Multimode Data Services

80-V6123-1 K

Confidential and Proprietary – Qualcomm Technologies, Inc.

Qualcomm Technologies, Inc.
5775 Morehouse Drive
San Diego, CA 92121
U.S.A.

Revision History

Revision	Date	Description
A	Mar 2003	Initial release
B	May 2003	Updated for editorial changes
C	Jun 2003	Added information on 1xEV-DO, added Appendix B, updated other information as noted by change bars
D	Jul 2003	Updated for engineering updates
E	Dec 2003	Updated for HDR-CDMA information and CDMA Voice Tuneaway information
F	Mar 2004	Updated UMTS information, WCDMA and GSM CS Data tasks, UTS and WCDMA CS file structures, and Packet Data Bearer services
G	Jul 2004	Updated WCDMA PS RAB configurations
H	Dec 2004	Updated for engineering changes
J	Feb 2005	Added supported CS Data Configurations table; note: there is no Rev. I per Mil. standards.
K	Feb 2013	Updated [Q2] to [Q3]

Contents

- Common Data
- Memory Management
- Applications Interface
- cdma2000
- 1xEV-DO
- UMTS
- Appendix A – IS-707 Async and 1X Packet Call Flows
- Appendix B – HDR Call Flows
- References
- Questions?

Common Data

Common Data

- Multimode
- Protocols (PS DEV)

Multimode Data Architecture

- Allows different technologies (GSM, WCDMA, 1X) to exist within the same code base
- Allows other technologies (802.11, Bluetooth™ (BT)) to more easily be added to this framework
- The technologies become plug-and-play – No technology relies on another technology to be present (or absent)
- Protocols (PPP, IP, UDP, TCP, etc.) are now independent of the underlying technology
- Allows for code to not be duplicated independently for each technology – Solves the problem only once

Example

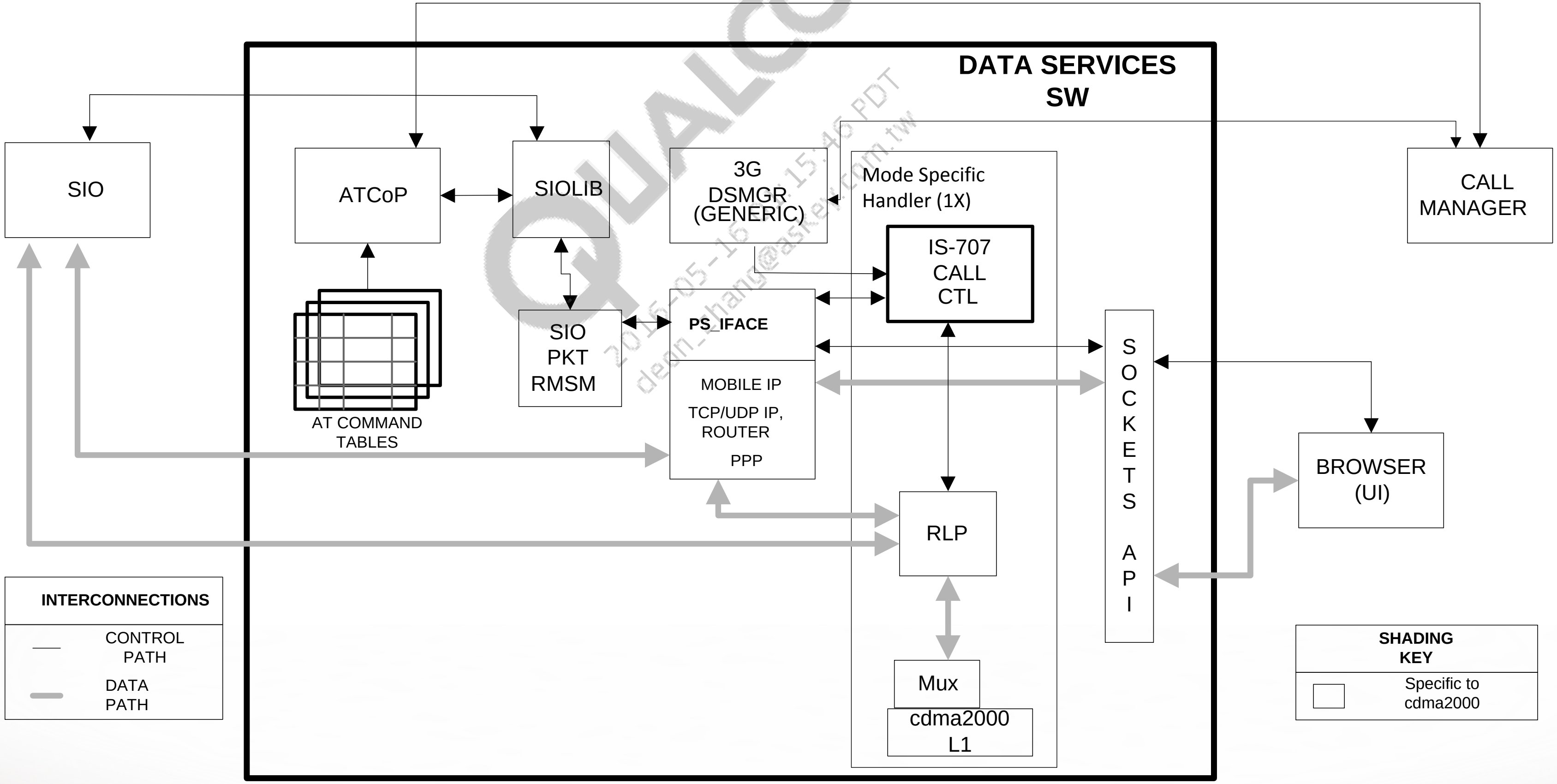
- Sockets
  - Originally developed for CDMA systems
  - Protocols code is tightly coupled with the CDMA link layer
  - New architecture abstracts sockets away from the underlying technology
- Now
  - Same protocols and sockets code base are used for 1X, GSM, WCDMA, and GPRS
  - Same code base will be used for BT, 802.11, and other potential technologies
- Customer benefit – A sockets application that works in WCDMA will also work in 1X (no changes needed in the application code)

Main Components

- 3G ATCoP – AT command processor
- 3G SIOLIB – API for SIO (UART, USB, BT)
- 3GDSMgr – Common data call control element
- PS IFACE – Generic interface to IP
- Mode-specific handlers – Call control code is specific to a particular technology
- Multimode PS – Common protocols (PPP, IP, TCP, UDP, PAP, CHAP, and sockets)

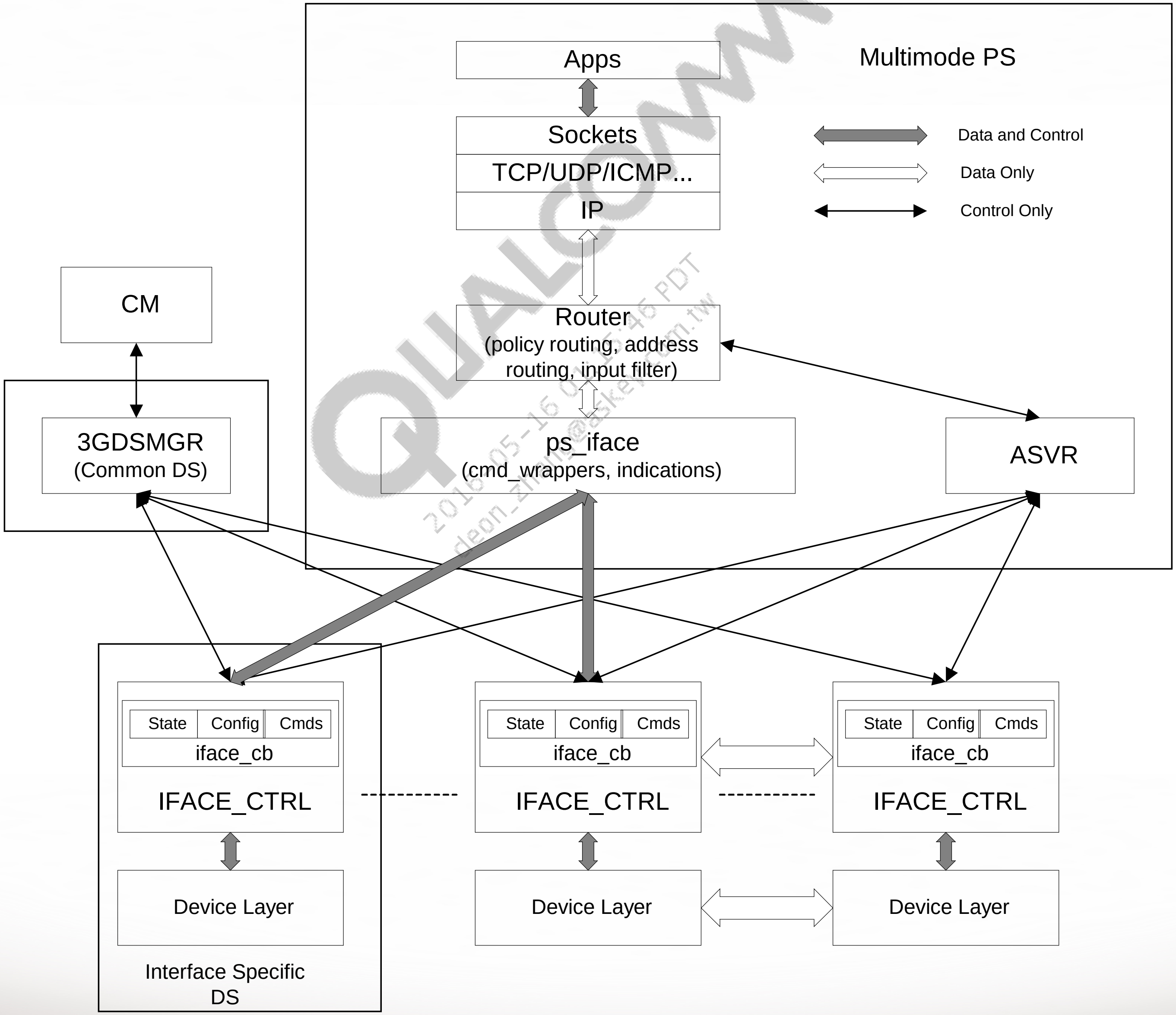
Architecture – Data Path

- A 1X mode-specific handler is shown as an example. There could be *n* other simultaneous handlers present, including ones for GSM, GPRS, WCDMA, 802.11, etc.



Interconnections between entities outside the Data Services SW are not shown.

Architecture – Call Control



Architecture – Main Points

- 3G DSMGR
  - Provides the sole interface to CM for call handling and control (call origination, termination, etc.)
  - Implements the generic parts of every call
    - Origination/incoming call
    - Call connection
    - Call end/hang up
    - Other parts
  - Calls the mode-specific handlers to execute the technology-specific parts of a call
  - Will eventually be able to handle multiple calls in multiple technologies simultaneously
  - Supports laptop and sockets calls; will eventually support simultaneous laptop and sockets calls
- ATCoP
  - Each mode-specific handler registers callbacks for common AT commands (ATDT)
  - Each mode-specific handler registers tables for technology-specific AT commands (AT$QCMDR)

Architecture – Main Points (cont.)

- 3G SIOLib
  - Provides an interface to SIO drivers (UART/USB/BT)
  - Has some management of AT command processing, i.e., Command mode, Online Data mode, and Offline Data mode
- PS interface
  - Each link layer that is capable of transporting IP packets is an interface.
  - Each interface is controlled by an interface controller (IFACE_CTRL). This is also referred to as the mode-specific handler.
  - Each IFACE_CTRL manages one or more interfaces.
  - IP packets are routed from one interface to another, or between the protocol stack and interfaces.
- Mode-specific handlers
  - Provide technology-specific call control
  - Register with 3G DSMGR for call control events/actions
  - Register with ATCoP for AT command processing
  - Mode-specific handlers are independent of each other

Architecture – Main Points (cont.)

- Sockets application example
  1. The Sockets application requests a connection.
  2. The router finds the interface (PS_IFACE) that will provide the connection.
  3. The Sockets layer tells PS_IFACE to make a connection.
  4. The mode-specific handler associated with that PS_IFACE initiates a call through 3G DSMGR.
  5. 3G DSMGR communicates the origination to CM.
  6. CM returns the origination status to 3G DSMGR, which is forwarded to the mode-specific handler.
  7. The mode-specific handler sets the PS_IFACE to indicate a connection (in the case of a success).
  8. The sockets react to this and initiate the protocols to come up (using multimode PS).
  9. Once the protocols are up, the sockets inform the application that it can proceed.

File List

- Data Services files are grouped logically as follows:
  - Data Services Task
  - Data Services Utility
  - 3G ATCoP – Filenames are dsat*.[c,h]
  - 3G DS Manager – Most filenames are ds3g*.[c,h]
  - 3G SIO library – Most filenames are ds3g*.[c,h]

SIOLIB

- A library of wrapper functions to manage and control the Rm interface
- Sets serial mode, e.g., Autodetect, rawdata, packet – ds3g_siolib_change_mode
- Configures watermarks – ds3g_siolib_setup_watermarks()
- Enables/disables flow – ds3g_siolib_set_inbound_flow()
- Controls the CD state – ds3g_siolib_set_cd_state()
- Handles DTR – ds3g_siolib_handle_dtr_changed_sig()
- Handles AT escape sequence processing (+++ detection) – ds3g_siolib_handle_sio_escape_sig()
- Sets desired baud rates – ds3g_siolib_change_baud_rate()

SIOLIB (cont.)

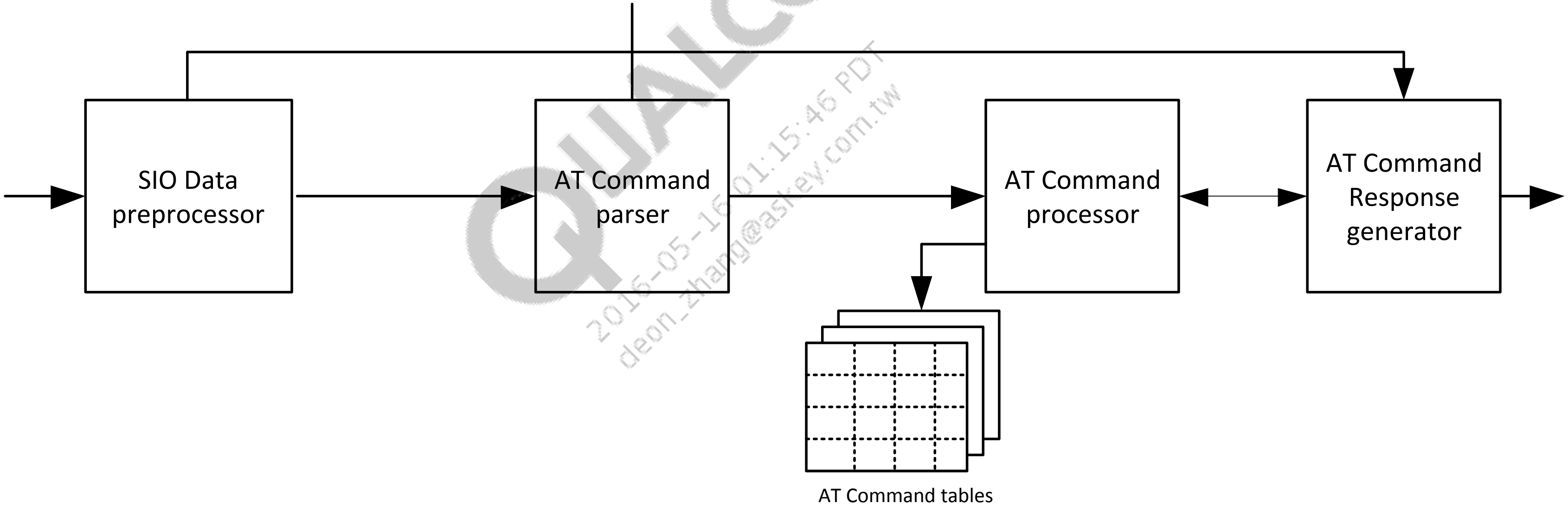
- Each mode-specific handler registers the following callbacks with SIOLIB:
  - at_escape_sequence_handler – Called when AT escape sequence is detected
  - dtr_changed_sig_handler – Called when DTR is asserted or deasserted
  - at_return_to_online_data_handler – Called when ATO is received
  - return_to_online_cmd_mode_complete_handler – Called when exiting Online Data mode and returning to Online Command mode
  - auto_answer_handler – Called when autoanswering is to be performed based on the value of ATS0

SIOLIB (cont.)

