SIMLOCK HOW TO

Department:	WTBU - Cellular Systems		
Creation Date:	2005-11-02		
Last Modified:	2005-11-02 by Durgesh		
ID:	SIML_Howto	Version:	0.1
Status:	Draft	ECCN:	

0 Document Control

Texas Instruments Incorporated and / or its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products, software and services at any time and to discontinue any product, software or service without notice. Customers should obtain the latest relevant information during product design and before placing orders and should verify that such information is current and complete.

All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment. TI warrants performance of its products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI products, software and / or services. To minimize the risks associated with customer products and applications, customers should provide adequate design, testing and operating safeguards.

Any access to and / or use of TI software described in this document is subject to Customers entering into formal license agreements and payment of associated license fees. TI software may solely be used and / or copied subject to and strictly in accordance with all the terms of such license agreements.

Customer acknowledges and agrees that TI products and / or software may be based on or implement industry recognized standards and that certain third parties may claim intellectual property rights therein. The supply of products and / or the licensing of software do not convey a license from TI to any third party intellectual property rights and TI expressly disclaims liability for infringement of third party intellectual property rights.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products, software or services are used.

Information published by TI regarding third–party products, software or services does not constitute a license from TI to use such products, software or services or a warranty, endorsement thereof or statement regarding their availability. Use of such information, products, software or services may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

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 TI.

0.1 Export Control Statement

Recipient agrees that it will not knowingly export or re-export, directly or indirectly, any product or technical data (as defined by the U.S, EU and other Export Administration Regulations) including software, or any controlled product restricted by other applicable national regulations, received from Disclosing party under this Agreement, or any direct product of such technology, to any destination to which such export or re-export is restricted or prohibited by U.S or other applicable laws, without obtaining prior authorisation from U.S. Department of Commerce and other competent Government authorities to the extent required by those laws. This provision shall survive termination or expiration of this Agreement.

According to our best knowledge of the state and end-use of this product or technology, and in compliance with the export control regulations of dual-use goods in force in the origin and exporting countries, this technology is classified as given on the front page.

This product or technology may require export or re-export license for shipping it in compliance with certain countries regulations.

0.2 Document History

Date	Changed by	Approved by	Version	Status	Notes
2005-Nov-02	Durgesh S Alageri		0.1	Draft	1

Notes:

1. Initial version.

0.3 References, Abbreviations, Terms

1. Personalization_LLD.doc
2. Personalization_HLD.doc
3. 3GPP TS 22.022 Personalization of ME

Table of Contents

1 How to Create MEPD Raw Data

Fill the data field in a text file and save it as <MEPD_File>.properties file. The details of the MEPD data fields to be entered are explained at length in the section 4. A sample of MEPD_File.properties (All the categories are disabled) file is attached below.

Note: This MEPD_File is created with all the personalization category disabled. It is for ease of team members don't want to bother about personalization feature and want the mobile to boot up without any hassles of entering passwords etc.

2 How to generate MEPD Bin file

It requires GenMEPD tool. The following steps are done to generate MEPD bin file

a. Create the MEPD.properties file. (mentioned in section 1)

b. Run the following commands in mepd_tool directory to convert the MEPD_File.properties file to MEPD_File.bin

Java GenMEPD <MEPD_File>.properites MEPD_File.bin

A sample generated MEPD_File.bin is attached below (for the above <MEPD_File>.properties file):

3 How to Store MEPD in ME

There are two ways of storing MEPD Data to mobile depending on usage of non secure or secured release of security driver. For non secure release of security driver MEPD Data can be written directly in the FFS. For secured release of security driver MEPD Data can be bind to the target using CSST tool.

3.1 Non Secured

It require ETM tool. The following steps are done to write MEPD data to FFS.

a. Copy generated MEPD_File.bin to etm directory

b. Run the command **tmsh** in etm directory.

c. If the target build traces is through USB, open the tmsh shell by running the command **tmsh –pn** in etm directory where 'n' is the port number. (E.g. : If USB is connected to COM4 then run tmsh –p4 in etm directory)

d. If FFS is not formatted then board will not boot up. To format FFS do the following steps

   i. Remove the sim card from the board

   ii. Restart the board

   iii. Wait until MMI asks Insert Sim

   iv. Connect USB/ UART

   v. Do the step 4 or 5 depending on USB / UART

vi. Run the command **mkfs –f** in tmsh shell

e. Write the **MEPD_File.bin** to FFS by running following command in tmsh shell

fwr MEPD_File.bin MEPD_File.bin

3.2 Secured (For Locosto Release)

It requires CSST I-Sample Release 1.001 tool.

In order to bind an MEPD file to target, rename **MEPD_File.bin** to **MEPD_File.raw**. Start the CSST tool by choosing "Start →All Programs → Texas Instruments → CSST I-Sample 1.001 → CSST GUI" Then choose BOOT ROM in the toolbar, double click on the Download task in the left pane, and press the Bind tab. Specify the binary MEPD file and the start address and tick the "Enable encryption" checkbox. Note that the .raw extension must be used for binary files, as .bin is used for WinCE binaries in CSST. The start address is 0x066F0000 (i.e., the last flash sector before the FFS).

CSST I-Sample Release 1.001 can also be used for binding IMEI to Locosto GP devices even if the CSST GUI has not been updated yet. A hex editor is needed in which a binary file (.raw) can be created. If the IMEI is 1122334455667788 then the contents of the binary file must be 0x44 0x33 0x22 0x11 0x88 0x77 0x66 0x55. Finally, specify the newly created binary file and 0x066EFF00 as start address for the platform certificate and disable encryption on the Bind tab in CSST.

A sample of imei.raw file is attached below

4 Details of Internal MEPD structure

This section describes the structure used internally by the security driver for the MEPD data. For the initial (unsecured) release of the driver, MEPD test data in this format can be written directly in the FFS. Once the secure version is available, the MEPD data will be stored in encrypted form outside the FFS.

