6)*

Reason for change:	⌘	Guidelines for the use of extension containers are currently not provided
Summary of change:	⌘	This CR introduces a recommendation to be used when performing late corrections introducing non critical extensions in the RRC messages
Consequences if not approved:	⌘	Extension containers may be used in a sub- optimal manner, resulting in additional signalling overhead

Clauses affected:	⌘	10.4.3.3, 10.4.3.5 (new)				
		Y	**N**			
Other specs affected:	⌘		**X**	Other core specifications	⌘	
			X	Test specifications		
			X	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/specs/CR.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.

10.4.3.3 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.

Before that Backward Compatibility is started for the following Release *N+1*, the non-critical extension information elements of the current Release *N* are added at the end of the message. At the point when Backward Compatibility is started for the following Release *N+1*, an optional BIT STRING container should be added before the information elements of the new release. In the case that further non-critical extension information elements need to be added to Release *N* they shall be placed within the BIT STRING container.

For example: As long as Backward Compatibility is not being enforced for Release 4, Release '99 extensions are added "normally" at the end of a message within a nonCriticalExtensions sequence. Once Backward Compatibility is started for Release 4, then new Release '99 specific extensions are introduced within an extension container. An extension container is a "normal" bit string field that encapsulates an extension structure. As a result:

- New extensions can be added **both** in Release '99 and Release 4 in a backward compatible way; and
- Release 4 systems are able to skip over unknown Release '99 extensions.

The extension container can be viewed as a specific type of non-critical extension and it is included in the same way. If the extension container is added to Release *N* before that Backward Compatibility has started for release *N+1*, further non-critical extensions to Release *N* should not be included in the container, but should be placed after it using the usual mechanism. In this way the extension container is not used until necessary, and therefore the corresponding length field overhead is not incurred unnecessarily. Additional guidelines concerning the use of extension containers are provided in 10.4.3.5.

The structure of the message of the example above is shown in Example 3 for Release '99 and 4 messages.

Examples for special non-critical extensions and MessageA-v440ext-IEs are given in the following subclauses.

```
-- This shows the message structure in Release '99 (including one non-critical extension)
-- before backward compatibility is started for Release 4.
MessageA ::=                CHOICE {
    r3                              SEQUENCE {
        messageA-r3                     MessageA-r3-IEs,
        v380nonCriticalExtensions           SEQUENCE {
            messageA-v380ext                    MessageA-v380ext-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-v380ext-IEs :: =           SEQUENCE {
    -- Here are information elements added to Release '99 as extensions to the information
    -- contained in MessageA-r3-IEs.
}



-- This shows the Release '99 message structure once backward compatibility
-- has been started for Release 4.
MessageA ::=                CHOICE {
    r3                              SEQUENCE {
        messageA-r3                     MessageA-r3-IEs,
        v380nonCriticalExtensions           SEQUENCE {
            messageA-v380ext                    MessageA-v380ext-IEs,
            laterNonCriticalExtensions          SEQUENCE {
                -- Container for additional Release '99 extensions
                messageA-r3-add-ext                 BIT STRING
                    (CONTAINING MessageA-r3-add-ext-IEs)            OPTIONAL,
                nonCriticalExtensions           SEQUENCE {} OPTIONAL
            }   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-v380ext-IEs :: =           SEQUENCE {
    -- Here are information elements added to Release '99 as extensions to the information
    -- contained in MessageA-r3-IEs.
}

MessageA-r3-add-ext-IEs :: =        SEQUENCE {
    -- Here are information elements added to Release '99 as extensions to the information
    -- contained in MessageA-r3-IEs after backward compatibility was started for Release 4.
}



-- This shows the structure of the Release 4 message
-- (including one Release 4 non-critical extension).
MessageA ::=                CHOICE {
    r3                              SEQUENCE {
        messageA-r3                     MessageA-r3-IEs,
        v380nonCriticalExtensions           SEQUENCE {
            messageA-v380ext                    MessageA-v380ext-IEs,
            laterNonCriticalExtensions          SEQUENCE {
                -- Container for additional Release '99 extensions
                messageA-r3-add-ext                 BIT STRING
                    (CONTAINING MessageA-r3-add-ext-IEs)            OPTIONAL,
                v440nonCriticalExtensions           SEQUENCE {
                    messageA-v440ext                    MessageA-v440ext-IEs,
                    nonCriticalExtensions           SEQUENCE {} OPTIONAL
                }   OPTIONAL
            }   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-v380ext-IEs :: =           SEQUENCE {
    -- Here are information elements added to Release '99 as extensions to the information
    -- contained in MessageA-r3-IEs.
}

MessageA-r3-add-ext-IEs :: =        SEQUENCE {
    -- Here are information elements added to Release '99 as extensions to the information
    -- contained in MessageA-r3-IEs after backward compatibility was started for Release 4.
}

MessageA-v440ext-IEs ::=            SEQUENCE {
    -- Here are information elements added to Release 4 as extensions to the information
    -- contained in MessageA-r3-IEs and MessageA-v380ext-IEs.
}
```

Example 3

10.4.3.5 Additional guidelines on the use of variable length extension containers

"Variable length extension containers" (i.e. non critical extension containers that have their abstract syntax defined using the ASN.1 type "BIT STRING") have been defined to support the introduction of extensions to a release after the subsequent release is frozen (and UEs based on that subsequent release may appear).

Extension containers should be introduced in each message unless the size of the message is critical and the likelihood of late corrections is low. For downlink messages for which different versions have been defined, an extensions container should be introduced for each message version (branch).

In case a variable length extension container (VLEC) includes an extension, the PER encoder will include an additional length determinant. In case a separate container is introduced for each release, this would result in a significant signalling overhead. In order to avoid this signalling overhead, the extensions container should not be dedicated to late corrections of one specific release. If the extensions container is required to support the introduction of late corrections in an order of release, one or more release specific extensions container(s) may be nested within the original extension container.

The above guidelines are illustrated by means of an example. Suppose a message includes a single VLEC and one late R99 correction has been defined using this extension container, When the need for a late REL-5 correction arises, this correction may be added to the extensions container in two different ways:

1. Just by adding the extension after the late R99 correction

2. By introducing a REL-5 extensions container, that is nested within the existing extensions container

In case the first option is used, the addition of another late R99 correction (after the REL-5 correction) would require the R99 receiver to comprehend the transfer syntax of the REL-5 extension. This would require the introduction of the late REL-5 correction (or a type with an equivalent encoding) in the R99 transfer syntax. The second option avoids this problem, although this comes at the cost of additional signalling overhead.

Which option to use should be decided when introducing an extension for a release later than R99. This can be decided on a case by case basis eg. depending on whether for the concerned message further late R99 corrections need to be accommodated.

