GSM PROTOCOL STACK

SOFTWARE LOW LEVEL DESIGN DOCUMENT

SIM PERSONALIZATION FEATURE

Document Version	**Author(s)**	**Approval(s)**
0.1 (Initial Draft) merging VLE5-SMLK and new LoCosto features	Sasken	
1.0 (re-formatting to suit TI-LoCosto design template)	Sasken	Manish
2.0 (Restoring 0.1 content with cover page, header, footer as per TI-LoCosto design template)	Sasken	

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

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

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

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third–party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Mailing Address:
Texas Instruments
Post Office Box 655303
Dallas, Texas 75265

TABLE OF CONTENTS

1 Introduction

This document presents the Low-level Design for ME Personalization (SIM Lock) functionality.

1.1 Scope

The scope of this document is limited to the ME Personalization functionality implemented at the ACI layer. The document also presents the data structure format for the personalization data to be stored in the Security Driver.

The document does not discuss details about the MMI design for the Personalzition functionality. MMI design shall be presented in a separate document.

1.2 Design Representation

Flow-charts and pseudo-code have been used to explain the detailed design of each functionality. Data structures have been presented using standard `C` convention.

1.3 Terms/Abbreviations

Sr. No.	Term	Expansion
1.	ACI	Application Control Interface
2.	API	Application Programming Interface
3.	ATCI	AT Command Interpreter
4.	CK	Control Key
5.	IMSI	International Mobile Subscriber Identity
6.	ME	Mobile Equipment
7.	SIM	Subscriber Identity Module

1.4 References:

Sr. No	Document/Filename	Version
1.	3GPP TS 22.022 Personalization of ME	V3.2.1 (2002-06)
2.	3GPP TS 27.007 AT Command set for User Equipment	V 3.13.0 (2003-03)
3.	3GPP TS 11.11 Mobile Equipment (SIM-ME) Interface	V 8.9.1 (2003-06)
4.	ME Personalization Requirements - SIM-ME-LOCK VLE5 SIM-ME-LOCK on VLE5 Handsets	0.2
5.	ME Personalization Requirements - SIM-ME-LOCK SIM-ME-LOCK on Alcatel Handsets	0.1
6.	TA_SIM_Lock_TR_Frozen.xls	
7.	ACI-ME Personalization – Interface Description	Draft
8.	Security Driver – High level design	0.2
9.	ME Personalization – High level design	0.1

2 Global variables

1) T_SEC_LCK_CFG aci_slock_config

2) T_ACI_SIM_CONFIG aci_slock_sim_config; /* SIM configuration*/

```
typedef struct
{
   UBYTE deper_key [16];
   UBYTE phase;
   UBYTE oper_mode;        /* SIM card functionality    */
   UBYTE pref_lang[5];
   UBYTE access_acm;
   UBYTE access_acmmax;
   UBYTE access_puct;
  UBYTE sim_gidl1[MAX_GID];
  UBYTE sim_gidl2[MAX_GID];

  T_SIM_TYPE sim_type;               -- newly introduced to identify the SIM Type
   UBYTE sim_read_gid1;  /* added for SP, CP */
   UBYTE sim_read_gid2;  /* added for SP, CP */
   UBYTE gid1_len;
   UBYTE gid2_len;
   UBYTE sim_read_ad_first_byte;
} T_ACI_SIM_CONFIG;
```

where T_SIM_TYPE is defined as

```
typedef enum
{
        SIM_NORMAL,
        SIM_TYPEAPPROVAL,
        SIM_TEST_CPHS
}T_SIM_TYPE ;
```

3) T_SIM_SHRD_PRM simShrdPrm; - Currently exists in the code

4) T_ACI_SLOCK_SHARED AciSLockShrd

```
typedef struct
{
   T_SIMLOCK_STATUS status[SIMLOCK_FIRST_SIM+1];  /* one status for every
                                                       personalisation lock type */
   UINT16 dependency[SIMLOCK_FIRST_SIM+1];
   UBYTE blocked;                /* blocked flag */
   T_SIMLOCK_TYPE current_lock;            /* currently checked lock */
   UBYTE auto_personalisation_done ;
   UBYTE pb_load ;
 }T_ACI_SLOCK_SHARED;
```

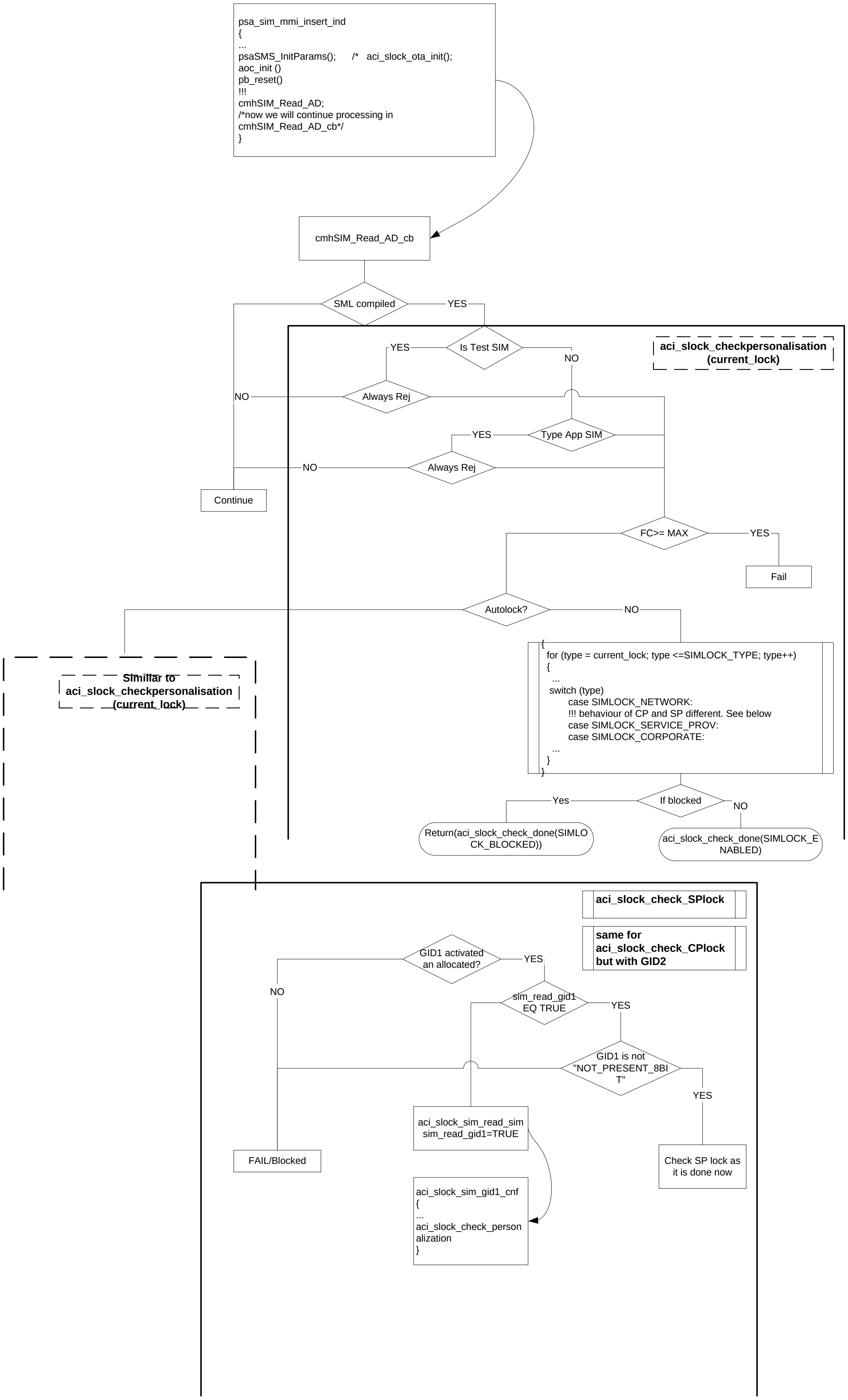
3 Functionality

3.1 Personalization Check – Power On

The personalization check is done when a SIM_MMI_INSERT_IND is received. The psa_sim_mmi_insert_ind receives this indication.

Firstly ACI sends a request to read the1st byte of EF(AD)=0x80 to identify if it is a type Approval SIM. Then it is checked if the SIM is a test sim. ACI checks if the PLMN value is 00101. If yes, then the personalization check proceeds based on theTestSIM Mode configured in MEPD. If not, then the personalizations check proceeds based on normal SIM. If it is a normal SIM, the aci_slock_check_personalization function is called. The personalization code in the SIM is compared with the normal codes stored in the MEPD. If none of the normal codes match, the personalization code in the SIM is verified using the Interval data stored in MEPD. If personalization check succeeds, ME start with normal operation. If it fails, the user is asked to enter the control keys for the failed categories.

