Technical Document

LLD SIXTIES SIM

LOW LEVEL DESIGN

Document Number:	xxxx.xxx.xx.xxx
Version:	1
Status:	Submitted
Approval Authority:	
Creation Date:	2004-Aug-10
Last changed:	2015-Mar-08 by Gareth Cleeves TI Employee
File Name:	lld_sim_sat_refresh.doc

Important Notice

Change History

Date	Changed by	Approved by	Version	Status	Notes
2004-Aug-10	Gareth Cleeves		0.1	Draft	1
2004-Aug-30	Gareth Cleeves	Comments from Martin Schroeder	1	Draft	

Notes:

1. Initial version

Table of Contents

List of References

[ISO 9000:2000]	International Organization for Standardization. Quality management systems - Fundamentals and vocabulary. December 2000
	SixTIes SAT Command Support High Level Design

1 Introduction

This document provides the low-level design for the modification to the SIM entity to be compatible with the SixTIes application.

An overview of the high-level design is provided in the HLD (reference xxxxx).

The modification to the SIM entity is restricted to the processing of the SAT Refresh command.

The SixTIes requirement is for the MMI to be informed about any SIM Refresh command and have the option to disallow the command.

There are no variants of the SIM entity. Therefore, on power up the ACI will inform the SIM entity if it is required to support the SixTIes functionality.

If SixTIes functionality is required then the Refresh command is forwarded to the ACI (it previously wasn't) and a response expected. The response will inform the SIM entity if the user accepts the Refresh or not.

If the user accepts the refresh command, the SIM entity proceeds as before by sending the appropriate command to the ACI, MM and SMS entities.

If the user rejects the refresh command, the SIM Entity will return an "ME currently unable to process command" Terminal Response to the SIM. The SIM will resend the Refresh Command to the SIM Entity, although the details of exactly how frequently are SIM specific.

2 The Refresh Proactive SIM Command

-	**SIM initialization <extra in 11.14 spec. see below>**	0x00	soft reset	11.14 mode 3
-	**File Change notification**	0x01	soft reset	11.14 mode 1
-	**SIM initialization and file change notification**	0x02	soft reset	11.14 mode 2
-	**SIM initialization and full file change notification**	0x03	soft reset	11.14 mode 0
-	**SIM Reset**	0x04	hard reset	11.14 mode 4

In contrast to the present behavior, SixTIes wants to be notified of all refresh processing:

Additionally SixTIes wants to have the possibility to allow the user to confirm or reject the refresh, which means, if the user rejects the refresh (including a SIM reset) a Terminal Response will be returned to the SIM with the value “ME currently unable to process command’

For each refresh type the original PDU must be delivered, in a SIM_TOOLKIT_IND, to the modem part and forwarded, in a %SATI indication, to the SixTIes MMI.

6.4.7 REFRESH <From 11.14>
The purpose of this command is to enable the ME to be notified of the changes to the SIM configuration that have occurred as the result of a SIM application activity. It is up to the SIM application to ensure that this is done correctly. The command supports five different modes:

- SIM Initialization. This mode tells the ME to carry out SIM initialization as it is defined in TS 11.11 [20], starting after the CHV1 verification procedure. The ME shall not reset the SIM electrically.

- File Change Notification. This mode advises the ME of the identity of the EFs that have been changed (in structure and/or contents) in the SIM. This information can be used by the ME if there is an image of SIM EFs (e.g. the ADN file) in the ME's memory, to determine whether it needs to update this image.

- SIM Initialization and File Change Notification. This is a combination of the first two modes above.

- SIM Initialization and Full File Change Notification. This mode causes the ME to perform the SIM initialization procedure of the first mode above and advises the ME that several EFs have been changed (in structure or contents) in the SIM. If there is an image of SIM EFs in the ME's memory, the ME shall completely update this image.

