# LLD ACI Only Adaptation

| | |
|---|---|
| Project | G23-GSM Protocol Stack |
| Document Type | Technical Documentation |
| Title | LLD ACI Only Adaptation |
| Author | Christophe Le Berre |
| Creation Date | 18 July, 2001 |
| Last Modified | 24 October, 2002 |
| ID and Version | 8462.706.02.001 |
| Status | Submitted |

# 0 Document Control



## 0.1 Document History

| ID | Author | Date | Status |
|---|---|---|---|
| 8462.706.02.001 | CLB | 24 October, 2002 | Submitted |

Initial version.**References, Abbreviations, Terms**

[TI 7010.801] 7010.801, References and Vocabulary, Texas Instruments.

## Table of Contents

# 1 Introduction

The module Application Control Interface (ACI) is the layer between the layer 2/3 of the /GSM /GPRS /UMTS protocols and the applications layer (often described as Man Machine Interface MMI), the interface to the protocol stack offered to the application(s) being based on AT commands. ACI is at the moment available in different configurations to allow different levels and types of implementations for the user applications.
There are basically two types of implementation: ***local*** for build-in MMIs (e.g keyboard/display on any regular mobile) using a functional interface with ACI and on the other hand ***remote*** for ascii AT-command strings control of ACI (e.g applications running on PDAs or on a computer connected via an serial interface).

Three different configurations exist at the moment in order to provide different levels of effort that a customer has to bring to develop a user application:
*Note: the remote interface is present in every of the following configurations.*

- **BMI/MFW**: Basic MMI/MMI framework.
  This is the most complete version offering an already quite complete build-in MMI in terms of features. There is little left to the customer to be implemented in terms of G23 related features.

- **SMI**: Slim MMI
  This configuration offers just some basic MMI features and the customer has to implement its own complex features using the ACI functional interface (e.g: phonebook, sim toolkit related displays...).

- **ATI only**: AT interpreter only
  This configuration is the lightest one. Only the remote interface is available to the user applications.

The ATI only version was broken since the "marriage" between the old GSM stack and the new GPRS functionalities. The purpose of this document is to describe what the problems are and to propose a solution to get this version able to run again.

It is indeed very important as most customers have very high requirements in terms of memory consumption and this version provides to the remote access customers a great optimisation in this area.

# 2 The ATI only version:

## 2.1 Why is it broken ?

The major reason why it is broken is a quite confusing use of the task name and of some related parameters.

Indeed, depending on the configuration different modules are packed in the same task as ACI. For instance, in the BMI version, the task contains three modules: BMI, MFW and ACI. In the SMI version the task contains two of them, SMI and ACI.

To follow this idea, the old GSM implementation used to have the name of the task changed according to the configuration. So there was either a "MMI" task, a "SMI" task or an "ACI" task.

It had been then decided for the GPRS versions that the task should always be called "MMI", whatever the version. This had been thoroughly changed in the case of the SMI and MMI versions, but not in the case of the ATI only version.

Some other closely related problems also exist: a few parameters related to the task itself (handles, prefix for functions...) are named differently for different configurations.

This all totally confuses the software so that it cannot start properly in case of the ATI only version.

## 2.2 Proposal Overview

The use of different names for a same thing should be avoided when possible. There is no obvious reason why depending on the configuration these tasks related parameters should have different denominations.

There are also some concerns about using the name MMI_NAME ("MMI") to characterise the task. There might be some confusion with the user applications standing on top of ACI. Moreover, in the future other modules (such as MFW or BMI) will run in their own tasks so that ACI is left alone in the task.

That is why the best solution is to name every task/pei related parameter or function after ACI, and not after MMI as it is at the moment.

It would be probably best though to proceed in two steps. First to remove MMI_NAME and replace it by ACI_NAME, but keeping it defined to "MMI" first.

## 2.3 How to proceed ?

Let's have a look in details at the changes that have to take place:

### 2.3.1 Entity name

#### 2.3.1.1 ACI_NAME:

Every entity needing to communicate with another one (ENT) has to call the frame function vsi_c_open with the parameter ENT_NAME (typically equal to "ENT").