4.1 Data sections

The internal data is structured as a series of sections, *each aligned to the nearest 8 byte boundary*. The reason for this alignment is the nature of the used encryption algorithm.

The following table depicts the data sections as they come in the file, where "n" represents the total number of categories stored in the structure.

Internal sections
Global configuration
Failure count key
MEPD configuration
Category 0 (NW)
Category 1 (NS)
Category 2 (SP)

Category 3 (CP)
Category 4 (SIM)
Category 5 (AP)
Category X (n-6 times)

Each category is in turn divided in 3 subsections, *each aligned to the nearest 8 byte boundary*.

Category section
Category key
Category header
Category body

4.2 Global configuration

This section is used internally by the security driver only. Its main use is to distinguish between 2 different formats; 1) The format coming from the operator that needs to be converted into the format described here. 2) The format as described here.

If this section is not initialized correctly, all API functions will return SEC_DRV_RETURN_NotPresent.

Global configuration		
Type	Name	**Value**
UINT8	firstboot_pattern	0xAA

4.3 Failure count key

Each key (that is; not only this one) is structured as a C-string ending with a '\0' termination, unless the length is at maximum in which case there is no room for the termination character. The maximum length is 16 chars.

Key		
Type	Name	**Value**
char[16]	digit	String used as key. Should contain 16 valid chars at all times, as this is what first boot will calculate!

4.4 MEPD Configuration

The following table shows the layout of the MEPD configuration section:

Offset (bytes)	Field	Size (bytes)	Description	Ref
0x0000	Flags	2	Keeps flags for customer specific behavior	TR8.3, 5.2, 5.3, 8.1, 6.7, 6.8
0x0002	AddNewIMSI	2	Describes the Personalization process	ETSI standard req, TR 5.6
0x0004	NumCategories	1	Contains number of categories, stored in MEPD	TR1.1
0x0005	FC_Max	1	Failure Counter maximum. 0xFF to disable the counter	TR10.1, 6.4, 6.5, 6.6
0x0006	FC_Current	1	Current value of the Failure Counter	TR10.1, 6.4, 6.5, 6.6
0x0007	FC_Key_Length	1	Length of generated FC Key	TR8.4
0x0008	FC_Reset_Fail_Max[1]	1	Maximum number of allowed failed FC resets. 0xFF to disable the counter.	TR10.1, 6.4, 6.5, 6.6
0x0009	FC_Reset_Fail_Current[1]	1	Current number of failed FC resets	TR10.1, 6.4, 6.5, 6.6
0x000a	FC_Reset_Success_Max[1]	1	Maximum number of allowed successful FC resets. 0xFF to disable the counter	TR10.1, 6.4, 6.5, 6.6
0x000b	FC_Reset_Success_Current[1]	1	Current number of successful FC resets	TR10.1, 6.4, 6.5, 6.6
0x000c	MNC_Len	1	Length of the MNC	TR11.2
0x000d	GID1_Len	1	Length of the GID1	TR11.3 (was later reduced to 4)
0x000e	GID2_Len	1	Length of the GID2	TR11.3 (was later reduced to 4)
0x000f	TypeAprovalSIM	1	Describes the behavior of the test SIM according to TR 4.1	TR 4.1
0x0010	TestCPHS	1	Describes the behavior of the test SIM according to TR 4.2	TR 4.2

4.4.1 Flags

Name: Flags				
Bit Nr	Name	Description	Values	Ref
0	Reserved			
1	Reserved			
2	SEC_DRV_HDR_FLAG_Truncation	Used to specify if Control Key shall be truncated during all SML control keys	1: truncate key to 8 b	TR8.3

[1] Place will be reserved. API and code change will be provided on official CR from the customer.

		verification Used to specify if Control Key shall be truncated during OTA de-personalization	0: use full length	
3	SEC_DRV_HDR_FLAG_ETSI_Flag	Used to specify Lock behaviour (ETSI or Customer Specific)	1: ETSI 0: Customer	TR5.2, 5.3
4	SEC_DRV_HDR_FLAG_Spec_Lock_Key	Used to specify if specific lock key used	1: Spec. key used 0: Spec. Key not used	TR8.1
5	SEC_DRV_HDR_FLAG_LAM_Unlock	Used to specify LAM unlock (all unlock passwords are considered as wrong)	1: LAM unlock active 0: LAM unlock off	TR6.7
6	SEC_DRV_HDR_FLAG_No_SIM_Unlock	Used to set if ME de-personalization shall be allowed without SIM inserted	1: If no SIM, no unlock possible 0: If no SIM, unlock possible	TR6.8
7	SEC_DRV_HDR_FLAG_Airtel_Ind	Is used to indicate to MMI that Airtel behavior is active. This Flag is newer processed by ACI. Sub-flags as SEC_DRV_HDR_FLAG_No_SIM_Unlock or SEC_DRV_HDR_FLAG_Spec_Lock_Key always have first priority	1: Airtel behavior 0: standard behavior	TR8.1, 6.8
8	SEC_DRV_HDR_FLAG_Unlock_Timer[2]	To inform MMI that there should be timer between Unlocking attempts	1: Timer shall be used 0: No timer shall be used	TR10.3

Examples

0x08 – Truncation Off, ETSI behaviour, No Specific Lock Keys, LAM unlock Off

Correct:

0x14 - Truncation On, Customer behaviour, Specific Lock Keys, LAM unlock Off, Airtel Off. Sub-flags for Specific lock or No SIM Unlock can be used separately

0x94 - Truncation On, Customer behaviour, Specific Lock Keys, LAM unlock Off, Airtel On. Sub-flags for Specific lock or No SIM Unlock can be used separately

Incorrect

[2] Place will be reserved. API and code change will be provided on official CR from the customer.

