Technical Document

LLD: NOTIFICATION OF SATK FILE CHANGE

Document Number:	20_04_04_02483
Version:	0.2
Status:	Draft
Approval Authority:	
Creation Date:	2005-Jun-1
Last changed:	2005-Jul-25 by Andrew Spruce
File Name:	lld_aci_simef.doc

Important Notice

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

All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment. TI warrants performance of its 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 products, software and/or services. To minimize the risks associated with customer products and applications, customers should provide adequate design, testing and operating safeguards.

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

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

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

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

Change History

Date	Changed by	Approved by	Version	Status	Notes
2005-Jun-1	Andrew Spruce		0.1	Draft	1
2005-Jul-25	Andrew Spruce		0.2	Draft	2

Notes:

1. Initial version.
2. File renamed.

Table of Contents

1 Introduction

During the design phase of the Atlas (XT6) project, it has been determined that enhancements to the ACI, ATI and BAT are required in order to provide notification of SIM elementary file updates.

This document details the work required to implement this feature, which is based on the proposals described in the documents "SIM_FILE_UPDATE_IND handling" by Liyi Yu and "SATK refresh for T&A" by Marcus Oakland. These are summarised below.

1.1 File Change Notification

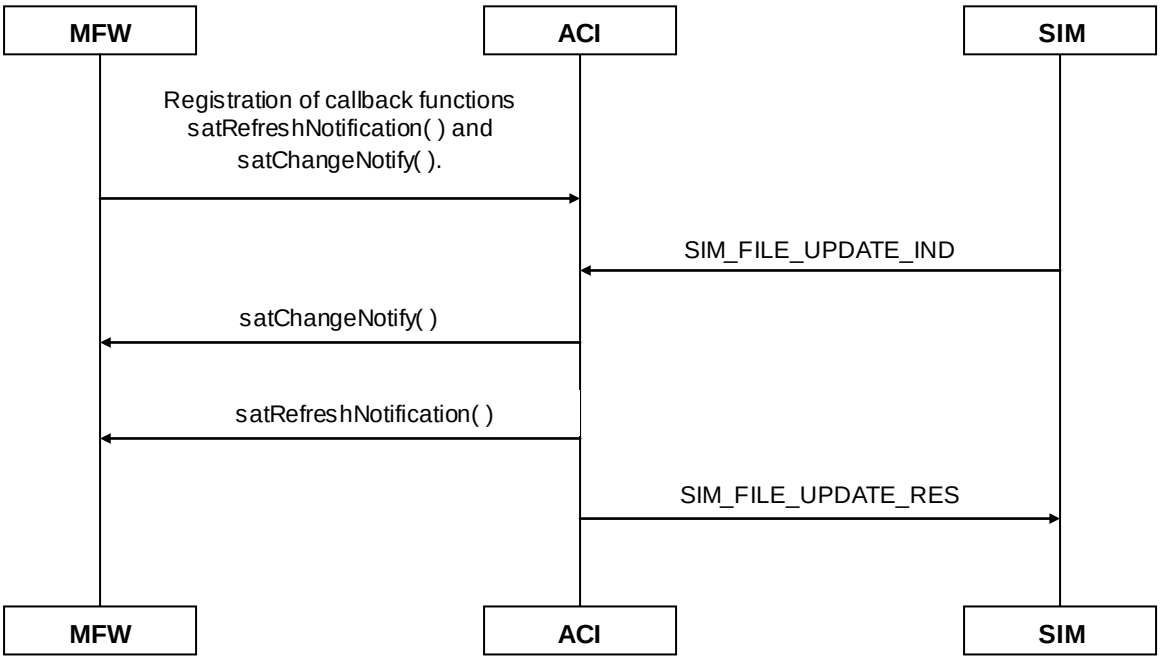
The Protocol Stack SIM handler sends a SIM_FILE_UPDATE_IND primitive to the ACI with a T_SIM_FILE_UPDATE_IND structure containing the list of the files to be updated.

```
typedef struct
{
  U16 val_nr;               /*<  0:  2> valid entries of a file ID list */
  U16 file_id[MAX_FILE_ID]; /*<  2:128> Identifier of an EF             */
  U8 _align0;               /*<130:  1> alignment                      */
  U8  align1;               /*<131:  1> alignment                      */
} T_SIM_FILE_UPDATE_IND;
```

The "file_id" array contains the file IDs for the SIM files that need to be re-read, the "val_nr" indicates the number of valid elements in this array.