In the case of ACI, different names are to be found in the current implementation: MMI_NAME, SMI_NAME, ACI_NAME and even GACI_NAME. Most entities use MMI_NAME for all different versions though.

The major change would be to have only one name for ACI and to avoid confusion with upper modules or entities it should be: ACI_NAME: "ACI".

As already said, this should be done in two steps: changing directly to "ACI" would indeed imply some major changes in the test applications (x-panel, PCO for windows simulation as well as for target testing). This could be easily left to a second step involving the developers of these applications. So in a first step, ACI_NAME will be used, but it will have the value "MMI".

This means some changes in some global files:

\g23m\condat\com\inc\gprs.h and \g23m\condat\com\include\custom.h where these ENT_NAMEs are defined. MMI_NAME, SMI_NAME and GACI_NAME are to be removed to have only ACI_NAME left (this will avoid possible confusion in the future when implementing newly created entities).

All references to the removed parameters are then to be replaced by ACI_NAME. This basically concerns every PEI module of entities having communication with ACI (about 30 files concerned for GSM/GPRS. UMTS case still to be checked).

#### 2.3.1.2 aci_pei_create( ):

When starting the software, the frame has to initialise each entity. It does so by calling for each entity a dedicated ent_pei_create( ) function. There is a table of such functions for each entity in \g23m\condat\com\src\config\gsmcomp.c for GSM and in \g23m\condat\frame\config\ gprscomp.c for GPRS: CondatComponentTable (for the UMTS case whether an extra table exists and what functions name it uses has to be checked ). This table actually has two different entries for ACI: one for the ATI only version (compiler flag ACI) and another for all other versions.

However, there should be only one such function: aci_pei_create( ). References to mmi_pei_create should be removed.

Moreover, this function (to be found in \g23m\condat\ms\src\aci\aci_pei.c) aci_pei_create( ) returns information about the task to the frame; its own name for instance. There again a difference is being made in the current implementation between the ATI only version and the others ("ACI" is returned in the first case whereas "MMI" is in the second). This shall be replaced by ACI_NAME in all versions (this will also mean no more changes on this side when the entity name will be later changed from "MMI" to "ACI").

After having done this, it is expected that the ATI only version software runs correctly.

### 2.3.2 Handles and Entity Prefix

It does make sense to make some further cleaning in the same direction to avoid confusion and therefore to ease future implementations and maintenance.

#### 2.3.2.1 Communication handles:

So-called handles are entity IDs that are to be used for inter-entity communication. It is usual that an entity requests from the frame such an ID at software initialisation: hCommENT.
Handles concerning ACI are currently found in different versions: hCommMMI, hCommACI or hCommSMI. This could become very confusing especially when ACI and MFW are to be split.
The most proper way would be to use only hCommACI for communication with ACI. In most entities hCommMMI is however used and this should eventually be changed. This is really an internal setting to each entity though (this is ensured by _ENTITY_PREFIXED see following), so this is something that each entity responsible should do for himself if he feels like it is needed. There is no real urge to take action here.
Different ACI handles are to be found in ACI (such handles are needed so that ACI sends itself messages for rescheduling): hCommACI should replace them independently from the version in use.

#### 2.3.2.2 Entity Prefix:

To avoid the problem of having global functions/parameters double defined, an "entity prefixed" is added to some global functions/parameters names. This is usually done by adding the name of the entity at the beginning of the function:

e.g.: #define hCommFAD _ENTITY_PREFIXED(hCommFAD) in t30.h
where: #define _ENTITY_PREFIXED(N) t30_##N for entity T30

_ENTITY_PREFIXED has different settings in case of ACI. "aci_", "mfw_", "mmi_" or "smi_".
This also has no sense since these parameters can be defined only once for each version.
There shall be only one _ENTITY_PREFIXED for ACI: "aci_".