0x80 - Truncation Off, Customer behaviour, No Specific Lock Keys, LAM unlock Off, Airtel On. Airtel flag cannot be used separately from sub flags

4.4.2 AddNewIMSI

This field is used to specify if new code group shall be added to MEPD while doing ME Personalization process. Thus if we try to apply anti-theft protection (SIM Personalization) to ME, we need to add IMSI of currently inserted SIM into MEPD. Optionally customer can prevent adding new code group to MEPD. Thus, if bit 4 is set to 1, new IMSI will be added to MEPD

Name: AddNewIMSI			
Bit Nr	Name	Description	Values
0	ADD_NW_CODE	While locking ME add NW new code group to MEPD	1: Add new code group 0: Don't add new code group
1	ADD_NS_CODE	While locking ME add NS new code group to MEPD	1: Add new code group 0: Don't add new code group
2	ADD_SP_CODE	While locking ME add SP new code group to MEPD	1: Add new code group 0: Don't add new code group
3	ADD_CP_CODE	While locking ME add CP new code group to MEPD	1: Add new code group 0: Don't add new code group
4	ADD_SIM_CODE	While locking ME add SIM new code group to MEPD	1: Add new code group 0: Don't add new code group
5 to 15		Describes the Personalization process behavior for proprietary categories	1: Add new code group 0: Don't add new code group
Examples			
0x0000	User can not add code groups of currently inserted SIM into MEPD		
0x1f	User can add code groups of currently inserted SIM into MEPD for all standard categories		
0x10	User can add code group of currently inserted SIM into MEPD only for SIM personalization (most useful)		

4.4.3 NumCategories

Name: NumCategories			
Size	Name	Description	Values
1	NumCategories	contains number of categories, stored in MEPD	For current implementation shall be

			always set to 5
Examples			
0x05	keeps 5 categories		

4.4.4 FC_Max

Name: FC_Max			
Size	Name	Description	Values
1	FC_Max	Contains maximum value of failure counter	0-0xff
Examples			
0x03	User can try maximum of 3 attempts to enter correct key		

4.4.5 FC_Current

Name: FC_Current			
Size	Name	Description	Values
1	FC_Current	Contains current value of failure counter	Shall be initially set to 0
Examples			
0x02	User entered the wrong key in 2 attempts		

4.4.6 FC_Key_Length

Name: FC_Key_Length			
Size	Name	Description	Values
1	FC_Key_Length	Specifies length of FC ke, that shall be generated	6-16
Examples			
0x08	Length of generated FC key shall be 8 digits		

4.4.7 FC__Reset_Fail_Max

Name: FC_Reset_Fail_Max			
Size	Name	Description	Values
1	FC_Reset_Fail_Max	Maximum number of allowed failed FC resets. 0xFF to disable the counter.	0-0xff
Examples			
0x03	User can try maximum of 3 attempts to reset FC with wrong key.		

4.4.8 FC_Reset_Fail_Current

Name: FC_Reset_Fail_Current			
Size	Name	Description	Values
1	FC_Reset_Fail_Current	Current number of failed FC resets	Shall be initially set to 0
Examples			
0x02	User entered the wrong key in 2 attempts		

4.4.9 FC__Reset_Success_Max

Name: FC_Reset_Success_Max			
Size	Name	Description	Values
1	FC_Reset_Success_Max	Maximum number of allowed successful FC resets. 0xFF to disable the counter	0-0xff
Examples			
0x03	User can try maximum of 3 attempts to reset FC with wrong key.		

4.4.10 FC_Reset_Success_Current

Name: FC_Reset_Success_Current			
Size	Name	Description	Values
1	FC_Reset_Success_Current	Current number of successful FC resets	Shall be initially set to 0
Examples			
0x02	User entered the wrong key in 2 attempts		

4.4.11 MNC_Len

Name: MNC_Len			
Size	Name	Description	Values
1	MNC_Len	Contains length of the MNC	
Examples			
0x02	MNC code has 2 digits		
0x03	MNC code has 3 digits		

4.4.12 GID1_Len

Name: GID1_Len			
Size	Name	Description	Values
1	GID1_Len	Contains length of the GID1 file, which shall be read from SIM	

Examples	
0x04	Length of the GID1 is 4 bytes

4.4.13 GID2_Len

Name: GID2_Len			
Size	Name	Description	Values
1	GID2_Len	Contains length of the GID2 file, which shall be read from SIM	
Examples			
0x04	Length of the GID2 is 4 bytes		

4.4.14 TypeAprovalSIM

Name: TypeAprovalSIM			
Size	Name	Description	Values
1	TypeAprovalSIM	Describes the behavior of the test SIM according to TR 4.1	0 - always accepted 1 - always rejected 2 - accepted until normal SIM is inserted
Examples			
0x00	Test SIM is always accepted and no check is done		
0x01	Test SIM is always rejected - if any lock category is active, Test SIM will be rejected		
0x02	Accepted until normal SIM is inserted - as soon as normal SIM is inserted.		

Comment [TJH1]: the nos. have to be changed in the code)

4.4.15 TestCPHS

Name: TestCPHS			
Size	Name	Description	Values
1	TestCPHS	Describes the behavior of the test SIM according to TR 4.2	0 - always accepted 1 - always rejected
Examples			
0x00	Test SIM is Always accepted: there is no SIM-ME-LOCK check. The ME is never blocked on a category.		
0x01	Test SIM is Always rejected: a SIM-ME-LOCK check is always performed. The ME goes in normal mode only if all personalized categories are unblocked		

Comment [TJH2]: (the nos. have to be changed in the code)

4.5 Category header

Category Header is represented by yellow part of MEPD Record in Fig. 2-2 of HLD Security Driver.

Offset	Field	Size	Description	Ref
0x0000	Status	1	Status defines for the different lock-categories	TR14.5, 5.x, 6.x
0x0001	Flags	1	Different flags for the lock-categories	TR6.2
0x0002	Dependency	2	Each bit represents a category that is a child of this one. Children locks/unlocks with their parents.	TR6.2
0x0004	DataLen	2	Number of bytes present in the category body.	
0x0006	KeyLength	1	Length of generated control key	TR8.4

4.5.1 Status

Name: Status			
Size	Name	Description	Values
1	Status	Status defines for the different lock-categories	0 - SEC_DRV_CAT_STAT_Unlocked 2 - SEC_DRV_CAT_STAT_Locked
Examples			
0x00	Category is unlocked.		
0x02	Category is locked. Writing of category body no longer possible.		