The ACI's psa_sim_file_update_ind( ) function handles the received SIM_FILE_UPDATE_IND primitive, and ultimately causes the response (SIM_FILE_UPDATE_RES) to be sent to the stack.

1.2 Existing MFW Implementation



The T_SIM_FILE_UPDATE_IND structure is passed to a callback function, satRefreshNotification( ) that has previously been registered using psaSAT_FUNotifyRegister( ). When the MFW has finished

updating the indicated files, or has failed to do so, it calls the ACI function psaSAT_FUConfirm( ). This causes the SIM_FILE_UPDATE_RES primitive to be sent.

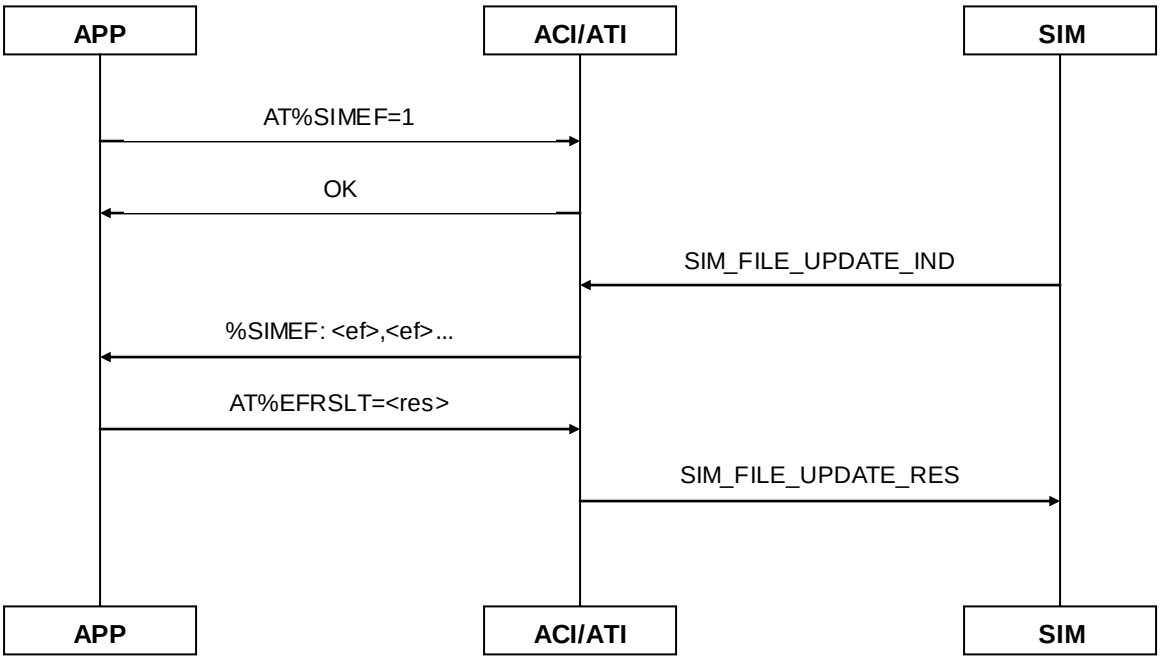
1.3 Proposed ACI/ATI Implementation

The ACI and ATI will be enhanced to provide :

A new unsolicited result code (%SIMEF:) that reports the information received in the T_SIM_FILE_UPDATE_IND structure.

A new command (%EFRSLT) that is used to indicate success or failure to update the indicated SIM files.

A new command (%SIMEF) that is used to turn this feature on or off.