#### 2.3.2.3 ENTITY_ENT:

This precompiler switch is used by global header files to identify which entity is currently compiled. There again confusion is not avoided in the current implementation since there are four different possible settings for ACI: ENTITY_ACI, ENTITY_MMI, ENTITY_SMI and ENTITY_MFW.
This switch is used for instance:

- to define the above mentioned _ENTITY_PREFIXED;
- to have a specific header file included instead of another one (e.g: in \com\include\custom.h decision on which cus_.h is to be included);
- to choose what SAP is relevant for a specific version (in \com\include\prim.h).

There again should be a separation between what is task relevant (in all cases there is only one task, and this is ACI) and what is module relevant (different settings for files belonging to

MFW/SMI/ACI ?). A close examination of the use of ENTITY_ENT uncovers that the switch is really meant to differentiate the tasks and not the modules.
So there shall be only a single ENTITY_ACI defined in every ACI file (ACI seen here as a task, not a module: MFW files are therefore here also affected) whatever the version. This #define could be put in an ACI global header file which per default should be included as first header in any ACI file.
There are at the moment no "modules" relevant settings, but if this is needed another type of precompiler switch could be used: MOD_MEMBER (MOD for module). There is already an ACI_MEMBER for ACI which is actually in the current implementation quite misused. Some SMI_MEMBER or MFW_MEMBER could easily be added. There is no obvious use for them at the moment though.

These changes would have two more consequences:
- the merge of different cus_.h files. There should be only one such file for the whole task, with possibly different settings according to the version (using the switches (ACI), (MFW) or (SMI)). Such things as VSI_CALLER which inform the frame about the task currently processing an action should only have one definition as it is definitely not depending on the version (this is also why the definition of VSI_CALLER should be changed from mmi_handle to aci_handle).
It would be thus much easier to update these settings rather than to have to change different files.
- no longer having different lists of SAPs for different version: having different lists is very unlikely to be needed. Moreover even if it is, configuring directly in a sole list means a much easier updating or maintenance of the list.

### 2.3.3 Test applications

Let's see what changes would be needed to proceed the second step: ACI_NAME changed from "MMI" to "ACI".

As already explained, this should involve only the test applications: PCO, x-panel...

#### 2.3.3.1 X-panel:

this application will barely be able to work further: pressing the "keypad" interface is expected to be broken and no more at-command will be able to be sent. The application sends indeed explicitly messages (keypad indication for instance) to the receiver "MMI" instead of "ACI".

#### 2.3.3.2 PCO:

Problems here are easy to overcome. It is actually only a question of configuration. Two files are mainly concerned: view.txt and pco_stack.xml (both to be found under C:\..\PCO\bin C:\..\PCO\ being the folder where the PCO application is installed).

pco_stack.xml is responsible for the tracing user interface accessed by the button "send stack configuration". In the file one just has to replace MMI (should occur only once) by ACI.

view.txt contains all config primitives proposed to the user (using buttons "send" and picking the wanted config primitives). In this file, all config primitives are defined as following:

Reference (name which will be offered to the user)

Config primitive (message which is really sent to the stack)

Receiver (entity receiving the config primitive)

For each config primitive where MMI stands as receiver, it will have to be replaced by ACI. It might be wise to leave the existing references so that all users do not get confused (and this concerns a lot of people including customers).

#### 2.3.3.3 Old Panel - windows simulation:

This still has to be analysed. It might not be relevant anymore.

## 2.4 Planned changes

Following are the changes to be undertaken (as described above):

### 2.4.1 Compulsory Steps:

These steps are compulsory to get a running ATI only version.

- *Action*: Replace MMI_NAME / SMI_NAME / GACI_NAME by ACI_NAME (still equal to "MMI")
  *Where*: \g23m\condat\com\inc\gprs.h
  \g23m\condat\com\include\custom.h
  Several _pei.c files of entities exchanging primitives with ACI (CC, SMS, GRR etc...)
- *Action*: Remove mmi_pei_create( )
  *Where*: aci_pei.c
  \g23m\condat\com\src\config\gsmcomp.c
  \g23m\condat\frame\config\gprscomp.c
- *Action*: Replace "MMI" or "ACI" by ACI_NAME in aci_pei_create( )
  *Where*: aci_pei.c

### 2.4.2 Further Steps:

These steps shall also be a part of this project.