4.5.2 Flags

Name: Flags			
Bit Nr	Name	Description	Values
1	SEC_DRV_CAT_FLAG_LinkLocked	if the category is dependent on a parent. Disable locking/unlocking on its own! TR6.2	0 – category can be unlocked on its own 1 – category can be only unlocked by unlocking parent category
2	SEC_DRV_CAT_FLAG_UseSeed	Use seed to generate key during first boot	1 – A seed is needed for this category.
Examples			
0x01	if the category is dependent on a parent. Disable locking/unlocking on its own!		
0x02	Use seed to generate key during first boot		

4.5.3 Dependency

Dependency flag is introduced to implement Linking between categories (TR6.2) Every bit in Dependency field points to chilled category, which is linked to actual one.

Name: Dependency			
Size	Name	Description	Values
2	Dependency	Each bit represents a category that is a child of this one. Children locks/unlocks with their parents.	0/1 for every bit
Examples			
If Dependency flag of SIM category is set to 0x08 (pointing to CP) and SEC_DRV_CAT_FLAG_LinkLocked bit of Flags field (please see "Flags" description) of CP category is set to 1, it would mean SIM-C lock (while unlocking SIM, CP will automatically be unlocked, CP cannot be unlocked separately from SIM) If Dependency flag of NS category is set to 0x04 (pointing to SP) and SEC_DRV_CAT_FLAG_LinkLocked bit of Flags field (please see "Flags" description) of SP category is set to 0, it would mean NS-SP lock (while unlocking NS, SP will automatically be unlocked, SP can be unlocked separately from NS) !!! Please be aware that by this mechanism customer can set any combination of linked/dependent personalization types. On the other hand it is in customer responsibility to guarantee that proper combination of Dependency/Flags field is stored on MEPD			

4.5.4 DataLen

Name: DataLen			
Size	Name	Description	Values
2	DataLen	Number of bytes present in the category body.	
Examples			
0x08	Length of the category body is 8 bytes		

4.5.5 KeyLength

Name: KeyLength			
Size	Name	Description	Values
1	KeyLength	Length of the key, which shall be generated by Security Driver on first boot in bytes	6-16
Examples			
0x08	Length of the control key shall be 8 digits		

4.6 Category Body

This is a sequence of bytes with no meaning to the security driver itself. These data are to be initialized and used by the ACI layer. The data however, can only be written if the corresponding category is unlocked.

The maximum number of bytes available is defined by the constant SEC_DRV_CAT_MAX_SIZE, currently set to 256 bytes.

When stored in the file/flash, these data must always occupy the maximum available space. That is, even though a category does not have any data, SEC_DRV_CAT_MAX_SIZE bytes should be reserved (but the DataLen field in the category header should be set to 0).

Following points should be noted for storing data in the MEPD.

1) Network, Network Subset & MSIN codes shall be stored in BCD format, just as present in the IMSI value.
2) GID 1 & GID2 values shall be stored just as stored in the Elementary file.
3) The parity nibble shall be included when specifying the network code. (A parity nibble does not need to be considered when specifying regular expressions.)
4) Regular expressions shall be stored in the ASCII format for simplicity of code and easy understanding.
5) As mentioned in the structures below, the part above 'Current user code index' shall be configurable. The part from 'Current user code index' (shaded in blue) shall be maintained by ACI.
6) Operator codes shall be identified by a code type:

   0x0a indicating Normal code,

   0x0b indicating Range,

   0x0c indicating Regular Expression

   0x0d indicating 8th digit intervals (Only for Network Subset category)

The following are the MEPD structures for each category:

4.6.1 Network category:

A network code (MCC+MNC) consists of 5/6 digits and a parity nibble.

One normal network code would occupy 4 bytes for storage including parity nibble.

A Range would require 8 bytes:

4 bytes for Start value + 4 bytes for End value of range.

A Regular Expression would require 6 bytes

The following structure depicts how the Network category codes will be stored in the MEPD

	Bytes
Max number of User Codes (To be configured)	1

Number of Operator Codes (To be configured)	1
Operator Code-group length (To be set during configuration) **where length = total number of bytes occupied by operator codes**	1
Code type = 0x0a (Type identifier for normal code)	1
Normal code (Including parity nibble)	4
Code type = 0x0b (Type identifier for range)	1
Start value (Including parity nibble)	4
End value (Including parity nibble)	4
Code type = 0x0c (Type identifier for regular expression)	1
Regular Expression	6
…	
…	
..	
~~**Current user code index**~~ ~~**(for FIFO based addition of user codes)**~~**Number of User Codes**	1
Current user code index **(for FIFO based addition of user codes)** **(should be set to 0xff if no user codes are present)**	1
User code1	4
User code 2	4
…	
…	
…	
User code n (n = max number of user codes)	4

Example for storing the Normal network code.

For e.g 1.: A Normal network code “123456” will be stored as follows:

0x19	Byte 1
0x32	Byte 2
0x54	Byte 3
~~0~~0xf6	Byte 4

Digit 9 in byte 1 indicate parity nibble. ~~0x0~~0xf in byte 4 ~~indicate~~indicates unused nibble.

For e.g 2.: A Normal network code “12345” will be stored as follows:

0x19	Byte 1
0x32	Byte 2
0x54	Byte 3
~~0x00~~0xff	Byte 4

Range codes shall also be stored in a similar manner.

For e.g 3.: A regular expression “xx1234” will be stored as follows:

‘x’	Byte 1
‘x’	Byte 2
‘1’	Byte 3
‘2’	Byte 4
‘3’	Byte 5
‘4’	Byte 6

If no user-codes are present, all user-code nibbles should be set to 0xff

4.6.2 Network Subset category:

The personalization check for this category involves Network code check + Network subset code check. Hence we shall store a combination of these 2 codes for this category.

i.e. If Network code is 123456 and NS code is 78, then we shall store 12345678 in the Network subset category.