3GPP TSG RAN WG2#40 — ***Tdoc ⌘R2-040249***
Sophia Antipolis, France, 6th – 10th September 2003

CR-Form-v7

CHANGE REQUEST

⌘	**25.921**	**CR**	**57**	⌘**rev**	**-**	⌘	Current version:	**5.3.0**	⌘

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

Proposed change affects: UICC apps⌘ ☐ ME ☐ Radio Access Network ☐ Core Network ☐

Title:	⌘	Guideline on the use of variable length containers for late extensions		
Source:	⌘	RAN WG2		
Work item code:	⌘	TEI	***Date:*** ⌘	14/01/2004
Category:	⌘	**A**	***Release:*** ⌘	REL-5
		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)* *Rel-6 (Release 6)*

Reason for change:	⌘	Guidelines for the use of extension containers are currently not provided
Summary of change:	⌘	This CR introduces a recommendation to be used when performing late corrections introducing non critical extensions in the RRC messages
Consequences if not approved:	⌘	Extension containers may be used in a sub- optimal manner, resulting in additional signalling overhead

Clauses affected:	⌘	10.4.3.3, 10.4.3.5 (new)				
		Y	**N**			
Other specs affected:	⌘		**X**	Other core specifications	⌘	
			X	Test specifications		
			X	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/specs/CR.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.

10.4.3.3 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.

Before that Backward Compatibility is started for the following Release *N+1*, the non-critical extension information elements of the current Release *N* are added at the end of the message. At the point when Backward Compatibility is started for the following Release *N+1*, an optional BIT STRING container should be added before the information elements of the new release. In the case that further non-critical extension information elements need to be added to Release N they shall be placed within the BIT STRING container.

For example: As long as Backward Compatibility is not being enforced for Release 4, Release '99 extensions are added "normally" at the end of a message within a nonCriticalExtensions sequence. Once Backward Compatibility is started for Release 4, then new Release '99 specific extensions are introduced within an extension container. An extension container is a "normal" bit string field that encapsulates an extension structure. As a result:

- New extensions can be added **both** in Release '99 and Release 4 in a backward compatible way; and

- Release 4 systems are able to skip over unknown Release '99 extensions.

The extension container can be viewed as a specific type of non-critical extension and it is included in the same way. If the extension container is added to Release *N* before that Backward Compatibility has started for Release *N+1*, further non-critical extensions to Release *N* should not be included in the container, but should be placed after it using the usual mechanism. In this way the extension container is not used until necessary, and therefore the corresponding length field overhead is not incurred unnecessarily. Additional guidelines concerning the use of extension containers are provided in 10.4.3.5.

The structure of the message of the example above is shown in Example 3 for Release '99 and 4 messages.

Examples for special non-critical extensions and MessageA-v440ext-IEs are given in the following subclauses.

```
-- This shows the message structure in Release '99 (including one non-critical extension)
-- before backwards compatibility is started for Release 4.
MessageA ::=                CHOICE {
    r3                              SEQUENCE {
        messageA-r3                         MessageA-r3-IEs,
        v380nonCriticalExtensions               SEQUENCE {
            messageA-v380ext                        MessageA-v380ext-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-v380ext-IEs :: =           SEQUENCE {
    -- Here are information elements added to Release '99 as extensions to the information
    -- contained in MessageA-r3-IEs.
}



-- This shows the Release '99 message structure once backward campatibility
-- has been started for Release 4.
MessageA ::=                CHOICE {
    r3                              SEQUENCE {
        messageA-r3                         MessageA-r3-IEs,
        v380nonCriticalExtensions               SEQUENCE {
            messageA-v380ext                        MessageA-v380ext-IEs,
            laterNonCriticalExtensions              SEQUENCE {
                -- Container for additional Release '99 extensions
                messageA-r3-add-ext                     BIT STRING
                    (CONTAINING MessageA-r3-add-ext-IEs)            OPTIONAL,
                nonCriticalExtensions               SEQUENCE {} OPTIONAL
            }   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-v380ext-IEs :: =           SEQUENCE {
    -- Here are information elements added to Release '99 as extensions to the information
    -- contained in MessageA-r3-IEs.
}

MessageA-r3-add-ext-IEs :: =        SEQUENCE {
    -- Here are information elements added to Release '99 as extensions to the information
    -- contained in MessageA-r3-IEs after backwards compatibility was started for Release 4.
}



-- This shows the structure of the Release 4 message
-- (including one Release 4 non-critical extension).
MessageA ::=                CHOICE {
    r3                              SEQUENCE {
        messageA-r3                         MessageA-r3-IEs,
        v380nonCriticalExtensions               SEQUENCE {
            messageA-v380ext                        MessageA-v380ext-IEs,
            laterNonCriticalExtensions          SEQUENCE {
                -- Container for additional Release '99 extensions
                messageA-r3-add-ext                 BIT STRING
                    (CONTAINING MessageA-r3-add-ext-IEs)            OPTIONAL,
                v440nonCriticalExtensions           SEQUENCE {
                    messageA-v440ext                    MessageA-v440ext-IEs,
                    nonCriticalExtensions           SEQUENCE {} OPTIONAL
                }   OPTIONAL
            }   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-v380ext-IEs :: =           SEQUENCE {
    -- Here are information elements added to Release '99 as extensions to the information
    -- contained in MessageA-r3-IEs.
}

MessageA-r3-add-ext-IEs :: =        SEQUENCE {
    -- Here are information elements added to Release '99 as extensions to the information
    -- contained in MessageA-r3-IEs after backwards compatibility was started for Release 4.
}

MessageA-v440ext-IEs ::=            SEQUENCE {
    -- Here are information elements added to Release 4 as extensions to the information
    -- contained in MessageA-r3-IEs and MessageA-v380ext-IEs.
}
```

Example 3

10.4.3.5 Additional guidelines on the use of variable length extension containers

"Variable length extension containers" (i.e. non critical extension containers that have their abstract syntax defined using the ASN.1 type "BIT STRING") have been defined to support the introduction of extensions to a release after the subsequent release is frozen (and UEs based on that subsequent release may appear).

Extension containers should be introduced in each message unless the size of the message is critical and the likelihood of late corrections is low. For downlink messages for which different versions have been defined, an extensions container should be introduced for each message version (branch).

In case a variable length extension container (VLEC) includes an extension, the PER encoder will include an additional length determinant. In case a separate container is introduced for each release, this would result in a significant signalling overhead. In order to avoid this signalling overhead, the extensions container should not be dedicated to late corrections of one specific release. If the extensions container is required to support the introduction of late corrections in an order of release, one or more release specific extensions container(s) may be nested within the original extension container.