The following flowcharts explain the personalization check process. If SP or CP category is set in MEPD, while doing check personalization first check will be done for whether GID1/GID2 is read from the sim. If it is not read from the sim, sim read request is called before check personalization of SP/CP category. If check done for all the category is successful the aci_slcok_check_done (SIMLOCK_ENABLED) function is called. If it is unsuccessful the aci_slcok_check_done (SIMLOCK_BLOCKED) function is called.

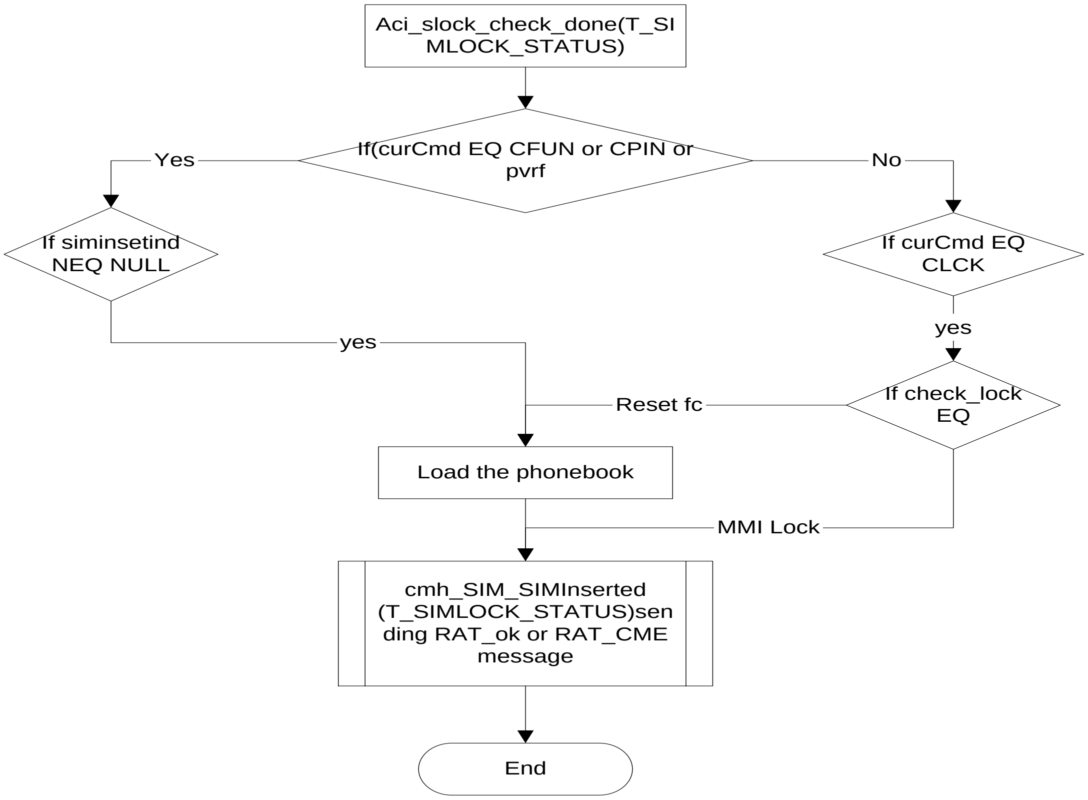


```
psa_sim_mmi_insert_ind
{
...
psaSMS_InitParams();     /*   aci_slock_ota_init();
aoc_init ()
pb_reset()
!!!
cmhSIM_Read_AD;
/*now we will continue processing in
cmhSIM_Read_AD_cb*/
}
```

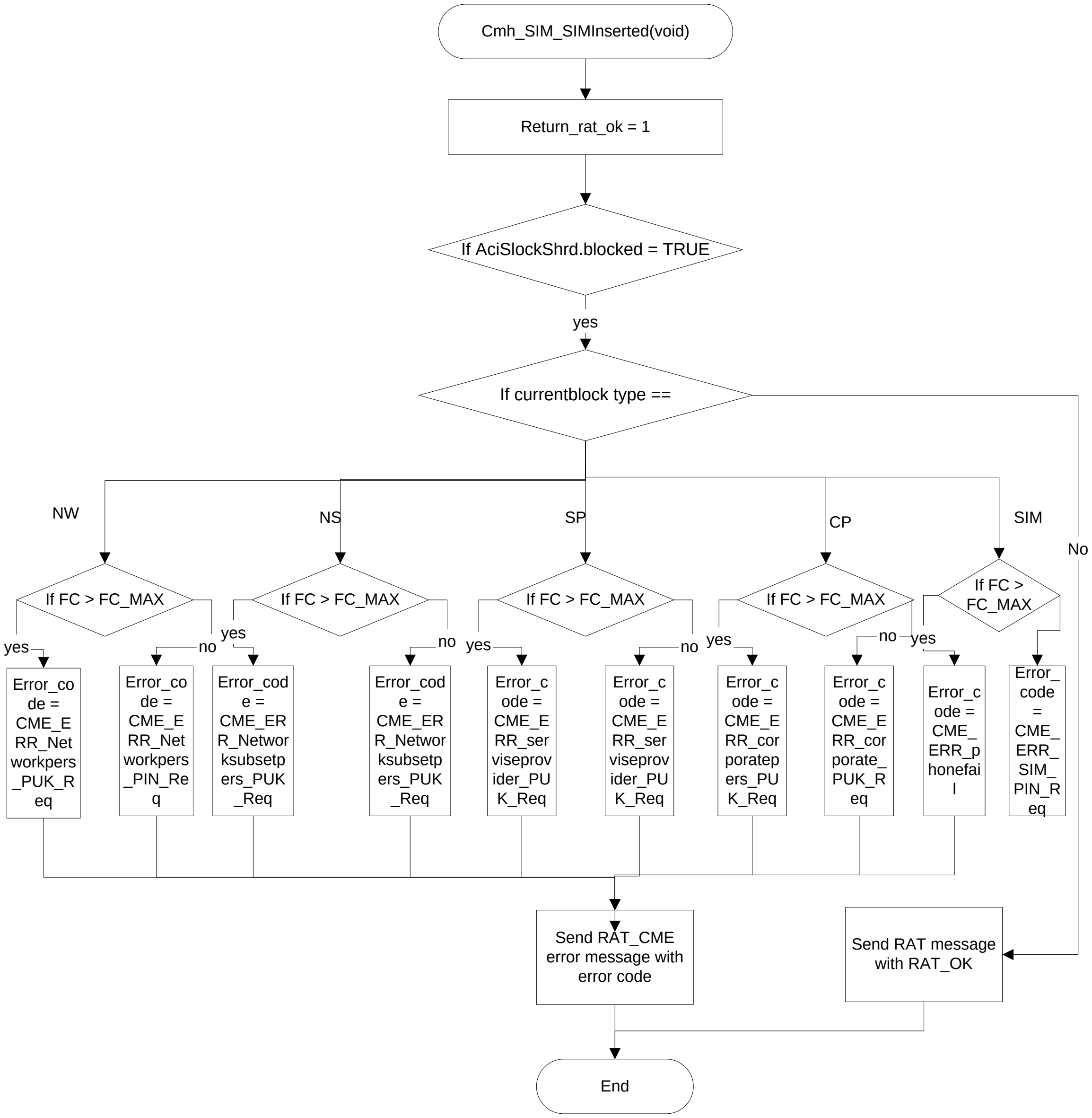
```
{
  for (type = current_lock; type <=SIMLOCK_TYPE; type++)
  {
    ...
   switch (type)
         case SIMLOCK_NETWORK:
         !!! behaviour of CP and SP different. See below
         case SIMLOCK_SERVICE_PROV:
         case SIMLOCK_CORPORATE:
    ...
  }
}
```

```
aci_slock_sim_gid1_cnf
{
...
aci_slock_check_person
alization
}
```

cmhSIM_Read_AD_cb
SML compiled
YES
NO
aci_slock_checkpersonalisation (current_lock)
YES
Is Test SIM
NO
Always Rej
NO
YES
Type App SIM
NO
Always Rej
Continue
FC>= MAX
YES
Fail
Autolock?
NO
Similar to aci_slock_checkpersonalisation (current_lock)
Yes
If blocked
NO
Return(aci_slock_check_done(SIMLOCK_BLOCKED))
aci_slock_check_done(SIMLOCK_ENABLED)
aci_slock_check_SPlock
same for aci_slock_check_CPlock but with GID2
GID1 activated an allocated?
YES
NO
sim_read_gid1 EQ TRUE
YES
GID1 is not "NOT_PRESENT_8BIT"
YES
aci_slock_sim_read_sim sim_read_gid1=TRUE
FAIL/Blocked
Check SP lock as it is done now


Aci_slock_check_done(T_SIMLOCK_STATUS)
If(curCmd EQ CFUN or CPIN or pvrf
Yes
No
If siminsetind NEQ NULL
If curCmd EQ CLCK
yes
yes
Reset fc
If check_lock EQ
Load the phonebook
MMI Lock
cmh_SIM_SIMInserted (T_SIMLOCK_STATUS)sending RAT_ok or RAT_CME message
End

The command handler function cmh_SIM_SIMInserted is called from aci_slock_check_done function after the personalization check is done. This handler function returns a Pin Request message to the MMI if the personalization check fails. (RAT_CME in the flow-chart is an error response to the AT Command in case of a failure. Post this response, the user is requested for the Control key in case of failure.)