- SIOLIB also maintains the following state information:
  - serial_info – Serial port-related information
  - at_state – AT command processing state
  - func_tbl – Table of callbacks registered
  - call_coming_up – Whether ATD was received and a call is coming up
  - ring_type – Ring result code to be sent to TE
  - ring_counter – Number of rings
- SIOLIB gets notification for the following events:
  - DTR went high or low – DS_DTR_CHANGED_SIG set
  - AT detected in Autodetect mode – DS_1ST_SIO_RX_SIG
  - Packet (0x7e ~) detected in autodetect – DS_SIO_RX_PKT_SIG
  - +++ detected in raw data – DS_SIO_ESCAPE_SIG
- Relevant files
  - ds3gsiolib.c
  - ds3gsiolib.h

ATCoP Architecture

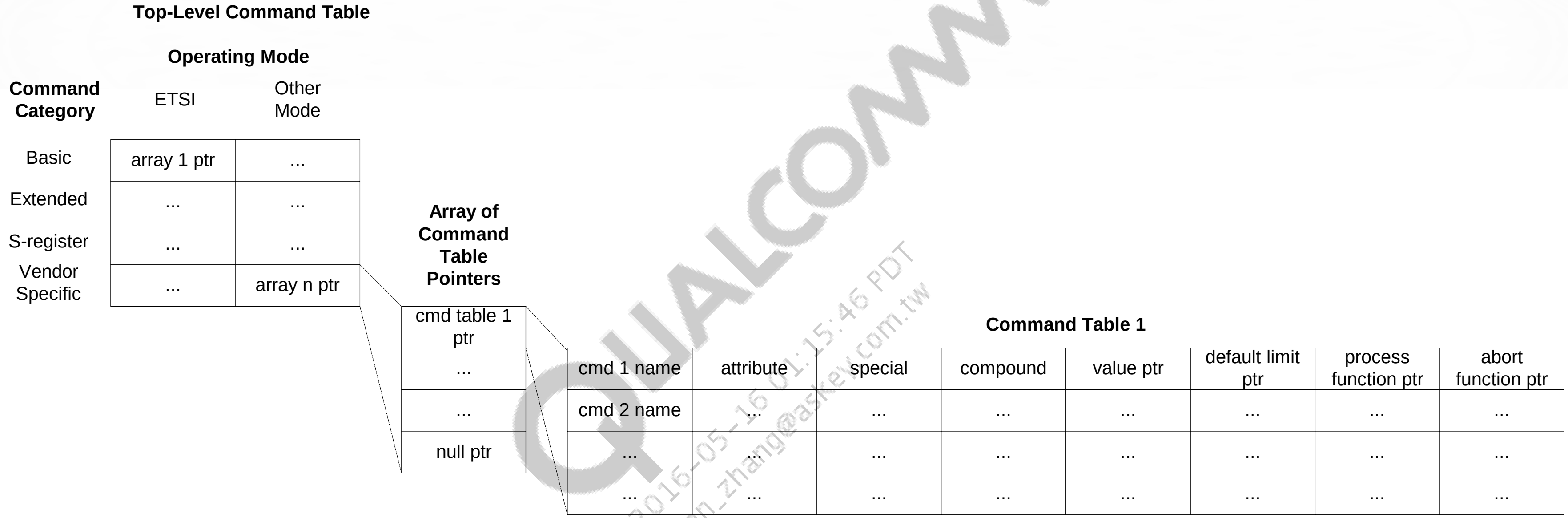
- ATCoP executes in the context of the data services task.
- Basic ATCoP architecture is shown in the graphic.
- Data is received from, and sent to, TE by means of serial drivers.



ATCoP Functional Blocks

- SIO Data preprocessor – Preprocesses data received from the SIO serial interface
  - Produces NULL-terminated command line
  - Initiates aborting of abortable commands
  - Handles Short Message Service (SMS) message entry
- AT Command parser – Parses all commands from the AT command line producing a token data structure for each command
- AT Command processor – Processes AT commands contained within each token data structure by the method of AT command table look-up
- AT Command Response generator – AT command response generating and formatting utilities

AT Command Table Structure



Top-Level Command Table

Command Category	**Operating Mode** ETSI	Other Mode
Basic	array 1 ptr	...
Extended	...	...
S-register	...	...
Vendor Specific	...	array n ptr

Array of Command Table Pointers

cmd table 1 ptr
...
...
null ptr

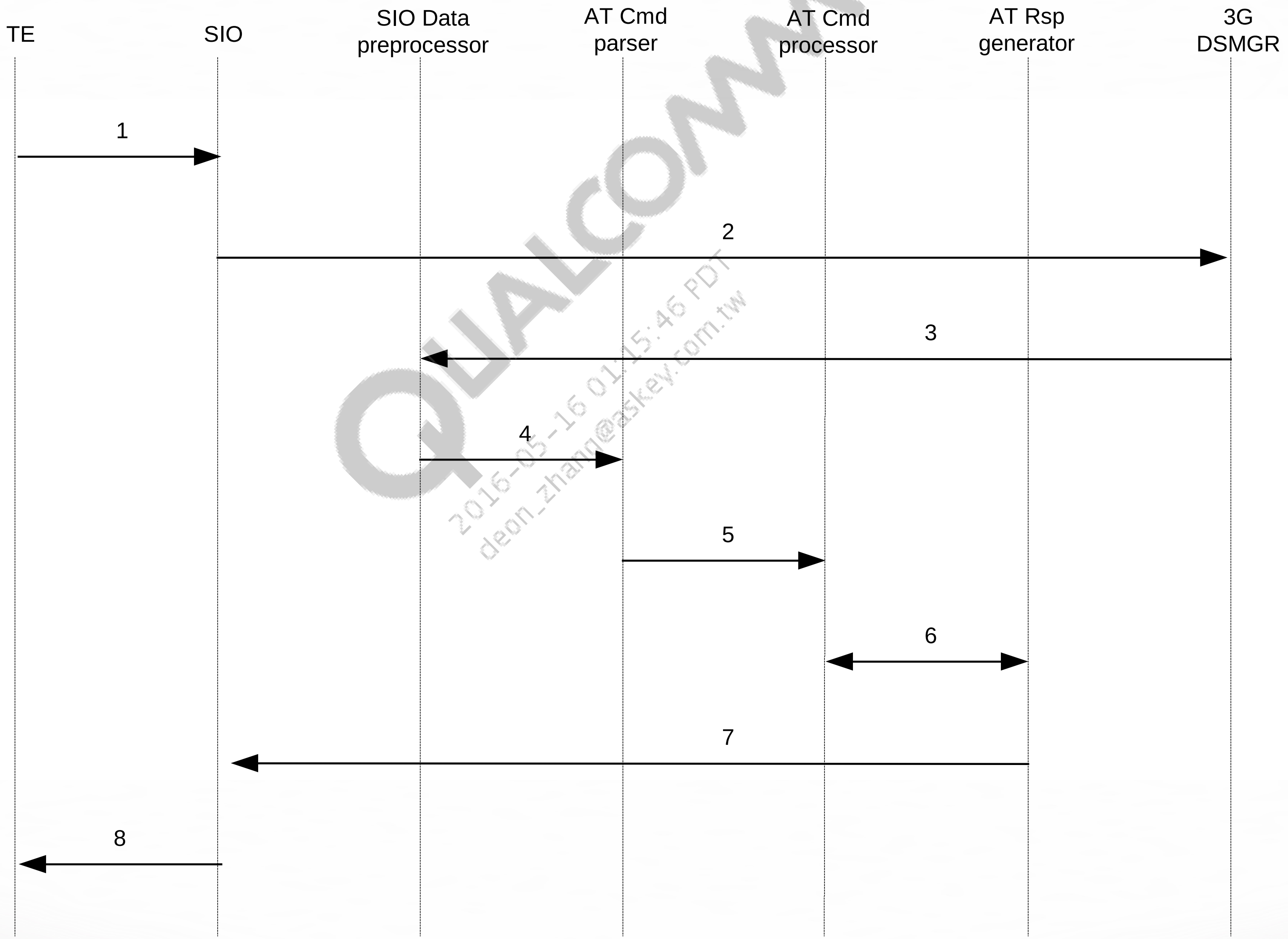
Command Table 1

cmd 1 name	attribute	special	compound	value ptr	default limit ptr	process function ptr	abort function ptr
cmd 2 name	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...

AT Command Table

- Top-level command table
  - Table of pointers to arrays of command table pointers
  - Rows represent command categories based on command parsing
  - Columns represent Operating mode for future multimode support
- Array of command table pointers
  - NULL-terminated array of pointers
  - Allows sharing of command tables across modes with less repetition of tables
- Command table
  - Each row identifies a command and data required to process the command
  - Pointer to function to execute command is included in each row
  - Pointer to function to abort command is included in each row if the command can be aborted, otherwise, it is a NULL pointer

General AT Command Flow



IS-707 ATCoP File Layout

- dsat707*ctab.c/.h – AT command table definitions
- dsat707*.c/.h – Actual command execution functions

PS Interface

- Core module for MM Data architecture
- Provides an abstraction layer that allows all link layers to provide the same interface to the upper layers
- Abstracts
  - IP Tx – Tx function same for all interfaces
  - Device control – Bringing up and tearing down the interface
  - Device configuration – Creation, setting up bridging, Tx functions, etc.
- An instance of PS interface is owned by a call control module, e.g., UMTS packet services or DS-707 packet manager
- Call control module controls the device being abstracted

PS Interface Interfaces

- Interfaces currently available
  - 1X_SN_IFACE (1X/HDR/95 – Service network)
  - 1X_AN_IFACE (HDR access network)
  - UMTS_IFACE (PDP-IP, PDP-PPP)
  - SIO_IFACE (PPP)
- Interfaces belong to one or more groups
  - ANY_IFACE_GROUP
  - ANY_DEFAULT_GROUP
  - WWAN_GROUP
  - RM_GROUP
- Groups are used to specify more than one interface

PS Interface Overview

- Provides a data structure known as the interface control block (iface_cb)
- Iface_cb memory is provided by the owning call control module
- Iface_cb contains
  - Name/instance
  - Interface state
  - Physical link state
  - Flow control state
  - IP address and relevant network parameters
  - Interface capabilities
  - Function pointers for device-specific actions – Typically commands
- Provides wrapper functions for all function pointers

PS Interface Overview (cont.)

- Three main functionalities provided by the PS interface
  - Commands
    - Used to issue commands to the underlying device
    - Used to bring up, tear down, and perform other functions associated with the interface
  - Configuration
    - Sets and retrieves the interface configuration, e.g., IP address, Retrieve state, set Tx function, etc.
  - Indications
    - Provides information to those who register for them, e.g., state changes, flow control changes

PS Interface States

- Should only be modified by commands or indications, never directly
- Stable states
  - DISABLED – No coverage, no entry in routing table
  - DOWN – Enabled but not up; can be brought up to route packets; interface exists in the routing table
  - UP – Interface can transfer IP traffic
  - ROUTABLE – Interface can only forward the IP traffic; local address is unavailable
- Transient states are COMING_UP or GOING_DOWN
- Interface will always go to DOWN before being DISABLED
- Physical link state can be COMING_UP, UP, GOING_DOWN, or DOWN
- Physical link state is partially dependent on the interface state, which is a separate state variable

PS Interface Commands

- Commands are used to modify the interface state, e.g., bring the interface up if down; flow control the interface
- Command wrappers call function pointers set by the owner of the interface
- Some of the function pointers in iface_cb
  - bring_up_cmd_f_ptr – Brings up the interface
  - tear_down_cmd_f_ptr – Tears down the interface
  - phys_link_up_cmd_f_ptr – Brings up the physical link
  - phys_link_down_cmd_f_ptr – Tears down the physical link

PS Interface Configuration

- PS Interface provides various types of configurations
  - Functions to retrieve a configuration, e.g., interface state, physical link state, IP addresses
  - Some configurations are available directly from the interface itself, e.g., whether the interface closes gracefully or aborts
  - Functions to set the configuration, e.g., IP address, Tx function, bridging function
  - Some configurations are set directly, e.g., command function pointers

PS Interface Indications

- Should only be called by the owning call code
- Same code for all interfaces
- Interface state can be changed only by indications (except for transient states)
- Available indications
  - ps_iface_enable_ind (DISABLED to DOWN)
  - ps_iface_disabled_ind
  - ps_iface_routable_ind
  - ps_iface_up_ind
  - ps_iface_down_ind (state other than DISABLED to DOWN)
  - ps_iface_phys_link_up_ind
  - ps_iface_phys_link_down_ind

PS Interface Events

- PS Interface indications and some configuration changes generate events for which clients can register
- Also has global events, which are called whenever any interface changes
- Available events are:
  - IFACE_ENABLED_EV
  - IFACE_DISABLED_EV
  - IFACE_DOWN_EV
  - IFACE_ROUTABLE_EV
  - IFACE_UP_EV
  - IFACE_PHYS_LINK_DOWN_EV
  - IFACE_PHYS_LINK_UP_EV
  - IFACE_FLOW_ENABLED_EV
  - IFACE_FLOW_DISABLED_EV
  - IFACE_ADDR_CHANGED_EV
  - IFACE_DELETE_EV

DS Task Structure

- DS task entities
  - Timers
  - NV read/write
  - Signal and command processing
  - 3G SIOLIB processing
  - 3G ATCoP processing
  - 3G DSMGR processing
  - Mode-specific handlers

DS Task Utilities

- Signals and commands are defined and implemented in dstask.c/.h; these are global across technologies.
- Timers are defined in ds3gtimer.c and .h files.
- Flow control bitmasks and macros (for any data entity and technology) are defined in ds_flow_control.h.

PS Interface and DS Task

- PS interface commands run in the context of the calling entity, which may not be the DS task.
- Therefore, the interface must send a command to the DS task, indicating:
  - Command to execute
  - Interface/mode-specific handler to which it refers
  - Interface instance (if multiple) to which it refers
- Normally, there is a separate command for each mode-specific handler (each mode-specific handler has its own unique phys_link_up_cmd enum).

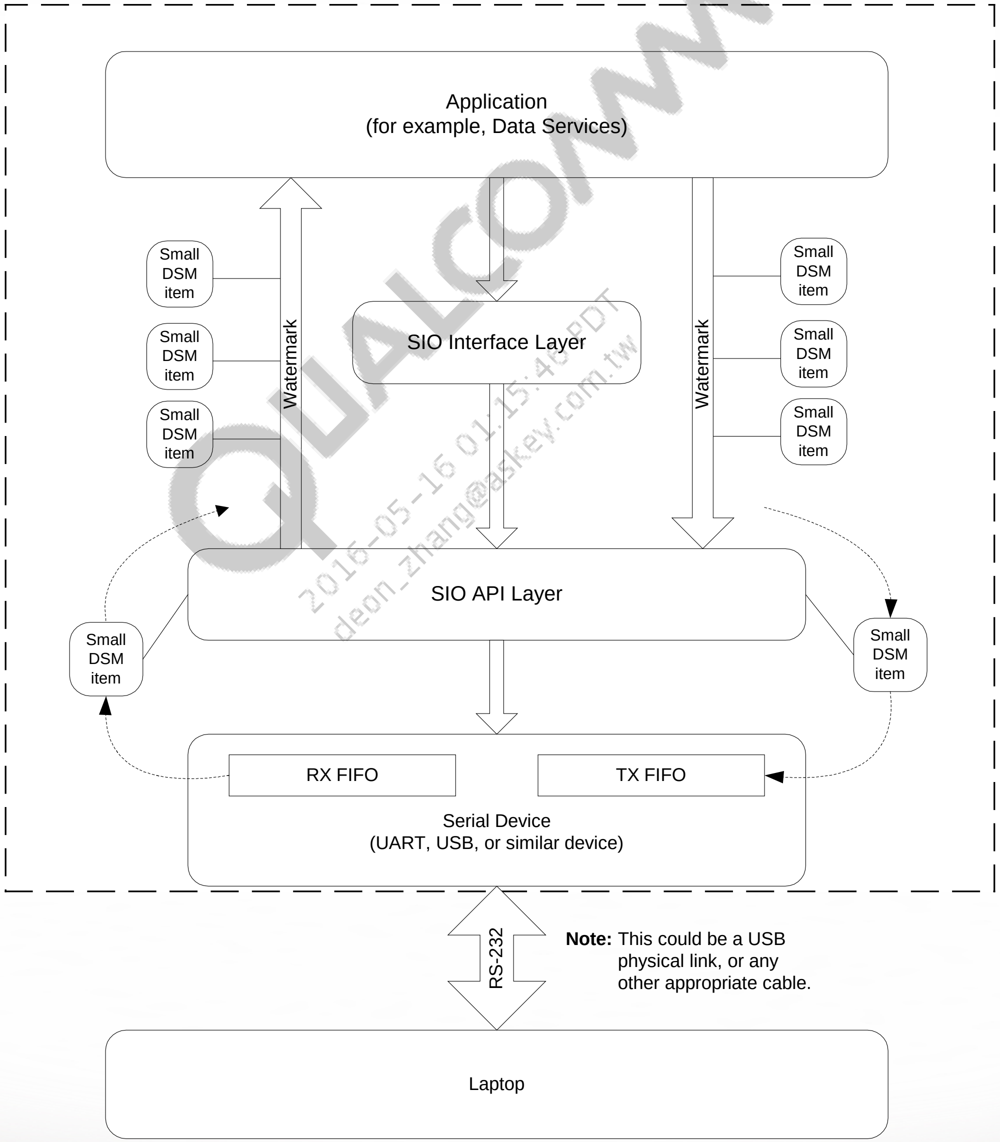
PS Interface and DS Task (cont.)

- For 1X, the PS_IFACE includes the PPP layer; however, the PPP layer processing takes place in the PS task.
- Therefore, as the PS_IFACE is brought up/down in the DS task, it may spur processing in the PS task.
- When that action finishes (PPP up/down, etc.), a signal is generated back to the PS_IFACE in the DS TASK context.

SIO

- Overview
  - Serial I/O (SIO) is an API layer that provides a tool to DMSS applications for transferring data to and from external devices through multiple serial devices.
  - It also provides a mechanism to applications to control the serial devices.
  - Multiple applications are supported, e.g., Diagnostic Monitor (DM), Data Services, RPC services, GPS services, etc.
  - Multiple devices are supported – Up to three Universal Asynchronous Receiver Transmitters (UARTs), USB, BT, etc., limited by MSM™ hardware and a multiplex configuration.
  - More devices can be easily integrated into the SIO API by the customer, provided their device driver fully supports this API.

SIO Architecture



Note: This could be a USB physical link, or any other appropriate cable.