Based on this, one normal network subset code shall need 5 bytes.

A range of network subset codes would occupy 6 bytes:

4 bytes for NW code + 1 byte for Start value + 1 byte for End value.

A regular expression would occupy 8 bytes. (including network code)

The following structure depicts how a Network subset code can be stored in the MEPD.

	Bytes
Max number of User Codes (To be configured)	1
Number of Operator Codes (To be configured)	1
Operator Code-group length (To be set during configuration)	1
Code type = 0x0a	1
Normal code	5
Code type = 0x0b	1
Network code	4
Start value of NS code	1
End value of NS code	1
Code type = 0x0c	1
Regular expression including Network + NS Code	8
Code type = 0x0d **(8th digit normal code/range)**	1
Network code (Including parity nibble)	4
No. of 8th digit values for which normal codes and/or intervals are to be stored	1
8th digit value	1
No. of normal codes to be associated with the digit (n1)	1
No. of intervals to be associated with the digit (n2)	1
Normal code 1	1
…	1
…	1
Normal code n1	1
Start value of interval 1	1
End value of interval 1	1
..	1
..	1
..	1
Start value of interval n2	1
End value of interval n2	1
Number of User Codes	1
Current user code index **(for FIFO based addition/deletion of user codes)** **(should be set to 0xff if no user codes are present)**	1
User code1	5

User code 2	5
….	
…	
…	
User code n (n = Max number of user codes for NS)	5

Examples for storing Network subset code:

For e.g 1.: A Normal network subset code "12345678" will be stored as follows:

0x19	Byte 1
0x32	Byte 2
0x54	Byte 3
0x76	Byte 4
0xf8	Byte 5

Digit 9 in byte 1 indicate parity nibble. 0xf in byte 5 indicate unused nibble.
(The value of the parity nibble shall be as per the 3GPP standard TS 04.08.)

For e.g 2.: A Normal network subset code "1234567" will be stored as follows:

0x11	Byte 1
0x32	Byte 2
0x54	Byte 3
0x76	Byte 4
0xff	Byte 5

Range codes shall also be stored in a similar manner.

For e.g 3.: A regular expression "12345x1" will be stored as follows:

'1'	Byte 1
'2'	Byte 2
'3'	Byte 3
'4'	Byte 4

'5'	Byte 5
'x'	Byte 6
'1'	Byte 7

Regarding the 8th Digit interval:

Just as Code type = 0x0a indicates that the following code is a normal code,

Code type = 0x0d indicates that the following code is a normal code/range associated with the 8th digit value. The 8th digit value could be any digit from 0 to 9. A normal code or range could be associated with any digit.

Range can also be specified for a few digit values say digit value 0 & digit value 1.

Consider the following example:

A CTS configuration of category codes (with associated 8th digit of IMSI) could be:

Definition of category codes with :
MNC/MCC: 1 plmn
NS8D : values of 8th digit of the IMSI
NS: normal code or interval of NS (HLR) values linked to previous value of 8th digit of the IMSI

MCC;262
MNC;02
NS8D;0 NS;10,19
NS8D;1 NS;10,19
NS8D;1 NS;58,59
NS8D;1 NS;63
NS8D;1 NS;68
NS8D;1 NS;71
NS8D;2 NS;10,29
NS8D;2 NS;35,89

This configuration define, for one PLMN (262 02) :
. 1 interval for NS8D=0
. 2 intervals and 3 normal code for NS8D=1
. 2 intervals for NS8D=2

In this configuration, for example :
. SIM card with N=26202, NS=15 and value of 8th digit = 0 will be authorized
. SIM card with N=12345, NS=15 and value of 8th digit = 0 will be NOT authorized
. SIM card with N=26202, NS=20 and value of 8th digit = 0 will be NOT authorized
. SIM card with N=26202, NS=57 and value of 8th digit = 1 will be NOT authorized
. SIM card with N=26202, NS=59 and value of 8th digit = 1 will be authorized
. SIM card with N=26202, NS=63 and value of 8th digit = 1 will be authorized
. SIM card with N=26202, NS=64 and value of 8th digit = 1 will be NOT authorized
. SIM card with N=26202, NS=63 and value of 8th digit = 1 will be authorized
. SIM card with N=26202, NS=25 and value of 8th digit = 2 will be authorized
. SIM card with N=26202, NS=30 and value of 8th digit = 2 will be NOTauthorized

0x0d	Byte 1	Identifier for 8th digit associated normal codes/range
0x21	Byte 2	

0x26	Byte 3	Network code
0x20	Byte 4	
0xff	Byte 5	
0x03	Byte 6	No of 8th digit values(In the given example, codes for 8th digit values 0,1 & 2 are to be specified. i.e. 3 digits)
0x00	Byte 7	8th digit value=0
0x00	Byte 8	No. of normal codes associated with digit value 0
0x01	Byte 9	No. of intervals associated with digit value 0
0x01	Byte 10	Packed BCD format of NS code 10 & 19 (Interval 1)
0x91	Byte 11	
0x01	Byte 12	8th digit value=1
0x03	Byte 13	No. of normal codes associated with digit value 1
0x02	Byte 14	No. of intervals associated with digit value 1
0x36	Byte 15	Packed BCD format of normal NS code 63
0x86	Byte 16	Packed BCD format of normal NS code 68
0x17	Byte 17	Packed BCD format of normal NS code 71
0x01	Byte 18	Packed BCD format of NS code 10 & 19 (Interval 1)
0x91	Byte 19	
0x85	Byte 20	Packed BCD format of NS code 58 & 59 (Interval 2)
0x95	Byte 21	
0x02	Byte 22	8th digit value=2
0x00	Byte 23	No. of normal codes associated with digit value 2
0x02	Byte 26	No. of intervals associated with digit value 2
0x01	Byte 27	Packed BCD format of NS code 10 & 29 (Interval 1)
0x92	Byte 28	
0x53	Byte 29	Packed BCD format of NS code 35 & 89 (Interval 2)
0x98	Byte 30	

If no user-codes are present, all user-code nibbles should be set to 0xff

4.6.3 SIM Code category:

The personalization check for this category involves the complete IMSI value check. Hence the complete IMSI value shall be stored for this category.