Cmh_SIM_SIMInserted(void)
Return_rat_ok = 1
If AciSlockShrd.blocked = TRUE
yes
If currentblock type ==
NW
NS
SP
CP
SIM
No
If FC > FC_MAX
If FC > FC_MAX
If FC > FC_MAX
If FC > FC_MAX
If FC > FC_MAX
yes
no
yes
no
yes
no
yes
no
yes
Error_code = CME_ERR_Networkpers_PUK_Req
Error_code = CME_ERR_Networkpers_PIN_Req
Error_code = CME_ERR_Networksubsetpers_PUK_Req
Error_code = CME_ERR_Networksubsetpers_PUK_Req
Error_code = CME_ERR_serviseprovider_PUK_Req
Error_code = CME_ERR_serviseprovider_PUK_Req
Error_code = CME_ERR_corporatepers_PUK_Req
Error_code = CME_ERR_corporate_PUK_Req
Error_code = CME_ERR_phonefail
Error_code = CME_ERR_SIM_PIN_Req
Send RAT_CME error message with error code
Send RAT message with RAT_OK
End

3.2 Auto-personalization

The ME personalizes to the first Normal SIM inserted if Autolock is enabled for any of the categories.

When a SIM is inserted for the first time into the ME:
ACI checks if it is a Test SIM.
If it is a Test SIM, ME is not auto-personalized.
If it is a Normal SIM, ACI gets the auto-lock record from Security Driver.
If the status of auto-lock record is DISABLED, then ME is not personalized.
If the status of the auto-lock record is ENABLED, ACI checks if the personalization code for the auto-lock categories already exists in MEPD(by default).

If code does not exist, it is added to the respective code group in the MEPD using the **sec_get_rec** & **sec_set_rec** APIs.
If the code already exists in the MEPD by default, no new code is added.

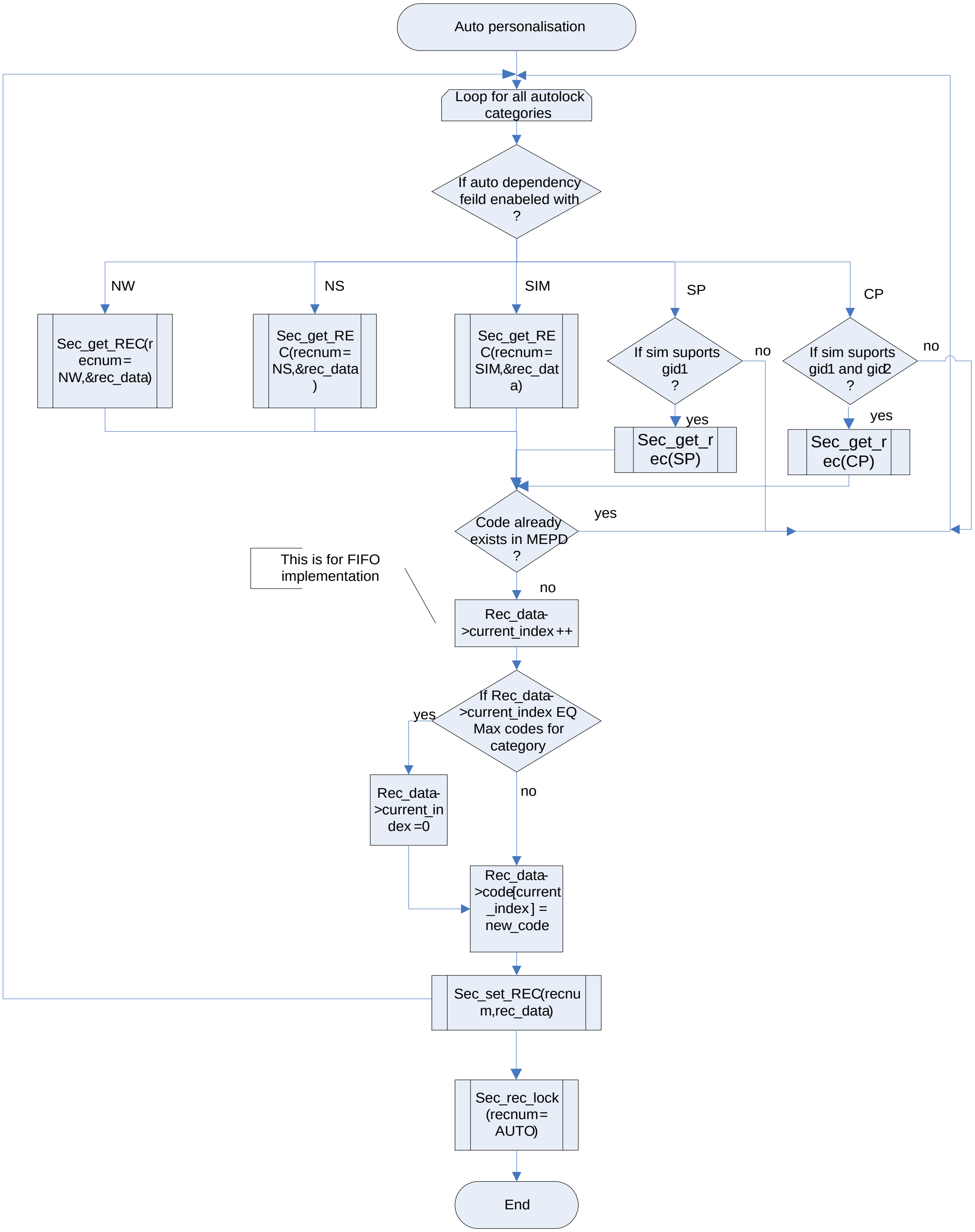
The security driver API **sec_rec_autolock**(record num) is then called to lock the autolock category(s). **sec_rec_autolock is a new API that needs to be added to the Security Driver. It should internally set the status of the category to LOCKED without any checks.**

After the first auto-personalization is done, the status of the AUTOLOCK record status is set to DISABLED to avoid any subsequent auto-personalization.

When a SIM is already auto-personalized and a new SIM is inserted:
No autopersonalization is done since the AUTOLOCK record status would be disabled.