How It Works

1. sio_init() get hardware and device driver are ready on power-up
2. Application calls sio_open() to specify port, bitrate, mode, Rx and Tx watermarks, flow control method, and optional Rx callback function
3. Application calls sio_transmit() to send data
4. Device driver puts received data into a dsm item, then passes it to Rx callback (if provided) or enqueues the item to registered Rx watermark
5. Application can change flow control method, mode, bitrate, and UART PIN status by calling sio_ioctl()
6. Application calls sio_close() after finished using a port

What is Supported

- Ability for DM to run on either UART or over USB
- Ability for DM to switch to the DS port (UART or USB) as needed
- Service programming through either UART or USB
- Flash programming through the primary UART
- Packet and AT command autodetection
- Escape character (+++) detection during async data calls
- Support packet, fax, and asynchronous data calls on primary UART or on USB
- AT command processing (Autodetect mode processing) on primary UART or on USB
- Raw data or Packet mode processing on primary UART or USB
- Each data services mode is independently available on primary UART or on USB

SIO File Layout

- sio.c – SIO generic interface layer implementation
- sio.h – Contains the public SIO interface; used by external tasks, i.e., data services and the diagnostic monitor

UART Driver

- Overview
  - Qualcomm Technology, Inc.’s (QTI) MSM chip has a build in the UART
  - One MSM chip can have up to three UART ports, limited by MSM hardware and a multiplex configuration
  - UART working voltage is 0 to 3.5 V – UART transceiver converts this logic signal to an RS-232 voltage level
  - UART driver fully supports SIO API – Controls UART hardware and sends and Rx data to and from hardware
  - Only the primary UART supports Data Services
  - All UARTs can support DM Services
  - A UART uses 8 data bits, 1 start bit, 1 stop bit, and 0 parity bits

What is Supported by the UART Driver

- Autobaud up to 19200 on the primary UART
- Up to 230.4 kbps on the primary UART (on some MSM chips)
- Primary UART supports Rx, Tx, CTS, RTS (RFR) DTR, DCD, and RI pins (which is why it only supports data)
- Second and/or third UART (if available) supports only the first 4 pins
- Primary UART usually has a 512 byte FIFO, while second and/or third usually has 64 byte FIFO – Check your chip hardware documentation for your specific MSM ASIC
- Power-saving feature is supported by shutting down UARTs

How Power Saving Works

- UART driver monitors UART transceiver whether or not it is plugged in
- UART transceiver usually lives outside of the phone
- Multiple UART transceivers are usually plugged in/out at the same time
- When the UART driver detects that the transceiver is unplugged, it shuts down UART and UART clock regime, and votes for TCXO shutting down (this vote is cast to sleep task)
- When the UART transceiver is plugged in again, the UART driver restarts the UARTs, turns on UART clock regimes, and votes against the TCXO shutting down
- UART driver monitors the UART transceiver plug/unplug on the primary UART only; it powers up/shuts down all UARTs once a plug/unplug action is detected

UART Driver File Layout

- siors232.c – UART device layer implementation
- siors232.h – UART device layer interface
- siorsreg.h – UART device layer UART register definitions

Runtime Device Mapper

- Overview
  - The Runtime Device Mapper (RDM) provides applications with a consistent way of mapping services (or applications) to the shared serial devices (or ports) at runtime.
  - All applications that will take part in using shared Serial I/O devices must:
    - Provide an open() and close() routine
    - Handle an assignment to the NULL port
  - Any application can request RDM to assign a specified application to a specified device.

How Does It Work?

1. At bootup the default port map assigns a port for Diag and Data Services dependent on what ports exist in the system.
2. An application requests a port by calling rdm_assign_port().
3. If the port to be requested is used by another service, RDM directs that service to close the port first. Then RDM assigns the requested port to the requesting service.
4. The new port mapping information is saved in EFS.
5. The entire process starts from the UI event.
6. Upon the next power-cycle, the saved port mapping information is used.

RDM File Layout

- rdevmap.c – Contains the RDM implementation
- rdevmap.h – Contains the RDM interface

Memory Management

Memory Management

- Queues
- Data Services Memory (DSM) items
- Watermarks

Queues

- Single-linked list of data blocks
- Each data block has a link field to the next data block
- Always use queue functions to use the queues
  - Declared in queue.h
    - q_init( ) – Initializes the queue
    - q_link( ) – Initializes the link field of the item
    - q_get() – Removes the element from the head of the queue
    - q_check() – Returns the pointer to the first element; does not remove
    - q_put() – Puts the element at the tail of the queue
    - q_cnt() – Counts the number of elements in the queue
- Queues – Used in the DS memory pool and throughout DS code

DSM Pool

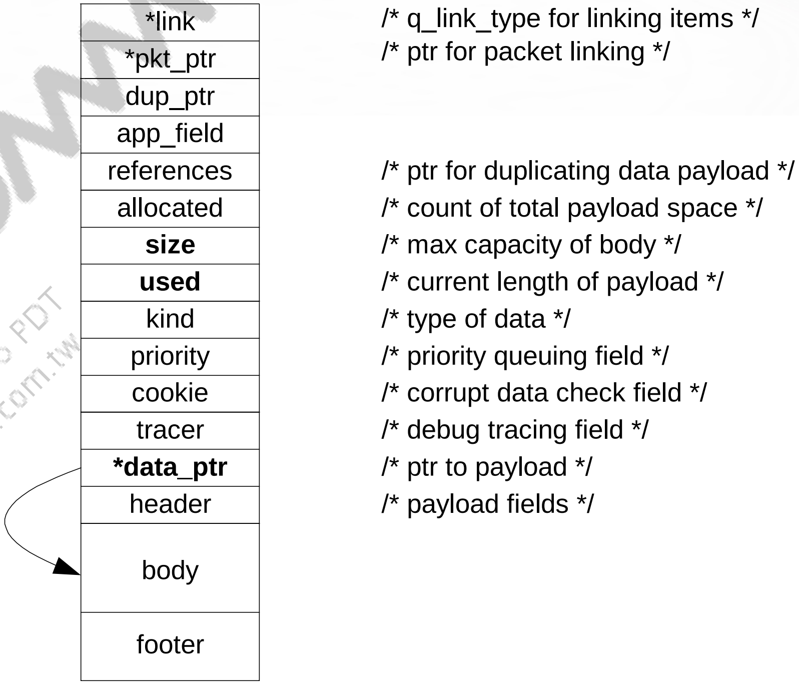
- Statically allocated memory in RAM
- Just like UNIX mbufs
- A number of DSM pools
  - Each pool is allocated as a static array
  - Flow control is built into each pool based on the number of free items, many, few, dont_exceed
- Each pool has a fixed number of items (called DSM items)
  - Each DSM item in each pool has the same fixed number of payload bytes
  - All DSM items across all pools have the same overhead bytes
- Can link each DSM two dimensionally
  - link_ptr for queues
  - pkt_ptr for items

All DSM Pools

- A number of pools based on payload size and type of usage
  - DSM_DS_SMALL_ITEM_POOL
    - Used for RLP/SIO and protocols
    - Payload size is based on RLP frame size and SIO RXLEV; it is 36/44
  - DSM_DS_LARGE_ITEM_POOL
    - Used for large TCP/PPP packets
    - Payload size is 668 bytes; TCP_MSS + room for TCP/PPP header/footer
  - DSM_DS_DUP_ITEM_POOL
    - Used for duplicating DSM items (pointer references)
    - Payload size is 0
  - DSM_DIAG_ITEM_POOL
    - Payload size is 100 bytes
    - Used by diag task to interface to SIO
  - DSM_BT_ITEM_POOL
    - Payload size is 120 bytes
    - Used within BT protocols

Data Structures

- Memory pool item – dsm_item_type
  - Purpose
    - Generic data type for carrying data payload; used throughout data services and serial driver code; modeled from UNIX ‘mbuf.’
    - Commonly used fields are in bold; remaining fields are used by DSM functions to provide the needs of protocol software units.
- DSM functions provide the means of encapsulating, decapsulating, duplicating, and removing any number of bytes from either end of an item (or string of items).
- Numerous fields are used only by DSM; Cookie and Tracer fields are used for debugging or error-checking purposes.



*link	/* q_link_type for linking items */
*pkt_ptr	/* ptr for packet linking */
dup_ptr	
app_field	
references	/* ptr for duplicating data payload */
allocated	/* count of total payload space */
size	/* max capacity of body */
used	/* current length of payload */
kind	/* type of data */
priority	/* priority queuing field */
cookie	/* corrupt data check field */
tracer	/* debug tracing field */
***data_ptr**	/* ptr to payload */
header	/* payload fields */
body	
footer	

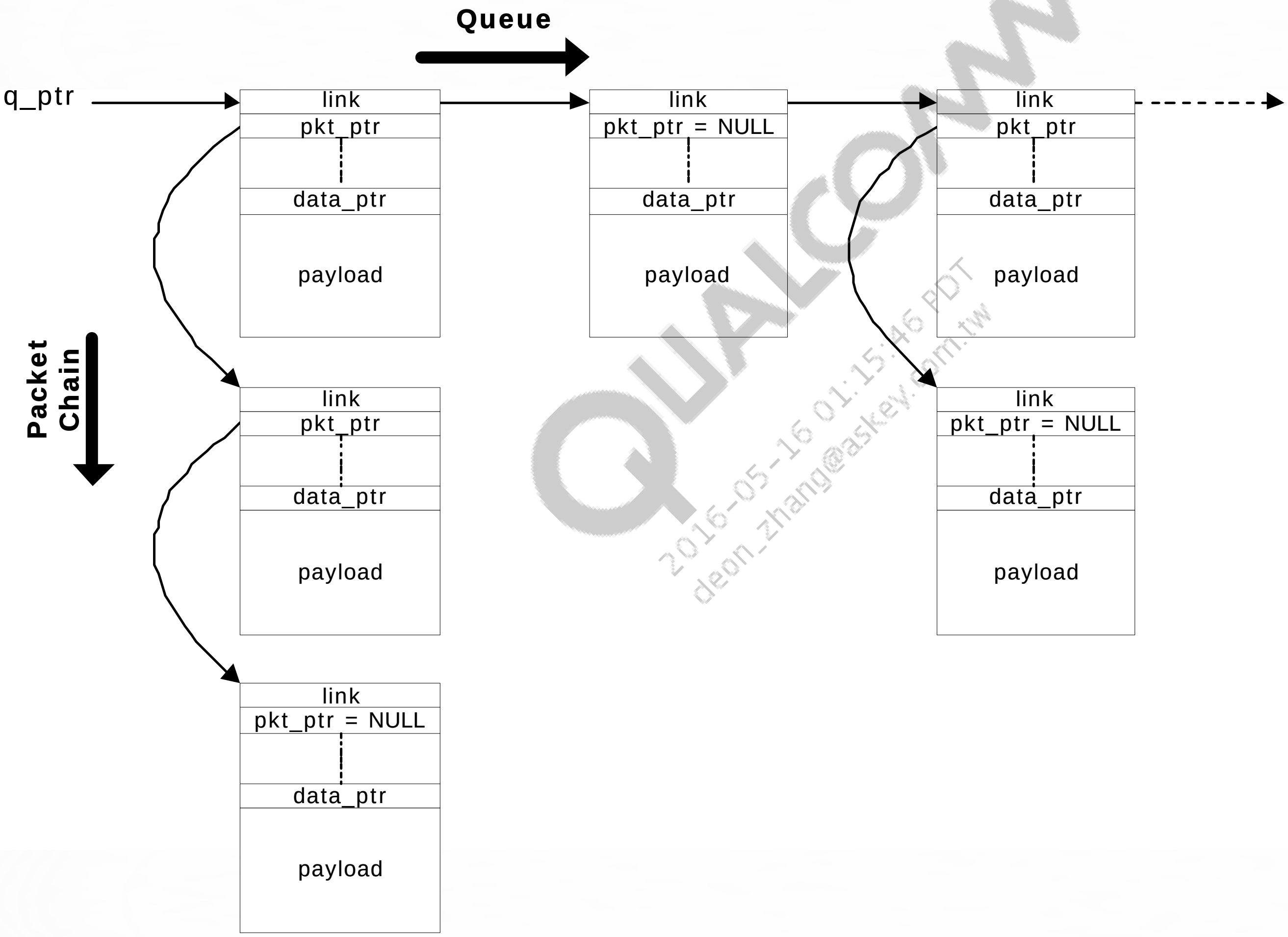
DSM Item Functions

- dsm_new_buffer(pool_id) – Gets a new item from the free stack; returns NULL when out of items in that pool
- dsm_free_buffer(item_ptr) – Puts pointer back in free stack
- dsm_free_packet(&item_ptr) – Frees entire packet chain
- dsm_length_packet(item_ptr) – Length in bytes of packet chain
- dsm_dup_packet() – Duplicates header; shares data
- dsm_trim_packet() – Removes bytes from end of chain
- dsm_offset_new_buffer() – Moves *data_ptr to the specified offset within the body when a new DSM item is allocated
- dsm_append() – Adds bytes to the end of the chain
- dsm_pushdown() – Puts *n* bytes in front of item chain
- dsm_pullup() – Copies and removes *n* bytes from front of item chain
- dsm_pullup_tail() – Copies and removes *n* bytes from end of item chain
- dsm_extract() – Copies m bytes from a given offset in a DSM item

Checklist for Using DSM Items

- Always use £ size
- Do not overwrite into the next item – Will cause pointer scribble of a queue
- Check the cookie – BEAF
- Use dsm_free_packet() instead of dsm_free_buffer() – Dangling items
- Check for memory leaks – Free counts and debug screen

Queues and DSM Packet Chains Example



Queue

q_ptr

Packet Chain

link
pkt_ptr
data_ptr
payload

link
pkt_ptr = NULL
data_ptr
payload

link
pkt_ptr
data_ptr
payload

link
pkt_ptr
data_ptr
payload

link
pkt_ptr = NULL
data_ptr
payload

link
pkt_ptr = NULL
data_ptr
payload

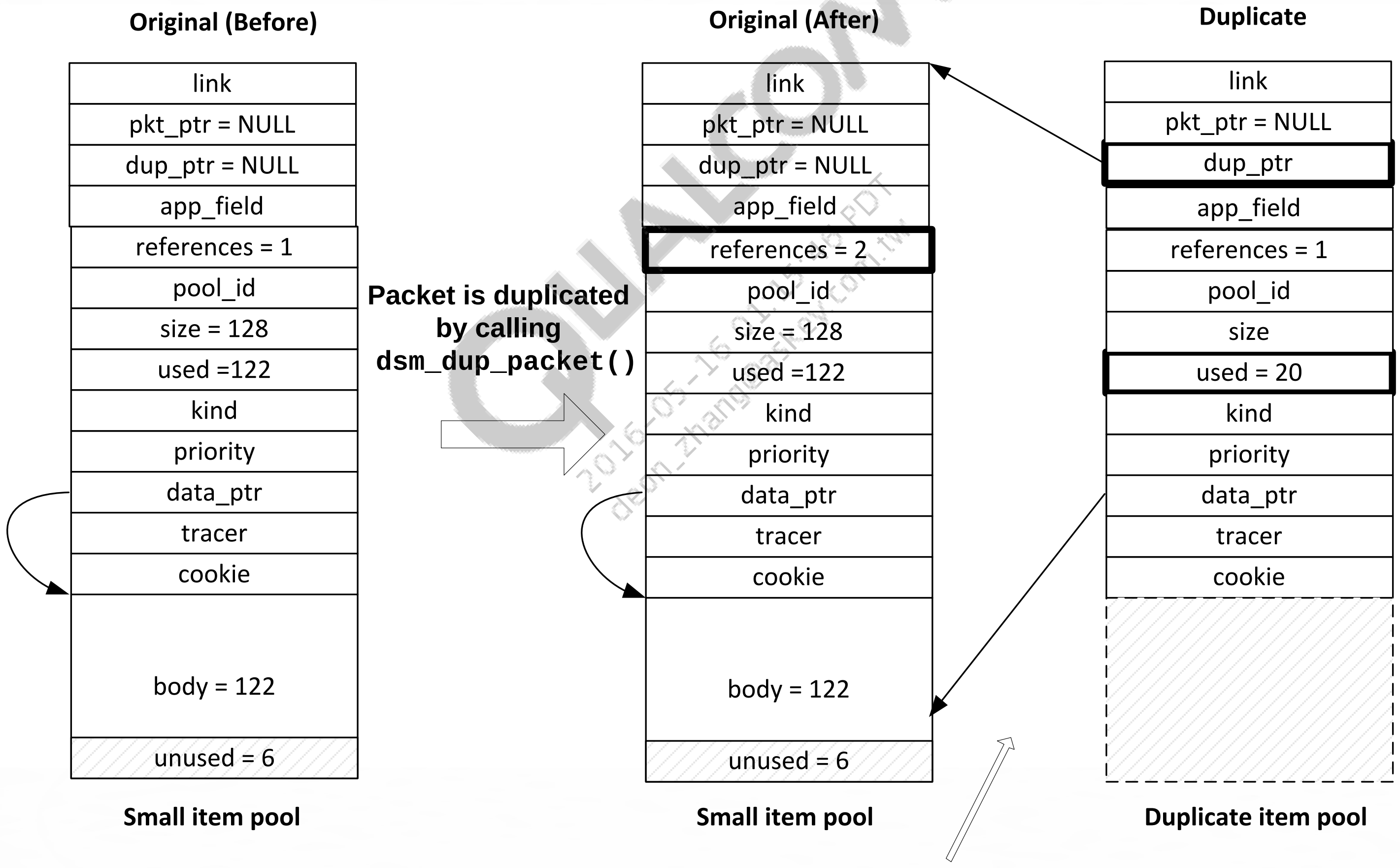
Associated DSM

Functions

- dsm_free_packet()
- dsm_append()
- dsm_trim_packet()
- dsm_pushdown()
- dsm_pullup()
- dsm_dup_packet()
- dsm_extract()
- dsm_length_packet()

Duplicating Packets Example

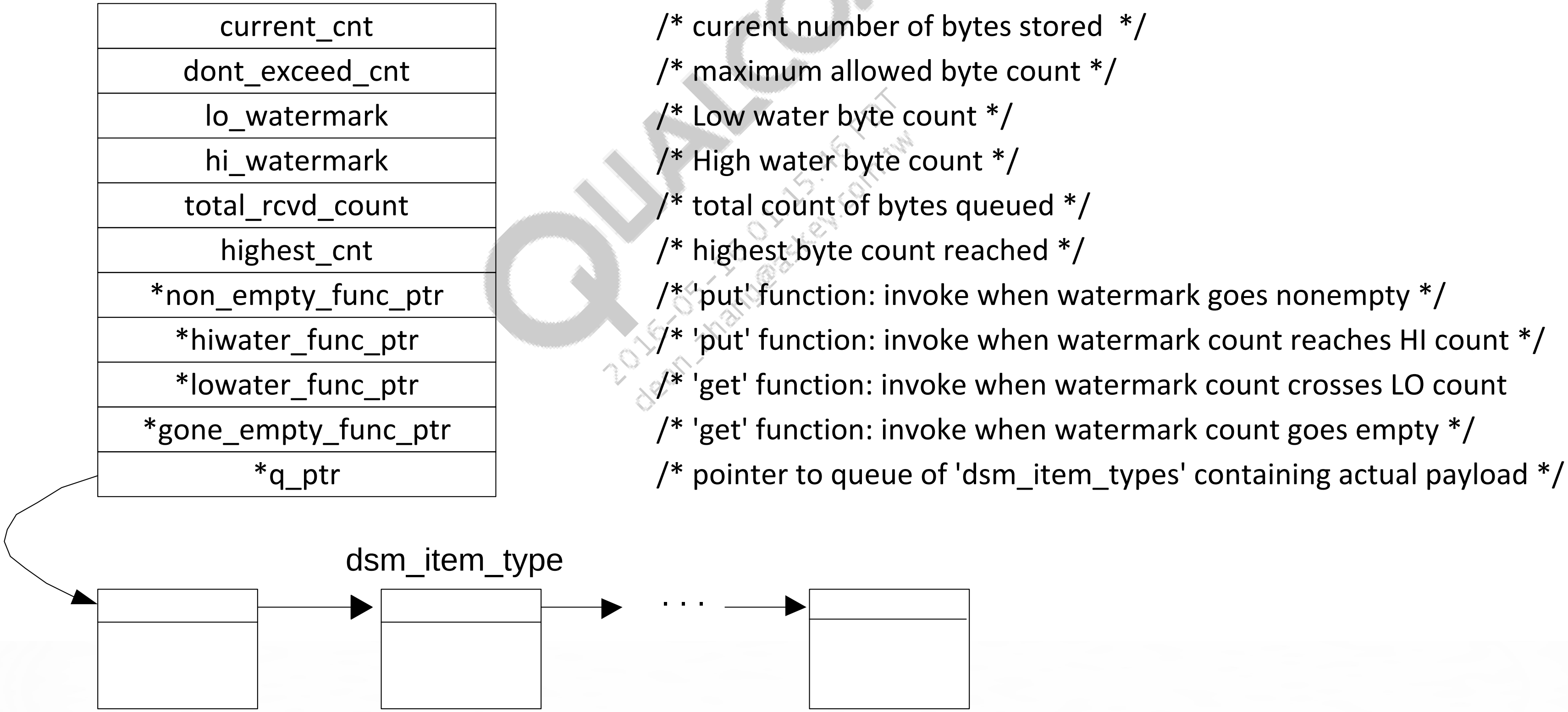
In this example, the last usable 20 bytes of the original packet are duplicated.



