G23-GSM Protocol Stack

PCM – Permanent Configuration Memory Driver

Functional Specification

Author: Condat AG
Alt Moabit 90a
10559 Berlin
Germany

Date: 27 February, 2003

ID: 8415.001.02.029

Status: Being Processed

Table of Contents

0 Document Control

Every effort has been made to ensure that the information contained in this document is accurate at the time of printing. However, the software described in this document is subject to continuous development and improvement. Condat AG reserves the right to change the specification of the software. Information in this document is subject to change without notice and does not represent a commitment on the part of Condat AG. Condat AG accepts no liability for any loss or damage arising from the use of any information contained in this document.

The software described in this document is furnished under a license agreement and may be used or copied only in accordance with the terms of the agreement. It is an offence to copy the software in any way except as specifically set out in the agreement. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying and recording, for any purpose without the express written permission of Condat AG.

Condat AG
Alt Moabit 90a
10559 Berlin
Germany

Telephone: +49.30.39 49 0
Fax: +49.30.39 49 1300
Internet: www.condat.de

0.1 Document History

ID	Author	Date	Status
8415.001.98.001	AK et al	Sep. 9, 98	Being Processed
8415.001.98.002	LM et al	Sep. 11, 98	Being Processed
8415.001.98.003	LM et al	Sep. 21, 98	Being Processed
8415.001.98.004	LE et al	Nov. 30, 98	Being Processed
8415.001.98.005	LM et al	Dec. 17, 98	Being Processed
8415.001.98.010	LM et al	Dec. 17, 98	Being Processed
8415.001.99.011	MS et al.	Mar. 2, 99	Being Processed
8415.001.99.012	LE et al.	Jun. 2, 99	Being Processed
8415.001.99.013	VO et al.	Jun. 22, 99	Being Processed
8415.001.99.014	AK et al.	Jul. 01, 99	Being Processed
8415.001.99.015	SAB et al.	July 02, 99	Being Processed
8415.001.99.016	VO et al.	Sep.7, 99	Being Processed
8415.001.99.017	SAB et al.	Sep. 16, 99	Being Processed
8415.001.99.018	UB et al.	Feb. 04, 00	Being Processed

8415.001.99.019	HM et al.	Jan. 10, 01	Being Processed
8415.001.99.020	HM et al.	Aug. 30, 01	Being Processed
8415.001.99.021	OT et al.	July 9, 01	Being Processed
8415.001.99.022	ENZ et al.	Oct. 16, 01	Being Processed
8415.001.99.023	VK et al.	Nov. 02, 01	Being Processed
8415.001.01.024	RM et al.	Dec. 06, 01	Being Processed
8415.001.06.025	HM et al.	Mar. 19, 02	Being Processed
8415.001.02.026	MSB	Apr. 22, 02	Being Processed
8415.001.02.027	SBK	June 21, 02	Being Processed
8415.001.02.028	SBK	July 8, 02	Being Processed
8415.001.02.029	SBK	Feb 27, 03	Being Processed

0.2 References

[C_7010.801] 7010.801, References and Vocabulary, Condat AG

[GSM_03.38] ETS 300 628: September 1994 (GSM 03.38 version 4.0.1)
Alphabets and language-specific information; ETSI

[GSM_03.40] ETS 300 536: January 1996 (GSM 3.40 version 4.13.0)
Technical Realization of the Short Message Service Point-to-Point; ETSI

[GSM_04.08] ETS 300 557: January 1996 (GSM 4.08 version 4.17.0)
Mobile Radio Interface Layer 3 Specification; ETSI

[GSM_04.11] ETS 300 559: September 1996 (GSM 4.11 version 4.10.1)
Point-to-Point Short Message Service Support on Mobile Radio Interface; ETSI

[GSM_11.11] ETS 300 608: January 1995 (GSM 11.11 version 4.13.1)
Specification of the Subscriber Identity Module -Mobile Equipment (SIM - ME) interface; ETSI

0.3 Abbreviations

DTM	Dual Transfer Mode
E-OTD	Enhanced Observed Timing Difference
ECSD	Enhanced Circuit-Switched Channels.
EDGE	Enhanced Data Rates for GSM Evolution.
EGPRS	Enhanced General Packet Radio Service
GPRS	General Packet Radio Service
HSCSD	High Speed Circuit Switched Data
ME	Mobile Equipment
MS	Mobile Station
SoLSA	Support of Localized Service Area
TA	Terminal Adapter
VBS	Voice Broadcast Service
VGCS	Voice Group Call Service

0.4 Terms

---

1 Introduction

G23 is a software package implementing Layers 2 and 3 of the ETSI-defined GSM air interface signaling protocol, and as such represents the part of a GSM mobile station's protocol software which is both, platform and manufacturer independent. Therefore, G23 can be viewed as a building block providing standardized functionality through generic interfaces for easy integration.

The G23 suite of products consists of the following items:

- Layers 2 and 3 for speech & short message services,
- Layers 2 and 3 for fax & data services,
- Application Control Interface,
- Slim MMI [02.30] and
- Test and integration support tools.

This document describes the functional interface of the G23 permanent configuration memory driver. Mobile Stations usually use two types of permanent storage areas to save volatile data even if the mobile is switched off. One type of memory is the SIM card which has a well defined content specified by GSM 11.11. On the other hand, an electrical erasable memory (e.g. EEPROM, FLASH EPROM) is provided by the mobile station for which the memory layout and contents are manufacturer dependent. The following concepts were taken into account when designing the interface.

- **Memory structure is compatible with SIM structure**
  Some kinds of information, such as dialing numbers, could be stored in both types of memories. Therefore, the memory structure for ME and SIM is designed to be compatible to support the implementation for data exchanges in an easy way.

- **Abstract application level**
  To avoid hardware dependencies, the actual presentation and implementation of the data in the ME memory is hidden for the application. The application only has access to small elementary files, such as the files on the SIM card, by calling driver functions. If the application requests an elementary file for reading, it will only get a copy of the requested information. Therefore, changes of the stored information can only be performed by the driver itself. A possible change of the hardware dependencies for the ME memory affects the implementation of the driver only and has no impact on the application.

- **Flexible extension**
  If an extension of the stored data is necessary, it is possible to add new elementary files. Recompilation is reduced to the driver and the affected component of the application software.

- **Power consumption**
  To reduce the power consumption and the number of write cycles of the permanent memory to a minimum, the data is cached in a RAM area. A driver call is provided to flush the RAM area and update the contents of the ME memory.

- **Data integrity**
  To ensure that only valid data is accessed by the application, a checksum is provided for each elementary file. After copying the ME memory data into the RAM area, the checksum of each file is

verified. If the application wants to access a corrupted file, an error code is returned. The algorithm for calculating the checksum must be defined.(complement of a modulo 256 sum)

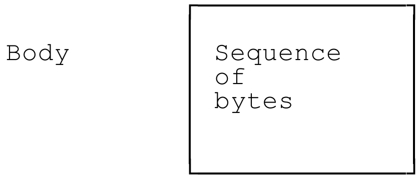
- **Version control**
  To support a version control, each elementary file has a version number to identify the state of implementation. Therefore, an application is able to check the version number of the file to ensure the correct access to the file data.

- **Structured data access**
  To provide a transparent way to access the data stored in the elementary files, a C structure is defined for each file.

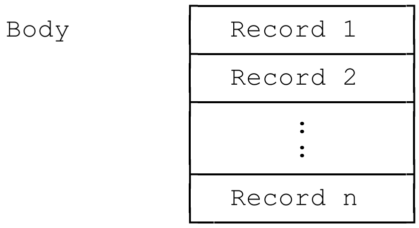
- **Data security**
  To protect the layout and structure of elementary files of which content is security related (e.g. IMEI, SIMlock), an encryption algorithm is applied to the data. This will ensure that a direct access to the permanent configuration memory will not present the data contents in a readable form.

- The permanent configuration memory of the ME is organized as a collection of elementary files (EF). To support different types of information elements, two types of elementary files are necessary:

- **Transparent EF**
  An EF with a transparent structure consists of a sequence of bytes. Information elements which are grouped into a transparent EF can be different sizes.



- **Linear fixed EF**
  An EF with linear fixed structure consists of a sequence of records all with the same (fixed) length. The first record is record number 1 (compatible with the SIM card). The length of a record as well as this value multiplied by the number of records is the total required file space in the permanent memory.



2 Interface description of the PCM driver

2.1 Data types

Name	Description
pcm_FileInfo_Type	File information
pcm_EFmscap_Type	Mobile capabilities
pcm_EFimei_Type	IMEI and IMEISV.
pcm_EFimsi_Type	IMSI
pcm_EFsms_Type	Storage area for short messages.
pcm_EFclass2_Type	MS classmark 2 parameters
pcm_EFclass3_Type	MS classmark 3 parameters
pcm_EFrfcap_Type	MS capabilities, related to RF capabilities and classmark information
pcm_EFmssup_Type	Setup parameters and factory defaults
pcm_EFclng_Type	current language
pcm_EFmsset_Type	User profiles
pcm_EFldn_Type	Last 10 MOC numbers
pcm_EFlrn_Type	Last 10 MTC numbers
pcm_EFlmn_Type	Last 10 MTC missed numbers
pcm_EFupn_Type	Storage area for personal phonebook
pcm_EFmbn_Type	Storage area for mail box numbers
pcm_EFvmn_Type	Voice Mail Number
pcm_EFctim_Type	Call timers for different call types
pcm_EFccnt_Type	Call counters for different call types
pcm_EFecc_Type	Storage for emergency call numbers
pcm_EForg_Type	Storage for organizer entries
pcm_EFccp_Type	Network/bearer capabilities and ME configurations
pcm_EFext1_Type	Storage area for extension data
pcm_EFsimlck_Type	SIM card identification information
pcm_EFsimlckext_Type	SIM card identification information (extended)
pcm_EFmnt_Type	Service and maintenance parameters
pcm_EFsfk_Type	Programmable key information
pcm_EFflt_Type	Storage area for fault conditions
pcm_EFdbg_Type	Storage area for debug information
pcm_EFbat_Type	Power management parameters
pcm_EFkbd_Type	Keyboard map
pcm_EFrdio_Type	Radio path parameters
pcm_EFplmn_Type	PLMN identifier
pcm_EFcgmi_Type	Manufacturer
pcm_EFcgmm_Type	Model
pcm_EFcgmr_Type	Revision
pcm_EFcgsn_Type	Product Serial Number
pcm_EFsmsprfl_Type	SMS Profile
pcm_EFbcch_info_Type	BCCH information
pcm_EFinf0_Type	information identifier (manufacturer)
pcm_EFals_Type	alternative line service

pcm_EFlocgprs_Type	location information(GPRS)
pcm_EFkcgprs_Type	ciphering key (GPRS)
pcm_EFimsigprs_Type	IMSI (GPRS)

2.1.1 pcm_FileInfo_Type – Elementary file information

Definition:

```
typedef struct pcm_FileInfo_Type
{
        UBYTE *                  FileLocation
        USHORT                   FileSize
        UBYTE                    Version
};
```

Description:

The file information type contains all available information about a transparent elementary file. The parameter FileLocation contains the pointer to the file located in the RAM area. The parameter FileSize stores the size of the file in bytes. The Version parameter contains the current version of the file and can be used for verification.

2.1.2 pcm_EFmscap_Type - Mobile capabilities

Definition:

```
#define SIZE_EF_MSCAP 6
typedef struct pcm_EFmscap_Type
{
      UBYTE          chnMode;  /* channel modes     */
      UBYTE          datCap1;  /* data capabilities */
      UBYTE          datCap2;  /* data capabilities */
      UBYTE          featLst1; /* feature list      */
      UBYTE          featLst2; /* feature list      */
      UBYTE          featLst3; /* feature list      */
} EF_MSCAP;
```

Description:

This record type is used by G23.

This EF provides the mobile capabilities for the protocol software.

Identifier: "MSCAP"	**Structure: transparent**	**Mandatory**	
File size: SIZE_EF_MSCAP +2 bytes	**Access: R/-**		
Bytes	**Description**	**M/O**	**Length**
1	checksum	M	1 byte
2	version	M	1 byte
3	channel modes	M	1 byte
4 until 5	data capabilities	M	2 bytes
6 until 8	feature list	M	3 bytes

- **channel modes**

  This defines the channel modes supported by the mobile station:

Octet	8	7	6	5	4	3	2	1
1	L1	Tm-	afs	ahs	spV3	efrV2	hr	spV1

	Value	Description
spV1	0	Speech not supported
	1	Speech supported
Hr	0	Half-rate not supported
	1	Half-rate supported
efrV2	0	Speech not supported (version 2, enhanced full-rate)
	1	Speech supported (version 2, enhanced full-rate)
spV3	0	Speech not supported (version 3, enhanced full-rate)

	1	Speech supported (version 3, enhanced full-rate)
ahs	0	Speech not supported (AMR, half rate speech)
	1	Speech supported (AMR, half rate speech)
afs	0	Speech not supported (AMR, full rate speech)
	1	Speech supported (AMR, full rate speech)
Tm	0	Normal Version
	1	Test Mobile Version
L1	0	Voice & SMS Layer 1
	1	Voice & SMS & Fax & Data Layer 1

- **data capabilities 1**

  This defines the data capabilities supported by the mobile station. The second byte is reserved for future use.

Octet	8	7	6	5	4	3	2	1
1	14.4	tfax	ntfax	tsyn	syn	asyn	rlp	ds

	Value	Description
Ds	0	data not supported
	1	data supported
rlp	0	radio link protocol not supported
	1	radio link protocol supported
asyn	0	asynchronous data service not supported
	1	asynchronous data service supported
syn	0	NT synchronous data service not supported
	1	NT synchronous data service supported
tsyn	0	T synchronous data service not supported
	1	T synchronous data service supported
ntfax	0	NT facsimile service not supported
	1	NT facsimile service not supported
tfax	0	T facsimile service not supported
	1	T facsimile service not supported
14.4	0	MS does not support 14.4 kBaud
	1	MS supports 14.4 kBaud

- **data capabilities 2**

  This defines the data capabilities supported by the mobile station. The second byte is reserved for future use.

Octet	8	7	6	5	4	3	2	1
1	DHR			NAS	TPD	NTPD	TP	NTP

	Value	Description
NTP	0	NT packet service not supported
	1	NT packet service supported
TP	0	T packet service not supported
	1	T packet service supported
NTPD	0	NT PAD access asynchronous service not supported
	1	NT PAD access asynchronous service supported
TPD	0	T PAD access asynchronous service not supported
	1	T PAD access asynchronous service supported
NAS	0	No Alternate Services Off (alternate serv.

		allowed)
	1	No Alternate Services On (alternate serv. forbidden)
DHR	0	Data half rate not supported
	1	Data half rate supported

- **feature list**
  This field is used for future enhancements.

2.1.3 pcm_EFimei_Type – IMEI and IMEISV

Definition:

```
#define SIZE_EF_IMEI 8
typedef struct pcm_EFimei_Type
{
    UBYTE          tac1;
    UBYTE          tac2;
    UBYTE          tac3;
    UBYTE          fac;
    UBYTE          snr1;
    UBYTE          snr2;
    UBYTE          snr3;
    UBYTE          svn;
} EF_IMEI;
```

Description:

This record type is used by G23.

This EF provides the IMEI and IMEISV.

Identifier: "IMEI"		**Structure: transparent**		**Mandatory**
File size: SIZE_EF_IMEI +2 bytes		**Access: R/- (encrypted)**		
Bytes	**Description**		**M/ O**	**Length**
1	checksum		M	1 byte
2	version		M	1 byte
3 until 10	IMEI		M	8 bytes

The IMEI has 15 digits. The IMEISV has 16 digits and differs from the IMEI in the last digit only.

Octet	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**
1	MSBD 2	..	..	LSBD2	MSBD 1	..	..	LSBD 1
2	MSBD 4	..	..	LSBD4	MSBD 3	..	..	LSBD 3
..								

2.1.4 pcm_EFimsi_Type – IMSI

Definition:

```
#define SIZE_EF_IMSI 9
typedef struct pcm_EFimsi_Type
{
        UBYTE              len;
        UBYTE              IMSI[8];
} EF_IMSI;
```

Description:

This record type is used by G23.