Based on this:

A normal SIM code shall occupy 8 bytes: (parity nibble + 15 digits)

A single range of SIM code would occupy 13bytes:

5 bytes for NW & NS code +

4 bytes for Start MSIN value + 4 bytes for End MSIN value.

A regular expression would occupy 15 bytes.

If no user-codes are present, all user-code nibbles should be set to 0xff

	Bytes
Max number of User Codes (To be configured)	1
Number of Operator Codes (To be configured)	1
Operator Code-group length (To be set during configuration)	1
Code type = 0x0a (Normal code)	1
Normal code1	8
Code type = 0x0b (Range)	1
NW + NS code	5
Start MSIN	4
End MSIN	4
Code type = 0x0c (Regular Expression)	1
Regular Expression	15
Code type = 0x0c (Regular Expression)	1
…	
…	
~~**Current user code index**~~ ~~**(for FIFO based addition/deletion of user codes)**~~**Number of User Codes**	1
Current user code index **(for FIFO based addition/deletion of user codes)** **(should be set to 0xff if no user codes are present)**	1
User code1	8
User code 2	8
….	.
User code n (n = Max number of user codes for SIM category)	8

4.6.4 Service Provider Category:

The personalization check for this category involves Network code check + GID1 value check. Hence we shall store a combination of these 2 codes for this category. GID1 value can be provided in a maximum of 4 bytes.

Based on this, one Normal Service Provider subset code shall need 8 bytes:

4 bytes for Network code + 4 bytes for GID1 value

A single range would occupy 12 bytes:

4 bytes for NW code + 4 bytes for Start GID1 value + 4 bytes for End GID1 value.

A regular expression would occupy 10 bytes (6 bytes for NW code + 4 bytes for GID1 value)

	Bytes
Max number of User Codes (To be configured)	1
Number of Operator Codes (To be configured)	1
Operator Code-group length (To be set during configuration)	1
Code type = 0x0a (Normal code)	1
Network code + GID1 value	8
Code type = 0x0b (Range)	1
Network code	4
Start GID1	4
End GID1	4
Code type = 0x0c (Regular Expression)	1
Regular Expression (including NW code and GID1 value)	10
Code type = 0x0c (Regular Expression)	1
..	
…	
~~**Current user code index**~~ ~~**(for FIFO based addition/deletion of user codes)**~~**Number of User Codes**	1
Current user code index **(for FIFO based addition/deletion of user codes)** **(should be set to 0xff if no user codes are present)**	1
User code1	8

User code 2	8
….	
…	
User code n (n = max number of user codes for SP category)	8

Example for storing SP codes:

For e.g 1.: For a NW code value of ”123456” & GID1 value of 0x1234, the code will be stored as follows:

0x19	Byte 1
0x32	Byte 2
0x54	Byte 3
0xf6	Byte 4
0x12	Byte 5
0x34	Byte 6
0xff	Byte 7
0xff	Byte 8

If no user-codes are present, all user-code nibbles should be set to 0xff

4.6.5 Corporate Category:

The personalization check for this category involves Network code check + GID1 value + GID 2 value check. Hence we shall store a combination of these 3 codes for this category.

Based on this, one Normal Corporate code shall need 12 bytes:

4 bytes for Network code + 4 bytes for GID1 value + 4 bytes for GID2

A single range of Corporate code would occupy 16 bytes:

4 bytes for NW Code + 4 bytes for GID1 value +

4 bytes for GID2 start value + 4 bytes for GID2 End value.

A regular expression would occupy 14 bytes

(Considering 6 bytes for NW Code + 4 bytes for GID1 value + 4 bytes for reg. exp. for GID2/Corporate code.)

If no user-codes are present, all user-code nibbles should be set to 0xff

	Bytes

Max number of User Codes (To be configured)	1
Number of Operator Codes (To be configured)	1
Operator Code-group length (To be set during configuration)	1
Code type = 0x0a (Normal code)	
Normal code	12
Code type = 0x0b (Range)	1
NW code	4
GID1 value	4
GID2 Start value	4
GID2 End value	4
Code type = 0x0c (Regular Expression)	1
Regular Expression	14
Code type = 0x0c	1
…	
…	
~~**Current user code index**~~ ~~**(for FIFO based addition/deletion of user codes)**~~**Number of User Codes**	1
Current user code index **(for FIFO based addition/deletion of user codes)** **(should be set to 0xff if no user codes are present)**	1
User code1	12
User code 2	12
….	.
User code n (n = Max number of user codes for CP category)	12

