High Level Design Specification

PSI Enhancement

Department:	WTBU - Cellular Systems		
Creation Date:	2005-01-17		
Last Modified:	2005-04-28 by RM		
ID:	16_04_03_00961	Version:	101
Status:	Updated	ECCN:	Not Applicable

0 Document Control

0.1 Export Control Statement

Recipient agrees that it will not knowingly export or re-export, directly or indirectly, any product or technical data (as defined by the U.S, EU and other Export Administration Regulations) including software, or any controlled product restricted by other applicable national regulations, received from Disclosing party under this Agreement, or any direct product of such technology, to any destination to which such export or re-export is restricted or prohibited by U.S or other applicable laws, without obtaining prior authorisation from U.S. Department of Commerce and other competent Government authorities to the extent required by those laws. This provision shall survive termination or expiration of this Agreement.

According to our best knowledge of the state and end-use of this product or technology, and in compliance with the export control regulations of dual-use goods in force in the origin and exporting countries, this

technology is classified as given on the front page.

This product or technology may require export or re-export license for shipping it in compliance with certain countries regulations.

0.2 Document History

Date	Version	Status	Author
2005-04-28 Updated version.	101	Updated	RM
2005-02-14 Reviewed version.	100	Approved	RM
2005-01-17 Initial version.	001	Draft	RM

0.3 References

[1] ISO 9000:2000: International Organization for Standardization. Quality management systems - Fundamentals and vocabulary. December 2000

[2] 8462.720.01.001High Level Description Protocol Stack Interface (PSI)

[3] 88_08_03_01166 Data I/O Interface (DIO)

[4] 8462.725.04.001Low Level Description Protocol Stack Interface (PSI)

[5] 19_04_03_01398: The High-Level Design Specification of the BAT library

0.4 Abbreviations

BAT	Binary AT Interface
DIO IL	Data I/O Interface Layer
FFS	Flash File System
PKTIO	Packet Data I/O Entity
PSI	Protocol Stack Interface
SAP	Service Access Point
UART	Universal Asynchronous Receiver-Transmitter

Table of Contents

List of Figures and Tables

1 Introduction

As described in [2] a unified interface for the Protocol Stack (PSI - Protocol Stack Interface) is introduced which increases the flexibility, reduces the effort of maintenance, and simplifies the internal handling and the tests. In addition it makes simpler extensions to new drivers and internal applications because only a small layer needs to be adapted. The DIO interface layer (DIO IL), version 4,[3]serves controlling of driver access to the TI Protocol Stack. It provides a functional interface to PSI. Up to now PSI is only used for access to an USB driver (phase 1B of PSI implementation project [2]).This document describes the needed effort of PSI/ACI adaptation for replacement of the entities UART and PKTIO (phase 2). From ACI and PSI point of view each driver or application using DIO interface layer is generally called “device”.

2 General Enhancement of PSI

1. The possibility of each device to transport AT command strings and/or user data is called “device data mode”. The values are part of the PSI SAP and have to be stored in the FFS in the manufacturing process. PSI uses these FFS entries for sending the according value to ACI in the device registration session.
2. The configuration parameters of each device have to be stored in the FFS in the manufacturing process too. PSI uses these FFS entries for the device default configuration.

In order to make PSI more stable it is necessary to improve the design in some points:

3. If ACI sends changed line states for serial devices like DCD state in the primitive PSI_LINE_STATE_REQ or in the sent DTI2 data parameters, PSI has to start the flush procedure as the same way as changing of device configuration parameters (Figure_10).

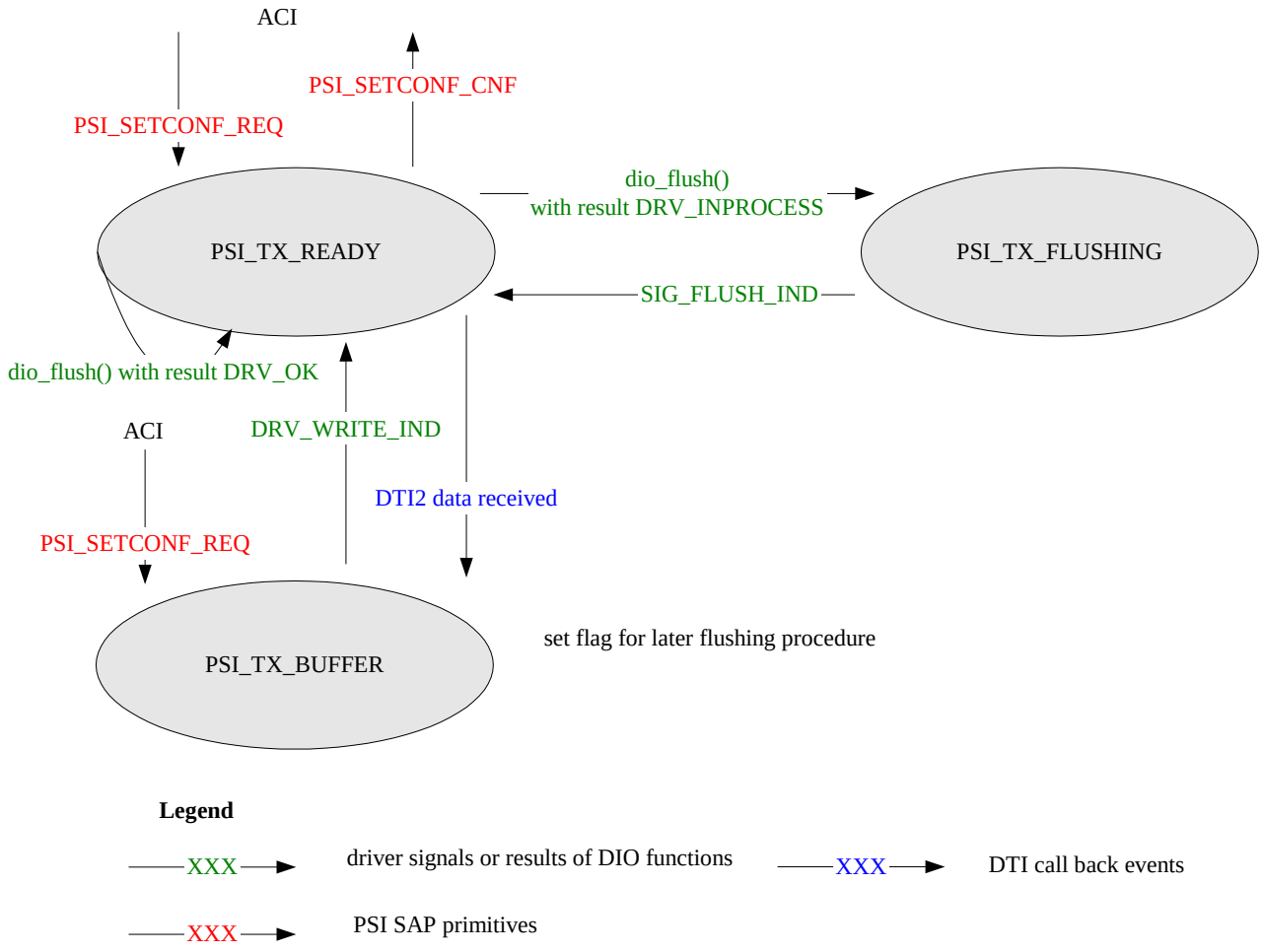

Figure 1 Fragment of current PSI TX state machine during the flushing procedure

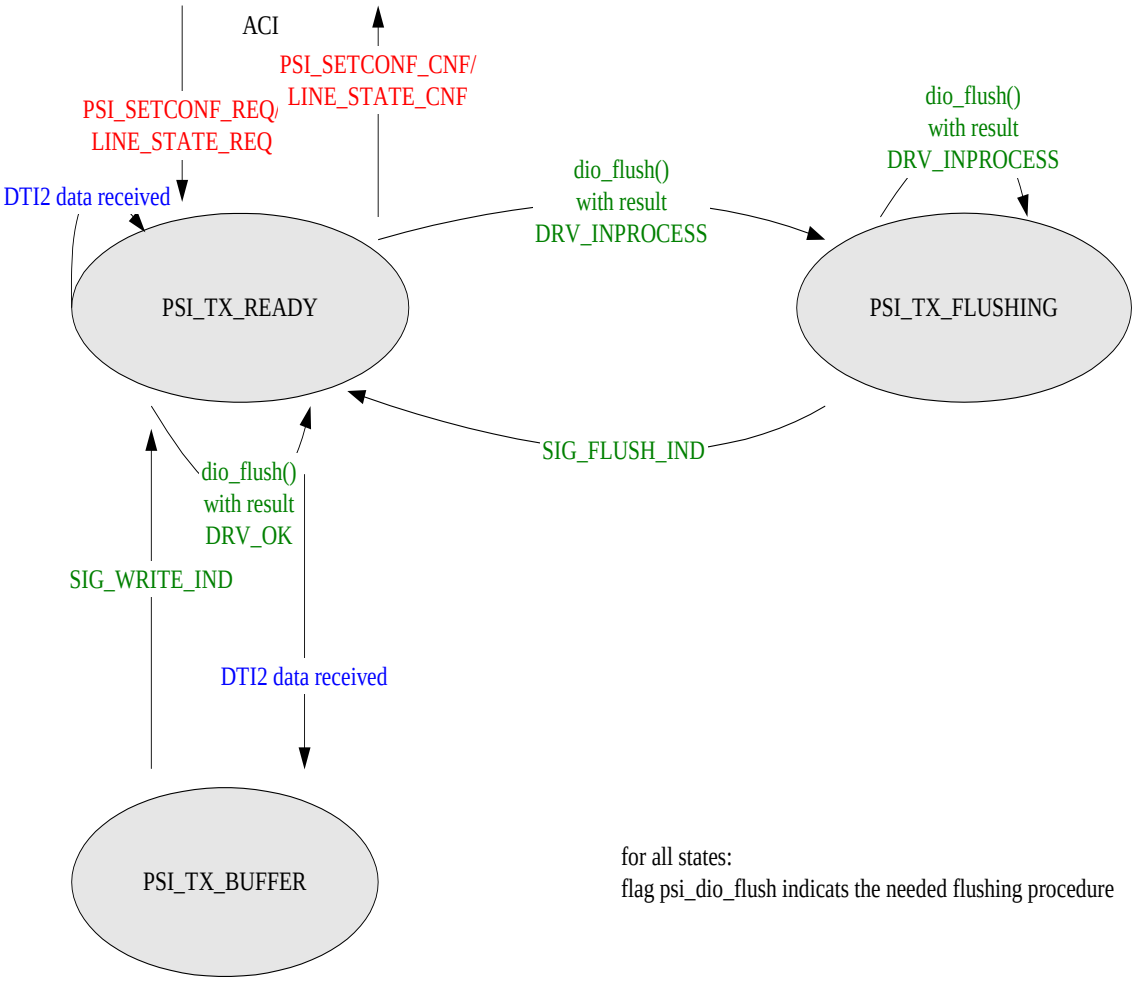

Figure 2 Fragment of enhanced PSI TX state machine during the flushing procedure

Minor changes:

4. PSI sets segment size of read buffer with respect to a more effectively usage of GPF memory partition pool.
5. If ACI wants to do a deregistration of the device and it sends PSI_CLOSE_REQ, PSI has to check the status of DTI connection

3 UART entity

3.1 Current Integration of UART driver

Retrospect: In the past multiplexer support (MUX) had been part of the UART entity.

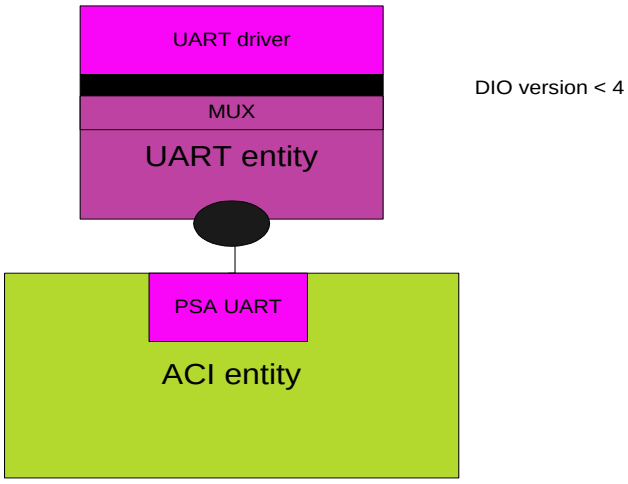

Figure 3 UART configuration before PSI implementation project phase 1A

The multiplexer functionality was removed from the UART entity in the phase 1A of the PSI implementation project [2]. Both the UART driver and a new separate multiplexer layer MUX process the multiplexed serial data transmission and support the DIO interface, v4. The small Serial Adaptation Module (SAM) adapts the UF interface of the UART driver to the DIO interface, v4. The UART entity is able to communicate via DIO interface, v4 with the DIO interface layer, as well.

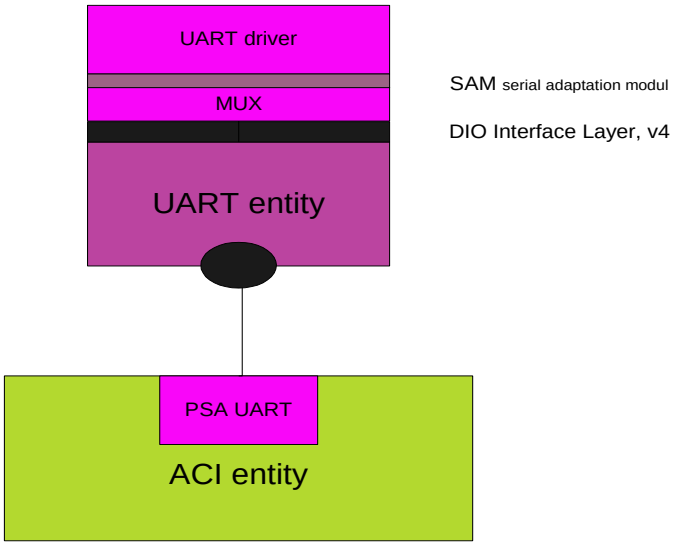

Figure 4 Current UART driver / UART entity configuration

3.2 New Integration of UART driver

Since the current PSI implementation supports serial data transmission the entity UART is not more necessary. The following scheme shows the new configuration of the according entities.

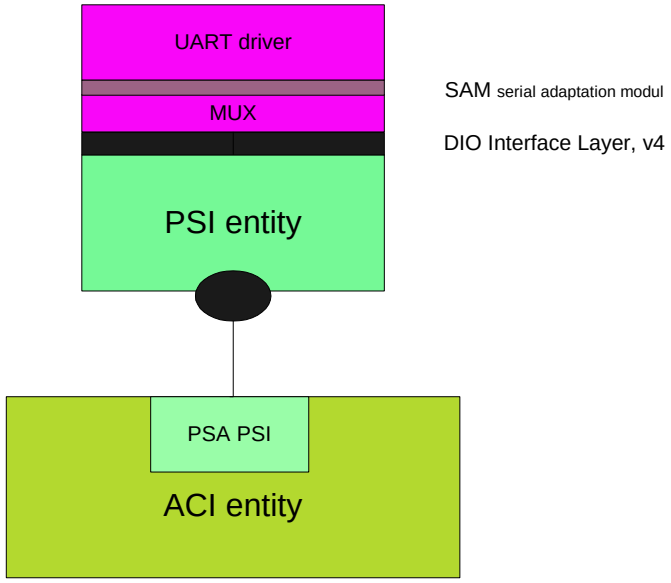

Figure 5 New configuration regarding UART driver and PSI entity

3.2.1 Enhancement in PSI

1. Handling of MUX device capability and MUX configuration parameter in the entity.
2. In general: PSI contains DIO dummy functions and test primitives for the simulation test of the functional DIO interface. The dummy functions send primitives called PSI_DIOSIM_XXX_REQ to TAP and wait for the confirmation primitives called PSI_DIOSIM_XXX_CNF sent by TAP. The provided different result values or parameters in these primitives allow testing several driver scenarios. Some further test primitives testing the PSI SAP are necessary because of the fact that TDC doesn't support casting. They are named PSI_XXX_TEST.
   For the MUX driver simulation enhancements of the simulation functions described above are necessary.
3. Sending of default configuration of UART driver read from FFS.
4. Testing: several PSI instances; handling more than one provided buffer segment, enhanced flush procedure

3.2.2 Adaptation in ACI

In general: ACI uses the parameter *device_type* of structure T_DIO_CAP for distinguish the device data type support (instead of evaluation of parameter *devId*). The number of max DTI links has to be calculated in a new way (used as parameter for DTI lib configuration).

3.2.2.1 Replace UART SAP by PSI SAP

The MUX related primitives of the UART SAP are not used anymore. The remained UART primitives are be covered by the PSI SAP.

3.2.2.2 Enhancement PSI part in ACI by handling device type UART

The UART lists *uart_src_params* and *uart_com_par* are covered by the existing lists *psi_dev_list* and *psi_src_params*. The list *psi_dcb_par* ACI used for PSI handling can be removed. The list content is added to *psi_dev_list*.

3.2.2.2.1 *No MUX mode*

1. The UART device registration ensues as same way as for the USB device in the PSI part of ACI. PSI sends the first UART device configuration to the device (instead of ACI). The DTI control manager starts immediately the new DTI connection (Figure 6).