This EF provides the IMSI.

Identifier: "IMSI"	Structure: transparent		Mandatory
File size: SIZE_EF_IMSI +2 bytes	Access: R/W		
Bytes	**Description**	**M/ O**	**Length**
1	checksum	M	1 byte
2	version	M	1 byte
3	len IMSI	M	1 bytes
4 until 11	IMSI	M	8 bytes

The IMSI has 8 digits.

Octet	8	7	6	5	4	3	2	1
1	MSBD 2	..	..	LSBD2	MSBD 1	..	..	LSBD 1
2	MSBD 4	..	..	LSBD4	MSBD 3	..	..	LSBD 3
..								

2.1.5 pcm_EFsms_Type – Short Message Service

Definition:

```
#define SIZE_EF_SMS 176
typedef struct pcm_EFsms_Type
{
        UBYTE           stat;
        UBYTE           rmd[175];
} EF_SMS;
```

Description:

This record type is used by G23.

This EF provides a storing area for short messages. It contains information in accordance with TS GSM 03.40 comprised of short messages (and associated parameters) which have either been received by the MS from the network or are to be used as an MS originated message.

Identifier: "SMS"		**Structure: linear fixed**	
File size: 2 + SIZE_EF_SMS bytes per record		**Access: R/W**	
Bytes	**Description**	**M/O**	**Length**
1	Checksum	M	1 byte
2	Version	M	1 byte
3	Status	M	1 byte
4 until 178	Remainder	M	175 bytes

- **status**

  Status byte of the record.

Octet	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**
1	reserved					status		

	Value	**Description**
status	0	data not supported
	1	mobile terminated short message; message read
	3	mobile terminated short message; message to be read
	5	Mobile originated short message; message sent
	7	Mobile originated short message; message to be sent

- **remainder**

The remainder contains the TS Service Center Address as specified in TS GSM 04.11 and the short message TPDU as specified in TS GSM 03.40, with identical coding and ordering parameters.

Any TP message reference contained in an MS originated message stored in the memory, should have a value as follows:

	Value of the TP message reference
message to be sent	0xFF
message sent to the network	TP Message Reference used in the message sent to the network

Any bytes in the record following the TPDU are filled with 'FF'.

It is possible for a TS Service Center Address of maximum permitted length, e.g. containing more than 18 address digits, to be associated with a maximum length TPDU, such that their combined length is 176 bytes. In this case, the ME stores the TS Service Center Address and the TPDU in the memory in bytes 2-176 without modification, except for the last byte of the TPDU, which is not stored.

2.1.6 pcm_EFclass2_Type - Mobile Station Classmark 2

Definition:

```
#define SIZE_EF_CLASS2  3
typedef struct pcm_EFclass2_Type
{
    UBYTE          byte1;
    UBYTE          byte2;
    UBYTE          byte3;
} EF_CLASS2;
```

Description:

This record type is used by G23.

This EF provides the mobile station classmark 2 parameters.

Identifier: "CLASS2"	Structure: transparent		Mandatory
File size: SIZE_EF_CLASS2+2 bytes	Access: R/-		
Bytes	**Description**	**M/O**	**Length**
1	Checksum	M	1 byte
2	Version	M	1 byte
3	class2 byte 1	M	1 byte
4	class2 byte 2	M	1 byte
5	class2 byte 3	M	1 byte

- **class 2, byte 1**

  This byte defines the mobile station classmark 2 parameter (byte 1) analogous with GSM 4.08.

Octet	8	7	6	5	4	3	2	1
1	-	rev		es	a5/1	rfpwr		

	Value	Description
rfpwr	0	class 1 (GSM 900), class 1 (DCS 1800)
	1	class 2 (GSM 900), class 2 (DCS 1800)
	2	class 3 (GSM 900), class 3 (DCS 1800)
	3	class 4 (GSM 900)
	4	class 5 (GSM 900)
a5/1	0	encryption algorithm A5/1 available
	1	encryption algorithm A5/1 not available
es	0	"Controlled Early Classmark Sending" option is not implemented
	1	"Controlled Early Classmark Sending" option is implemented
rev	0	Reserved for phase 1
	1	Used by phase 2 MSs

- **class 2, byte 2**

  This byte defines the mobile station classmark 2 parameter (byte 2) analogous with GSM 4.08.

Octet	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**
1	-	ps	ss		sm	reserved		frq

	Value	Description
frq	0	no support of E-GSM (GSM 900), reserved (DCS 1800)
	1	support of E-GSM (GSM 900)
sm	0	MS does not support MT-PP SMS
	1	MS supports MT-PP SMS
ss	0-3	defined in GSM 04.80
ps	0	PS capability not present
	1	PS capability present

- **class 2, byte 3**

  This byte defines the mobile station classmark 2 parameter (byte 3) analogous with GSM 4.08.

Octet	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**
1	CM3	reserved				cmsp	A5/3	A5/2

	Value	Description
A5/2	0	encryption algorithm A5/2 not available
	1	encryption algorithm A5/2 available
A5/3	0	encryption algorithm A5/3 not available
	1	encryption algorithm A5/3 available
cmsp	0	CM SERVICE PROMPT not supported
	1	CM SERVICE PROMPT supported
CM3	0	No additional MS capability information available
	1	Additional MS capabilities are described in the Classmark 3 info element

2.1.7 pcm_EFclass3_Type - Mobile Station Classmark 3

Definition:

```
#define SIZE_EF_CLASS3  3
typedef struct pcm_EFclass3_Type
{
        UBYTE           byte1;
        UBYTE           byte2;
        UBYTE           byte3;
} EF_CLASS3;
```

Description:

This EF provides the mobile station classmark 3 parameters.

Identifier: "CLASS3"	**Structure: transparent**		**Mandatory**
File size: SIZE_EF_CLASS3+2 bytes	**Access: R/-**		
Bytes	**Description**	**M/O**	**Length**
1	checksum	M	1 byte
2	version	M	1 byte
3	class3 byte 1	M	1 byte
4	class3 byte 2	M	1 byte
5	class3 byte 3	M	1 byte

- **class 3, byte 1**

  This byte defines the mobile station classmark 3 parameter (byte 1) analogous with GSM 4.08.

Octet	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**
1	-	bnd 3	bnd 2	bnd 1	a5/7	a5/6	a5/5	a5/4

	Value	**Description**
A5/4	0	encryption algorithm A5/4 not available
	1	encryption algorithm A5/4 available
A5/5	0	encryption algorithm A5/5 not available
	1	encryption algorithm A5/5 available
a5/6	0	encryption algorithm A5/6 not available
	1	encryption algorithm A5/6 available
a5/7	0	encryption algorithm A5/7 not available
	1	encryption algorithm A5/7 available
bnd1	0	P-GSM not supported
	1	P-GSM supported
bnd2	0	E-GSM not supported
	1	E-GSM supported
bnd3	0	DCS 1800 not supported
	1	DCS 1800 supported

- **class 3, byte 2**

  This byte defines the mobile station classmark 3 parameter (byte 2) analogous with GSM 4.08.

Octet	8	7	6	5	4	3	2	1
1	rfcap2				rfcap1			

	Value	Description
rfcap1		see power definitions
rfcap2		see power definitions

- **class 3, byte 3**

  This byte defines the mobile station classmark 3 parameter (byte 3) analogous with GSM 4.08, specific for support of Extended Measurement.

Octet	8	7	6	5	4	3	2	1
1	0	0	0	xm	0	0	0	0

	Value	Description
xm	0	Extended Measurement not supported
xm	1	Extended Measurement supported

2.1.8 pcm_EFrfcap_Type - Mobile Station RF Capability

Definition:

```
#define SIZE_EF_RFCAP
typedef struct pcm_EFrfcap_Type
{
    UBYTE       setbands;
    UBYTE       bands;
    UBYTE       power1;
    UBYTE       power2;
    UBYTE       power3;
    UBYTE       msGSM;
    UBYTE       msEDGE;
    UBYTE       msHSCSD;
    UBYTE       msGPRS;
    UBYTE       msECSD;
    UBYTE       msEGPRS;
    UBYTE       capability1;
    UBYTE       capability2;
    UBYTE       switchmeasure;
    UBYTE       encryption;
    UBYTE       positioning;
} EF_RFCAP;
```

Description:

This EF provides the mobile station RF capability (set and supported frequency bands and its radio power classes) and classmark related information.

NOTE: Presence of this field in non-volatile memory is a prerequisite for any operation of the MS!

Identifier: "RFCAP"	**Structure: transparent**		**Mandatory**
File size: SIZE_EF_RFCAP+2 bytes	**Access: R/-**		
Bytes	**Description**	**M/O**	**Length**
1	checksum	M	1 byte
2	version	M	1 byte
3	set bands (set frequency bands)	M	1 byte
4	bands (supported frequency bands)	M	1 byte
5	power 1 (power classes of GSM900 and DCS1800)	M	1 byte
6	power 2 (power classes of PCS1900 and GSM850)	M	1 byte
7	power 3 (power classes of GSM400 and EGDE)	M	1 byte
8	msGSM (GSM multi slot capability and classes)	M	1 byte
9	msEDGE (EDGE multi slot capability and classes)	M	1 byte
10	msHSCSD (HSCSD multi slot capability and	M	1 byte

	classes)		
11	msGPRS (GPRS multi slot capability and classes)	M	1 byte
12	msECSD (ECSD multi slot capability and classes)	M	1 byte
13	msEGPRS (EGPRS multi slot capability and classes)	M	1 byte
14	capability 1 (divers capabilities and options)	M	1 byte
15	capability 2 (divers capabilities and options)	M	1 byte
16	switch measure values	M	1 byte
17	encryption (A5/n encryption algorithm availability)	M	1 byte
18	positioning (supported positioning methods)	M	1 byte

- **RF capabilities, set bands**

This byte defines the frequency bands of the mobile station set by AT command %BAND. It reflects the setting of the user which allows to switch between automatic band selection, i.e. the MS will scan all bands which are supported by the hardware as indicated in the next field "bands", and manual band selection, where only those bands are scanned which the user allows (and which are supported by the hardware). For automatic mode, for this octet a value of 0b00000000 is used. For manual mode each frequency band which shall be scanned (if the hardware supports) shall have the corresponding bit set.
NOTE: As the field is used to indicate the user's choice, the customer of Condat shall ensure that a value of 0 is set during production time.

Octet	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**
1	R-GSM	GSM 480	GSM 450	GSM 850	E-GSM	PCS1900	DCS1800	GSM 900

	Value	**Description**
GSM 900	0	GSM 900 shall not be scanned
	1	GSM 900 shall be scanned
DCS 1800	0 1	DCS 1800 shall not be scanned DCS 1800 shall be scanned
PCS 1900	0	PCS 1900 shall not be scanned
	1	PCS 1900 shall be scanned
E-GSM	0	E-GSM shall not be scanned
	1	E-GSM shall be scanned (includes GSM 900)
GSM 850	0	GSM 850 shall not be scanned
	1	GSM 850 shall be scanned
GSM 450	0	GSM 450 shall not be scanned
	1	GSM 450 shall be scanned
GSM 480	0	GSM 480 shall not be scanned
	1	GSM 480 shall be scanned
R-GSM	0	Railway-GSM shall not be scanned
	1	Railway-GSM shall be scanned (includes GSM 900 and E-GSM)

- **RF capabilities, bands**

  This byte defines the supported frequency bands of the (hardware part of the) mobile station.
  NOTE1: If the E-GSM bit is set, the Condat software will behave as if the GSM 900 bit was set, irrespectively of the value of the GSM 900 bit.

Octet	8	7	6	5	4	3	2	1
2	R-GSM	GSM 480	GSM 450	GSM 850	E-GSM	PCS1900	DCS1800	GSM 900

	Value	Description
GSM 900	0	GSM 900 is not supported
	1	GSM 900 is supported
DCS 1800	0	DCS 1800 is not supported
	1	DCS 1800 is supported
PCS 1900	0	PCS 1900 is not supported
	1	PCS 1900 is supported
E-GSM	0	E-GSM is not supported
	1	E-GSM is supported (includes GSM 900)
GSM 850	0	GSM 850 is not supported
	1	GSM 850 is supported
GSM 450	0	GSM 450 is not supported
	1	GSM 450 is supported
GSM 480	0	GSM 480 is not supported
	1	GSM 480 is supported
R-GSM	0	Railway-GSM is not supported
	1	Railway-GSM is supported (includes GSM 900 and E-GSM)

- **RF capabilities, power 1**

  This byte defines the power classes for the supported frequency bands of the mobile station.

Octet	8	7	6	5	4	3	2	1
3	Power Class GSM 900				Power Class DCS 1800			

	Value	Description
Power Class GSM 900	0	GSM 900 is not supported
	1	Power Class 1
	2	Power Class 2
	3	Power Class 3
	4	Power Class 4
	5	Power Class 5
	> 5	reserved
Power Class DCS 1800	0	DCS 1800 is not supported
	1	Power Class 1

	2	Power Class 2
	3	Power Class 3
	> 3	reserved

- **RF capabilities, power 2**

  This byte defines the power classes for the supported frequency bands of the mobile station.

Octet	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**
4	Power Class PCS 1900				Power Class GSM 850			

	Value	**Description**
Power Class PCS 1900	0	PCS 1900 is not supported
	1	Power Class 1
	2	Power Class 2
	3	Power Class 3
	> 3	reserved
Power Class GSM 850	0	GSM 850 is not supported
	1	Power Class 1
	2	Power Class 2
	3	Power Class 3
	4	Power Class 4
	5	Power Class 5
	> 5	reserved

- **RF capabilities, power 3**

  This byte defines the power classes for the supported frequency bands of the mobile station.

Octet	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**
5	Power Class GSM 400				EDGE Power Class 1		EDGE Power Class 2	

	Value	**Description**
Power Class GSM 400	0	Neither GSM 450 nor GSM 480 are supported
	1	Power Class 1
	2	Power Class 2
	3	Power Class 3
	4	Power Class 4
	5	Power Class 5
	> 5	reserved
EDGE Power Class 1	0	No EGDE RF Power Capability 1
	1	Power Class E1

	2	Power Class E2
	3	Power Class E3
EDGE Power Class 2	0	No EGDE RF Power Capability 2
	1	Power Class E1
	2	Power Class E2
	3	Power Class E3

- **RF capabilities, msGSM**

  This byte defines the multi slot classes defined in TS GSM 05.02.

Octet	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**
6	ms_class					0	0	0

	Value	**Description**
ms_class	0	MS does not support the use of multiple timeslot
	1	Multi Slot Class 1
	2	Multi Slot Class 2
	3	Multi Slot Class 3
	…	…
	29	Multi Slot Class 29
	> 29	reserved
	0	reserved
	0	reserved
	0	reserved

- **RF capabilities, msEDGE**

  This byte defines the multi slot classes defined in TS GSM 05.02.

Octet	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**

7	egde_ms_class					0	0	0

	Value	Description
edge_ms_class	0	EDGE MS does not support the use of multiple timeslot
	1	Multi Slot Class 1
	2	Multi Slot Class 2
	3	Multi Slot Class 3
	…	…
	29	Multi Slot Class 29
	> 29	reserved
	0	reserved
	0	reserved
	0	reserved

- **RF capabilities, msHSCSD**

  This byte defines the multi slot classes defined in TS GSM 05.02. For HSCSD only multi slot classes 1 to 18 are recognized. For HSCSD applications the MS shall indicate a suitable multi slot class less than 19.

Octet	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**
8	hscsd_ms_class					0	0	0

	Value	Description
hscsd_ms_class	0	HSCSD MS does not support the use of multiple timeslot
	1	Multi Slot Class 1
	2	Multi Slot Class 2
	3	Multi Slot Class 3
	…	…
	18	Multi Slot Class 18
	> 18	reserved
	0	reserved
	0	reserved
	0	reserved

- **RF capabilities, msGPRS**

  This byte defines the multi slot classes defined in TS GSM 05.02.