Note: The data_ptr now points to the beginning of the last usable 20 bytes in the item.

Flow Control Point – dsm_watermark_type

- Purpose – Provides a generic means of performing Data Payload flow control.



current_cnt	/* current number of bytes stored */
dont_exceed_cnt	/* maximum allowed byte count */
lo_watermark	/* Low water byte count */
hi_watermark	/* High water byte count */
total_rcvd_count	/* total count of bytes queued */
highest_cnt	/* highest byte count reached */
*non_empty_func_ptr	/* 'put' function: invoke when watermark goes nonempty */
*hiwater_func_ptr	/* 'put' function: invoke when watermark count reaches HI count */
*lowater_func_ptr	/* 'get' function: invoke when watermark count crosses LO count
*gone_empty_func_ptr	/* 'get' function: invoke when watermark count goes empty */
*q_ptr	/* pointer to queue of 'dsm_item_types' containing actual payload */

DSM Watermark Functions

- dsm_enqueue(wmark, item)
  - Adds an item to the watermark
  - Calls each_item_func(), if any
  - If this is the first item in the watermark, calls non_empty_func_ptr
  - If high watermark in bytes is reached, calls hiwater_func_ptr (deassert CTS)
  - If it does not_exceed_cnt bytes in the watermark, drops the item
- dsm_dequeue(wmark)
  - Removes the first item from the watermark
  - If no other items are left in the watermark, calls gone_empty_func_ptr
  - If low watermark is reached, calls lowater_func_ptr (assert CTS)

Note: Watermark counts are in bytes; not number of items.

PPP – How It Works

- PPP – Point-to-Point Protocol
- RFC 1661 – The protocol
- RFC 1662 – PPP in HDLC-like framing
- Frame format
  - Like HDLC – 0x7e(~) begins packet, 2-byte CRC at end
  - 0x7D – Used to escape out 0x7e, 0x7D
    - ACCM can also define characters below 0 x 20 to escape
  - Inside HDLC, PPP packets header contains protocol, type, e.g., c-req, ID, length, and options
  - Each option has option #, length, and option data
- PPP consists of various phases
  - Dead – Startup
  - Establish – LCP negotiation
  - Authenticate – PAP/CHAP
  - Network – IPCP negotiation (IPv6CP in the future)
  - Terminate – LCP term-req and ACKs

PPP – The Protocols, LCP

- Link Control Protocol (LCP)
  - Defined in RFC 1661
  - Protocol number C021
  - Negotiates
    - Async Character Control Map (ACCM)
    - Address and Control Field Compression (ACFC)
    - Protocol Field Compression (PFC)
    - Magic number
    - Authentication protocols, i.e., PAP or CHAP

PPP – The Protocols, IPCP

- IP Control Protocol (IPCP)
  - Defined in RFC 1332
  - Protocol number 8021
  - Negotiates
    - IP address
    - VJ header compression
    - DNS server addresses – Defined in RFC 1887

PPP – The Protocols, PAP

- Password Authentication Protocol (PAP)
  - Defined in RFC 1334
  - Protocol number C023
  - Most basic form of authentication
  - User name and password are transmitted to authenticator
    - Done in clear text
    - Significant weakness
  - User name and password are stored in NV
  - Well-defined interface for configuring user name and password in NV is provided

PPP – The Protocols, CHAP

- Challenge Handshake Authentication Protocol (CHAP)
  - Defined in RFC 1994
  - Protocol number C223
  - CHAP is more secure than PAP because the password is never sent in clear text
  - User name and password are stored in NV
    - Same as those used for PAP
    - Means that PAP is used – CHAP security is compromised
  - A well-defined interface for configuring user name and password in NV is provided
  - Provides protection against playback attack by using a random challenge and an incremental identifier
  - CHAP uses a one-way hash function for encryption
    - Shared secret (or password) is used to hash the challenge
  - MD-5 is currently the only hash function supported
  - Authenticator, i.e., the PDSN, sends challenge
  - Authenticatee, i.e., the mobile, sends a response
  - Authenticator then either accepts or fails authentication
  - CHAP challenges can arrive anytime, i.e., it is not limited to authentication phase

PPP – The Implementation

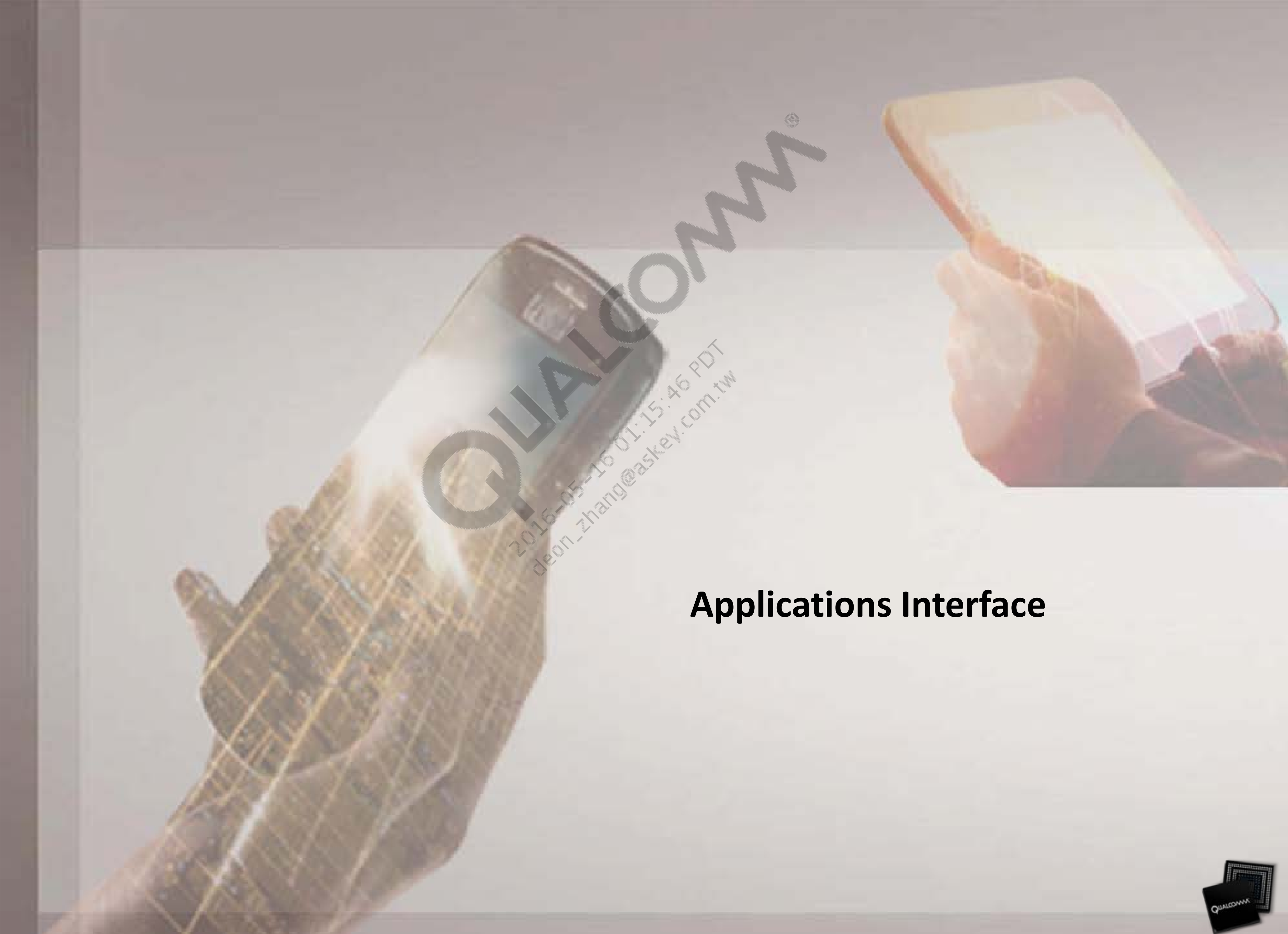
- PPP API can be called from any task context
- Includes a configuration API
  - Allows almost everything to be configured before initializing/starting a PPP instance
    - Includes authentication parameters and protocol behavior
- Number of PPP instances is defined at compile time
- All instances can run simultaneously
  - No dependence between instances
  - Most API calls require the PPP instance to be passed in
- All PPP information is stored in the PPP control block
  - Includes authentication information
  - One control block per instance

PPP – The API

- ppp_start() – Starts a PPP instance with the specified configuration
- ppp_stop() – Gracefully closes PPP instance; may abort based on the PS Interface parameter if the interface is dormant
- ppp_abort() – Aborts a PPP instance
- ppp_resync() – Initiates a PPP resync on a particular instance; restarts the state machine
- ppp_get_default_opts()
  - Will populate a configuration data structure with the default setup
  - Most clients need only to modify minor options in the PPP configuration, e.g, disable authentication for IS-707 async/fax calls
- ppp_get_dev_setup() – Will populate a configuration data structure with the setup of a specified PPP device, e.g., HDLC Framing mode and the associated PS interface
- ppp_get_protocol_opts() – Will populate a configuration data structure with the protocol configuration of the specified PPP instance

PPP – The API (cont.)

- ppp_reg_event()
  - PPP now supports three events
    - Up
    - Down
    - Resync
  - A single callback can be registered for each event
- ppp_set_mode() – Sets the mode of a PPP instance, e.g., internal, pseudo_net, etc.
- ppp_get_mode() – Retrieves the mode of a PPP instance



Applications Interface

Applications Interface

- Sockets API
- Brew™
- DSS_IFACE_IOCTL API
- Applications Requirements and Recommendations

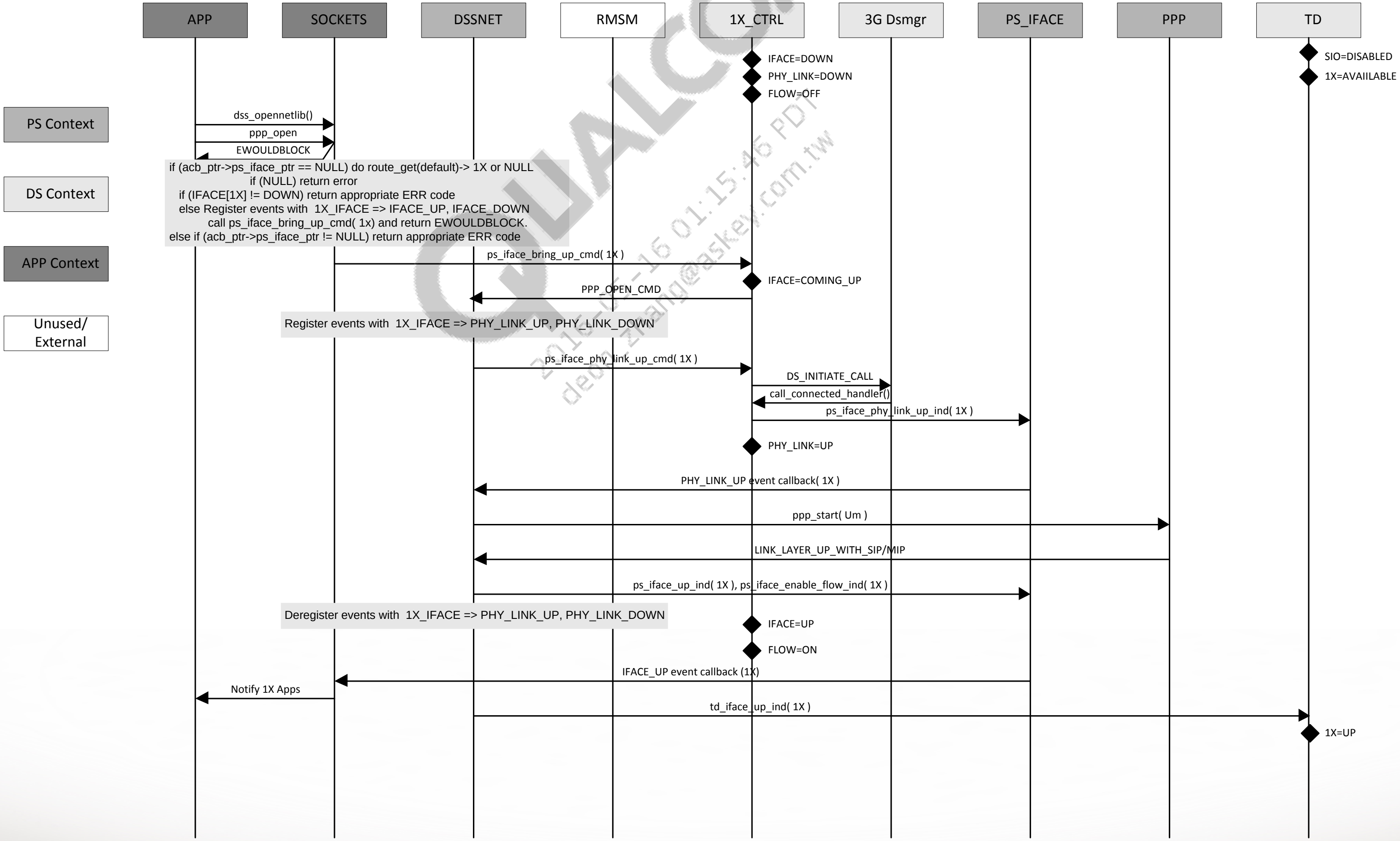
Sockets Calls

- Provides a generic API for Socket applications running on the mobile
- Supports simultaneous UDP and TCP sockets
- Call originates by means of PS_IFACE
- Supports dormancy
- Works on QNC as well as regular packet calls
- API is defined in dssocket.h and dssdns.h
  - Other sockets files include dssocket.c, dssocki.c, dss_ps.c, dssifcbacks.c, dsstcp.c, dssudp.c, dssdns.c

Sockets Call Flows 1X Call Setup

NOTES

- 1X_IFACE created and initialized at powerup
- 1X Interface is enabled when service becomes available, default route is added
- Default operational mode configured for Sockets: data from and to SIO_IFACE is denied by default



```
if (acb_ptr->ps_iface_ptr == NULL) do route_get(default)-> 1X or NULL
                    if (NULL) return error
  if (IFACE[1X] != DOWN) return appropriate ERR code
  else Register events with  1X_IFACE => IFACE_UP, IFACE_DOWN
          call ps_iface_bring_up_cmd( 1x) and return EWOULDBLOCK.
else if (acb_ptr->ps_iface_ptr != NULL) return appropriate ERR code
```