- SIM Reset. This mode causes the ME to run the GSM session termination procedure and to deactivate the SIM in accordance with TS 11.11 [20]. Subsequently, the ME activates the SIM again and starts a new card session. In case of a 3 Volt technology ME, the ME shall restart the SIM with the same supply voltage as in the previous session, if the ME can ensure that the SIM has not been changed in between. Otherwise, the ME shall perform the supply voltage switching in accordance with TS 11.12 [21]. The ME shall not send the TERMINAL RESPONSE; this is an exception from the normal procedure, where TERMINAL RESPONSE is sent after completion of the command. The SIM Application shall interpret a new activation of the contacts of the SIM as an implicit TERMINAL RESPONSE. The SIM Reset mode is used when a SIM application requires ATR or complete SIM initialization procedures to be performed. SIM Applications should take into account that early implementations of SIM Application Toolkit in some MEs may send a TERMINAL RESPONSE after performing the REFRESH command involving resetting the SIM electrically.

If the ME performs the REFRESH command successfully for only those EFs indicated in the mode, the ME shall inform the SIM using TERMINAL RESPONSE (OK), after it has completed its refreshing.
For REFRESH commands with mode other than "SIM Reset", it is permissible for the ME, as part of its execution of the REFRESH command, to read EFs in addition to those notified by the SIM, or to perform a SIM initialisation, provided that the procedure executed wholly encompasses the mode requested by the SIM. The ME shall not electrically reset the SIM. If the ME does the refreshing successfully, it shall inform the SIM using TERMINAL RESPONSE (Refresh performed with additional EFs read), after the ME has completed its refreshing. It should be noted that reading additional EFs will lengthen the refresh procedure.
If the ME receives a REFRESH command while in a state where execution of the command would be unacceptable, upsetting the current user operation (e.g. notification during a call that the IMSI has changed), the ME shall inform the SIM using TERMINAL RESPONSE (ME currently unable to process command - currently busy on call) or TERMINAL RESPONSE (ME currently unable to process command - screen is busy) as appropriate.

NOTE: Many MEs copy an image of the SIM's memory to the ME at initialization to speed up access to these fields during a GSM session. One of the purposes of this coding of the REFRESH command is to enable MEs to change such an image efficiently.

If, on receipt of the REFRESH command, the ME replies that it is busy (e.g. in call or navigating menus), the toolkit application may shorten the polling interval utilising the POLL INTERVAL command in order to resend the REFRESH command more frequently.
It is recommended for the ME to minimise the use of sending temporary problem TERMINAL RESPONSE, as during the period between the SIM issuing a REFRESH command and the ME performing the refresh procedure, there may be inconsistencies between data held in the ME and in the SIM. However, responsibility for retrying of all pro-active commands lies with the SIM Application.

12.6 Command details

Byte(s)	Description	Length
1	Command details tag	1
2	Length = '03'	1
3	Command number	1
4	Type of command	1
5	Command Qualifier	1

- Command number

For contents and coding, see subclause 6.5.1.

- Type of command:

Contents: The Type of Command specifies the required interpretation of the data objects which follow, and the required ME procedure.

Coding:
See section 13.4
The ME shall respond to reserved values (i.e. values not listed) with the result "Command type not understood".

- Command Qualifier:

Contents: Qualifiers specific to the command.

Coding:
- REFRESH;
  '00' =SIM Initialization and Full File Change Notification;
  '01' = File Change Notification;
  '02' = SIM Initialization and File Change Notification;
  '03' = SIM Initialization;
  '04' = SIM Reset;
  '05' to 'FF' = reserved values.

3 New Messages

Two new messages are required (a) to inform the SIM Entity on Power up if SixTIes support is required and (b) to indicated the user response to the request for the SIM Refresh command.

(a) To inform the SIM Entity on Power up if SixTIes support is required:-
This information is provided by adding an additional field to the SIM_ACTIVATE_REQ primative. The new field is **cust_mode** (short for custom mode). Cust_mode=1, SixTIes Support. Cust_mode=0, normal behaviour to apply.