Octet	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**

9	gprs_ms_class	dtm_g	dtm_g_ms_class

	Value	Description
gprs_ms_class	0	GPRS MS does not support the use of multiple timeslot
	1	Multi Slot Class 1
	2	Multi Slot Class 2
	3	Multi Slot Class 3
	…	…
	29	Multi Slot Class 29
	> 29	reserved
dtm_g	0	GPRS MS does not support GPRS Dual Transfer Mode (DTM)
	1	GPRS MS supports GPRS Dual Transfer Mode (DTM)
dtm_g_ms_class	0	Sub-Class 1 supported
	1	Sub-Class 5 supported
	2	Sub-Class 9 supported
	3	reserved

- **RF capabilities, msECSD**

  This byte defines the multi slot classes defined in TS GSM 05.02 for an enhanced circuit-switched channels (ECSD) MS.

Octet	8	7	6	5	4	3	2	1
10	ecsd_ms_class					0	0	0

	Value	Description
ecsd_ms_class	0	ECSD MS does not support the use of multiple timeslot
	1	Multi Slot Class 1
	2	Multi Slot Class 2
	3	Multi Slot Class 3
	…	…
	29	Multi Slot Class 29
	> 29	reserved
	0	reserved
	0	reserved
	0	reserved

- **RF capabilities, msEGPRS**

  This byte defines the multi slot classes defined in TS GSM 05.02 for an enhanced GPRS MS.

Octet	8	7	6	5	4	3	2	1

11	egprs_ms_class	dtm_e	dtm_e_ms_class

	Value	Description
egprs_ms_class	0	EGPRS MS does not support the use of multiple timeslot
	1	Multi Slot Class 1
	2	Multi Slot Class 2
	3	Multi Slot Class 3
	…	…
	29	Multi Slot Class 29
	> 29	reserved
dtm_e	0	EGPRS MS does not supports EGPRS Dual Transfer Mode (DTM)
	1	EGPRS MS supports EGPRS Dual Transfer Mode (DTM)
dtm_e_ms_class	0	Sub-Class 1 supported
	1	Sub-Class 5 supported
	2	Sub-Class 9 supported
	3	reserved

- **RF capabilities, capability 1**

  This byte defines several capabilities of the MS.

Octet	8	7	6	5	4	3	2	1
12	es_ind	ps	mt_pp_sms	lcsva	solsa	cmsp	mod	mac_support

	Value	Description
es_ind	0	"Controlled Early Classmark Sending" is implemented (1) or not (0)
	1	
ps	0	Pseudo Synchronisation capability is present (1) or not present (0)
	1	
mt_pp_sms	0	MS does support (1) mobile terminated point to point SMS or not (0)
	1	
lcsva	0	LCS value added location request notification supported (1) or not (0)
	1	
solsa	0	MS does supports SoLSA (1) or not (0)
	1	
cmsp	0	MS does supports (1) CM service Prompt (network initiated MO CM connection request) or not (0)
	1	
mod	0	The EDGE Modulation Capability indicates the supported modulation scheme by MS in addition to GMSK. 8-PSK is supported for downlink reception only (0) or for uplink transmission and downlink reception (1)
	1	
mac_support	0	MS supports Dynamic and Fixed Allocation (1) or only supports Exclusive Allocation (0)
	1	

- **RF capabilities, capability 2**

This byte defines several capabilities of the MS.

Octet	8	7	6	5	4	3	2	1
13	meas	ext_meas	compact	vbs	vgcs	ucs2_treat	ss_screen	

	Value	Description
meas	0 1	Indicates whether an IE, e.g. classmark 3, shall contain any value (see sms_value and sm_value below in the next byte) about the measurement capabilities (1) or not (0).
ext_meas	0 1	The MS supports "Extended Measurement (on SACCH)" (1) or not (0)
compact	0 1	The MS supports COMPACT Interference Measurement (1) or not (0)
vbs	0 1	VBS capability and notifications wanted (1) or vice versa (0)
vgcs	0 1	VGCS capability and notifications wanted (1) or vice versa (0)
ucs2_treat	0 1	Indicates the likely treatment by the MS of UCS2 encoded character strings. The ME has a preference for the default alphabet over UCS2 (0) or the ME has no preference between the use of the default alphabet and the use of UCS2 (1).
ss_screen	0	Default value of phase 1
	1	Capability of handling of ellipsis notation and phase 2 error handling
	2	for future use (The network shall interpret these value the same as 1)
	3	for future use (The network shall interpret these value the same as 1)

- **RF capabilities, switch measure**

  This byte defines the SMS and SM values for switching time. The SMS field indicates the time for the MS to switch from one radio channel to another, perform a neighbour cell power measurement, and the switch from that radio channel to another radio channel. The SM field indicates the time needed for the MS to switch from one radio channel to another and perform a neighbour cell power measurement

  NOTE: If in the previous octet "capability 2", for bit "meas" the value 0 is set, then any value can be chosen for both sms_value and sm_value as it won't be relevant as over the air interface no information about sms_value and sm_value will be sent. However, presence of this octet 14 (in the FFS) is required in any case.

Octet	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**
14	sms_value				sm_value			

	Value	**Description**
sms_value	0	1/4 timeslot (~144 ms)
	1	2/4 timeslot (~288 ms)
	2	3/4 timeslot (~433 ms)
	3	4/4 timeslot
	…	
	14	15/4 timeslot
	15	16/4 timeslot (~2307 ms)
sm_value	0..15	the same values as sms_value

- **RF capabilities, encryption**

  This byte defines the availability (i.e. support) of the A5/n encryption algorithm.

  NOTE: Please note, that the indication of availability is here always indicated via value 1 although the encoding over the air interface may be partially (concrete for A5/1) different.

Octet	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**
15	A5/1	A5/2	A5/3	A5/4	A5/5	A5/6	A5/7	0

	Value	**Description**
A5/1	0	Encryption algorithm A5/1 available (1) or not (0)
	1	
A5/2	0 1	Encryption algorithm A5/2 available (1) or not (0)
A5/3	0	Encryption algorithm A5/3 available (1) or not (0)
	1	
A5/4	0	Encryption algorithm A5/4 available (1) or not (0)
	1	
A5/5	0	Encryption algorithm A5/5 available (1) or not (0)
	1	
A5/6	0	Encryption algorithm A5/6 available (1) or not (0)

	1	
A5/7	0 1	Encryption algorithm A5/7 available (1) or not (0)
	0	reserved

- **RF capabilities, positioning and other things**

  This byte defines the availability and using of positioning methods as well as indication of support for extended dynamic allocation capabilities.

Octet	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**
16	assist eotd	based eotd	assist gps	based gps	conv gps	gprs_eda	egprs_eda	0

	Value	**Description**
assist_eotd	0 1	MS does support assisted E-OTD as positioning method (1) or not (0)
based_eotd	0 1	MS does support based E-OTD as positioning method (1) or not (0)
assist_gps	0 1	MS does support assisted GPS as positioning method (1) or not (0)
based_gps	0 1	MS does support based GPS as positioning method (1) or not (0)
conv_gps	0 1	MS does support conventional GPS as positioning method (1) or not (0)
gprs_eda	0	reserved for future use for GPRS Extended Dynamic Allocation Capability
egprs_eda	0	reserved for future use for EGPRS Extended Dynamic Allocation Capability
	0	reserved

2.1.9 pcm_EFmssup_Type – Mobile Setup

Definition:

```
#define SIZE_EF_MSSUP 5
typedef struct pcm_EFmssup_Type
{
    UBYTE           lng1;
    UBYTE           lng2;
    UBYTE           lng3;
    UBYTE           feat1;
    UBYTE           feat2;
} EF_MSSUP;
```

Description:

This record type is used by G23.

This EF provides the mobile station setup parameters and factory defaults.

Identifier: "MSSUP"	**Structure: transparent**	**Mandatory**
File size: SIZE_EF_MSSUP +2 bytes	**Access: R/-**	

Bytes	**Description**	**M/O**	**Length**
1	checksum	M	1 byte
2	version	M	1 byte
3	language 1	M	1 byte
4	language 2	M	1 byte
5	language 3	M	1 byte
6	features byte 1	M	1 byte
7	features byte 2	M	1 byte

- **language 1**

  The bits of these bytes define the support for different languages. The total number of languages and therefore the number of language bytes has to be defined.

Octet	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**
1	por	swe	spa	ita	dut	ger	fre	eng

	Value	**Description**
eng	0	English is not supported
	1	English is supported
fre	0	French is not supported
	1	French is supported
ger	0	German is not supported
	1	German is supported
dut	0	Dutch is not supported
	1	Dutch is supported

ita	0	Italian is not supported
	1	Italian is supported
spa	0	Spanish is not supported
	1	Spanish is supported
swe	0	Swedish is not supported
	1	Swedish is supported
por	0	Portuguese is not supported
	1	Portuguese is supported

- **language 2**

Octet	8	7	6	5	4	3	2	1
1	rus	pol	slo	hun	tur	gre	nor	fin

	Value	Description
fin	0	Finnish is not supported
	1	Finnish is supported
nor	0	Norwegian is not supported
	1	Norwegian is supported
gre	0	Greek is not supported
	1	Greek is supported
tur	0	Turkish is not supported
	1	Turkish is supported
hun	0	Hungarian is not supported
	1	Hungarian is supported
slo	0	Slovenian is not supported
	1	Slovenian is supported
pol	0	Polish is not supported
	1	Polish is supported
rus	0	Russian is not supported
	1	Russian is supported

- **language 3**

Octet	8	7	6	5	4	3	2	1
1	-	ara	tai	man	can	chi	cze	ind

	Value	Description
ind	0	Indonesian is not supported
	1	Indonesian is supported
cze	0	Czech is not supported
	1	Czech is supported
chi	0	Chinese is not supported
	1	Chinese is supported
can	0	Cantonese is not supported
	1	Cantonese is supported
man	0	Mandarin is not supported
	1	Mandarin is supported
tai	0	Taiwanese is not supported
	1	Taiwanese is supported
ara	0	Arabic is not supported
	1	Arabic is supported

- **feature byte 1**

  The bits of these bytes define the support for different features. The total number of features and therefore the number of feature bytes must be defined.

  NOTE: With the exception of the "aoc" field which is used by the ACI G23 makes no use of the information contained in this byte anymore. MFW reads "aoc", "dtmf", "cf", "cb", "ussd" and "etc" into an internal data structure, but aside from this it also has no further use for this information. Note especially that for some of these features the support in the protocol stack is *always* present (like e.g. for Explicit Call Transfer).

Octet	8	7	6	5	4	3	2	1
1	stk	irda	etc	uss d	cb	cf	dtm f	aoc

	Value	Description
aoc	0	Advice of charge not supported
	1	Advice of charge supported
dtmf	0	DTMF not supported
	1	DTMF supported
cf	0	call forwarding not supported
	1	call forwarding supported
cb	0	call barring not supported
	1	call barring supported
ussd	0	unstructured SS data not supported
	1	unstructured SS data supported
etc	0	ECT (Explicit Call Transfer) not supported
	1	ECT (Explicit Call Transfer) supported
irda	0	IRDA not supported
	1	IRDA 7 supported
stk	0	SIM Toolkit not supported
	1	SIM Toolkit supported

- **feature byte 2**

  The bits of these bytes define the support for different features. The total number of features and therefore the number of feature bytes must be defined.

2.1.10 pcm_EFclng_Type – current language

Definition:

```
#define SIZE_EF_CLNG_DATA 2

typedef struct pcm_EFclng_Type
{
      UBYTE              data[SIZE_EF_CLNG_DATA];
} EF_CLNG;
#define SIZE_EF_CLNG SIZE_EF_CLNG_DATA
```

Description:

This record type is used by G23.

Identifier: "CLNG "	**Structure: transparent**	**Optional**	
File size: SIZE_EF_CLNG_DATA + **2bytes**		**Access: R/W**	
Bytes	**Description**	**M/O**	**Length**
1	checksum	M	1 byte
2	version	M	1 byte
3 until (SIZE_EF_CLNG_DATA+2)	data	M	SIZE_EF_CLNG_DATA bytes

- **data**

The EF contains the two-letter abbreviation of the current default or manually selected language of the mobile station.

pcm_EFmsset_Type – Mobile station settings

Definition:

```
#define SIZE_EF_MSSET 10
typedef struct pcm_EFmsset_Type
{
    UBYTE           buzzer1;
    UBYTE           buzzer2;
    UBYTE           buzzer3;
    UBYTE           audio;
    UBYTE           misc;
    UBYTE           display;
    UBYTE           language;
    UBYTE           recent_ldn_ref;
    UBYTE           recent_lrn_ref;
    UBYTE           recent_upn_ref;
}EF_MSSET;
```

Description:

This EF provides a storage area for user dependent settings (profiles) of the mobile station parameters.

Identifier: "MSSET"	**Structure: transparent**		**Mandatory**
File size: SIZE_EF_MSSET +2 bytes	**Access: R/W**		
Bytes	**Description**	**M/ O**	**Length**
1	checksum	M	1 byte
2	version	M	1 byte
3 until 5	buzzer	M	3 bytes
6	audio	M	1 byte
7	miscellaneous	M	1 byte
8	display	M	1 byte
9	language	M	1 byte
10	recent ldn reference	M	1 byte
11	recent lrn reference	M	1 byte
12	recent upn reference	M	1 byte

- **buzzer**

  This defines the user setting for buzzer including ringer type and ringer volume for calls and messages as well as the beep type for pressed keys.

Octet	8	7	6	5	4	3	2	1
1	vib		callvol			calltype		
2	reserved		msgvol			msgtype		
3	reserved					batw	keytone	

	Value	Description
calltype	0-7	ringer type 0-7 for incoming calls
callvol	0	quiet
	1-7	ringer volume for incoming calls
vib	0	vibrator inactive
	1	vibrator only
	2	vibrator then ring
	3	reserved
msgtype	0-7	ringer type 0-7 for incoming messages
msgvol	0	quiet
	1-7	ringer volume for incoming messages
keytone	0	key tones inactive
	1	beep
	2	dtmf
batw	0	low battery warning disabled
	1	low battery warning enabled

- **audio**

  The bits define the user settings for audio input and output volume.

Octet	8	7	6	5	4	3	2	1
1	VoiceRec	Ext	Outvol			Inamp		

	Value	Description
Inamp	0-7	amplification of microphone
Outvol	0-7	speaker output volume
Ext	0	external audio disabled
	1	external audio enabled
voiceRec	0	voice recorder disabled
	1	voice recorder enabled

- **miscellaneous**

  The bits define miscellaneous settings.

Octet	8	7	6	5	4	3	2	1
1			redial		calinf	clip	clir	pmod

	Value	Description
pmod	0	automatic PLMN selection
	1	manual PLMN selection
clir	0	CLIR not suppressed
	1	CLIR suppressed
clip	0	CLIP not suppressed
	1	CLIP suppressed
calinf	0	call information display off
	1	call information display on
redial	0	Redial off
	1	Automatic
	2	Manual

- **display**

  The bits define the user settings for the display.

Octet	8	7	6	5	4	3	2	1
1	bckdr			brgt		ctrt		

	Value	Description
ctrt	0-7	contrast
brgt	0-3	brightness
bckdr	0-7	duration for back light

- **language**

  The bits define the user language

Octet	8	7	6	5	4	3	2	1
1	language							

- **RecentLdnRef, RecentLrnRef, RecentUpnRef**

  These three information fields are used to store the reference to the record of the most recently used number. In the initial state, these fields should contain 0xFF, this means no reference available (no recently used number available).

Octet	8	7	6	5	4	3	2	1
1	Recent*xxx*Ref							

2.1.11 pcm_EFldn_Type – Last MOC numbers

Definition:

```
#define SIZE_EF_LDN 22
typedef struct pcm_EFldn_Type
{
    UBYTE          calDrMsb;
    UBYTE          calDrLsb;
    UBYTE          year;
    UBYTE          month;
    UBYTE          day;
    UBYTE          hour;
    UBYTE          minute;
    UBYTE          second;
    UBYTE          len;
    UBYTE          numTp;
    UBYTE          dldNum [10];
    UBYTE          ccp;
    UBYTE          ext1;
} EF_LDN;
```

Description:

This record type is used by ACI.

This EF provides a storage area for the called party bcd numbers of the last 10 mobile originated calls.

