Interface Specification - SEC Module

Subject: Software Specification

Revision: 4.3
Author: IFWD
Last Updated: 13/09/2005 10:27
Last Updated By: Bjarlkram Rodaz
File: N:\dwddrv\SEC\doc\SEC Module Specification.doc

1 History

Date	Author	Revision	Comments
2003-05-21	Flanhart	3.4	Document updated with Document History section.
2003-06-15	Flanhart	3.5	Added description of NS qualifier number 5. Added SEC function interface description.
2003-07-28	Gombert	3.6	Function interface updated.
2003-12-11	Gombert	4.0	Document restructured. Document formatting updated. Title renamed from “ME Personalization”. Introduction and Tool Access Restriction added. Verify control key request primitive added. Function SEC_unique_id_read() added. New NS Qualifier 0x06 specified. PSC file version object now default on all PSC requests. Information related to Security Library Concept removed.
2004-08-13	Gombert	4.1	No longer valid document reference removed from figures in Appendix B
2005-02-17	Gombert	4.2	Minor Document Fix
2005-09-13	Gombert	4.3	Update to IFX template.

1.1 Table of Figures

2 Scope

This document contains a description of the Security Module. The document is divided into the following overall parts, each presented in a separate chapter:

- **Overall Description**: This chapter gives an introduction to the security module and discuses the need for the module.
- **IMEI Security**: This chapter presents the security related issues related to IMEI programming and protection against tampering.
- **ME Personalization**: This chapter describes what is understood by the term *Personalization* and how the personalization mechanism can be used/interfaced.
- **Interface Description**: This chapter describes the programming interface against the security module.

2.1 Symbols and Abbreviations

CNL	Corporate Network List
DCK	Depersonalization Control Key
GID1	Group Identification Level 1
GID2	Group Identification Level 2
IMSI	International Mobile Subscriber Identity
ME	Mobile Equipment
MCC	Mobile Country Code
MNC	Mobile Network Code
MS	Mobile Station (ME+SIM)
MSIN	Mobile Subscriber Identification Number
OTA	Over The Air
SIM	Subscriber Identity Module
SAT	SIM Application Toolkit

2.2 Definitions

De-personalization: The process that deactivates the personalization in such that the ME no longer will carry out the personalization checks during startup.

Personalization Code group: The data elements used in the verification of a given personalization mechanism.

Personalization Control key: The key that allows the de-personalization of a personalized ME.

Personalization: The process that enables the mobile to perform a check to see if the used SIM can be operated in the mobile or not.

2.3 References

[1] GSM 02.22: ”Personalization of GSM Mobile Equipment (ME)”

[2] GSM 11.11: “Specification of the Subscriber Identity Module – Mobile Equipment (SIM – ME) interface”
[3] GSM 03.03: “Numbering, addressing and identification”

3 Introduction

Mobile terminals being subsidized by Network Operators or Service Providers are subject to tampering, i.e. attempts to break the personalization SIM locks, change the IMEI and other application oriented security parameters. The hackers typically purchase a large number of subsidized phones at a price significantly below the manufacturing cost, crack the phones and resell them on a complete different market at a substantial higher price. This harms the business of the subsidizing operators/providers because the expected airtime charges does not compensate for their investment.

It is therefore of the highest importance that some highly sensitive information is kept protected against fraudulent tampering. The Security Module implements a number of mechanisms in the effort of preventing unauthorized persons in carrying out undesirable updates or tampering. This is partly done by ensuring that only authorized tools can gain access to the Mobile but also by protecting all sensitive data in the flash.

The security module also implements all requirements related to the ME Personalization (SIM Lock), which includes lock programming and configuration and SIM data comparison on power up.

4 Tool Access Restriction

The MS provides functionality which only authorized persons are allowed to gain access to. This functionality is provided via the PhoneTool, Production SW, or any other specific application. To prevent that not just anyone is allowed to get access to the MS, a secure connection check must be carried out before access is granted. This concept is shown in Figure 1.

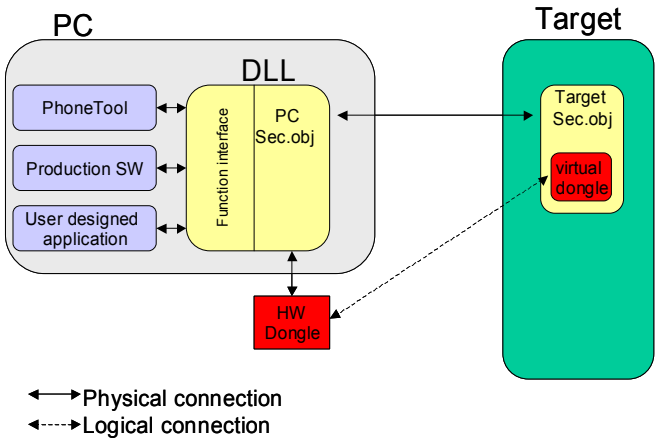


Figure 1: Connection Check.

The connection check is carried out between the “DWDIO.DLL” and the target, and verifies that there is a match between a HW dongle connected to the PC and a virtual dongle data stored in the “Sec.obj” on the target side. The connection check is carried out as shown in the following example below.

- PC Sends: check_connection_1_req : a, f(a+b), b.

If MS receives a valid sequence then:

- MS replies: check_connection_1_con f(b+c), c, f(d).

This is done a few times with different functions and random numbers, thus making it impossible to use a recorded “Check connection sequence” to unlock the mobile.

5 IMEI Security

To prevent fraudulent tampering and potential ME cloning of the IMEI the following measures has been taken to protect the IMEI from being changed:

- The stored IMEI is encrypted using a secured encryption algorithm. Alternatively the customer can add their own encryption algorithm to be used in the encryption and decryption of the IMEI.
- The tool used for programming the IMEI is protected by a HW dongle.

The DWD encryption/ciphering algorithm is making use of:

- A unique hardware specific identification number. The use of the unique hardware specific identification number ties the encoded IMEI to this specific hardware device and makes it impossible to install a raw FLASH copy to a similar hardware device.
- A customer specific component. The customer specific component makes the algorithm unique for each customer.

Combining the hardware specific identification number and the customer specific component ensures that the used algorithm will be unique for the customer but also unique for all the terminals within the range of the customer.

6 ME Personalization

6.1 Concepts of Personalization

The personalization of a mobile phone is a mechanism within the mobile that limits the SIM's with which it will operate. As a starting point all SIM cards can be operated in all brands of mobile equipment, but mostly out of commercial considerations, personalization is a way to limit the range of SIM cards that a particular mobile is allowed to operate.

The key effort in doings so is to store some personalization (comparison) information within the mobile. Whenever a SIM is operated the information stored in the mobile will be matched with specific data or data ranges from the SIM. The usage of a SIM within the mobile is therefore only allowed if a positive match can be found.

6.1.1 Personalization categories

GSM 02.22 specifies 5 personalization mechanisms and each of these mechanisms may contain more than one Personalization code, but there is only one Control Key (unlock key) per personalization mechanism.

The 5 personalization mechanisms specified by GSM 02.22 are shortly described in the following chapters.

6.1.1.1 Network personalization (NO-lock)

A personalization lock, which allows a ME to become personalized to a particular Network Operator - or optionally to a specified range of Network Operators. Personalization data is called Network Code group (or groups) and are represented by MCC [3] and MNC [3] of the IMSI.

6.1.1.2 Network Subset personalization (NS-lock)

A refinement of the Network Operator personalization mechanism, which allows the network operator to limit the usage of the ME to a well defined subset of SIM's. Personalization data is called Network Subset code group (or groups) and are represented by the 6th and 7th digit of the IMSI (first two digits of MSIN [3]) associated with the network code.

6.1.1.3 Service Provider personalization (SP-lock)

A personalization lock, which enables a Service Provider to associate a ME with the Service Provider. Personalization data is called Service Provider code group (or groups) and are represented by the data stored in the GID1 file on the SIM associated with the network code.

6.1.1.4 Corporate personalization (CP-lock)

A refinement of the Service Provider personalization, which gives the Service Provider the ability to prevent the use of other SIM's in ME's they provide for their employees or customers. Personalization data is called Corporate code group (or groups) and are represented by the data stored in the GID2 file on the SIM associated with GID1 and the network code.

6.1.1.5 SIM personalization (SM-lock)

The SIM Personalization lock is an anti theft mechanism. When a ME is SIM Personalized to a particular SIM, it will refuse to operate with another SIM. Personalization data is called SIM code group (or groups) and are represented by the entire IMSI from the SIM.

6.1.2 Proprietary Personalization Requirements

A lot of the Network Operators and Service Providers have their own additional requirements (on top of GSM 02.22) to the data or data ranges to be used to identify the SIM's to be accepted by their lock mechanism(s). Also the requirements to how the ME should behave in the different lock scenarios may vary from operator to operator, and for these reasons the Personalization concept has been designed in a way where configurability and flexibility are the key words.

When defining the data comparison requirements attached to a given lock mechanism, the programming should be accompanied with a 'qualifier'. This 'qualifier' specifies the comparison method to be used for this specific lock, and it is thereby easy to add additional 'qualifiers' should the existing comparison method prove not to be sufficient. The qualifiers valid for each of the lock mechanism are described in the subchapters of 6.3.3.12.

The behavior of the ME in different scenarios can be controlled when the personalization mechanism are programmed during the production phase.

- E.g. should it be possible to perform unlock attempts when the 'wrong' SIM is inserted or is this action only allowed when the 'right' SIM is inserted.
- E.g. should it be possible to unlock a lock mechanism and should there be a maximum limit for unlock attempts.

The configurations of the different lock mechanism are described in detail in the chapter 6.3.3.3 and 6.3.3.4.

6.1.3 De-Personalization Control Key

The Personalization feature is controlled by a Control Key, which basically means that it is possible to de-personalize a personalized ME by applying the appropriate Control Key. The Control key will, in most cases, only be known by authorized personal (such as the network operator, service provider etc.). De-personalization can also be achieved using the OTA De-personalization mechanisms (see ref. [1]) also controlled by authorized personal.

6.1.4 Personalization

Personalization can occur in the following scenarios:

- At the production site the mobile can become personalized to one or more of the Personalization locks under the control of the production test equipment. The respective control code must be chosen and kept secured by the manufacturer.
- If a lock has been armed from the production, it will lock to the first used SIM-card using data from this very SIM to obtain the Personalization code(s). CNL can be used as an optional feature

to extract log data from. The respective control code must however still be chosen and kept secured by the manufacturer.

- Whenever the state of a given lock is in its disable state meaning that the lock is inactive, it is possible to activate the lock and thereby personalize the mobile.

6.1.5 De-personalisation

De-personalization can occur in the following scenarios:

- Whenever the correct control key is applied.
- If an OTA specific SMS has been received by the ME.
- Optionally, the manufacturer could have a Master Control Key allowing the re-programming of the lock mechanisms (see chapter 6.1.6).

6.1.6 Master Control Key

Once a Personalization mechanism has been programmed, it can – as default – never be re-programmed again. The manufacturer has however the optional possibility to allow the Personalization mechanism to be re-programmed under the control of a Master Control Key. There is one Master Control Key per mobile and it must be selected and kept properly secured by the manufacturer.

The purpose of this feature is to allow the manufacturer – at a later stage – to re-personalize an already personalized mobile e.g. a number of mobiles has been personalized with the requirements of one customer (Network Operator/Service Provider). If it turns out that these mobiles needs to be shipped to another customer (Network Operator/Service Provider), it will be possible to re-personalize an already personalized mobile (against the wishes of the new customer) by applying the correct Master Control Key (via the production test equipment).

The verification of the appropriate Master Control Key does not itself de-personalize the mobile, but it will afterwards – as long to mobile is powered on – be possible to re-program the mobile in the same way as it was done the very first time (in the production). Note that the first re-programming attempt will reset the settings of all previously programmed locks.

As default this feature is not enabled, but if the re-programming feature is desired, the manufacturer must enable the feature in the initial production phase using the appropriate programming OBJECT (see chapter 6.3.1).

6.2 Personalization Description

Since personalization is a mechanism to ensure that only SIM's of a given Network Operator or Service Provider can be operated by the mobile, it is very crucial that the data related to the personalization mechanisms are securely stored within the mobile. Also the procedures handling the necessary comparison between the required data from the SIM and the data stored within the mobile must be handled with special care.

To prevent fraudulent tampering with the personalization mechanisms, it has proven necessary to encapsulate the core personalization functionality in a special security (SEC) module. The idea of having

the SEC module is basically to have one and only one module where the algorithms can be kept secrete. This means that only the SEC object will be available in a SW delivery. The access to the module is restricted to a very limited number of DWD engineers and the actual source code is never surrendered to customers.

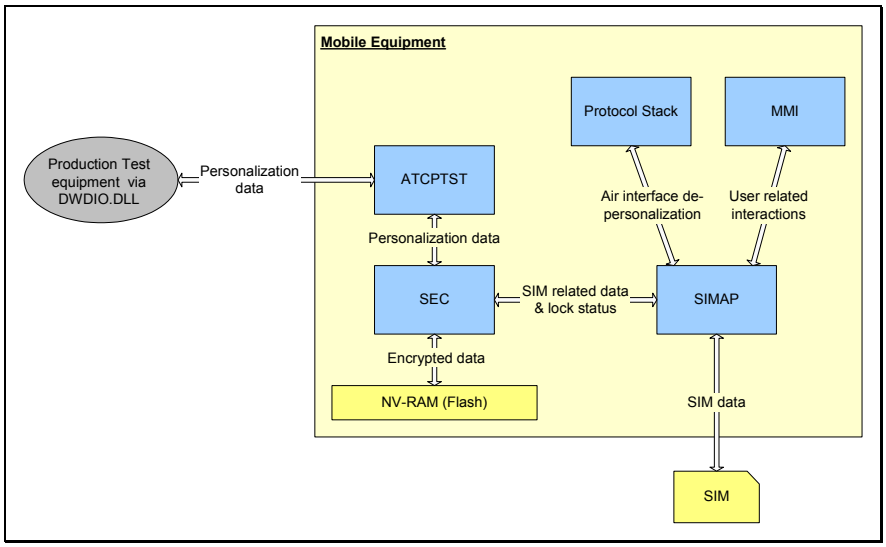


Figure 2: Personalization modularization

Figure 2 illustrates the logical modularization of the Mobile Equipment with respect to personalization.

The idea is that the mobile manufacturer initializes the mobile personalization during the production of the mobile in accordance to the wishes from their customer (Network Operator and/or the Service Provider). The locks that need to be utilized when leaving the production and the exact behavior of each of the locks are settled at this stage. The SEC module is in other words “programmed” from the Production Test equipment via the DWDIO.DLL, which means that the mode of operation of each lock mechanism and the related lock data are store in the non-volatile memory (Flash) in an encrypted format.

In normal operating mode (after leaving the production area), it is of the responsibility of the SIMAP module to feed the SEC module with the appropriate SIM data. The SIM data needed is indicated during power-on by the SEC module. The SEC module will then – based on the scrambled data stored in the Flash and the data from the operated SIM – determine if operation with the inserted SIM is allowed or if the operation should be rejected. In case the SIM is rejected, it will only be possible to operate that SIM if the correct unlock key can be applied.

The user interaction between the SEC module and the MMI is handled via the SIMAP module, but it is important to bear in mind that it is always the SEC module that makes the final decision whether operation of the SIM is allowed or not. SIMAP also serves as an interface module to the protocol stack, both with respect to OTA de-personalization but also to prevent actual registration in cases where the SIM is rejected.

The MMI is with respect to the user interaction of the Personalization mechanism serving SIMAP. This means that the MMI will be informed if normal operation is allowed or if the user needs to verify one or more Control Keys. Should Control Key verification be needed, the MMI must feed the entered Control Key to SIMAP, who then again will pass it on to SEC where the actual verification is carried out. The MMI will receive the verification result from SEC via SIMAP. If more locks needs to be verified (as a

result of the activeness of more locks while using the "wrong" SIM card) this will happen in a sequence (one by one) i.e. the user will have to verify the Control Keys for the different lock one by one.

The roles of the modules:

(Chapter 9 holds examples describing a number of different interworking scenarios).

SEC:

- Handles the formatting and storing of the personalization related data programmed from the production equipment (DWDIO.DLL).
- Handles the encryption and decryption of all personalization related data.
- Controls the changes of state for each of the involved lock mechanisms.
- Indicates to SIMAP which files from the SIM are needed for the personalization startup check.
- Performs the personalization startup check based on the data from the SIM and the encrypted data from the Flash.
- Carries out Control Key verifications entered by the user (which means that the Control Key stored in Flash will never be visible outside SEC).

SIMAP:

- Reads the required personalization data from the SIM and send it to SEC.
- Feeds the states of all the locks and the general mode of operation to the MMI.
- Passes user entered Control Keys from the MMI to SEC (and transfers the verification result back to the MMI).
- Makes sure that the ME is not entering service when the SIM is not allowed to be operated.
- Handles the Control Key verification after a SMS-PP Data Download OTA De-personalization (STK – DCK Refresh).
- Handles the Control Key verification after a SMS-PP ME Specific OTA De-personalization.

MMI:

- Determine – based on the information received from SIMAP – if operation is allowed or if the user needs to verify control key(s).
- Prompts the user for lock Control Key(s) if the "wrong" SIM is inserted.
- Sends the entered Control Key to SIMAP for verification.
- Informs the user of the verification result.