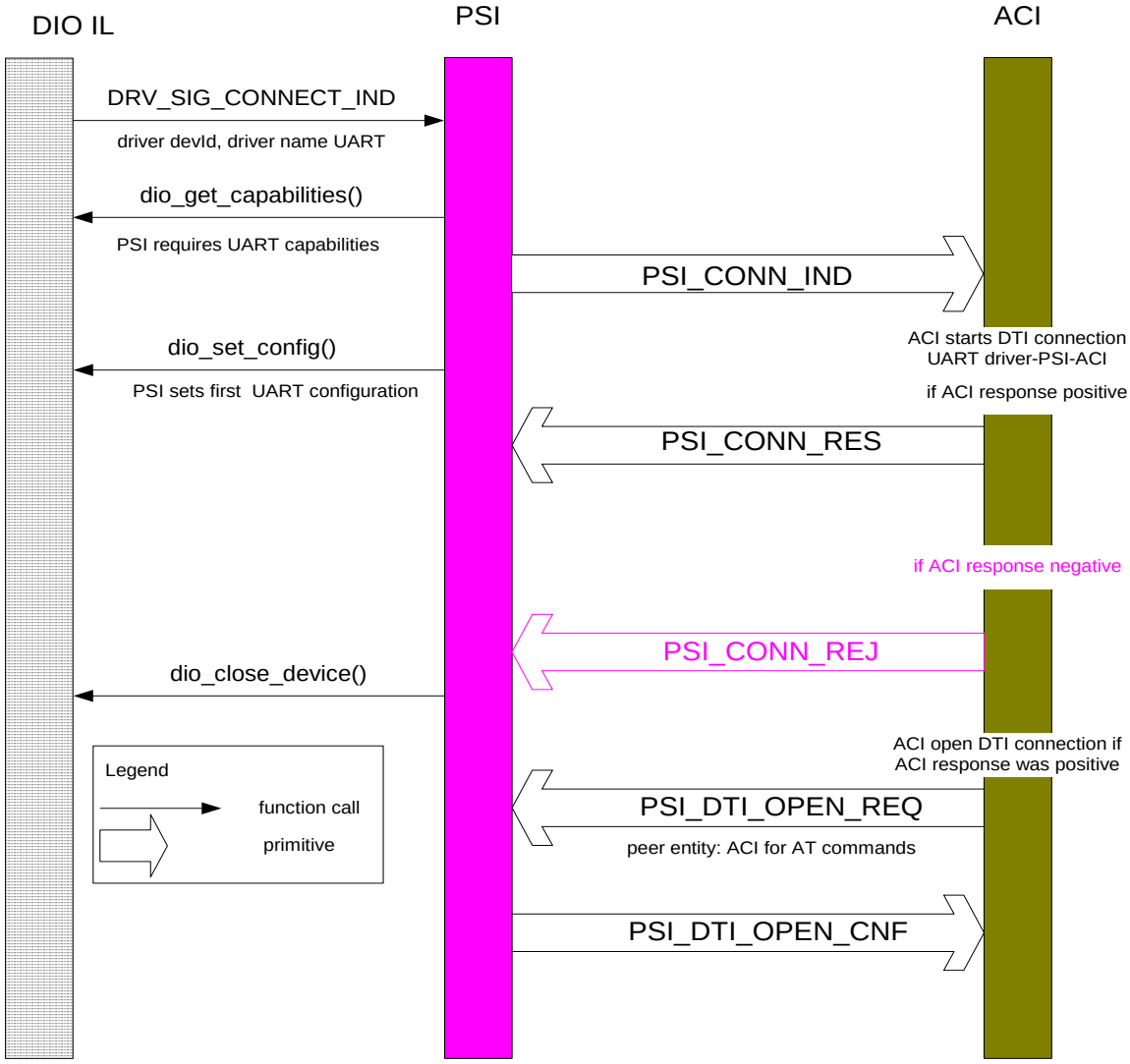

Figure 6 MSC → UART device registration in PSI and ACI

2. The DTI manager closes the existing DTI connection before it switches to a new DTI connection, i.e. UART device – PSI – L2R entity (Figure_7)

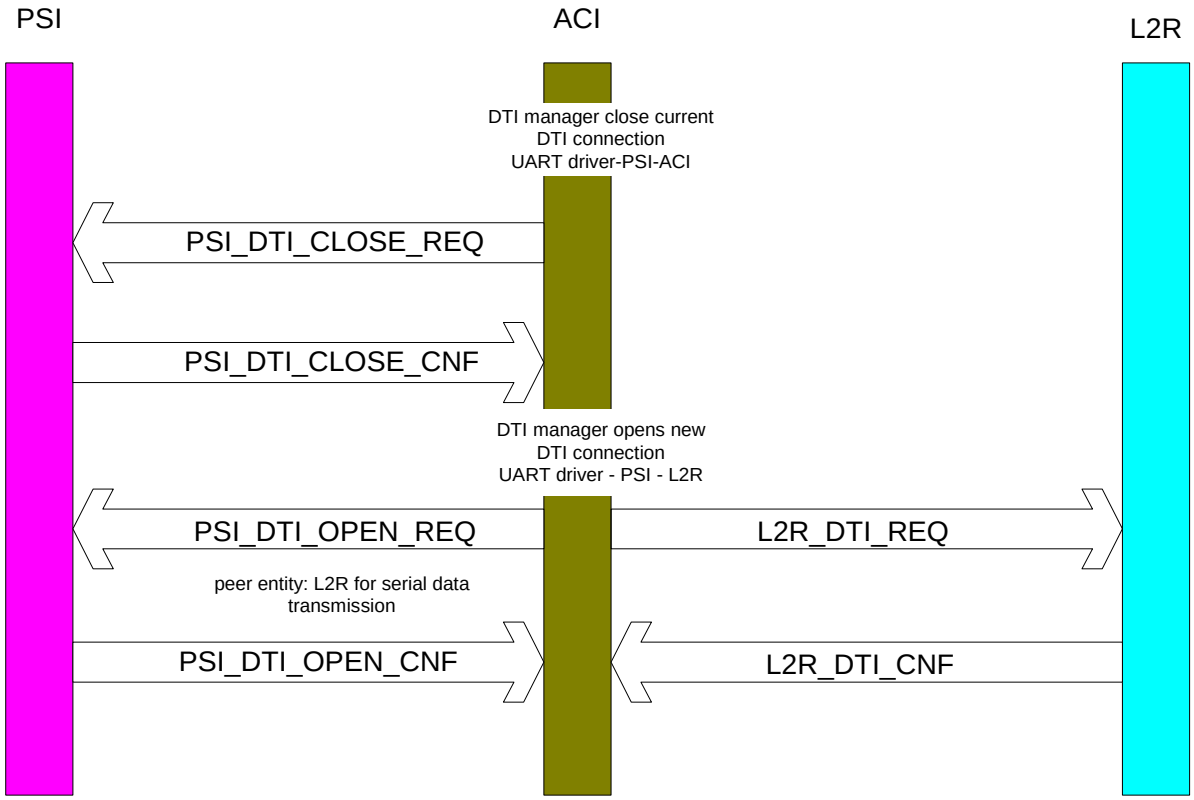

Figure 7 MSC → DTI connection close/open between PSI and ACI

2. PSI informs ACI about disconnected DTI connection i.e. if the data transmission breaks down (Figure 8).

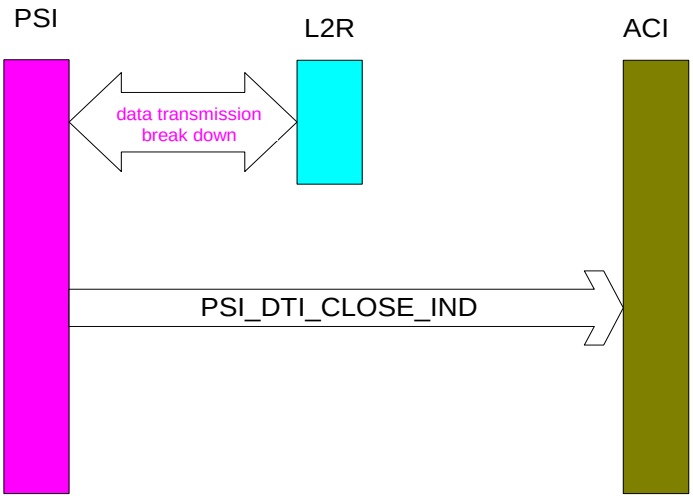

Figure 8 MSC → DTI close indication from PSI to ACI

3. If the UART device disconnects the connection PSI starts the closing of the according DTI connection and of the according device controlling in PSI and informs ACI (Figure 9.)

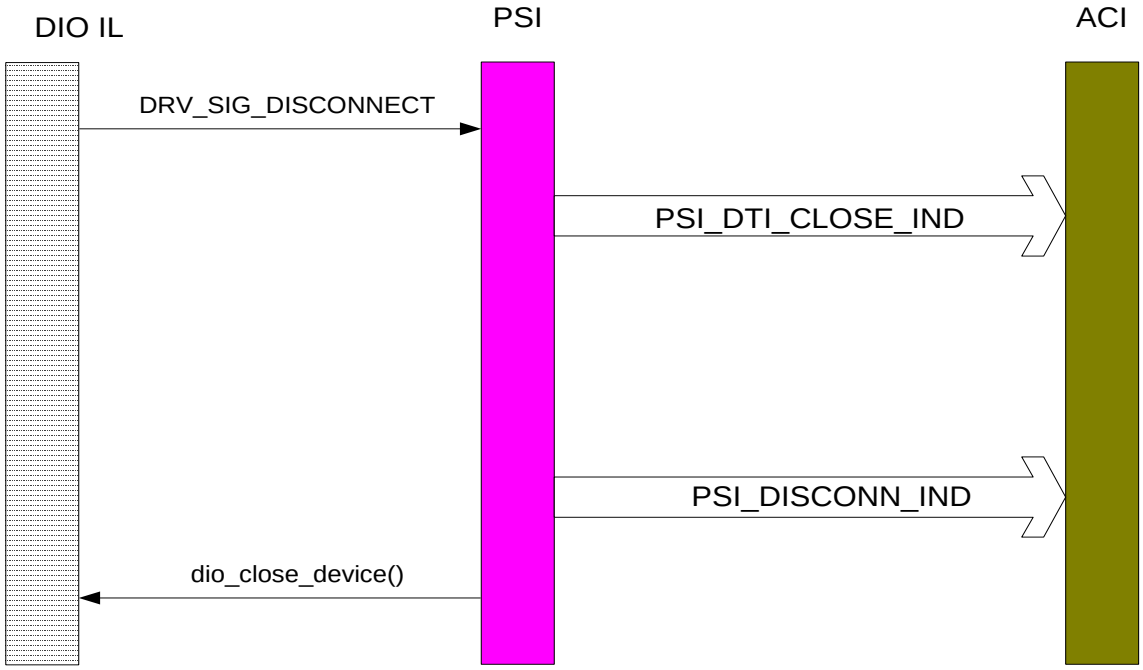

Figure 9 MSC → UART device disconnection from PSI and ACI

4. The UART configuration parameter are changeable by AT commands like AT+IPR, AT+ICF and setting of Escape Sequence Detection On/Off. It is handled in the primitive PSI_SETCONF_REQ. PSI agrees the changing by PSI_SETCONF_CNF (Figure_10) .