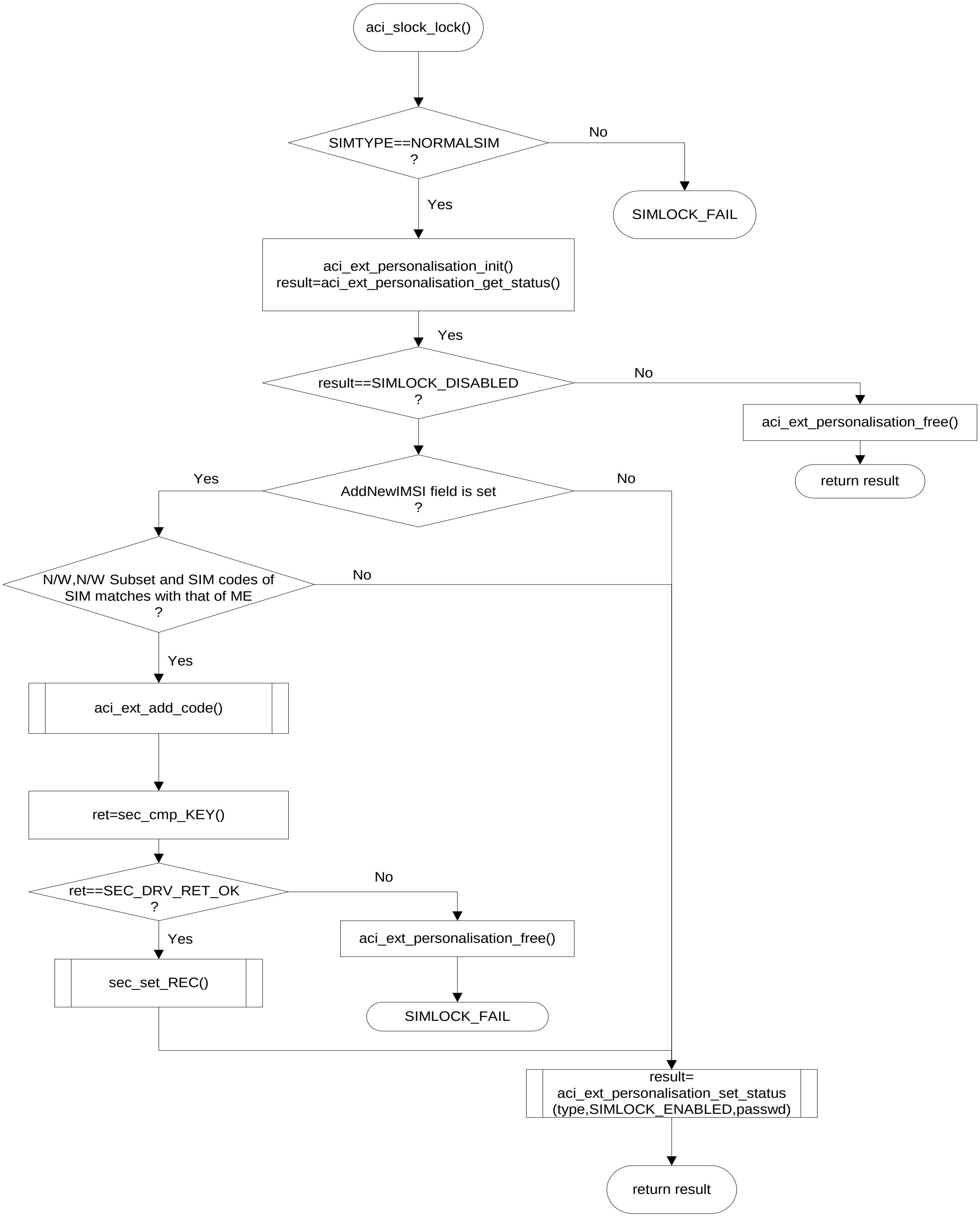
When the same SIM is inserted again:
Here again, no autopersonalization is done since the AUTOLOCK record status would be disabled.

Flow-charts for auto-personalization are as follows. A separate flow-chart is provided for auto-personalization for SP and CP categories. This is because auto-personalization for these categories is initiated only when a read confirmation for GID1 and GID2 files is received.


Auto personalisation
Loop for all autolock categories
If auto dependency feild enabeled with ?
NW
NS
SIM
SP
CP
Sec_get_REC(recnum= NW,&rec_data)
Sec_get_REC(recnum= NS,&rec_data)
Sec_get_REC(recnum= SIM,&rec_data)
If sim suports gid1 ?
no
yes
Sec_get_rec(SP)
If sim suports gid1 and gid2 ?
no
yes
Sec_get_rec(CP)
Code already exists in MEPD ?
yes
no
This is for FIFO implementation
Rec_data->current_index++
If Rec_data->current_index EQ Max codes for category
yes
no
Rec_data->current_index=0
Rec_data->code[current_index] = new_code
Sec_set_REC(recnum,rec_data)
Sec_rec_lock (recnum= AUTO)
End

3.3 Personalization Process

This functionality is implemented for the SIM Personalization option provided from the MMI.


aci_slock_lock()
SIMTYPE==NORMALSIM ?
No
SIMLOCK_FAIL
Yes
aci_ext_personalisation_init()
result=aci_ext_personalisation_get_status()
Yes
result==SIMLOCK_DISABLED ?
No
aci_ext_personalisation_free()
return result
AddNewIMSI field is set ?
Yes
No
N/W,N/W Subset and SIM codes of SIM matches with that of ME ?
No
Yes
aci_ext_add_code()
ret=sec_cmp_KEY()
ret==SEC_DRV_RET_OK ?
No
Yes
aci_ext_personalisation_free()
sec_set_REC()
SIMLOCK_FAIL
result= aci_ext_personalisation_set_status (type,SIMLOCK_ENABLED,passwd)
return result

3.4 Locking & Unlocking

This functionality is provided to lock or unlock a personalization category from the UI or via AT commands issued to the ME. The following steps are performed for Locking/Unlocking of the category from the MMI.

- Select the category to be locked/unlocked from the MMI.
- Enter the password for the category
- sAT_PlusCLCK function is called with the category type and password.
- sAT_PlusCLCK calls the aci_slock_lock/aci_slock_unlock function for locking & unlocking respectively.

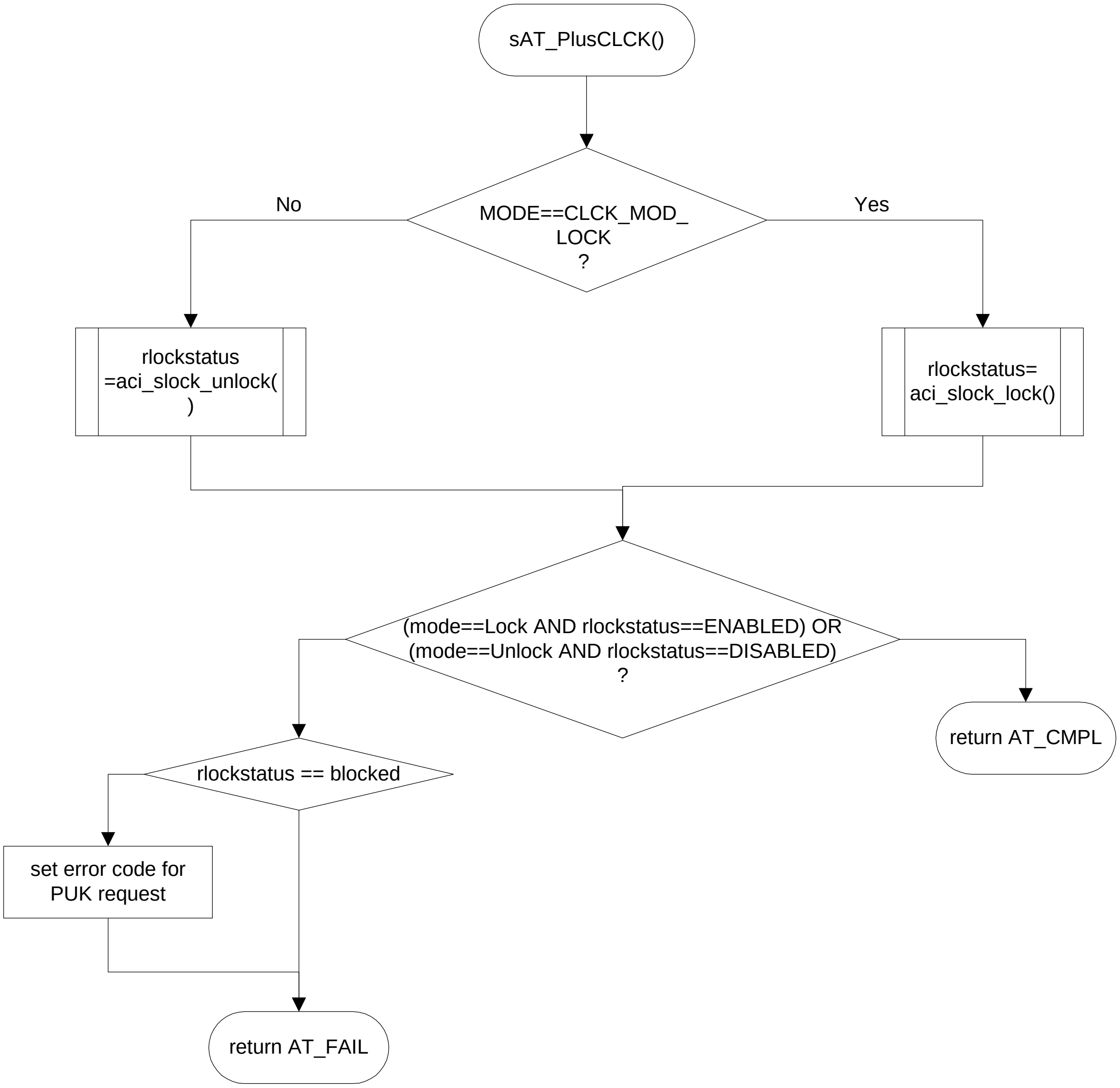
For Lock operation:
ACI checks if the category code already exists in the MEPD. If yes, then ACI calls aci_slock_lock with the category and password.
If the category code does not exist in the MEPD, ACI checks the 'Add New Imsi' flag to check if a new code can be added to MEPD.
If the Add New Imsi flag is on, ACI adds the code to MEPD using sec_get_rec & sec_set_rec APIs.