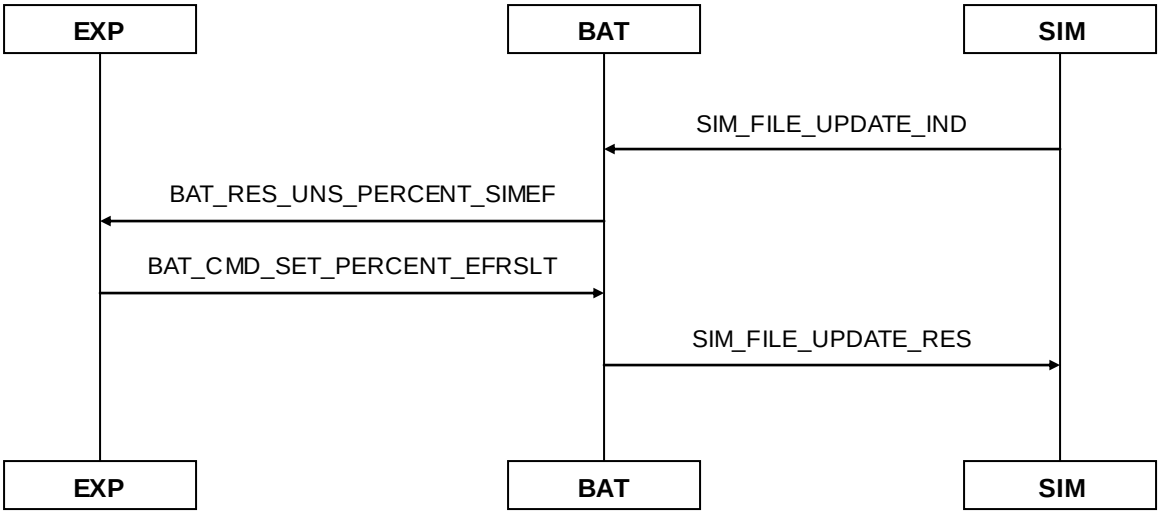
1.4 Proposed EXP/BAT Implementation

BAT will be enhanced to provide :

A new unsolicited result code (BAT_RES_UNS_PERCENT_SIMEF) that passes the information received in the T_SIM_FILE_UPDATE_IND structure to the EXP.

A new BAT command (BAT_CMD_SET_PERCENT_EFRSLT) that is used by the EXP to indicate success or failure to update the indicated SIM files.

BAT will respond to the new BAT command with BAT_RES_AT_OK if there is a file change notification in progress, or will respond with BAT_RES_PLUS_CME_ERROR if there is not.


EXP
BAT
SIM
SIM_FILE_UPDATE_IND
BAT_RES_UNS_PERCENT_SIMEF
BAT_CMD_SET_PERCENT_EFRSLT
SIM_FILE_UPDATE_RES
EXP
BAT
SIM

2 ACI

Definitions for the command parameters will be added to **aci_cmh.h** :

```
typedef enum
{
  EFRSLT_RES_FAIL,
  EFRSLT_RES_OK
}
T_ACI_EFRSLT_RES;

typedef enum
{
  SIMEF_MODE_OFF,
  SIMEF_MODE_ON
}
T_ACI_SIMEF_MODE;
```

RAT_P_SIMEF will be added to RAT_ID in **cmh.h**, and the corresponding modification to RATJmpTbl in **cmh_f.c** will be made.

```
GLOBAL const T_VOID_FUNC RATJmpTbl[RAT_MAX][CMD_MODE_MAX] =
{
.
.
.
  CB_VC( PercentCPRI ),   /*RAT_P_CPRI*/
  CB_VC( PercentSIMEF ),  /*RAT_P_SIMEF*/

#ifdef  SIMULATION
,
  CB_VC( PercentCNIV )    /*RAT_P_CNIV*/
#endif

};
```

The SIMEF mode will be added to the T_MM_CMD_PRM structure in **cmh.h**.

```
typedef struct mmCmdPrm          /* command parameters related to MM */
{
.
.
.
  T_ACI_SIMEF_MODE SIMEFMode;
} T_MM_CMD_PRM;
```

This new parameter will be initialised with the rest of the T_MM_CMD_PRM values in cmh_Reset( ) in **cmh_f.c**, to SIMEF_MODE_OFF.

