Joint 3GPP TSG-S4#10 - SMG11#15 Meeting
Feb 28 to Mar 3 2000, Helsinki, Finland

Document ***S4-(00)0134***
e.g. for 3GPP use the format TP-99xxx
or for SMG, use the format P-99-xxx

Draft CHANGE REQUEST

Please see embedded help file at the bottom of this page for instructions on how to fill in this form correctly.

08.62 CR A002 Current Version: 7.0.0

GSM (AA.BB) or 3G (AA.BBB) specification number ↑ *↑ CR number as allocated by MCC support team*

For submission to:	SA4#??	for approval		strategic	X	*(for SMG*
list expected approval meeting # here ↑		for information		non-strategic		*use only)*

Form: CR cover sheet, version 2 for 3GPP and SMG *The latest version of this form is available from: ftp://ftp.3gpp.org/Information/CR-Form-v2.doc*

Proposed change affects: *(at least one should be marked with an X)*	(U)SIM		ME		UTRAN / Radio	X	Core Network	

Source:	Ericsson	**Date:**	29-02-2000
Subject:	TFO Messages for AMR		
Work item:	TFO		

Category:	F	Correction		**Release:**	Phase 2	
	A	Corresponds to a correction in an earlier release			Release 96	
(only one category	B	Addition of feature			Release 97	
shall be marked	C	Functional modification of feature	X		Release 98	
with an X)	D	Editorial modification			Release 99	
					Release 00	X

Reason for change:	To allow introduce AMR into TFO for GSM
Clauses affected:	6

Other specs affected:	Other 3G core specifications		→ List of CRs:	
	Other GSM core specifications		→ List of CRs:	
	MS test specifications		→ List of CRs:	
	BSS test specifications		→ List of CRs:	
	O&M specifications		→ List of CRs:	

Other comments:	

help.doc

<--------- double-click here for help and instructions on how to create a CR.

6 TFO Message Structure

Several TFO Messages are defined, based on the general IS_Message principle, as defined in Annex A.
Definition for ***Sender*** side:

TFO_REQ (): Identifies the source of the message as a TFO capable device, using a defined speech Codec_Type. TFO_REQ contains the following parameters ():

- the System_Identification of the sender;
- the specific Local_Signature of the sender (e.g. TRAU or TCME);
- the Local_Used_Codec_Type at sender side;
- possibly additional attributes for the Local_Used_Codec_Type.

TFO_ACK (): Is the response to a TFO_REQ Message.
TFO_ACK contains the corresponding parameters to TFO_REQ, but the Local_Signature is replaced by the Reflected_Signature, copied from the received TFO_REQ Message.

TFO_REQ_L (): Is sent in case of Codec Mismatch or for sporadic updates of information.
TFO_REQ_L contains the following parameters ():

- the System_Identification of the sender;
- the specific Local_Signature of the sender (e.g. TRAU or TCME);
- the Local_Used_Codec_Type at sender side;
- the Local_Codec_List of alternative Codec_Types;
- possibly additional attributes for the alternative Codec_Types.

TFO_ACK_L (): Is the response to a TFO_REQ_L Message.
TFO_ACK_L contains the corresponding parameters to TFO_REQ_L, but the Local_Signature is replaced by the Reflected_Signature, copied from the received TFO_REQ_L Message.

TFO_REQ_P (): Is used to indicate during ongoing TFO that an other Codec_Type would be preferred.
TFO_REQ_P contains the following parameters ():

- the System_Identification of the sender;
- the specific Local_Signature of the sender;
- the Preferred_Codec_Type at sender side (only used by TCME);
- possibly additional attributes for the Preferred_Codec_Type.

TFO_TRANS (): Commands possible IPEs to let the TFO Frames pass transparently within the LSB (8 kBit/s) or the two LSBs (16 kBit/s). TFO_TRANS contains the following parameter ():

- the Local_Channel_Type (8 kBit/s or 16 kBit/s).

TFO_NORMAL: Commands possible IPEs to revert to normal operation.
TFO_NORMAL has no parameters.