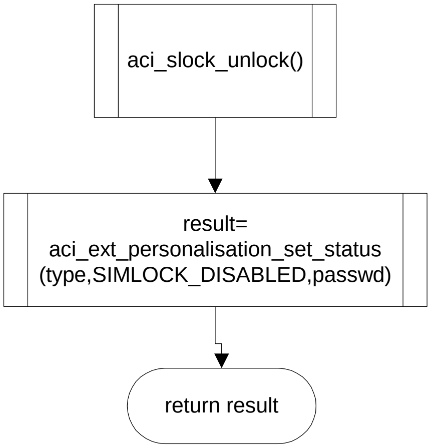
If the Add New Imsi flag is off, ACI just sets the category status to locked.

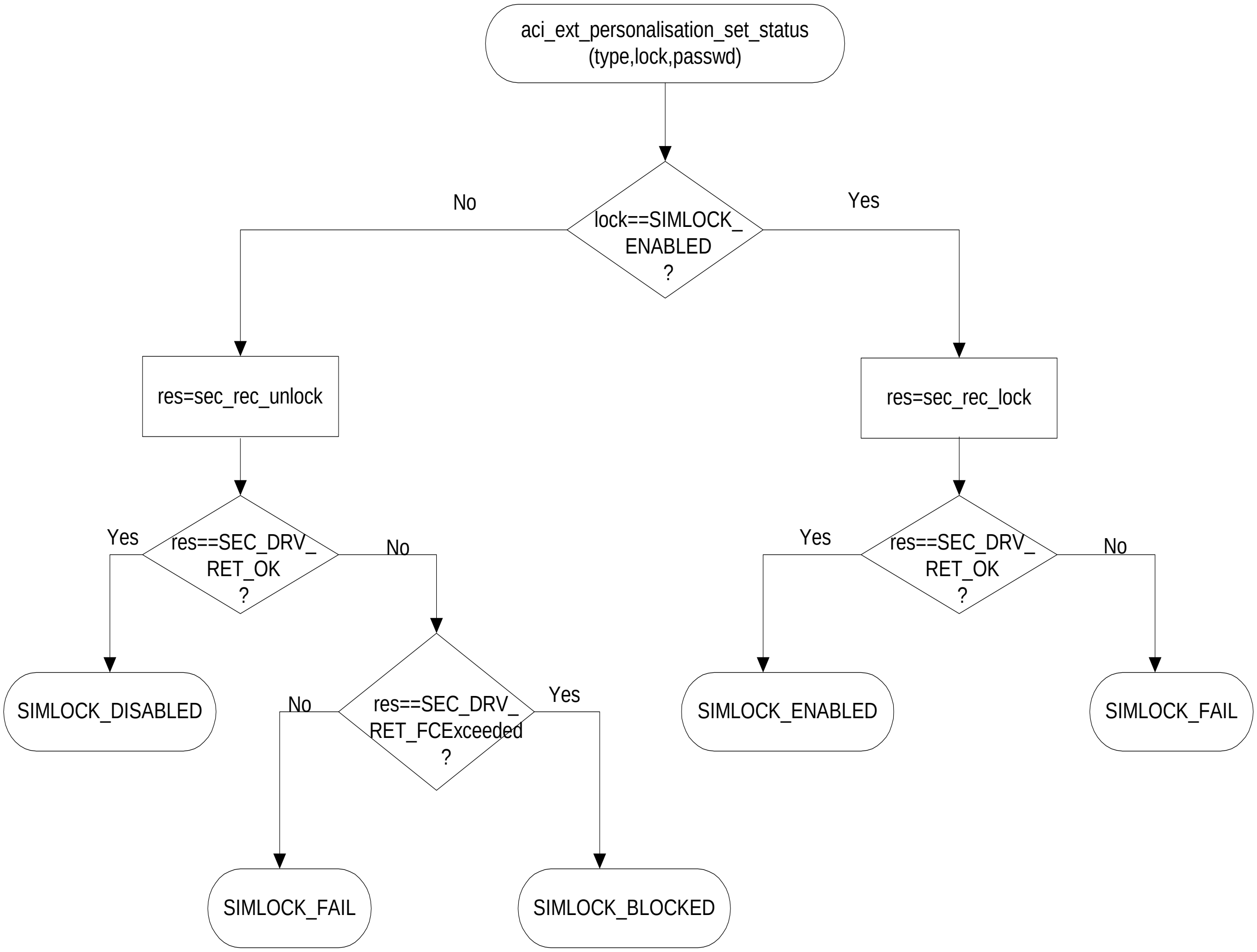
If the password entered by the user is Incorrect, the user is informed of failure.
If the category is already locked, the user is notified.

For Unlock operation:
ACI calls the sec_rec_unlock function with the category and password. If the password is incorrect, Security Driver increments the failure counter. User is notified of the incorrect password.

The user is given FC_MAX number of chances to enter the right password.(as configured in the security driver.). If the user doesn't enter the right password in within the give chances, user is notified that the ME is blocked.


sAT_PlusCLCK()
No
MODE==CLCK_MOD_ LOCK ?
Yes
rlockstatus =aci_slock_unlock( )
rlockstatus= aci_slock_lock()
(mode==Lock AND rlockstatus==ENABLED) OR (mode==Unlock AND rlockstatus==DISABLED) ?
return AT_CMPL
rlockstatus == blocked
set error code for PUK request
return AT_FAIL


aci_slock_unlock()
result= aci_ext_personalisation_set_status (type,SIMLOCK_DISABLED,passwd)
return result


aci_ext_personalisation_set_status
(type,lock,passwd)
lock==SIMLOCK_
ENABLED
?
No
Yes
res=sec_rec_unlock
res=sec_rec_lock
res==SEC_DRV_
RET_OK
?
Yes
No
SIMLOCK_DISABLED
res==SEC_DRV_
RET_FCExceeded
?
No
Yes
SIMLOCK_FAIL
SIMLOCK_BLOCKED
res==SEC_DRV_
RET_OK
?
Yes
No
SIMLOCK_ENABLED
SIMLOCK_FAIL

3.5 Disable Personalization (Permanent Unlocking)

The sAT_PercentCLCK API shall be provided for permanent unlocking of a category. (Currently, this option shall not be available from the MMI as it is not a part of the current requirements.) ACI shall provide an additional API aci_slock_permanent_unlock.