The above guidelines are illustrated by means of an example. Suppose a message includes a single VLEC and one late R99 correction has been defined using this extension container, When the need for a late REL-5 correction arises, this correction may be added to the extensions container in two different ways:

1. Just by adding the extension after the late R99 correction

2. By introducing a REL-5 extensions container, that is nested within the existing extensions container

In case the first option is used, the addition of another late R99 correction (after the REL-5 correction) would require the R99 receiver to comprehend the transfer syntax of the REL-5 extension. This would require the introduction of the late REL-5 correction (or a type with an equivalent encoding) in the R99 transfer syntax. The second option avoids this problem, although this comes at the cost of additional signalling overhead.

Which option to use should be decided when introducing an extension for a release later than R99. This can be decided on a case by case basis eg. depending on whether for the concerned message further late R99 corrections need to be accommodated.

3GPP TSG RAN WG2#40 ***Tdoc ⌘R2-040300***
Sophia Antipolis, France, 6th – 10th September 2003

CR-Form-v7

CHANGE REQUEST

⌘	**25.921**	**CR**	**58**	⌘**rev**	**-**	⌘	Current version:	**3.9.0** ⌘

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

Proposed change affects: UICC apps⌘ ☐ ME ☐ Radio Access Network ☐ Core Network ☐

Title:	⌘ Guideline for the naming of extensions to the RRC ASN.1		
Source:	⌘ RAN WG2		
Work item code:	⌘ TEI	***Date:*** ⌘	15/01/2004
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)* *Rel-6 (Release 6)*	

Reason for change:	⌘	Currently the REL-4 and REL-5 extensions are not named in a consistent manner. This is because the guidelines for the naming of extensions to the RRC ASN.1 are currently neither clear nor complete
Summary of change:	⌘	The is CR introduces a recommendation to be used when performing late corrections introducing non critical extensions in the RRC messages
Consequences if not approved:	⌘	There will be no way to achieve consistent naming of extensions to the RRC ASN.1

Clauses affected:	⌘	10.4.2			
		Y	**N**		
Other specs affected:	⌘		**X**	Other core specifications	⌘
			X	Test specifications	
			X	O&M Specifications	
Other comments:	⌘	The new convention means that some IEs introduced in a later release/ version need not apply a specific suffix. This means that it will be somewhat more difficult to track in which release/ version the IE was introduced. This requires some additional care when applying the backwards compatibility rules. However, since it has already been common practice to not always use an extension showing the release/ version suffix, this is not considered to be a problem			

How to create CRs using this form:

Comprehensive information and tips about how to create CRs can be found at http://www.3gpp.org/specs/CR.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.

For critical 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 this should be the same as for the new message root. For this naming, "`r3`" is used for Release '99, "`r4`" for Release 4, "`r5`" for Release 5 and so on.

For non-critical extensions the new elements introduced to specify the extensions should be grouped together in an element with a name showing the version of the specification where this extension will first be included, e.g. if the version of the specification being corrected is v3.7.0, then the suffix added to the name will be -v380ext (i.e. the next version).

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.

An example is given below to illustrate these principles, on the message named "`Test-msg`".

```
-- In Release '99, the Test-msg is defined as following:
Test-msg ::= CHOICE {
    r3                              SEQUENCE {
        test-msg-r3                     Test-msg-r3-IEs,
        nonCriticalExtensions           SEQUENCE {} OPTIONAL
    },
    later-than-r3                   SEQUENCE {
        rrc-TransactionIdentifier       RRC-TransactionIdentifier,
        criticalExtensions              SEQUENCE {}
    }
}
-- A later correction to Release '99 adds a non-critical extension in v3.8.0
-- of the specification
Test-msg ::= CHOICE {
    r3                              SEQUENCE {
        test-msg-r3                     Test-msg-r3-IEs,
        v380nonCriticalExtensions           SEQUENCE {
            test-msg-v380ext                    Test-msg-v380ext-IEs,
            nonCriticalExtensions               SEQUENCE {} OPTIONAL
        } OPTIONAL
    },
    later-than-r3                   SEQUENCE {
        rrc-TransactionIdentifier       RRC-TransactionIdentifier,
        criticalExtensions              SEQUENCE {}
    }
}
-- The Test-msg gets the following structure, if only a non-critical
-- extensions is introduced for Release 4 in v4.4.0 of the specification.
Test-msg ::= CHOICE {
    r3                              SEQUENCE {
        test-msg-r3                     Test-msg-r3-IEs,
        v380nonCriticalExtensions           SEQUENCE {
            test-msg-v380ext                    Test-msg-v380ext-IEs,
            laterNonCriticalExtensions          SEQUENCE {
                -- Container for additional Release '99 extensions
                test-msg-r3-add-ext                 BIT STRING
                    (CONTAINING Test-msg-r3-add-ext-IEs)            OPTIONAL,
                v440nonCriticalExtensions           SEQUENCE {
                    test-msg-v440ext                    Test-msg-v440ext-IEs,
                    nonCriticalExtensions               SEQUENCE {} OPTIONAL
                }   OPTIONAL
            }   OPTIONAL
        } OPTIONAL
    },
    later-than-r3                   SEQUENCE {
        rrc-TransactionIdentifier       RRC-TransactionIdentifier,
        criticalExtensions              SEQUENCE {}
    }
}

-- In Release 5, the Test msg gets the following structure when a critical
-- extension is added
Test-msg ::= CHOICE {
    r3                              SEQUENCE {
        test-msg-r3                     Test-msg-r3-IEs,
        v380nonCriticalExtensions           SEQUENCE {
            test-msg-v380ext                    Test-msg-v380ext-IEs,
            laterNonCriticalExtensions          SEQUENCE {
                -- Container for additional Release '99 extensions
                test-msg-r3-add-ext                 BIT STRING
                    (CONTAINING Test-msg-r3-add-ext-IEs)            OPTIONAL,
                v440nonCriticalExtensions           SEQUENCE {
                    test-msg-v440ext                    Test-msg-v440ext-IEs,
                    nonCriticalExtensions               SEQUENCE {} OPTIONAL
                }   OPTIONAL
            }   OPTIONAL
        } OPTIONAL
    },
    later-than-r3                   SEQUENCE {
        rrc-TransactionIdentifier       RRC-TransactionIdentifier,
        criticalExtensions              CHOICE {
            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).

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. In case the type that is introduced in Release *N* includes one or more new (nested) types, the additional suffix need not be used for these nested types. In case the type that is introduced in Release *N* includes one or more revisions of exixting types, the suffix is needed to distinguish them from the earlier revisions. In case a revision of an abstract type that is introduced in Release *N* includes an IE for which the abstract type already existed in earlier releases, while that IE was not present in the previous revision(s) of the revised abstract type, the IE name should have a suffix "`-r`*N*".

If an abstract type is introduced in a release to extend an already existing type "`TypeX`", it should get the same name with a non-critical extension type suffix ("`-vXYZext`", e.g. "`TypeX-v380ext`") although in this case the final "–IEs" suffix is not added. In case the type that is introduced in Release *N* to extend an already existing type includes one or more new (nested) types that are extensions of an already existing type, the additional suffix should not be used for these nested types. In case the type that is introduced in Release *N* to extend an already existing type includes one or more new (nested) types, the abovely specified rules for new abstract types apply.

The above naming conventions are further illustrated in the example below:

```
Test-msg-v380ext-IEs ::= SEQUENCE {
        existingIE-A-v380ext            ExistingIE-A-v380ext        OPTIONAL,
        newIE-B                         NewIE-B
}
```

```
Test-msg-v440ext-IEs::= SEQUENCE {
        newIE-C-r4                      NewIE-C                     OPTIONAL,
        existingIE-D-v440ext            ExistingIE-D-v440ext
}

Test-msg-r5-IEs::= SEQUENCE {
        existingIE-E                    ExistingIE-E
        newUseOfexistingIE-E-rF         ExistingIE-F                OPTIONAL,
        newIE-G-r5                      NewIE-G
        revisionOfExistingIE-H-r5       ExistingIE-H-r5
}
```

The abovely described naming convention means that some IEs introduced in a later release/ version need not apply a specific suffix. This means that it will not allways be clear from the name of an IE whether or not backwards incompatible changes to it are allowed. ~~Using the above naming rules, when changes are done in Release *N*, only changes in types with a suffix "-r*N*" or "-vXYZext" are allowed, in order to avoid conflicts with previous releases. An exception is ~~T~~t~~he Message type ~~itself~~is a special case, 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,
        v380nonCriticalExtensions               SEQUENCE {
            test-msg-v380ext                    Test-msg-v380ext-IEs,
            nonCriticalExtensions               SEQUENCE {} OPTIONAL
        } OPTIONAL
    },
    later-than-r3                       SEQUENCE {
        rrc-TransactionIdentifier           RRC-TransactionIdentifier,
        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 1: The structure presented in this clause and the proposed naming rules are one possibility. Further possibilities are FFS.

NOTE 2: When non-critical extensions are introduced in a message that does not have yet a criticalExtension branch, they are introduced in the "Test-msg-v380ext-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.

- To be 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 ::=                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 ::= CHOICE {
    r3                              SEQUENCE {
        messageA-r3                     MessageA-r3-IEs,
        nonCriticalExtensions           SEQUENCE {} OPTIONAL
    },
    later-than-r3                   SEQUENCE {
        rrc-TransactionIdentifier       RRC-TransactionIdentifier,
        criticalExtensions              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.

Before that Backward Compatibility is started for the following Release *N+1*, the non-critical extension information elements of the current Release *N* are added at the end of the message. At the point when Backward Compatibility is started for the following Release *N+1*, an optional BIT STRING container should be added before the information elements of the new release. In the case that further non-critical extension information elements need to be added to Release *N* they shall be placed within the BIT STRING container.

For example: As long as Backward Compatibility is not being enforced for Release 4, Release '99 extensions are added "normally" at the end of a message within a nonCriticalExtensions sequence. Once Backward Compatibility is started for Release 4, then new Release '99 specific extensions are introduced within an extension container. An extension container is a "normal" bit string field that encapsulates an extension structure. As a result:

- New extensions can be added **both** in Release '99 and Release 4 in a backward compatible way; and
- Release 4 systems are able to skip over unknown Release '99 extensions.

The extension container can be viewed as a specific type of non-critical extension and it is included in the same way. If the extension container is added to Release *N* before that Backward Compatibility has started for Release *N+1*, further non-critical extensions to Release *N* should not be included in the container, but should be placed after it, using the usual mechanism. In this way the extension container is not used until necessary, and therefore the corresponding length field overhead is not incurred unnecessarily.

The structure of the message of the example above is shown in Example 3 for Release '99 and 4 messages.

Examples for special non-critical extensions and MessageA-v440ext-IEs are given in the following subclauses.

```
-- This shows the message structure in Release '99 (including one non-critical extension)
-- before backward compatibility is started for Release 4.
MessageA ::=                CHOICE {
    r3                              SEQUENCE {
        messageA-r3                     MessageA-r3-IEs,
        v380nonCriticalExtensions           SEQUENCE {
            messageA-v380ext                    MessageA-v380ext-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-v380ext-IEs :: =           SEQUENCE {
    -- Here are information elements added to Release '99 as extensions to the information
    -- contained in MessageA-r3-IEs.
}



-- This shows the Release '99 message structure once backward compatibility
-- has been started for Release 4.
MessageA ::=                CHOICE {
    r3                              SEQUENCE {
        messageA-r3                     MessageA-r3-IEs,
        v380nonCriticalExtensions           SEQUENCE {
            messageA-v380ext                    MessageA-v380ext-IEs,
            laterNonCriticalExtensions          SEQUENCE {
                -- Container for additional Release '99 extensions
                messageA-r3-add-ext                 BIT STRING
```

```
                    (CONTAINING MessageA-r3-add-ext-IEs)            OPTIONAL,
                nonCriticalExtensions           SEQUENCE {} OPTIONAL
            }   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-v380ext-IEs :: =           SEQUENCE {
    -- Here are information elements added to Release '99 as extensions to the information
    -- contained in MessageA-r3-IEs.
}

MessageA-r3-add-ext-IEs :: =        SEQUENCE {
    -- Here are information elements added to Release '99 as extensions to the information
    -- contained in MessageA-r3-IEs after backward compatibility was started for Release 4.
}



-- This shows the structure of the Release 4 message
-- (including one Release 4 non-critical extension).
MessageA ::=                CHOICE {
    r3                              SEQUENCE {
        messageA-r3                     MessageA-r3-IEs,
        v380nonCriticalExtensions           SEQUENCE {
            messageA-v380ext                    MessageA-v380ext-IEs,
            laterNonCriticalExtensions          SEQUENCE {
                -- Container for additional Release '99 extensions
                messageA-r3-add-ext                 BIT STRING
                    (CONTAINING MessageA-r3-add-ext-IEs)            OPTIONAL,
                v440nonCriticalExtensions           SEQUENCE {
                    messageA-v440ext                    MessageA-v440ext-IEs,
                    nonCriticalExtensions           SEQUENCE {} OPTIONAL
                }   OPTIONAL
            }   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-v380ext-IEs :: =           SEQUENCE {
    -- Here are information elements added to Release '99 as extensions to the information
    -- contained in MessageA-r3-IEs.
}

MessageA-r3-add-ext-IEs :: =        SEQUENCE {
    -- Here are information elements added to Release '99 as extensions to the information
    -- contained in MessageA-r3-IEs after backward compatibility was started for Release 4.
}

MessageA-v440ext-IEs ::=            SEQUENCE {
    -- Here are information elements added to Release 4 as extensions to the information
    -- contained in MessageA-r3-IEs and MessageA-v380ext-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-v440ext-IEs.