The function psa_sim_file_update_ind( ) in **psa_satp.c**, which is called when the primitive SIM_FILE_UPDATE_IND is received, will be modified in order to generate the %SIMEF unsolicited event. The function psaSAT_FUConfirm( ) will then only be called if MFW is present - it is expected that any other application will use %EFRSLT.

```
GLOBAL const void psa_sim_file_update_ind
                      ( T_SIM_FILE_UPDATE_IND *sim_file_update_ind )
{
  SHORT i;
  BOOL send_confirm_now=TRUE;

  TRACE_FUNCTION ("psa_sim_file_update_ind()");

/*
 *-------------------------------------------------------------------
 * store the primitive and start processing
 *-------------------------------------------------------------------
 */
  satShrdPrm.fu_ind = sim_file_update_ind;

  /*
  *   Broadcast %SIMEF unsolicited event to all interested parties.
  */
  for(i=CMD SRC LCL;i<CMD SRC MAX;i++)
  {
    if ((cmhPrm[i].mmCmdPrm.SIMEFMode==SIMEF_MODE_ON)
      || (aci cmd src mode get(i)==CMD MODE BAT))
    {
      /*
      *   As we have reported the EF update, it is not now the
      *   responsibility of this function to send the response
      *   immediately.
      */
      send_confirm_now=FALSE;

      R_AT(RAT_P_SIMEF,i)(
        (SHORT *)sim_file_update_ind->file_id,
        (UBYTE)sim_file_update_ind->val_nr);
    }
  }

  /*
  *   If we didn't report the EF update to anyone we must send
  *   the response now.
  */
  if (send_confirm_now==TRUE)
    psaSAT_FUConfirm (-1, NOT_PRESENT_16BIT);
}
```

In **cmh_sats.c** (corresponding prototypes in **aci_cmh.h**), the functions sAT_PercentEFRSLT( ) and sAT_PercentSIMEF( ) will be implemented.

```
GLOBAL T_ACI_RETURN sAT_PercentEFRSLT (T_ACI_CMD_SRC srcId,
                                       T_ACI_EFRSLT_RES result)
{
  if(!cmh_IsVldCmdSrc(srcId))
  {
    ACI_ERR_DESC(ACI_ERR_CLASS_Ext,EXT_ERR_Parameter);
    return(AT_FAIL);
  }

  switch(result)
  {
    case EFRSLT_RES_FAIL:
      psaSAT FUConfirm(simShrdPrm.fuRef,SIM FU ERROR);
      break;

    case EFRSLT_RES_OK:
      psaSAT_FUConfirm(simShrdPrm.fuRef,SIM_FU_SUCCESS);
      break;

    default:
      ACI_ERR_DESC(ACI_ERR_CLASS_Ext,EXT_ERR_Parameter);
      return(AT_FAIL);
  }

  return(AT_CMPL);
}


GLOBAL T_ACI_RETURN sAT_PercentSIMEF (T_ACI_CMD_SRC srcId,
                                      T_ACI_SIMEF_MODE mode)
{
  if(!cmh_IsVldCmdSrc(srcId))
  {
    ACI_ERR_DESC(ACI_ERR_CLASS_Ext,EXT_ERR_Parameter);
    return(AT_FAIL);
  }

  switch(mode)
  {
    case SIMEF_MODE_ON:
    case SIMEF_MODE_OFF:
      cmhPrm[srcId].mmCmdPrm.SIMEFMode=mode;
      break;

    default:
      ACI_ERR_DESC(ACI_ERR_CLASS_Ext,EXT_ERR_Parameter);
      return(AT_FAIL);
  }
  return(AT_CMPL);
}
```

In **cmh_satq.c** (corresponding prototype in **aci_cmh.h**), qAT_PercentSIMEF( ) will be implemented.

```
GLOBAL T_ACI_RETURN qAT_PercentSIMEF (T_ACI_CMD_SRC srcId,
                                      T_ACI_SIMEF_MODE *mode)
{
  TRACE_FUNCTION ("qAT_PercentSIMEF()");

  /*
  *   Check command source
  */
  if(!cmh_IsVldCmdSrc (srcId))
```

```
    return(AT_FAIL);

  *mode=cmhPrm[srcId].mmCmdPrm.SIMEFMode;

  return (AT_CMPL);
}
```