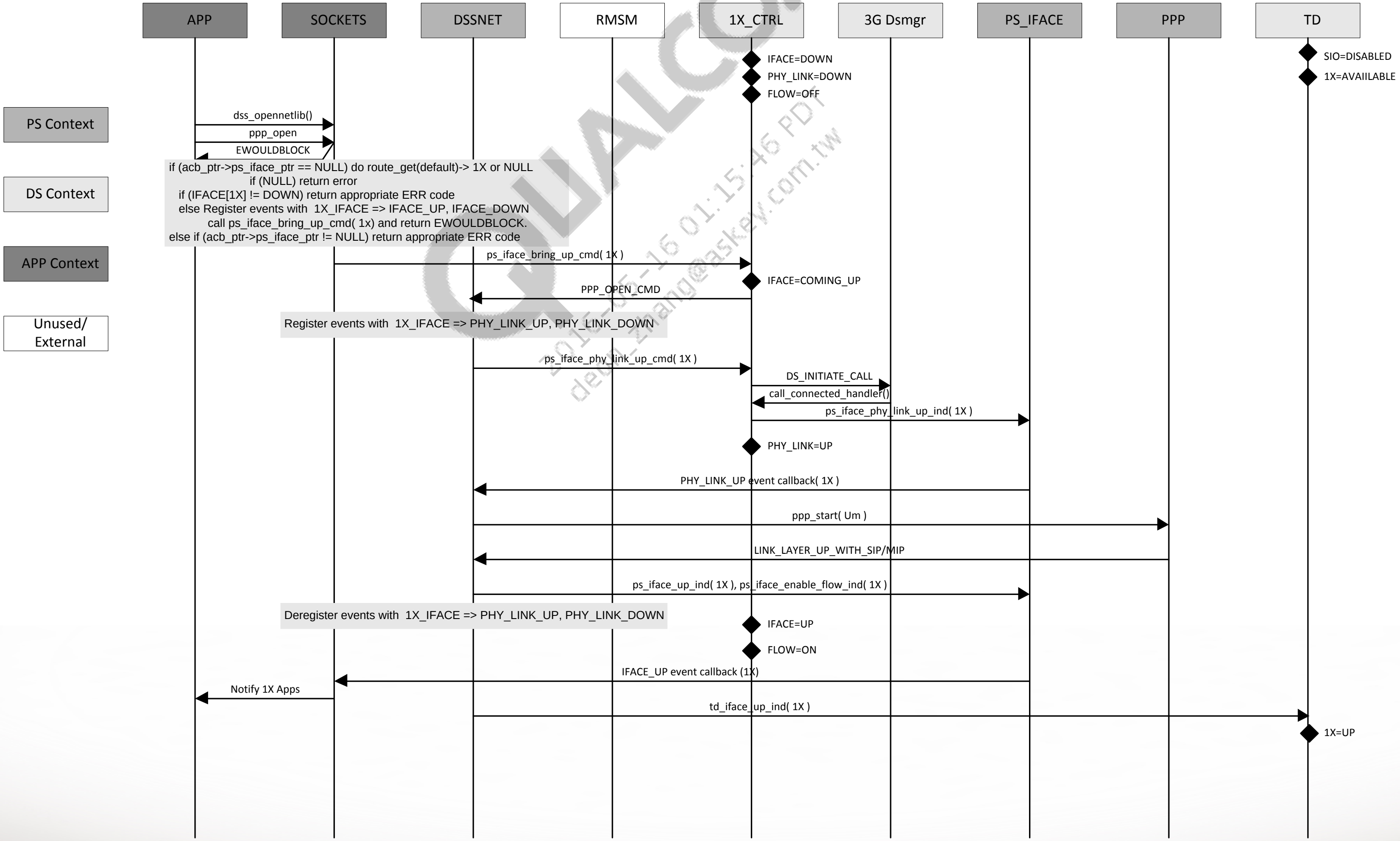
Register events with 1X_IFACE => PHY_LINK_UP, PHY_LINK_DOWN

Deregister events with 1X_IFACE => PHY_LINK_UP, PHY_LINK_DOWN

Sockets Call Flows 1X Call Release

NOTES

- 1X_IFACE created and initialized at powerup
- 1X Interface is enabled when service becomes available, default route is added
- Default operational mode configured for Sockets: data from and to SIO_IFACE is denied by default



```
if (acb_ptr->ps_iface_ptr == NULL) do route_get(default)-> 1X or NULL
                    if (NULL) return error
   if (IFACE[1X] != DOWN) return appropriate ERR code
   else Register events with  1X_IFACE => IFACE_UP, IFACE_DOWN
          call ps_iface_bring_up_cmd( 1x) and return EWOULDBLOCK.
else if (acb_ptr->ps_iface_ptr != NULL) return appropriate ERR code
```

Register events with 1X_IFACE => PHY_LINK_UP, PHY_LINK_DOWN

Deregister events with 1X_IFACE => PHY_LINK_UP, PHY_LINK_DOWN

Sockets API Usage

- Network library concept
  - dss_open_net_lib() – Assigns an application ID and sets the application-defined network and sockets callbacks
- PPP needs to be opened before the socket can be opened (does not apply to loopback sockets)
  - dss_open() uses the interface (ps_iface) returned by routing the lookup
  - dss_pppopen() brings up the physical link and establishes the link layer associated with the interface (ps_iface) being used
- Sockets API is indifferent to the type of air interface
- Application uses dss_socket() to open the socket
- Application uses dss_read()/dss_write() for the I/O

Sockets Event Mechanism

- Asynchronous notification mechanism is based on callback functions
- Four events are defined
  - READ_EVENT
  - WRITE_EVENT
  - CLOSE_EVENT
  - ACCEPT_EVENT
- dss_async_select() tells the API the events application in which it is interested
- API calls the socket callback function if any interested event for the socket occurs
- Application uses dss_get_next_event() to identify which event occurred

Sockets API Functions

- Application-based functions
  - dss_open_netlib(), dss_close_netlib()
  - dss_pppopen(), dss_pppclose()
- Socket-based functions
  - dss_socket(), dss_close()
  - dss_bind(), dss_connect(), dss_accept(), dss_listen()
  - dss_write(), dss_writev(), dss_sendto()
  - dss_read(), dss_readv(), dss_recvfrom()
  - dss_setsockopt(), dss_getsockopt(), dss_getsockname(), dss_getpeername(), dss_shutdown()
- Asynchronous messaging routines
  - dss_async_select(), dss_async_deselect()
  - dss_ getnextevent()

Sockets API Functions (cont.)

- Byte-ordering routines
  - dss_htonl(), dss_htons()
  - dss_ntohl(), dss_ntohs()
- DNS routines
  - dss_getipnodebyname(), dss_getipnodebyaddr(), dss_freehostent(), dss_gethostaddrs()
  - dss_inet_aton(), dss_inet_ntoa()
- For detailed information on the Sockets API refer to [Q2]

Sockets and Dormancy

- Dormant mode is the state in which the traffic channel is down but PPP is up.
- The state of the traffic channel, in terms of dormancy, is hidden from the user.
  - Application gets DS_EWOULDBLOCK when attempting to write to a socket when the mobile is dormant
  - Application gets the DS_WRITE_EVENT once the mobile comes out of dormancy

Loopback Sockets

- Allows applications on the phone to communicate with each other
- Always on the interface
- Support for both TCP and UDP loopback
- Does not require a call to dss_pppopen()
- Local loopback address is 127.0.0.1
- Loopback traffic traverses the complete IP stack

dss_iface_ioctl()

- Application interface to set/get the interface configuration
- IOCTLs are classified into three categories
  - Common IOCTLs – Implementation is common to all interfaces, e.g., 1X, UMTS, etc.; it is supported on all interfaces; IOCTLs include:
    - Retrieving IP and DNS addresses
    - Retrieving interface and physical link states
    - Ability to register/deregister for events (IP address change, interface, and physical link state changes)
  - Interface-specific common IOCTLs – Implementation of these IOCTLs are interface-specific; it is supported by a subset of the available interfaces; IOCTLs include:
    - Go Dormant
    - Go Active
    - Go NULL
  - Interface-specific IOCTLs – These IOCTLs are interface-specific and are supported only by one of the interfaces; 1X-specific IOCTLs in this category are:
    - Get/Set various RLP parameters
    - Get/Set MDR and dormant timer value

cdma2000

cdma2000

- Standards Overview
- Packet Data
- Circuit-Switched Data
- File Layout

IS-707 Data Overview

- CDMA and Data Services complement each another
  - Both have the same objective, to transmit bits over the air
  - CDMA with soft handoff is ideal for data, as opposed to AMPS
- Initial concept was that data would be free with CDMA
  - Requires only software changes – Primarily true
  - All-in-phone data connectivity would boost CDMA usage
- Common characteristics of Data Services
  - All perform vocoder bypass
  - All run RLP to improve link layer for upper layers
  - Primarily PPP-over-RLP

Background

- CDMA data began circa 1990 at QTI
- QTI worked quickly toward the production of initial data demos and deployment
- IS-99 was standardized
- First demos circa 1995 at the CTIA show in San Diego
- Official announcement of Data Services in phones and CBS was made at the 1997 CTIA show
- IS-707-based Data Services was deployed in Japan, Korea, and North America

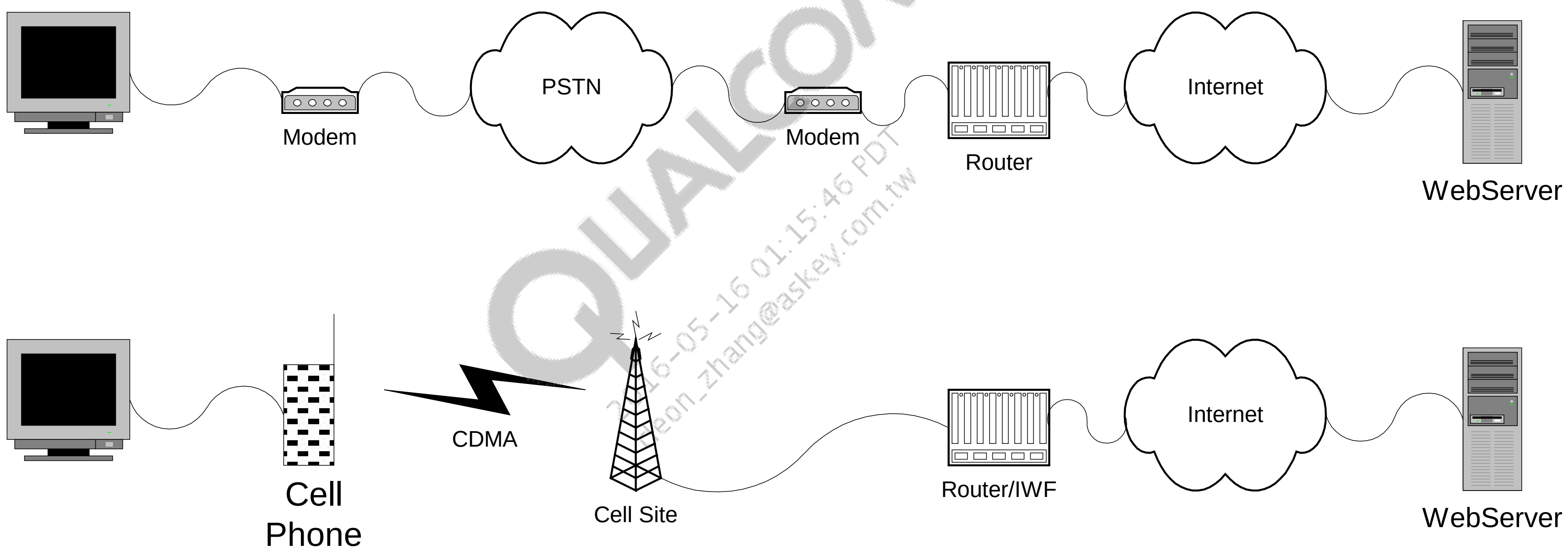
Data Standards

- Initially, IS-99
  - Circuit-switched – Async and fax (separate Service Options 4, 5)
  - New RLP, all IETF protocols (TCP/IP/PPP), EIA-602 ATcmds, 617 reflection
  - Rate Set 1 only
- Later, IS-657
  - Relay and Network Model packet – Service Options 7, 8
  - Dormant mode, RLP encryption, Rate Set 1 only
- Features were merged in IS-707
  - All of the Data features – Async, fax, packet, STU-III
  - Rate Set 2, Service Options 12, 13, 15 (+8 rule)
  - Service Option mess and IS-707 functionality with the IS-99 service option
- IS-707A – IS-95B, MDR RLP Type II, end-to-end analog fax
- IS-707A-1 – IS-2000 Rel 0
- IS-707A-2 – IS-2000 Rel

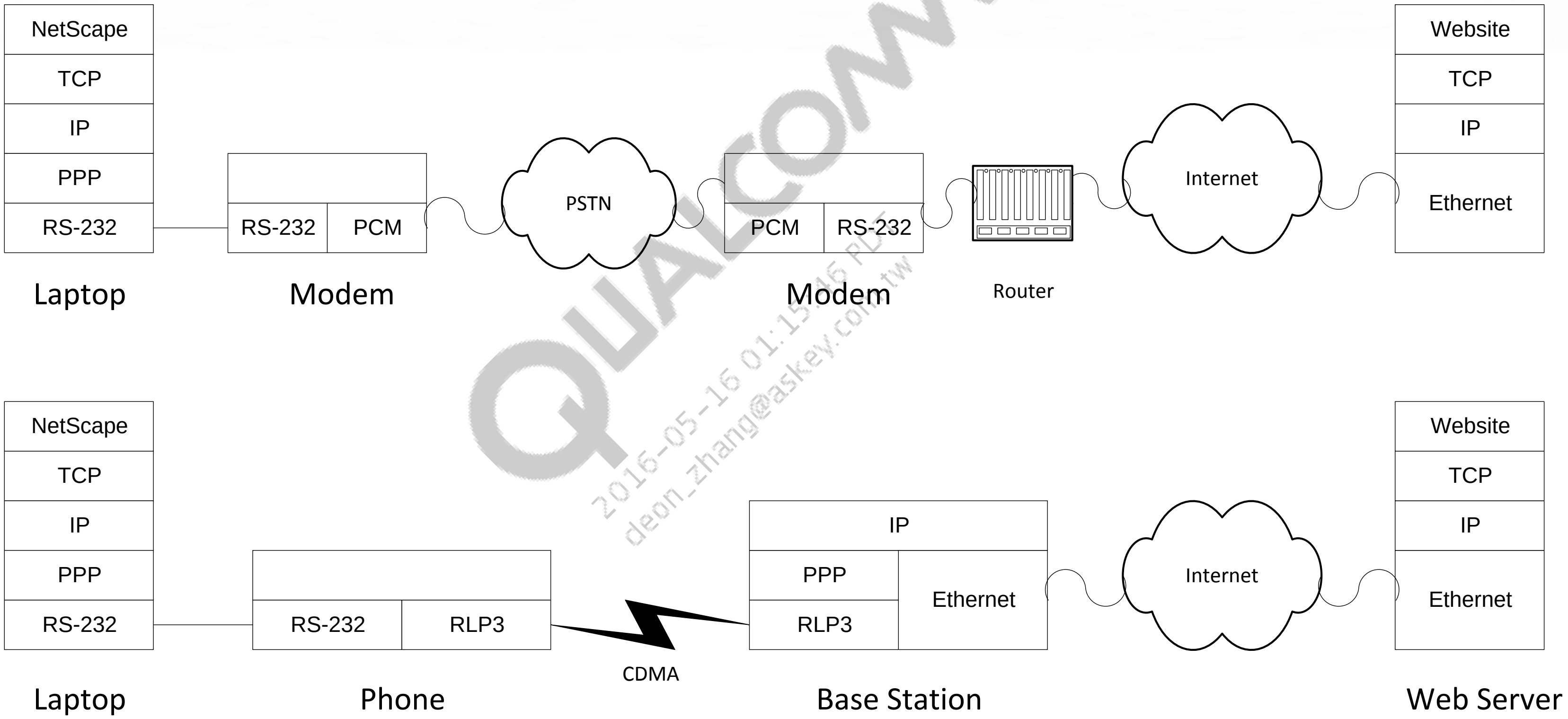
IS-707 Packet Data – Support

- Supports speeds up to 307 kbps forward and reverse (Rel A)
- Supports laptop calls (relay and network model)
- Supports Sockets applications
- Currently supports only one IP address
  - Mobile and base station issue

Internet Connection Comparison



Internet Connection Comparison (cont.)



NetScape
TCP
IP
PPP
RS-232
Laptop

RS-232 | PCM
Modem

PSTN

PCM | RS-232
Modem

Router

Internet

Website
TCP
IP
Ethernet

NetScape
TCP
IP
PPP
RS-232
Laptop

RS-232 | RLP3
Phone

CDMA

IP
PPP | Ethernet
RLP3
Base Station

Internet

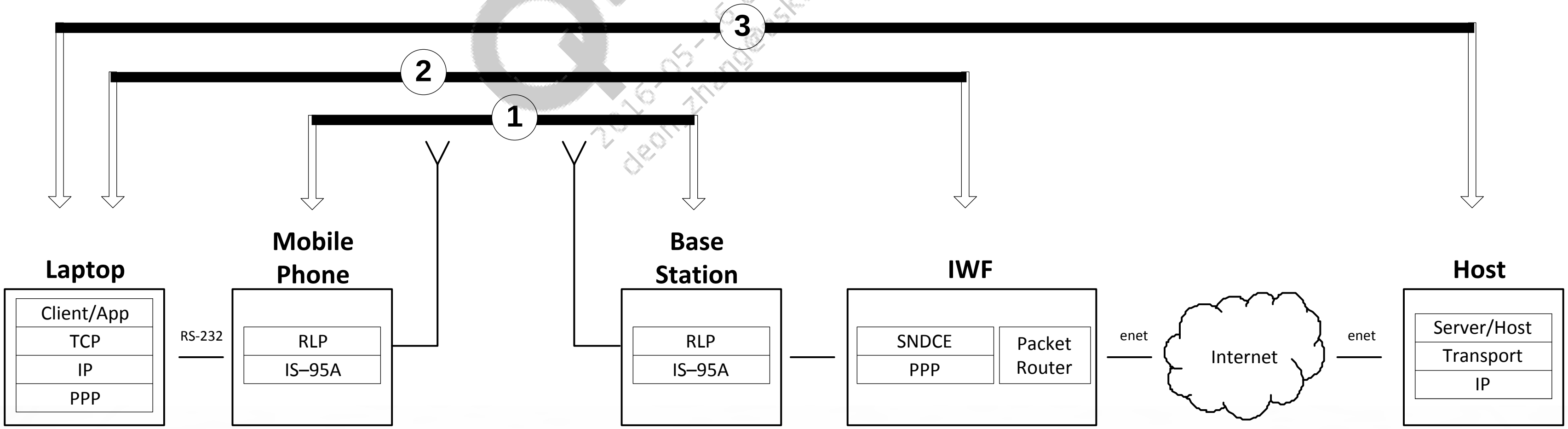
Website
TCP
IP
Ethernet
Web Server

Packet Sync Stages

1 RLP Sync – Occurs once the Traffic Channel Conversation substate is reached (~6 to 8 frames (120 to 60 ms))

2 Serial end-to-end PPP Neg – Negotiates IP and Link parameters, including TE2 IP address (~500 ms or less)

3 End-to-end TCP Sync – Establishes end-to-end connection (~2 sec)



Data Rates

- Low speed
  - Uses fundamental channel
  - Maximum rate is 9.6/14.4 kbps (RS1/RS2)
- Medium Data Rate (MDR)
  - Dynamically employs extra Supplemental Code Channels (SCCH)
    - Each has its own Walsh code
    - Much like having extra fundamental channels
  - Maximum rate 64/76.8 kbps (RS1/RS2)
- 1X
  - Dynamically employs extra Supplemental Channel (SCH)
  - Fat-Pipe concept – Can receive multiple frames in a 20-ms period
  - Maximum rate 307.2 kbps (Rel A)

Packet Data Service Options

- Supported service options
  - 1X – 33; up to 307 kbps
  - MDR – 25, 22; up to 76.8 kbps
  - Low rate – 7, 15, x8020, x1007; up to 14.4 kbps
  - Low rate – QNC 4, 12; up to 14.4 kbps
- Service options sets – Defined by AT&QCSO
  - 0, Pre-707 – 7, 15, 22, 25, 33
  - 1, Proprietary – 7, x8020, 22, 25, 33
  - 2, 707 – x1007, 15, 22, 25, 33 (default)

Laptop Calls

- TE2 sees the mobile as a standard modem; the following laptop elements are unchanged:
  - Dial-up networking
  - PPP/IP/TCP/UDP protocols
  - Applications, e.g., any browser)
- TE2 can connect to the mobile through RS-232, USB, or BT
- Laptop initiates a data call with ATDT#777; all other dial strings result in an async data call
- Packet dial-string can be configured in NV
- If dormant, any data from a laptop results in a reorigination
- AT&D is ignored
  - Deassertion of DTR always results in the traffic channel being torn down
  - Online Command mode is not supported in a packet data call
- Mobile notifies the laptop of a data session termination by deasserting DCD

Laptop Calls (cont.)