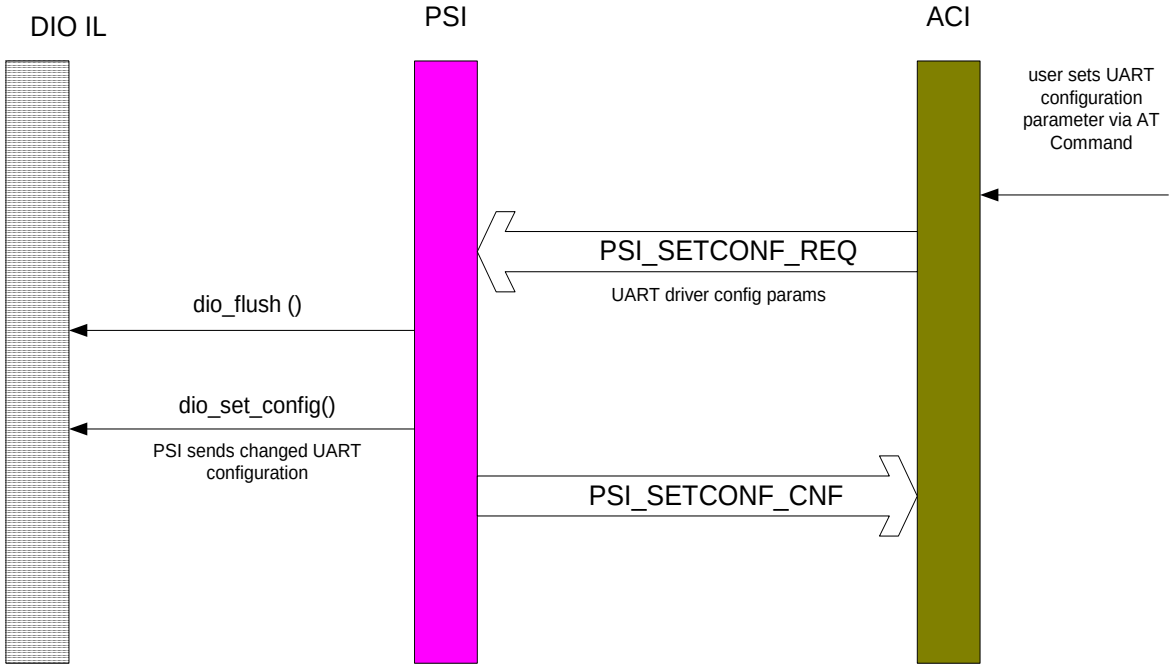

Figure 10 MSC → Updating of UART configuration

5. ACI sends the serial line states like DCD state and RING which are to be set by requirements of several PS entities and AT commands in primitive PSI_LINE_STAT_REQ. PSI agrees with the primitive PSI_LINE_STATE_CNF (Figure_11).

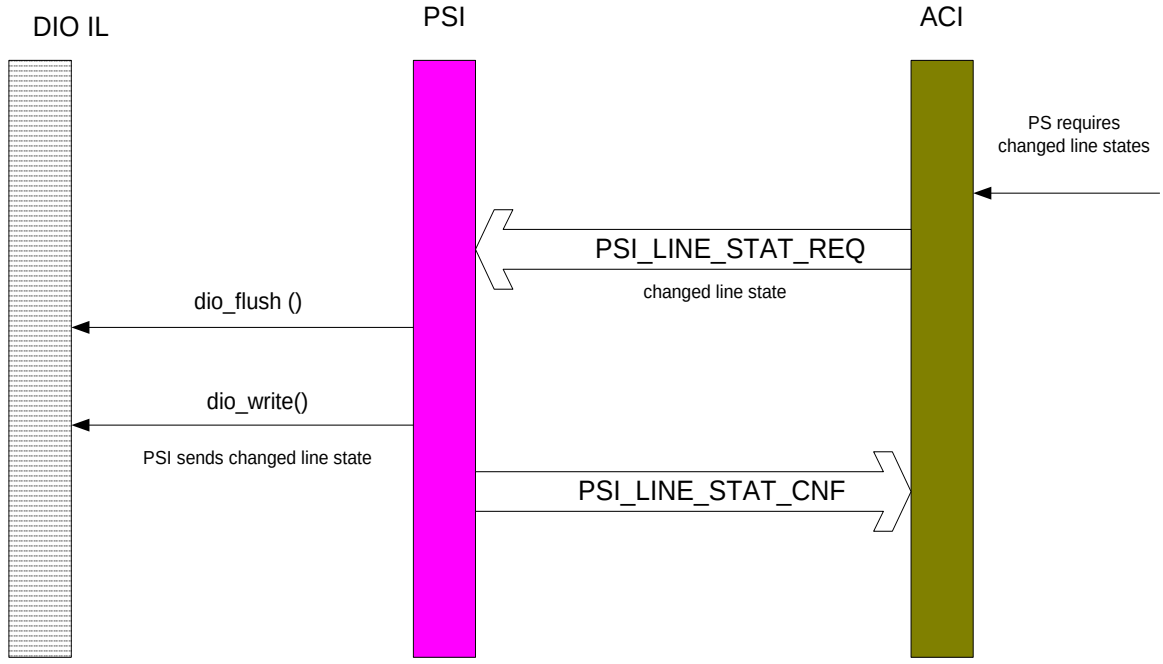

Figure 11 MSC → Changing of UART line states

6. The UART device indicates Escape Sequence Detection and DTR line drop. PSI sends these indications in primitive PSI_LINE_STAT_IND to ACI. The according DTI connection is closed (Figure_12)..

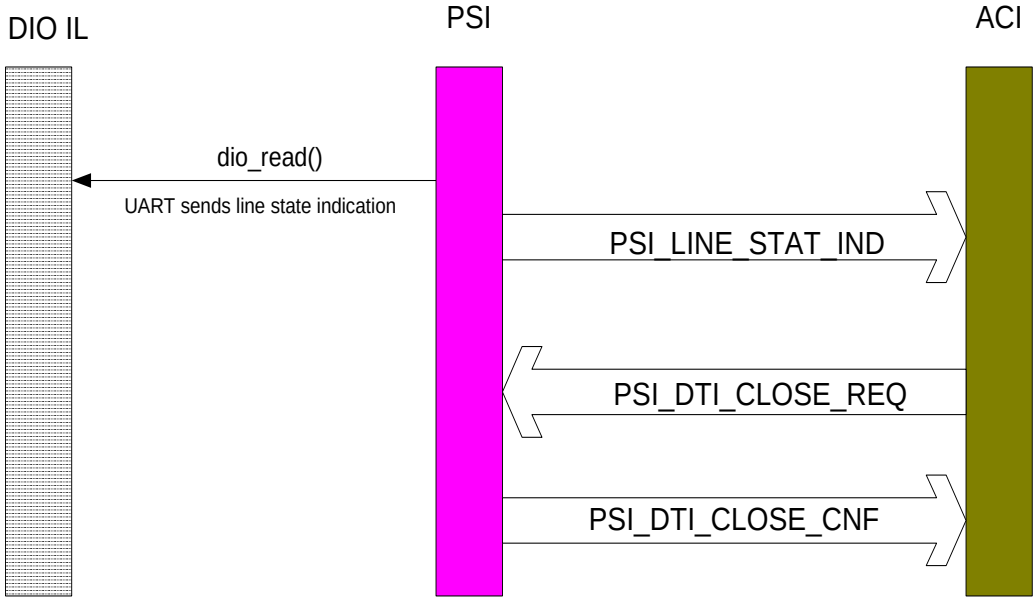

Figure 12 MSC → UART line states indication

7. If the user wants to close the UART device connection ACI sends this requirement in primitive PSI_CLOSE_REQ to PSI. After closing the DIO connection PSI confirms with PSI_CLOSE_CNF (Figure_13).

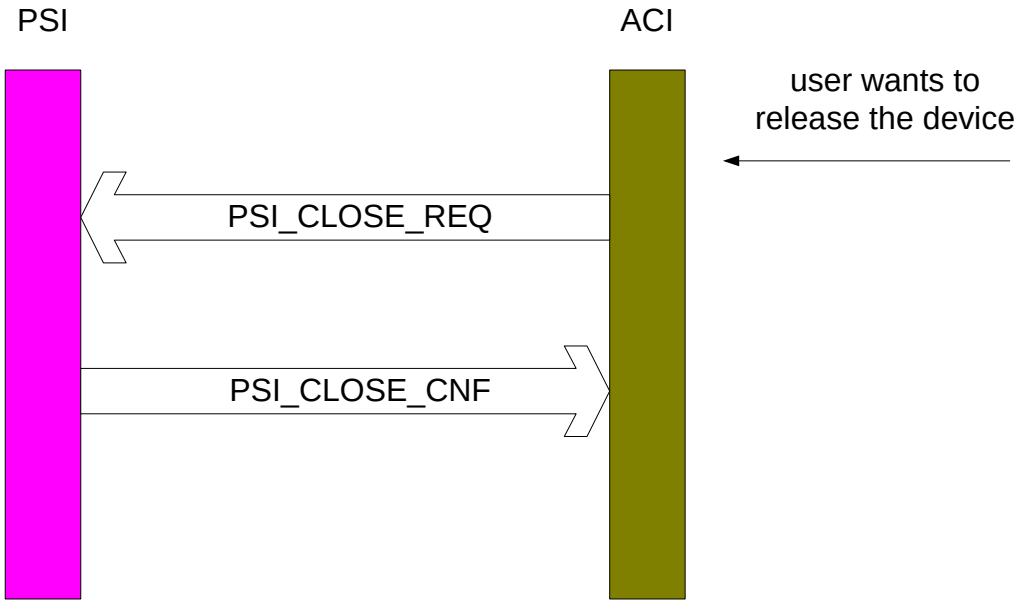

Figure 13 MSC → UART device close request by ACI

3.2.2.2.2 *MUX mode*

1. Precondition is a registered device which is able to work in MUX mode. The device is handled as described from Figure 6 to Figure_13 and works first in serial mode (the parameter *device_ mode* of structure T_DIO_DCB_SER_MUX is set on DIO_MODE_SER). The driver name is UART. The device capabilities contain serial and MUX parameters.

2. The MUX feature is activated by AT+CMUX. Before the switching from serial in MUX mode ACI checks if data are still buffered in ACI. If all of buffered data are sent ACI disconnects the DTI connection and sends the MUX configuration parameter in primitive PSI_SETCONF_REQ to PSI (the parameter *device_ mode* of structure T_DIO_DCB_SER_MUX has to be set on DIO_MODE_MUX) (Figure_14) .