3 ATI

AT_CMD_P_SIMEF and AT_CMD_P_EFRSLT will be added to T_ACI_AT_CMD in **aci_cmh.h**. The new commands will be added to the 'cmds' table in **ati_cmd.c** :

```
LOCAL const ATCommand cmds [] =
{
.
.
.
  SAT_CMD("%SIMEF",AT_CMD_P_SIMEF,setatPercentSIMEF,test_gen,queatPercentSIMEF,0)
  SAT_CMD("%EFRSLT",AT_CMD_P_EFRSLT,setatPercentEFRSLT,test_gen,0,0)
  {NULL,0,0,0,0,0}
};
```

In **ati_sat.c** , setatPercentEFRSLT( ), setatPercentSIMEF( ) and queatPercentSIMEF( ) will be added.

```
GLOBAL T ATI RSLT setatPercentEFRSLT(char *cl,UBYTE srcId)
{
  T_ACI_EFRSLT_RES res;
  T ACI RETURN ret;

  TRACE_FUNCTION("setatPercentEFRSLT()");

  switch(*cl)
  {
    case '0':
      res=EFRSLT_RES_FAIL;
      break;

    case '1':
      res=EFRSLT_RES_OK;
      break;

    default:
      cmdCmeError(CME_ERR_Unknown);
      return(ATI_FAIL);
  }

  ret=sAT_PercentEFRSLT(srcId,res);

  return(map_aci_2_ati_rslt(ret));
}


GLOBAL T_ATI_RSLT setatPercentSIMEF(char *cl,UBYTE srcId)
{
  T_ACI_SIMEF_MODE mode;
  T_ACI_RETURN ret;

  TRACE_FUNCTION("setatPercentSIMEF()");

  switch(*cl)
  {
    case '0':
      mode=SIMEF_MODE_OFF;
      break;

    case '1':
      mode=SIMEF MODE ON;
      break;

    default:
```

```
      cmdCmeError(CME_ERR_Unknown);
      return(ATI_FAIL);
  }

  ret=sAT_PercentSIMEF(srcId,mode);

  return(map_aci_2_ati_rslt(ret));
}

GLOBAL T_ATI_RSLT queatPercentSIMEF(char *cl, UBYTE srcId)
{
  T_ACI_RETURN ret;
  T_ACI_SIMEF_MODE mode;

  TRACE_FUNCTION("queatPercentSIMEF");

  ret=qAT_PercentSIMEF(srcId,&mode);

  if (ret==AT_CMPL)
  {
    sprintf(g_sa,"%s: %d","%SIMEF",mode);
    io_sendMessage(srcId,g_sa,ATI_NORMAL_OUTPUT);
  }

  return(map aci 2 ati rslt(ret));
}
```

In **ati_ret.c** , rCI_PercentSIMEF( ) will be added.

```
GLOBAL void rCI_PercentSIMEF(
  SHORT *ef,
  UBYTE count)
{
  UBYTE srcId=srcId_cb;
  SHORT pos;
  UBYTE n;

  TRACE_FUNCTION("rCI_PercentSIMEF()");

  pos=sprintf(g_sa,"%s: ","%SIMEF");

  for (n=0;n<count;n++)
  {
    pos+=sprintf(g_sa+pos,"%04X",ef[n]);

    if (n!=count-1)
      *(g_sa+pos++)=',';
  }

  io_sendMessage(srcId,g_sa,ATI_NORMAL_OUTPUT);
}
```

4 BAT

The script file **cmd_list.log** will be modified to include the new messages :

```
{
%SIMEF : SIM EF Update Indication
[uns rsp
-ef : EF : U16[0..MAX_SIMEF_EF_LEN=64] : man : List of files changed
]
}

{
%EFRSLT : SIM EF Update Indication Response
[cmd
-result : RES : U8 : man : Result of update
0 : FAIL : Update failed
1 : OK   : Update was successful
]
}
```

This will result in the following structures being added to **p_bat.h**,

```
typedef struct
{
  U8            _align0;                  /*<  0:  1> alignment */
  U8            c_ef;                     /*<  1:  1> counter */
  U16           ef[BAT_MAX_SIMEF_EF_LEN]; /*<  2: 64> List of files changed */
  U8            _align1;                  /*<130:  1> alignment */
  U8            _align2;                  /*<131:  1> alignment */
} T_BAT_res_uns_percent_simef;

typedef struct
{
  T_BAT_percent_efrslt_result result;     /*<  0:  4> Result of update */
} T_BAT_cmd_set_percent_efrslt;
```

and in **p_bat_val.h**, definitions for BAT_CMD_SET_PERCENT_EFRSLT and BAT_RES_UNS_PERCENT_SIMEF will be created.