Identifier: "LDN"	**Structure: linear fixed**		**Optional**
File size: 2 + SIZE_EF_LDN bytes per record	**Access: R/W**		
Bytes	**Description**	**M/O**	**Length**
1	Checksum	M	1 byte
2	Version	M	1 byte
3 until 4	call duration	M	2 byte
5 until 10	date/time	M	6 byte
11	length of BCD number	M	1 byte
12	TON and NPI	M	1 byte
13 until 22	called number	M	10 bytes
23	capability/configuration identifier	M	1 byte
24	extension1 record identifier	M	1 byte

- **call duration**

  Contains a 16 bit integer value which represents the call duration in seconds.

Octet	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**
1	MSB							
2	LSB							

- **date/time**

  Contains a BCD coded value (2 digits) representing the date and time of the call.

Octet	8	7	6	5	4	3	2	1
1	Year							
2	Month							
3	Day							
4	Hour							
5	Minute							
6	Second							

- **length of Called Party BCD number**

  This byte stores the number of bytes of the following two data elements containing the called number and additional information. If the called number is longer that 20 digits, the remaining digits are stored in EF$_{EXT1}$ field which is referenced by the data element "extension1 identifier". If the number of digits of the called number is <= 20 the extension1 identifier is set to 'FF'.

- **TON and NPI**

  Type of number and numbering plan identification according to GSM 04.08.

Octet	8	7	6	5	4	3	2	1
1	-	ton			npi			

- **Called number**

  Contains the first 20 digits of the called number coded according to the called party bcd number of GSM 04.08.

Octet	8	7	6	5	4	3	2	1
1	MSBD 2	..	..	LSBD2	MSBD 1	..	..	LSBD 1
2	MSBD 4	..	..	LSBD4	MSBD 3	..	..	LSBD 3
..								...

- **Capability/configuration identifier**

  The capability/configuration identification byte is the reference to an EF$_{CCP}$ record containing call related capability/configuration parameters. If the this byte is unused it is set to 'FF'.

- **Extension1 record identifier**

  This byte is the reference to an EF$_{EXT1}$ record containing an associated called party sub-address or an overflow. Set this byte to 'FF' if neither a sub-address nor an overflow is indicated.

2.1.12 pcm_EFlrn_Type – Last MTC numbers

Definition:

```
#define SIZE_EF_LRN 23
typedef struct pcm_EFlrn_Type
{
    UBYTE           calDrMsb;
    UBYTE           calDrLsb;
    UBYTE           year;
    UBYTE           month;
    UBYTE           day;
    UBYTE           hour;
    UBYTE           minute;
    UBYTE           second;
    UBYTE           id;
    UBYTE           len;
    UBYTE           numTp;
    UBYTE           dldNum[10];
    UBYTE           ccp;
    UBYTE           ext1;
} EF_LRN;
```

Description:

This record type is used by ACI.

This EF contains a storage area for calling party bcd numbers of the last 10 mobile terminated calls.

Identifier: "LRN"	**Structure: linear fixed**		**Optional**
File size: 2 + SIZE_EF_LRN bytes per record	**Access: R/W**		
Bytes	**Description**	**M/O**	**Length**
1	Checksum	M	1 byte
2	Version	M	1 byte
3 until 4	Call duration	M	2 byte
5 until 10	Date/time	M	6 byte
11	Identifier	M	1 byte
12	length of BCD number	M	1 byte
13	TON and NPI	M	1 byte
14 until 23	calling number	M	10 bytes
24	capability/configuration identifier	M	1 byte
25	extension1 record identifier	M	1 byte

- **call duration**

  Contains a 16 bit integer value which represents the call duration in seconds.

Octet	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**
1	MSB							

2	LSB

- **date/time**

  Contains a BCD coded value (2 digits) representing the date and time of the call.

Octet	8	7	6	5	4	3	2	1
1	Year							
2	Month							
3	Day							
4	Hour							
5	Minute							
6	Second							

- **identifier**

  This identifies the type of entry.

Octet	8	7	6	5	4	3	2	1
1	reserved					type		

	Value	Description
type	0	unknown
	1	call accepted
	2	call rejected

- **length of Called Party BCD number**

  This byte stores the number of bytes of the following two data elements containing the called number and additional information. If the called number is longer that 20 digits the remaining, digits are sorted in EF_{EXT1} field which is referenced by the data element “extension1 identifier”. If the number of digits of the called number is <= 20, the extension1 identifier is set to 'FF'.

- **TON and NPI**

  Type of number and numbering plan identification according to GSM 04.08.

Octet	8	7	6	5	4	3	2	1
1	-	ton			npi			

- **Called number**

  Contains the first 20 digits of the called number coded according to the called party bcd number of GSM 04.08.

Octet	8	7	6	5	4	3	2	1
1	MSBD 2	..	..	LSBD2	MSBD 1	..	..	LSBD 1
2	MSBD 4	..	..	LSBD4	MSBD 3	..	..	LSBD 3
..								...

- **Capability/configuration identifier**

  The capability/configuration identification byte is the reference to an EF_{CCP} record containing call related capability/configuration parameters. If the this byte is unused it is set to 'FF'.

- **Extension1 record identifier**

  This byte is the reference to an EF_{EXT1} record containing an associated called party sub-address or an overflow. Set this byte to 'FF' if neither a sub-address nor an overflow is indicated.

2.1.13 pcm_EFlmn_Type – Last MTC missed numbers

Definition:

```
#define SIZE_EF_LMN 21
typedef struct pcm_EFlmn_Type
{
    UBYTE           year;
    UBYTE           month;
    UBYTE           day;
    UBYTE           hour;
    UBYTE           minute;
    UBYTE           second;
    UBYTE           id;
    UBYTE           len;
    UBYTE           numTp;
    UBYTE           dldNum[10];
    UBYTE           ccp;
    UBYTE           ext1;
} EF_LMN;
```

Description:

This record type is used by ACI.

This EF contains a storage area for calling party bcd numbers of the last 10 missed mobile terminated calls.

Identifier: "LMN"	**Structure: linear fixed**		**Optional**
File size: 2 + SIZE_EF_LMN bytes per record	**Access: R/W**		
Bytes	**Description**	**M/O**	**Length**
1	Checksum	M	1 byte
2	Version	M	1 byte
3 until 8	Date/time	M	6 byte
9	Identifier	M	1 byte
10	Length of BCD number	M	1 byte
11	TON and NPI	M	1 byte
12 until 21	calling number	M	10 bytes
22	capability/configuration identifier	M	1 byte
23	extension1 record identifier	M	1 byte

- **date/time**

  Contains a BCD coded value (2 digits) representing the date and time of the call.

Octet	8	7	6	5	4	3	2	1
1	Year							
2	Month							
3	Day							

4	Hour
5	Minute
6	Second

- **identifier**

  This identifies the type of entry.

Octet	8	7	6	5	4	3	2	1
1	reserved					type		

	Value	Description
type	0	unknown
	1	call accepted
	2	call rejected

- **length of Called Party BCD number**

  This byte stores the number of bytes of the following two data elements containing the called number and additional information. If the called number is longer that 20 digits the remaining, digits are sorted in EF$_{EXT1}$ field which is referenced by the data element “extension1 identifier”. If the number of digits of the called number is <= 20, the extension1 identifier is set to 'FF'.

- **TON and NPI**

  Type of number and numbering plan identification according to GSM 04.08.

Octet	8	7	6	5	4	3	2	1
1	-	ton			npi			

- **Called number**

  Contains the first 20 digits of the called number coded according to the called party bcd number of GSM 04.08.

Octet	8	7	6	5	4	3	2	1
1	MSBD 2	..	..	LSBD2	MSBD 1	..	..	LSBD 1
2	MSBD 4	..	..	LSBD4	MSBD 3	..	..	LSBD 3
..								...

- **Capability/configuration identifier**

  The capability/configuration identification byte is the reference to an EF$_{CCP}$ record containing call related capability/configuration parameters. If the this byte is unused it is set to 'FF'.

- **Extension1 record identifier**

  This byte is the reference to an EF$_{EXT1}$ record containing an associated called party sub-address or an overflow. Set this byte to 'FF' if neither a sub-address nor an overflow is indicated.

2.1.14 pcm_EFupn_Type – User personal numbers

Definition:

```
#define SIZE_EF_UPN 24
typedef struct pcm_EFupn_Type
{
     UBYTE          alphId[10];
     UBYTE          len;
     UBYTE          numTp
     UBYTE          usrNum[10];
     UBYTE          ccp;
     UBYTE          ext1;
} EF_UPN;
```

Description:

This record type is used by ACI.

This EF is used as a personal address book. It provides a storage area for x entries containing phone numbers and associated names. There is a fixed number of records for special numbers such as voice mail, SMS Service Center and the users own number. Additional user number can be defined.

<table>
<tr><th colspan="2">Identifier: "UPN"</th><th colspan="1">Structure: linear fixed</th><th colspan="1">Optional</th></tr>
<tr><th colspan="2">File size: 2 + SIZE_EF_UPN bytes per record</th><th colspan="2">Access: R/W</th></tr>
<tr><th>Bytes</th><th>Description</th><th>M/
O</th><th>Length</th></tr>
<tr><td>1</td><td>Checksum</td><td>M</td><td>1 byte</td></tr>
<tr><td>2</td><td>Version</td><td>M</td><td>1 byte</td></tr>
<tr><td>3 until 12</td><td>alpha identifier</td><td>M</td><td>10 bytes</td></tr>
<tr><td>13</td><td>length of BCD number</td><td>M</td><td>1 byte</td></tr>
<tr><td>14</td><td>TON and NPI</td><td>M</td><td>1 byte</td></tr>
<tr><td>15 until 24</td><td>number</td><td>M</td><td>10 bytes</td></tr>
<tr><td>25</td><td>capability/configuration identifier</td><td>M</td><td>1 byte</td></tr>
<tr><td>26</td><td>extension1 record identifier</td><td>M</td><td>1 byte</td></tr>
</table>

- **alpha identifier**

  This alpha tagging uses the SMS default 7-bit coded alphabet as defined in TS GSM 03.38 with bit 8 set to 0. The alpha identifier is left justified. Unused bytes are set to ‘FF’.

- **length of number**

  This byte stores the number of bytes of the following two data elements containing the called number and additional information. If the called number is longer that 20 digits, the remaining digits are stored in EF_{EXT1} field which is referenced by the data element “extension1 identifier”. If the number of digits of the called number is <= 20 the extension1 identifier is set to 'FF'.

- **TON and NPI**

  Type of number and numbering plan identification according to GSM 04.08.

Octet	8	7	6	5	4	3	2	1
1	-	ton			npi			

- **number**

  Contains the first 20 digits of the called number coded according to the called party bcd number of GSM 04.08.

Octet	8	7	6	5	4	3	2	1
1	MSBD 2	..	..	LSBD2	MSBD 1	..	..	LSBD 1
2	MSBD 4	..	..	LSBD4	MSBD 3	..	..	LSBD 3
..								...

- **Capability/configuration identifier**

  The capability/configuration identification byte is the reference to an EF_{CCP} record containing call related capability/configuration parameters. If the this byte is unused it is set to 'FF'.

- **Extension1 record identifier**

  This byte is the reference to an EF_{EXT1} record containing an associated called party sub-address or an overflow. Set this byte to 'FF' if neither a sub-address nor an overflow is indicated.

2.1.15 pcm_EFmbn_Type – mailbox numbers

Definition:

```
#define SIZE_EF_MBN 22
typedef struct pcm_EFmbn_Type
{
     UBYTE           alphId[10];
     UBYTE           len;
     UBYTE           numTp
     UBYTE           mbNum[10];
} EF_MBN;
```

Description:

This record type is used by MFW.

This EF contains a storage area for 4 mailbox numbers and associated names.

Note: This EF is not used currently.

Identifier: "MBN"	**Structure: linear fixed**		**Optional**
File size: 2 + SIZE_EF_MBN bytes per record	**Access: R/W**		
Bytes	**Description**	**M/O**	**Length**
1	Checksum	M	1 byte
2	Version	M	1 byte
3 until 12	alpha identifier	M	10 bytes
13	length of BCD number	M	1 byte
14	TON and NPI	M	1 byte
15 until 24	number	M	10 bytes

- **alpha identifier**

  This alpha tagging uses the SMS default 7-bit coded alphabet as defined in TS GSM 03.38 with bit 8 set to 0. The alpha identifier is left justified. Unused bytes are set to 'FF'.

- **length of number**

  This byte stores the number of bytes of the following two data elements containing the called number and additional information.

- **TON and NPI**

  Type of number and numbering plan identification according to GSM 04.08.

Octet	8	7	6	5	4	3	2	1
1	-	ton			npi			

- **number**

  Contains the first 20 digits of the called number coded according to the called party bcd number of GSM 04.08.

Octet	8	7	6	5	4	3	2	1
1	MSBD 2	..	..	LSBD2	MSBD 1	..	..	LSBD 1
2	MSBD 4	..	..	LSBD4	MSBD 3	..	..	LSBD 3
..								...

2.1.16 pcm_EFvmn_Type – Voice Mail Number

Definition:

```
#define SIZE_EF_VMN MAX_CALLED_PARTY_BCD_NO_OCTETS + 2
typedef struct pcm_EFvmn_Type
{
        UBYTE           vmNum[41];
        UBYTE           numTp;
} EF_VMN;
```

Description:

This record type is used by AT Command Interpreter.

This EF constitutes the voice mail server number and the type of number.

Identifier: "VMN"	**Structure: transparent**		**Optional**
File size: 2+ SIZE_EF_VMN MAX_CALLED_PARTY_BCD_NO_OCTETS + 2bytes	**Access: R/W**		
Bytes	**Description**	**M/O**	**Length**
1	checksum	M	1 byte
2	version	M	1 byte
3 until 43	Voice mail number	M	41 bytes
44	Type of number	M	1 byte

- **Voice mail number**

  The voice mail number coded according to the called party BCD number of GSM 04.08, with the addition that a delimiter 0xFF is used for the end of the array vmNum.

Octet	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**
1	MSBD 2	..	..	LSBD2	MSBD 1	..	..	LSBD 1
2	MSBD 4	..	..	LSBD4	MSBD 3	..	..	LSBD 3
…								…
n	1	1	1	1	1	1	1	1
…								

- **TON and NPI**

  Type of number and numbering plan identification according to GSM 04.08.