4.7 CTS-MEPD cross table

	SIM LOCK SETTINGS	Value 1 / Min	Value 2 / Max	Value 3	Custom value (Default value)	MEPD Fi
N	SIMLOCK NCK KeyLength	From 8	To 16		10	"KeyLength" field of NW MEPD record. Key of "KeyLength"-bytes will be generated
	SIMLOCK NCK Status	locked	unlocked		unlocked	"Status" field of NW MEPD record.
	SIMLOCK NCK Automatic add on lock	Y	N		N	AddNewIMSI field of MEPD Configuration
NS	SIMLOCK NSCK Key Length	From 8	To 16		10	"KeyLength" field of NS MEPD record. Key of "KeyLength"-bytes will be generated
	SIMLOCK NSCK Status	locked	unlocked		unlocked	"Status" field of NS MEPD record.
	SIMLOCK NSCK slave	Y	N		N	"Dependency" field of MEPD Record
	SIMLOCK NSCK Automatic add on lock	Y	N		N	AddNewIMSI field of MEPD Configuration
SP	SIMLOCK SPCK Key Length	From 8	To 16		10	"KeyLength" field of SP MEPD record. Key of "KeyLength"-bytes will be generated
	SIMLOCK SPCK Status	locked	unlocked		unlocked	"Status" field of SP MEPD record.
	SIMLOCK SPCK slave	Y	N		N	"Dependency" field of MEPD Record
	SIMLOCK SPCK Automatic add on lock	Y	N		N	AddNewIMSI field of MEPD Configuration
SIM	SIMLOCK PCK KeyLength	From 10	To 16		10	"KeyLength" field of SIM MEPD record. Key of "KeyLength"-bytes will be generated
	SIMLOCK PCK Status	locked	unlocked		unlocked	"Status" field of SIM MEPD record.
	SIMLOCK PCK slave	Y	N		N	"Dependency" field of MEPD Record
	SIMLOCK PCK Automatic add on lock	Y	N		N	AddNewIMSI field of MEPD Configuration
C	SIMLOCK CCK KeyLength	From 8	To 16		10	"KeyLength" field of CP MEPD record. Key of "KeyLength"-bytes will be generated
	SIMLOCK CCK Status	locked	unlocked		unlocked	"Status" field of CP MEPD record.
	SIMLOCK CCK slave	Y	N		N	"Dependency" field of MEPD Record
	SIMLOCK CCK Automatic add on lock	Y	N		N	AddNewIMSI field of MEPD Configuration
	SIMLOCK Key comparaison truncte to 8 digit	Y	N		N	SEC_DRV_HDR_FLAG_Truncation. If set, 8 bytes are compared, otherwise full length. Please be aware since TR specifies truncation only for OTA depersonalization and OTA depersonalization is out of TR, this flag is not used now. Please inform if it is needed to use that flag for other types of unlock
	SIMLOCK Global failure counter	1	0xff		10	FC_Max Field of MEPD Configuration. Disabled, if set to 0xff

	SIMLOCK Global failure counter value	From 0	To 254	Removed	10	FC_Current Field of MEPD Configuration.
	SIMLOCK Reset failure counter	1	0xff		10	FC_Reset_Fail_Max Field of MEPD Configuration. Disabled, if set to 0xff
	SIMLOCK Failure timer	Y	N		N	SEC_DRV_HDR_FLAG_Unlock_Timer
	SIMLOCK Failed reset counter value	From 0	To 254	Removed	10	FC_Reset_Fail_Current Field of MEPD Configuration
	SIMLOCK Used reset counter value	From 0	To 254	Removed	10	FC_Reset_Success_Current Field of MEPD Configuration. Please be aware that there is a FC_Reset_Success_Max, which specifies, how many times user allowed to reset FC succesfuly. Similar to FC_Reset_Fail_Max
	SIMLOCK Categoryunlock disabling	Y	N		N	SEC_DRV_HDR_FLAG_LAM_Unlock
	SIMLOCK behaviour	ALCATEL	ETSI		ALCATEL	SEC_DRV_HDR_FLAG_ETSI_Flag
	SIMLOCK AIRTEL config	Y	N		N	SEC_DRV_HDR_FLAG_Airtel_Ind
	SIMLOCK D2 config	Y	N		N	The 8th digit of IMSI associated with ~~digit or interval~~NS Code is configured by normal codes, range~~,~~ or their combination~~regular expression and plain code or their combination. Please refer HLD for sec drv 55.~~
	SIMLOCK TELEFONICA config	Y	N		N	Realized by "dependency" field and SEC_DRV_CAT_FLAG_LinkLocked flag of MEPD record
	SIMLOCK MNC Key	2 digits	3 digits		2 digits	MNC_Len
	SIMLOCK GID1Size	1 Byte	4 Bytes		1	GID1_Len
	SIMLOCK GID2 Size	1 Byte	4 Bytes		1	GID2_Len
	GSM Test simcard accepted	Y	N	restricted *	Y	TR 4.1: typeApprovalSIM field of MEPD configuration
	CPHS Test simcard accepted	Y	N		Y	TR 4.2: testCPHS field of MEPD configuration

5 External MEPD Format

This section describes the data structure used to represent encrypted MEPD data to be downloaded to the flash of the mobile phone. This format is converted by the first boot process to the structure described in chapter 1.

All fields of more than 8 bits in this format will need to be encoded using the same endianess as the handset.

5.1 Data sections

The data is structured as a series of sections.

External sections	Bits	Description
Seed count	8	Number of seeds needed
DES key + Checksum	1024	RSA encrypted.
Size	16	Included in the checksum.
External lock structure	?	Encrypted using the above random DES key. Included in the checksum.

Seed count:

This is simply an unsigned integer, containing the number of categories in the external lock structure that needs a seed in order to generate a key. NOTE: A seed for the failure counter reset key is always included, so the seed count cannot be less than 1.

If the during first-boot scenario, this number doesn't comply with the actual number of categories that have a set `SEC_DRV_CAT_FLAG_UseSeed` flag, then the handset will reset and act as if no lock file has been flashed.

DES key + checksum:

This is a simple structure as shown below

DES key + checksum		
Type	Name	**Value**
UINT8 [8]	**Key**	These 8 bytes contain a random DES key, used to encrypt the external lock structure.
UINT8 [20]	**Checksum**	This is a 160 bit SHA-1 checksum over the size and the encrypted external lock structure.
UINT8 [100]	Padding	Padding due to the nature of the RSA algorithm.

The structure is encrypted using the RSA algorithm using the manufacturer's private key. NOTE: The 1024 bits stored are not packaged using PKCS#1 (ASN.1/BER) encoding. They are stored like the signatures in the firmware certificate.