In **aci_bat.c**, the new command and response will be inserted into the appropriate tables :

```
LOCAL const BATCommand bat_cmds [] =
{
.
.
.
  { /* BAT_CMD_SET_PERCENT_EFRSLT     */ sBAT_PercentEFRSLT  },



static T_map_response_2_size unsolicited_2_size [] =
{
.
.
.
  {/* BAT_RES_UNS_PERCENT_SIMEF      */ sizeof(T_BAT_res_uns_percent_simef)},
```

In **aci_bat_cb.c**, rBAT_PercentSIMEF( ) will be implemented (corresponding prototype in **aci_bat_cb.h** and **aci_cmh.h**) :

```
GLOBAL void rBAT_PercentSIMEF(
  SHORT *ef,
  UBYTE count)
{
  T_ACI_CMD_SRC src_id;
  T_ACI_DTI_PRC_PSI *src_infos;
  T_BAT_cmd_response resp;
  T_BAT_res_uns_percent_simef simef_data;
  UBYTE n;

  TRACE_FUNCTION ("rBAT_PercentSIMEF()");

  /*
  *   Get the source ID and a pointer to the PSI source information.
  */
  aci_bat_src_info(&src_id,&src_infos);

  resp.ctrl response=BAT RES UNS PERCENT SIMEF;
  resp.response.ptr_res_percent_simef=&simef_data;

  /*
  *   Copy the data in. Note that any excess data will be lost, however
  *   this should not happen as the BAT message has been designed to
  *   carry the entire contents of the T_SIM_FILE_UPDATE_IND primitive.
  */
  for (n=0;((n<count) && (n<BAT_MAX_SIMEF_EF_LEN));n++)
  {
    simef_data.ef[n]=(U16)ef[n];
  }

  simef_data.c_ef=(U8)n;

  aci_bat_send(src_infos,&resp);
}
```

In **aci_bat_sim.c**, sBAT_PercentEFRSLT( ) will be implemented (the corresponding prototype will be in **aci_bat_cmh.h**).

```
GLOBAL T_ACI_BAT_RSLT sBAT_PercentEFRSLT   (T_ACI_DTI_PRC_PSI *src_infos_psi,
                                            T_BAT_cmd_send    *cmd)
{
  T_ACI_EFRSLT_RES result;
  T_ACI_BAT_RSLT ret;

  TRACE_FUNCTION ("sBAT_PercentEFRSLT()");

  /*
  *   This relies on T_BAT_percent_efrslt_result being identical to
  *   T_ACI_EFRSLT_RES.
  */
  result=(T_ACI_EFRSLT_RES)cmd->params.ptr_set_percent_efrslt->result;

  /*
  *   Call the corresponding sAT function. T_ACI_BAT_RSLT is
  *   assumed to be equivalent to T_ACI_RESULT.
  */
  ret=(T_ACI_BAT_RSLT)sAT_PercentEFRSLT(src_infos_psi->srcId,result);

  return(ret);
}
```

5 Testing

Tapcaller test cases will be written, for example :

```
T_CASE ACISAT621()
{
   BEGIN_CASE ("simef_efrslt_acisat621")
   {
      power_on_with_various_terminal_profiles_acisat012('F');

      SEND (aci_cmd_req("AT%SIMEF=1"));
      AWAIT (aci_cmd_ind ("OK"));

      SEND (sim_file_update_ind_1('A'));
      AWAIT (aci_cmd_ind ("%SIMEF: 6F20"));

      SEND (aci cmd req("AT%EFRSLT=1"));
      AWAIT (sim_file_update_res_1());
      AWAIT (aci_cmd_ind ("OK"));
   }
}
```

It is anticipated that the BAT messages will be tested the same way. In order to achieve this it will be necessary to modify the corresponding ATI routines to generate and process BAT messages. As this area is currently under development, and the mechanism may change significantly before the work described in this document is implemented, this will not be detailed at present.