6.2.1 Modes of operation

Each of the personalization mechanisms can be operated in different modes and these modes are tightly related to the actual personalization behavior needed. The different modes are individually described in the following chapters, but keep in mind that there is one mode for each of the personalization mechanisms.

The mode initialization is an irreversible process, which means that the initialization of each of the lock mechanisms can never happen again. Once initialized, the personalization functionality will follow the state path as illustrated in Figure 3. Optionally, a master control key can allow that the personalization mechanisms can be re-programmed in the production, but this must be programmed at the very initial initialization.

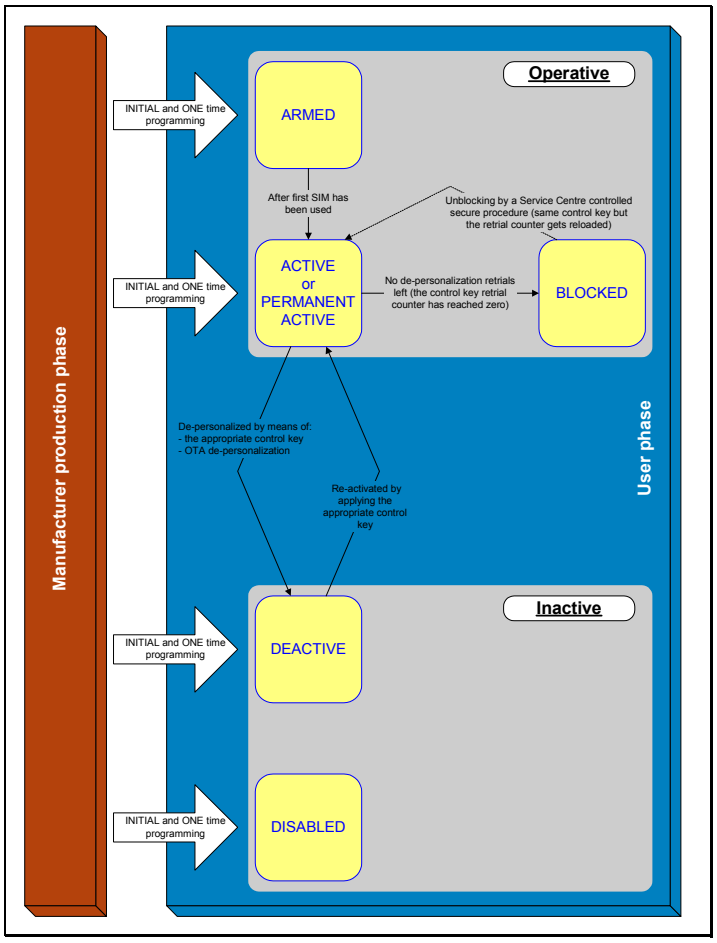


Figure 3: Modes of operation

6.2.1.1 Disabled mode

If the personalization lock never is to be used (no restrictions are ever needed for that lock mechanism), the personalization mechanism can be put into the disable mode. Once this mode has been programmed for a given lock, there is not re-activation of that lock again.

6.2.1.2 De-active mode

The deactivation mode means that the personalization mechanism currently is not in use (no restrictions are at current checked for that lock mechanism). From this mode it is however possible at a later stage to utilize the mechanism – to activate the mechanism.

If the *de-active* mode has been reached as a result of a successful unlock verification, it might optionally be possible to set the personalization mechanism back into the mode that was originally programmed in the production. This optional feature must be programmed in the production phase using the object described in chapter 6.3.3.3.

6.2.1.3 Active mode

When a personalization mechanism is in the active mode it means that only SIM cards fulfilling the personalization requirements (matching data content) are accepted and are allowed to be operated. Only the appropriate control key (unlock code) can change the mode (to de-active).

In case the user applies an incorrect control key, there are two possible reactions (reactions that must be determined in the production using the object described in chapter 6.3.3.3:

- A retrial counter is incremented for each unsuccessful attempt. Once a maximum limit is reached the mode is changed to *blocked* and no further unlock attempts are allowed. The maximum limit can be settled in the production using the object described in chapter 6.3.1. (default maximum retrial value is 10).

- Unlimited attempts are allowed but the acceptable time between new unlock attempts will continue to increase with the number of failing unlock attempts.

It might be a requirement that the user must never be able to deactivate the personalization mechanism (no control key will be accepted nor initialized), and the mode will in that case be *permanent active*.

6.2.1.4 Armed mode

It might be desired that a personalization mechanism lock against the very first SIM card operated within the mobile - provided that other active lock checks has passed. The personalization feature is in other words armed – ready to string to the first inserted SIM.

In case other locks are *active*, they will be processed before the *armed* locks are taken care of. This means that the code groups will be examined for the *active* locks and only if the code group(s) of all active locks has a positive match, the *armed* locks will be handled (which means that appropriate data from the SIM is store as code group(s) in the FLASH).

6.2.1.5 Blocked mode

If the De-personalization control key has been entered wrongly several times in such that the maximum number of failed attempts has been reached, the mode of operation becomes *blocked*. This means that it is no longer possible to de-personalize the mobile – but the personalization mechanism is still active (the personalization startup check is still carried out). Alternatively, it is possible to prevent further ME operation (completely block further ME operation) in case the blocked state is reached, and that is configured in the lock characteristics object described in 6.3.3.4.

6.2.2 Personalization data

Each of the different types of personalization locks can contain a number of code groups e.g. the Network personalization mechanism may not only hold one MCC and MNC but a list of MCC's and MNC's. The length of the code group and the number of code groups may vary from lock to lock.

Figure 4 illustrates for the different personalization mechanisms which data components each of the personalization code groups are made up of. Although some of the personalization mechanism might lock against the same personalization data (e.g. MCC and MNC), it is necessary for each lock to have the complete personalization code group attached, because the personalization mechanisms can be operated

independently. Figure 4 also illustrates the additional data structures attached to each single personalization mechanism (state, control key and retrial counter).

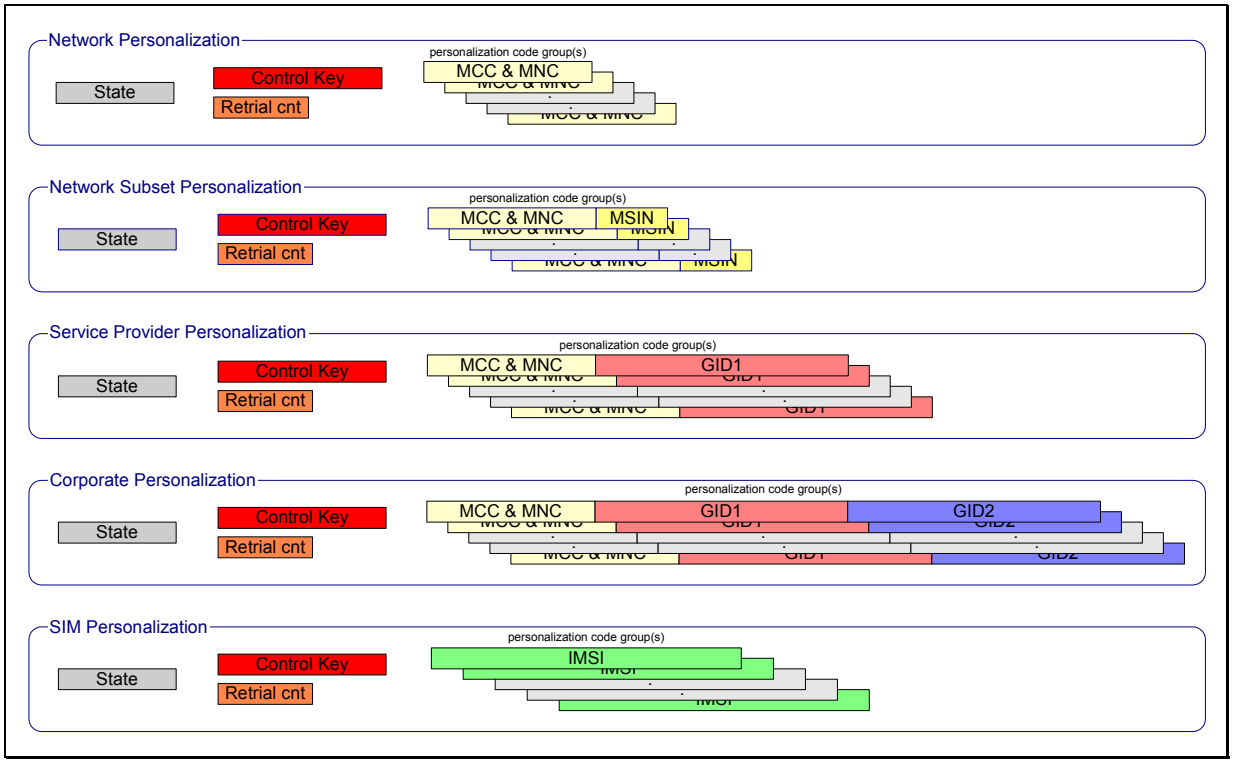


Figure 4: Personalization data

6.3 Production Programming

When performing the Personalization programming in the production, it is important first to consider which locks needs to be utilized and secondly how the different locks should be configured. Figure 5 illustrates the possible steps for Personalization of the locks and 'burning the programming fuse' ends the programming sequence. The latter action must be performed to ensure that no unauthorized personal tries to tamper with unused locks once the mobile has left the production site.

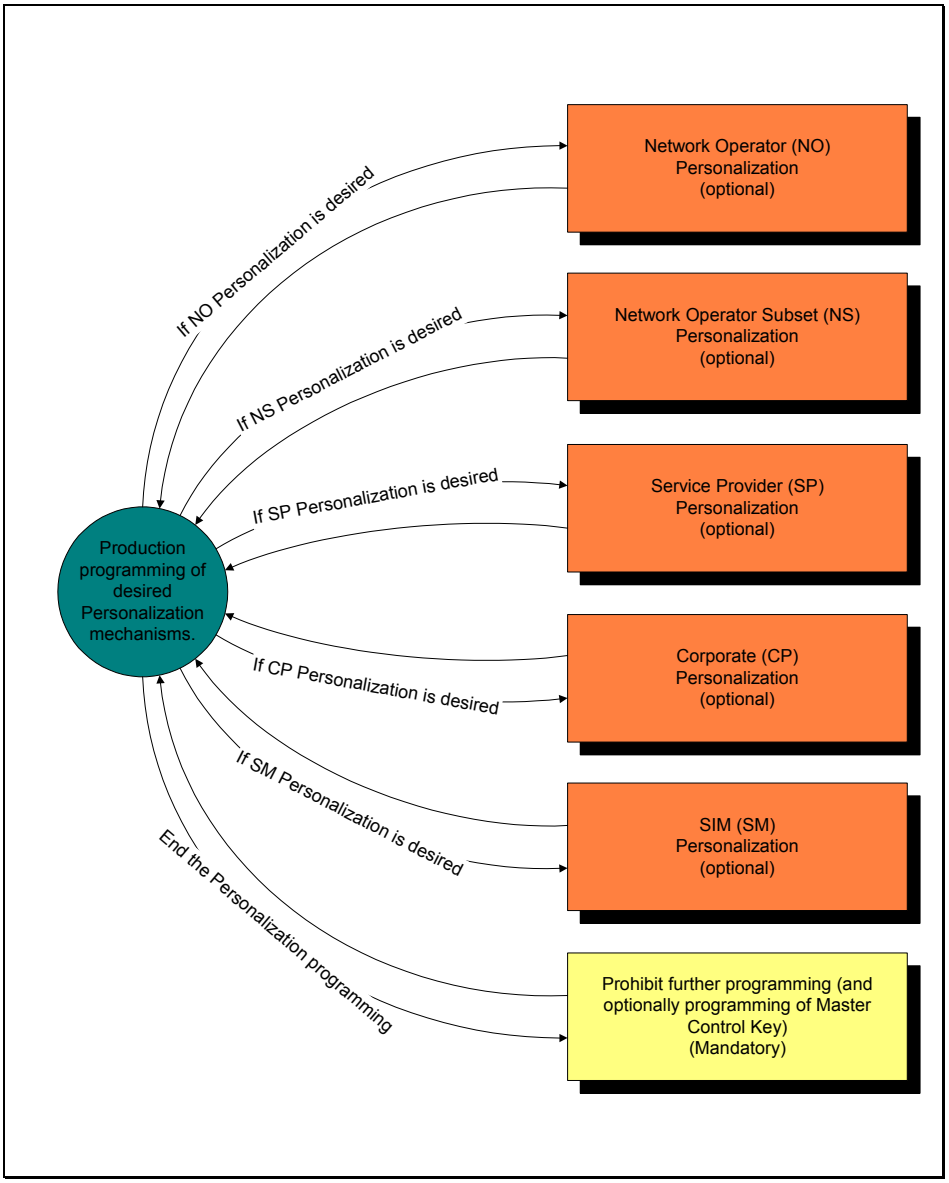


Figure 5: Production Programming

Programming of the individual locks basically means selecting the state/mode in which the lock should operate from i.e. should the lock be *armed*, should it be *active* or should it simply be *deactive/disabled* (unused). The lock characteristics must also be selected i.e. unlock retrial mode, unlock possibilities etc. The Control Key must be selected (and stored securely in the production database) and the code groups

chosen. Note that code groups are not to be programmed for *armed* locks; only the armed characteristics should be given.

Once all the desired locks have been programmed, it is possible to prevent further programming once the mobile has left the production. It is however possible to allow a re-programming under the control of a Master Control Key (see chapter 6.1.6). The programming prevention is implicit obtained using the master control key object but should the Master Control Key be undesired, the key length must simply be set to zero to prevent further programming.

The Master Control Key can only be verified via the DLL (not via the MMI). If the Master Control Key is verified, it will afterwards be possible to re-personalize the locks but only as long as the ME is powered (the access conditions obtained from verifying the Master Control Key is lost when powering off the ME). The first re-personalization attempt after successful Master Control Key verification will reset all prior programmed locks and the flow illustrated in Figure 5 will then again apply to the initialization process.

6.3.1 Programming Interface

Programming in the production is carried out be feeding the DWDIO.DLL (using the DDL function DWD_program_pers_code – either directly or through the Security submenu of the PhoneTool) with a number of files (scripts) containing the desired application command requirements.

The DLL function DWD_program_pers_code is used for the transfer of all personalization related commands and is as such completely transparent to application data. The DLL function simply transfers a string of request data to the ME and returns a response data string when the ME has carried out the requested task. The only error check mechanism on this transmission level is merely an indication to whether or not the transferred data was successfully received by the ME.

The content of the script(s) feed into the DLL function forms the actual application command and each programming command script can be generated from the Security submenu of the PhoneTool or can simply be made by hand (chapter 10 lists some application programming scripts). Either way, the command syntax described in this paragraph must apply all scripts.

The production programming scripts are compounded by objects to make the initialization of the personalization mechanisms as flexible as possible. Each object will – as described in the following – form the ‘building stones’ for the commands to be sent to the mobile (to the SEC module) hence making the initialization very flexible should there be a need for additional features or simply an extension of existing features.

A command for personalizing one of the lock mechanisms would as a minimum contain the data objects listed below. What goes in the different objects are described in detail in the following chapters.

- Function code (Primitive ID)
- File Version Object
- Lock ID Object
- Lock Characteristics Object
- Unlock Code (Control Key) Object
- Code Group Object

6.3.2 Function codes (primitives)

The following table holds a list of primitives (function codes) that are defined when programming the personalization mechanisms from the DWDIO.DLL.

Function Code (Primitive)	ID	Description
Personalize Request (6.3.2.1), (6.3.2.2), (6.3.2.3)	0x01	Used when personalization of a given lock is required.
Get Lock Profiles (6.3.2.7)	0x02	Used to readout the lock profiles.
Get Code Groups (6.3.2.8)	0x05	Used to retrieve code groups for a give lock.
Verify Control Key (6.3.2.4)	0x06	Used to verify the Control Key of a given lock.
General Settings (6.3.2.5)	0x07	Used to set the overall settings for the Personalization mechanism and to ensure that no further programming can be done..
Verify Master Control Key (6.3.2.6)	0x08	Used to verify the Master Control Key.
Response	0xF0	Used as the identifier for the response data returned from the ME to the DLL (the result of one of the above requested actions).

6.3.2.1 ACTIVATE Programming

When a lock mechanism needs to be activated there are a number of objects required in the application command programming string. Some of the objects are mandatory and some are optional but the order in which the objects are used must apply the order listed in the following table.

Primitive: ***Personalize request***		
Object name	M/O	Comments
PSC file version (6.3.3.1)	**Mandatory**	Object for identifying the version of the used PSC file (programming file).
Lock ID (6.3.3.2)	**Mandatory**	Object carrying the involved lock identification.
Lock Characteristics (6.3.3.4)	Optional	Lock Characteristics settings. When absent, all bits are interpreted as if they where set to 0.
Retrial Counter (6.3.3.5)	Optional	Object used for setting the maximum retrial counter for the lock. When absent, the default retrial counter is 10.
Time Verification Method (6.3.3.6)	Optional	Object used for selecting the time verification method. When absent, the method as described in descriptor 1 for that object will be used as the default method used
Unlock Code (6.3.3.7)	**Mandatory**	Object holding the unlock code. Must be within the range of 8-16 digits. If no Unlock Code is required (permanent_active meaning that it will not be possible for the user to deactivate the lock), the unlock code length should be set to 0.
SIM File ID (6.3.3.11) (SP lock only)	Optional	Object identifying the path to a SIM file in case the SP lock is used to lock against data from another file than GID1.
Code Group(s) (6.3.3.12)	**Mandatory**	Object(s) holding the comparison data needed for the lock mechanism.