Example: Consider the following DES key and SHA-1 digest:

DES key	31 32 33 34 35 36 37 38
SHA-1 digest	73 86 B1 21 25 5F 08 E2 7C 90 3A 1E 77 79 8C FF C8 EB E7 83

Suppose we concatenate these and encrypt the resulting block using the following RSA private key (padding consists of all zeros, key values listed below with least significant byte first):

Exponent	31 D6 04 EC 80 23 F8 B1 73 8D C6 24 1F 29 35 77 44 E1 12 3F E8 8E 02 06 DD 01 A6 D4 CC 3E 7F 8B 1E 51 F2 09 3B CF DE 6B 69 39 B3 0A CA 93 4B DC B4 0A C9 1C F2 2D 88 D5 1B 72 6A D0 B5 21 95 7B 0B 7A 0B C3 89 07 8D F3 8F 80 BC 5E 90 18 DB 18 53 FD D8 1E CF 3F 57 4E 71 BA F1 E2 C2 D8 7D F2 C9 A2 D5 F6 18 A5 8A 34 7F D3 57 11 85 D3 17 DB 10 34 E4 15 31 C3 83 FA 5B FA 37 41 EC 15 B0 67
Modulus	4F 33 66 9B 34 80 E5 C8 16 4B 49 90 76 9F EB 29 5F DB 22 0D 0E 8B 81 E6 4C E7 34 4D 5D 03 7F 80 28 3F 45 44 97 38 1A C4 38 9C 30 E3 76 59 23 E6 BE 0A 08 34 11 71 13 7C 04 57 84 3C 33 0A 1D 8A E1 E3 09 1D 11 16 0C 71 F3 C5 22 5E 70 62 81 43 10 24 ED 9F C4 CF 70 B1 EC EC 42 50 03 C2 31 3A 5D 81 BA DA 87 66 B3 EC EC A5 51 AC 3E 78 E6 8B 35 26 7F 73 88 8C 21 7D DC 5E 5E 52 C9 75 80 BE

This will produce the following 128-bytes value to be included in the MEPD file:

```
5F 73 1E 6B DA 65 C2 47 47 7E 1D 86 53 D9 2C 65 B9 0E AF BB
53 0F C1 6D CD 80 CE F5 FD 0C FB 47 E0 C6 6C 51 B6 6E 1A 6D
37 36 6E A2 13 AB B1 84 C2 54 BC C2 93 15 00 94 A5 95 31 8E
BE DC 8B 87 93 81 E1 C1 6A 8E B8 BB 6E 8D BF F2 7E 54 D2 30
89 B8 E8 16 24 4D 45 2D 03 99 D2 60 82 0D 1A 8C 3F 24 99 6F
58 2A E9 59 13 CE C5 77 F9 30 F0 75 1C 30 07 76 FD 7A 75 CB
93 DB 8C F7 D9 45 F6 56
```

The example used padding of all zeros, but it is possible to use any values for the padding.

External lock structure:

This is the actual structure defining the individual lock categories. The layout of these data resembles part of the internal structure very closely. Size of each section must be padded to be a multiplum of 8 (size modulus 8 = 0). The entire external lock structure is encrypted using the DES algorithm with the key contained in the RSA encrypted field, before the checksum is calculated.

External lock structure
MEPD configuration
Category 0 (NW)
Category 1 (NS)
Category 2 (SP)
Category 3 (CP)
Category 4 (SIM)
Category 5 (AP)
Category X (n-6 times)

Each category is in turn divided in 2 subsections. The size of each section here must also be padded to be a multiplum of 8 (size modulus 8 = 0).

Category section
Category header
Category data

The layout of each of these data sections is described in chapter 4.1.

Example: The following listing contains a simple external lock structure:

```
07 01 01 00 01 00 01 01 01 01 01 00 00 00 00 00 01 06 03 00
04 00 00 00 A1 B2 C3 D4 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

When encrypted using the DES key from the previous example, the result is the following block to be inserted in the MEPD file:

```
1A 55 BA 23 B6 80 71 BE A6 F0 A6 8A 2A 75 53 97 E0 2E C0 CC
2F 7A C6 56 AE 80 FE 03 D4 7E 26 26 3D 75 95 A9 8B FF 80 9D
```

```
3D 75 95 A9 8B FF 80 9D 3D 75 95 A9 8B FF 80 9D 3D 75 95 A9
8B FF 80 9D 3D 75 95 A9 8B FF 80 9D 3D 75 95 A9 8B FF 80 9D
3D 75 95 A9 8B FF 80 9D 3D 75 95 A9 8B FF 80 9D 3D 75 95 A9
8B FF 80 9D 3D 75 95 A9 8B FF 80 9D 3D 75 95 A9 8B FF 80 9D
3D 75 95 A9 8B FF 80 9D 3D 75 95 A9 8B FF 80 9D 3D 75 95 A9
8B FF 80 9D 3D 75 95 A9 8B FF 80 9D 3D 75 95 A9 8B FF 80 9D
3D 75 95 A9 8B FF 80 9D 3D 75 95 A9 8B FF 80 9D 3D 75 95 A9
8B FF 80 9D 3D 75 95 A9 8B FF 80 9D 3D 75 95 A9 8B FF 80 9D
3D 75 95 A9 8B FF 80 9D 3D 75 95 A9 8B FF 80 9D 3D 75 95 A9
8B FF 80 9D 3D 75 95 A9 8B FF 80 9D 3D 75 95 A9 8B FF 80 9D
3D 75 95 A9 8B FF 80 9D 3D 75 95 A9 8B FF 80 9D 3D 75 95 A9
8B FF 80 9D 3D 75 95 A9 8B FF 80 9D 3D 75 95 A9 8B FF 80 9D
```

When passed to the SHA-1 algorithm together with the size, this data produces the SHA-1 digest listed in the previous example.

Appendices

A. Abbreviation and Acronyms

ACI	Access Control Interface
DSP	Digital Signal Processor
DES	
MMI	Man Machine Interface
FLUID	Flash Loader Utility Independent of Device
ME	Mobile Equipment
SIM	Subscriber Identity Module
GPF	
OTP	One Time Programmable memory
RAM	Random Access Memory
MPK	Manufacturer Private Key

B. Glossary

Certificate	A certificate is a form of digital *passport* or *credential* that simplifies the task of establishing whether a public key truly belongs to the purported owner.
Firmware	Legacy code delivered by the manufacturer and permanently resident on the platform. The firmware is authenticated by the manufacturer's certificate.