- AT&C is used to control DCD
  - 0 – Never deassert DCD
  - 1 – DCD follows traffic channel exactly (no dormancy)
  - 2 – Always asserted, except at end of call, where DCD winks (to support issues seen with WinNT™) (default)

Relay and Network Models

- Both are laptop calls
- Relay model (AT+CRM = 0 or 1)
  - Only RLP on the phone
  - PPP and above reside on the laptop
  - Data passes directly between SIO and RLP – Mobile performs no processing of data
- Network model (AT+CRM = 2)
  - Phone runs PPP on Rm and Um
  - Data is parsed to look for PPP control packets
  - Allows PPP resyncs to be hidden from the laptop (as long as renegotiated PPP options remain unchanged)

cdma2000 Sockets Calls – dssnet State Machine

- Sockets network state machine used by 1X
- Brings up and tears down the physical link during call setup/release
- Establishes/stops PPP during call setup/release
- Triggers the establishment of MIP registration process when in a MIP call
- Responsible for flow controlling socket apps during PPP establishment and resync, and during initial MIP establishment
- For detailed information about dssnet, refer to [Q3]

QNC Data

- What is QNC?
  - Backdoor packet – On async Service Options 4 and 12
  - Available only on P_REV = 3 systems
  - IWF looks at IP address and decides
    - If local – Circuit call
    - If Internet – Packet call
  - Does not support dormancy – If traffic channel goes down, then so does PPP
  - Enabled by default; AT$QCQNC toggles are:
    - 0 – Disabled
    - 1 – Enabled

Dormancy

- If no data was sent or received on the RLP for *n* seconds, traffic channel is torn down, but PPP state is maintained
- Is only supported for TE2-MT2 usage model; requires IWF/PDSN dormancy support
- Sockets application/laptop are not notified of dormant status
- Base station-initiated – All base station call terminations result in dormancy
- Mobile-initiated
  - Timeout – AT+CTA=n (seconds)
  - Default = 0 sec (mobile does not initiate dormancy)
  - If mobile terminates the call, an ORDQ bit is sent to the base station to allow it to clean up its PPP instance
    - ORDQ = 0 – Mobile is going dormant
    - ORDQ = 2 – Mobile is going NULL (tearing down PPP)

Hold-Down Timer

- Base station can tell the mobile to not reoriginate a dormant packet data call for a specified amount of time
- Time is specified in Service Option Control Message (SOCM)
- Does not apply to originations from NULL
- Mobile defaults this value to 0

Retry Delay Timer

- Base station can also tell the mobile to not originate any data calls for a specified amount of time
- Applies to all packet data originations (NULL and dormant)
- Received in the retry delay message

AT$QCMDR

- Used to select allowable data rates on the mobile
  - 0 – MDR only; mobile will only originate with SO22 and 25
  - 1 – MDR preferred; SO22 and 25 are preferred, but low-rate SOs are also enabled; SO33 is not enabled
  - 2 – No MDR; only low-rate SOs are enabled
  - 3 – 1X-enabled; SO33, MDR, and low-rate SOs are enabled; SO33 is preferred
- AT$QCMDR = 3 is the default; other values are for test purposes

PZID/SID/NID

- Mobile will reoriginate dormant calls when PZID, SID, or NID change.

SCRM

- 1X allows the user to send a Supplemental Channel Request Message (SCRM) to request extra bandwidth on the reverse link.
- It is available only with SO33.
- It can be toggled with AT$QCSCRM.
- The mobile has a proprietary algorithm to decide how and when to send these messages.

Control/Hold

- When using DCCH, the mobile can enter a control/hold state where the call remains up, but no data is exchanged between the mobile and the base station.
- The base station must explicitly tell the mobile to enter and exit the control/hold state.
- The mobile can request the initiation of the control/hold state by sending a Resource Request Message (RRM).
- The mobile timer is set by AT$QCCHS = *n*, which indicates the number of idle 20-ms periods before requesting a transition to the control/hold state.
  - Default is 0 (mobile never requests transition to control/hold)
- The mobile requests a transition out of the control/hold state when it has data or RLP control frames to send.

RLP – TCP First

- TCP frames are typically anywhere from 500 to 1500 bytes in length
- Reliable delivery – Lost frames are retransmitted
- TCP has backoff algorithms
  - If data is lost or corrupt, then TCP will back off
  - Backoff is exponential
  - TCP increases speed as good data is received

TCP and CDMA

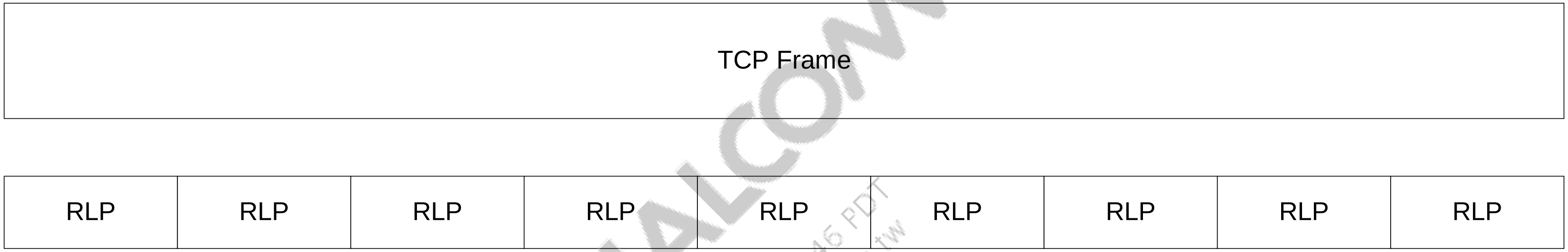
- CDMA is naturally lossy
  - Lost TCP frames
  - Many retransmissions – May retransmit 1500 bytes to recover only 30 bytes
  - A lot of backoff
  - TCP will either time out or transmission will be slow

Solution = RLP

- Reduces/mitigates lossy nature of CDMA
- Best-effort protocol
  - Tries a few times
  - Lets higher layers recover
- RLP frames are much smaller, 20 to 40 bytes
  - Retransmissions have finer granularity
- TCP sees fewer bad frames→TCP backs off less→data throughput

TCP vs RLP

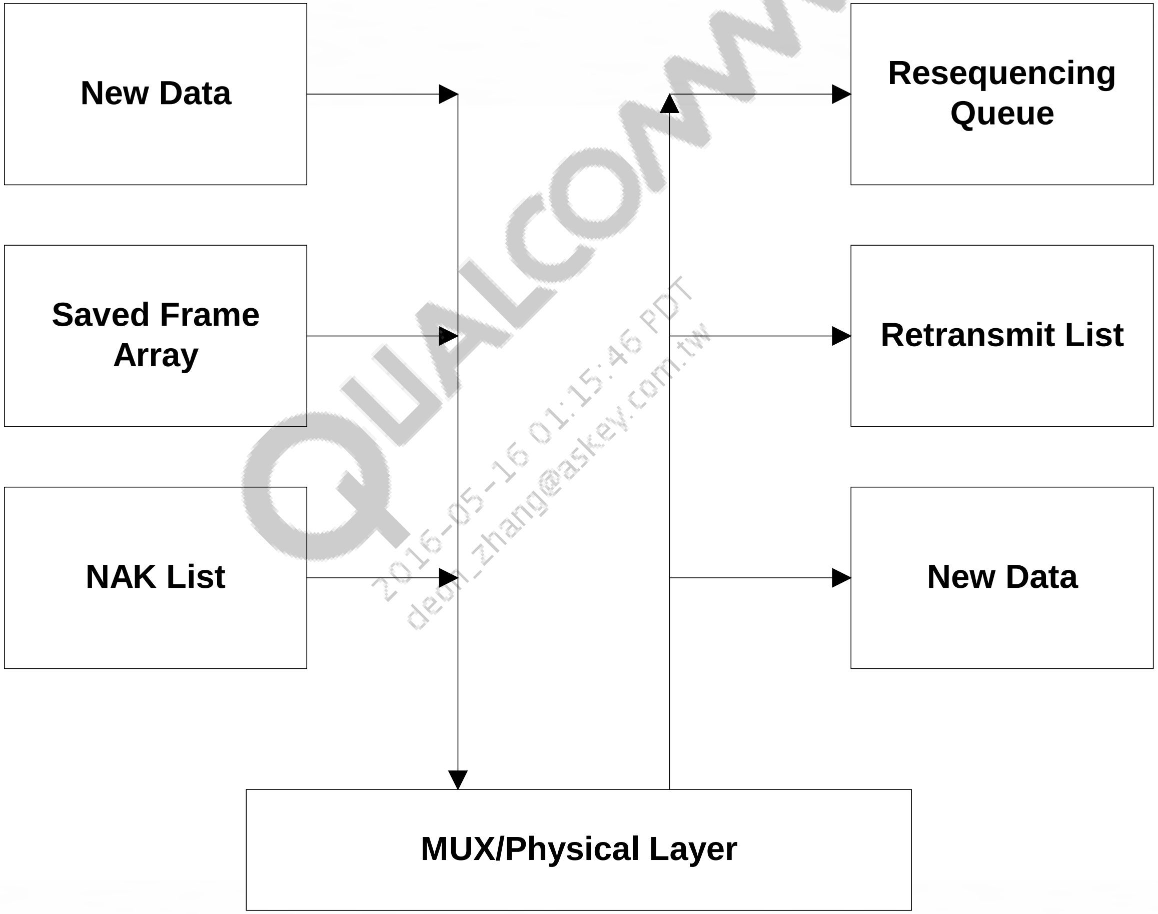
- Multiple RLP frames can make up a TCP frame



TCP Frame

RLP RLP RLP RLP RLP RLP RLP RLP RLP

RLP General Architecture



Radio Link Protocol (RLP)

- Link Layer Protocol on top of IS-95 with limited NAK-based ARQ mechanism
- Initially designed by Phil Karn – QTI-distributed free source code
- Runs on traffic channels; 20-ms frames
- Sync/ACK process is used to establish link and determine Round-Trip Time (RTT) between peers
- Repairs holes in the Rx stream within 3 RTTs (~ 400 ms) or gives up
- Uses channel capacity on an as-needed basis
- Does Rate Set 1 and Rate Set 2 (different SOs)
  - Maximum payload
    - rateset1 = 20 bytes ∗ 50 = 1000 bytes/sec = 8000 bps
    - rateset2 = 32 bytes ∗ 50 = 1600 bytes/sec = 12800 bps
- RLP 2 and 3 make use of SCHs to augment data throughput

RLP 1 and 2 Interfaces

- RLP1 (rlp.c, rlp.h) for IS-707 support only
- RLP2 (mdrrlp.?) for IS-707A MDR
  - Superset of RLP 1; can also do RLP 1
- Interface functions
  - rlp_init() – One-time initialization
  - rlp_reset() – Reset RLP state
  - rlp_establish_rate_set() – Invoked each time that a rate set change occurs
- A build either has RLP 1 or RLP 2 enabled

Medium Data Rate – RLP2

- IS-707A supports up to 8 forward and 8 reverse channels
- RLP produces 1+NUM_SUP frames every 20 ms
- 12-bit sequence numbers (L_SEQ), but only 8-bit SEQ transmitted
- Supplemental frames at either full rate or blank
- NAK scheme – 2 followed by 3 NAKs instead of 1, 2, 3
- New frame formats – Bitmap NAKs; NAK list
- New interfaces to RXC – To account for more channels
- MDR features
  - FEATURE_SPECIAL_MDR
  - FEATURE_IS95B_MDR

RLP 2 (MDR) Interfaces

- Interface functions
  - rlp_init() – One-time initialization
  - rlp_reset() – Reset RLP state
  - rlp_establish_rate_set() – Invoked each time a rate set change occurs
  - rlp_reg_srvc(tx_watermark, *post_rx_fn(), post_rx_q) – Used to configure RLP-DS data flow
  - rlp_establish_srvc_type(traffic_type, for_mux, rev_mux, rlp_type) – Called by MC after service negotiation
- During traffic channel operation

Tx

```
rlp_rate_enum_type
   rlp_tx_get_next_frame
(
byte                     **frame_ptr,
boolean                  primary_frame,
rlp_rate_enum_type       allowed_rate
);
```

Rx

```
rlp_rx_status_type rlp_rx_process_frames
(
rlp_fwd_frame_block_type *for_fr_blk
);
typedef struct{
  byte num_sup;
  struct {
    rlp_rate_enum_type frame_rate,
    dsm_item_type *rlp_frame_ptr
  } fwd_frames[ 1+ RLP_MAX_FWD_SUP];
} rlp_fwd_frame_block_type;
```

RLP3

- cdma2000 supports data rates of up to 2 Mbps
- RLP3 is designed to support the high data rates required by cdma2000 (currently 307 Kbps)
- Advantages
  - Flexibility of byte addressing with low overhead of frame addressing
  - As low an overhead as RLP1, RLP2
- RLP3 is used in RC3 and higher; for RC1 and RC2, RLP2 is used

What is New in RLP3

- Enlarged sequence number spaces – 24-bit sequence numbers
  - Flexibility of octet addressing, with low overhead of frame addressing
- Improved segmentation procedures (byte-addressed)
- Use of peer V(N)s to avoid retransmit ambiguity problem
- Negotiated NAK schemes
- Format C and format D frames for SCH
- New NAK formats

RLP3 Interfaces – dsrlp.c

- dsrlp_startup() – Startup function called once after phone is powered up; it initializes all RLP3 structures
- dsrlp_init() – Initializes the RLP3 session; it is called every time a data call is set up
- dsrlp_cleanup() – Cleans up the memory and resets the RLP3 state; should be called at the end of an RLP3 session
- dsrlp_process_rx_frames() – Processes the RLP3 frame(s) received over the air
- dsrlp_build_tx_frames() – Builds RLP3 frame(s) to be transmitted over the air
- dsrlp_build_rlp_blob() – Builds an RLP3 BLOB (block of bytes)
- dsrlp_process_rlp_blob() – Processes the received RLP3 BLOB
- dsrlp_get_stats() – Gets the statistics for an RLP3 session
- dsrlp_clear_stats() – Clears the statistics for an RLP3 session
- dsrlp_get_max_ms_nak_rounds_fwd() – Returns the maximum number of NAK rounds the mobile will support on the forward channel

RLP3 Interfaces – dsrlp.c (cont.)