Example4 shows how the MessageA is extended to include a new element, "element3".

```
MessageA-r3-IEs ::=                 SEQUENCE {
    element1                            Element1,
    element2                            Element2
}

MessageA-v440ext-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-v440ext-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-v440ext-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-v440ext-IEs ("element1a-3").
    choice1                             CHOICE {
        choice1a                            SEQUENCE {
            element1a-1                         Element1a-1
        },
        choice1b                            SEQUENCE {
            element1a-2                         Element1a-2
        }
    }
}

MessageA-v440ext-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-v440ext-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-v440ext-IEs" the branch
-- "choice1b" of the CHOICE "choice1" is used.
    element1                            Element1,
    element2                            Element2
}

MessageA-v440ext-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-v440ext-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-v440ext-IEs is present, and the
-- value of that element used instead.
    element1                            INTEGER (0..7)
    element2                            Element2
}

MessageA-v440ext-IEs ::=                SEQUENCE {
    element1                            INTEGER (0..15)         OPTIONAL
}
```

Example 7

10.4.3.4.5 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.

```
-- 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 appropriate type in the new version, the name of the element replaces the spare name in the CHOICE, but the type cannot 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.

```
-- In the previous version, MessageA-r3-IEs was defined:
MessageA-r3-IEs ::=                 SEQUENCE {
    element1                            CHOICE {
        e1                                  E1,
        e2                                  E2,
        spare                               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
    }
}

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

10.4.3.4.6 Introducing new System Information Block Types

In general new message types are introduced by replacing a spare value as described in subclause 10.4.3.4.5. That subclause also shows that in case there are insufficient spare values available, the last spare value can be replaced by a special extension value. If that value is used, an additional message type extension IE is included to distinguish between the additional message types, as shown in Example 10.

```
DL-CCCH-Message ::= SEQUENCE {
    integrityCheckInfo      IntegrityCheckInfo      OPTIONAL,
    message                 DL-CCCH-MessageType
}

DL-CCCH-MessageType ::= CHOICE {
    cellUpdateConfirm                   CellUpdateConfirm-CCCH,
    rrcConnectionReject                 RRCConnectionReject,
    rrcConnectionRelease                RRCConnectionRelease-CCCH,
    rrcConnectionSetup                  RRCConnectionSetup,
    uraUpdateConfirm                    URAUpdateConfirm-CCCH,
    ext1                                Ext1Message-CCCH,
    ext2                                Ext2Message-CCCH,
    extension                           DL-CCCH-MessageTypeExt
}

DL-CCCH-MessageTypeExt ::= CHOICE {
    Ext3                                Ext3Message-CCCH,
    spare3                              NULL,
    spare2                              NULL,
    spare1                              NULL
}
```

Example 10

For system information block types, the "SIB type" information element is also included in each of the segments. If in this case there are insufficient spare values, the last value can again be used to indicate "extension". If that value is used, an additional SIB type extension IE is included to distinguish between the additional SIB types. This additional IE is not included in the segments; it is only included in the scheduling information included in the MIB and/or the SBs.

NOTE: One could include this additional IE in the segments e.g. by changing the SIB-type into a choice as shown in example 11. This option should not be used since it involves additional overhead (more scarce BCH bits are needed to indicate the SIB type) and complicates the scheduling (more different SIB data sizes are to be considered).

```

FirstSegment ::=                    SEQUENCE {
    -- Other information elements
        sib-Type                        SIB-Type,
        seg-Count                       SegCount,
        sib-Data-fixed                  SIB-Data-fixed
}

SIB-Type ::=                            CHOICE {
        MasterInformationBlock              NULL,
        systemInformationBlockType1         NULL,
        systemInformationBlockType2         NULL,
        systemInformationBlockType3         NULL,
        systemInformationBlockType4         NULL,
        systemInformationBlockType5         NULL,
        systemInformationBlockType6         NULL,
        systemInformationBlockType7         NULL,
        systemInformationBlockType8         NULL,
        systemInformationBlockType9         NULL,
        systemInformationBlockType10        NULL,
        systemInformationBlockType11        NULL,
        systemInformationBlockType12        NULL,
        systemInformationBlockType13        NULL,
        systemInformationBlockType13-1      NULL,
        systemInformationBlockType13-2      NULL,
        systemInformationBlockType13-3      NULL,
        systemInformationBlockType13-4      NULL,
        systemInformationBlockType14        NULL,
        systemInformationBlockType15        NULL,
        systemInformationBlockType15-1      NULL,
        systemInformationBlockType15-2      NULL,
        systemInformationBlockType15-3      NULL,
        systemInformationBlockType16        NULL,
        systemInformationBlockType17        NULL,
        systemInformationBlockType15-4      NULL,
        systemInformationBlockType18        NULL,
        schedulingBlock1                    NULL,
        schedulingBlock2                    NULL,
        systemInformationBlockType15-5      NULL,
        ext1                                NULL,
        extension                           SIB-TypeExt
}

SIB-TypeExt ::=                     CHOICE {
        ext2                            NULL,
        spare7                          NULL,
        spare6                          NULL,
        spare5                          NULL,
        spare4                          NULL,
        spare3                          NULL,
        spare2                          NULL,
        spare1                          NULL
}
```

Example 11 – Not recommended

The addition of new SIB types to the scheduling information is illustrated by example 12. The example shows the extension of the choice. The example also shows that the information applicable for the extended choice values is appended at the end of the SIB (in this case the MIB), as a non critical extension.