Note: the Retrial Counter object is mutually exclusive to the Time Verification Method object.

The application command response will hold the following information.

Primitive: ***Response cause***		
Object name	M/O	Comments

Response cause (6.3.3.15)	**Mandatory**	Object carrying the response of the requested action.

6.3.2.2 ARMED Programming

When a lock mechanism needs to be armed (stringing to the first inserted SIM) there are a number of objects required in the application command programming string. Some of the objects are mandatory and some are optional but the order in which the objects are used must apply the order listed in the following table.

Primitive: ***Personalize request***		
Object name	M/O	Comments
PSC file version (6.3.3.1)	**Mandatory**	Object for identifying the version of the used PSC file (programming file).
Lock ID (6.3.3.2)	**Mandatory**	Object carrying the involved lock identification.
Lock Characteristics (6.3.3.4)	Optional	Lock Characteristics settings. When absent, all bits are interpreted as if they where set to 0.
Retrial Counter (6.3.3.5)	Optional	Object used for setting the maximum retrial counter for the lock. When absent, the default retrial counter is 10.
Time Verification Method (6.3.3.6)	Optional	Object used for selecting the time verification method. When absent, the method as described in descriptor 1 for that object will be used as the default method used
Unlock Code (6.3.3.7)	**Mandatory**	Object holding the unlock code. Must be within the range of 8-16 digits. If no Unlock Code is required (permanent_active meaning that it will not be possible for the user to deactivate the lock), the unlock code length should be set to 0.
SIM File ID (6.3.3.11) (SP lock only)	Optional	Object identifying the path to a SIM file in case the SP lock is used to lock to data from another file than GID1.
Armed characteristics (6.3.3.8)	**Mandatory**	Object holding the characteristics involved in the arming process.

Note: the Retrial Counter object is mutually exclusive to the Time Verification Method object.

The application command response will hold the following information.

Primitive: ***Response cause***		
Object name	M/O	Comments
Response cause (6.3.3.15)	**Mandatory**	Object carrying the response of the requested action.

6.3.2.3 DISABLED Programming

When a lock mechanism is never going to be needed the state must be changed to armed using the following application command structure.

Primitive: ***Personalize request***		
Object name	M/O	Comments
PSC file version (6.3.3.1)	**Mandatory**	Object for identifying the version of the used PSC file (programming file).
Lock ID (6.3.3.2)	**Mandatory**	Object carrying the involved lock identification.
Disable characteristics (6.3.3.9)	**Mandatory**	Disable object.

The application command response will hold the following information:

Primitive: ***Response cause***		
Object name	M/O	Comments
Response cause (6.3.3.15)	**Mandatory**	Object carrying the response of the requested action.

6.3.2.4 Control Key Verification

Verification of the Control Key of a given lock requires the following application command structure.

Primitive: ***Verify Control Key request***		
Object name	M/O	Comments
PSC file version (6.3.3.1)	**Mandatory**	Object for identifying the version of the used PSC file (programming file).
Lock ID (6.3.3.2)	**Mandatory**	Object carrying the involved lock identification.
Unlock Code (6.3.3.7)	**Mandatory**	The Control Key to be verified. Must be within 6-16 digits.

The application command response will hold the following information:

Primitive: ***Response cause***		
Object name	M/O	Comments
Response cause (6.3.3.15)	**Mandatory**	Object carrying the response of the requested action.
Verify Result (6.3.3.14)	**Mandatory**	Object carrying the verify control key result.

6.3.2.5 General Setting Programming

When all the personalization mechanisms have been programmed in accordance to the desired requirements, the following application command structure offers the possibility to setup some general/overall lock characteristics for the ME. If required, it is also possible to apply a Master Control Key for authorized re-initialization. It is also important to be aware that issuing the General Settings programming command will prevent any further personalization programming (unless the correct Master Control key is applied). Some of the objects are mandatory and some are optional but the order in which the objects are used must apply the order listed in the following table.

Primitive: ***General Settings request***		
Object name	M/O	Comments
PSC file version (6.3.3.1)	**Mandatory**	Object for identifying the version of the used PSC file (programming file).
Overall Characteristics (6.3.3.3)	Optional	Object used for setting the overall characteristics for the overall personalization behavior. When absent, all bits are interpreted as if they where set to 0.
Unlock Order (6.3.3.10)	Optional	Object to define the required unlock order for the different lock mechanisms. Should only be set if the unlock order deviates from the order NO, NS, SP, CP and last SIM.
Retrial Counter (6.3.3.5)	Optional	Object used for setting the Master Control Key retrial counter. When absent, the default retrial counter is 5. When this limit has been reached (as a result of successive failing attempts) there is NO way out.
Unlock Code (6.3.3.7)	**Mandatory**	Object for setting the Master Control Key. Must be within the range of 8-16 digits. If no Master Control Key is desired

		(meaning re-programming will never be possible), the Master Control Key length should be set to 0.

The application command response will hold the following information:

Primitive: ***Response cause***		
Object name	M/O	Comments
Response cause (6.3.3.15)	**Mandatory**	Object carrying the response of the requested action.

6.3.2.6 Master Key Verification

Verification of the Master Control Key requires the following application command structure.

Primitive: ***Verify Master Key request***		
Object name	M/O	Comments
PSC file version (6.3.3.1)	**Mandatory**	Object for identifying the version of the used PSC file (programming file).
Unlock Code (6.3.3.7)	**Mandatory**	The Master Control Key to be verified. Must be within 8-16 digits.

The application command response will hold the following information:

Primitive: ***Response cause***		
Object name	M/O	Comments
Response cause (6.3.3.15)	**Mandatory**	Object carrying the response of the requested action.

6.3.2.7 Lock Status Interrogation

The current status of all the locks can be fetched using the following application command structure.

Primitive: ***Get Lock profile***		
Object name	M/O	Comments
PSC file version (6.3.3.1)	**Mandatory**	Object for identifying the version of the used PSC file (programming file).

The application command response will hold the following information:

Primitive: ***Response cause***		
Object name	M/O	Comments
Response cause (6.3.3.15)	**Mandatory**	Object carrying the response of the requested action.
Lock Profiles (6.3.3.13)	**Mandatory**	Object carrying the profile of the different locks.

6.3.2.8 Code Groups Interrogation

The code groups assigned to a given lock mechanism can be fetched using the following application command structure.

Primitive: ***Get Code Groups request***		
Object name	M/O	Comments
PSC file version (6.3.3.1)	**Mandatory**	Object for identifying the version of the used PSC file (programming file).

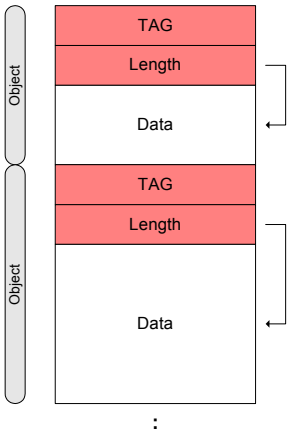
Lock ID (6.3.3.2)	**Mandatory**	Object carrying the involved lock identification.

The application command response will hold the following information:

Primitive: ***Response cause***		
Object name	M/O	Comments
Response cause (6.3.3.15)	**Mandatory**	Object carrying the response of the requested action.
Code Group(s) (6.3.3.12)	**Mandatory**	Object(s) holding the comparison data (Code Groups) programmed for the lock mechanism.

6.3.3 Object Structure Description

An object is always initiated by a TAG (a unique identification) identifying the whereabouts of a given object. The second part of the object is a length indicator holding the length of the object data body (excluding the TAG and the length field itself). The actual object body holds the data specific to that object.



It is allowed to have a number of objects present in order to carry the required information for a request to the ME. All the different objects and their coding will be described in the following sub chapters.

6.3.3.1 PSC File Version Object

Object for defining the version of the current PSC-file format.

File Version Object:		
Byte(s)	**Description**	**Length**
1	PSC File Version TAG (0x0F)	1
2	Length (=1)	1
3	File version number	1

6.3.3.2 Lock ID Object

Object for indicating the lock ID for a given programming request.

Lock ID Object:		
Byte(s)	**Description**	**Length**
1	Lock ID TAG (0x10)	1
2	Length (=1)	1
3	Identification	1

Identification coding	ID
NO Lock	0x00
NS Lock	0x01
SP Lock	0x02
CP Lock	0x03
SM Lock	0x04

6.3.3.3 Overall Characteristics Object

Object for setting the overall personalization characteristics.

Lock Characteristics Object:		
Byte(s)	**Description**	**Length**
1	Overall Characteristics TAG (0x15)	1
2	Length	1
3	Characteristics	1-2

Overall characteristics coding:								
Characteristics byte 1								
B7	B6	B5	B4	B3	B2	B1	B0	Usage
-	-	-	-	-	-	-	√	**Test SIM (001-01) acceptance.** B0=0: The use of test SIM's will bypass the Personalization checks. This means that test SIM's can be operated by the EM even though one or more of the personalization mechanisms are in use. This is also the default value if this object is absent. B0=1: Test SIM's are NOT allowed to bypass the Personalization checks. Test SIM's are – when this not is set – not allowed to be operated by a personalized ME.
-	-	-	-	-	-	√	-	**One Common Control Key (Unlock key):** B7=0: Each individual lock mechanism has it's own control key (unlock key) attached. B7=1: One control key (unlock key) is valid for all the lock mechanisms; which basically means that one correct control key verification unlocks all lock mechanisms. Note that the identical unlock key must be programmed for all the operative locks.
-	-	-	-	√	√	-	-	**Unlocking configuration using right SIM:** B3B2=00: Unlocking allowed, both via MENU and keyboard string. B3B2=x1: Unlocking not allowed, via MENU. B3B2=1x: Unlocking not allowed, via keyboard string.
-	-	√	√	-	-	-	-	**Unlocking configuration using incorrect SIM:** B5B4=00: Unlocking allowed, both via power on prompt and keyboard string. B5B4=x1: Unlocking not allowed via prompting user at power on. B5B4=1x: Unlocking not allowed via keyboard string.
√	√	-	-	-	-	-	-	**Unlocking configuration without SIM:** B7B6=00: Unlocking allowed, both via MENU and keyboard string. B7B6=x1: Unlocking not allowed via MENU. B7B6=1x: Unlocking not allowed via keyboard string.
Characteristics byte 2								
B7	B6	B5	B4	B3	B2	B1	B0	
-	-	-	-	-	-	-	-	**RFU**

6.3.3.4 Lock Characteristics Object

Object for setting the characteristics for a given lock.

Lock Characteristics Object:		
Byte(s)	**Description**	**Length**
1	Lock Characteristics TAG (0x20)	1
2	Length	1
3	Characteristics	1-2

Lock characteristics coding:

Characteristics byte 1:

B7	B6	B5	B4	B3	B2	B1	B0	Usage
-	-	-	-	-	-	-	√	**Unlocking Allowed.** Unlock using the appropriate Control Key is allowed. If this bit is not set Unlocking of the lock will not be possible (and no Control Key object is needed).
-	-	-	-	-	-	√	-	**CNL Arming Supported.** Enable arming against elements from the CNL list (in case the mode of operation is A*rmed*).
-	-	-	-	-	√	-	-	**DCK Supported.** De-personalization via DCK supported. If this bit is not set this means that it will not be possible to OTA depersonalize the lock.
-	-	-	-	√	-	-	-	**Retrial Maximum.** Maximum number of retrial supported if the bit is set. Otherwise will there be an unlimited number of retrial attempts but instead there will be an increasingly time period between attempts.
-	-	-	√	-	-	-	-	**User Re-activation Allowed.** If the bit is set the user is allowed re-enable (using the appropriate control key) the same mode of operation as the lock mechanism originally had when leaving the production – that is if the lock has already been disabled by the user using the appropriate control key.
-	-	√	-	-	-	-	-	**Maximum retrial reached means blocked ME:** B5=0: When maximum retrial counter is reached, only the possibilities to perform further unlock operations are blocked. The ME can still be used with the right SIM. B5=1: When maximum retrial counter is reached, the ME be completely blocked (can not be operated no matter if the right or the incorrect SIM is inserted).
-	-	-	-	-	-	-	-	**RFU**

Characteristics byte 2:

B7	B6	B5	B4	B3	B2	B1	B0	Usage
-	-	-	-	-	-	-	-	**RFU**

6.3.3.5 Retrial Counter Object

This object is used to set the retrial counter of the Control Key or of the Master Control Key.

Retrial Counter Object:		
Byte(s)	**Description**	**Length**

1	Retrial Counter TAG (0x21)	1
2	Length (=1)	1
3	Retrial counter	1

6.3.3.6 Verification Time Object

This object is used for setting the time separation method when running in unlimited retrial mode.

Verification Time Object:		
Byte(s)	**Description**	**Length**
1	Verification Time TAG (0x22)	1
2	Length (=1)	1
3	Verification time descriptor	1

Descriptor	Time formula
1	Delta time = 1<<retrial [minutes] ~ Exponential graduation
2	Delta time = retrial [minutes] ~ Linear graduation
TPD	

Note: If the Verification Time Object is absent, descriptor 1 will be the default method used.

6.3.3.7 Unlock Code Object

Object for programming the Unlock Code (Control Key).

Unlock Code Object:		
Byte(s)	**Description**	**Length**
1	Unlock Code TAG (0x25)	1
2	Length (X)	1
3 to X+2	Unlock Code (Control Key)	X-1

6.3.3.8 Armed Characteristics Object

Object for setting the characteristics for the Armed mode.

Armed Characteristics Object:		
Byte(s)	**Description**	**Length**
1	Armed Characteristics TAG (0x26)	1
2	Length (X)	1
3	Qualifier	1
4 to X+3	Characteristics descriptor	X-1

For the SM-lock there will be not data to be carried in the Armed Characteristics Object (length = 0) and there will as a result be no associated object data. The presence of the object itself will order the armed mode.

The remaining locks however have a need for indicating different armed data characteristics which is indicated in the following table.

Lock	Qualifier	Byte	Characteristics descriptor

NO	0x01	1	Indication if 2 digit or 3 digit MNC is expected when stringing to the SIM.
NS	0x01	1	Indication if 2 digit or 3 digit MNC is expected when stringing to the SIM. Using this qualifier implicit means that digit 1 and 2 of MSIN will be used when stringing to the SIM (as specified by GSM 02.22).
	0x02	1	Indication if 2 digit or 3 digit MNC is expected when stringing to the SIM.
		2	Number (N) of digits used when stringing to the SIM.
		3..3+(N-1)/2	The position of the required MSIN digits. Each byte holds two digit positions – one per nipple in the following order (unused nipples must be set to F): MSIN#(x-1) and MSIN#(x). If for instance the MSIN digits 2, 4 and 7 is needed in the stringing process, byte 3 and 4 of this characteristics must contain the following information: [3]: 0x24 [4]: 0x7F
SP	0x01	1	Indication if 2 digit or 3 digit MNC is expected when stringing to the SIM.
		2	Required GID1 data length needed when stringing to the SIM.
	0x02	1	Indication if 2 digit or 3 digit MNC is expected when stringing to the SIM.
		2	Required data length needed when stringing to a file specified in the SIM file ID object.
CP	0x01	1	Indication if 2 digit or 3 digit MNC is expected when stringing to the SIM.
		2	Required GID1 data length needed when stringing to the SIM.
		3	Required GID2 data length needed when stringing to the SIM.

Note: characteristics set to 0 means 2 digit MNC whereas 1 means 3 digit MNC.

6.3.3.9 Disable Lock Object

Object to be used when a lock needs to be disabled (never used).

Disable Lock Object:

Byte(s)	Description	Length
1	Disable Lock TAG (0x27)	1
2	Length (=0)	1

Note: There is no need for data in this object – the presence of the object indicates the lock disable requirements.

6.3.3.10 Unlock Order Object

Object to be used when defining the order in witch the operative locks should be unlocked.

Unlock Order Object:		
Byte(s)	**Description**	**Length**
1	Disable Lock TAG (0x28)	1
2	Length (=number of locks)	1
3	NO unlock order	1
4	NS unlock order	1
5	SP unlock order	1
6	CP unlock order	1
7	SIM unlock order	1

Note: Order number 1 means unlock is required first whereas 5 means unlock is required last.

Object examples:
0x28 0x05 0x03 0x01 0x02 0x05 0x04 gives the following unlock order: NS, SP, NO, SIM and last CP.

6.3.3.11 SIM File ID Object

Object defining the SIM file id, in case SP lock needs lock against another SIM file than GID1.

SIM File ID Object:		
Byte(s)	**Description**	**Length**
1	SIM File ID TAG (0x29)	1
2	Length (X ≤ 9)	1
3	Elementary file type 0: Transparent elementary file 1: Record based (linear fixed or cyclic) elementary file.	1
4 to X+3	File path (excluding MF)	X-1

Object examples:
0x29 0x05 0x00 0x7F 0x20 0x6F 0x05 describes: Language Preference list (transparent file: '6F05') in the GSM directory ('7F20').

6.3.3.12 Code Group Objects

The Code Group Object is used to indicate that the following data area is used for Personalization Code Groups. In the Code Group object it is possible to group the desired elements in such a way that a minimum of entries needs to be used in case of overlapping information.