TFO_DUP: Informs the distant partner that TFO Frames are received, while still transmitting PCM samples.
TFO_DUP has no parameters.

TFO_SYL: Informs the distant partner (if still possible) that TFO Frames are no longer received.
TFO_SYL has no parameters.

TFO_FILL: Message without specific meaning, used to pre-synchronise IPEs or to bridge over gaps in TFO protocols. TFO_FILL has no parameters.

6.1 Extendibility

A mechanism for future extensions is defined in a way that existing implementations in the field shall be able to ignore future, for them unknown Codec_Types and their potential attributes. The existing implementations shall be able to decode the reminder of the messages (which is known to them) uncompromised. This mechanism allows to extent:

- the number of Local_Used_Codec_Types from 15 (short form) up to 255 (long form) for one System_Identification;
- the Codec_List;
- the Codec_Attributes (if needed).

In case of the TFO_REQ or TFO_ACK messages the attributes of the Local_Used_Codec_Type shall be sent in the Codec specific way, without a preceding Codec_Attribute_Head Extension_Block. Existing equipment, that do not know a future Codec_Type and therefore do not know if and how many attribute Extension_Blocks do follow, shall skip these Extension_Blocks, until they find a TFO Message Header again.

In case of the TFO_REQ_L or TFO_ACK_L Messages the simple Codec_List shall be sent immediately after the SIG_LUC and possible Codec_x Extension_Blocks. Then the attributes of all alternative Codec_Types shall follow. Each set of Codec attributes shall be preceded by the Codec_Attribute_Head Extension_Block (with Codec_Type Identifier and Length Indicater) followed by the Codec specific attributes.

TFO_REQ_P shall not contain the list of alternative Codecs, i.e. it shall be based on TFO_REQ and not on TFO_REQ_L.

6.2 Regular and Embedded TFO Messages

A TFO Message is called **”regular**”, if it is sent inserted into the PCM sample stream. A TFO Message is called ”**embedded**”, if it is sent together with (embedded into) TFO Frames, see also subclause 7.2. The bit stealing scheme (see Annex A) is identical for regular and embedded TFO Messages. Control bit C5 marks redundantly (in general) all TFO Frames that are affected by embedding a TFO Message. Due to the specific construction of the TFO Messages, they replace some of the synchronisation bits of the TFO Frames. TFO Frame synchronisation is in case of embedded TFO Messages therefore different, however, not endangered. Data and other control bits of the TFO Frames are not affected by embedded TFO Messages.

6.3 Cyclic Redundancy Check

The Extension_Blocks, defined in the following sub-clauses, shall be protected by three CRC parity bits. These shall be generated as define in GSM 08.60 for the Enhanced Full Rate. For simplicity this specification is reprinted here:

“These parity bits are added to the bits of the subset, according to a degenerate (shortened) cyclic code using the generator polynomial:

$$g(D) = D^3 + D + 1$$

The encoding of the cyclic code is performed in a systematic form which means that, in GF(2), the polynomial:

$$d(m)D^n + d(m+1)D^{n-1} + ...+ d(m + n-3)D^3 + p(0)D^2 + p(1)D + p(2)$$

where p(0), p(1), p(2) are the parity bits, when divided by g(D), yields a remainder equal to:

$$1 + D + D^2$$

For every CRC, the transmission order is p(0) first followed by p(1) and p(2) successively.”

In case of Extension_Blocks p(0)..p(2) are mapped to bits 16..18.

6.4 Definition of the TFO_REQ Messages

Symbolic Notation:
TFO_REQ (Sys_Id, LSig, Local_Used_Codec_Type[, Used_Codec_Attributes]).

TFO_REQ_L (Sys_Id, LSig, Local_Used_Codec_Type, Codec_List [, Alternative_Codec_Attributes]).

TFO_REQ_P (Sys_Id, LSig, Preferred_Codec_Type[, Preferred_Codec_Attributes]).

The TFO_REQ Messages conform to the IS_REQ Message, defined in the Annex A, with IS_System_Identification, followed by the SIG_LUC Extension_Block, optionally the Codec_x Extension_Block, the Codec_List Extension_Block(s) and the Codec_Attribute Extension_Blocks.