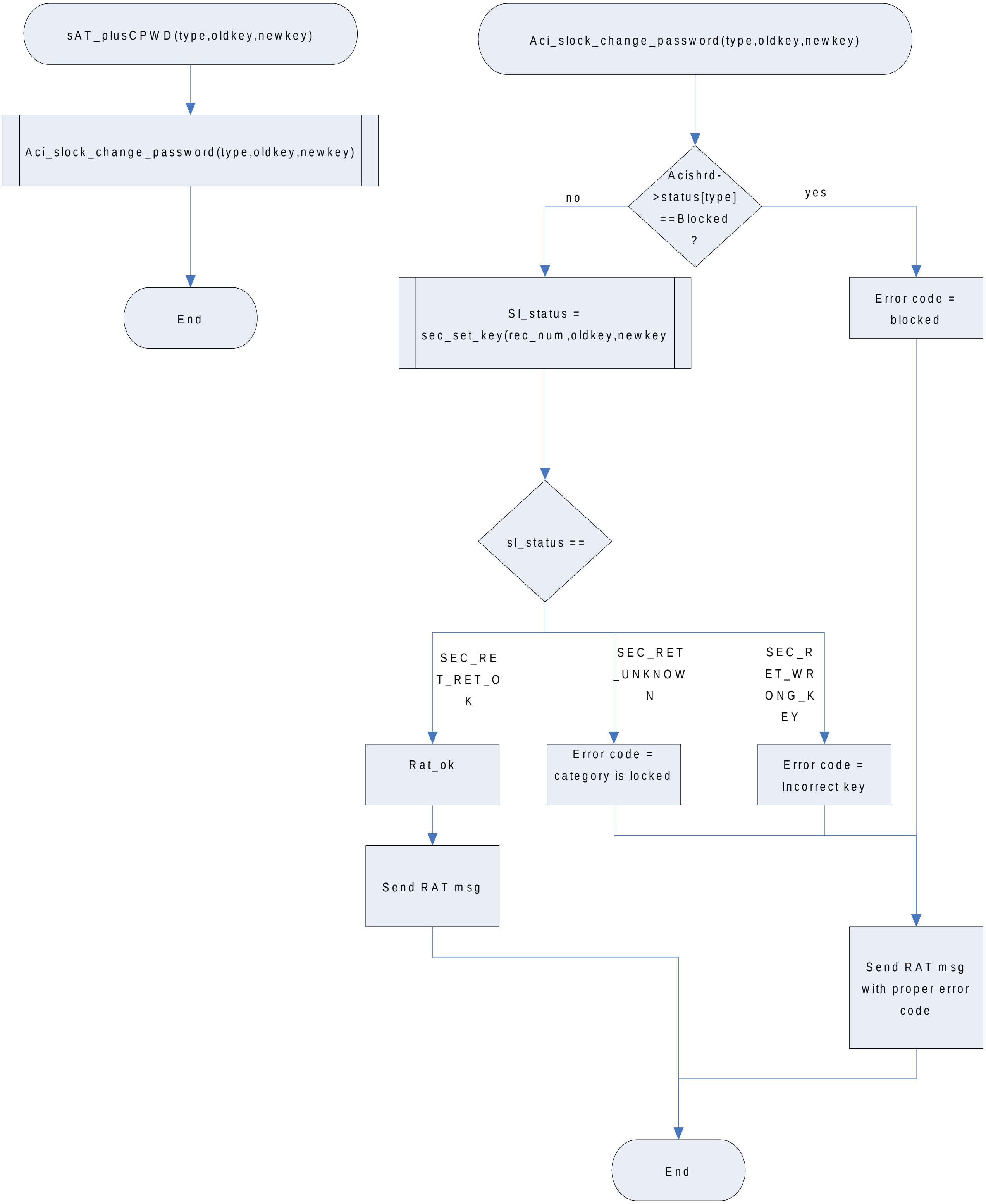
3.6 Password Change

MMI provides options to change passwords for a particular category. The following steps are performed for password change from the MMI:

a. Select the 'password change' option from the MMI.
b. Enter the current password and new password.
c. If the current password is incorrect, the user is prompted to enter the current password again.
d. If the current password is correct, the new password is updated in the Security Driver. The user is informed of a successful password change.

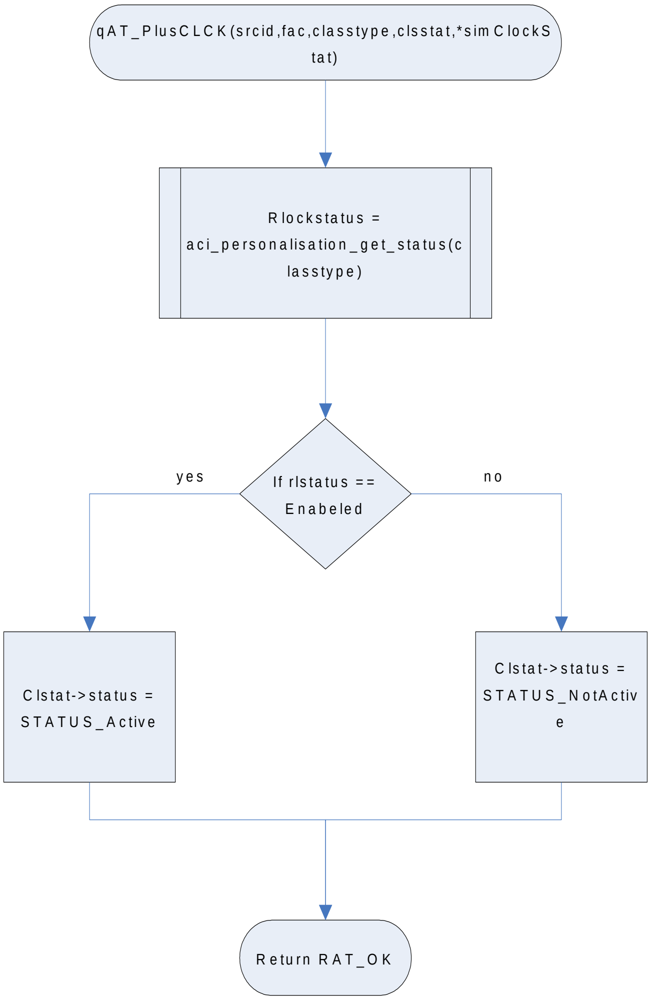
For AT commands issued over Modem SW/AT interface, the sAT_PlusCPWD function is invoked with the current and new password. The rest of the operations are similar to that in password change from MMI.

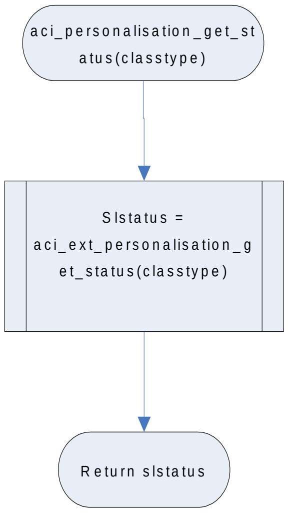
(Flowchart on next page)

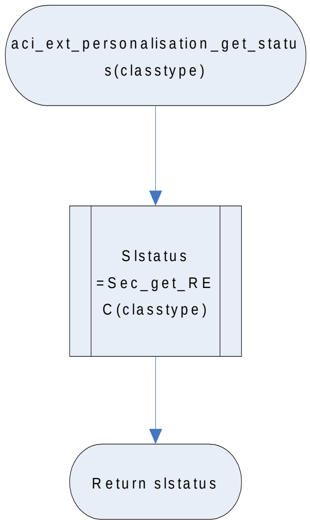

sAT_plusCPWD(type,oldkey,newkey)
Aci_slock_change_password(type,oldkey,newkey)
End
Aci_slock_change_password(type,oldkey,newkey)
Acishrd->status[type] ==Blocked ?
no
yes
Sl_status = sec_set_key(rec_num,oldkey,newkey
Error code = blocked
sl_status ==
SEC_RET_RET_OK
SEC_RET_UNKNOWN
SEC_RET_WRONG_KEY
Rat_ok
Error code = category is locked
Error code = Incorrect key
Send RAT msg
Send RAT msg with proper error code
End

3.7 Status Check

This functionality is provided to check the lock/unlock status of a particular category from the MMI or via AT Commands. The qAT_PlusCLCK function receives the AT command. This function calls the ACI API aci_personalization_get_status which retrieves the status from the Security Driver. The status is returned to the MMI or to the AT Command interface.


qAT_PlusCLCK(srcid,fac,classtype,clsstat,*simClockStat)
Rlockstatus = aci_personalisation_get_status(classtype)
If rlstatus == Enabeled
yes
no
Clstat->status = STATUS_Active
Clstat->status = STATUS_NotActive
Return RAT_OK


aci_personalisation_get_status(classtype)
Slstatus = aci_ext_personalisation_get_status(classtype)
Return slstatus


aci_ext_personalisation_get_status(classtype)
Slstatus =Sec_get_REC(classtype)
Return slstatus

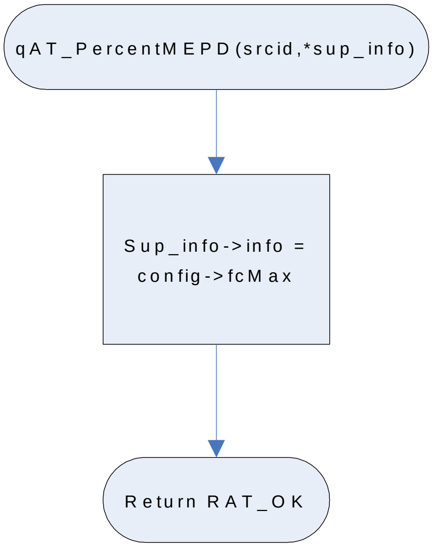
3.8 Failure Counter Reset

The MMI provides an option to reset the failure counter.. The AT+CLCK command shall be used for this functionality. A new <fac> value FC shall be introduced.

e. Select the 'Reset Failure Counter' option from the MMI.
f. Enter the failure counter reset password.
g. sAT_PlusCLCK(FC,0,PWD) is invoked. This function calls the **aci_slock_reset_fc** API provided by ACI-SLOCK.
h. aci_slock_reset_fc invokes the aci_ext_slock_reset_fc function.
i. aci_ext_reset_fc calls the Security Driver API sec_FC_Reset with the user-entered password and the password length.
j. If Security Driver returns failure, the user is informed about the reset failure due to incorrect password.