The following sub-chapters contain a detailed description of the basic code group element objects. But before looking at the objects descriptions, a SP Lock example will illustrate the grouping principles. The following personalization restrictions will apply for the example (meaning that only SIM's that can meet the following requirements are accepted and will be operated):

MCC MNC & GID1:
- 238 01 & “Hello”
- 238 01 & “Tie”
- 238 02 & “Hello”
- 238 02 & “Key”
- 238 02 & “Bye”
- 240 256 to 240 259 & “Hello”
- 240 256 to 240 259 & “Aqua”

It is evident that the GID1 value “Hello” is a common requirement to several of the operators, and instead of making an entry for all the above listed elements, the grouping below can be used:

Code Group:
Sub Group 1:
NO Element: 238 01
NO Element: 238 02
NO Element: 240 256->259
SP Element: “Hello”
Sub Group 2:
NO Element: 238 01
SP Element: “Tie”
Sub Group 3:
NO Element: 238 02
SP Element: “Key”
SP Element: “Bye”
Sub Group 4:
NO Element: 240 256->259
SP Element: “Aqua”

The original requirement listed 7 groups, but utilizing the fact that some information is common to more entries, limits the number of groups to 4. Group 1 means that only 238 01 or 238 02 or 240 256..240 259 combined with “Hello” will be operated. Group 3 allows 238 02 with either “Key” or “Bye” to be operated.

The comparison data section (the code groups) is always constructed starting with the overall code group tag (SEC_TAG_CODE_GROUP). It is hereafter possible to use a number of sub code groups tags (SEC_TAG_GROUP) if it is reasonable to make a logical sub grouping of the comparison requirements. The sub group tag shall always be use in the code group description – also if there only is a need for one sub group. Which lock data elements (described in the following chapters) go into the different sub code groups depends on the lock in concern.

6.3.3.12.1 NO Element Object

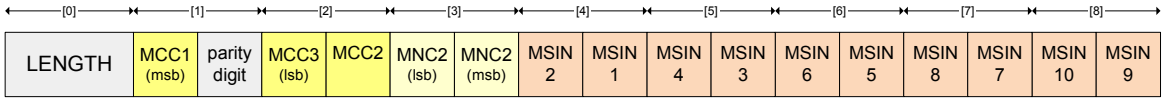
Network Operator Element Object:		
Byte(s)	**Description**	**Length**
1	NO Element TAG (0x32)	1
2	Length (X)	1
3	Qualifier	1
4 to X+3	NO Element data	X-1

Qualifier	Description	Byte(s)	Element data
0x01	Absolute MCC and	1	MCC#1 and MCC#2

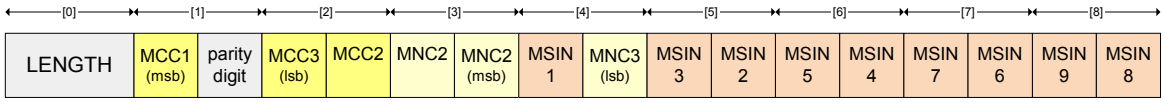
	absolute MNC	2	MCC#3 and MNC#1
		3	MNC#2 and MNC#3 (2 digit MNC -> MNC#3=F)
0x02	Absolute MCC with range of MNC: MNC_{low} -> MNC_{high} (both range limits included)	1	MCC#1 and MCC#2
		2	MCC#3 and MNC_{low}#1
		3	MNC_{low}#2 and MNC_{low}#3 (2 digit MNC -> MNC_{low}#3=F)
		4	F and MNC_{high}#1
		5	MNC_{high}#2 and MNC_{high}#3 (2 digit MNC -> MNC_{high}#3=F)
TBD			

Object examples:
Qualifier 1: 0x32 0x04 0x01 0x23 0x80 0x1F ~ MCC: 238 and MNC: 01
Qualifier 1: 0x32 0x04 0x01 0x23 0x82 0x46 ~ MCC: 238 and MNC: 246
Qualifier 2: 0x32 0x06 0x02 0x23 0x81 0x0F 0xF1 0x5F ~ MCC: 238 and MNC: [10..15]
Qualifier 2: 0x32 0x06 0x02 0x23 0x81 0x03 0xF1 0x24 ~ MCC: 238 and MNC: [103..124]

SIM IMSI structure - 2 digit MNC



SIM IMSI structure - 3 digit MNC



6.3.3.12.2 NS Element Object

Network Subset Element Object:		
Byte(s)	**Description**	**Length**
1	NS Element TAG (0x33)	1
2	Length (X)	1
3	Qualifier	1
4 to X+3	NS Element data	X-1

Qualifier	Description	Byte(s)	Element data
0x01	GSM 02.22 specified NS lock (MSIN digit 1 and 2; fixed values)	1	MSIN#1 and MSIN#2
0x02	One or more MSIN digits with fixed value (V denotes the number of values).	V	Digit position and Digit value.
0x03	2 MSIN digits (D_x and	1	D_x position and D_y position.

	D_y) in a range (R denotes the number of D_x/D_y ranges).	2 + (R-1)	$Range_{min}$ (D_{x_min}*10 + D_{y_min})
		3 + (R-1)	$Range_{max}$ (D_{x_max}*10 + D_{y_max})
0x04	2 MSIN digits (D_x and D_y) in a range based on a third MSIN digit D_z (R denotes the number of D_x/D_y ranges).	1	D_z position and D_z value.
		2	D_x position and D_y position.
		3 + (R-1)	$Range_{min}$ (D_{x_min}*10 + D_{y_min})
		4 + (R-1)	$Range_{max}$ (D_{x_max}*10 + D_{y_max})
0x05	Range of 3 MSIN digits (D_{y1}, D_{y2} and D_{y3}) based on another 2 digit MSIN range (D_{x1} and D_{x2}). R denotes the number of different D_x ranges. D_x Range value = D_{x1}*10+D_{x2} D_y Range value = D_{y1}*100+ D_{y2}*10+D_{y3}	1	D_{x1} position and D_{x2} position.
		2	D_{x1_min} and D_{x2_min}
		3	D_{x1_max} and D_{x2_max}
		4	D_{y1} position and D_{y2} position.
		5	D_{y3} position (LSB nipple set to 'F')
		6 + (R-1)	D_{y1_min} and D_{y2_min}
		7 + (R-1)	D_{y3_min} and D_{y3_max}
		8 + (R-1)	D_{y1_max} and D_{y2_max}
0x06	Any number of values, where each value is composed by max D=9 MSIN digits and must exists in one of max I=15 intervals. Notation N(X) means nibble X, where: N(1) = Byte 1, bit 4-7. N(2) = Byte 1, bit 0-3. N(3) = Byte 2, bit 4-7. N(4) = byte 2, bit 0-3. etc. Each value definition will use 2+2ID+D nibbles of the byte array.	N(1)	D = number if digits < 10.
		N(2)	I = number of intervals < 16.
		N(3) … N(2+D)	Nibbles giving positions of the D MSIN digits, starting with position of most significant value digit first.
		N(3+D) … N(2+D+D)	Nibbles specifying D digits of lower value for interval 1, starting with most significant digit first.
		N(3+2D) … N(2+2D+D)	Nibbles specifying D digits of upper value for interval 1, starting with most significant digit first.
		…	Specification of interval 2 to I-1.
		N(3+2ID-D) … N(2+2ID)	Nibbles specifying D digits of lower value for interval 1, starting with most significant digit first.
		N(3+2ID) … N(2+2ID+D)	Nibbles specifying D digits of upper value for interval I, starting with most significant digit first.
		N(3+2ID+D) … etc.	Nibbles specifying second value of MSIN digits which must be within certain intervals. Definition of: D_2, I_2, D_2 MSIN digits, I_2 intervals of D_2I_2 digits.
TBD	-	-	-

Note: Qualifier 0x06 is supported from SEC version 01.17 or later and 00.05S or later.

Object examples:

Qualifier 1: 0x33 0x02 0x01 0x67 ~ $MSIN_{1,2}$ = 6,7

Qualifier 2: 0x33 0x04 0x02 0x13 0x38 0x61 ~ $MSIN_{1,3,6}$ = 3,8,1

Qualifier 3: 0x33 0x08 0x03 0x24 0x23 0x29 0x36 0x36 0x45 0x51 ~ $MSIN_{24}$: [35..41] or 54 or [69..81]

Qualifier 4: 0x33 0x07 0x04 0x56 0x13 0x31 0x3A 0x43 0x43 ~ $MSIN_5$ = 6 and $MSIN_{13}$: [49..58] or 67
Qualifier 5: 0x33 0x09 0x05 0x12 0x14 0x20 0x67 0x8F 0x24 0x51 0x32 ~ $MSIN_{12}$: [14-20] and $MSIN_{678}$: [245..321]
Qualifier 6: 0x33 0x0D 0x06 0x32 0x34 0x51 0x11 0x22 0x23 0x33 0x44 0x41 0x29 0x56 0x89
~ $MSIN_{345}$: [111-222] or [333-444] and $MSIN_9$: [5-6] or [8-9]

6.3.3.12.3 SP Element Object

Service Provider Element Object:		
Byte(s)	**Description**	**Length**
1	SP Element TAG (0x34)	1
2	Length (X)	1
3	Qualifier	1
4	SIM file offset (O)	1
5	Number of bytes (N)	1
6 to X+5	SP Element data	X-3

Qualifier	Description	Byte(s)	Element data
0x01	GID1 data	1..N	GID1 data counting from O
0x02	Data from file specified in SIM file ID object	1..N	File data counting from O

Object example:
Qualifier 1, O=0, N=5: 0x34 0x08 0x01 0x00 0x05 0x48 0x65 0x6C 0x6C 0x6F ~ GID1: “Hello”

6.3.3.12.4 CP Element Object

Corporate Element Object:		
Byte(s)	**Description**	**Length**
1	CP Element TAG (0x35)	1
2	Length (X)	1
3	Qualifier	1
4	SIM file offset (O)	1
5	Number of bytes (N)	1
6 to X+5	CP Element data	X-3

Qualifier	Description	Byte(s)	Element data
0x01	GID2 data	1..N	GID2 data counting from O
TBD			

Object example:
Qualifier 1, O=0, N=3: 0x34 0x06 0x01 0x00 0x03 0x42 0x79 0x65 ~ GID2: “Bye”

6.3.3.12.5 SM Element Object

SIM Element Object:		
Byte(s)	**Description**	**Length**
1	SM element TAG (0x36)	1

2	Length (X)	1
3	Qualifier	1
4 to X+2	SM Element data	X-1

Qualifier	Description	Byte(s)	Element data
0x01	IMSI	1	IMSI length (N)
		2..N	IMSI as coded in GSM 11.11
TBD			

Object example:
Qualifier 1: 0x36 0x0A 0x01 0x08 0x29 0x83 0x20 0x51 0x00 0x70 0x90 0x41 ~ IMSI: 238021500070914

6.3.3.13 Lock Profile Object

Object used for carrying the lock profiles.

SIM File ID Object:		
Byte(s)	**Description**	**Length**
1	Lock Profiles TAG (0x50)	1
2	Length (X)	1
3 to X+2	Lock Profile data formatted in accordance with the structure T_SEC_LOCK_PROFILES (defined in sec.h)	X

6.3.3.14 Verification Result Object

Object used for carrying the verify control key result.

SIM File ID Object:		
Byte(s)	**Description**	**Length**
1	Verification Result TAG (0x56)	1
2	Length (X)	1
3 to X+2	Control key verification result data formatted in accordance with the structure T_SEC_VERIFY_RESULT (defined in sec.h)	X

6.3.3.15 Response Cause Object

Object giving a response to a given target command indicating command result cause.

SIM File ID Object:		
Byte(s)	**Description**	**Length**
1	Response Cause TAG (0x60)	1
2	Length (X)	1
3 to X+2	Cause data	X

Response cause	Value	Description
OK	0x00	The requested command action was successfully carried out.
UNKNOWN ERROR	0x01	An unknown error was encountered. A sub-cause for internal debugging will be attached in the following data element.

UNKNOWN REQ	0x02	Unknown command request.
UNKNOWN TAG	0x03	Unknown object TAG in command.
UNKNOWN LOCK ID	0x04	Unknown lock identification in command.
UNKNOWN CHARACTERISTICS	0x05	Unknown lock characteristics in command.
WRONG STATE	0x06	The requested action cannot be carried in the current lock state.
LOCK ID REQUIRED	0x07	The lock ID must be given in order to carry out the requested command.
UNLOCK KEY ERROR	0x08	Error in the unlock key format.
CODE GROUP CODING ERROR	0x09	Error in the code group data (comparison data).
LENGTH ERROR	0x0A	Error in object length.
WRONG PSC FILE FORMAT	0x0B	The PCS file script format is not supported.
PSC SYNTAX ERROR	0x0C	A syntax error was found in the PSC file.

7 Function Interface

This chapter contains a description of the interface functions provided by the SEC object. For detailed information regarding the specific data structures used in the function calls please refer to the “sec.h” include file.

7.1 IMEI related functions

7.1.1 SEC_imei_read

Function:	SEC_imei_read
Purpose:	Function for reading the IMEI in 04.08 mobile identity format (BCD) excluding LENGTH FIELD and IEI. The LSB nipple of the first BCD coded IMEI character must be set to 0xA (according to GSM 04.23). Example: if IMEI is 004999010640000 the return value will point to an array holding the following values: '0A 40 99 09 01 46 00 00'
Parameters:	None.
Return:	A pointer to a char array containing the IMEI. The function will return a NULL pointer if an invalid IMEI is detected.

7.1.2 SEC_imei_read_ascii

Function:	SEC_imei_read_ascii
Purpose:	Function for reading the IMEI in an ASCII format consisting of 14 digits and a check digit. The format is the one used when the *#06# is activated from the keyboard. The check digit (digit 15 is calculated in accordance with GSM 02.16). Example: if IMEI is 004999010640000 the return value will point to an array holding the following values: "004999010640000"
Parameters:	None.
Return:	A pointer to a char array containing the IMEI. The function will return a NULL pointer if an invalid IMEI is detected.

7.1.3 SEC_store_imei

Function:	SEC_store_imei		
Purpose:	This function is used for storing the IMEI.		
Parameters:	I	*imei	Pointer to an array containing the IMEI to be stored - in ASCII format consisting of 14 digits and a check digit. The format is the one used when the *#06# is activated from the keyboard. The check digit (digit 15 is calculated in accordance with GSM 02.16). Example: input IMEI must be "004999010640000" if the IMEI is 004999010640000.
	I	*index	Pointer to a random number.

Return:	Programming result: atctst_prog_completed, atctst_already_programmed, atctst_content_error or atctst_programming_error

7.1.4 SEC_verify_imei

Function:	SEC_verify_imei
Purpose:	Function for verifying that a real IMEI has been programmed.
Parameters:	None.
Return:	True: A real IMEI has been programmed and it is valid. False: No real IMEI s programmed.

7.2 SIM Lock related functions

7.2.1 SEC_get_lock_profiles

Function:	SEC_get_lock_profiles			
Purpose:	Function to be used to get the number of locks and the profile of each of the locks.			
Parameters:	O	*nof_locks	Number of supported locks.	
	O	*nof_required_files	Number of files required for the personalization mechanism. The actual file characteristics is obtained using the function SEC_get_file_profile.	
	O	*lock[nof_locks]	lock_id	The lock ID for the following information.
			lock_state	The state in which the lock resides in.
			dck_supported	Indicates whether or not Depersonalization Control Key (DCK) is supported for this lock.
			retrial_method	Indicates the trial method set for this lock; whether maximum retrial or time limited separation is set [see 6.3.3.4].
			time_to_verification	Indicates the time to next possible unlock verification (if unlimited unlock is set for this lock).
			retrial_cnt	The remaining number of retrials (if limited unlock verification is set for this lock).
			lock_order	The order in which the unlocking should be carried out [see 6.3.3.10]

	O	*mmi_configuration[0]	Holds a bit-field for configuring the MMI for the overall lock behavior. **Unlocking configuration using right SIM:** B1B0=00: Unlocking allowed, both via MENU and keyboard string. B1B0=x1: Unlocking not allowed, via MENU. B1B0=1x: Unlocking not allowed, via keyboard string. **Unlocking configuration using incorrect SIM:** B3B2=00: Unlocking allowed, both via power on prompt and keyboard string. B3B2=x1: Unlocking not allowed via prompting user at power on. B3B2=1x: Unlocking not allowed via keyboard string. **Unlocking configuration without SIM:** B5B4=00: Unlocking allowed, both via MENU and keyboard string. B5B4=x1: Unlocking not allowed via MENU. B5B4=1x: Unlocking not allowed via keyboard string.
	O	*mmi_configuration[1]	RFU
Return:	True: Function execution was successfully carried out. False: Function execution failed.		

7.2.2 SEC_get_file_profile

Function:	SEC_get_file_profile			
Purpose:	Function to be used to obtain the characteristics for each of the files indicated in the mask file_id_mask from the function SEC_get_lock_profile.			
Parameters:	I	file_id	The file number for each of the files indicated in the function SEC_get_lock_profile. E.g. if nof_required_files indicates 3 from the function SEC_get_lock_profile, the allowed file_id of this function can either be 1, 2 or 3. This file_id is used directly when obtaining the actual file data using function SEC_compare_lock_data.	
	O	*file_profile	The actual file characteristics for the file indicated in the file_id input parameter. The file characteristics are described by:	
			file_path	The absolute path to the file as described in GSM 11.11 (the MF is implicit part of the path).
			file_type_flag	Indicates the file type: Transparent or record (linear or cyclic).
			Transparent_offset	The offset into a transparent file.
			Size	data size (transparent file) or data size and record number (for the record based file).
Return:	True: Function execution was successfully carried out. False: Function execution failed.			