(b) To indicated the user response to the request for the SIM Refresh command:-
SIM_REFRESH_USER_RES. This primitive will be sent from the ACI to the SIM.
Primitive contains a Field:- BOOL user_accepts. TRUE if user accepted the refresh command, FALSE if user has rejected it.
The primitive also contains an additional field:- stk_cmd. In which the MMI sends a pdu if the user rejects the refresh command. A Terminal Response is then generated from the pdu and sent to the SIM card.

4 Implementing the New Messages

4.1 New cust_mode field in the SIM_ACTIVATE_REQ primitive

The structure of the SIM_ACTIVATE_REQ primitive is defined by T_SIM_ACTIVE_REQ in ms\CDGINC\p_sim.h
The additional field, **cust_mode**, in the SIM_ACTIVATE_REQ primitive will be appended to the current structure:-

```
typedef struct
{
  U8                        proc;                    /*<  0:  1> procedure type       */
  U8                        mmi_pro_file;            /*<  1:  1> MMI profile          */
  U8                        stk_pro_file[MAX_STK_PRF]; /*<  2: 20> SIM toolkit profile */
  U8                        cust_mode;
  U8                        _align0;                 /*< 22:  1> alignment            */
} T_SIM_ACTIVATE_REQ;
```

This primitive is processed by app_sim_activate_req() in sim_app.c :-

When sim_activate_req.proc is SIM_INITIALISATION read the **cust_mode** value and store it in sim_data.
i.e.
if sim_activate_req.proc EQ SIM_INITIALISATION
then sim_data.cust_mode = sim_activate_req.cust_mode

The **cust_mode** value is held globally in sim_data structure .
sim_data is a data structure of type T_SIM_DATA which is defined in sim.h .
The additional field cust_mode, is added to the current structure:-

```
typedef struct
{
  // General SIM Variables
  ULONG          flags;                  // contains several flags for internal handling
  USHORT         act_directory;          // last selected directory
  USHORT         act_field;              // last selected elementary field
  T_TIME         status_time;            // status poll time
  USHORT         act_length;             // actual file or record length
  USHORT         sim_data_len;           // response data len
  USHORT         dir_status_len;         // size of directory status information
  USHORT         remove_error;           // provides error code for SIM Remove
  UBYTE          sim_phase;              // sim card phase
  UBYTE          last_requested_pin_no;  // last requested pin number
  UBYTE          act_access;             // actual access condition
  UBYTE          max_record;             // records of actual file
  UBYTE          field_type;             // of the last selected record
  UBYTE          sw1;                    // last status code SW1 from SIM
  UBYTE          sw2;                    // last status code SW2 from SIM
  // SIM Toolkit Variables
#ifdef SIM_TOOLKIT
  UBYTE          file_change_resp;       // tracks response to FILE CHANGE
  SHORT          proactive_sim_data_len;
  USHORT         stk_resp_len;
  UBYTE          stk_response[16];       // save TERMINAL RESPONSE
  UBYTE          stk_profile[MAX_STK_PRF];
  USHORT         cell_identity;
  T_loc_info     location_info;
  T_SAT_TIMER    timer[MAX_SAT_TIMER];
  UBYTE          sat_session;            /* SAT session started */
  UBYTE          ext_sat_cmd;            /* indicator for external SAT commands */
  UBYTE          term_resp_sent;         /* indicator if Terminal Response was sent */
  UBYTE          idle_polling;           /* poll SIM in idle mode */
```

```
  UBYTE          chk_sat_avail;          /* check SAT command to be fetched */
#ifdef FF_SAT_E
  UBYTE          dti_connection_state;   /* state of DTI connection */
  UBYTE          dti_rx_state;           /* state of DTI reception */
  UBYTE          dti_tx_state;           /* state of DTI transmission */
  UBYTE          event_data_avail;       /* Data available event */
  UBYTE          bip_ch_id;              /* BIP channel identifier */
  UBYTE          bip_suspend;            /* BIP suspension state */
  UBYTE          bip_state;              /* state of BIP channel */
  UBYTE          bip_rx_state;           /* state of BIP reception */
  UBYTE          bip_tx_state;           /* state of BIP transmission */
  UBYTE          bip_timer_state;        /* state of BIP release timer */
  UBYTE          bip_general_result;     /* general result code for on demand link */
  UBYTE          bip_add_info_result;    /* additional information for on demand link */
  UBYTE          con_type;               /* connection type */
  USHORT         received_data_pos;      /* start of not yet delivered data */
  DTI_HANDLE     hDTI;                   /* DTILIB handle */
  T_TIME         bip_release_time;       /* BIP release time */
  ULONG          link_id;                /* DTI link identifer */
  T_desc_list2   data_to_send;           /* DTI data to send */
  T_desc_list2   prev_data_to_send;      /* last values of DTI data to send */
  T_desc_list2   received_data;          /* received DTI data */
#ifdef WAP
  T_SRC_DES      udp_parameters;         /* ports and IP addresses for UDP */
#endif /* WAP */
  T_SIM_DTI_REQ* sim_dti_req;            /* storage of SIM_DTI_REQ primitive */
  T_cmd_details  bip_rx_cmd_details;     /* command details for RX */
  T_cmd_details  bip_tx_cmd_details;     /* command details for TX */
  BUF_cmd_prms   bip_cmd_prms;           /* parameter buffer for RX and TX */
#endif /* FF_SAT_E */
  UBYTE          cust_mode;              /* custom: 0=default, 1=SixTIes*/
#endif /* SIM_TOOLKIT */
  // Windows Simulation Variables
#if defined (_SIMULATION_)
  USHORT         mode;
#endif /* _SIMULATION_ */


} T_SIM_DATA;
```