The shortest TFO_REQ takes 140 ms for transmission, see Figure 2a.
The shortest TFO_REQ_L takes 180 ms (Figure 2b).
The shortest TFO_REQ_P takes 180 ms for transmission (Figure 2c).

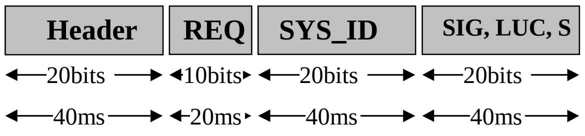

Figure 2a: Construction of the shortest possible TFO_REQ Message

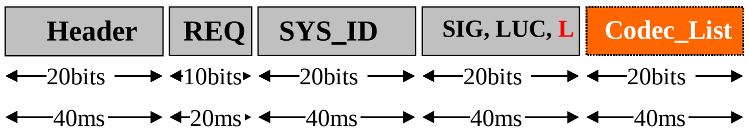

Figure 2b: Construction of the shortest possible TFO_REQ_L Message

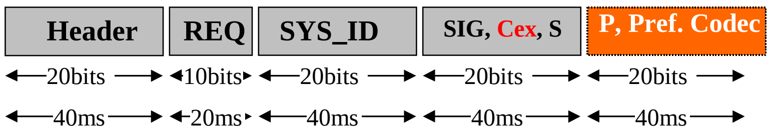

Figure 2c: Construction of the shortest possible TFO_REQ_P Message

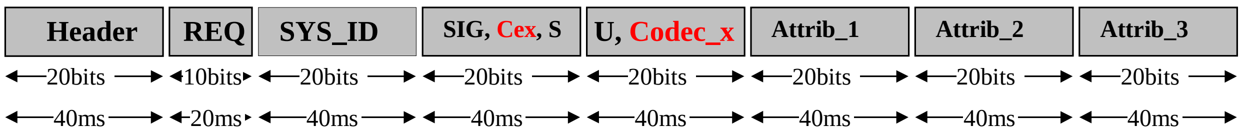

Figure 2d:Example of a TFO_REQ Message with a Codec with an index higher than 15 and with three Attribute Extension_Blocks (300 ms length)

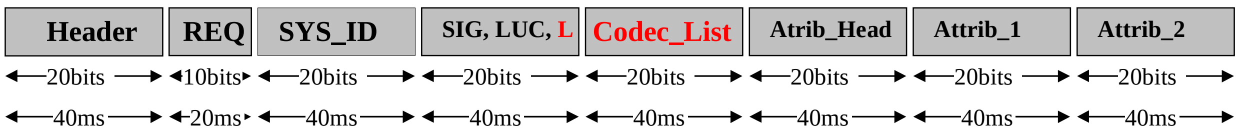

Figure 2e: Example of a TFO_REQ_L Message with Codec_List and one alternative Codec with two Attribute Extension_Blocks (300 ms length)

6.4.1 Definition of the SIG_LUC Extension_Block

The SIG_LUC Extension_Block consists of 20 bits, as defined in Table 4. It shall follow always immediately after the SYS_ID Extension_Block. It differentiates between TFO_REQ and TFO_REQ_L messages, respectively between TFO_ACK and TFO_ACK_L messages.

In case of a TFO_REQ_P message it shall be followed immediately by the Codec_x Extension_Block (Table 5).

The Codec_x Extension_Block shall also be used if the Local_Used_Codec_Type has a CoID higher than 14.