7.2.3 SEC_compare_lock_data

Function:	SEC_compare_lock_data
Purpose:	Function to be used to carry out the comparison of the data read from the SIM.

Parameters:	I	*cnl	The CNL data and the number of bytes in the CNL list.	
	I	*data_ptr_array	A pointer to a store holding:	
			cont_idx[]	Contents index where the first byte holds the number of entries the number of returned files). The following bytes are indexes to the cont[] array (the first of them is always zero).
			cont[]	Contents array holding a iteration over: - number_of_bytes - databytes[number of bytes] If a required file does not exist or is not readable the number_of_bytes entry for that file will be zero and there won't be any databytes entry for it. The sequence of the data begins with the data from the file required for the first lock and ends with that one for the last one.
	O	*result	A pointer to a store holding:	
			cause	The result cause for the operation.
			next_verification	If control key (unlock key) verifications is needed, the lock ID is indicated here.
			lock_status	Array holding the lock status of the verified locks, each element represented by the following information: - lock id: The lock identification - lock status: The lock state.
Return:	True: Function execution was successfully carried out. False: Function execution failed.			

7.2.4 SEC_minute_tick

Function:	SEC_minute_tick
Purpose:	Function to be called by the Real Time Clock unit every minute until powerdown.
Parameters:	None.
Return:	None.

7.2.5 SEC_ptst_lock_control

Function:	SEC_ptst_lock_control		
Purpose:	Function for feeding control data and lock data between PC and SEC (via ATCPTST).		
Parameters:	I	req_len	Request data length.
	I	*req_data_ptr	Pointer to request data.
	O	*con_len	Pointer to response data length.
	O	*con_data_ptr	Pointer to response data length.
Return:	True: Function execution was successfully carried out. False: Function execution failed.		

7.2.6 SEC_verify_control_key

Function:	SEC_verify_control_key

Purpose:	Function to be used to carry out the verification of the control keys. This function can be used iterative in the startup sequence or as a standalone call should the user chose to unlock one of the personalization mechanisms - either after having powered up or without SIM inserted. This function must also be used to carry control keys from the DCK file after a DCK related Refresh.			
Parameters:	I	lock_id	The lock identification.	
	I	*control_key	A pointer to the control key, containing the ASCII formatted control key (unlock key) and the length of that key.	
	O	*result	The result of the control key verification containing the following information:	
			cause	The result cause for the operation.
			next_verification	If control key (unlock key) verifications is needed, the lock ID is indicated here.
			lock_status	Array holding the lock status of the verified locks, each element represented by the following information: - lock id: The lock identification - lock status: The lock state.
			time_to_verification	The time until a new control key verification is allowed (in minutes). If time limited verification is supported.
			retrial_cnt	The retrial counter in case maximum retrial limit is supported.
Return:	True: Function execution was successfully carried out. False: Function execution failed.			

7.2.7 SEC_apply_customer_key

Function:	SEC_apply_customer_key		
Purpose:	Function for applying a customer key.		
Parameters:	I	*key	Pointer to a 16 character array containing a customer key.
Return:	None.		

7.3 Other Functions

7.3.1 SEC_unique_id_read

Function:	SEC_unique_id_read
Purpose:	Function for reading the unique id. This function is available on SEC version 01.17 or later and 00.02S or later.
Parameters:	None.
Return:	A pointer to a 16-byte array containing the unique id for the phone. If id has less than 16 bytes, the remaining bytes are set to zero.

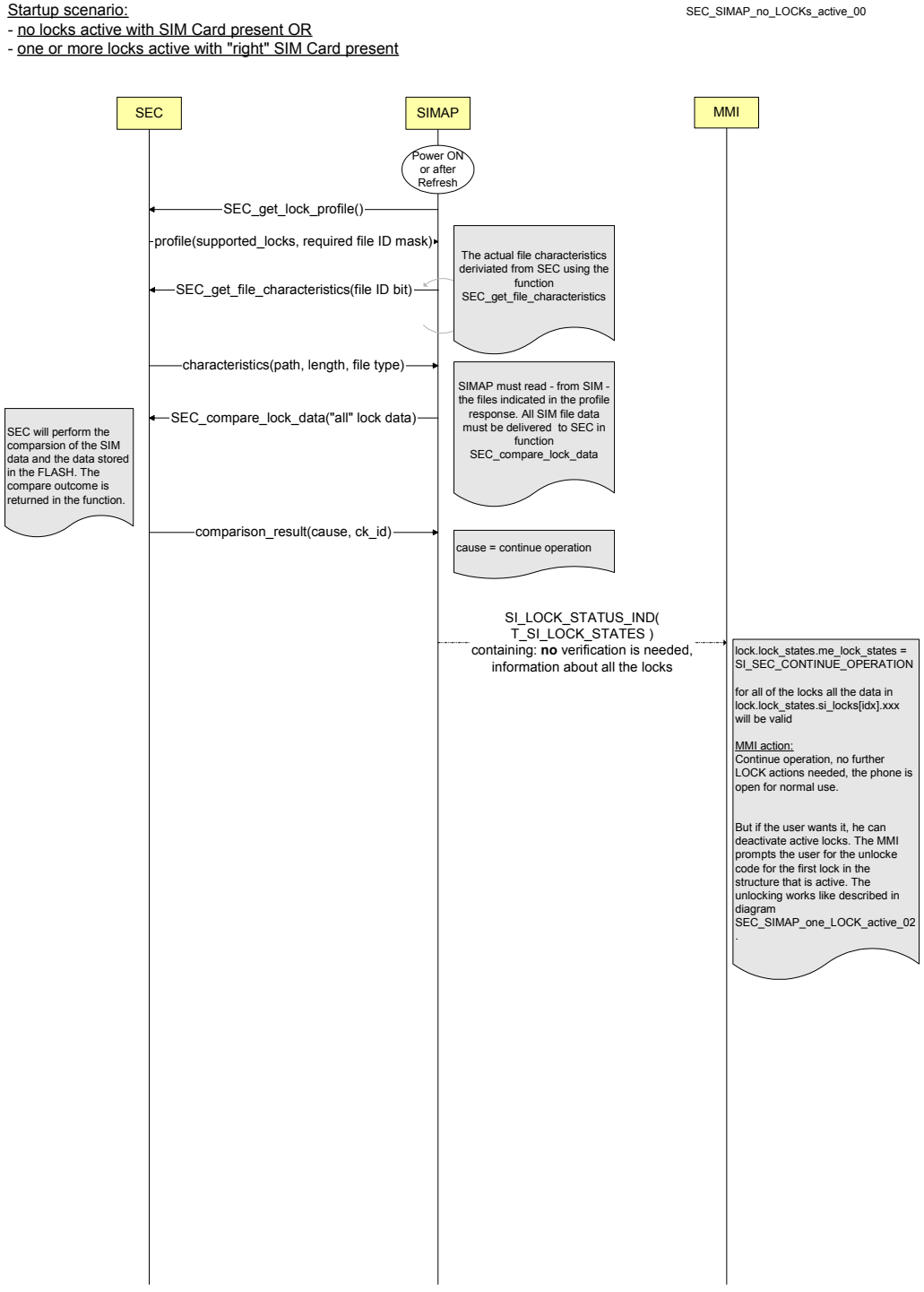
8 Appendix A: List of TAGS

Tag name	Identification
PSC File Version TAG	0x0F
Lock ID TAG	0x10
Overall Characteristics TAG	0x15
Lock Characteristics TAG	0x20
Retrial counter TAG	0x21
Time Verification Method TAG	0x22
Unlock Code TAG	0x25
ARM Characteristics TAG	0x26
Disable TAG	0x27
Unlock Order TAG	0x28
SIM File ID TAG	0x29
Code Group TAG	0x30
Group TAG	0x31
NO Element TAG	0x32
NS Element TAG	0x33
SP Element TAG	0x34
CP Element TAG	0x35
SM Element TAG	0x36
Lock Profiles TAG	0x50
Verification Result TAG	0x56
Response Cause TAG	0x60

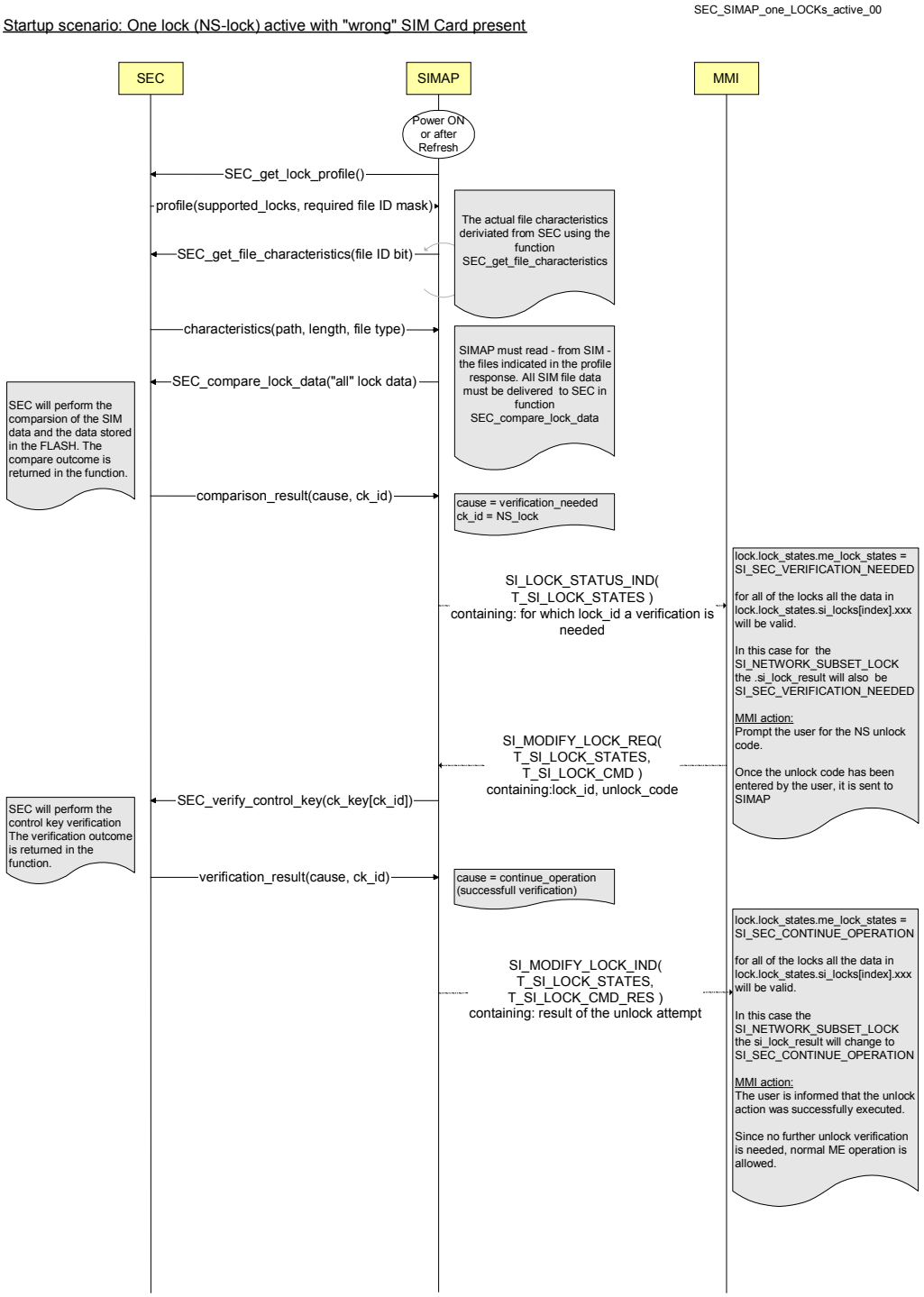
9 Appendix B: Module Flow Examples

The following sections hold a number of examples of some different startup scenarios with respect to personalization. Please note that the examples are not exhaustive and are merely included to give the reader an idea of how the three identities interact.

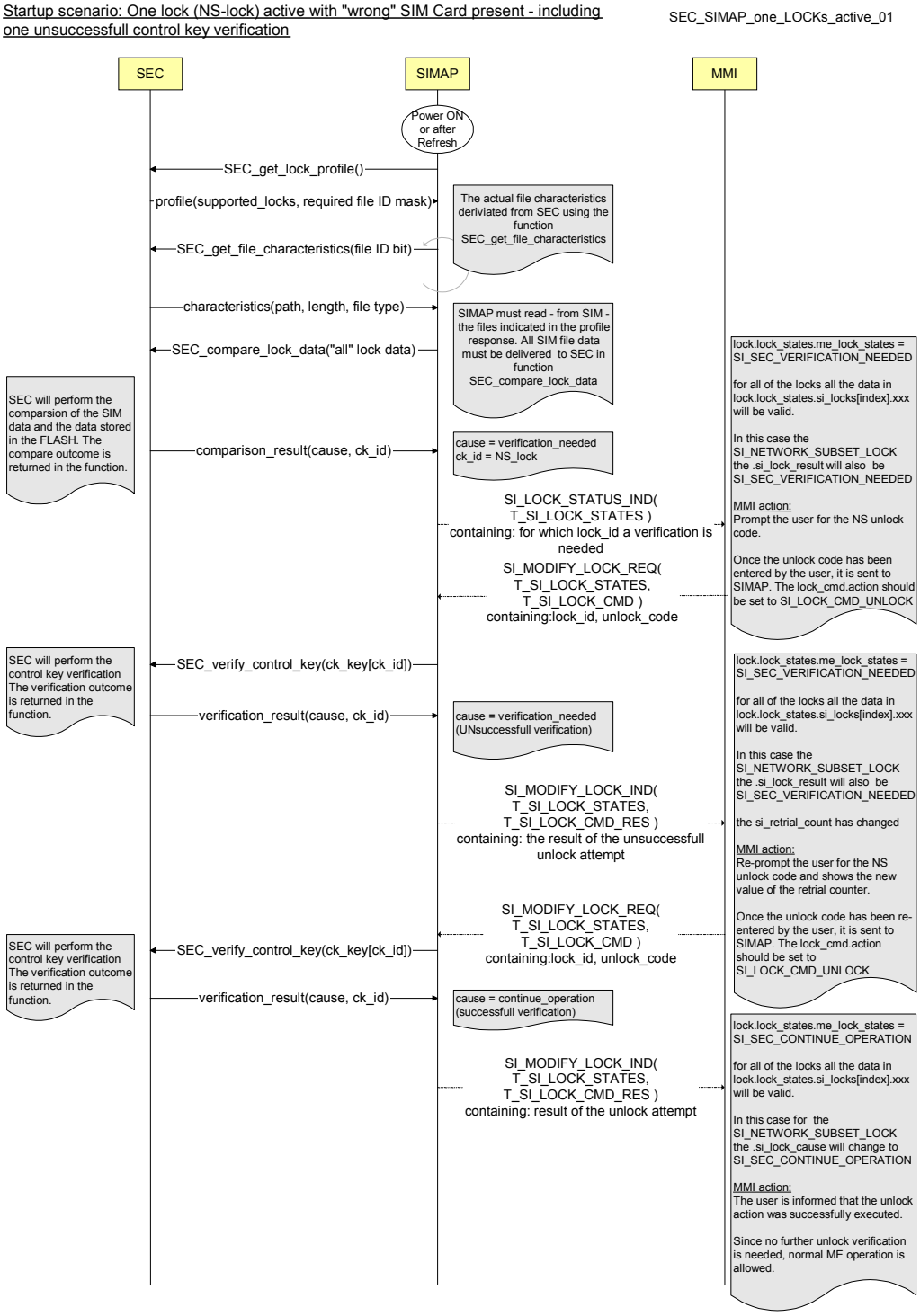
9.1 Example 1


Startup scenario:
- no locks active with SIM Card present OR
- one or more locks active with "right" SIM Card present
SEC_SIMAP_no_LOCKs_active_00
SEC
SIMAP
MMI
Power ON or after Refresh
SEC_get_lock_profile()
profile(supported_locks, required file ID mask)
The actual file characteristics deriviated from SEC using the function SEC_get_file_characteristics
SEC_get_file_characteristics(file ID bit)
characteristics(path, length, file type)
SIMAP must read - from SIM - the files indicated in the profile response. All SIM file data must be delivered to SEC in function SEC_compare_lock_data
SEC_compare_lock_data("all" lock data)
SEC will perform the comparsion of the SIM data and the data stored in the FLASH. The compare outcome is returned in the function.
comparison_result(cause, ck_id)
cause = continue operation
SI_LOCK_STATUS_IND(T_SI_LOCK_STATES) containing: no verification is needed, information about all the locks
lock.lock_states.me_lock_states = SI_SEC_CONTINUE_OPERATION
for all of the locks all the data in lock.lock_states.si_locks[idx].xxx will be valid
MMI action:
Continue operation, no further LOCK actions needed, the phone is open for normal use.
But if the user wants it, he can deactivate active locks. The MMI prompts the user for the unlocke code for the first lock in the structure that is active. The unlocking works like described in diagram SEC_SIMAP_one_LOCK_active_02 .