3.9 Supplementary Info

This facility is provided to inform the maximum number of attempts allowed for entering the Control key (FCMAX), current attempts left to enter control key (FCATTEMPTSLEFT), maximum number of reset failure counter allowed (FCRESETFAILMAX), current reset failure counter attempts left (FCRESETFAILATTEMPTSLEFT), maximum number of successful attempts allowed to reset failure counter(FCRESETSUCCESSMAX), current successful attempts left for reset failure counter(FCRESETSUCCESSATTEMPTSLEFT), Timer flag , ETSI Flag and Airtel indication flag . The qAT_PercentMEPD function is implemented for the same. This function reads the global config data to get the Max failure counter value.
This value is returned via the pointer variable passed to the qAT_PercentMEPD function.


qAT_PercentMEPD(srcid,*sup_info)
Sup_info->info = config->fcMax
Return RAT_OK

3.10 Over The Air De-personalisation

As an optional ME feature, the ME may be de-personalised over-the-air (OTA) by the network. The network (mcc+mnc in imsi), network subset (hlr code in imsi), Service Provider (gid1) and corporate (gid1 + gid2) categories may be de-personalised in this way. More than one category may be de-personalised at the same time. The process results in the relevant personalisation indicator(s) being set to "off". The ME must be registered on a network.
Two OTA methods are defined both of which use MT SMS-PP messages. With the first method, the IMEI of the ME to be de-personalised and the Control Key(s) of the personalisation categories to be

de-personalised are sent directly to the ME. The ME performs checks on both the IMEI and the key values and the outcome of the attempted de-personalisation(s) is acknowledged to the network.
With the second method, the keys of the personalisation categories to be de-personalised are sent to the ME via the SIM/USIM. The IMEI is not included and the de-personalisation process only checks the keys. The outcome of the attempted de-personalisation(s) is acknowledged to the network.
The network de-personalises the ME by one of the following methods:

3.10.1 SMS-PP, ME-specific:

a) A point-to-point SMS message is sent by the network to the MS , the message being marked as being destined for the ME only and for the purposes of ME de-personalisation. The message will have (in SMS PDU)

1. Data Coding Scheme – Uncompressed, Default Alphabet, Message Class1 (ME specific) i.e. DCS = 0x11
2. Protocol Identifier – PID = 0x7E
3. User Data – De-personalisation keys and IMEI as below (default alphabet coding). 'FFFFFFF' to denote 'de-personalisation not required

Character	Description
1 – 40	Operator specific text padded with spaces to character 40.
41 – 48	Network control key
49 – 56	Network subset control key
57 – 64	SP control key
65 – 72	Corporate control key
73 – 88	IMEI

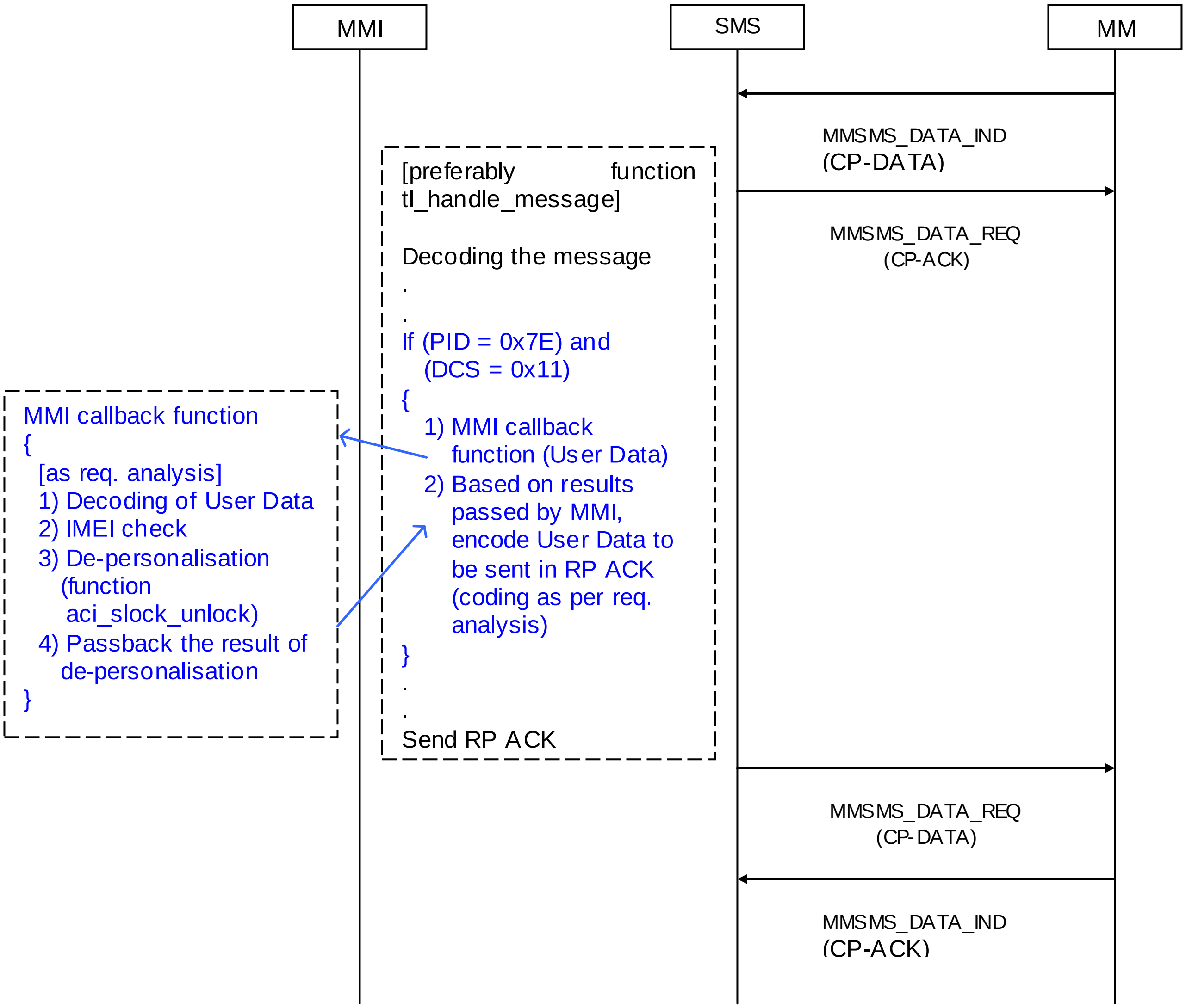
b) ME compares the IMEI value and then category key values

c) ME sends acknowledgement to network; User Data in SMS-DELIVER-REPORT for RP-ACK is as follows:

Character	Description
1-16	IMEI of ME
17	Network personalisation status
18	Network subset personalisation status
19	SP personalisation status
20	Corporate personalisation status

Status code	Description
0	Currently not personalised
1	Permanently not personalised
2	Personalised
3	IMEI mismatch
Other	RFU

1. If IMEI values differ, no effect on personalisation status and return status code = '3' for all the categories
2. If any key values differ, no effect on corresponding personalisation status and return result code = '0/1/2'
3. If key value match, de-personalisation for corresponding category and return result code = '0'


MMI
SMS
MM
MMSMS_DATA_IND
(CP-DATA)
MMSMS_DATA_REQ
(CP-ACK)
[preferably function tl_handle_message]
Decoding the message
.
.
If (PID = 0x7E) and
(DCS = 0x11)
{
1) MMI callback function (User Data)
2) Based on results passed by MMI, encode User Data to be sent in RP ACK (coding as per req. analysis)
}
.
.
Send RP ACK
MMI callback function
{
[as req. analysis]
1) Decoding of User Data
2) IMEI check
3) De-personalisation (function aci_slock_unlock)
4) Passback the result of de-personalisation
}
MMSMS_DATA_REQ
(CP-DATA)
MMSMS_DATA_IND
(CP-ACK)

3.10.2 SMS-PP Data Download:

a) The network sends a PP SMS message to the SIM for updating EFDCK (via Data download procedure of SIM Tool Kit). As a result, SIM issues proactive command REFRESH.