- *Action*: Replace hCommMMI and hCommSMI by hComACI in ACI
  (other entities should also do this: yet this is to be decided by the corresponding responsibles)
  *Where*: several ACI/SMI/BTI/MFW source files.
- *Action*: Have _ENTITY_PREFIXED always set to aci_
  *Where*: \g23m\condat\com\include\gsm.h
- *Action*: Remove ENTITY_MMI, ENTITY_SMI and ENTITY_SMI and where it is needed set ENTITY_ACI instead.
  *Where*: Every single file belonging to the ACI task
  \g23m\condat\com\include\custom.h, prim.h and message.h
- *Action*: Merge of different ACI related cus_....h files
  *Where*: cus_aci.h, cus_smi.h, cus_mmi.h and cus_asb.h are to merged
  \g23m\condat\com\include\custom.h

### 2.4.3 Future Steps:

These steps will not be a part of this project. This is only a proposal for the future.

- *Action*: Change ACI_NAME from "MMI" to "ACI"
  *Where*: This will have to be checked by the GPF team as this mostly impacts test and tracing

applications.

# 3 Summary of the changes made

## 3.1 Changes directly related to this project

These changes and the reason why they had to be done are thoroughly detailed in the former parts of this document. As planned changes described in chapters 2.4.1 and 2.4.1 have been made.

Further changes (e.g changes described in chapter 2.4.3) have been left for future projects.

## 3.2 Other changes undertaken

This project has been the occasion to take further changes.

The first goal has been to get a good testing coverage to ensure the best quality for this project. This meant having functioning windows test documents and this implied some maintenance work both on documents and code sides.

Other changes concern the overall improvement of the ACI module, mainly in two areas: readability of the code (for better maintenance) and memory consumption.

Following is a list of the major changes that have taken place:

### 3.2.1 Huge functions split

Split of some HUGE functions (to improve readability):
cmhCC_CallDisconnected( ), cmhCC_NotifySS( ), sAT_PlusCHLD(), process_CHLDaddInfo()

### 3.2.2 cmhCC_getcalltype( )

Creation of a function returning the type of call from the callId:
cmhCC_getcalltype( ). Avoid the constant use of bearer capabilities to check the type of call.
(e.g in cmhCC_CallConnected( ))

### 3.2.3 function tables

all function tables between PSA and CMH have been removed

### 3.2.4 Removal of ccShrdPrm.cId

Instead cId is passed between CMH and PSA as a function parameter: cmhCC_CallReleased(), psaCC_RetrieveMPTY(), psaCC_RetrieveCall(), psaCC_BuildMPTY(), psaCC_SplitMPTY(), psaCC_SendSS(), psaCC_ECT(), cmhCC_CallConnected(), cmhCC_CallAlerted(), cmhCC_CallProceeding(), cmhCC_CallModified(), cmhCC_DisconnectCall()...

### 3.2.5 Less use of ssShrdPrm.sId

Number of ssShrdPrms has been reduced from 277 to 70. sId is now passed as parameter in functions. The remaining ones are really needed to keep tracks in asynchronous cases (waiting for answer from the network....)

#### 3.2.6 aci_all.h

Creation of a header file to be included in all ACI files (ACI as a module, not as a task: so not in MFW/BMI or SMI files): aci_all.h Most global header files (such as prim.h, custom.h) are included in this header file. Moreover switch ENTITY_XXX is defined according to the current configuration.
message.h and prim.h are not included anymore in aci_cmh.h (so these 2 files had to be included in numerous MFW, BMI or SMI files).

#### 3.2.7 ati_int.h

creation of ati_int.h: prototypes of parameters and functions internal to the ATI module.