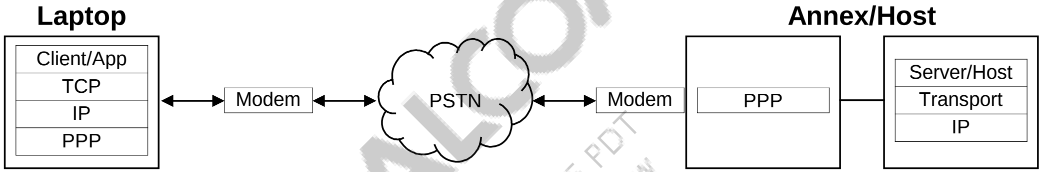
- dsrlp_get_max_ms_nak_rounds_rev() – Returns the maximum number of NAK rounds the mobile will support on the reverse channel
- dsrlp_reg_srvc() – Registers services, i.e., watermarks and queues/callbacks, for an RLP3 session
- dsrlp_dne_mem_event() – Called when the do not exceed watermark is reached for small items
- dsrlp_high_mark_mem_event() – Called when the high watermark is reached for small items

Filenames

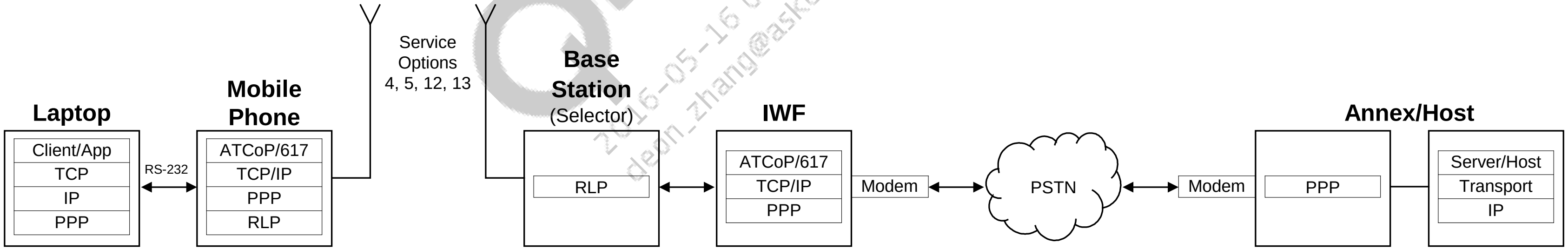
- ds707_pkt_mgr.c/.h – Main control file for IS-707 packet-switched data
- ds707*.c/.h – IS-707 call control files for specific IS-707 functionality
- rlp.c/h – RLP I
- mdrrlp.c/.h – MDR/RLP II
- dsrlp*.c/.h – RLP 3 (1X)

cdma2000 Circuit-Switched Data Services Overview

- Objective – Provide a data service that emulates landline modem data capabilities



Landline Model



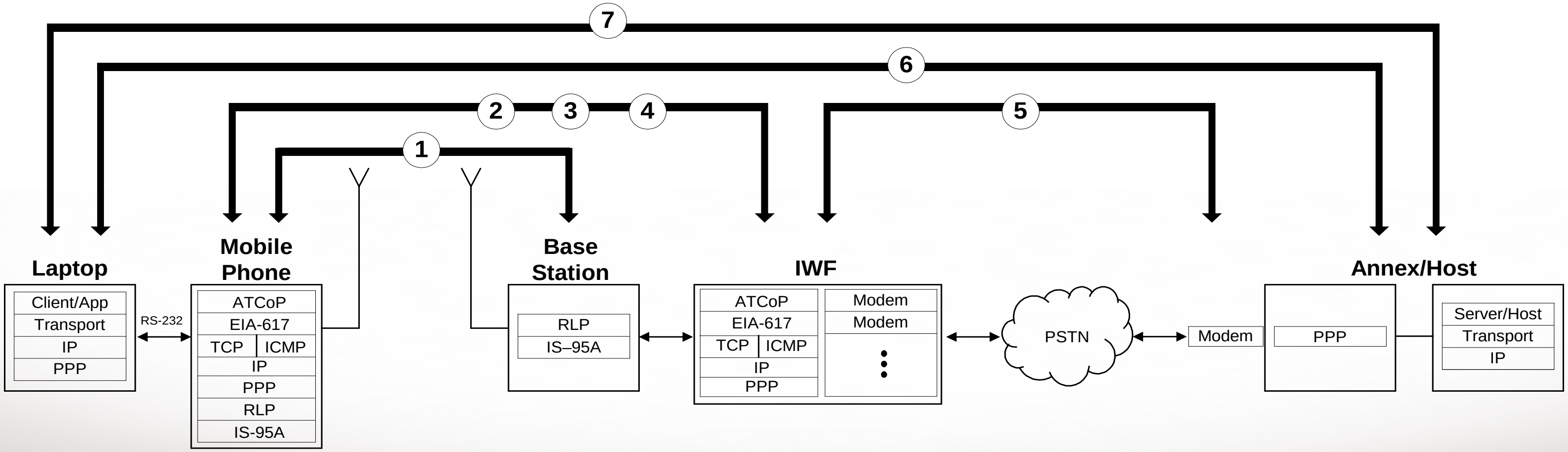
CDMA Async/G3 Fax Model

cdma2000 Circuit-Switched Data Services – SOs, Rates, Etc.

- Async and fax included under 707 circuit-switched data services
- SOs used
  - Async – 4, 4100, 12, 0x8021
  - Fax – 5, 4101, 13, 0x8022
- Supported rates
  - Rateset1 – 9.6 Kbps
  - Rateset2 – 14.4 Kbps
- Standards
  - See [S7]
  - See [S8]

cdma2000 Circuit-Switched Data Services – Call Flow

1. RLP Sync – Occurs once Traffic Channel Conversation substate is reached; takes ~ 6 to 8 frames (120 to 160 ms)
2. Async PPP Neg – Negotiates IP and Link parameters; takes 3 or 4 round trips (~ 500 ms or less)
3. Async TCP Establish – Establishes TCP connection from mobile to IWF (~ 1 sec)
4. AT Initialization Command Exchange – Mobile forwards AT parameter commands to IWF (~ 500 ms)
5. PSTN Modem Training – Negotiates landline vs protocols, including bandwidth; takes 10 or more sec
6. Serial End-to-End PPP Neg – Negotiates IP and Link parameters, including TE2 IP address (~ 500 ms or less)
7. End-to-End TCP Sync – Establishes end-to-end connection (~ 2 sec)



cdma2000 Circuit-Switched Data Services – Mode Handler Details

- Two main functions
  - Call control
  - Data transfer over protocol stack (TCP/IP/PPP/RLP)
- Executes in DS and PS tasks
- Interaction with ps_iface
  - Separate interface for 707 circuit-switched data and 707 packet data
  - Both async data and fax under 707 circuit-switched mode handler
  - Controls both Rm (SIO link) and Um (air link) (different from 707 packet where separate units for Rm and Um)
  - Before coming up, ensures packet call not already up, disables packet interface
- Interaction with SIO, ATCoP
  - Client of SIOLIB and ATCoP, registers callbacks with them
  - Flow controls SIO, (de)asserts CD, registers for DTR, Esc, etc.
  - During data call, Esc sequence +++ leads to Online Command mode in which AT commands are sent to the IWF and reflected
- Interaction with 3gdsmgr
  - Client of 3gdsmgr for traffic-channel-related functionality, i.e., call origination, teardown, etc.

cdma2000 Circuit-Switched Data Services – Mode Handler Details (cont.)

- Interaction with the protocol stack
  - Uses full data protocol stack on phone (TCP/IP/PPP/RLP)
  - Talks to TCP/PPP directly without going through sockets
  - Has special TCB for async connection with reserved port numbers for active and passive opens
  - Configures PPP options
  - Configures RLP watermarks
- 617 layer
  - In-band communication between mobile and IWF
  - Used in Online Command mode to facilitate AT command reflection between IWF and mobile for providing modem functionality

cdma2000 Circuit-Switched Data Services – Fax

- Supports FCLASS 2.0 standard
- Standards
  - See [S9]
  - See [S10]
- During mobile origination, Circuit-Switched mode handler decides whether to originate with async or fax SO by looking at AT+FCLASS value
- Most other functionality and code in mobile is common between async data and fax

cdma2000 Circuit-Switched Data Services – Incoming Calls

- 3gdsmgr notifies mode handler of incoming page
- Mode handler makes sure packet call is not already up; if not, proceed; else, reject the page
- Mode handler rings the laptop
- If the user answers (ATA) or there is autoanswer, the mode handler asks 3gdsmgr to accept the call
- Async and fax work the same way

cdma2000 Circuit-Switched Data Services – File Layout

- ds707_async_mgr.c/.h
- ds707_async_timer.c/.h
- ds707_async_wmk.c/.h
- ds707_async_defs.h
- ds707_async_617.h
- ds707_so_async.c/.h
- ps707_async.c/.h

1xEV-DO

1xEV-DO

- Standards Overview
- RLP Overview
- Packet Application (RLP) Bound to Service Network (SN)
- Packet Application (RLP) Bound to Access Network (AN)
- Data Session Handoff Between 1X and HDR
- Files/More Resources

1xEV-DO Data Services Overview

- IS-856 is the 1xEV-DO air interface standard; it defines the RLP used.
- IS-856 Data Services is integrated into the existing IS-707 DMSS Packet Data Services.
- The RLP is part of the Packet application.
- IS-856 has a concept of streams. Each RLP runs on a particular stream and the stream is bound for a particular application subtype.
  - Packet application bound to the SN
  - Packet application bound to the AN

RLP Overview

- The purpose is to reduce the radio link error rate, as seen by higher layer protocols.
- Sequence numbers are 22 bits in length and are octet-addressable.
- RLP control packets are treated as separate signaling messages.
- A flush packet is sent out at the end of a data burst to detect potentially missing data and to create a NAK.
- Control messages are specified as signaling messages.
- There are three types of messages, Reset, resetAck, and NAK
- The initialization procedures require no synchronization.
- It uses a fixed default NAK timeout (abort) of 500 ms.
- The RLP protocol states are reset or established.

RLP Implementation

- Reuses much of the existing DSRLP framework
  - Common data structures and accessor functions for the resequencing queue, Tx queue, and NAK list
- RLP registers Rx and Tx functions for a particular stream with PCP at initialization
  - Registered callbacks hdrrlp_get_pkt_cb() and hdrrlp_put_pkt_cb() are called by PCP for the Tx and Rx bytes
- Tx signaling messages routed directly to SLP (bypassing HMP)
- An internal message queue is provided for HMP to queue all received RLP signaling messages directly to RLP through the callback mechanism
- For reverse link (Tx) RLP signaling and data, packets are created in the following priority order:
  - Retransmit data
  - New data
  - Signaling messages

RLP Implementation (cont.)

- A fail safe RLP Reset timer is provided to prevent deadlock in case RLP signaling messages are lost during reset procedures. If RLP is in reset for more than 1.5 sec, the connection is terminated.
- Files
  - hdrrlp.c, hdrrlp.h
  - hdrrlpi.c, hdrrlpi.h
- Existing files from DSRLP framework that have changed
  - dsrlp*.c, dsrlp*.h

RLP Interfaces

- Common interfaces, i.e., RLP3, are used along with some new interfaces specific to HDR
- dsrlp_startup() – Startup function called once after phone is powered up; it initializes all RLP structures
- dsrlp_reg_srvc() – Registers services, i.e., watermarks and queues/callbacks, for an RLP session
- dsrlp_init() – Initializes the RLP session; called every time a data call is originated from NULL state
- dsrlp_cleanup() – Cleans up the memory and resets the RLP state; called when the data session is torn down completely (goes to NULL state)
- hdrrlp_resume() – Called when the data call gets connected; starts RLP timers
- hdrrlp_suspend() – Called when the call is torn down (goes dormant); stops RLP timers and resets the RLP

Mode-Specific Handler for SN – Packet Application Bound to the SN

- The RLP running on this stream carries user data.
- The call control path follows from the 1X packet call control. Implementation is embedded in the implementation of the 1X packet call control, thus, it runs in the DS task and uses the DS command queue.
- It uses a separate RLP control block other than 1X.
- The RLP is set up and initialized whenever a packet data call is originated. Eventually, the call may get connected on 1X or HDR. If the call connected is 1X, the HDR RLP will not Rx/Tx any packets.
- When an HDR call is connected, the HDR RLP timers are started by calling hdrrlp_resume().
- When the call goes dormant, the RLP timers are suspended and the RLP is reset by calling hdrrlp_suspend().
- When the data session is torn down, RLP is cleaned up completely.
- An incoming HDR page is always accepted since it could be either for user data, authentication, or protocol signaling.

Packet Application Bound to the Access Network

- The RLP running on this stream carries data used for authenticating the access terminal.
- IS-878 defines the interoperability specification for HRPD access network interfaces, including authentication procedures that place requirements on access terminals.
- CHAP is used for authentication.
- DMSS supports multiple, simultaneous RLP instances, one for supporting traffic on the packet application bound to the AN and one for traffic on the packet application bound to the SN.
- The AN is expected to negotiate AN stream as part of 1xEV session configuration for initiating authentication procedures. AT is completely passive, i.e., will not initiate these procedures but will respond to them.
- The AN stream is used only for CHAP authentication; no IP traffic is supported on this stream.
- The AT does not maintain authentication status.

Packet Application Bound to the Access Network (cont.)

- The module managing the authentication is called DS HDR AN manager.
  - It runs in the DS task and uses the DS command queue.
  - This is a client of CM; even though it runs in the DS context, it is not used in the 3GDSMGR interface to CM.
  - It registers for Call Connected and Call End events from CM.
- RLP management
  - RLP is set up and initialized when the HDR session is opened.
  - When an HDR traffic channel is connected, the RLP is resumed.
  - When the HDR traffic channel is closed, the RLP is suspended.
  - RLP is cleaned up when the HDR session is closed.
- PPP management
  - The module initializes PPP in Passive mode. LCP negotiation will not be started until the network initiates it. IPCP is disabled so that IPCP negotiation will be rejected. CHAP is used for authentication.
  - PPP is initialized whenever the HDR session is negotiated with a stream for authentication.
  - If the HDR session is closed, the PPP is torn down.
  - If the PPP interface goes down for any other reason, it is reinitialized.

Data Session Handoff

- A packet data session can be handed off between 1X and HDR systems.
- 3gdsmgr treats 1X and HDR as the same. The 707 mode-specific handler keeps track of whether the current data session is on 1X or HDR.
- On getting a call connected from CM, the current data session is set to 1X or HDR, depending on the SO.
- If HDR is acquired or lost during dormancy, CM sends an Idle Digital Mode Changed command (routed as part of SS_PREF changed command) to DS. If the data session needs to be moved from 1X to HDR or from HDR to 1X, DS-707 will send an origination with DRS = 0 to notify the network and then update the current data session network it maintains.
- In MIP calls, if DS detects that the MS needs to resync PPP, DS will send an origination with DRS = 1 on a dormant data session handoff and initiate PPP resync when the call is connected.
- DS will process SID/NID/PZID changed commands and SOCMs only if the dormant data session is on 1X.

More Resources and Files

- Refer to Appendix B of this document for basic HDR data call flows.
- For more information on 1xEV-DO data services, see [Q4].
- Files
  - Packet application bound to the SN
    - ds707_pkt_mgr.c
    - ds707_rmsm.c
  - Packet application bound to the AN
    - dshdr_an_mgr.c/.h
    - dshdr_an_wmk.c/.h

UMTS

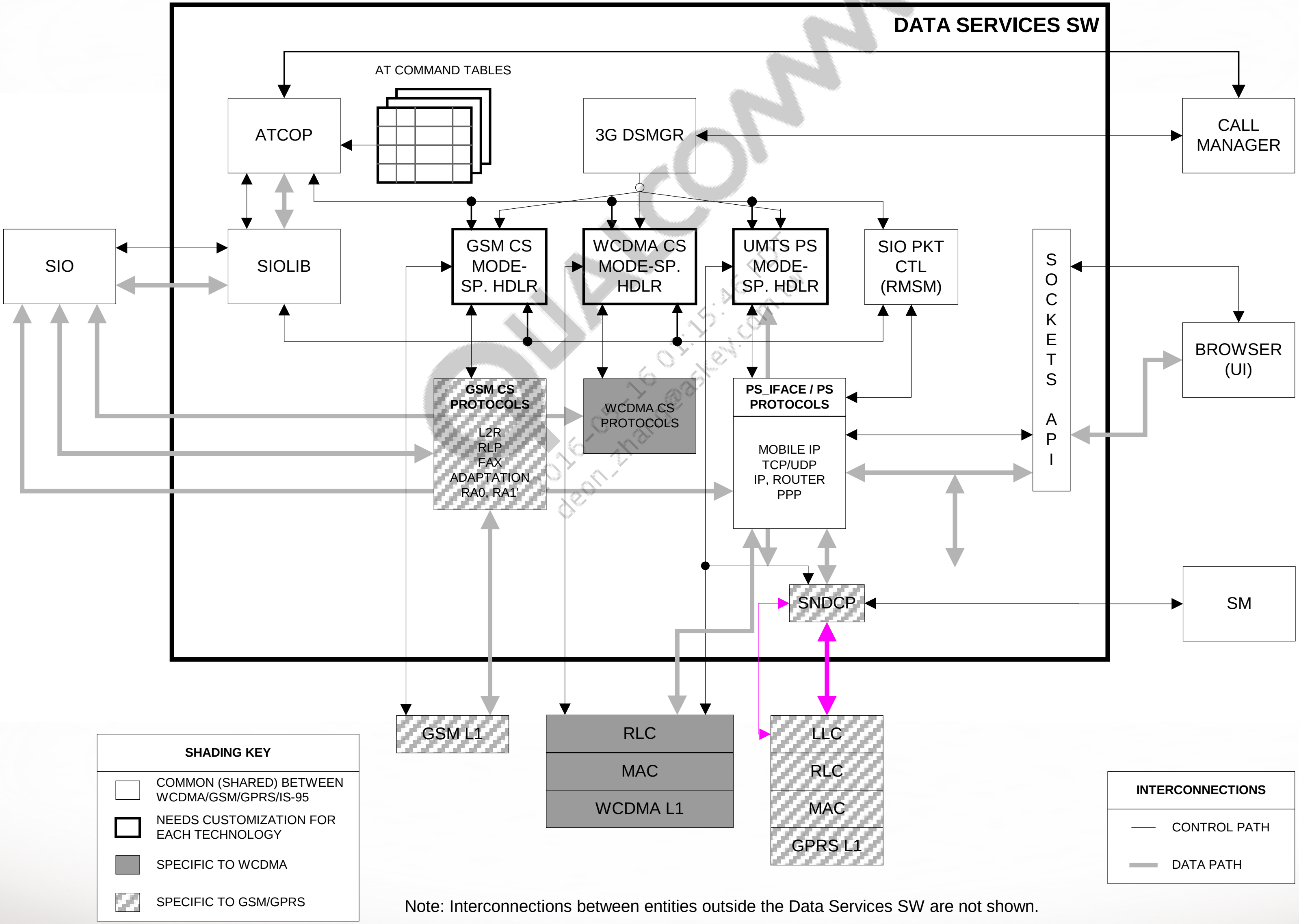
UMTS

- Overview
- UMTS Data Architecture
- Circuit Services
- Packet Services
- Feature Usage

UMTS Data Services Overview

- UMTS Data Services software currently supports:
  - WCDMA Circuit-Switched (CS) data
    - Synchronous data using the transparent connection type
    - Asynchronous data using the nontransparent connection type
  - GSM CS data
    - 9600 bps and 14400 bps data rates
    - Transparent and nontransparent connection types
    - V.42bis compression (nontransparent only)
  - GSM transparent fax
    - 7200 bps and 9600 bps data rates
  - WCDMA packet data
    - PDP type PPP using transparent PPP negotiation (TE+MT model only)
    - PDP type IP (both TE+MT model and MT-only model, i.e., embedded Sockets app)
  - GPRS
    - PDP type IP (both TE+MT model and MT-only model, i.e., embedded Sockets app)