Table 4: SIG_LUC Extension_Block

Bit	**Description**	**Comment**
Bit 1	”**0**”	normal IS-Message Sync Bit, constant.
Bit 2	**List_Ind**	Indicates, whether the Codec_List is included in the TFO Message or not **0: S: TFO_REQ or TFO_REQ_P** or **TFO_ACK**: Codec_List is not included (short) **1: L: TFO_REQ_L** or **TFO_ACK_L**: Codec_List is included (long)
Bit 3..10	**Sig**	An 8-bit random number to facilitate the detection of circuit loop back conditions and to identify the message source
Bit 11	”**0**”	normal IS-Message Sync Bit, constant
Bit 12.. 15:	**Codec_Type** **CoID_s** (short form)	Identifies the Local_Used_Codec_Type, which is currently used by the sender **0000…1110: reserved for 15 Codec_Types** **1111: Codec_X** Extension_Block follows immediately, e.g. for “Preferred Codec” Type (Codec_Type, long form)
Bit 16..18:	**CRC**	3 CRC bits protecting Bits 2 to 10 and 12 to 15
Bit 19..20:	**EX** EX == "0.0" EX == "1.1"	The normal 2 bits for IS_Message Extension. No other extension block follows An other extension block follows

6.4.2 Definition of the Codec_x Extension_Block

The Codec_x Extension_Block consists of 20 bits, as defined in Tabel 5. It shall follow always immediately after the SIG_LUC Extension_Block, if the Codec_Type field is set to “1111”.

Table 5: Codec_x Extension_Block

Bit	**Description**	**Comment**
Bit 1	”**0**”	normal IS-Message Sync Bit, constant.
Bit 2	**Codec_Sel**	Differentiates the Codec_x Extension_Block **0: U: Used_Codec_Type** is defined in Codec_Type_x field **1: P: Preferred_Codec_Type** is defined in Codec_Type_x field Note: The Preferred_Codec_Type is only defined in TFO_REQ Messages. It is reserved for future use in TFO_ACK messages.
Bit 3..10	**Codec_Type_x** **CoID** (long form)	Identifies the Local_Used_Codec_Type, which is currently used by the sender **0000.0000 … 1111.1111 reserved for 255 Codec_Types** 0000.1111 is undefined and shall not be used.
Bit 11	”**0**”	normal IS-Message Sync Bit, constant
Bit 12.. 15:	“1010”	reserved for future use, set to “1010” to minimise audible effects
Bit 16..18:	**CRC**	3 CRC bits protecting Bits 2 to 10 and 12 to 15
Bit 19..20:	**EX**	The normal 2 bits for IS_Message Extension. 00: No other extension block follows 11: An other extension block follows

6.4.3 Definition of the Codec_List_Extension_Block

The Codec_List Extension Block consists of 20 bits, as defined in Table 6. It identifies the Codec_Types that are supported by the sender, respectively the BSS subsystem, including the mobile station and the radio resource at sender side. The Codec_List must at least contain the Local_Used_Codec_Type. If a system supports more than 12 Codec_Types, then possibly other Codec_List Extension_Blocks (Table 7) may follow.

Table 6: Codec_List Extension Block

Bit	Description	Comment
Bit 1	”**0**”	normal IS-Message Sync Bit, constant.
Bit 2..10	**Codec_List_1**	First part of Codec_List. For each Codec_Type one bit is reserved. If the bit is set to ”0” then the specific Codec_Type is not supported; if the bit is set to ”1” then the specific Codec_Type could be used.
Bit 11	”**0**”	normal IS-Message Sync Bit, constant
Bit 12.. 14:	**Codec_List_2**	Second part of the Codec_List; All three bits are reserved for future Codec_Types (up to Codec_Type 12)
Bit 15	**Codec_List_x**	If set to “1” a further Codec_List Extension_Block follows; otherwise set to “0”
Bit 16..18:	**CRC**	3 CRC bits protecting Bits 2 to 10 and 12 to 15
Bit 19..20:	**EX**	The normal 2 bits for IS_Message Extension: 00: No other extension block follows 11: An other extension block follows

Table 7: Further Codec_List Extension Block(s)