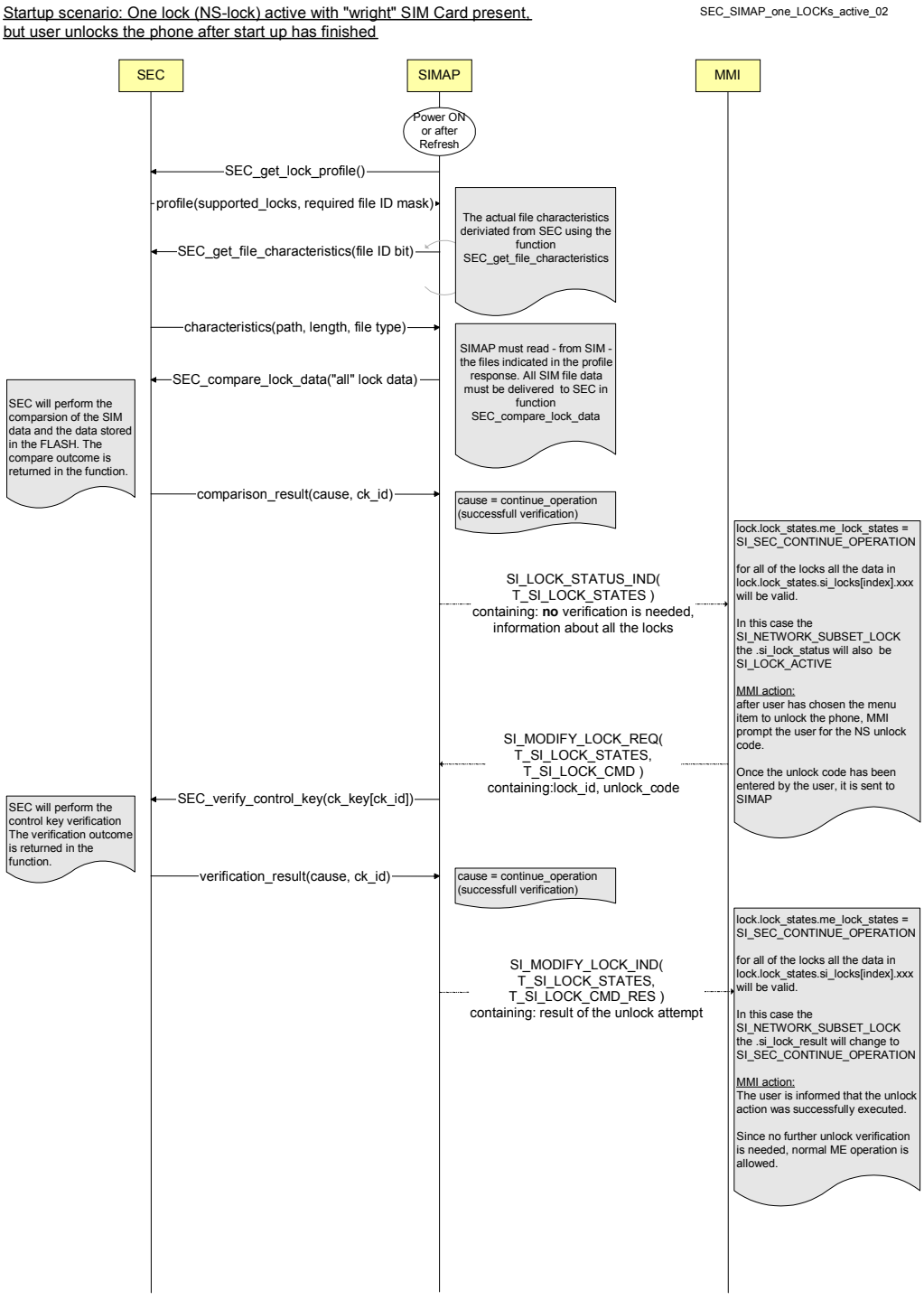
9.2 Example 2


SEC_SIMAP_one_LOCKs_active_00
Startup scenario: One lock (NS-lock) active with "wrong" SIM Card present
SEC
SIMAP
MMI
Power ON or after Refresh
SEC_get_lock_profile()
profile(supported_locks, required file ID mask)
The actual file characteristics deriviated from SEC using the function SEC_get_file_characteristics
SEC_get_file_characteristics(file ID bit)
characteristics(path, length, file type)
SIMAP must read - from SIM - the files indicated in the profile response. All SIM file data must be delivered to SEC in function SEC_compare_lock_data
SEC_compare_lock_data("all" lock data)
SEC will perform the comparsion of the SIM data and the data stored in the FLASH. The compare outcome is returned in the function.
comparison_result(cause, ck_id)
cause = verification_needed
ck_id = NS_lock
SI_LOCK_STATUS_IND(T_SI_LOCK_STATES) containing: for which lock_id a verification is needed
lock.lock_states.me_lock_states = SI_SEC_VERIFICATION_NEEDED
for all of the locks all the data in lock.lock_states.si_locks[index].xxx will be valid.
In this case for the SI_NETWORK_SUBSET_LOCK the .si_lock_result will also be SI_SEC_VERIFICATION_NEEDED
MMI action:
Prompt the user for the NS unlock code.
Once the unlock code has been entered by the user, it is sent to SIMAP
SI_MODIFY_LOCK_REQ(T_SI_LOCK_STATES, T_SI_LOCK_CMD) containing:lock_id, unlock_code
SEC_verify_control_key(ck_key[ck_id])
SEC will perform the control key verification The verification outcome is returned in the function.
verification_result(cause, ck_id)
cause = continue_operation (successfull verification)
SI_MODIFY_LOCK_IND(T_SI_LOCK_STATES, T_SI_LOCK_CMD_RES) containing: result of the unlock attempt
lock.lock_states.me_lock_states = SI_SEC_CONTINUE_OPERATION
for all of the locks all the data in lock.lock_states.si_locks[index].xxx will be valid.
In this case the SI_NETWORK_SUBSET_LOCK the si_lock_result will change to SI_SEC_CONTINUE_OPERATION
MMI action:
The user is informed that the unlock action was successfully executed.
Since no further unlock verification is needed, normal ME operation is allowed.

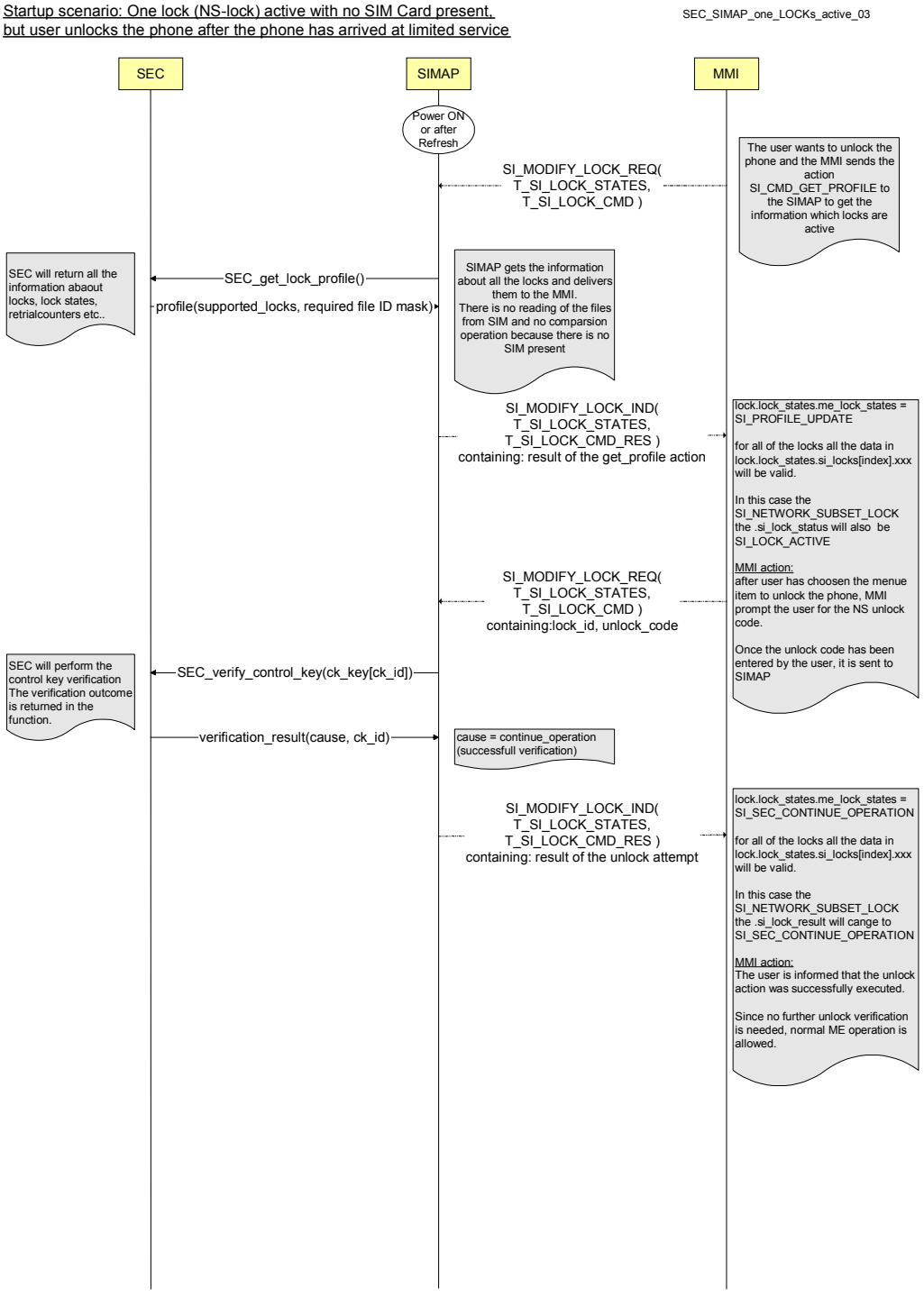
9.3 Example 3


Startup scenario: One lock (NS-lock) active with "wrong" SIM Card present - including one unsuccessfull control key verification
SEC_SIMAP_one_LOCKs_active_01
SEC
SIMAP
MMI
Power ON or after Refresh
SEC_get_lock_profile()
profile(supported_locks, required file ID mask)
The actual file characteristics deriviated from SEC using the function SEC_get_file_characteristics
SEC_get_file_characteristics(file ID bit)
characteristics(path, length, file type)
SIMAP must read - from SIM - the files indicated in the profile response. All SIM file data must be delivered to SEC in function SEC_compare_lock_data
SEC_compare_lock_data("all" lock data)
lock.lock_states.me_lock_states = SI_SEC_VERIFICATION_NEEDED
for all of the locks all the data in lock.lock_states.si_locks[index].xxx will be valid.
In this case the SI_NETWORK_SUBSET_LOCK the .si_lock_result will also be SI_SEC_VERIFICATION_NEEDED
MMI action:
Prompt the user for the NS unlock code.
Once the unlock code has been entered by the user, it is sent to SIMAP. The lock_cmd.action should be set to SI_LOCK_CMD_UNLOCK
SEC will perform the comparsion of the SIM data and the data stored in the FLASH. The compare outcome is returned in the function.
comparison_result(cause, ck_id)
cause = verification_needed
ck_id = NS_lock
SI_LOCK_STATUS_IND(T_SI_LOCK_STATES) containing: for which lock_id a verification is needed
SI_MODIFY_LOCK_REQ(T_SI_LOCK_STATES, T_SI_LOCK_CMD) containing:lock_id, unlock_code
SEC will perform the control key verification The verification outcome is returned in the function.
SEC_verify_control_key(ck_key[ck_id])
verification_result(cause, ck_id)
cause = verification_needed (UNsuccessfull verification)
lock.lock_states.me_lock_states = SI_SEC_VERIFICATION_NEEDED
for all of the locks all the data in lock.lock_states.si_locks[index].xxx will be valid.
In this case the SI_NETWORK_SUBSET_LOCK the .si_lock_result will also be SI_SEC_VERIFICATION_NEEDED
the si_retrial_count has changed
MMI action:
Re-prompt the user for the NS unlock code and shows the new value of the retrial counter.
Once the unlock code has been re-entered by the user, it is sent to SIMAP. The lock_cmd.action should be set to SI_LOCK_CMD_UNLOCK
SI_MODIFY_LOCK_IND(T_SI_LOCK_STATES, T_SI_LOCK_CMD_RES) containing: the result of the unsuccessfull unlock attempt
SI_MODIFY_LOCK_REQ(T_SI_LOCK_STATES, T_SI_LOCK_CMD) containing:lock_id, unlock_code
SEC will perform the control key verification The verification outcome is returned in the function.
SEC_verify_control_key(ck_key[ck_id])
verification_result(cause, ck_id)
cause = continue_operation (successfull verification)
lock.lock_states.me_lock_states = SI_SEC_CONTINUE_OPERATION
for all of the locks all the data in lock.lock_states.si_locks[index].xxx will be valid.
In this case for the SI_NETWORK_SUBSET_LOCK the .si_lock_cause will change to SI_SEC_CONTINUE_OPERATION
MMI action:
The user is informed that the unlock action was successfully executed.
Since no further unlock verification is needed, normal ME operation is allowed.
SI_MODIFY_LOCK_IND(T_SI_LOCK_STATES, T_SI_LOCK_CMD_RES) containing: result of the unlock attempt

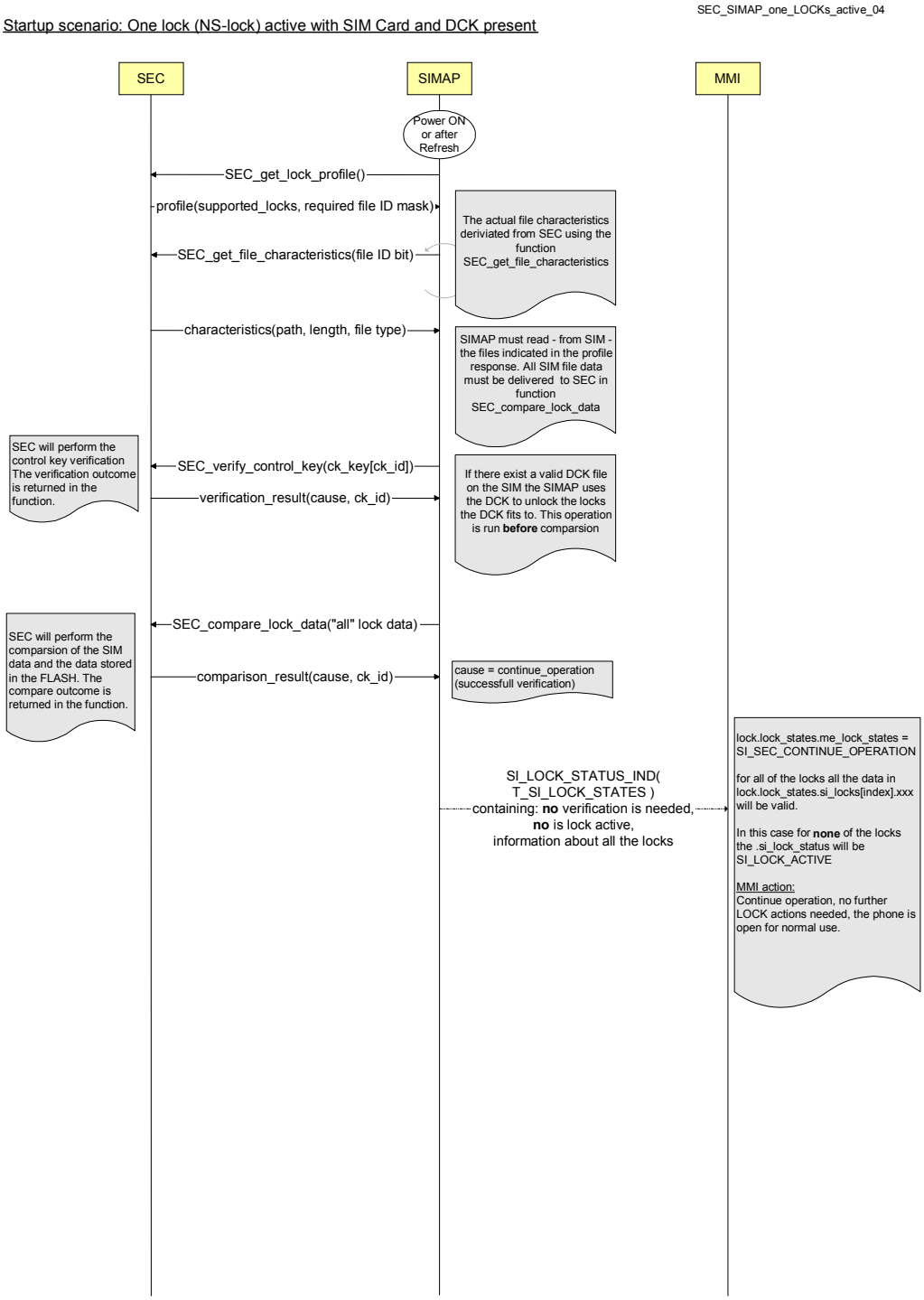
9.4 Example 4


Startup scenario: One lock (NS-lock) active with "wright" SIM Card present, but user unlocks the phone after start up has finished
SEC_SIMAP_one_LOCKs_active_02
SEC
SIMAP
MMI
Power ON or after Refresh
SEC_get_lock_profile()
profile(supported_locks, required file ID mask)
The actual file characteristics deriviated from SEC using the function SEC_get_file_characteristics
SEC_get_file_characteristics(file ID bit)
characteristics(path, length, file type)
SIMAP must read - from SIM - the files indicated in the profile response. All SIM file data must be delivered to SEC in function SEC_compare_lock_data
SEC_compare_lock_data("all" lock data)
SEC will perform the comparsion of the SIM data and the data stored in the FLASH. The compare outcome is returned in the function.
comparison_result(cause, ck_id)
cause = continue_operation (successfull verification)
lock.lock_states.me_lock_states = SI_SEC_CONTINUE_OPERATION
for all of the locks all the data in lock.lock_states.si_locks[index].xxx will be valid.
In this case the SI_NETWORK_SUBSET_LOCK the .si_lock_status will also be SI_LOCK_ACTIVE
MMI action:
after user has chosen the menu item to unlock the phone, MMI prompt the user for the NS unlock code.
Once the unlock code has been entered by the user, it is sent to SIMAP
SI_LOCK_STATUS_IND(T_SI_LOCK_STATES) containing: no verification is needed, information about all the locks
SI_MODIFY_LOCK_REQ(T_SI_LOCK_STATES, T_SI_LOCK_CMD) containing:lock_id, unlock_code
SEC_verify_control_key(ck_key[ck_id])
SEC will perform the control key verification The verification outcome is returned in the function.
verification_result(cause, ck_id)
cause = continue_operation (successfull verification)
lock.lock_states.me_lock_states = SI_SEC_CONTINUE_OPERATION
for all of the locks all the data in lock.lock_states.si_locks[index].xxx will be valid.
In this case the SI_NETWORK_SUBSET_LOCK the .si_lock_result will change to SI_SEC_CONTINUE_OPERATION
MMI action:
The user is informed that the unlock action was successfully executed.
Since no further unlock verification is needed, normal ME operation is allowed.
SI_MODIFY_LOCK_IND(T_SI_LOCK_STATES, T_SI_LOCK_CMD_RES) containing: result of the unlock attempt