NOTE: The field cust_mode in sim_data is not required if a flag is used to represent the cust_mode value received in the SIM_ACTIVATE_REQ primitive

4.2 New message SIM_REFRESH_USER_RES

This new primative indicates the user response to the request for the SIM Refresh command. SIM_REFRESH_USER_RES primitive is sent from the MMI to the SIM.

The structure of the SIM_REFRESH_USER_RES primitive is defined by T_SIM_REFRESH_USER_RES in ms\CDGINC\p_sim.h

```
typedef struct
{

  U8                      user_accepts;      /*<  0:  1> 1 = accepted, 0 = rejected   */
  U8                       _align0;          /*<  1:  1> alignment                    */
  U8                       _align1;          /*<  2:  1> alignment                    */
  U8                       _align2;          /*<  3:  1> alignment                    */
  T_stk_cmd                stk_cmd;
} T_SIM_REFRESH_USER_RES;
```

The new primitive is processed by stk_sim_refresh_user_res() in sim_stk.c
An entry into sim_table[] (ms\src\sim_pei.c) is required to facilitate this processing :-

```
LOCAL const T_FUNC sim_table[] = {
  MAK_FUNC_0 (app_sim_read_req,          SIM_READ_REQ            ), /* 4500 */
  MAK_FUNC_0 (app_sim_update_req,        SIM_UPDATE_REQ          ), /* 4501 */
  MAK_FUNC_0 (app_sim_read_record_req,   SIM_READ_RECORD_REQ     ), /* 4502 */
  MAK_FUNC_N (pei_not_supported,         0                       ), /* 4503 */
  MAK_FUNC_0 (app_sim_update_record_req, SIM_UPDATE_RECORD_REQ   ), /* 4504 */
  MAK_FUNC_N (pei_not_supported,         0                       ), /* 4505 */
  MAK_FUNC_N (pei_not_supported,         0                       ), /* 4506 */
#if defined SIM_TOOLKIT
  MAK_FUNC_0 (stk_sim_refresh_user_res,   SIM_REFRESH_USER_RES  ), /*  */

#else /* SIM_TOOLKIT */
  MAK_FUNC_N (pei_not_supported,         0                       ), /* 4507 */
#endif /* else SIM_TOOLKIT */
  MAK_FUNC_0 (app_sim_increment_req,     SIM_INCREMENT_REQ       ), /* 4508 */
  MAK_FUNC_0 (app_sim_verify_pin_req,    SIM_VERIFY_PIN_REQ      ), /* 4509 */
  MAK_FUNC_0 (app_sim_change_pin_req,    SIM_CHANGE_PIN_REQ      ), /* 450a */
  MAK_FUNC_0 (app_sim_disable_pin_req,   SIM_DISABLE_PIN_REQ     ), /* 450b */
  MAK_FUNC_0 (app_sim_enable_pin_req,    SIM_ENABLE_PIN_REQ      ), /* 450c */
  MAK_FUNC_0 (app_sim_unblock_req,       SIM_UNBLOCK_REQ         ), /* 450d */
  MAK_FUNC_0 (app_sim_auth_req,          SIM_AUTHENTICATION_REQ  ), /* 450e */
  MAK_FUNC_0 (app_sim_mm_update_req,     SIM_MM_UPDATE_REQ       ), /* 450f */
  MAK_FUNC_0 (app_sim_sync_req,          SIM_SYNC_REQ            ), /* 4510 */
  MAK_FUNC_0 (app_sim_activate_req,      SIM_ACTIVATE_REQ        ), /* 4511 */
#if defined SIM_TOOLKIT