NOTE: In this example only the number of SIB types is increased; the number of SIBs that can be scheduled (as reflected in the size of the list in the scheduling information) is not extended.

```
MasterInformationBlock ::=          SEQUENCE {
        mib-ValueTag                    MIB-ValueTag,
        -- TABULAR: The PLMN identity and ANSI-41 core network information
        -- are included in PLMN-Type.
        plmn-Type                       PLMN-Type,
        sibSb-ReferenceList             SIBSb-ReferenceList,
        vxy0NonCriticalExtensions           SEQUENCE {
            masterInformationBlock-vxy0ext      MasterInformationBlock-vxy0ext-IEs,
            nonCriticalExtensions               SEQUENCE {}                     OPTIONAL
        } OPTIONAL
}

SIBSb-ReferenceList ::=             SEQUENCE (SIZE (1..maxSIB)) OF
                                        SchedulingInformationSIBSb

SchedulingInformationSIBSb ::=          SEQUENCE {
    sibSb-Type                          SIBSb-TypeAndTag,
    scheduling                          SchedulingInformation
}

SIBSb-TypeAndTag ::=                    CHOICE {
    sysInfoType1                        PLMN-ValueTag,
    sysInfoType2                        CellValueTag,
    sysInfoType3                        CellValueTag,
    sysInfoType4                        CellValueTag,
    sysInfoType5                        CellValueTag,
    sysInfoType6                        CellValueTag,
    sysInfoType7                        NULL,
    sysInfoType8                        CellValueTag,
    sysInfoType9                        NULL,
    sysInfoType10                       NULL,
    sysInfoType11                       CellValueTag,
    sysInfoType12                       CellValueTag,
    sysInfoType13                       CellValueTag,
    sysInfoType13-1                     CellValueTag,
    sysInfoType13-2                     CellValueTag,
    sysInfoType13-3                     CellValueTag,
    sysInfoType13-4                     CellValueTag,
    sysInfoType14                       NULL,
    sysInfoType15                       CellValueTag,
    sysInfoType16                       PredefinedConfigIdentityAndValueTag,
    sysInfoType17                       NULL,
    sysInfoTypeSB1                      CellValueTag,
    sysInfoTypeSB2                      CellValueTag,
    sysInfoType15-1                     CellValueTag,
    sysInfoType15-2                     SIBOccurrenceIdentityAndValueTag,
    sysInfoType15-3                     SIBOccurrenceIdentityAndValueTag,
    sysInfoType15-4                     CellValueTag,
    sysInfoType18                       CellValueTag,
    sysInfoType15-5                     CellValueTag,
    ext1                                NULL,
    ext2                                NULL,
    extension                           NULL
}

SIBSb-TypeAndTagExt ::=                 CHOICE {
    ext3                                NULL,
    spare7                              NULL,
    spare6                              NULL,
    spare5                              NULL,
    spare4                              NULL,
    spare3                              NULL,
    spare2                              NULL,
    spare1                              NULL
}

MasterInformationBlock-vxy0ext-IEs ::= SEQUENCE {
    extSIBTypeInfoSchedulingInfo-List       ExtSIBTypeInfoSchedulingInfo-List   OPTIONAL
}

-- For each extended SIB type the value tag information is added at the end
ExtSIBTypeInfoSchedulingInfo-List::=    SEQUENCE (SIZE (1..maxSIB)) OF
                                            ExtSIBTypeInfoSchedulingInfo

ExtSIBTypeInfoSchedulingInfo-List::= SEQUENCE {
    schedulingInfoListIndex             INTEGER (1..maxSIB),
    valueTagInfo                        ValueTagInfo
}

ValueTagInfo ::= CHOICE {
    None                                NULL.
    sysInfoType2                        CellValueTag,
    sysInfoType1                        PLMN-ValueTag,
    sysInfoType15-3                     SIBOccurrenceIdentityAndValueTag
}
```

Example 12 – Recommended method

3GPP TSG RAN WG2#40 **_Tdoc ⌘R2-040301_**
Sophia Antipolis, France, 12th – 16th January 2004

CR-Form-v7

CHANGE REQUEST

⌘	**25.921**	**CR**	**59**	⌘ **rev**	**-**	⌘	Current version:	**4.6.0** ⌘

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

Proposed change affects: UICC apps⌘ ☐ ME ☐ Radio Access Network ☐ Core Network ☐

Title:	⌘	Guideline for the naming of extensions to the RRC ASN.1		
Source:	⌘	RAN WG2		
Work item code:	⌘	TEI	***Date:*** ⌘	15/01/2004
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)* *Rel-6 (Release 6)*

Reason for change:	⌘	Currently the REL-4 and REL-5 extensions are not named in a consistent manner. This is because the guidelines for the naming of extensions to the RRC ASN.1 are currently neither clear nor complete
Summary of change:	⌘	The is CR introduces a recommendation to be used when performing late corrections introducing non critical extensions in the RRC messages
Consequences if not approved:	⌘	There will be no way to achieve consistent naming of extensions to the RRC ASN.1

Clauses affected:	⌘	10.4.2			
		Y	**N**		
Other specs affected:	⌘		**X**	Other core specifications	⌘
			X	Test specifications	
			X	O&M Specifications	
Other comments:	⌘	The new convention means that some IEs introduced in a later release/ version need not apply a specific suffix. This means that it will be somewhat more difficult to track in which release/ version the IE was introduced. This requires some additional care when applying the backwards compatibility rules. However, since it has already been common practice to not always use an extension showing the release/ version suffix, this is not considered to be a problem			

How to create CRs using this form:

Comprehensive information and tips about how to create CRs can be found at http://www.3gpp.org/specs/CR.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.

10.4.2 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.

For critical 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 this should be the same as for the new message root. For this naming, "`r3`" is used for Release '99, "`r4`" for Release 4, "`r5`" for Release 5 and so on.

For non-critical extensions the new elements introduced to specify the extensions should be grouped together in an element with a name showing the version of the specification where this extension will first be included, e.g. if the version of the specification being corrected is v3.7.0, then the suffix added to the name will be -v380ext (i.e. the next version).

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.