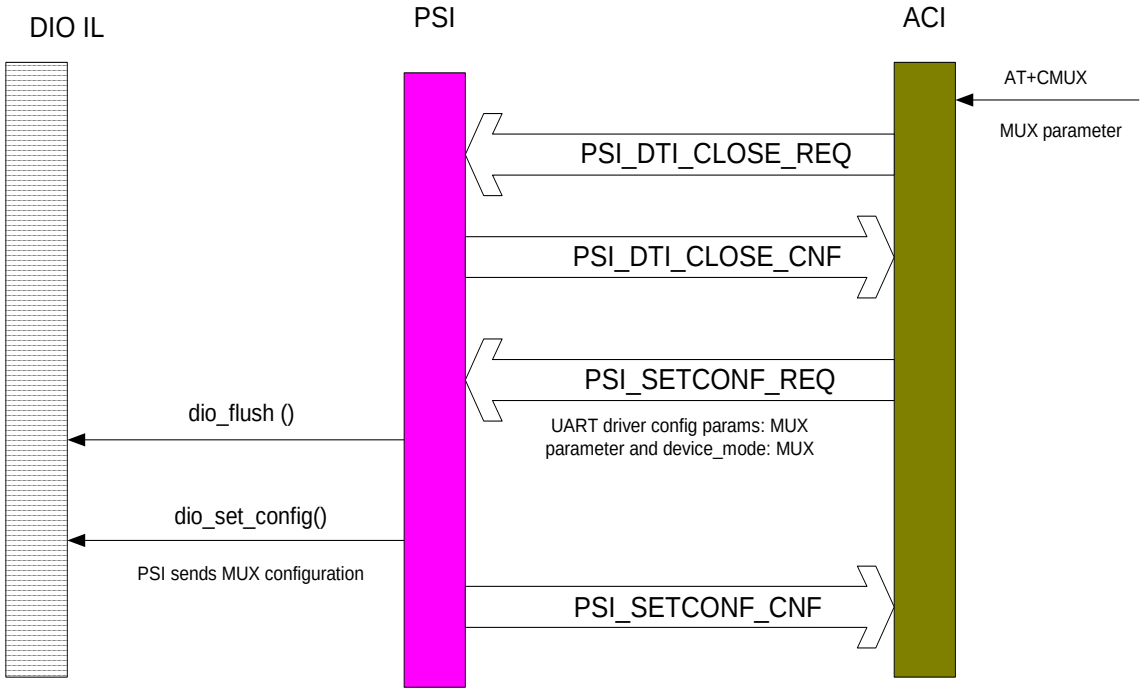

Figure 14 MSC → MUX configuration by ACI and PSI

The multiplex data channels are registered in the same way as described from Figure 6 to Figure_13. They are handled as usual serial data channels. The driver name is MUX.

3. The multiplex data channel release procedure is the same like in (Figure_9)

4. If the multiplex control channel does not work any longer in MUX mode, PSI informs ACI in primitive PSI_LINE_STAT_IND. ACI starts a new DTI connection (Figure_15).

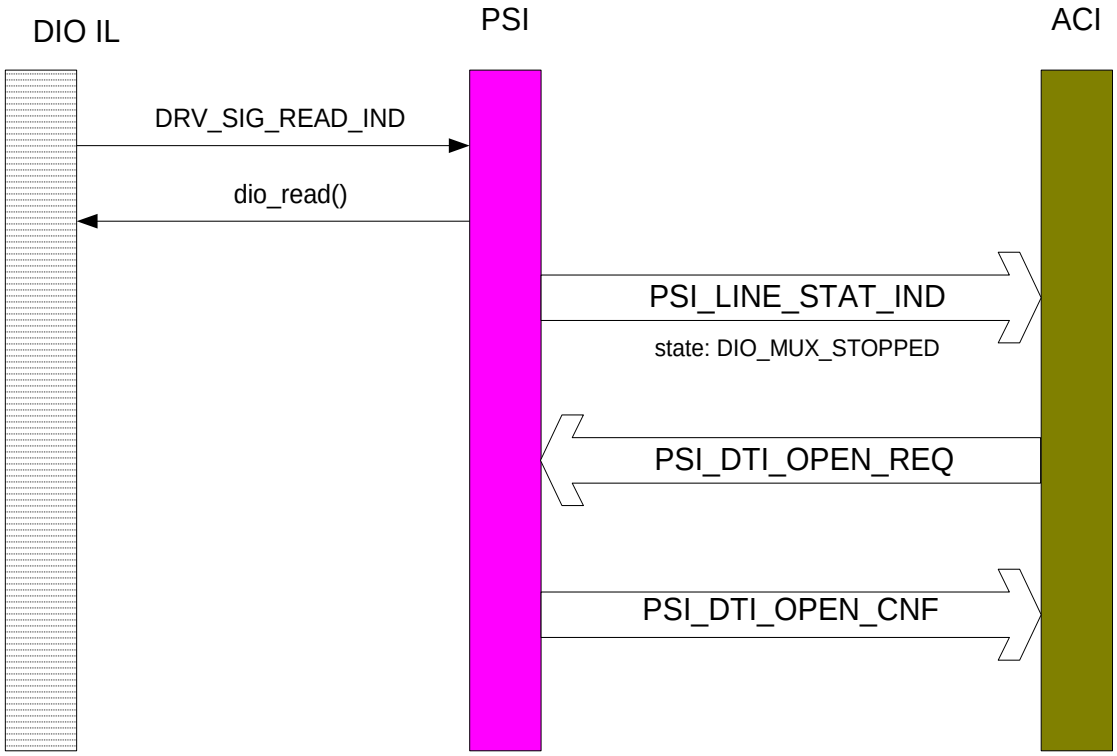

Figure 15 MSC → MUX release by UART/MUX driver

3.2.2.3 Handling specific UART AT commands and additional enhancements

AT+CMUX:

The functions sAT_PlusCMUX (cmh_uarts.c) and cmh_sig_dti_tx_buffer_ready_ind (sap_dti.c) are to be changed with respect to the MUX handling of PSI. Query of MUX settings are not supported currently.

AT+IPR:

It is not necessary to change the process of this command. The structure uart_ipr_param [] should be renamed by a more common name, the same for the function cmhUART_init_ipr_params ().

AT+ICF:

Currently ACI doesn't send OK after changing of character framing. For this reason it is necessary to extend the function psa_psi_setconf_cnf ().

AT+IFC:

Currently ACI doesn't send OK after changing of flow control (V25 ter). For this reason it is necessary to extend the function psa_psi_setconf_cnf ().

AT+FLO:

Currently ACI doesn't send OK after changing of flow control (T.32). For this reason it is necessary to extend the function psa_psi_setconf_cnf ().

Further open issues:

UART constants for **VAL parity** (defined in UART SAP): has to be defined in ACI and renamed. The DIO SAP does not support proper parity values but the supported parity is part of character frame constants.

UART constants for **VAL_nsb** (defined in UART SAP): has to be defined in ACI and renamed. The DIO SAP does not support proper stop bit values but the supported number of stop bits is part of character frame constants.

UART constants for **VAL_bpc** (defined in UART SAP): has to be defined in ACI and renamed. The DIO SAP does not support proper bits per character values but the supported number of bits per character is part of character frame constants.

The parameter **dlci** is not longer used because each multiplexed serial channel has its own device identifier.

The UART SAP constant **UART_DLCI_NOT_MULTIPLEXED** has to be replaced by new constant and to be defined in ACI for backward compatibility.

Since the UART constants **UART_DTI_FLOW_OFF**, **UART_DTI_SB_BIT**, **UART_DTI_SA_BIT** and **UART_RING_RUNNING** (defined in psa_uart.h) are not more used all functions processing these constants have to be adapted to the PSI constants PSI_DTI_FLOW_OFF, PSI_DTI_SB_BIT, PSI_DTI_SA_BIT and PSI_RING_RUNNING (defined in psa_psi.h).

The function **cmhUARTtest_srcId** *()* has to be replaced by cmhPSItest_srcId () completely (T30).

Minor issues:

GPRS connection: parameter of T_GPRS_CONT_CLASS **link_uart_id** is to be renamed with link_psi_id.

The parameter **uart_conn** of global structure T_L2R_SHRD_PRM is to be renamed with psi_conn.

The global used structure T_ACI_AT_CMD **uartEntcurCmd [CMD_SRC_MAX]** is to be renamed with psiEntcurCmd [CMD_SRC_MAX].

The function **PPP_UART_connect_dti_cb ()** is to be renamed with PPP_PSI_connect_dti_cb ().

The function **uart_is_mt_te_link ()** is to be renamed with psi_is_mt_te_link ().

3.2.2.4 UART/MUX simulation tests

The primitives and structures of the PSI SAP have to replace the current used of UART SAP in the UART/MUX simulation tests of ACI entity. It concerns ACI_DTI, ACI_FD, ACI_SS, GACI and ASC.

4 PKTIO entity

4.1 Current Integration of PKT driver

The following interface for packet data transmission via the entity PKTIO is currently implemented.

The PKTIO entity is able to communicate via DIO Interface, v4.

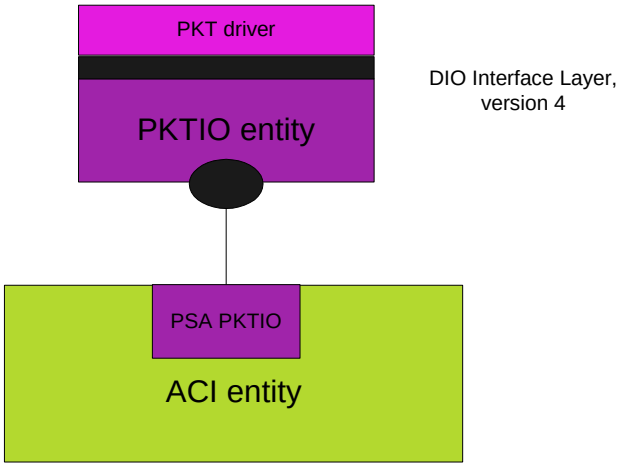

Figure 16 Current PKT driver / PKTIO entity configuration

4.2 New Integration of PKTIO driver

The PSI entity is able to support packet data transmission. The following scheme shows the new entity configuration.