  MAK_FUNC_0 (stk_sim_toolkit_req,       SIM_TOOLKIT_REQ         ), /* 4512 */
  MAK_FUNC_0 (stk_sim_toolkit_res,       SIM_TOOLKIT_RES         ), /* 4513 */
#else /* SIM_TOOLKIT */
  MAK_FUNC_N (pei_not_supported,         0                       ), /* 4512 */
  MAK_FUNC_N (pei_not_supported,         0                       ), /* 4513 */
#endif /* else SIM_TOOLKIT */
  MAK_FUNC_0 (app_sim_access_req,        SIM_ACCESS_REQ          ), /* 4514 */
#if defined SIM_TOOLKIT
  MAK_FUNC_0 (stk_file_update_res,       SIM_FILE_UPDATE_RES     ), /* 4515 */
#else /* SIM_TOOLKIT */
  MAK_FUNC_N (pei_not_supported,         0                       ), /* 4515 */
#endif /* else SIM_TOOLKIT */
#if defined GPRS
  MAK_FUNC_0 (gprs_sim_gmm_update_req,   SIM_GMM_UPDATE_REQ      ), /* 4516 */
#else /* GPRS */
```

```
  MAK_FUNC_N (pei_not_supported,         0                        ), /* 4516 */
#endif /* else GPRS */
#if defined FF_SAT_E
  MAK_FUNC_0 (stk_sim_dti_req,           SIM_DTI_REQ              ), /* 4517 */
  MAK_FUNC_0 (stk_sim_eventlist_req,     SIM_EVENTLIST_REQ        ), /* 4518 */
#else
  MAK_FUNC_N (pei_not_supported,         0                        ), /* 4517 */
  MAK_FUNC_N (pei_not_supported,         0                        ), /* 4518 */
#endif /* FF_SAT_E */
};



GLOBAL void stk_sim_refresh_user_res (T_SIM_REFRESH_USER_RES * sim_refresh_user_res)
{
   TRACE_FUNCTION ("stk_sim_refresh_user_res()");

   /*check primitive is expected */
   <If cust_mode NEQ 1 then log error and exit>
   <IF sim_data.user_confirmation_expected NEQ TRUE then log error and exit>


   if ((sim_refresh_user_res ->user_accepts) EQ FALSE)
   {
      /*user rejected Refresh Command Request      */
      /*Send Terminal_Resp to SIM                  */
      FKT_TerminalResponse (sim_toolkit_res->stk_cmd.cmd,
                            (USHORT)(sim_toolkit_res->stk_cmd.l_cmd>>3));
      /*terminate */
      <check any Messages to Free ??????>
   }
   else
   {
      /*user accepted Refresh Command Request */
      <Process as before>
      <Call procedure to process Refresh Command>
      <NOTE:Ensure Messages are freed correctly>
   }
   sim_data.user_confirmation_expected = FALSE;
   PFREE (sim_refresh_user_res);

}
```

The above function, in sim_app.c, is called to process the SIM_REFRESH_USER_RES primitive. If the user accepts the Refresh Request (`user_accepts` = 1) then this function continues with the processing of the Refresh Command that used to be handled in the stk_proactive_polling() function. If the user rejects the Refresh Request (`user_accepts` = 0, stk_cmd = pdu from MMI) a Terminal Response is sent to the SIM built from the the pdu received from the MMI.