An example is given below to illustrate these principles, on the message named "`Test-msg`".

```
-- In Release '99, the Test-msg is defined as following:
Test-msg ::= CHOICE {
    r3                              SEQUENCE {
        test-msg-r3                     Test-msg-r3-IEs,
        nonCriticalExtensions           SEQUENCE {} OPTIONAL
    },
    later-than-r3                   SEQUENCE {
        rrc-TransactionIdentifier       RRC-TransactionIdentifier,
        criticalExtensions              SEQUENCE {}
    }
}
-- A later correction to Release '99 adds a non-critical extension in v3.8.0
-- of the specification
Test-msg ::= CHOICE {
    r3                              SEQUENCE {
        test-msg-r3                     Test-msg-r3-IEs,
        v380nonCriticalExtensions           SEQUENCE {
            test-msg-v380ext                    Test-msg-v380ext-IEs,
            nonCriticalExtensions               SEQUENCE {} OPTIONAL
        } OPTIONAL
    },
    later-than-r3                   SEQUENCE {
        rrc-TransactionIdentifier       RRC-TransactionIdentifier,
        criticalExtensions              SEQUENCE {}
    }
}
-- The Test-msg gets the following structure, if only a non-critical
-- extensions is introduced for Release 4 in v4.4.0 of the specification.
Test-msg ::= CHOICE {
    r3                              SEQUENCE {
        test-msg-r3                     Test-msg-r3-IEs,
        v380nonCriticalExtensions           SEQUENCE {
            test-msg-v380ext                    Test-msg-v380ext-IEs,
            laterNonCriticalExtensions          SEQUENCE {
                -- Container for additional Release '99 extensions
                test-msg-r3-add-ext                 BIT STRING
                    (CONTAINING Test-msg-r3-add-ext-IEs)            OPTIONAL,
                v440nonCriticalExtensions           SEQUENCE {
                    test-msg-v440ext                    Test-msg-v440ext-IEs,
                    nonCriticalExtensions               SEQUENCE {} OPTIONAL
                }   OPTIONAL
            }   OPTIONAL
        } OPTIONAL
    },
    later-than-r3                   SEQUENCE {
        rrc-TransactionIdentifier       RRC-TransactionIdentifier,
        criticalExtensions              SEQUENCE {}
    }
}

-- In Release 5, the Test msg gets the following structure when a critical
-- extension is added
Test-msg ::= CHOICE {
    r3                              SEQUENCE {
        test-msg-r3                     Test-msg-r3-IEs,
        v380nonCriticalExtensions           SEQUENCE {
            test-msg-v380ext                    Test-msg-v380ext-IEs,
            laterNonCriticalExtensions          SEQUENCE {
                -- Container for additional Release '99 extensions
                test-msg-r3-add-ext                 BIT STRING
                    (CONTAINING Test-msg-r3-add-ext-IEs)            OPTIONAL,
                v440nonCriticalExtensions           SEQUENCE {
                    test-msg-v440ext                    Test-msg-v440ext-IEs,
                    nonCriticalExtensions               SEQUENCE {} OPTIONAL
                }   OPTIONAL
            }   OPTIONAL
        } OPTIONAL
    },
    later-than-r3                   SEQUENCE {
        rrc-TransactionIdentifier       RRC-TransactionIdentifier,
        criticalExtensions              CHOICE {
            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).

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. In case the type that is introduced in Release *N* includes one or more new (nested) types, the additional suffix need not be used for these nested types. In case the type that is introduced in Release *N* includes one or more revisions of exixting types, the suffix is needed to distinguish them from the earlier revisions. In case a revision of an abstract type that is introduced in Release *N* includes an IE for which the abstract type already existed in earlier releases, while that IE was not present in the previous revision(s) of the revised abstract type, the IE name should have a suffix "`-r`*`N`*".

If an abstract type is introduced in a release to extend an already existing type "`TypeX`", it should get the same name with a non-critical extension type suffix ("`-vXYZext`", e.g. "`TypeX-v380ext`") although in this case the final "–IEs" suffix is not added. In case the type that is introduced in Release *N* to extend an already existing type includes one or more new (nested) types that are extensions of an already existing type, the additional suffix should not be used for these nested types. In case the type that is introduced in Release *N* to extend an already existing type includes one or more new (nested) types, the abovely specified rules for new abstract types apply.

The above naming conventions are further illustrated in the example below:

```
Test-msg-v380ext-IEs ::= SEQUENCE {
        existingIE-A-v380ext            ExistingIE-A-v380ext        OPTIONAL,
        newIE-B                         NewIE-B
}

Test-msg-v440ext-IEs::= SEQUENCE {
        newIE-C-r4                      NewIE-C                     OPTIONAL,
        existingIE-D-v440ext            ExistingIE-D-v440ext
}

Test-msg-r5-IEs::= SEQUENCE {
        existingIE-E                    ExistingIE-E
        newUseOfexistingIE-E-rF         ExistingIE-F                OPTIONAL,
        newIE-G-r5                      NewIE-G
        revisionOfExistingIE-H-r5       ExistingIE-H-r5
}
```

The abovely described naming convention means that some IEs introduced in a later release/ version need not apply a specific suffix. This means that it will not allways be clear from the name of an IE whether or not backwards incompatible changes to it are allowed. ~~Using the above naming rules, when changes are done in Release *N*, only changes in types with a suffix "-rN" or "-vXYZext" are allowed, in order to avoid conflicts with previous releases. An exception is t~~The Message type ~~itself~~is a special case, 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,
        v380nonCriticalExtensions           SEQUENCE {
            test-msg-v380ext                Test-msg-v380ext-IEs,
            nonCriticalExtensions           SEQUENCE {} OPTIONAL
        } OPTIONAL
    },
    later-than-r3                   SEQUENCE {
        rrc-TransactionIdentifier       RRC-TransactionIdentifier,
        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 1: The structure presented in this clause and the proposed naming rules are one possibility. Further possibilities are FFS.

NOTE 2: When non-critical extensions are introduced in a message that does not have yet a criticalExtension branch, they are introduced in the "Test-msg-v380ext-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.

- To be 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.

3GPP TSG RAN WG2#40
Sophia Antipolis, France, 12th – 16th January 2004

Tdoc ⌘R2-040302

CR-Form-v7

CHANGE REQUEST

⌘	**25.921**	**CR**	**60**	⌘**rev**	**-**	⌘	Current version:	**5.3.0**	⌘

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

Proposed change affects: UICC apps⌘ ☐ ME ☐ Radio Access Network ☐ Core Network ☐

Title:	⌘	Guideline for the naming of extensions to the RRC ASN.1		
Source:	⌘	RAN WG2		
Work item code:	⌘	TEI	***Date:*** ⌘	15/01/2004
Category:	⌘	**A**	***Release:*** ⌘	REL-5
		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)* *Rel-6 (Release 6)*

Reason for change:	⌘	Currently the REL-4 and REL-5 extensions are not named in a consistent manner. This is because the guidelines for the naming of extensions to the RRC ASN.1 are currently neither clear nor complete
Summary of change:	⌘	The is CR introduces a recommendation to be used when performing late corrections introducing non critical extensions in the RRC messages
Consequences if not approved:	⌘	There will be no way to achieve consistent naming of extensions to the RRC ASN.1

Clauses affected:	⌘	10.4.2				
		Y	**N**			
Other specs affected:	⌘		**X**	Other core specifications	⌘	
			X	Test specifications		
			X	O&M Specifications		
Other comments:	⌘	The new convention means that some IEs introduced in a later release/ version need not apply a specific suffix. This means that it will be somewhat more difficult to track in which release/ version the IE was introduced. This requires some additional care when applying the backwards compatibility rules. However, since it has already been common practice to not always use an extension showing the release/ version suffix, this is not considered to be a problem				

How to create CRs using this form:

Comprehensive information and tips about how to create CRs can be found at http://www.3gpp.org/specs/CR.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.

10.4.2 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.

For critical 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 this should be the same as for the new message root. For this naming, "`r3`" is used for Release '99, "`r4`" for Release 4, "`r5`" for Release 5 and so on.

For non-critical extensions the new elements introduced to specify the extensions should be grouped together in an element with a name showing the version of the specification where this extension will first be included, e.g. if the version of the specification being corrected is v3.7.0, then the suffix added to the name will be -v380ext (i.e. the next version).

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.

An example is given below to illustrate these principles, on the message named "`Test-msg`".