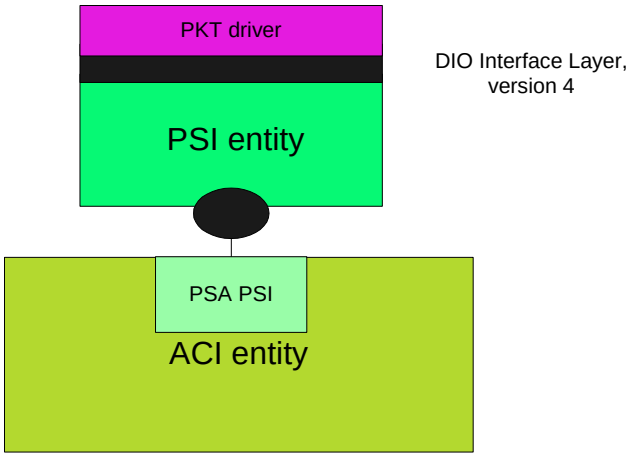

Figure 17 New configuration regarding PKT driver and PSI entity

4.2.1 Enhancement in PSI

1. Handling of packet device capability and packet configuration parameter
2. Enhancement of buffer handling regarding IP protocol field
3. Testing like described in Chap_4223
   For the packet driver simulation enhancements of the simulation functions described above are necessary
4. Sending of default configuration of PKT driver read from FFS.
5. Testing: several PSI instances; handling more than one provided buffer segment

4.2.2 Adaptation in ACI

4.2.2.1 Replace PKTIO SAP by PSI SAP

The PKT primitives are be covered by the PSI SAP.

4.2.2.2 Enhancement PSI part in ACI by handling device type PKT

The PKTIO list *pktio_dev_list* is covered by the existing list *psi_dev_list*.

1. The PKT device registration uses the same primitive as for the USB device in the PSI part of ACI but this device is not treated as ACI source. PSI sends the first PKT device configuration to the device (instead of ACI)(Figure_18).

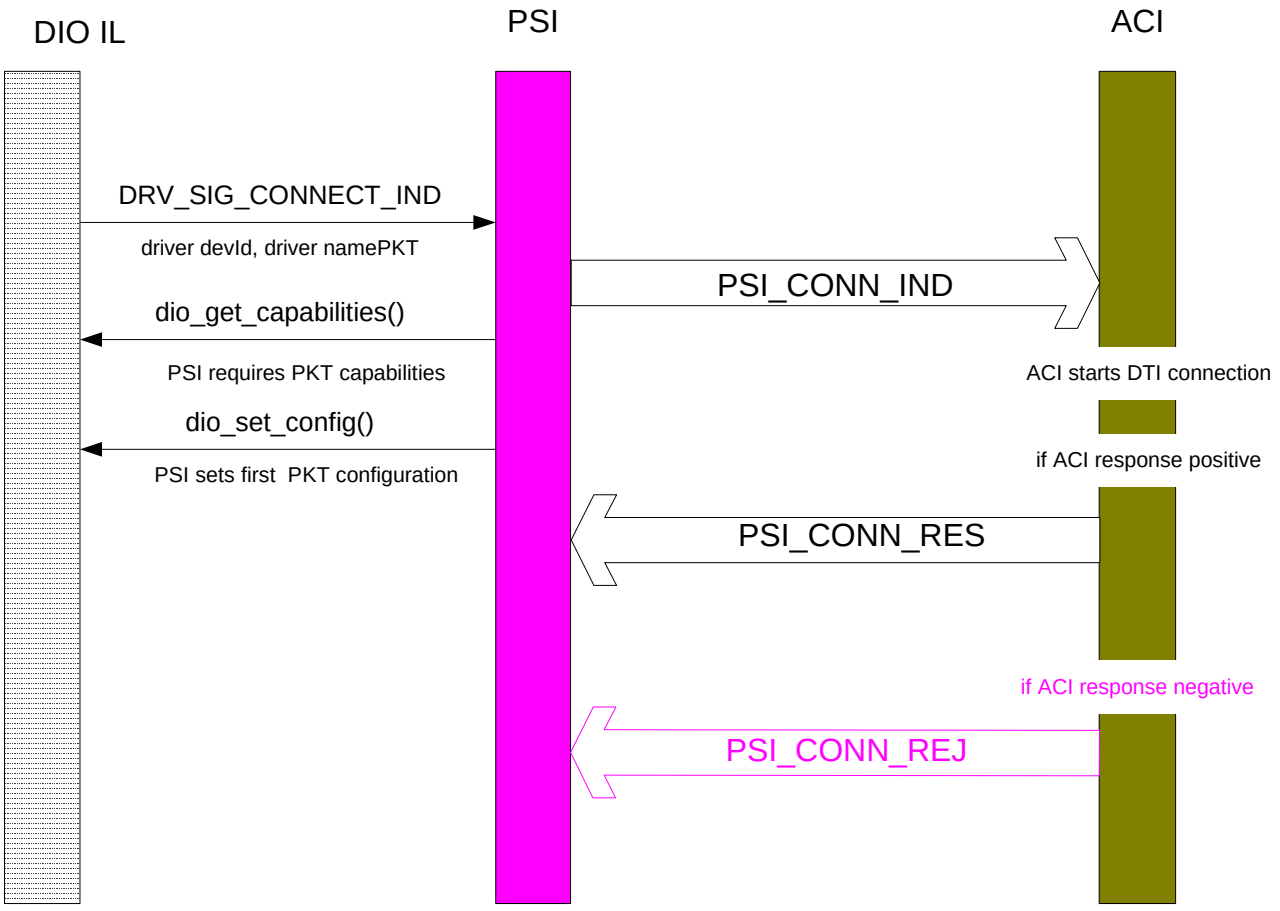

Figure 18 MSC → PKT device registration in PSI and ACI

2. The user must redirect the data flow via the AT command %DATA because no AT commands (ATD, AT+CGDATA) can received on the PKTIO channel. In the GPRS Call set up phase the DTI manager initiates the needed DTI connection (Figure_19) .

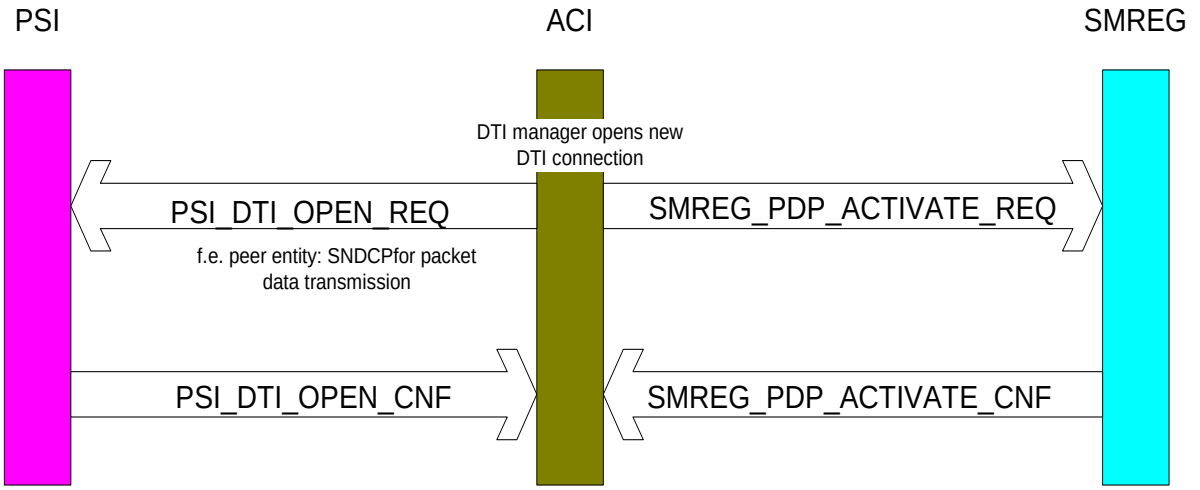

Figure 19 MSC → DTI connection open between PSI and ACI

3. PSI informs ACI about disconnected DTI connection i.e. if the data transmission breaks down (Figure_20).

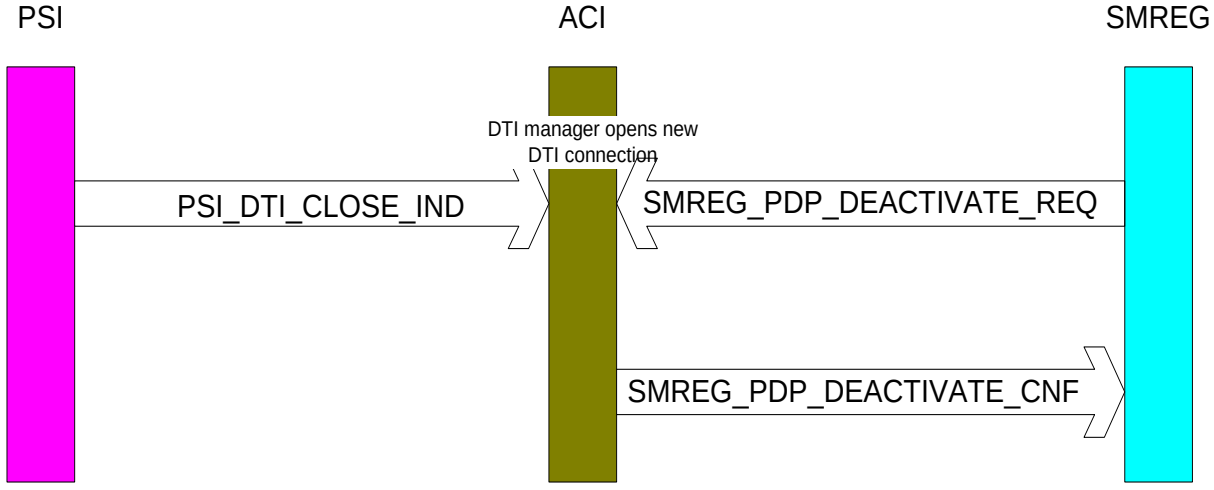

Figure 20 MSC → DTI close indication from PSI to ACI

4. If the PKT device disconnects the connection PSI starts the closing of the according DTI connection and of the according device controlling in PSI and informs ACI (Figure_21)