Octet	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**
1	-	ton			npi			

2.1.17 pcm_EFctim_Type – Call timer

Definition:

```
typedef struct pcm_EFctim_Type
{
        UBYTE           moVcDrHm [4];
        UBYTE           mtVcDrHm [4];
        UBYTE           moDtDrHm [4];
        UBYTE           mtDtDrHm [4];
        UBYTE           moFxDrHm [4];
        UBYTE           mtFxDrHm [4];
        UBYTE           moVcDrRm [4];
        UBYTE           mtVcDrRm [4];
        UBYTE           moDtDrRm [4];
        UBYTE           mtDtDrRm [4];
        UBYTE           mtFxDrRm [4];
        UBYTE           mtFxDrRm [4];
} EF_CTIM;
```

Description:

This EF provides call timers for different call types. Each timer contains a 32 bit integer value which represents the call duration in seconds. For example, byte 3 stores the most significant byte and byte 6 stores the least significant byte of the 32 bit value. The field “Total duration” contains the sum of all call duration fields. Additional timers must be defined. **Note: This EF is not used currently.**

Identifier: "CTIM"	**Structure: transparent**		**Mandatory**
File size: 48 + 2 bytes	**Access: R/W**		
Bytes	**Description**	**M/O**	**Length**
1	Checksum	M	1 byte
2	Version	M	1 byte
3 until 6	Total duration	M	4 bytes
7 until 10	MO voice duration home PLMN	M	4 bytes
11 until 14	MT voice duration home PLMN	M	4 bytes
15 until 18	MO data duration home PLMN	M	4 bytes
19 until 22	MT data duration home PLMN	M	4 bytes
23 until 26	MO fax duration home PLMN	M	4 bytes
27 until 30	MT fax duration home PLMN	M	4 bytes
31 until 34	MO voice duration roaming	M	4 bytes
35 until 38	MT voice duration roaming	M	4 bytes
39 until 42	MO data duration roaming	M	4 bytes
43 until 46	MT data duration roaming	M	4 bytes
47 until 50	MO fax duration roaming	M	4 bytes
51 until 54	MT fax duration roaming	M	4 bytes

2.1.18 pcm_EFccnt_Type – Call counter

Definition:

```
#define SIZE_EF_CCNT 52
typedef struct pcm_EFccnt_Type
{
    UBYTE          Total [4];
    UBYTE          moVcDrHm [4];
    UBYTE          mtVcDrHm [4];
    UBYTE          moDtDrHm [4];
    UBYTE          mtDtDrHm [4];
    UBYTE          moFxDrHm [4];
    UBYTE          mtFxDrHm [4];
    UBYTE          moVcDrRm [4];
    UBYTE          mtVcDrRm [4];
    UBYTE          moDtDrRm [4];
    UBYTE          mtDtDrRm [4];
    UBYTE          mtFxDrRm [4];
    UBYTE          mtFxDrRm [4];
} EF_CCNT;
```

Description:

This EF provides call counters for different call types. Each counter contains a 32 bit integer value which represents the number of calls for this type. For example, byte 3 stores the most significant byte and byte 6 stores the least significant byte of the 32 bit value. The field “Total” contains the sum of all call counters. Additional counters must be defined. **Note: This EF is not used currently.**

<table>
<tr><th colspan="2">Identifier: "CCNT"</th><th colspan="1">Structure: transparent</th><th>Mandatory</th></tr>
<tr><th colspan="2">File size: SIZE_EF_CCNT + 2 bytes</th><th colspan="2">Access: R/W</th></tr>
<tr><th>Bytes</th><th>Description</th><th>M/
O</th><th>Length</th></tr>
<tr><td>1</td><td>Checksum</td><td>M</td><td>1 byte</td></tr>
<tr><td>2</td><td>Version</td><td>M</td><td>1 byte</td></tr>
<tr><td>3 until 6</td><td>Total duration</td><td>M</td><td>4 bytes</td></tr>
<tr><td>7 until 10</td><td>MO voice counter home PLMN</td><td>M</td><td>4 bytes</td></tr>
<tr><td>11 until 14</td><td>MT voice counter home PLMN</td><td>M</td><td>4 bytes</td></tr>
<tr><td>15 until 18</td><td>MO data counter home PLMN</td><td>M</td><td>4 bytes</td></tr>
<tr><td>19 until 22</td><td>MT data counter home PLMN</td><td>M</td><td>4 bytes</td></tr>
<tr><td>23 until 26</td><td>MO fax counter home PLMN</td><td>M</td><td>4 bytes</td></tr>
<tr><td>27 until 30</td><td>MT fax counter home PLMN</td><td>M</td><td>4 bytes</td></tr>
<tr><td>31 until 34</td><td>MO voice counter roaming</td><td>M</td><td>4 bytes</td></tr>
<tr><td>35 until 38</td><td>MT voice counter roaming</td><td>M</td><td>4 bytes</td></tr>
<tr><td>39 until 42</td><td>MO data counter roaming</td><td>M</td><td>4 bytes</td></tr>
<tr><td>43 until 46</td><td>MT data counter roaming</td><td>M</td><td>4 bytes</td></tr>
<tr><td>47 until 50</td><td>MO fax counter roaming</td><td>M</td><td>4 bytes</td></tr>
<tr><td>51 until 54</td><td>MT fax counter roaming</td><td>M</td><td>4 bytes</td></tr>
</table>

2.1.19 pcm_EFecc_Type – Emergency call codes

Definition:

```
#define SIZE_EF_ECC 15
typedef struct pcm_EFecc_Type
{
    UBYTE           ecc1[3];
    UBYTE           ecc2[3];
    UBYTE           ecc3[3];
    UBYTE           ecc4[3];
    UBYTE           ecc5[3];
} EF_ECC;
```

Description:

This record type is used by ACI.

This EF provides up to five emergency call codes. An emergency call code consists of 3 bytes containing a maximum of 6 BCD coded digits representing the number. Unused digits are coded as 'F'.

Identifier: "ECC"	**Structure: transparent**		**Mandatory**
File size: SIZE_EF_ECC + 2 bytes	**Access: R/W**		
Bytes	**Description**	**M/O**	**Length**
1	Checksum	M	1 byte
2	Version	M	1 byte
3 until 5	Emergency call code 1	M	3 bytes
6 until 8	Emergency call code 2	M	3 bytes
9 until 11	Emergency call code 3	M	3 bytes
12 until 14	Emergency call code 4	M	3 bytes
15 until 17	Emergency call code 5	M	3 bytes

- **emergency call code**

  Contains up to six BCD coded digits.

Octet	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**
1	MSBD 2	..	..	LSBD2	MSBD 1	..	..	LSBD 1
2	MSBD 4	..	..	LSBD4	MSBD 3	..	..	LSBD 3
3	MSBD 6	..	..	LSBD6	MSBD 5	..	..	LSBD 5

2.1.20 pcm_EForg_Type – Organizer and Alarm

Definition:

```
#define SIZE_EF_ORG 23
typedef struct pcm_EForg_Type
{
    UBYTE           date [6];
    UBYTE           alrm;
    UBYTE           alphMem[16];
} EF_ORG;
```

Description:

This EF provides a storing area for x organizer entries. Each record defines an event such as an appointment, specified by a date, a short descriptive text and the setting whether or not an alarm should be generated.

Note: This EF is not used currently.

Identifier: "ORG"	**Structure: linear fixed**		**Optional**
File size: 2 + SIZE_EF_ORG bytes per record	**Access: R/W**		
Bytes	**Description**	**M/O**	**Length**
1	Checksum	M	1 byte
2	Version	M	1 byte
3 until 8	Date/Time	M	6 byte
9	Alarm	M	1 bytes
10 until 23	alpha memo	M	16 bytes

- **date/time**

  Contains a BCD coded value (2 digits) representing the date and time of the call.

Octet	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**
1	Year							
2	Month							
3	Day							
4	Hour							
5	Minute							
6	Second							

- **alarm**

  The bits define the alarm.

Octet	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**
1	reserved				type			alrm

	Value	**Description**
alrm	0	alarm disabled

	1	alarm enabled
type	0	once
	1	daily
	2	weekly
	3	monthly
	4	yearly

- **alpha memo**

  This element is used to store user information (up to x bytes) of an event the user to be remembered.

2.1.21 pcm_EFccp_Type – Capability and configuration parameters

Definition:

```
#define SIZE_EF_CCP 7
typedef struct pcm_EFccp_Type
 {
     UBYTE           usrRate;
     UBYTE           bearServ;
     UBYTE           conElem;
     UBYTE           stopBits;
     UBYTE           dataBits;
     UBYTE           parity;
     UBYTE           flowCntrl;
} EF_CCP;
```

Description:

This EF contains parameters of required network and bearer capabilities and ME configurations associated with phone numbers of EF$_{LDN}$, EF$_{LRN}$, or EF$_{UPN}$. The contents and coding of the capabilities and configuration parameters are analogous to the bearer capabilities parameter defined in the MNCC SAP description.

Note: This EF is not used currently.

Identifier: "CCP"	**Structure: linear fixed**		**Optional**
File size: 2 + SIZE_EF_CCP bytes per record		**Access: R/W**	
Bytes	**Description**	**M/O**	**Length**
1	Checksum	M	1 byte
2	Version	M	1 byte
3	user rate	M	1 byte
4	bearer service	M	1 byte
5	connection element	M	1 byte
6	stop bits	M	1 byte
7	data bits	M	1 byte
8	Parity	M	1 byte

9	flow control	M	1 byte

2.1.22 pcm_EFext1_Type - Extension 1

Definition:

```
#define SIZE_EF_EXT1 13
typedef struct pcm_EFext1_Type
{
    UBYTE           recTp;
    UBYTE           extDat[11];
    UBYTE           id;
} EF_EXT1;
```

Description:

This EF provides a storage area for additional information concerning the records EF$_{LDN}$ EF$_{LRN}$.or EF$_{UPN}$. The information stored in this record can either be the overflow digits of a phone number or the sub-address of the phone number. EF$_{EXT1}$ fields can be concatenated.

Note: This EF is not used currently.

Identifier: "EXT1"	**Structure: linear fixed**		**Optional**
File size: 2 + SIZE_EF_EXT1 **bytes per record**	**Access: R/W**		
Bytes	**Description**	**M/O**	**Length**
1	checksum	M	1 byte
2	version	M	1 byte
3	record type	M	1 byte
4 until 14	extension data	M	11 bytes
15	identifier	M	1 bytes

- **record type**

  This byte identifies the type of extension data stored in the parameter ‘extension data’.

Octet	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**
1	reserved						type	

	Value	**Description**
type	0	unknown
	1	sub-address
	2	overflow digits
	3	reserved

- **extension data**

  If the record type indicates overflow data, the first byte of the extension data contains the number of bytes following (phone number). The coding of the digits following is the same as for the phone number stored in the elementary file. Unused nibbles of the extension data must be set to ‘F’.

  If the record type indicates a sub-address, the extension data contains information as defined in GSM 04.08. All information defined in GSM 04.08, except the information element identifier, is stored in the record. The length of the sub-address can be up to 22 bytes.

Extension fields can be concatenated if the storage capacity is exceeded. The reference to the next extension record is stored in the parameter 'identifier'. If unused, the contents of this field are set to '0xFF'.

- **Extension1 record identifier**

  This byte is the reference to another EF$_{EXT1}$ record. Set this byte to 'FF' if neither a sub-address nor an overflow is indicated.

2.1.23 pcm_EFsimlck_Type – SIM lock

Definition:

```
#define SIZE_EF_SIMLCK 62
typedef struct pcm_EFsimlck_Type
{
        UBYTE           locks1;
        UBYTE           locks2;
        UBYTE           cnt;
        UBYTE           maxcnt;
        UBYTE           PKey [8];
        UBYTE           SPKey [8];
        UBYTE           NSKey[8];
        UBYTE           CKey [8];
        UBYTE           NKey [8];
        UBYTE           len_imsi;
        UBYTE           imsi [15] ;
        UBYTE           gidl1;
        UBYTE           gidl2;
} EF_ SIMLCK;
```

Description:

This EF stores information to identify special SIM cards on which the mobile station is disabled for further use. There are five different locks defined which can be implemented by verifying different parts of the IMSI and the GID1 of the SIM with the stored data of this elementary file.

REMARK: Order of the various key parameter aligned to the implementation ! Additional the EF_SIMLCKEXT type is available, which allows a more generic access to SIM LOCK data. One of both shall be used.

Identifier: "SIMLCK"	**Structure: transparent**		**Mandatory**
File size: SIZE_EF_SIMLCK + 2 bytes	**Access: R/W (encrypted)**		
Bytes	**Description**	**M/O**	**Length**
1	Checksum	M	1 byte
2	Version	M	1 byte
3 until 4	Locks	M	2 bytes
5	unlock attempt counter	M	1 byte
6	Maximum attempt	M	1 byte

7 until 14	P control key	M	8 bytes
15 until 22	SP control key	M	8 bytes
23 until 30	NS control key	M	8 bytes
31 until 38	C control key	M	8 bytes
39 until 46	N control key	M	8 bytes
47	Length of IMSI	M	1 byte
48 until 62	IMSI	M	15 bytes
63	Group identifier level 1	M	1 byte
64	Group identifier level 2	M	1 byte

- **locks**

  Status of the different types of locks.

Octet	8	7	6	5	4	3	2	1
1	splock		nslock		nlock		plock	
2	reserved		reserved		reserved		clock	

	Value	Description
Plock	0	SIM lock disabled
	1	SIM lock enabled
	2	SIM lock locked
	3	SIM lock blocked
Nlock	0	Service provider lock disabled
	1	Service provider lock enabled
	2	Service provider lock locked
	3	Service provider lock blocked
Nslock	0	Network subset lock disabled
	1	Network subset lock enabled
	2	Network subset lock locked
	3	Network subset lock blocked
Splock	0	Network subset lock disabled
	1	Network subset lock enabled
	2	Network subset lock locked
	3	Network subset lock blocked
Clock	0	Network subset lock disabled
	1	Network subset lock enabled
	2	Network subset lock locked
	3	Network subset lock blocked

- **unlock attempt counter**

  The actual unlock value of attempt. It indicates the number of attempt to enter the appropriate key.

- **Maximum attempt**

  The maximum allowed value of unlock attempt.

- **control keys**

  A control key exists for each kind of lock which must be entered if the status of the lock must be changed. The length of a control key may be up to 16 digits. If a control key is less than 8 bytes(16 digits), unused nibbles are to be set to 'F'.

- **length of IMSI / IMSI**

  The length indicator refers to the number of significant bytes, not including this length byte, required for the IMSI. The IMSI field stores the range allowed for the each digit of IMSI, eg. 0x09 means that all values 0 to 9 are allowed for a digit, 0x55 means that the digit must be 5.

- **group identifier level 1**

  t.b.d.

- **group identifier level 2**

  t.b.d.

2.1.24 pcm_EFsimlckext_Type – Extended SIM lock

Definition:

```
#define SIZE_EF_SIMLCKEXT 142
typedef struct pcm_EFsimlckext_Type
{
        UBYTE              locks1;
        UBYTE              locks2;
        UBYTE              cnt;
        UBYTE              maxcnt;
        UBYTE              PKey [8];
        UBYTE              SPKey [8];
        UBYTE              NSKey[8];
        UBYTE              CKey [8];
        UBYTE              NKey [8];
        UBYTE              len_p_imsi;
        UBYTE              p_imsi [15] ;
        UBYTE              len_sp_imsi;
        UBYTE              sp_imsi [15] ;
        UBYTE              len_ns_imsi;
        UBYTE              ns_imsi [15] ;
        UBYTE              len_c_imsi;
        UBYTE              c_imsi [15] ;
        UBYTE              len_n_imsi;
        UBYTE              n_imsi [15] ;
        UBYTE              len_u_imsi;
        UBYTE              u_imsi [15] ;
        UBYTE              gidl1;
        UBYTE              gidl2;
} EF_ SIMLCKEXT;
```

Description:

This EF stores information to identify special SIM cards on which the mobile station is disabled for further use. There are five different locks defined which can be implemented by verifying different parts of the IMSI (individual definitions for each lock) and the GID1 of the SIM with the stored data of this elementary file. Additional an unblock IMSI is defined, which resets the unblock attempt counter if needed.

REMARK: Additional the EF_SIMLCK type is available, which allows a simplier definition of SIM LOCK data. One of both shall be used.

Identifier: "SIMLCKEXT"	**Structure: transparent**		**Mandatory**
File size: SIZE_EF_SIMLCKEXT + 2 bytes	**Access: R/W (encrypted)**		
Bytes	**Description**	**M/O**	**Length**
1	Checksum	M	1 byte
2	Version	M	1 byte
3 until 4	Locks	M	2 bytes
5	unlock attempt counter	M	1 byte
6	Maximum attempt	M	1 byte
7 until 14	P control key	M	8 bytes
15 until 22	SP control key	M	8 bytes
23 until 30	NS control key	M	8 bytes
31 until 38	C control key	M	8 bytes
39 until 46	N control key	M	8 bytes
47	Length of IMSI P-LOCK	M	1 byte
48 until 62	IMSI P-LOCK	M	15 bytes
63	Length of IMSI SP-LOCK	M	1 byte
64 until 78	IMSI SP-LOCK	M	15 bytes
79	Length of IMSI NS-LOCK	M	1 byte
80 until 94	IMSI NS-LOCK	M	15 bytes
95	Length of IMSI C-LOCK	M	1 byte
96 until 110	IMSI C-LOCK	M	15 bytes
111	Length of IMSI N-LOCK	M	1 byte
112 until 126	IMSI N-LOCK	M	15 bytes
127	Length of IMSI U-LOCK	M	1 byte
128 until 142	IMSI U-LOCK	M	15 bytes
143	Group identifier level 1	M	1 byte
144	Group identifier level 2	M	1 byte

- **locks**

  Status of the different types of locks.