```
-- In Release '99, the Test-msg is defined as following:
Test-msg ::= CHOICE {
    r3                              SEQUENCE {
        test-msg-r3                     Test-msg-r3-IEs,
        nonCriticalExtensions           SEQUENCE {} OPTIONAL
    },
    later-than-r3                   SEQUENCE {
        rrc-TransactionIdentifier       RRC-TransactionIdentifier,
        criticalExtensions              SEQUENCE {}
    }
}
-- A later correction to Release '99 adds a non-critical extension in v3.8.0
-- of the specification
Test-msg ::= CHOICE {
    r3                              SEQUENCE {
        test-msg-r3                     Test-msg-r3-IEs,
        v380nonCriticalExtensions           SEQUENCE {
            test-msg-v380ext                    Test-msg-v380ext-IEs,
            nonCriticalExtensions               SEQUENCE {} OPTIONAL
        } OPTIONAL
    },
    later-than-r3                   SEQUENCE {
        rrc-TransactionIdentifier       RRC-TransactionIdentifier,
        criticalExtensions              SEQUENCE {}
    }
}
-- The Test-msg gets the following structure, if only a non-critical
-- extensions is introduced for Release 4 in v4.4.0 of the specification.
Test-msg ::= CHOICE {
    r3                              SEQUENCE {
        test-msg-r3                     Test-msg-r3-IEs,
        v380nonCriticalExtensions           SEQUENCE {
            test-msg-v380ext                    Test-msg-v380ext-IEs,
            laterNonCriticalExtensions          SEQUENCE {
                -- Container for additional Release '99 extensions
                test-msg-r3-add-ext                 BIT STRING
                    (CONTAINING Test-msg-r3-add-ext-IEs)            OPTIONAL,
                v440nonCriticalExtensions           SEQUENCE {
                    test-msg-v440ext                    Test-msg-v440ext-IEs,
                    nonCriticalExtensions               SEQUENCE {} OPTIONAL
                }   OPTIONAL
            }   OPTIONAL
        } OPTIONAL
    },
    later-than-r3                   SEQUENCE {
        rrc-TransactionIdentifier       RRC-TransactionIdentifier,
        criticalExtensions              SEQUENCE {}
    }
}

-- In Release 5, the Test msg gets the following structure when a critical
-- extension is added
Test-msg ::= CHOICE {
    r3                              SEQUENCE {
        test-msg-r3                     Test-msg-r3-IEs,
        v380nonCriticalExtensions           SEQUENCE {
            test-msg-v380ext                    Test-msg-v380ext-IEs,
            laterNonCriticalExtensions          SEQUENCE {
                -- Container for additional Release '99 extensions
                test-msg-r3-add-ext                 BIT STRING
                    (CONTAINING Test-msg-r3-add-ext-IEs)            OPTIONAL,
                v440nonCriticalExtensions           SEQUENCE {
                    test-msg-v440ext                    Test-msg-v440ext-IEs,
                    nonCriticalExtensions               SEQUENCE {} OPTIONAL
                }   OPTIONAL
            }   OPTIONAL
        } OPTIONAL
    },
    later-than-r3                   SEQUENCE {
        rrc-TransactionIdentifier       RRC-TransactionIdentifier,
        criticalExtensions              CHOICE {
            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-rN-IEs`" (*N*=3 is used for Release '99).

If an abstract type is introduced in Release *N* when new elements are included in an extension, it should have a suffix "`-rN`". For Release '99 types, no such suffix is used. In case the type that is introduced in Release *N* includes one or more new (nested) types, the additional suffix need not be used for these nested types. In case the type that is introduced in Release *N* includes one or more revisions of exixting types, the suffix is needed to distinguish them from the earlier revisions. In case a revision of an abstract type that is introduced in Release *N* includes an IE for which the abstract type already existed in earlier releases, while that IE was not present in the previous revision(s) of the revised abstract type, the IE name should have a suffix "`-rN`".

If an abstract type is introduced in a release to extend an already existing type "`TypeX`", it should get the same name with a non-critical extension type suffix ("`-vXYZext`", e.g. "`TypeX-v380ext`") although in this case the final "–IEs" suffix is not added. In case the type that is introduced in Release *N* to extend an already existing type includes one or more new (nested) types that are extensions of an already existing type, the additional suffix should not be used for these nested types. In case the type that is introduced in Release *N* to extend an already existing type includes one or more new (nested) types, the abovely specified rules for new abstract types apply.

The above naming conventions are further illustrated in the example below:

```
Test-msg-v380ext-IEs ::= SEQUENCE {
        existingIE-A-v380ext            ExistingIE-A-v380ext        OPTIONAL,
        newIE-B                         NewIE-B
}

Test-msg-v440ext-IEs::= SEQUENCE {
        newIE-C-r4                      NewIE-C                     OPTIONAL,
        existingIE-D-v440ext            ExistingIE-D-v440ext
}

Test-msg-r5-IEs::= SEQUENCE {
        existingIE-E                    ExistingIE-E
        newUseOfexistingIE-E-rF         ExistingIE-F                OPTIONAL,
        newIE-G-r5                      NewIE-G
        revisionOfExistingIE-H-r5       ExistingIE-H-r5
}
```

The abovely described naming convention means that some IEs introduced in a later release/ version need not apply a specific suffix. This means that it will not allways be clear from the name of an IE whether or not backwards incompatible changes to it are allowed. ~~Using the above naming rules, when changes are done in Release *N*, only changes in types with a suffix "-r*N*" or "-vXYZext" are allowed, in order to avoid conflicts with previous releases. An exception is t~~The Message type ~~itself~~is a special case, 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,
        v380nonCriticalExtensions           SEQUENCE {
            test-msg-v380ext                Test-msg-v380ext-IEs,
            nonCriticalExtensions           SEQUENCE {} OPTIONAL
        } OPTIONAL
    },
    later-than-r3                   SEQUENCE {
        rrc-TransactionIdentifier       RRC-TransactionIdentifier,
        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 1: The structure presented in this clause and the proposed naming rules are one possibility. Further possibilities are FFS.

NOTE 2: When non-critical extensions are introduced in a message that does not have yet a criticalExtension branch, they are introduced in the "Test-msg-v380ext-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.

- To be 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.