Identifier: '6F2C'	Structure: transparent	Optional
File size: 16 bytes	Update activity: low	

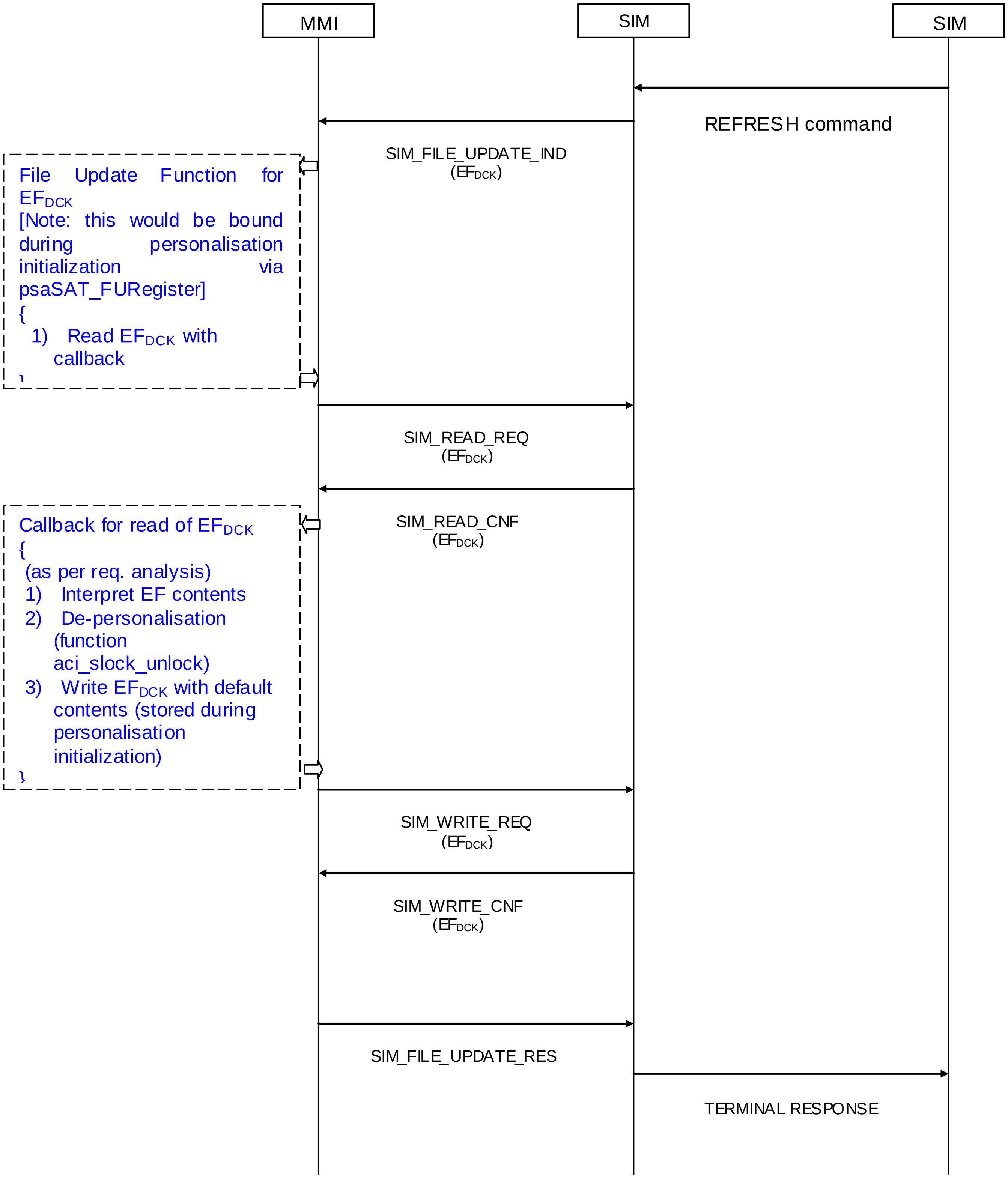
Access Conditions:

READ	CHV1
UPDATE	CHV1
INVALIDATE	ADM
REHABILITATE	ADM

Bytes	Description	M/O	Length
1 to 4	8 digits of network de-personalization control key	M	4 bytes
5 to 8	8 digits of network subset de-personalization control key	M	4 bytes
9 to 12	8 digits of service provider de-personalization control key	M	4 bytes
13 to 16	8 digits of corporate de-personalization control key	M	4 bytes

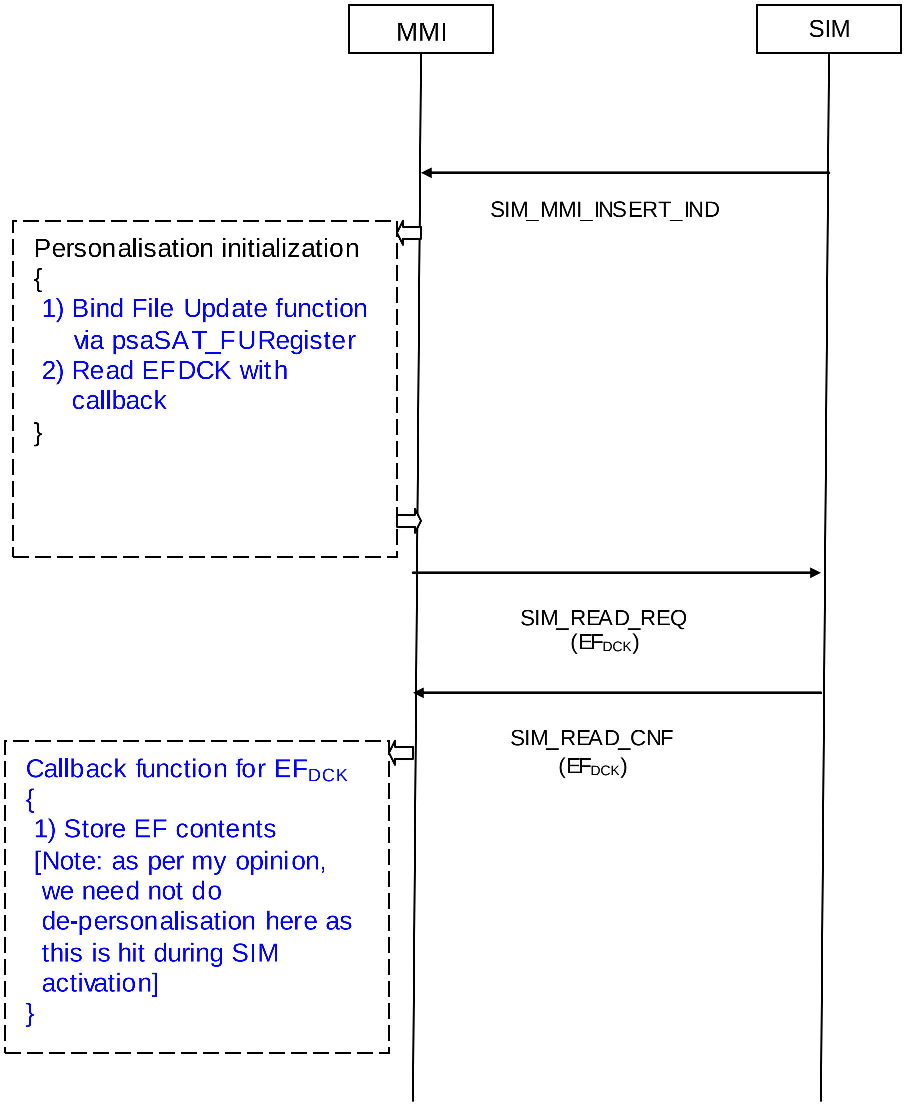
b) During initialization procedure (triggered due to REFRESH command), ME reads EFDCK if DCK service is “allocated and activated”.
1. For empty control key (coded as all ‘FF’), personalisation status shall not be changed
2. If control key matches, de-personalisation for that category
3. If control key differ, no changes in personalisation status

c) EFDCK is reset to default values.



```
File Update Function for EF_DCK
[Note: this would be bound during personalisation initialization via psaSAT_FURegister]
{
  1) Read EF_DCK with
     callback
}
```

```
Callback for read of EF_DCK
{
 (as per req. analysis)
 1) Interpret EF contents
 2) De-personalisation
    (function
    aci_slock_unlock)
 3) Write EF_DCK with default
    contents (stored during
    personalisation
    initialization)
}
```


MMI
SIM
SIM_MMI_INSERT_IND
Personalisation initialization
{
1) Bind File Update function via psaSAT_FURegister
2) Read EFDCK with callback
}
SIM_READ_REQ
(EFDCK)
SIM_READ_CNF
(EFDCK)
Callback function for EFDCK
{
1) Store EF contents
[Note: as per my opinion, we need not do de-personalisation here as this is hit during SIM activation]
}

4 New Interfaces added to ACI

1) aci_slock_set_CFG
To initialize the global configuration data by reading part1 of the MEPD from Security Driver.

2) aci_slock_autopersonalize
To auto-personalize to the first normal SIM inserted into the ME based on the autolock settings in the Security Driver.

3) aci_slock_autopersonalize_SP_CP
To auto-personalize ME on the SP & CP categories for the first normal SIM inserted into the ME.

4) aci_slock_permanent_unlock
To permanently unlock a category.

5) aci_slock_reset_fc
To reset the Failure counter.

6) aci_ext_add_code
To add code in ME Data.

Appendix A

1) **Supported regular expressions for storing interval information**

“xx123x“ where x is a single digit from 0-9