Octet	8	7	6	5	4	3	2	1
1	splock		nslock		nlock		plock	
2	reserved		reserved		reserved		clock	

	Value	Description
Plock	0	SIM lock disabled
	1	SIM lock enabled
	2	SIM lock locked
	3	SIM lock blocked
Nlock	0	Service provider lock disabled
	1	Service provider lock enabled
	2	Service provider lock locked
	3	Service provider lock blocked
Nslock	0	Network subset lock disabled
	1	Network subset lock enabled
	2	Network subset lock locked
	3	Network subset lock blocked
Splock	0	Network subset lock disabled
	1	Network subset lock enabled
	2	Network subset lock locked
	3	Network subset lock blocked
Clock	0	Network subset lock disabled
	1	Network subset lock enabled
	2	Network subset lock locked
	3	Network subset lock blocked

- **unlock attempt counter**

  The actual unlock value of attempt. It indicates the number of attempt to enter the appropriate key.

- **Maximum attempt**

  The maximum allowed value of unlock attempt.

- **control keys**

  A control key exists for each kind of lock which must be entered if the status of the lock must be changed. The length of a control key may be up to 16 digits. If a control key is less than 8 bytes(16 digits), unused nibbles are to be set to ‘F’.

- **length of IMSI / IMSI**

  The length indicator refers to the number of significant bytes, not including this length byte, required for the IMSI. The IMSI field store the range allowed for the each digit of IMSI, eg. 0x09 means that all values 0 to 9 are allowed for a digit, 0x55 means that the digit must be 5. For each lock a single IMSI definition is available. Additional one IMSI definition will be used to reset the unblock attempt counter if needed.

- **group identifier level 1**

  t.b.d.

- **group identifier level 2**

  t.b.d.

2.1.25 pcm_EFmnt_Type – Maintenance information

Definition:

```
#define SIZE_EF_MAIN 8
```

Description:

This EF provides service and maintenance information such as SW/HW version numbers, manufacturer identifier, manufacturing date, line identifier etc.

T.B.D.

<table>
<tr><th colspan="2">Identifier: "MAIN"</th><th colspan="2">Structure: transparent</th><th>Mandatory</th></tr>
<tr><th colspan="3">File size: SIZE_EF_MAIN + 2 bytes</th><th colspan="2">Access: R/W</th></tr>
<tr><th>Bytes</th><th colspan="2">Description</th><th>M/O</th><th>Length</th></tr>
<tr><td>1</td><td colspan="2">checksum</td><td>M</td><td>1 byte</td></tr>
<tr><td>2</td><td colspan="2">version</td><td>M</td><td>1 byte</td></tr>
<tr><td></td><td colspan="2">t.b.d</td><td>M</td><td></td></tr>
</table>

2.1.26 pcm_EFsfk_Type – Special function keys

Definition:

```
#define SIZE_EF_SFK 8
```

Description:

This EF stores information for a programmable key, for example, a key that directly opens the personal phonebook

Identifier: "SFK"	**Structure: transparent**		**Mandatory**
File size: SIZE_EF_SFK + 2 bytes	**Access: R/W**		
Bytes	**Description**	**M/ O**	**Length**
1	checksum	M	1 byte
2	version	M	1 byte
	t.b.d	M	

2.1.27 pcm_EFflt_Type – Fault conditions

Definition:

```
#define SIZE_EF_FAULT 8
```

Description:

This EF provides a storage area to keep track of certain fault conditions of the mobile to be used by the manufacturer for debugging purposes. t.b.d.

Identifier: "FAULT"	**Structure: linear fixed**		**Optional**
File size: 2 + SIZE_EF_FAULT bytes per record		**Access: R/W**	
Bytes	**Description**	**M/O**	**Length**
1	checksum	M	1 byte
2	version	M	1 byte
	t.b.d.	M	

2.1.28 pcm_EFdbg_Type – Debug information

Definition:

```
#define SIZE_EF_DEBUG 8
```

Description:

This EF provides several debug information elements for traces and tracking of the mobile stations entities. t.b.d.

<table>
<tr><th colspan="2">Identifier: "DEBUG"</th><th colspan="2">Structure: transparent</th><th>Mandatory</th></tr>
<tr><th colspan="3">File size: SIZE_EF_DEBUG + 2 bytes</th><th colspan="2">Access: R/W</th></tr>
<tr><th>Bytes</th><th colspan="2">Description</th><th>M/O</th><th>Length</th></tr>
<tr><td>1</td><td colspan="2">checksum</td><td>M</td><td>1 byte</td></tr>
<tr><td>2</td><td colspan="2">version</td><td>M</td><td>1 byte</td></tr>
<tr><td></td><td colspan="2">t.b.d</td><td>M</td><td></td></tr>
</table>

2.1.29 pcm_EFbat_Type – Power management

Definition:

```
#define SIZE_EF_POWER 8
```

Description:

This EF provides values required to manage power related functions, such as battery charging, inactivity timer, etc. t.b.d.

<table>
<tr><td colspan="2"><b>Identifier: "POWER"</b></td><td colspan="2"><b>Structure: transparent</b></td><td><b>Mandatory</b></td></tr>
<tr><td colspan="3"><b>File size: SIZE_EF_POWER + 2 bytes</b></td><td colspan="2"><b>Access: R/W</b></td></tr>
<tr><td><b>Bytes</b></td><td colspan="2"><b>Description</b></td><td><b>M/O</b></td><td><b>Length</b></td></tr>
<tr><td>1</td><td colspan="2">checksum</td><td>M</td><td>1 byte</td></tr>
<tr><td>2</td><td colspan="2">version</td><td>M</td><td>1 byte</td></tr>
<tr><td></td><td colspan="2">t.b.d</td><td>M</td><td></td></tr>
</table>

2.1.30 pcm_EFkbd_Type – Keyboard mapping

Definition:

```
#define SIZE_EF_KEYB 64
typedef struct pcm_EFkbd_Type
{
      UBYTE            logical_key [32];
      UBYTE            raw_key [32];
} EF_KBD;
```

Description:

This EF contains a keyboard map to adjust the keyboard layout. Therefore, the key codes of the keys can be changed to alter the functionality or keys may be deactivated.

Note: This EF is not used currently.

Identifier: "KEYB"	**Structure: transparent**		**Mandatory**
File size: SIZE_EF_KEYB + 2 bytes	**Access: R/W**		
Bytes	**Description**	**M/ O**	**Length**
1	Checksum	M	1 byte
2	Version	M	1 byte
3 until 34	Logical key map	M	32 bytes
35 until 66	Raw key map	M	32 bytes

2.1.31 pcm_EFrdio_Type - Radio parameters

Definition:

```
#define SIZE_EF_RADIO 8
```

Description:

This EF stores information to adjust the radio path.

t.b.d.

<table>
<tr><th colspan="2">Identifier: "RADIO"</th><th>Structure: transparent</th><th>Mandatory</th></tr>
<tr><th colspan="2">File size: SIZE_EF_RADIO + 2 bytes</th><th colspan="2">Access: R/W</th></tr>
<tr><th>Bytes</th><th>Description</th><th>M/O</th><th>Length</th></tr>
<tr><td>1</td><td>checksum</td><td>M</td><td>1 byte</td></tr>
<tr><td>2</td><td>version</td><td>M</td><td>1 byte</td></tr>
<tr><td></td><td>t.b.d</td><td>M</td><td></td></tr>
</table>

2.1.32 pcm_EFcgmi_Type – Manufacturer

Definition:

```
#define SIZE_EF_CGMI_DATA 20 /* value depends on manufacturer specification */

typedef struct pcm_EFcgmi_Type
{
     UBYTE            data[SIZE_EF_CGMI_DATA];
} EF_CGMI;
```

Description:

This record type is used by AT Command Interpreter.

This EF is intended to permit the user of the TA to identify the manufacturer of the ME to which it is connected to.

Identifier: "CGMI"	**Structure: transparent**	**Optional**	
File size: 2 + SIZE_EF_CGMI_DATA bytes		**Access: R**	
Bytes	**Description**	**M/O**	**Length**
1	checksum	M	1 byte
2	version	M	1 byte
3 until (SIZE_EF_CGMI_DATA + 2)	name of manufacturer	M	SIZE_EF_CGMI_DATA byte

- **Name of Manufacturer**

  The text will consist of a single line containing the name of the manufacturer, but manufacturers may choose to provide more information if desired. The length of the text shall not exceed the value of MAX_CMD_LEN defined in the ACI SAP document.

  Text shall not contain the sequence `0<CR>` or `OK<CR>`.

2.1.33 pcm_EFinf0_Type – Identification Information

Definition:

```
#define SIZE_EF_INF0_DATA 20 /* value depends on manufacturer specification */

typedef struct pcm_EFinf0_Type
{
      UBYTE              data[SIZE_EF_INF0_DATA];
} EF_INF0;
```

Description:

This record type is used by AT Command Interpreter.

This EF is intended to permit the user of the TA to identify the manufacturer of the ME to which it is connected to.

Identifier: "INF0"	**Structure: transparent**	**Optional**	
File size: 2 + SIZE_EF_INF0_DATA bytes		**Access: R**	
Bytes	**Description**	**M/O**	**Length**
1	checksum	M	1 byte
2	version	M	1 byte
3 until (SIZE_EF_INF0_DATA + 2)	identification information	M	SIZE_EF_INF0_DATA byte

- **Name of Manufacturer**

  The text will consist of a single line containing the name of the manufacturer, but manufacturers may choose to provide more information if desired. The length of the text shall not exceed the value of MAX_CMD_LEN defined in the ACI SAP document.

  Text shall not contain the sequence `0<CR>` or `OK<CR>`.

2.1.34 pcm_EFcgmm_Type – Model

Definition:

```
#define SIZE_EF_CGMM_DATA 20 /* value depends on manufacturer specification */

typedef struct pcm_EFcgmm_Type
{
     UBYTE              data[SIZE_EF_CGMM_DATA];
} EF_CGMM;
```

Description:

This record type is used by AT Command Interpreter.

This EF is intended to permit the user of the TA to identify the specific model of the ME to which it is connected to.

Identifier: "CGMM"	**Structure: transparent**	**Optional**	
File size: 2 + SIZE_EF_CGMM_DATA bytes		**Access: R**	
Bytes	**Description**	**M/O**	**Length**
1	checksum	M	1 byte
2	version	M	1 byte
3 until (SIZE_EF_CGMM_DATA + 2)	name of product	M	SIZE_EF_CGMM_DATA byte

- **Name of Product**

  The text will consist of a single line containing the name of the product, but manufacturers may choose to provide more information if desired. The length of the text shall not exceed the value of MAX_CMD_LEN defined in the ACI SAP document.

  Text shall not contain the sequence `0<CR>` or `OK<CR>`.

2.1.35 pcm_EFcgmr_Type – Revision

Definition:

```
#define SIZE_EF_CMGR_DATA 20 /* value depends on manufacturer specification */

typedef struct pcm_EFcgmr_Type
{
    UBYTE           data[SIZE_EF_CMGR_DATA];
} EF_CGMR;
```

Description:

This record type is used by AT Command Interpreter.

This EF is intended to permit the user of the TA to identify the version, revision level or date, or other pertinent information of the ME to which it is connected to.

Identifier: "CGMR"	**Structure: transparent**	**Optional**	
File size: 2 + SIZE_EF_CGMR_DATA bytes		**Access: R**	
Bytes	**Description**	**M/O**	**Length**
1	checksum	M	1 byte
2	version	M	1 byte
3 until (SIZE_EF_CGMR_DATA + 2)	version of product	M	SIZE_EF_CGMR_DATA byte

- **Version of Product**

  The text will consist of a single line containing the version of the product, but manufacturers may choose to provide more information if desired. The length of the text shall not exceed the value of MAX_CMD_LEN defined in the ACI SAP document.

  Text shall not contain the sequence `0<CR>` or `OK<CR>`.

2.1.36 pcm_EFcgsn_Type – Product Serial Number

Definition:

```
#define SIZE_EF_CGSN_DATA 20 /* value depends on manufacturer specification */

typedef struct pcm_EFcgsn_Type
{
    UBYTE           data[SIZE_EF_CGSN_DATA];
} EF_CGSN;
```

Description:

This record type is used by AT Command Interpreter.

This EF is intended to permit the user of the TA to identify the individual ME to which it is connected to.

Identifier: "CGSN"	**Structure: transparent**	**Optional**	
File size: 2 + SIZE_EF_CGSN_DATA bytes		**Access: R**	
Bytes	**Description**	**M/O**	**Length**
1	checksum	M	1 byte
2	version	M	1 byte
3 until (SIZE_EF_CGSN_DATA + 2)	serial number of product	M	SIZE_EF_CGSN_DATA byte

- **Serial Number of Product**

  The text will consist of a single line containing the IMEI (International Mobile station Equipment Identity) number of the ME, but manufacturers may choose to provide more information if desired. The length of the text shall not exceed the value of MAX_CMD_LEN defined in the ACI SAP document.

  Text shall not contain the sequence `0<CR>` or `OK<CR>`.

2.1.37 pcm_EFsmsprfl_Type – SMS Profile

Definition:

```
#define SIZE_EF_SMSPRFL_SCA   20
#define SIZE_EF_SMSPRFL_MIDS  40
#define SIZE_EF_SMSPRFL_DCSS  20
#define SIZE_EF_SMSPRFL_VPABS 14
#define SIZE_EF_SMSPRFL ( SIZE_EF_SMSPRFL_SCA  + SIZE_EF_SMSPRFL_MIDS +
SIZE_EF_SMSPRFL_DCSS  + SIZE_EF_SMSPRFL_VPABS + 9)

typedef struct pcm_EFsmsprfl_Type
{
    UBYTE          vldFlag;
    UBYTE          CSCAsca[SIZE_EF_SMSPRFL_SCA];
    UBYTE          CSCAlenSca;
    UBYTE          CSCAton;
    UBYTE          CSCAnpi;
    UBYTE          CSCBmode;
    UBYTE          CSCBmids[SIZE_EF_SMSPRFL_MIDS];
    UBYTE          CSCBdcss[SIZE_EF_SMSPRFL_DCSS];
    UBYTE          CSMPfo;
    UBYTE          CSMPvprel;
    UBYTE          CSMPvpabs[SIZE_EF_SMSPRFL_VPABS];
    UBYTE          CSMPpid;
    UBYTE          CSMPdcs;
} EF_SMSPRFL;
```

Description:

This record type is used by the SMS component of the ACI

This EF is used to store message service settings in non-volatile memory. A TA can contain several profiles of settings. All settings specified in AT commands Service Centre Address `+CSCA`, Set Message Parameters `+CSMP` and Select Cell Broadcast Message Types `+CSCB` are stored.

Identifier: "SMSPRFL"	**Structure: linear fixed**		**Optional**
File size: 2 + SIZE_EF_SMSPRFL bytes per record	**Access: R/W**		
Bytes	**Description**	**M/O**	**Length**
1	checksum	M	1 byte
2	version	M	1 byte
3	valid flag	M	1 bytes
4 until (SIZE_EF_SMSPRFL_SCA +3)	service center address	M	SIZE_EF_SMSPRFL_SCA bytes
(SIZE_EF_SMSPRFL_SCA +4)	length of service center address	M	1 bytes

(SIZE_EF_SMSPRFL_SCA +5)	type of number	M	1 bytes
(SIZE_EF_SMSPRFL_SCA +6)	numbering plan identification	M	1 bytes
(SIZE_EF_SMSPRFL_SCA +7)	cell broadcast mode	M	1 bytes
(SIZE_EF_SMSPRFL_SCA +8) until (SIZE_EF_SMSPRFL_SCA +SIZE_EF_SMSPRFL_MIDS+7)	message identifiers	M	SIZE_EF_SMSPRFL_MIDS bytes
(SIZE_EF_SMSPRFL_SCA +SIZE_EF_SMSPRFL_MIDS+8) until (SIZE_EF_SMSPRFL_SCA +SIZE_EF_SMSPRFL_MIDS + SIZE_EF_SMSPRFL_DCSS +7)	data coding schemes	M	SIZE_EF_SMSPRFL_DCSS bytes
(SIZE_EF_SMSPRFL_SCA +SIZE_EF_SMSPRFL_MIDS + SIZE_EF_SMSPRFL_DCSS +8)	first octet	M	1 bytes
(SIZE_EF_SMSPRFL_SCA +SIZE_EF_SMSPRFL_MIDS + SIZE_EF_SMSPRFL_DCSS +9)	validity period relative	M	1 bytes
(SIZE_EF_SMSPRFL_SCA +SIZE_EF_SMSPRFL_MIDS + SIZE_EF_SMSPRFL_DCSS+10) until (SIZE_EF_SMSPRFL_SCA +SIZE_EF_SMSPRFL_MIDS + SIZE_EF_SMSPRFL_DCSS + SIZE_EF_SMSPRFL_VPABS+9)	validity period absolute	M	SIZE_EF_SMSPRFL_VPABS bytes
(SIZE_EF_SMSPRFL_SCA +SIZE_EF_SMSPRFL_MIDS + SIZE_EF_SMSPRFL_DCSS + SIZE_EF_SMSPRFL_VPABS+10)	protocol identifier	M	1 bytes
(SIZE_EF_SMSPRFL_SCA +SIZE_EF_SMSPRFL_MIDS + SIZE_EF_SMSPRFL_DCSS + SIZE_EF_SMSPRFL_VPABS+11)	data coding scheme	M	1 bytes

- **Valid Flag**

  This byte is set to '0xFF' if the SMS profile is not valid. In any other case this byte should be set to '0x00'.