5 Implementing the SAT Refresh Command

The SAT Refresh Command is processed within stk_proactive_polling() as well as in stk_sim_refresh_user_res() (see previous section).
The stk_proactive_polling() function is in sim_stk.c .
The stk_proactive_polling() code needs to be modified to reflect the new functionality:-

If cust_mode EQ 1
Then send Refresh command to MMI and terminate
If cust_mode EQ 0
Process Refresh Command as before

Modifications to stk_proactive_polling() function are :-

```
Global void stk_proactive_polling (void)
{

    -------
    PALLOC (sim_toolkit_ind, SIM_TOOLKIT_IND)
    <clear primitive>
    -------
    -------

        switch (cmd_type)
        {
          case STK_REFRESH:

            If (sim_data.cust_mode EQ 0)
            {
               /*default operation*/
               call function for processing Refresh Commands
            }
            ElseIf (sim_data.cust_mode EQ 1)
            {
               /*sixTIes operation*/

               sim_data.user_confirmation_expected = TRUE;

               /*send Refresh command to MMI*/
               PALLOC (sim_toolkit_ind, SIM_TOOLKIT_IND);
               memset (sim_toolkit_ind, 0, sizeof (T_SIM_TOOLKIT_IND));
               /* copy PDU into message ???????? */
               PSENDX (MMI, sim_toolkit_ind);

               /*REFRESH COMMAND NOW PROCESSED IN stk_sim_refresh_user_res() */
               /*Terminate Command                                            */
               break;
            }

             ElseIf (sim_data.cust_mode EQ 2)
            {
               /*Next Custom Requirement*/
            }

            < PROCESS AS BEFORE >
            NOTE: This code for processing the Refresh Commands will be placed in a separate
                  Function.
            switch (cmd_qual)
            {
                -------
                -------


            }/*end of switch (cmd_qual)*/


            -------
```

```
            -------
            -------

            PFREE (sim_toolkit_ind)
            Break; /* end of case STK_REFRESH */
```

```
Add STK_RES_SCREEN_IS_BUSY 0x01 (11.14 12.12.2 ) to sim_stk.c
#define STK_RES_EXT_DEF                  0x00
#define STK_RES_SCREEN_IS_BUSY           0x01
#define STK_RES_EXT_BUSY_CALL            0x02
#define STK_RES_EXT_NO_SERVICE           0x04
```

6 Testing with Windows Simulation

The following tests need to be performed on the SIM entity:-

1) SIM_ACTIVATE_REQ primitive to be sent with cust_mode = 0.
Ensure that all current tests pass.

2) SIM_ ACTIVATE_REQ primitive to be sent with cust_mode=1.
Test Refresh File Change Notification
Test Refresh Reset
Test Refresh Sim Initialization and File Change Notification

3) Test error conditions. e.g.:-
SIM_REFRESH_USER_RES primitive received when cust_mode=0
SIM_REFRESH_USER_RES primitive received when cust_mode=1 but a Refresh command has not occurred (i.e. user_confirmation_expected EQ FALSE)

A. Appendices

B. Acronyms

DS-WCDMA	Direct Sequence/Spread Wideband Code Division Multiple Access

C. Glossary

International Mobile Telecommunication 2000 (IMT-2000/ITU-2000)	Formerly referred to as FPLMTS (Future Public Land-Mobile Telephone System), this is the ITU's specification/family of standards for 3G. This initiative provides a global infrastructure through both satellite and terrestrial systems, for fixed and mobile phone users. The family of standards is a framework comprising a mix/blend of systems providing global roaming. <URL: http://www.imt-2000.org/>