9.5 Example 5


Startup scenario: One lock (NS-lock) active with no SIM Card present,
but user unlocks the phone after the phone has arrived at limited service
SEC_SIMAP_one_LOCKs_active_03
SEC
SIMAP
MMI
Power ON or after Refresh
SI_MODIFY_LOCK_REQ(T_SI_LOCK_STATES, T_SI_LOCK_CMD)
The user wants to unlock the phone and the MMI sends the action SI_CMD_GET_PROFILE to the SIMAP to get the information which locks are active
SEC will return all the information abaout locks, lock states, retrialcounters etc..
SEC_get_lock_profile()
profile(supported_locks, required file ID mask)
SIMAP gets the information about all the locks and delivers them to the MMI.
There is no reading of the files from SIM and no comparsion operation because there is no SIM present
SI_MODIFY_LOCK_IND(T_SI_LOCK_STATES, T_SI_LOCK_CMD_RES)
containing: result of the get_profile action
lock.lock_states.me_lock_states = SI_PROFILE_UPDATE
for all of the locks all the data in lock.lock_states.si_locks[index].xxx will be valid.
In this case the SI_NETWORK_SUBSET_LOCK the .si_lock_status will also be SI_LOCK_ACTIVE
MMI action:
after user has choosen the menue item to unlock the phone, MMI prompt the user for the NS unlock code.
Once the unlock code has been entered by the user, it is sent to SIMAP
SI_MODIFY_LOCK_REQ(T_SI_LOCK_STATES, T_SI_LOCK_CMD)
containing:lock_id, unlock_code
SEC will perform the control key verification The verification outcome is returned in the function.
SEC_verify_control_key(ck_key[ck_id])
verification_result(cause, ck_id)
cause = continue_operation (successfull verification)
SI_MODIFY_LOCK_IND(T_SI_LOCK_STATES, T_SI_LOCK_CMD_RES)
containing: result of the unlock attempt
lock.lock_states.me_lock_states = SI_SEC_CONTINUE_OPERATION
for all of the locks all the data in lock.lock_states.si_locks[index].xxx will be valid.
In this case the SI_NETWORK_SUBSET_LOCK the .si_lock_result will cange to SI_SEC_CONTINUE_OPERATION
MMI action:
The user is informed that the unlock action was successfully executed.
Since no further unlock verification is needed, normal ME operation is allowed.

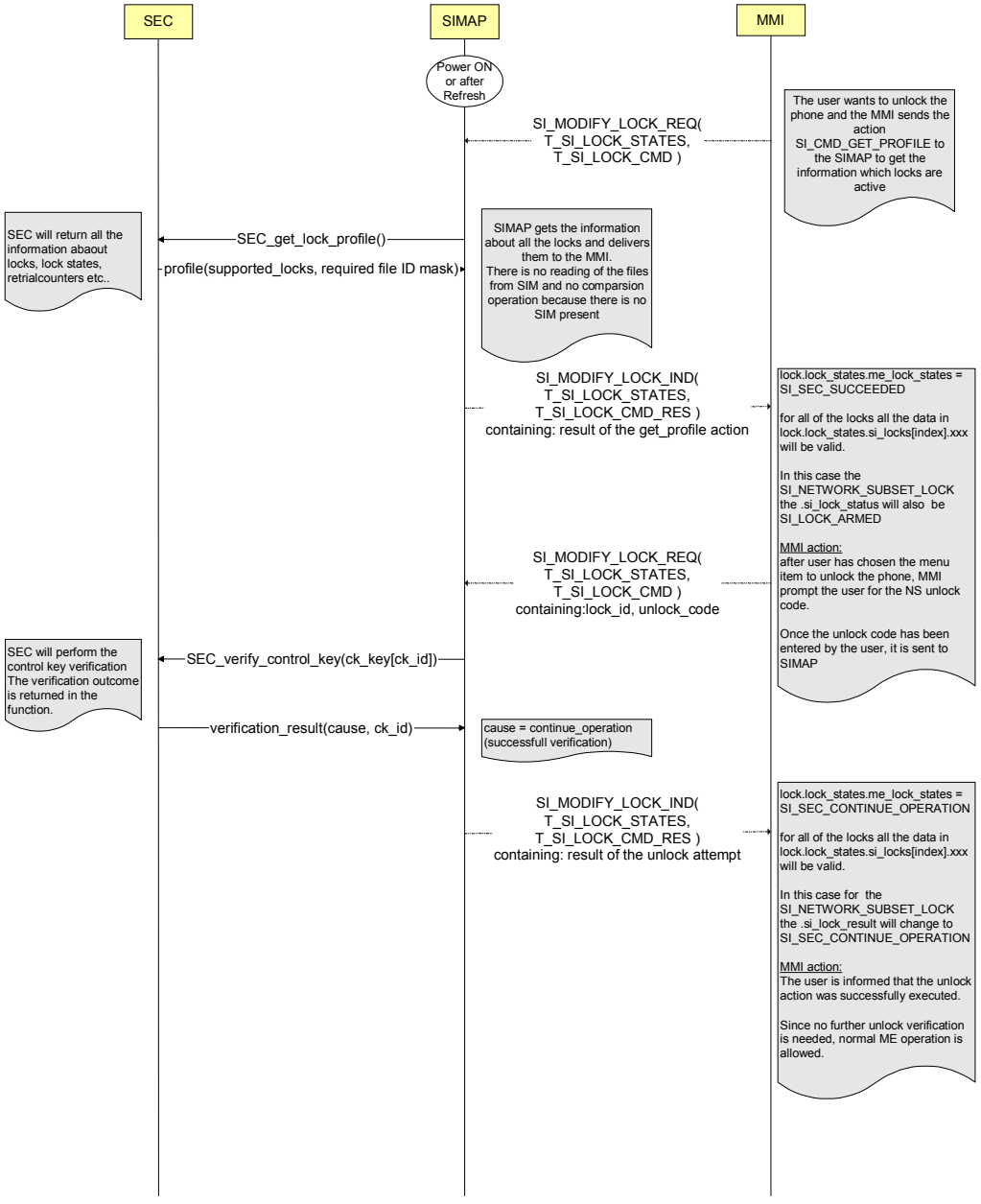
9.6 Example 6


SEC_SIMAP_one_LOCKs_active_04
Startup scenario: One lock (NS-lock) active with SIM Card and DCK present
SEC
SIMAP
MMI
Power ON or after Refresh
SEC_get_lock_profile()
profile(supported_locks, required file ID mask)
The actual file characteristics deriviated from SEC using the function SEC_get_file_characteristics
SEC_get_file_characteristics(file ID bit)
characteristics(path, length, file type)
SIMAP must read - from SIM - the files indicated in the profile response. All SIM file data must be delivered to SEC in function SEC_compare_lock_data
SEC will perform the control key verification The verification outcome is returned in the function.
SEC_verify_control_key(ck_key[ck_id])
verification_result(cause, ck_id)
If there exist a valid DCK file on the SIM the SIMAP uses the DCK to unlock the locks the DCK fits to. This operation is run **before** comparsion
SEC will perform the comparsion of the SIM data and the data stored in the FLASH. The compare outcome is returned in the function.
SEC_compare_lock_data("all" lock data)
comparison_result(cause, ck_id)
cause = continue_operation (successfull verification)
SI_LOCK_STATUS_IND(T_SI_LOCK_STATES) containing: **no** verification is needed, **no** is lock active, information about all the locks
lock.lock_states.me_lock_states = SI_SEC_CONTINUE_OPERATION
for all of the locks all the data in lock.lock_states.si_locks[index].xxx will be valid.
In this case for **none** of the locks the .si_lock_status will be SI_LOCK_ACTIVE
MMI action:
Continue operation, no further LOCK actions needed, the phone is open for normal use.

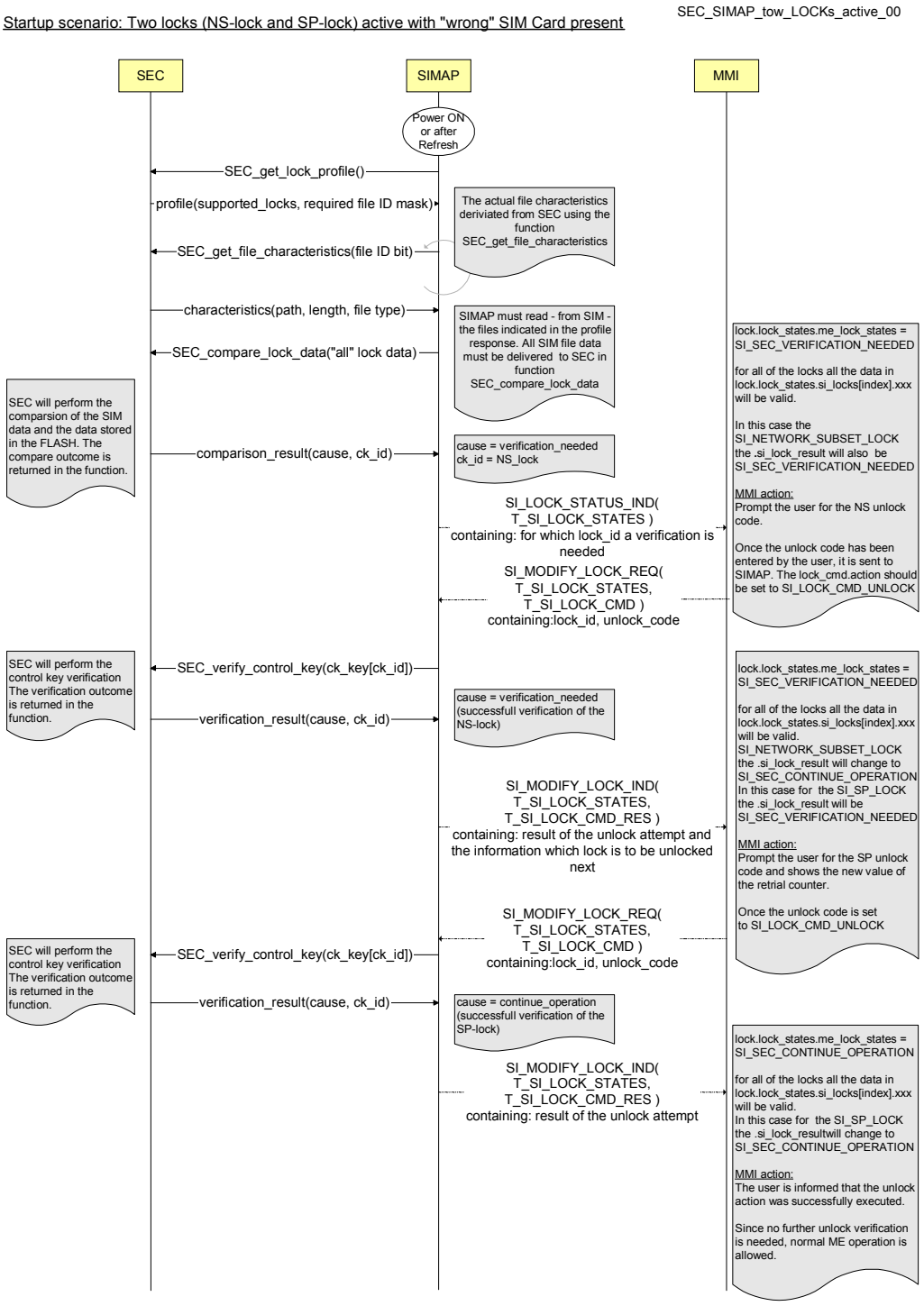
9.7 Example 7

Startup scenario: One lock (NS-lock) armed with no SIM Card present and there wasn't ever a simcard inserted since the state of the NS-lock was changed to ARMED, but user unlocks the phone after the phone has arrived at limited service

SEC_SIMAP_one_LOCKs_active_05



9.8 Example 8


SEC_SIMAP_tow_LOCKs_active_00
Startup scenario: Two locks (NS-lock and SP-lock) active with "wrong" SIM Card present
SEC
SIMAP
MMI
Power ON or after Refresh
SEC_get_lock_profile()
profile(supported_locks, required file ID mask)
The actual file characteristics deriviated from SEC using the function SEC_get_file_characteristics
SEC_get_file_characteristics(file ID bit)
characteristics(path, length, file type)
SIMAP must read - from SIM - the files indicated in the profile response. All SIM file data must be delivered to SEC in function SEC_compare_lock_data
SEC_compare_lock_data("all" lock data)
lock.lock_states.me_lock_states = SI_SEC_VERIFICATION_NEEDED
for all of the locks all the data in lock.lock_states.si_locks[index].xxx will be valid.
In this case the SI_NETWORK_SUBSET_LOCK the .si_lock_result will also be SI_SEC_VERIFICATION_NEEDED
MMI action:
Prompt the user for the NS unlock code.
Once the unlock code has been entered by the user, it is sent to SIMAP. The lock_cmd.action should be set to SI_LOCK_CMD_UNLOCK
SEC will perform the comparsion of the SIM data and the data stored in the FLASH. The compare outcome is returned in the function.
comparison_result(cause, ck_id)
cause = verification_needed
ck_id = NS_lock
SI_LOCK_STATUS_IND(T_SI_LOCK_STATES)
containing: for which lock_id a verification is needed
SI_MODIFY_LOCK_REQ(T_SI_LOCK_STATES, T_SI_LOCK_CMD)
containing:lock_id, unlock_code
SEC will perform the control key verification The verification outcome is returned in the function.
SEC_verify_control_key(ck_key[ck_id])
verification_result(cause, ck_id)
cause = verification_needed (successfull verification of the NS-lock)
lock.lock_states.me_lock_states = SI_SEC_VERIFICATION_NEEDED
for all of the locks all the data in lock.lock_states.si_locks[index].xxx will be valid.
SI_NETWORK_SUBSET_LOCK the .si_lock_result will change to SI_SEC_CONTINUE_OPERATION
In this case for the SI_SP_LOCK the .si_lock_result will be SI_SEC_VERIFICATION_NEEDED
MMI action:
Prompt the user for the SP unlock code and shows the new value of the retrial counter.
Once the unlock code is set to SI_LOCK_CMD_UNLOCK
SI_MODIFY_LOCK_IND(T_SI_LOCK_STATES, T_SI_LOCK_CMD_RES)
containing: result of the unlock attempt and the information which lock is to be unlocked next
SI_MODIFY_LOCK_REQ(T_SI_LOCK_STATES, T_SI_LOCK_CMD)
containing:lock_id, unlock_code
SEC will perform the control key verification The verification outcome is returned in the function.
SEC_verify_control_key(ck_key[ck_id])
verification_result(cause, ck_id)
cause = continue_operation (successfull verification of the SP-lock)
lock.lock_states.me_lock_states = SI_SEC_CONTINUE_OPERATION
for all of the locks all the data in lock.lock_states.si_locks[index].xxx will be valid.
In this case for the SI_SP_LOCK the .si_lock_resultwill change to SI_SEC_CONTINUE_OPERATION
MMI action:
The user is informed that the unlock action was successfully executed.
Since no further unlock verification is needed, normal ME operation is allowed.
SI_MODIFY_LOCK_IND(T_SI_LOCK_STATES, T_SI_LOCK_CMD_RES)
containing: result of the unlock attempt

10 Appendix C: PSC Examples

The following sections hold examples of how the programming scripts for the different lock mechanisms can be constructed.