6 Documentation

This section describes changes that will be made to existing documents.

6.1 AT Command Interface Description (8415.052)

6.1.1 %SIMEF: SIM EF Update Indication

Command	Possible Response(s)
%SIMEF=<mode>	OK +CME ERROR: <err>
%SIMEF?	%SIMEF: <mode>

Description

This command is used to select or deselect the reporting of EF updates.

When <mode>=1, and the ACI receives the primitive SIM_FILE_UPDATE_IND, unsolicited result code %SIMEF: <ef>,<ef>,<ef>... (where each <ef> is a 4-digit hexadecimal number representing an elementary file ID) is sent to the TE.

Defined Values

<mode> : EF update indication mode

0 Off - EF updates will not be reported.
1 On - EF updates will be reported (and will have to be acknowledged using %EFRSLT).

6.1.2 %EFRSLT: SIM EF Update Indication Response

Command	Possible Response(s)
%EFRSLT=<res>	OK +CME ERROR: <err>

Description

This command is sent in response to the %SIMEF unsolicited event in order to report the success or otherwise of the update. It causes the ACI to send the primitive SIM_FILE_UPDATE_RES.

Defined Values

<res> : Indicates whether the update was successfully handled or not.
0 Not successfully handled
1 Successfully handled

6.2 ACI Functional Interface Description (8411.802)

The following will be added :

6.2.1 T_ACI_EFRSLT_RES

enum type	value	symbolic constant	comment
T_ACI_EFRSLT_RES	0	EFRSLT_RES_FAIL	Refresh failed
	1	EFRSLT_RES_OK	Refresh was successful

6.2.2 T_ACI_SIMEF_MODE

enum type	value	symbolic constant	comment
T_ACI_SIMEF_MODE	0	SIMEF_MODE_OFF	No reporting of EF update.
	1	SIMEF_MODE_ON	EF update reported, application must respond with %EFRSLT.

6.2.3 sAT_PercentEFRSLT( ) - SIM EF Update Indication Response

Command Reference :

8415.052

Function Definition :

```
T_ACI_RETURN sAT_PercentEFRSLT(
        T_ACI_CMD_SRC srcId,
        T_ACI_EFRSLT_RES result);
```

Parameters:

name	buffer size	comment	
srcId	---	command source identifier	IN
result	---	success or failure of update	IN

Return:

symbolic constant	comment
AT_CMPL	successfully completed
AT_FAIL	failed

6.2.4 sAT_PercentSIMEF( ) - SIM EF Update Indication

Command Reference :

8415.052

Function Definition :

```
T_ACI_RETURN sAT_PercentSIMEF(
        T_ACI_CMD_SRC srcId,
        T_ACI_SIMEF_MODE mode);
```

Parameters:

name	buffer size	comment	
srcId	---	command source identifier	IN
mode	---	enable/disable SIM EF update indication	IN

Return:

symbolic constant	comment
AT_CMPL	successfully completed
AT_FAIL	failed

6.2.5 qAT_PercentSIMEF( ) - SIM EF Update Indication

Command Reference :

8415.052

Function Definition :

```
T_ACI_RETURN qAT_PercentSIMEF(
        T_ACI_CMD_SRC srcId,
        T_ACI_SIMEF_MODE *mode);
```

Parameters:

name	buffer size	comment	
srcId	---	command source identifier	IN
mode	---	enable/disable SIM EF update indication	OUT

Return:

symbolic constant	comment
AT_CMPL	successfully completed
AT_FAIL	failed

6.2.6 rAT_PercentSIMEF( ) - SIM EF Update Indication

Command Reference:

8415.052

Function Definition:

```
void rAT_PercentSIMEF(
        SHORT *ef,
        UBYTE count);
```

Parameters:

name	buffer size	comment	
ef	MAX_FILE_ID	pointer to array of elementary file IDs	IN
count	---	number of files changed	IN

Appendices

A. Abbreviations

ACI	Application Control Interface
EF	Elementary File
EXP	Alcatel Exploitation Layer
SAP	Service Access Point
SIM	Subscriber Identity Module