- **Service Center Address**

  Service Center Address as an ASCII string. The most significant character of the string starts with octet 1. The string is not NULL terminated, unused octets are set to '0xFF'.

Octet	8	7	6	5	4	3	2	1
1	most significant character							
…								
20	least significant character							

- **Length of Service Center Address**

This byte stores the number of characters of the service center address.

- **Type of Number**

  Type of number according to GSM 04.08. If this information field is not present the byte should be set to '0xFF'.

- **Numbering Plan Identification**

  Numbering plan identification according to GSM 04.08. If this information field is not present the byte should be set to '0xFF'.

- **Cell Broadcast Mode**

  The byte defines the use of message identifiers and data coding schemes for CBCH.

Octet	8	7	6	5	4	3	2	1
1	mode							

	Value	Description
mode	0	accept the message identifiers (CSCBmids) and data coding schemes (CSCBdcss)
	1	ignore the message identifiers (CSCBmids) and data coding schemes (CSCBdcss)

- **Message Identifiers**

  Message Identifiers will be represented by 16 bit values.

  Values listed show the types of message which shall be accepted by the MS. Two successive values define a whole range of types. The first value of a pair is the lowest accepted value the second the highest one. If a single type of message should be accepted the two values of a pair should be set to the same value. Unused entries shall be set to '0xFF 0xFF'.

Octet	8	7	6	5	4	3	2	1
1	MID range 1 low value MSB							
2	MID range 1 low value LSB							
3	MID range 1 high value MSB							
4	MID range 1 high value LSB							
..	..							

- **Data Coding Schemes**

  Data Coding Schemes will be represented by 8 bit values.

  Values listed show the data coding schemes of message which shall be accepted by the MS. Two successive values define a whole range of schemes. The first value of a pair is the lowest accepted value the second the highest one. If a single data coding scheme of message should be accepted the two values of a pair should be set to the same value. Unused entries shall be set to '0xFF'.

Octet	8	7	6	5	4	3	2	1
1	DCS range 1 low value							
2	DCS range 1 high value							
3	DCS range 2 low value							
4	DCS range 2 high value							

..	..

- **First Octet**

  First octet of the SMS-DELIVER REPORT, SMS-COMMAND or SMS-SUBMIT PDUs according to GSM 03.40.

- **Validity Period Relative**

  TP-Validity-Period according to GSM 03.40 giving the length of the validity period, counted from when the SMS-SUBMIT is received by the Service Center.

- **Validity Period Absolute**

  TP-Validity-Period according to GSM 03.40 giving the absolute time of the validity period termination.

- **Protocol Identifier**

  TP-Protocol-Identifier according to GSM 03.40

- **Data Coding Scheme**

  TP-Data-Coding-Scheme according to GSM 03.40.

2.1.38 pcm_EFplmn_Type – PLMN Identifier

Definition:

```
#define SIZE_EF_PLMN_LONG 20
#define SIZE_EF_PLMN_SHRT 10
#define SIZE_EF_PLMN_MCC   2
#define SIZE_EF_PLMN_MNC   2
#define SIZE_EF_PLMN ( SIZE_EF_PLMN_MCC  + SIZE_EF_PLMN_MCC + SIZE_EF_PLMN_LONG +
SIZE_EF_PLMN_SHRT )


typedef struct pcm_EFplmn_Type
{
    UBYTE           mcc [SIZE_EF_PLMN_MCC];
    UBYTE           mnc [SIZE_EF_PLMN_MNC];
    UBYTE           lngNam [SIZE_EF_PLMN_LONG];
    UBYTE           shrtNam [SIZE_EF_PLMN_SHRT];
} EF_PLMN;
```

Description:

This EF provides PLMN descriptions. Each PLMN is represented by a long name, a short name, the mobile network code and the mobile country code. The values for mnc and mcc are represented using a 16 bit integer value. For example, mnc[0] stores the most significant byte and mnc[1] stores the least significant byte of the 16 bit value. The bytes are encoded as BCD values. For two digit MNC networks, the third MNC digit is encoded as 0xF. Coding example: MCC=262, MNC=01 => mcc[0] = 0x02, mcc[1] = 0x62, mnc[0] = 0x00, mnc[1] = 0x1F.

Identifier: "PLMN"	**Structure: linear fixed**		**Optional**
File size: 2 + SIZE_EF_PLMN bytes per record	**Access: R**		
Bytes	**Description**	**M/O**	**Length**
1	Checksum	M	1 byte
2	Version	M	1 byte
3 until (SIZE_EF_PLMN_MCC + 2)	Mobile country code	M	SIZE_EF_PLMN_MCC bytes
(SIZE_EF_PLMN_MCC + 3) until (SIZE_EF_PLMN_MCC + 2 + SIZE_EF_PLMN_MNC)	Mobile network code	M	SIZE_EF_PLMN_MNC bytes
(SIZE_EF_PLMN_MCC + SIZE_EF_PLMN_MNC + 3) until (SIZE_EF_PLMN_MCC + SIZE_EF_PLMN_MNC + SIZE_EF_PLMN_LONG+ 2)	MT voice counter home PLMN	M	SIZE_EF_PLMN_LONG bytes
(SIZE_EF_PLMN_MCC + SIZE_EF_PLMN_MNC + SIZE_EF_PLMN_LONG+ 3)	MO data counter home PLMN	M	SIZE_EF_PLMN_SHRT bytes

until SIZE_EF_PLMN_MCC + SIZE_EF_PLMN_MNC + SIZE_EF_PLMN_LONG+ SIZE_EF_PLMN_SHRT + 2)			

2.1.39 pcm_EFbcch_info_Type – BCCH information

Definition:

```
#define SIZE_EF_BCCHINFO 54
typedef struct pcm_EFbcch_info_Type
{
        UBYTE               bcch_info[54];
} EF_BCCHINFO;
```

Description:

This record type is used by G23.

This EF provides the BCCH channel numbers for non GSM 900 frequency standards. The GSM 900 BCCH channel numbers are stored on the SIM.

Identifier: "BCCHINF"	**Structure: transparent**		**Optional**
File size: SIZE_EF_BCCHINFO +2 bytes	**Access: R/W**		
Bytes	**Description**	**M/O**	**Length**
1	checksum	M	1 byte
2	version	M	1 byte
3 until 56	bcch_info	M	54 byte

- **bcch_info**

  The bits of these bytes define the used BCCH channel numbers for a faster access to the network after switching on via the stored cell selection procedure. The content of the field depends on the configured frequency standard.

GSM 900

Not used (realized with the SIM card field).

DCS 1800, Dualband GSM 900 / DCS 1800

channels 512 to 885 = 374 channels

Byte / Bit	7	6	5	4	3	2	1	0
0	---	---	885	884	883	882	881	880
1	879	878	877	876	875	874	873	872
...	...	...	...	...	...	...	...	...
45	527	526	525	524	523	522	521	520
46	519	518	517	516	515	514	513	512

PCS 1900

channels 512 to 810 = 299 channels

Byte / Bit	7	6	5	4	3	2	1	0
0	---	---	---	---	---	810	809	808
1	807	806	805	804	803	802	801	800
...	...	...	...	...	...	...	...	...
36	527	526	525	524	523	522	521	520
37	519	518	517	516	515	514	513	512

Dualband Extended GSM 900 / E-GSM / DCS 1800

channels 512 to 885 = 374 channels
channels 975 to 1023 plus 0 = 50 channels

Byte / Bit	7	6	5	4	3	2	1	0
0	---	---	---	---	---	---	0	1023
...	...	...	...	...	...	...	...	...
6	982	981	980	979	978	977	976	975
7	---	---	885	884	883	882	881	880
8	879	878	877	876	875	874	873	872
...	...	...	...	...	...	...	...	...
52	527	526	525	524	523	522	521	520
53	519	518	517	516	515	514	513	512

2.1.40 pcm_EFals_Type – alternative line service

Definition:

```
#define SIZE_EF_ALS 2
typedef struct pcm_EFals_Type
{
        UBYTE           selLine;
        UBYTE           statLine;
} EF_ALS;
```

Description:

This record type is used by MFW.

This EF is intended to permit the user of the TA to identify the used line and its status.

Identifier: "ALS"	**Structure: transparent**	**Optional**	
File size: SIZE_EF_ALS + 2 bytes		**Access: R/W**	
Bytes	**Description**	**M/O**	**Length**
1	checksum	M	1 byte
2	version	M	1 byte
3	selected line	M	1 byte
4	status line	M	1 byte

- **selLine**

  The text will contain the information which line is selected currently.

- **statLine**

  The text will contain the status of the selected line.

2.1.41 pcm_EFlocgprs_Type – location information (GPRS)

Definition:

```
#define SIZE_EF_LOCGPRS 14
typedef struct pcm_EFlocgprs_Type
{
        UBYTE           ptmsi[4];
        UBYTE           ptmsi_signature[3];
        UBYTE           rai[6];
        UBYTE           ra_status;
} EF_LOCGPRS;
```

Description:

This record type is used by G23.

This EF is used for results of attach, routing area update or P-TMSI reallocation procedures for authentication and identification purposes.

Identifier: "LOCGPRS"	**Structure: transparent**	**Optional**	
File size: 2 + SIZE_EF_LOCGPRS bytes		**Access: R/W**	
Bytes	**Description**	**M/O**	**Length**
1	checksum	M	1 byte
2	version	M	1 byte
3 until 6	packet TMSI	M	4 byte
7 until 9	packet TMSI signature value	M	3 byte
10 until 15	routing area information	M	6 byte
16	status of rai	M	1 byte

2.1.42 pcm_EFkcgprs_Type – Ciphering Key (GPRS)

Definition:

```
#define SIZE_EF_KCGPRS 9
typedef struct pcm_EFkcgprs_Type
{
    UBYTE           kc[8];
    UBYTE           cksn;
} EF_KCGPRS;
```

Description:

This record type is used by G23.

This EF is used for GPRS ciphering and authentication procedures.

Identifier: "KCGPRS"	**Structure: transparent**	**Optional**	
File size: 2 + SIZE_EF_KCGPRS bytes		**Access: R/W**	
Bytes	**Description**	**M/O**	**Length**
1	checksum	M	1 byte
2	version	M	1 byte
3 until 10	currently used ciphering key	M	8 byte
11	ciphering key sequence number of kc	M	1 byte

2.1.43 pcm_EFimsigprs_Type – IMSI (GPRS)

Definition:

```
#define SIZE_EF_IMSIGPRS 9
typedef struct pcm_EFimsigprs_Type
{
        UBYTE           len;
        UBYTE           IMSI[8];
} EF_IMSIGPRS;
```

Description:

This record type is used by G23.

This EF provides the IMSI. The field is used for the validation check of the other two GPRS fields. The requirement is given by GSM 03.60.

<table>
<tr><th colspan="2">Identifier: "IMSIGPRS"</th><th>Structure: transparent</th><th colspan="2">Mandatory</th></tr>
<tr><th colspan="2">File size: SIZE_EF_IMSIGPRS +2 bytes</th><th colspan="3">Access: R/- (encrypted)</th></tr>
<tr><th>Bytes</th><th colspan="2">Description</th><th>M/O</th><th>Length</th></tr>
<tr><td>1</td><td colspan="2">checksum</td><td>M</td><td>1 byte</td></tr>
<tr><td>2</td><td colspan="2">version</td><td>M</td><td>1 byte</td></tr>
<tr><td>3</td><td colspan="2">len IMSI</td><td>M</td><td>1 bytes</td></tr>
<tr><td>4 until 11</td><td colspan="2">IMSI</td><td>M</td><td>8 bytes</td></tr>
</table>

The IMSI has 8 digits.

Octet	8	7	6	5	4	3	2	1
1	MSBD 2	..	..	LSBD2	MSBD 1	..	..	LSBD 1
2	MSBD 4	..	..	LSBD4	MSBD 3	..	..	LSBD 3
..								

2.2 Constants

Name	Description
PCM_INVALID_FILE	Unknown file name
PCM_INVALID_SIZE	File sizes differs
PCM_INVALID_CKSM	Invalid checksum
PCM_INVALID_RECORD	Invalid record number
PCM_NVRAM_ACCS_FAIL	Access of the non volatile RAM failed

2.3 Functions

Name	Description
pcm_Init	Initialization of PCM
pcm_Exit	Termination of PCM
pcm_ReadFile	Read transparent elementary file
pcm_ReadRecord	Read linear fixed elementary file
pcm_WriteFile	Write transparent elementary file
pcm_WriteRecord	Write linear fixed elementary file
pcm_Flush	Save changes in the EEPROM at once

2.3.1 pcm_Init – Driver Initialization

Definition:

```
UBYTE pcm_Init
(
    void
);
```

Parameters:

Name	Description
-	-

Return values:

Name	Description
DRV_OK	Initialization successful
DRV_INITFAILURE	Initialization failed (the PCM could not be copied to the RAM)
DRV_INITIALIZED	Driver already initialized

Description

This function is used to initialize the permanent configuration memory driver. PCM reads the data of the permanent configuration memory, usually stored in an EEPROM, into a the RAM. After initialization, all other functions of the driver can be called.

The function returns DRV_INITIALIZED if the driver has already been initialized and is ready to be used or is already in use.

2.3.2 pcm_Exit – De-initialization of the driver

Definition:

```
void pcm_Exit
(
    void
);
```

Parameters:

Name	Description
-	-

Return values:

Name	Description
-	-

Description

This function is used to indicate to PCM that its functionality is no longer needed. PCM writes the data of the permanent configuration memory of the RAM into in the non volatile RAM, e.g. EEPROM.

2.3.3 pcm_ReadFile – Read transparent elementary file

Definition:

```
UBYTE pcm_ReadFile
(
        UBYTE *         in_FileName
        USHORT          in_BufferSize
        UBYTE *         out_BufferPtr
        UBYTE *         out_VersionPtr
) ;
```

Parameters:

Name	Description
in_FileName	Elementary file name
in_BufferSize	Size of the buffer where the file is copied
out_BufferPtr	Pointer to the buffer where the file is copied
out_VersionPtr	Pointer to the buffer where the file version is stored (size is one byte)

Return values:

Name	Description
DRV_OK	Function completed successful
PCM_INVALID_FILE	Unknown file name
PCM_INVALID_SIZE	File sizes differ
PCM_INVALID_CKSM	Invalid checksum

Description

The function reads a transparent elementary file. Therefore, the copy of the permanent configuration memory in the RAM is accessed. If the file exists, the contents of the file is copied into the buffer provided. The version of the file is copied into the buffer to which out_VersionPtr points. The function returns PCM_OK.

If the file is unknown, the function returns PCM_INVALID_FILE.

If the file is known, but the file sizes differ, the function returns PCM_INVALID_SIZE.

If the checksum of the file is invalid, the function returns PCM_INVALID_CKSM.

2.3.4 pcm_GetFileInfo – Get information about a dedicated file

Definition:

```
UBYTE pcm_GetFileInfo
(
        UBYTE *                 in_FileName
        pcm_FileInfo_Type *     out_FileInfoPtr
);
```

Parameters:

Name	Description
in_FileName	Elementary file name
out_FileInfoPtr	Pointer to the buffer where the file information is copied

Return values:

Name	Description
DRV_OK	Function completed successful
PCM_INVALID_FILE	Unknown file name

Description

The function reads the common information of a transparent elementary file. Therefore, the copy of the permanent configuration memory in the RAM is accessed. If the file exists, the file information is copied into the file information buffer to which out_FileInfoPtr points. The function returns PCM_OK if the file exists.