#### 3.2.8 ati_is_src_type()

Creation of ati_is_src_type() to detect any given source type: used to replace buggy check against CMD_SRC_ATI_5 in case of run at command source...

#### 3.2.9 #ifndef NEW_FRAME

Removal of #ifndef NEW_FRAMEs in ACI

#### 3.2.10 GSM only

Get a GSM only build: mfw.mak and bmi.mak had to be changed (add #ifdef GPRS for GPRS only files). includes of ffs.h need #undef GPRS

#### 3.2.11 Improvement of code readability

- Removal of sa_length parse()
- prototype in aci_prs.h
- at_PercentCPRIM moved from cmh_mmis.c to ati_csc.c
- ErrBuff[MAX_CMD_LEN] moved from ati_cmd.c to ati_err.c
- psaMMI_ConfigPrim declared in psa_mmi.h
- rename mem into phb_mem_names
- removal of T_RA_CMD_PRM (unused)
- setat_plusCUSD( ) reworked
- removal of psaSS_CCDDecodeErr( )
- removal of psa_mncc_user_ind( )
- psa_mnss_end_ind: SAT case is now handled in an extra function
- Global buffer subBuf replaced by dynamically allocated memory
- add if( ccShrdPrm.ctb[cId].ntryUsdFlg EQ TRUE ) in ATZ
- less traces in getparameteroverid (check if uart source)
- setatpluscops( ): check oper[0] EQ 0 now done in cmh

#### 3.2.12 Test document aci.doc

Several changes in aci.doc:

- use only NUM_ELEMENTS( )
- CCBS fac had 83 for basicservicegroup tag: A3 now
- ACI931 erased: create variant C in ACI153 instead
- Removal of ACI920 to 924 (AT%TEST does not exist anymore) + remove useless declarations
- Remove ACI031B and all 500rs tests

#### 3.2.13 Several bugfixes

Several bugfixes:

- ATA in case of CCBS did not get an OK
- from 1.3.2 SHORT cId = raShrdPrm.cId used in cmhCC_L2R_or_TRA_Deactivated()
- in sAT_A( ) set curCmd to AT_CMD_A also for CCBS (and not AT_CMD_D)
- remove pCtbNtry -> inBndTns = FALSE; at the end of psaCC_ClearCall (TC aci056 / 058 )
- cmhSS_CLIR_Interrogated(): add an else otherwise more than one +CLIR
- For +CSVM: if no phone number then ton=145 (see 07.07)
- CHLD=12 problem: use X_prim[2] instead of *X_Prim
- CMODmode moved from cmdPrm to ccShrdPrm (not source dependant)

#### 3.2.14 Memory consumption improvement

To spare some code:

- creation of send_CSSU_notification() (resp. same for CSSI)
- creation of cmhrat_ccbs( ) for CCBS callbacks: this solves also a bug. Sometimes %CCBS is an indication, sometimes it is an intermediate result
- creation of chld_ratlog( ) to inform MFW within CHLD
- New Function psaCC_send_satevent()
- cmhCC_atdsendok(cId) introduced in cmhCC_CallConnected and psaCC_NewCall.()
- creation of cmhCC_accept_call( )
- more use of cmhCC_ClearCall (for CHUP or CHLD for instance)
- Split T_CMH_PRM into nonFD and FD to avoid having to include aci_fd.h before cmh.h
- removal of mtBc from ccShrdPrms

#### 3.2.15 Removal of warnings

- in aci_ext_pers.c add (UBYTE) in personalisation_status.locks1 = (UBYTE)FldSet(tra);

- add (USHORT)(frame_len << 3) in psaDTI_data_test_req( )
- cast (UBYTE) added concShrdPrm.udh.ref_num = (UBYTE)concSMS_findMaxRefNum() in concSMS_split()

#### 3.2.16 Removal of warnings

- Description of AT%CUNS has been added.
- Description of AT%ALS has been updated
- Descriptions of AT%ETAI and AT%ECDI have been removed (they do not exist)