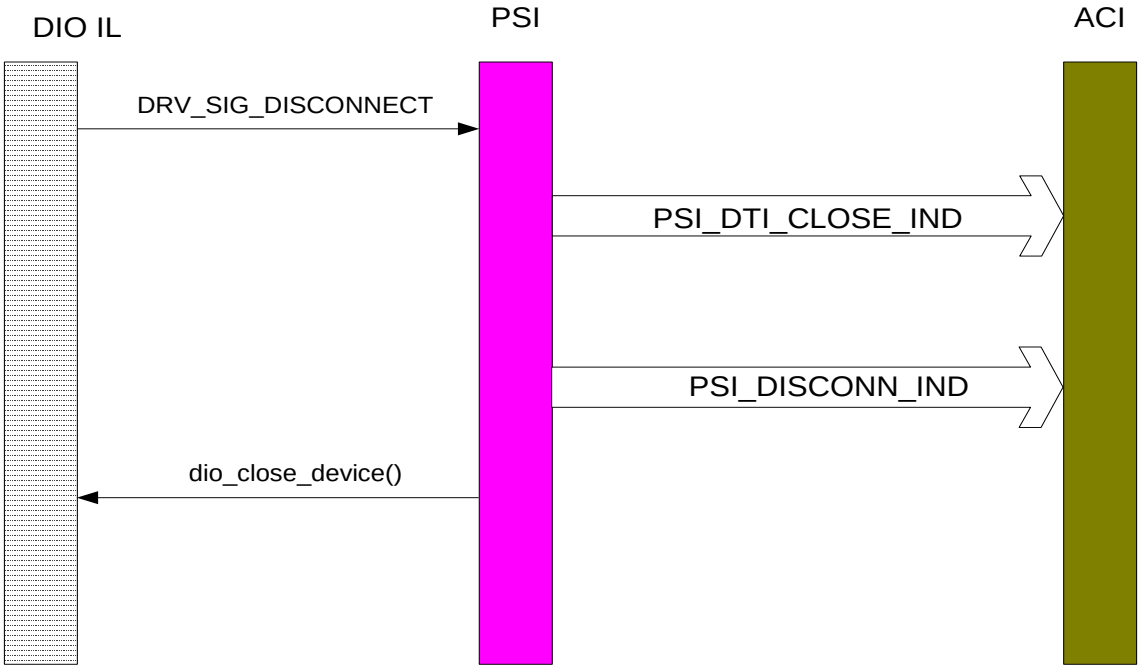

Figure 21 MSC → PKT device disconnection from PSI and ACI

5. The changeable PKT device parameters are handled in the primitive PSI_SETCONF_REQ. PSI agrees the changing by PSI_SETCONF_CNF. (Figure_22)

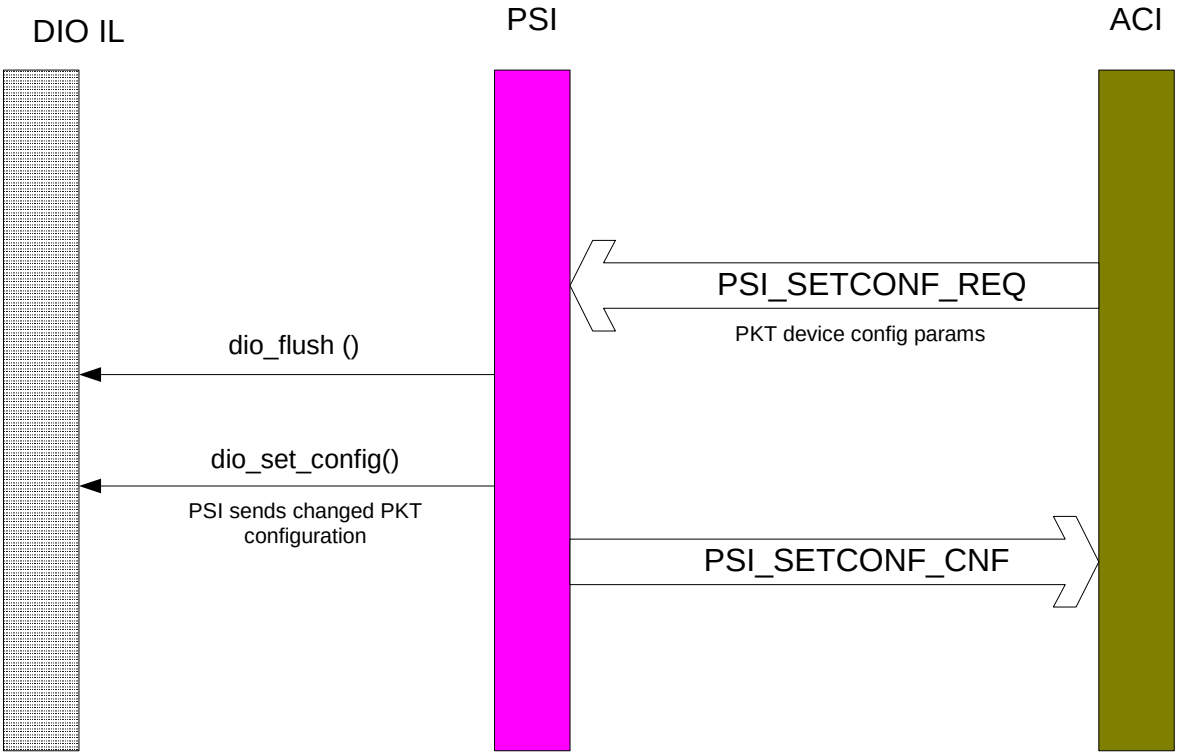

Figure 22 MSC → Updating of PKT configuration parameter

6. If the user wants to close the PKT device connection ACI sends this requirement in primitive PSI_CLOSE_REQ to PSI. PSI confirms with PSI_CLOSE_CNF (Figure_23) .

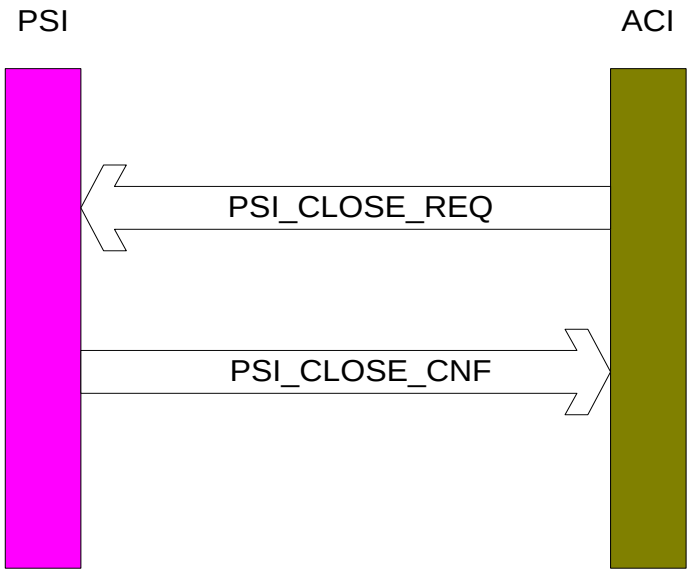

Figure 23 MSC → PKT device close request by ACI

4.2.2.3 PKTIO simulation tests

The primitives and structures of the PSI SAP have to replace the current used of PKTIO SAP in the PKTIO simulation tests of ACI entity. It concerns ACI_PKT.

5 BAT enhancement

The BAT library offers a functional interface for execution of AT commands. Several mobile phone configurations (such as one-processor, two-processor, and hybrid execution environments) must be taken into consideration. From a general and high-level point of view, the application is running in the so-called *application domain*, and sends binary AT commands via the BAT library to the protocol stack (PS) running in the so-called *modem domain*. The BAT library uses a *communication channel* between the two domains to transfer information about the transmitted AT commands and their responses.
The BAT Adapter is a software component situated in the application-domain. It connects the BAT library to PSI. In this scenario, the application and the protocol stack communicate via the protocol stack interface (PSI) entity, the entry point to the protocol stack executing AT commands. The main task of the BAT adapter is to decouple the BAT lib from DIOv4 being communication medium to transfer data to PSI. It also provides a context switch between PSI and the application.

5.1 New Integration of BAT driver

The data transmitted between application and ACI are formated as packets. The following scheme shows the new entity configuration.

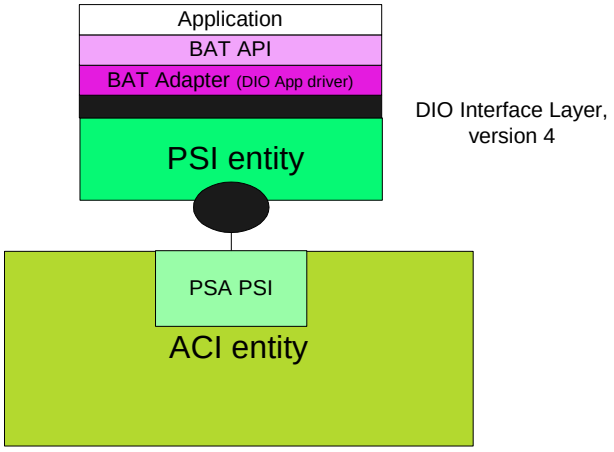

Figure 24 New configuration regarding BAT driver and PSI entity

5.1.1 Enhancement in PSI

1. Handling of packet device capability and packet configuration parameter
2. Enhancement of buffer handling regarding IP protocol field
3. Testing like described in Chap_5121 .
   For the packet driver simulation additional enhancements of the simulation functions (described above) are necessary
4. Sending of default configuration of BAT driver read from FFS.
5. Testing: several PSI instances; handling more than one provided buffer segment

5.1.2 MCS Communication BAT driver - PSI

1. The BAT device registration uses the same primitive as for the USB device in the PSI part of ACI. PSI sends the first BAT device configuration to the device (instead of ACI) (Figure_25).