Bit	Description	Comment
Bit 1	”**0**”	normal IS-Message Sync Bit, constant.
Bit 2..10	**Codec_List_1x**	First part of Codec_List. For each Codec_Type one bit is reserved. If the bit is set to ”0” then the specific Codec_Type is not supported; if the bit is set to ”1” then the specific Codec_Type could be used. Bit 2: Codec_Type 13 (+ x*12; x=1..2..3) Bit 4: Codec_Type 14 (+ x*12; x=1..2..3) and so on
Bit 11	”**0**”	normal IS-Message Sync Bit, constant
Bit 12.. 14:	**Codec_List_2x**	Second part of the Codec_List;All three bits are reserved for future Codec_Types (up to Codec_Type 24 (+x*12; x=1..2..3)
Bit 15	**Codec_List_xx**	If set to “1” a further Codec_List Extension_Block follows; otherwise set to “0”
Bit 16..18:	**CRC**	3 CRC bits protecting Bits 2 to 10 and 12 to 15
Bit 19..20:	**EX**	The normal 2 bits for IS_Message Extension: 00: No other extension block follows 11: An other extension block follows

6.4.4 Definition of the Codec_Attribute_Head Extension_Block

The Codec_Attribute_Head Extension_Block (Table 8) shall precede the Codec Attribute Extension_Blocks of a Codec_Type, if this Codec_Type needs additional attributes. Then this Codec_Attribute_Head identifies the Codec_Type and the number of additional Extension_Blocks for it.

Table 8: Codec_Attribute_Head Extension_Block

Bit	Description	Comment
Bit 1	”**0**”	normal IS-Message Sync Bit, constant.
Bit 2	**PAR_Sel**	Differentiates this Extension_Block **0:** Parameters included in **PAR** field: Simple Codec_List_Extension **1:** Length Indicator (**LI**) included: Parameters follow in subsequent Extension_Blocks
Bit 3..10	**CoID**	This field identifies the Codec_Type for which the subsequent attributes are valid. The same coding as in the Codec_x Extension_Block is used (long form)
Bit 11	”**0**”	normal IS-Message Sync Bit, constant
Bit 12.. 15:	**LI / PAR**	If Par_Sel==1: LI: Length Indicator: 0000: reserved; 0001: one other Extension_Block follows, etc. If Par_Sel==0: PAR: Codec specific definition of these four bits
Bit 16..18:	**CRC**	3 CRC bits protecting Bits 2 to 10 and 12 to 15
Bit 19..20:	**EX**	The normal 2 bits for IS_Message Extension: 00: No other extension block follows 11: An other extension block follows

Note : this Extension_Block shall be used for the codecs introduced in the future that need attributes. It shall precede the Attribute Extension_Blocks. This allows earlier versions to skip the blocks they do not understand. It shall not be used for the FR, HR and EFR Codec_Types.

6.5 Definition of the TFO_ACK Messages

Symbolic Notation:
TFO_ACK (Sys_Id, RSig, Local_Used_Codec_Type [, Used_Codec_Attributes])

TFO_ACK_L (Sys_Id, RSig, Local_Used_Codec_Type, Codec_List [, Alternative_Codec_Attributes]).

TFO_ACK_P: undefined, reserved for future use.

The TFO_ACK Messages conform to the IS_ACK Message, defined in the Annex A, with IS_System_Identification, followed by the SIG_LUC Extension_Block, optionally the Codec_x Extension_Block, the Codec_List Extension_Block(s) and the Codec_Attribute Extension_Blocks.

TFO_ACK and TFO_REQ Messages differ only in the ACK / REQ Command block and the construction of the Signature: Local_Signature in case of TFO_REQ, Reflected_Signature in case of TFO_ACK. All extension blocks defined for the TFO_REQ are valid as well for TFO_ACK.

The shortest TFO_ACK takes 140 ms for transmission.
The shortest TFO_ACK_L takes 180 ms.

6.6 Definition of the TFO_TRANS Messages

Symbolic Notation: TFO_TRANS (Channel_Type).

Two TFO_TRANS Messages are defined in conformity to the IS_TRANS Messages in Annex A.
For 8 kBit/s submultiplexing the "**TFO_TRANS (8k)**" is used and is identical to "IS_TRANS_1_u".
For 16 kBit/s submultiplexing the "**TFO_TRANS (16k)**" is used and is identical to "IS_TRANS_2_u".

TFO_TRANS() takes 100 ms for transmission.