If the file is unknown, the function returns PCM_INVALID_FILE.

2.3.5 pcm_ReadRecord – Read a linear fixed elementary file

Definition:

```
UBYTE pcm_ReadRecord
(
        UBYTE *          in_FileName
        USHORT           in_Record
        USHORT           in_BufferSize
        UBYTE *          out_BufferPtr
        UBYTE *          out_VersionPtr
        USHORT *         out_MaxRecordsPtr
);
```

Parameters:

Name	Description
in_FileName	Elementary file name
in_Record	Number of the record to read
in_BufferSize	Size of the buffer where the record is copied
out_BufferPtr	Pointer to the buffer where the record is copied
out_VersionPtr	Pointer to the buffer where the file version is stored (size is one byte)
out_MaxRecordsPtr	Pointer to the buffer where the number of available records is copied

Return values:

Name	Description
DRV_OK	Function completed successful
PCM_INVALID_FILE	Unknown file name
PCM_INVALID_SIZE	File sizes differ
PCM_INVALID_RECORD	Invalid record number
PCM_INVALID_CKSM	Invalid checksum

Description

The function reads a record of a linear fixed elementary file. Therefore, the copy of the permanent configuration memory in the RAM is accessed. If the file exists, the content of the requested record is copied into the buffer provided. The version of the file is copied into the buffer to which out_VersionPtr points. The function returns PCM_OK

The valid range of record numbers is 1 to *max_records*. The maximum number of records available is returned.

If the file is unknown, the function returns PCM_INVALID_FILE.

If the file is known, but the file sizes differ, the function returns PCM_INVALID_SIZE.

If the requested record number is invalid, the function returns with the value PCM_INVALID_RECORD.

If the checksum of the file is invalid, the function returns PCM_INVALID_CKSM.

2.3.6 pcm_WriteFile – Write transparent elementary file

Definition:

```
UBYTE pcm_WriteFile
(
        UBYTE *          in_FileName
        USHORT           in_BufferSize
        UBYTE *          in_BufferPtr
);
```

Parameters:

Name	Description
in_FileName	Elementary file name
in_BufferSize	Size of the buffer containing the file
in_BufferPtr	Pointer to the buffer containing the file

Return values:

Name	Description
DRV_OK	Function completed successful
PCM_INVALID_FILE	Unknown file name
PCM_INVALID_SIZE	File sizes

Description

The function writes a transparent elementary file. Therefore, the copy of the permanent configuration memory in the RAM is accessed. If the file exists, the content of the file is copied into the permanent configuration memory. The function returns PCM_OK.

NOTE: The file is not stored in the permanent configuration memory of the non volatile RAM. Call the pcm_Flush() function to save all changes to the non volatile RAM.

If the file is unknown, the function returns PCM_INVALID_FILE.

If the file is known, but the file sizes differ, the function returns PCM_INVALID_SIZE.

2.3.7 pcm_WriteRecord – Write a linear fixed elementary file

Definition:

```
UBYTE pcm_WriteRecord
(
        UBYTE *         in_FileName
        USHORT          in_Record
        USHORT          in_BufferSize
        UBYTE *         in_BufferPtr
);
```

Parameters:

Name	Description
in_FileName	Elementary file name
in_Record	Number of the record to read
in_BufferSize	Size of the buffer containing the record
in_BufferPtr	Pointer to the buffer containing the record

Return values:

Name	Description
DRV_OK	Function completed successful
PCM_INVALID_FILE	Unknown file name
PCM_INVALID_SIZE	File sizes differ
PCM_INVALID_RECORD	Invalid record

Description

The function writes a record of a linear fixed elementary file. Therefore, the copy of the permanent configuration memory in the RAM is accessed. If the file exists, the content of the buffer is copied into the indicated record of the permanent configuration memory. The function returns PCM_OK

The valid range of record numbers is 1 to *max_records*. The maximum number of records can be requested by calling the function pcm_ReadRecord.

NOTE: The record is not stored in the permanent configuration memory of the non volatile RAM. Call the pcm_Flush() function to save all changes to the non volatile RAM.

If the file is unknown, the function returns PCM_INVALID_FILE.

If the file is known, but the file sizes differ the function returns PCM_INVALID_SIZE.

If the indicated record number is invalid, the function returns with the value PCM_INVALID_RECORD.

2.3.8 pcm_Flush – Transfer changes to the PCM of the non volatile RAM

Definition:

```
UBYTE pcm_Flush
(
        void
);
```

Parameters:

Name	Description
-	-

Return values:

Name	Description
DRV_OK	Function completed successfully
PCM_ERASE_ERROR	erase failed
PCM_WRITE_ERROR	write failed

Description

The function transfers all changes made to the PCM in the RAM to the PCM of the non volatile RAM. If the data could not be transferred, the function returns PCM_FAIL.

3 Application Example

The following code fragment shows how RR reads the mobile station capabilities and how it uses this information for checking incoming information.

```
#include "pcm.h"                        /* include prototypes and constants */

GLOBAL void rr_csf_ms_cap (T_RR_DATA * rr_data)
{
  UBYTE version;

  /*
   * Read the elementary field MSCAP and store the
   * information
   */
  pcm_ReadFile ((UBYTE*)EF_MSCAP_ID, SIZE_EF_MSCAP,
                (UBYTE*)&rr_data->mscap, &version);
}



GLOBAL void for_check_channel_mode (UBYTE      ch_mod,
                                    T_RR_DATA *rr_data)
{
  switch (ch_mod)
  {
    case 0x03:                    /* data 12  k         */
    case 0x0B:                    /* data 6   k         */
    case 0x0F:                    /* data 6   k         */
    case 0x13:                    /* data 3.6 k         */
    case 0x17:                    /* data 3.6 k         */
      if (FldGet(rr_data->mscap.datCap1, datSup) EQ 0)
        for_set_content_error (RRC_CHANNEL_MODE, rr_data);
      break;
    case 0x21:                    /* enhanced full rate */
      if (FldGet(rr_data->mscap.chnMode, EFRSupV2) EQ 0)
        for_set_content_error (RRC_CHANNEL_MODE, rr_data);
      break;
    case 0x05:                    /* speech half rate   */
      if (FldGet(rr_data->mscap.chnMode, hrSup) EQ 0)
        for_set_content_error (RRC_CHANNEL_MODE, rr_data);
      break;
    case 0x00:                   /* signaling only    */
    case 0x01:                   /* speech full rate   */
      break;
    default:
      for_set_content_error (RRC_CHANNEL_MODE, rr_data);
      break;
  }
}
```

4 Transition towards Flash File System (FFS)

4.1 Intention and scope

In the long term, data logically stored in the PCM will finally be moved to the Flash File System (FFS) and PCM will become obsolete and FFS will become the only non-volatile storage. However, during a transition period, some non-volatile data will be have to be kept (logically) in PCM which is described in the previous chapters of this document and other data will have to be kept in the FFS. The following chapters provide information on the latter category of data until PCM is becoming obsolete.

4.2 Data held in FFS

4.2.1 Identification of data held in FFS

The following data mentioned in chapters 4.2.1.x will be held in FFS and not in PCM already during the transition period mentioned in chapter 4.1. The description will be in the previous chapters.

4.2.1.1 EF_RFCAP – Mobile Station RF Capabilities

The full file name of this data will be

/gsm/com/rfcap

. For details on the semantics see also chapter 2.1.8.

4.2.2 Format of data held in FFS

The format of data held in FFS will be as described in chapter 2.1.x with the exception that the first 2 bytes (checksum and version) will *not* be part of the data which shall be stored in FFS. For example, the data "EF_RFCAP", see chapters 4.2.1.1 and 2.1.8, will start with the "set bands" byte and consist of SIZE_EF_RFCAP bytes (and not of SIZE_EF_RFCAP + 2 bytes).

4.3 Example settings of data held in the FFS

4.3.1 EF_RFCAP – Mobile Station RF Capabilities

An example setting for this file is

00 0B 41 00 00 00 00 00 50 00 00 A5 05 00 C0 00

corresponding to

- **RF capabilities, set bands: 0x00 = automatic band selection mode**

Octet	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**
1	R-GSM	GSM 480	GSM 450	GSM 850	E-GSM	PCS1900	DCS1800	GSM 900
	0	0	0	0	0	0	0	0

- **RF capabilities, bands: 0x0B = dual band extended**

Octet	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**
2	R-GSM	GSM 480	GSM 450	GSM 850	E-GSM	PCS1900	DCS1800	GSM 900
	0	0	0	0	1	0	1	1

	Value	**Description**
GSM 900	1	GSM 900 is supported
DCS 1800	1	DCS 1800 is supported
PCS 1900	0	PCS 1900 is not supported
E-GSM	1	E-GSM is supported (includes GSM 900)
GSM 850	0	GSM 850 is not supported
GSM 450	0	GSM 450 is not supported
GSM 480	0	GSM 480 is not supported
R-GSM	0	Railway-GSM is not supported

- **RF capabilities, power 1: 0x41 = power class 4 for GSM 900, class 1 for DCS 1800**

Octet	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**
3	Power Class GSM 900				Power Class DCS 1800			
	0100				0001			

	Value	**Description**
Power Class GSM 900	4	Power Class 4
Power Class DCS 1800	1	Power Class 1

- **RF capabilities, power 2: 0x00 = neither PCS 1900 nor GSM 850 supported**

Octet	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**
4	Power Class PCS 1900				Power Class GSM 850			
	0000				0000			

	Value	**Description**
Power Class PCS 1900	0	PCS 1900 is not supported
Power Class GSM 850	0	GSM 850 is not supported

- **RF capabilities, power 3: 0x00 = neither GSM 400 nor EDGE supported**

Octet	8	7	6	5	4	3	2	1
5	Power Class GSM 400				EDGE Power Class 1		EDGE Power Class 2	
	0000				00		00	

	Value	Description
Power Class GSM 400	0	Neither GSM 450 nor GSM 480 are supported
EDGE Power Class 1	0	No EGDE RF Power Capability 1
EDGE Power Class 2	0	No EGDE RF Power Capability 2

- **RF capabilities, msGSM: 0x00 = no GSM multi slot (i.e. HSCSD) supported**

Octet	8	7	6	5	4	3	2	1
6	ms_class					0	0	0
	00000					0	0	0

	Value	Description
ms_class	0	MS does not support the use of multiple timeslot

- **RF capabilities, msEDGE: 0x00 = no EDGE multi slot supported**

Octet	8	7	6	5	4	3	2	1
7	egde_ms_class					0	0	0
	00000					0	0	0

	Value	Description
edge_ms_class	0	EDGE MS does not support the use of multiple timeslot

- **RF capabilities, msHSCSD: 0x00 = no HSCSD multi slot supported**

Octet	8	7	6	5	4	3	2	1

8	hscsd_ms_class					0	0	0
	00000					0	0	0

	Value	Description
hscsd_ms_class	0	HSCSD MS does not support the use of multiple timeslot

- **RF capabilities, msGPRS: 0x50 = GPRS multi slot class 10, no DTM support**

Octet	8	7	6	5	4	3	2	1
9	gprs_ms_class					dtm_g	dtm_g_ms_class	
	01010					0	00	

	Value	Description
gprs_ms_class	10	Multi Slot Class 10
dtm_g	0	GPRS MS does not support GPRS Dual Transfer Mode (DTM)
dtm_g_ms_class	0	Sub-Class 1 supported

- **RF capabilities, msECSD: 0x00 = no ECSD multi slot supported**

Octet	8	7	6	5	4	3	2	1
10	ecsd_ms_class					0	0	0
	00000					0	0	0

	Value	Description
ecsd_ms_class	0	ECSD MS does not support the use of multiple timeslot

- **RF capabilities, msEGPRS: 0x00 = no EGPRS multi slot supported, no DTM support for EGPRS**

Octet	8	7	6	5	4	3	2	1

11	egprs_ms_class	dtm_e	dtm_e_ms_class
	00000	0	00

	Value	Description
egprs_ms_class	0	EGPRS MS does not support the use of multiple timeslot
dtm_e	0	EGPRS MS does not supports EGPRS Dual Transfer Mode (DTM)
dtm_e_ms_class	0	Sub-Class 1 supported

- **RF capabilities, capability 1: 0xA5 = early classmark, no pseudo-sync HO, MT SMS, no LCS, no SOLSA, CM service prompt, no EDGE modulation, fixed allocation**

Octet	8	7	6	5	4	3	2	1
12	es_ind	ps	mt_pp_sms	lcsva	solsa	cmsp	mod	mac_support
	1	0	1	0	0	1	0	1

	Value	Description
es_ind	1	“Controlled Early Classmark Sending” is implemented
ps	0	Pseudo Synchronisation capability is not present
mt_pp_sms	1	MS does support mobile terminated point to point SMS
lcsva	0	LCS value added location request notification not supported
solsa	0	MS does not support SoLSA
cmsp	1	MS does supports CM service Prompt (network initiated MO CM connection request)
mod	0	The EDGE Modulation Capability indicates the supported modulation scheme by MS in addition to GMSK. 8-PSK is supported for downlink reception only
mac_support	1	MS supports Dynamic and Fixed Allocation

- **RF capabilities, capability 2: 0x05 = no switching times, no ext. measurements, no COMPACT, no VBS, no VGCS, UCS2 support, screening indicator of phase 2**

Octet	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**
13	meas	ext_meas	compact	vbs	vgcs	ucs2_treat	ss_screen	
	0	0	0	0	0	1	01	

	Value	**Description**
meas	0	Indicates that no IE, e.g. classmark 3, shall contain any value (see sms_value and sm_value below in the next byte) about the measurement capabilities
ext_meas	0	The MS does not supports "Extended Measurement (on SACCH)"
compact	0	The MS does not support COMPACT Interference Measurement
vbs	0	No VBS capability and no notifications wanted
vgcs	0	No VGCS capability and no notifications wanted
ucs2_treat	1	Indicates the likely treatment by the MS of UCS2 encoded character strings. The ME has no preference between the use of the default alphabet and the use of UCS2.
ss_screen	1	Capability of handling of ellipsis notation and phase 2 error handling

- **RF capabilities, switch measure: 0x00**

Octet	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**
14	sms_value				sm_value			
	0000				0000			

	Value	**Description**
sms_value	0	¼ timeslot (~144 ms)
	1	2/4 timeslot (~288 ms)
	2	¾ timeslot (~433 ms)
	3	4/4 timeslot
	…	
	14	15/4 timeslot
	15	16/4 timeslot (~2307 ms)
sm_value	0..15	the same values as sms_value

- **RF capabilities, encryption: 0xC0 = only A5/1 and A5/2 available**

Octet	8	7	6	5	4	3	2	1
15	A5/1	A5/2	A5/3	A5/4	A5/5	A5/6	A5/7	0
	1	1	0	0	0	0	0	0

	Value	Description
A5/1	1	Encryption algorithm A5/1 available
A5/2	1	Encryption algorithm A5/2 available
A5/3	0	Encryption algorithm A5/3 not available
A5/4	0	Encryption algorithm A5/4 not available
A5/5	0	Encryption algorithm A5/5 not available
A5/6	0	Encryption algorithm A5/6 not available
A5/7	0	Encryption algorithm A5/7 not available
	0	reserved

- **RF capabilities, positioning and other things: 0x00 = no positioning support**

Octet	8	7	6	5	4	3	2	1
16	assist eotd	based eotd	assist gps	based gps	conv gps	gprs_eda	egprs_eda	0
	0	0	0	0	0	0	0	0

	Value	Description
assist_eotd	0	MS does not support assisted E-OTD as positioning method
based_eotd	0	MS does not support based E-OTD as positioning method
assist_gps	0	MS does not support assisted GPS as positioning method
based_gps	0	MS does not support based GPS as positioning method
conv_gps	0	MS does not support conventional GPS as positioning method
gprs_eda	0	reserved for future use for GPRS Extended Dynamic Allocation Capability
egprs_eda	0	reserved for future use for EGPRS Extended Dynamic Allocation Capability
	0	reserved