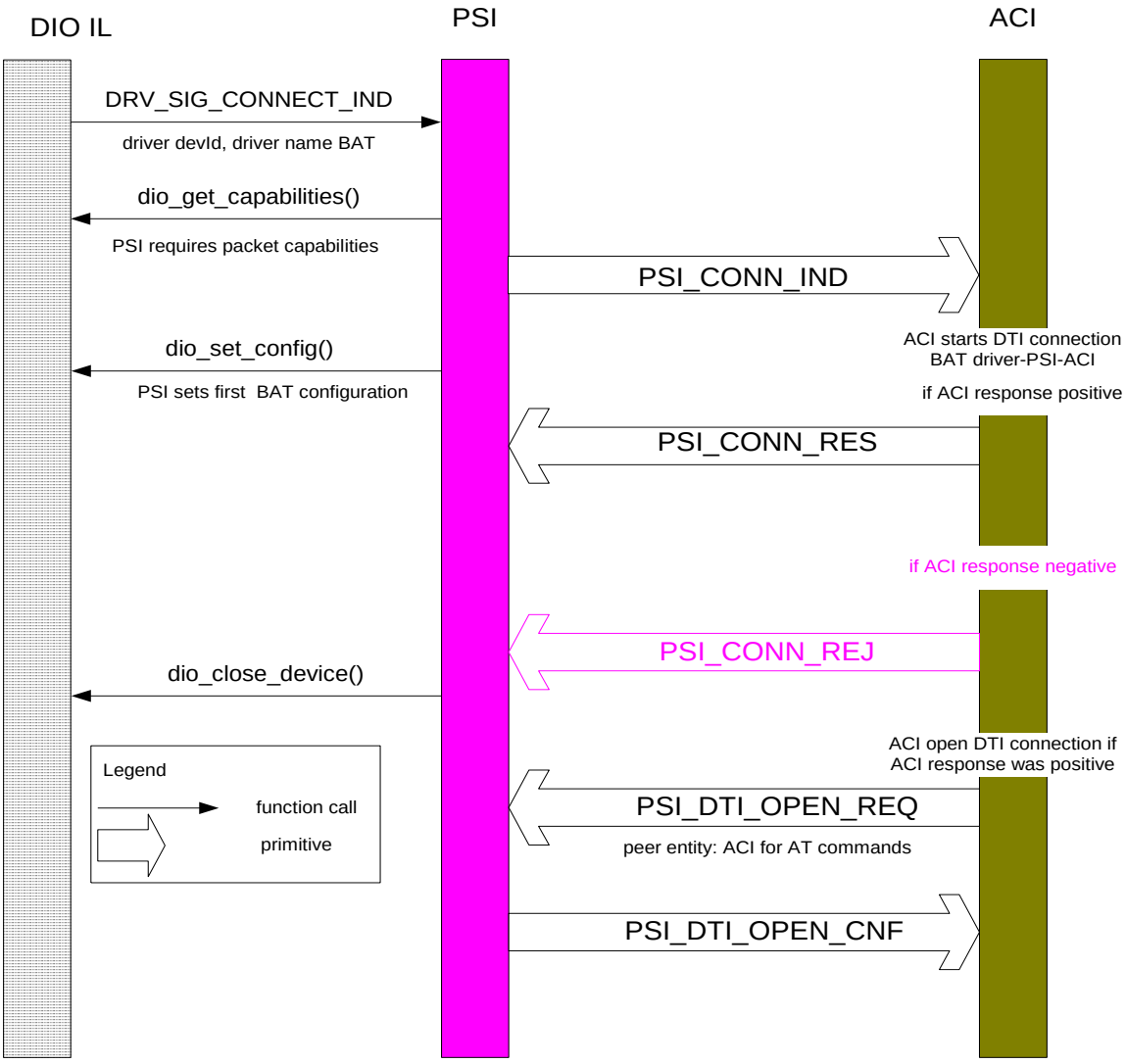

Figure 25 MSC → BAT device registration in PSI and ACI

2. PSI informs ACI about disconnected DTI connection i.e. if the data transmission breaks down (Figure_26).

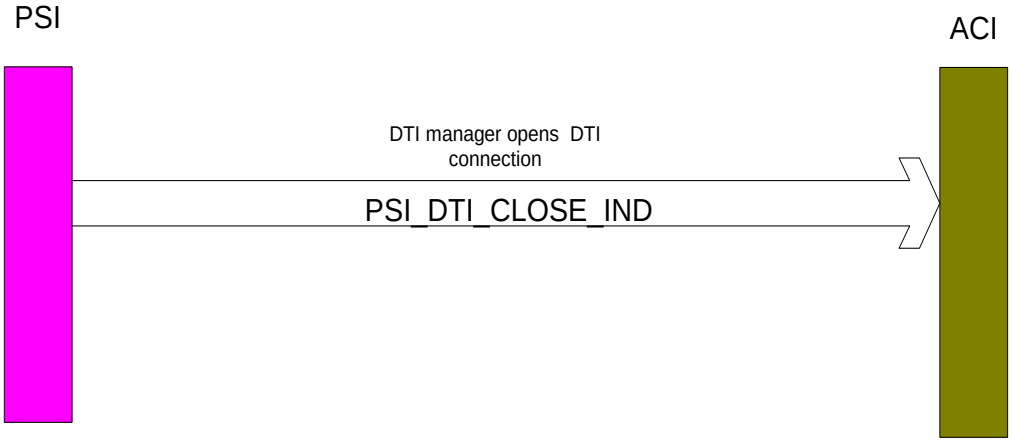

Figure 26 MSC → DTI close indication from PSI to ACI

3. If the BAT device disconnects the connection PSI starts the closing of the according DTI connection and of the according device controlling in PSI and informs ACI (Figure_27)

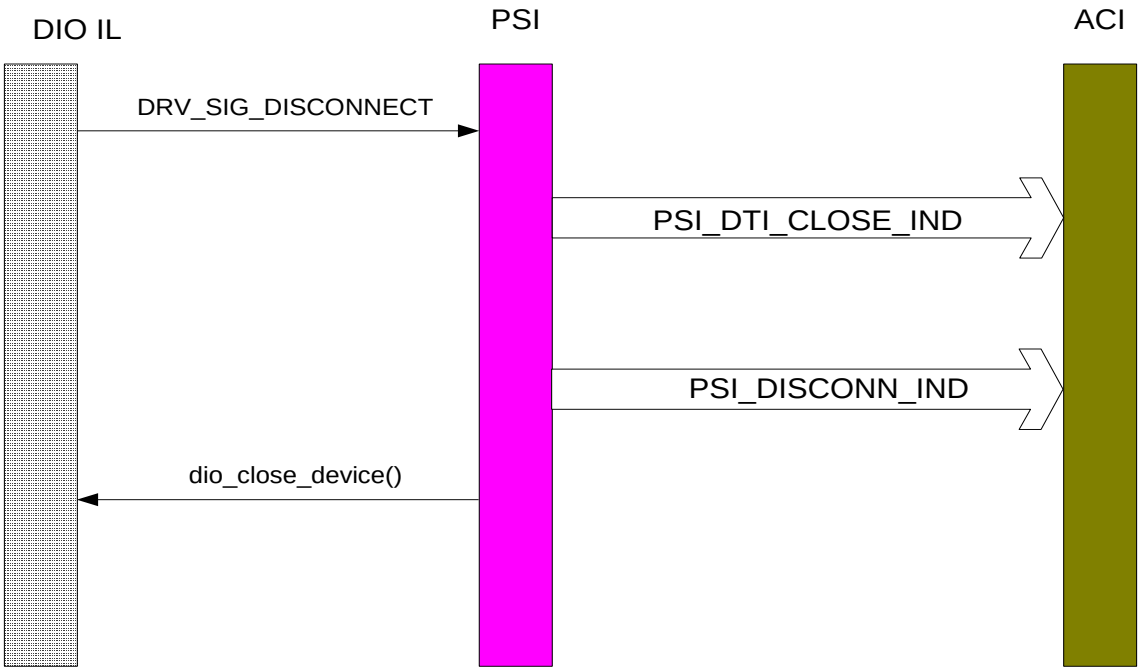

Figure 27 MSC → BAT device disconnection from PSI and ACI

7. If the user wants to close the BAT device connection ACI sends this requirement in primitive PSI_CLOSE_REQ to PSI. PSI confirms with PSI_CLOSE_CNF(Figure_28).

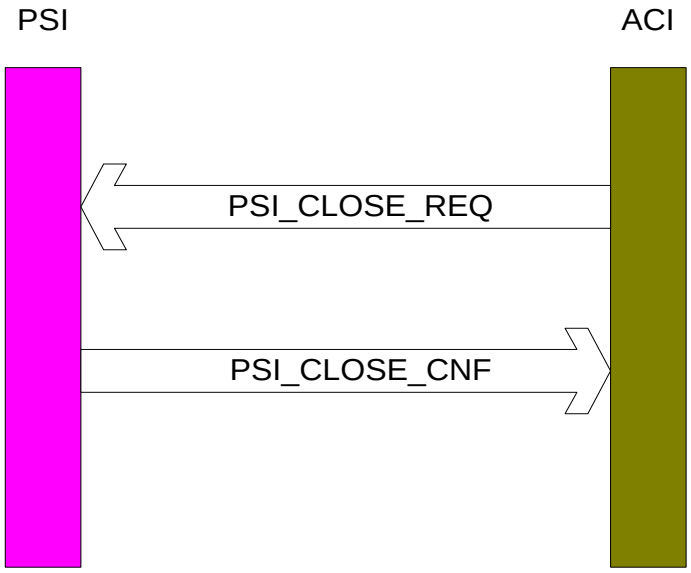

Figure 28 MSC → BAT device close request by ACI

5.1.2.1 BAT simulation tests

The existing PSI simulation tests of the entity PSI are to be enhanced for packet data.

Appendices

A. Acronyms

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

B. 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/>