10.1 Program NO Lock – ACTIVATE

```
/**
 *       Copyright (C) Danish Wireless Design A/S. All rights reserved.
 *
 * This document contains proprietary information belonging to Danish Wireless
 * Design A/S. Passing on and copying of this document, use and communication
 * of its contents is not permitted without prior written authorization.
 *
 * Description:
 *   NO Activation example.
 *
 * Revision Information:
 *   File name: NO_personalize_ACTIVE.psc
 *   Version: \main\1
 *   Date: 2003-02-18 09:38:43
 *   Responsible: knm
 *   Comment:
 *     -
 */


0x01 //SEC_TAG_PERSONALIZE_REQ
0x0F //SEC_TAG_PSC_FILE_VERSION
1    //Length
0x01 //Version
0x10 //SEC_TAG_LOCK_ID
1    //Length
0x00 //SEC_NO_LOCK
0x20 //SEC_TAG_LOCK_CHARACTERISTICS
1    //Length
0x07 //Lock Characteristics: Unlock allowed, CNL supported, DCK supported
0x25 //SEC_TAG_UNLOCK_CODE
8    //Length
0x31 0x32 0x33 0x34 0x35 0x36 0x37 0x38 //Unlock code
0x30 //SEC_TAG_CODE_GROUP
38   //Length
0x31 //SEC_TAG_GROUP
18   //Length
0x32 //SEC_TAG_NO_ELEMENT
4 0x01 0x23 0x80 0x1F,                    /* 238 01      */
0x32 //SEC_TAG_NO_ELEMENT
4 0x01 0x23 0x80 0x2F                     /* 238 02      */
0x32 //SEC_TAG_NO_ELEMENT
4 0x01 0x23 0x95 0x13,                    /* 239 513     */
0x31 //SEC_TAG_GROUP
16    //Length
0x32 //SEC_TAG_NO_ELEMENT
6 0x02 0x23 0x81 0x0F 0xF1 0x5F           /* 238 10-15   */
0x32 //SEC_TAG_NO_ELEMENT
6 0x02 0x23 0x95 0x67 0xF5 0x72           /* 239 567-572 */
```

10.2 Program NS Lock – ACTIVATE

```
/**
 *       Copyright (C) Danish Wireless Design A/S. All rights reserved.
 *
 * This document contains proprietary information belonging to Danish Wireless
 * Design A/S. Passing on and copying of this document, use and communication
 * of its contents is not permitted without prior written authorization.
 *
 * Description:
 *   NS Activation example.
 *
 * Revision Information:
 *   File name: NS_personalize_ACTIVE.psc
 *   Version: \main\1
 *   Date: 2003-02-18 11:05:21
 *   Responsible: knm
 *   Comment:
 *     -
 */

0x01 //SEC_TAG_PERSONALIZE_REQ
0x0F //SEC_TAG_PSC_FILE_VERSION
1    //Length
0x01 //Version
0x10 //SEC_TAG_LOCK_ID
1    //Length
0x01 //SEC_NS_LOCK
0x20 //SEC_TAG_LOCK_CHARACTERISTICS
1    //Length
0x07 //Lock Characteristics: Unlock allowed, CNL supported, DCK supported
0x25 //SEC_TAG_UNLOCK_CODE
8    //Length
0x31 0x32 0x33 0x34 0x35 0x36 0x37 0x38       //Unlock code
0x30 //SEC_TAG_CODE_GROUP
79   //Length
0x31 //SEC_TAG_GROUP
10   //Length
0x32 //SEC_TAG_NO_ELEMENT
4 0x01 0x23 0x80 0x1F                        /* 238 01    */
0x33 //SEC_TAG_NS_ELEMENT
2    //Length
0x01 0x30                                    /* 30        */
0x31 //SEC_TAG_GROUP
65   //Length
0x32 //SEC_TAG_NO_ELEMENT
4    //Length
0x01 0x23 0x85 0x0F                          /* 238 50    */
0x32 //SEC_TAG_NO_ELEMENT
6    //Length
0x02 0x23 0x81 0x0F 0xF1 0x5F                /* 238 10-15 */
0x32 //SEC_TAG_NO_ELEMENT
4    //Length
0x01 0x23 0x95 0x13                          /* 239 513   */
0x33 //SEC_TAG_NS_ELEMENT
2    //Length
0x01  0x12                                   /* 12        */
0x33 //SEC_TAG_NS_ELEMENT
4    //Length
0x02 0x13 0x48 0x61                          /* d1=3 and d4=8 and d6=1 */
0x33 //SEC_TAG_NS_ELEMENT
8    //Length
0x03 0x24 0x23 0x29 0x36 0x36 0x45 0x51      /* d2,d4= 35-41 or 54 or 69-81 */
0x33 //SEC_TAG_NS_ELEMENT
5    //Length
0x04 0x30 0x12 0x0A 0x0D                     /* d3=0 and d1,d2=10-13 */
0x33 //SEC_TAG_NS_ELEMENT
5    //Length
0x04 0x31 0x12 0x31 0x31                     /* d3=1 and d1,d2=49 */
0x33 //SEC_TAG_NS_ELEMENT
9    //Length
0x04 0x32 0x12 0x32 0x34 0x37 0x38 0x3B 0x3B  /* d3=2 and d1,d2=50,51,52,55,56,59 */
```

10.3 Program SP Lock – ACTIVE

```
/**
 *       Copyright (C) Danish Wireless Design A/S. All rights reserved.
 *
 * This document contains proprietary information belonging to Danish Wireless
 * Design A/S. Passing on and copying of this document, use and communication
 * of its contents is not permitted without prior written authorization.
 *
 * Description:
 *   SP Activation example.
 *
 * Revision Information:
 *   File name: SP_personalize_ACTIVE.psc
 *   Version: \main\1
 *   Date: 2003-02-18 09:38:43
 *   Responsible: knm
 *   Comment:
 *     -
 */

0x01 //SEC_TAG_PERSONALIZE_REQ
0x0F //SEC_TAG_PSC_FILE_VERSION
1    //Length
0x01 //Version
0x10 //SEC_TAG_LOCK_ID
1    //Length
0x02 //SEC_SP_LOCK
0x20 //SEC_TAG_LOCK_CHARACTERISTICS
1    //Length
0x07 //Lock Characteristics: Unlock allowed, CNL supported, DCK supported
0x25 //SEC_TAG_UNLOCK_CODE
8    //Length
0x31 0x32 0x33 0x34 0x35 0x36 0x37 0x38
0x30 //SEC_TAG_CODE_GROUP
65   //Length
0x31 //SEC_TAG_GROUP
22   //Length
0x32 //SEC_TAG_NO_ELEMENT
4 0x01 0x23 0x80 0x1F                        /* 238 01    */
0x32 //SEC_TAG_NO_ELEMENT
4 0x01 0x23 0x95 0x13                        /* 239 513   */
0x34 //SEC_TAG_SP_ELEMENT
8 0x01 0x00 0x05 0x48 0x65 0x6C 0x6C 0x6F    /* "Hello"   */
0x31 //SEC_TAG_GROUP
39   //Length
0x32 //SEC_TAG_NO_ELEMENT
4 0x01 0x23 0x85 0x0F                        /* 238 50    */
0x32 //SEC_TAG_NO_ELEMENT
6 0x02 0x23 0x81 0x0F 0xF1 0x5F              /* 238 10-15 */
0x32 //SEC_TAG_NO_ELEMENT
4 0x01 0x23 0x95 0x13                        /* 239 513   */
0x34 //SEC_TAG_SP_ELEMENT
8 0x01 0x00 0x05 0x48 0x61 0x6C 0x6C 0x69    /* "Halli"   */
0x34 //SEC_TAG_SP_ELEMENT
7 0x01 0x00 0x04 0x41 0x71 0x75 0x61         /* "Aqua"    */
```

10.4 Program CP Lock – ACTIVE

```
/**
 *       Copyright (C) Danish Wireless Design A/S. All rights reserved.
 *
 * This document contains proprietary information belonging to Danish Wireless
 * Design A/S. Passing on and copying of this document, use and communication
 * of its contents is not permitted without prior written authorization.
 *
 * Description:
 *   CP Activation example.
 *
 * Revision Information:
 *   File name: CP_personalize_ACTIVE.psc
 *   Version: \main\1
 *   Date: 2003-02-18 09:38:43
 *   Responsible: knm
 *   Comment:
 *     -
 */

0x01 //SEC_TAG_PERSONALIZE_REQ_TAG
0x0F //SEC_TAG_PSC_FILE_VERSION
1    //Length
0x01 //Version
```

```
0x10 //SEC_TAG_LOCK_ID_TAG
1    //Length
0x03 //SEC_CP_LOCK
0x20 //SEC_TAG_LOCK_CHARACTERISTICS
1    //Length
0x07 //Lock Characteristics: Unlock allowed, CNL supported, DCK supported
0x25 //SEC_TAG_UNLOCK_CODE
8    //Length
0x31 0x32 0x33 0x34 0x35 0x36 0x37 0x38
0x30 //SEC_TAG_CODE_GROUP
90   //Length
0x31 //SEC_TAG_GROUP
30   //Length
0x32 //SEC_TAG_NO_ELEMENT
4 0x01 0x23 0x95 0x13                       /* 239 513   */
0x32 //SEC_TAG_NO_ELEMENT
4 0x01 0x23 0x80 0x1F                       /* 238 01    */
0x34 //SEC_TAG_SP_ELEMENT
8 0x01 0x00 0x05 0x48 0x65 0x6C 0x6C 0x6F   /* "Hello"   */
0x35 //SEC_TAG_CP_ELEMENT
6 0x01 0x00 0x03 0x42 0x79 0x65             /* "Bye"     */
0x31 //SEC_TAG_GROUP
56   //Length
0x32 //SEC_TAG_NO_ELEMENT
4 0x01 0x23 0x85 0x0F                       /* 238 50    */
0x32 //SEC_TAG_NO_ELEMENT
4 0x01 0x23 0x95 0x13                       /* 239 513   */
0x32 //SEC_TAG_NO_ELEMENT
6 0x02 0x23 0x81 0x0F 0xF1 0x5F             /* 238 10-15 */
0x34 //SEC_TAG_SP_ELEMENT
8 0x01 0x00 0x05 0x48 0x61 0x6C 0x6C 0x69   /* "Halli"   */
0x34 //SEC_TAG_SP_ELEMENT
7 0x01 0x00 0x04 0x41 0x71 0x75 0x61        /* "Aqua"    */
0x35 //SEC_TAG_CP_ELEMENT
6 0x01 0x00 0x03 0x54 0x69 0x65             /* "Tie"     */
0x35 //SEC_TAG_CP_ELEMENT
7 0x01 0x00 0x03 0x54 0x65 0x65 0x6E        /* "Teen"    */
```

10.5 Program SM Lock – ACTIVE

```
/**
 *       Copyright (C) Danish Wireless Design A/S. All rights reserved.
 *
 * This document contains proprietary information belonging to Danish Wireless
 * Design A/S. Passing on and copying of this document, use and communication
 * of its contents is not permitted without prior written authorization.
 *
 * Description:
 *   SM Activation example.
 *
 * Revision Information:
 *   File name: SM_personalize_ACTIVE.psc
 *   Version: \main\1
 *   Date: 2003-02-18 09:38:43
 *   Responsible: knm
 *   Comment:
 *     -
 */

0x01 //SEC_TAG_PERSONALIZE_REQ
0x0F //SEC_TAG_PSC_FILE_VERSION
1    //Length
0x01 //Version
0x10 //SEC_TAG_LOCK_ID
1    //Length
0x04 //SEC_SIM_LOCK
0x20 //SEC_TAG_LOCK_CHARACTERISTICS
1    //Length
0x07 //Lock Characteristics: Unlock allowed, CNL supported, DCK supported
0x25 //SEC_TAG_UNLOCK_CODE
8    //Length
0x31 0x32 0x33 0x34 0x35 0x36 0x37 0x38
0x30 //SEC_TAG_CODE_GROUP
35   //Length
0x31 //SEC_TAG_GROUP
33   //Length
0x36 //SEC_TAG_SM_ELEMENT,
10 0x01 0x08 0x29 0x83 0x10 0x04 0x05 0x30 0x09 0x84  /* IMSI: 238014050039048 */
0x36 //SEC_TAG_SM_ELEMENT,
10 0x01 0x08 0x29 0x83 0x20 0x51 0x00 0x70 0x90 0x41  /* IMSI: 238021500070914 */
0x36 //SEC_TAG_SM_ELEMENT,
7 0x01 0x05 0x29 0x93 0x21 0x43 0x65                  /* IMSI: 239123456       */
```

10.6 General Setting Programming

```
/**
 *       Copyright (C) Danish Wireless Design A/S. All rights reserved.
 *
 * This document contains proprietary information belonging to Danish Wireless
 * Design A/S. Passing on and copying of this document, use and communication
 * of its contents is not permitted without prior written authorization.
 *
 * Description:
 *   General LOCK settings example including OVERALL CHARACTERISTICS object indicating
 *   common unlock key.
 *
 * Revision Information:
 *   File name: General_settings_03.psc
 *   Version: \main\1
 *   Date: 2003-02-18 11:05:21
 *   Responsible: knm
 *   Comment:
 *     -
 */

0x07 //SEC_TAG_GENERAL_SETTING_REQ
0x0F //SEC_TAG_PSC_FILE_VERSION
1    //Length
0x01 //Version
0x15 //SEC_TAG_OVERALL_CHARACTERISTICS
0x01 //Length
0x02 //Common Control Key
0x25 //SEC_TAG_UNLOCK_CODE
10   //Length
0x30 0x31 0x32 0x33 0x34 0x35 0x36 0x37 0x38 0x39 //Master Control Key
```

10.7 Master Control Key Verification

```
/**
 *       Copyright (C) Danish Wireless Design A/S. All rights reserved.
 *
 * This document contains proprietary information belonging to Danish Wireless
 * Design A/S. Passing on and copying of this document, use and communication
 * of its contents is not permitted without prior written authorization.
 *
 * Description:
 *   General LOCK settings example including OVERALL CHARACTERISTICS object indicating
 *   common unlock key.
 *
 * Revision Information:
 *   File name: Verify_master_key_OK.psc
 *   Version: \main\1
 *   Date: 2003-02-18 11:05:21
 *   Responsible: knm
 *   Comment:
 *     -
 */

0x08 //SEC_TAG_VERIFY_MASTER_KEY_REQ
0x0F //SEC_TAG_PSC_FILE_VERSION
1    //Length
0x01 //Version
0x25 //SEC_TAG_UNLOCK_CODE
10   //Length
0x30 0x31 0x32 0x33 0x34 0x35 0x36 0x37 0x38 0x39 //Control Key
```

10.8 Interrogate Lock Profiles

```
/**
 *       Copyright (C) Danish Wireless Design A/S. All rights reserved.
 *
 * This document contains proprietary information belonging to Danish Wireless
 * Design A/S. Passing on and copying of this document, use and communication
 * of its contents is not permitted without prior written authorization.
 *
 * Description:
 *   General LOCK settings example including OVERALL CHARACTERISTICS object indicating
 *   common unlock key.
 *
 * Revision Information:
 *   File name: Get_lock_profiles.psc
 *   Version: \main\1
 *   Date: 2003-02-18 11:05:21
 *   Responsible: knm
```

```
 *   Comment:
 *     -
 */

0x02 //SEC_TAG_GET_LOCK_PROFILES_REQ
```

10.9 Program NO lock – ARMED

```
/**
 *       Copyright (C) Danish Wireless Design A/S. All rights reserved.
 *
 * This document contains proprietary information belonging to Danish Wireless
 * Design A/S. Passing on and copying of this document, use and communication
 * of its contents is not permitted without prior written authorization.
 *
 * Description:
 *   NS Arm example.
 *
 * Revision Information:
 *   File name: NO_personalize_ACTIVE.psc
 *   Version: \main\1
 *   Date: 2003-02-18 09:38:43
 *   Responsible: knm
 *   Comment:
 *     -
 */

0x01 //SEC_TAG_PERSONALIZE_REQ
0x0F //SEC_TAG_PSC_FILE_VERSION
1    //Length
0x01 //Version
0x10 //SEC_TAG_LOCK_ID
1    //Length
0x01 //SEC_NS_LOCK
0x20 //SEC_TAG_LOCK_CHARACTERISTICS
1    //Length
0x07 //Lock Characteristics: Unlock allowed, CNL supported, DCK supported
0x25 //SEC_TAG_UNLOCK_CODE
8    //Length
0x31 0x32 0x33 0x34 0x35 0x36 0x37 0x38
0x26 //SEC_TAG_ARM_CHARACTERISTICS
0x02 //Length
0x01 //QUALIFIER
0x00 //2 digit MNC to be assumed when meeting the first SIM (3 digit MNC is 0x01).
```

10.10 Program NO lock – DISABLED

```
/**
 *       Copyright (C) Danish Wireless Design A/S. All rights reserved.
 *
 * This document contains proprietary information belonging to Danish Wireless
 * Design A/S. Passing on and copying of this document, use and communication
 * of its contents is not permitted without prior written authorization.
 *
 * Description:
 *   NO Disable example.
 *
 * Revision Information:
 *   File name: NO_personalize_ACTIVE.psc
 *   Version: \main\1
 *   Date: 2003-02-18 09:38:43
 *   Responsible: knm
 *   Comment:
 *     -
 */

0x01 //SEC_TAG_PERSONALIZE_REQ
0x0F //SEC_TAG_PSC_FILE_VERSION
1    //Length
0x01 //Version
0x10 //SEC_TAG_LOCK_ID
1    //Length
0x00 //SEC_NO_LOCK
0x27 //SEC_TAG_DISABLE
0x00 //Length
```