In most cases the respective TFO_TRANS Message shall be sent twice: once as a regular TFO Message, exactly before any series of TFO Frames, and once embedded into the first TFO Frames, see clause 10.

6.7 Definition of the TFO_NORMAL Message

Symbolic Notation: TFO_NORMAL.

The TFO_NORMAL Message is identical to the IS_NORMAL Message defined in the Annex A.

It shall be sent at least once whenever an established tandem free operation need to be terminated in a controlled way.

TFO_NORMAL takes 100 ms for transmission.

6.8 Definition of the TFO_FILL Message

Symbolic Notation: TFO_FILL.

The TFO_FILL Message is identical to the IS_FILL Message, defined in the Annex A.

TFO_FILL may be used to pre-synchronise IPEs. Since IS_FILL is one of the shortest IS Messages, this is the fastest way to synchronize IPEs, without IPEs swallowing other protocol elements. By default three TFO_Messages shall be sent at the beginning; this number may be, however, configuration dependent.

One TFO_FILL takes 60 ms for transmission.

6.9 Definition of the TFO_DUP Message

Symbolic Notation: TFO_DUP

The TFO_DUP Message is identical to the IS_DUP Message, defined in Annex A.

TFO_DUP informs the distant TFO Partner, that TFO Frames have been received unexpected, e.g. during Establishment. This enables a fast re-establishment of TFO after a ***local*** handover.

TFO_DUP takes 60 ms for transmission.

6.10 Definition of the TFO_SYL Message

Symbolic Notation: TFO_SYL

The TFO_SYL Message is identical to the IS_SYL Message, defined in Annex A.

TFO_SYL informs the distant TFO Partner, that tandem free operation has existed, but suddenly no TFO Frame were received anymore. This enables a fast re-establishment of TFO after a ***distant*** handover.

TFO_SYL takes 60 ms for transmission.

6.11 Specification of the TFO Messages for GSM

6.11.1 The GSM Codec_Types

The GSM Codec_Types are defined in long form (Codec_Type_x, CoID) as
Bit 3…Bit10 (Codec_Type_x):

CoID	**Codec_Type**
0000.0000:	GSM Full Rate
0000.0001:	GSM Half Rate
0000.0010:	GSM Enhanced Full Rate
0000.0011:	GSM Adaptive Multi-Rate – FR (GSM AMR_FR)
0000.0100:	GSM Adaptive Multi-Rate – HR (GSM AMR_HR)
other codes:	reserved for future use.

The short form (CoID_s) exists for all Codec_Types with indices below 15 and consists of the last four bits of the long form (CoID).

6.11.2 The GSM Codec_List

For GSM the Codec_List is defined as:

	Codec_Type
Bit 2:	GSM Full Rate
Bit 3:	GSM Half Rate
Bit 4:	GSM Enhanced Full Rate
Bit 5:	GSM Adaptive Multi-Rate – FR (GSM AMR_FR)
Bit 6:	rGSM Adaptive Multi-Rate – HR (GSM AMR_HR)

The remaining bits are reserved for future Codec_Types.

6.11.3 The GSM Codec_Type Attributes

The GSM Codec_Types Full Rate, Half Rate and Enhanced Full Rate do not need additional attibutes. They are fully defined by their System_Identification (see Annex A.5) and Codec_Type.

6.11.3.1 The GSM AMR Codec_Type Attributes

The GSM Adaptive Multi-Rate Codec_Types, for both the full rate and half rate radio channels, need several attributes within the TFO_REQ and TFO_ACK as well as in the TFO_REQ_L and TFO_ACK_L Messages.

There are two major kinds of attributes: the ACS (Active Codec Set) and potentially the SCS (Supported Codec Set).

The ACS is related to the Local_Used_Codec_Type and is part of the Used_Codec_Attributes or Preferred_Codec_Attributes. One and exactly one ACS shall be sent in all cases where the Local_Used_Codec_Type is AMR_FR or AMR_HR within one ACS_Extention_Block. This ACS_Extention_Block carries some more parameters, see next subclause, the most important one is the "Full_Sub" flag, indicating whether or not the full set or a sub-set of the AMR is suppported. In TFO_REQ and TFO_ACK Messages the ACS shall follow immediately after the SIG_LUC_Extention_Block, respectively the Codec_x Extension_Block. In TFO_REQ_L and_TFO_ACK_L Messages an Attribute_Head_Extention_Block shall follow after the Local_Codec_List, indicating the Codec_Type it specifies, followed by the corresponding ACS_Extention_Block.

The SCS shall be sent in TFO_REQ or TFO_ACK only if the ACS_Extention_Block indicates that the BSS does not support the full AMR mode set, but a subset (Full_Sub flag). In the sub-set case the SCS_Extention_Block shall follow immediately after the ACS_Extention_Block. Note: In that way immediately after TFO_REQ or TFO_ACK the TFO_Protocol can decide whether TFO is possible or not and can report the distant TFO parameters to BTS and BSC.

Within TFO_REQ_L and TFO_ACK_L at maximum one ACS is necessary (for the Local_Used_Codec_Type, if it is AMR_FR or AMR_HR), but possibly two different SCSs may be needed: one for AMR_FR and one for AMR_HR, if both are flagged in the Local_Codec_List. In that case an Attribute_Head_Extension_Block shall follow after the Local_Codec_List, indicating the Codec_Type it specifies, followed by the corresponding SCS_Extention_Block. If both AMR_Codec_Types are flagged, then possibly two Attribute_Heads and SCSs may be needed. If the full sets are supported, then neither the Attribute_Head nor the SCS shall be sent for the alternative Codec_Type(s).

The following figures give the examples for the full-set AMR TFO Messages.

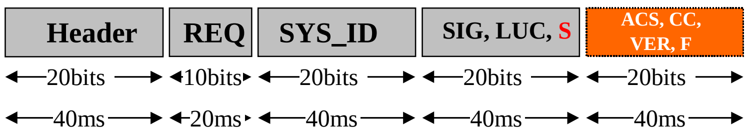

Figure 6.11a: Construction of the shortest possible TFO_REQ Message for AMR_FR or AMR_HR
TFO_ACK follows the same construction. Both have a length of 180ms

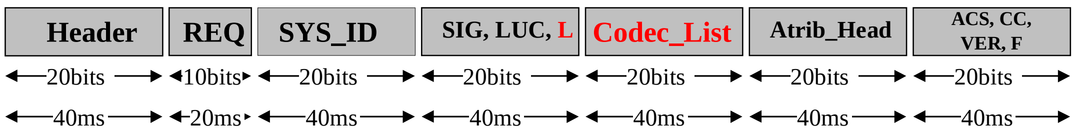

Figure 6.11b: Construction of the shortest possible TFO_REQ_L Message for AMR_FR or AMR_HR
TFO_ACK_L follows the same construction. Both have a length of 260ms.

Note: In TFO_REQ_L (TFO_ACK_L) at least one Attribute_Head is needed, if the Local_Used_Codec_Type is AMR_FR or AMR_HR, because otherwise a TFO partner that does not know the Local_Used_Codec_Type can not know how many attributes are needed – if any. Since these longer messages are used only when mismatch is identified or in other situations, where protocol speed is not important, this additional 40ms message length is not important.

In the worst case, when both AMR Codec_Types are flagged in the Codec_List, but none supports the full set, then five Extention_Blocks need to follow after the Codec_List (example with AMR_FR == Local_Used_Codec_Type): Attribute_Head(FR) – ACS(FR) – SCS(FR) – Attribute_Head(HR) – SCS(HR).

6.11.3.1.1 GSM AMR Active_Codec_Set Attributes

Within the TFO_REQ and TFO_ACK Messages one AMR_ACS Extension_Block shall be added after the SIG_LUC Extension_Block, respectively the Codec_x Extension_Block, if the AMR is the Local_Used_Codec_Type.

Table 9: GSM_AMR_ACS Extension Block

Bit	**Description**	**Comment**
Bit 1	”**0**”	Normal IS-Message Sync Bit, constant.
Bit 2..9	**Active Codec Set (ACS)**	Active Codec Set: For each Codec_Mode of the AMR one bit is reserved. If the bit is set to ”0” then the specific Codec_Mode is not in the ACS, otherwise it is in and may be used by the adaptation algorithm. Bit 2: AMR_Mode 12,2 kbit/s (undefined for AMR_HR) Bit 3: AMR_Mode 10,2 kbit/s (undefined for AMR_HR) Bit 4: AMR_Mode 7,95 kbit/s Bit 5: AMR_Mode 7,40 kbit/s Bit 6: AMR_Mode 6,70 kbit/s Bit 7: AMR_Mode 5,90 kbit/s Bit 8: AMR_Mode 5,15 kbit/s Bit 9: AMR_Mode 4,75 kbit/s
Bit 10	**Full_Sub (F/S)**	0: **F**ull Set supported, SCS is not following 1: **S**ubset only supported, SCS is following immediately
Bit 11	”**0**”	Normal IS-Message Sync Bit, constant
Bit 12 & 13	**Change_Category (CC)**	ACS Change Category (see subclause 8.3.2.2) 00 reserved for future use 01 No change supported 10 Slow Change supported by BSC 11 Fast change by RATSCCH supported
Bit 14 & 15	**Ver**	**Ver**sion Number of the AMR TFO Scheme
Bit 16..18	**CRC**	3 CRC bits protecting Bits 2 to 10 and 12 to 15
Bit 19..20:	**EX**	The normal 2 bits for IS_Message Extension: 00: No other extension block follows 11: An other extension block follows (i.e. SCS)

6.11.3.1.2 GSM AMR Supported Codec Set Attributes

The AMR_SCS Extension Block contains the information on the AMR Supported Codec Set. It shall be omitted, if the full set is supported. Table 12 gives the description of the SCS Extension_Block.

For the Local_Used_Codec_Type the SCS Extension_Block shall follow immediately after the corresponding ACS Extension_Block. In that case the Full_Sub flag shall be set within the ACS Extrension_Block. For alternative Codec_Types, as flagged in the Local_Codec_List, the SCS shall follow immediately after the corrsponding Attribute_Head Extension_Block, without the (unnecessary and unspecified) ACS Extension_Block.

Note: The VERsion numbers in ACS and SCS Extension_Blocks shall be identical (and are therefore redundant) for one Codec_Type, but may be different for different Codec_Types (e.g. AMR_FR and AMR_HR).

Table 10: GSM_AMR_SCS Extension Block

Bit	**Description**	**Comment**
Bit 1	"**0**"	Normal IS-Message Sync Bit, constant.
Bit 2…9	**Supported Codec Set (SCS)**	Supported Codec Set: For each Codec_Mode of the AMR one bit is reserved. If the bit is set to "0" then the specific Codec_Mode is not supported; if the bit is set to "1" then the specific Codec_Mode is supported and may be considered for the optimisation of the common ACS. Bit 2: AMR_Mode 12,2 kbit/s (undefined in SCS(H)) Bit 3: AMR_Mode 10,2 kbit/s (undefined in SCS(H)) Bit 4: AMR_Mode 7,95 kbit/s Bit 5: AMR_Mode 7,4 kbit/s Bit 6: AMR_Mode 6,7 kbit/s Bit 7: AMR_Mode 5,9 kbit/s Bit 8: AMR_Mode 5,15 kbit/s Bit 9: AMR_Mode 4,75 kbit/s
Bit 10	spare	Set to "0" (for future use)
Bit 11	"**0**"	normal IS-Message Sync Bit, constant
Bit 12…13	**MACS**	The maximally supported number of Codec_Modes in this radio leg. Coding: "0.1" 1 Mode "1.0" 2 Modes "1.1" 3 Modes "0.0" 4 Modes
Bit 14…15	**Ver**	Version Number of the AMR TFO Scheme for that Codec_Type
Bit 16..18	**CRC**	3 CRC bits protecting Bits 2 to 10 and 12 to 15
Bit 19 20	**EX**	The normal 2 bits for IS_Message Extension: 00: No other extension block follows 11: An other extension block follows