UMTS Data Architecture Overview

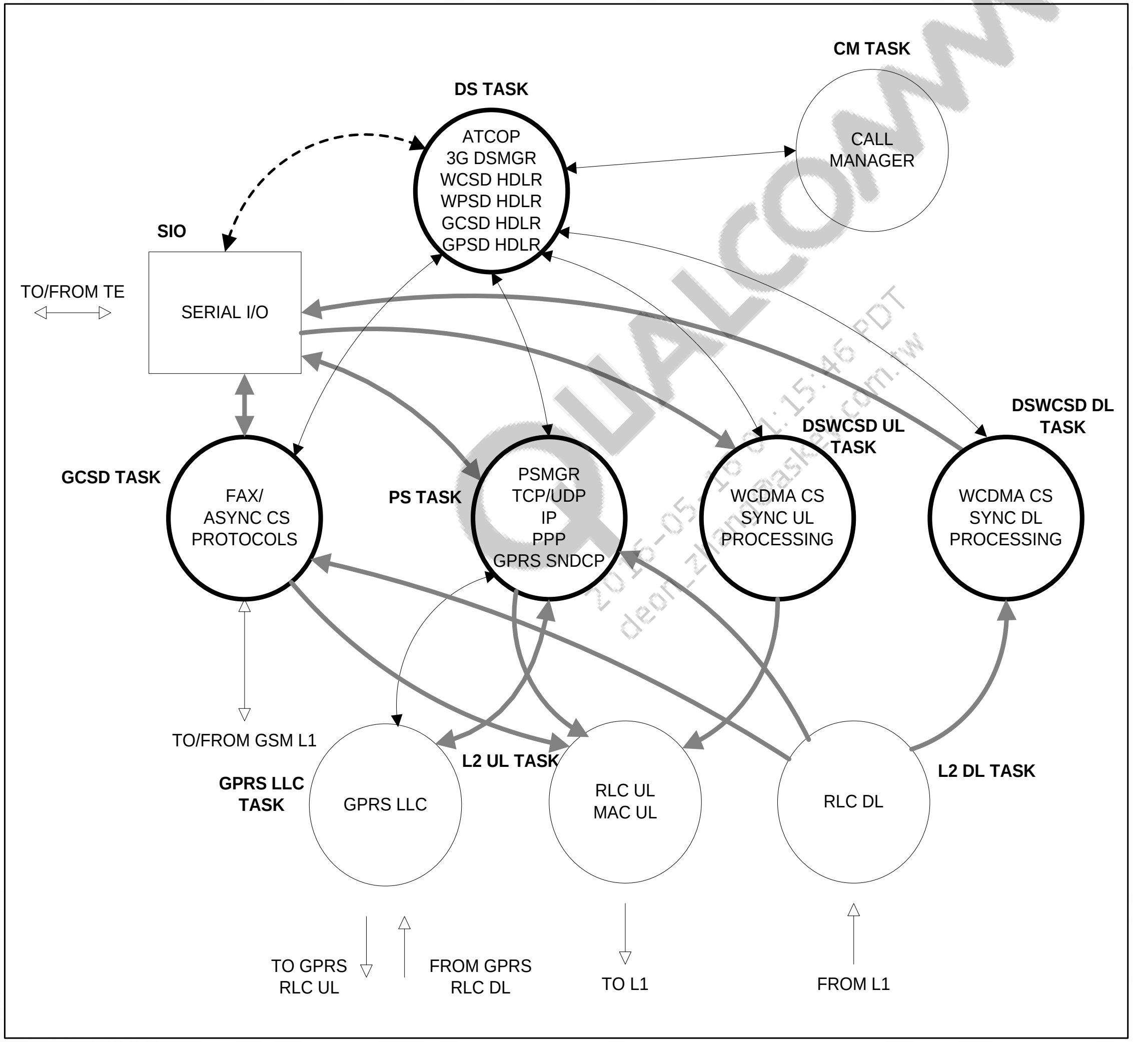


Note: Interconnections between entities outside the Data Services SW are not shown.

UMTS Data Software Components

- UMTS CS mode-specific handler
  - Mode-specific call control for WCDMA/GSM CS data and GSM fax calls
  - Interfaces to 3G DSMgr, SIOLIB, ATCoP, WCDMA RLC, and GSM L1
  - Specifies/validates/negotiates bearer-capability parameters; configures CS and fax protocols
- UMTS CS protocols
  - Transparent and nontransparent 3.1 kHz and UDI protocols
  - T.31 and GSM 03.45 protocols (for fax)
- UMTS PS mode-specific handler includes:
  - Mode-specific call control common to WCDMA PS and GPRS
    - Specifies/validates PDP context parameters including QoS, configures PS protocols
  - WCDMA PS-specific call control for interfacing to RLC
  - GPRS-specific call control for interfacing to SNDCP
- SIO packet controller (RMSM)
  - Generic Rm interface state machine
  - Manages the Um and Rm interfaces for packet data calls initiated from the TE
  - Interfaces to SIOLIB, ATCoP, PPP, and PS Interface
  - Controls the serial port during a packet data call

Task Architecture



ACRONYMS

ATCOP	- AT COMMAND PROCESSOR
CM	- CALL MANAGER
CS	- CIRCUIT SWITCHED
DL	- DOWNLINK
DS	- DATA SERVICES
DSMGR	- DATA SERVICES MANAGER
GCSD	- GSM CIRCUIT-SWITCHED DATA
GPSD	- GPRS PACKET-SWITCHED DATA
IP	- INTERNET PROTOCOL
L1	- LAYER 1
L2	- LAYER 2
LLC	- LOGICAL LINK CONTROL
MAC	- MEDIUM ACCESS CONTROL
PPP	- POINT-TO-POINT PROTOCOL
PS	- PROTOCOL SERVICES
PSMGR	- PROTOCOL SERVICES MANAGER
RLC	- RADIO LINK CONTROL
SIO	- SERIAL INPUT/OUTPUT
SNDCP	- SUBNETWORK DEPENDENT CONVERGENCE PROTOCOL
TCP	- TRANSMISSION CONTROL PROTOCOL
UDP	- USER DATAGRAM PROTOCOL
UL	- UPLINK
WCSD	- WCDMA CIRCUIT-SWITCHED DATA
WPSD	- WCDMA PACKET-SWITCHED DATA

KEY

○	TASK
→	INTERTASK MESSAGES
--	DATA FLOW (IDLE)
➔	DATA FLOW (IN CALL)

UMTS Data Task Architecture

- All mode-specific handlers (CS, PS) execute in the context of the DS task.
- RLC processing (for both CS and PS Data Services) occurs in the L2 UL and L2 DL tasks.
- GPRS RLC processing occurs in the GPRS RLC UL and GPRS RLC DL tasks.
- GPRS LLC processing occurs in the GPRS LLC task.
- For PS services:
  - Packet data protocol processing, e.g., PPP/IP/TCP, etc., occurs in the PS task.
  - GPRS SNDCP processing occurs in the PS task.
- For CS services, additional tasks are needed.

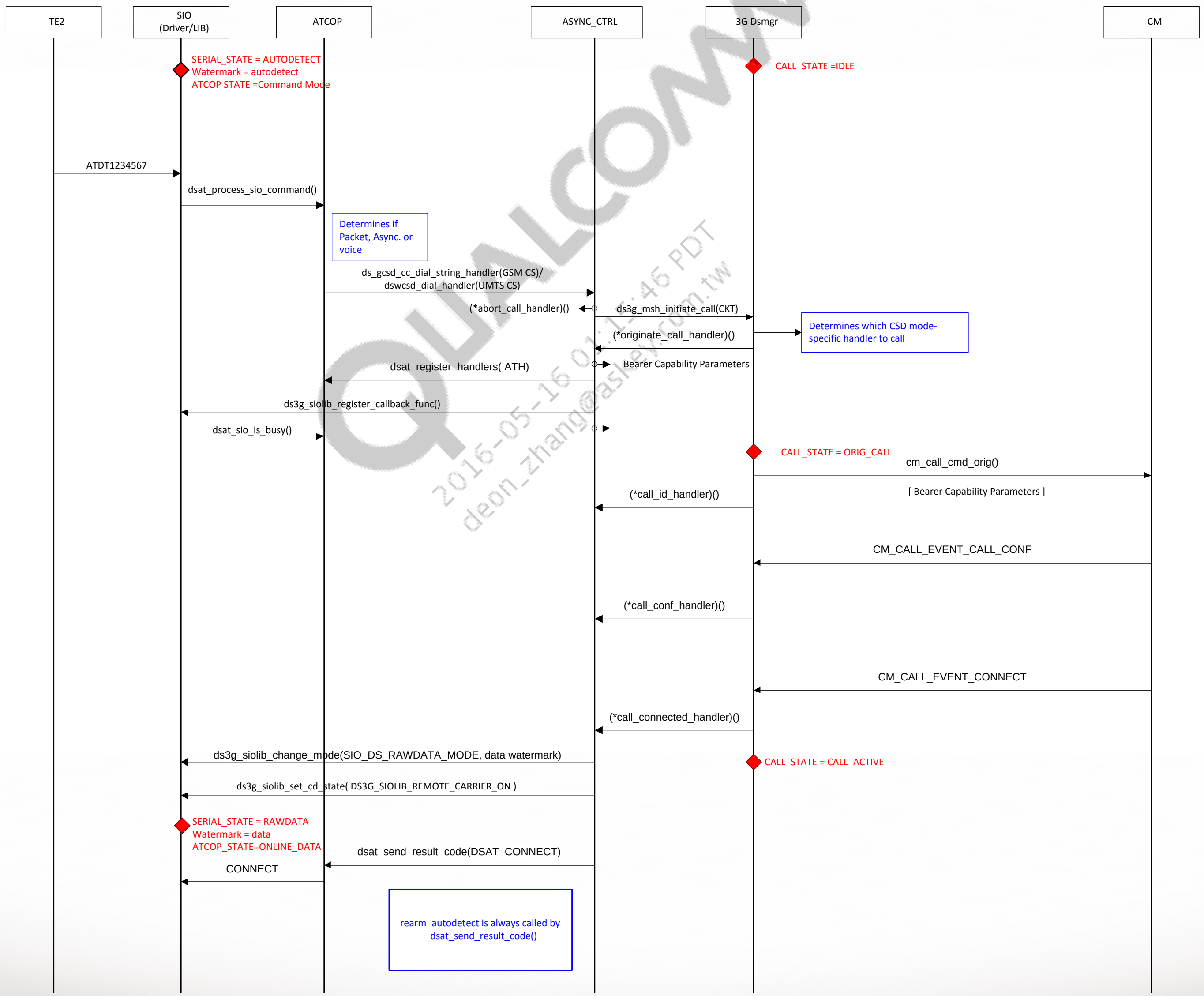
WCDMA CS Data Tasks

- DSWCSD UL task
  - Uplink protocol processing for WCDMA synchronous/transparent UDI
  - Gets data from SIO, builds SDUs, and sends SDUs to RLC for transmission on uplink
- DSWCSD DL task
  - Downlink protocol processing for WCDMA synchronous/transparent UDI
  - Gets downlink SDUs from RLC and sends data to SIO

GSM CS Data Task

- Performs both uplink and downlink processing for GSM and WCDMA asynchronous data calls
- Performs RA0/RA1 protocol processing for GSM transparent data calls
- Performs L2RCOP/RLP protocol processing for both GSM and WCDMA nontransparent data calls
- Performs fax protocol processing, which utilizes transparent CS protocol for fax data transport

MO CS Data Call Setup



Mobile-Originated Circuit-Switched Data Call

TE2
SIO (Driver/LIB)
ATCOP
ASYNC_CTRL
3G Dsmgr
CM

SERIAL_STATE = AUTODETECT
Watermark = autodetect
ATCOP STATE =Command Mode

CALL_STATE =IDLE

ATDT1234567

dsat_process_sio_command()

Determines if Packet, Async. or voice

ds_gcsd_cc_dial_string_handler(GSM CS)/
dswcsd_dial_handler(UMTS CS)

(*abort_call_handler)()

ds3g_msh_initiate_call(CKT)

(*originate_call_handler)()

Determines which CSD mode-specific handler to call

Bearer Capability Parameters

dsat_register_handlers(ATH)

ds3g_siolib_register_callback_func()

dsat_sio_is_busy()

CALL_STATE = ORIG_CALL

cm_call_cmd_orig()

[Bearer Capability Parameters]

(*call_id_handler)()

CM_CALL_EVENT_CALL_CONF

(*call_conf_handler)()

CM_CALL_EVENT_CONNECT

(*call_connected_handler)()

ds3g_siolib_change_mode(SIO_DS_RAWDATA_MODE, data watermark)

CALL_STATE = CALL_ACTIVE

ds3g_siolib_set_cd_state(DS3G_SIOLIB_REMOTE_CARRIER_ON)

SERIAL_STATE = RAWDATA
Watermark = data
ATCOP_STATE=ONLINE_DATA

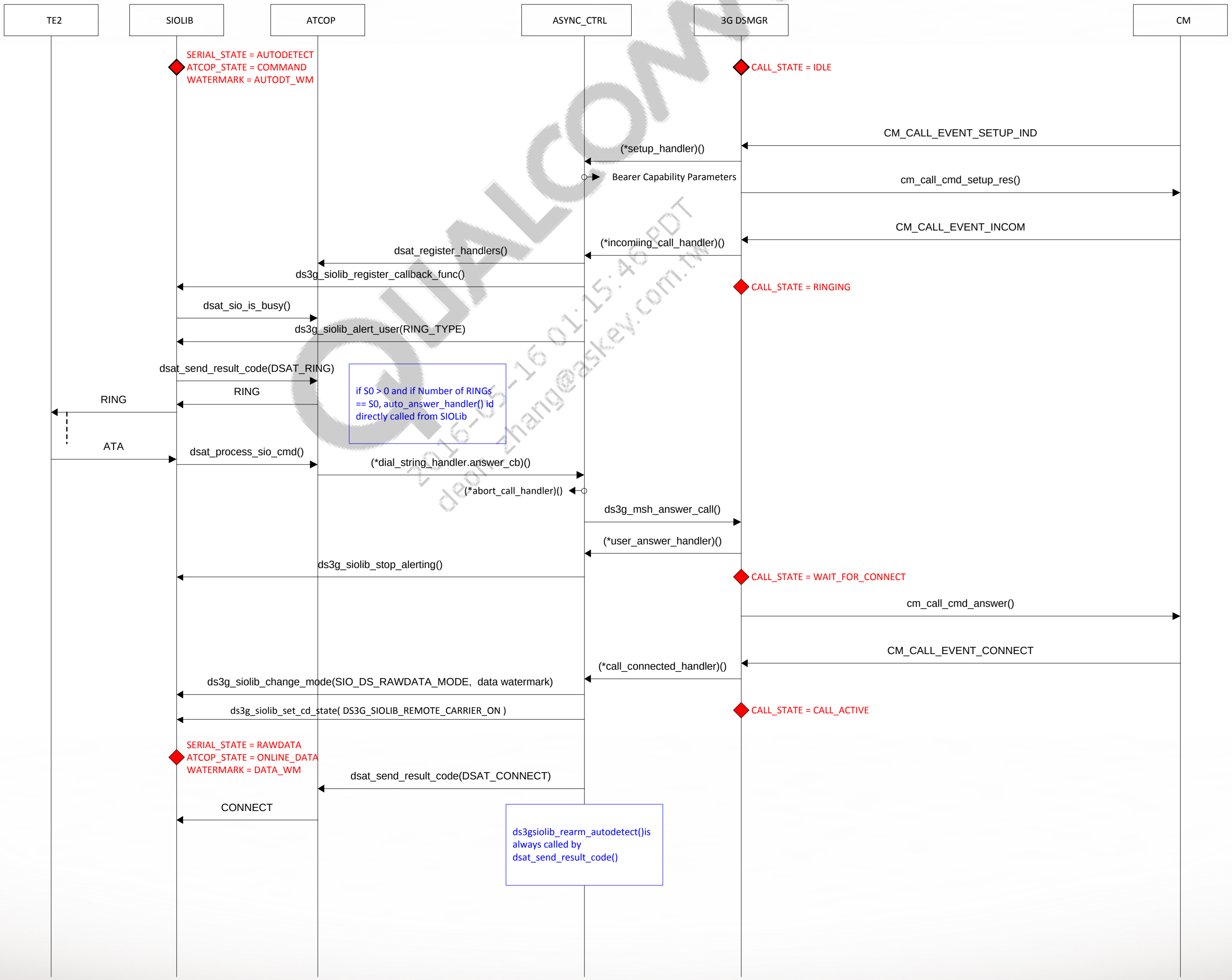
dsat_send_result_code(DSAT_CONNECT)

CONNECT

rearm_autodetect is always called by dsat_send_result_code()

MT CS Data Call Setup

Mobile-Terminated Circuit-Switched Data Call



TE2

SIOLIB

ATCOP

ASYNC_CTRL

3G DSMGR

CM

SERIAL_STATE = AUTODETECT
ATCOP_STATE = COMMAND
WATERMARK = AUTODT_WM

CALL_STATE = IDLE

CM_CALL_EVENT_SETUP_IND

(*setup_handler)()

Bearer Capability Parameters

cm_call_cmd_setup_res()

CM_CALL_EVENT_INCOM

(*incomiing_call_handler)()

dsat_register_handlers()

ds3g_siolib_register_callback_func()

CALL_STATE = RINGING

dsat_sio_is_busy()

ds3g_siolib_alert_user(RING_TYPE)

dsat_send_result_code(DSAT_RING)

RING

RING

if S0 > 0 and if Number of RINGs == S0, auto_answer_handler() id directly called from SIOLib

ATA

dsat_process_sio_cmd()

(*dial_string_handler.answer_cb)()

(*abort_call_handler)()

ds3g_msh_answer_call()

(*user_answer_handler)()

ds3g_siolib_stop_alerting()

CALL_STATE = WAIT_FOR_CONNECT

cm_call_cmd_answer()

CM_CALL_EVENT_CONNECT

(*call_connected_handler)()

ds3g_siolib_change_mode(SIO_DS_RAWDATA_MODE, data watermark)

ds3g_siolib_set_cd_state(DS3G_SIOLIB_REMOTE_CARRIER_ON)

CALL_STATE = CALL_ACTIVE

SERIAL_STATE = RAWDATA
ATCOP_STATE = ONLINE_DATA
WATERMARK = DATA_WM

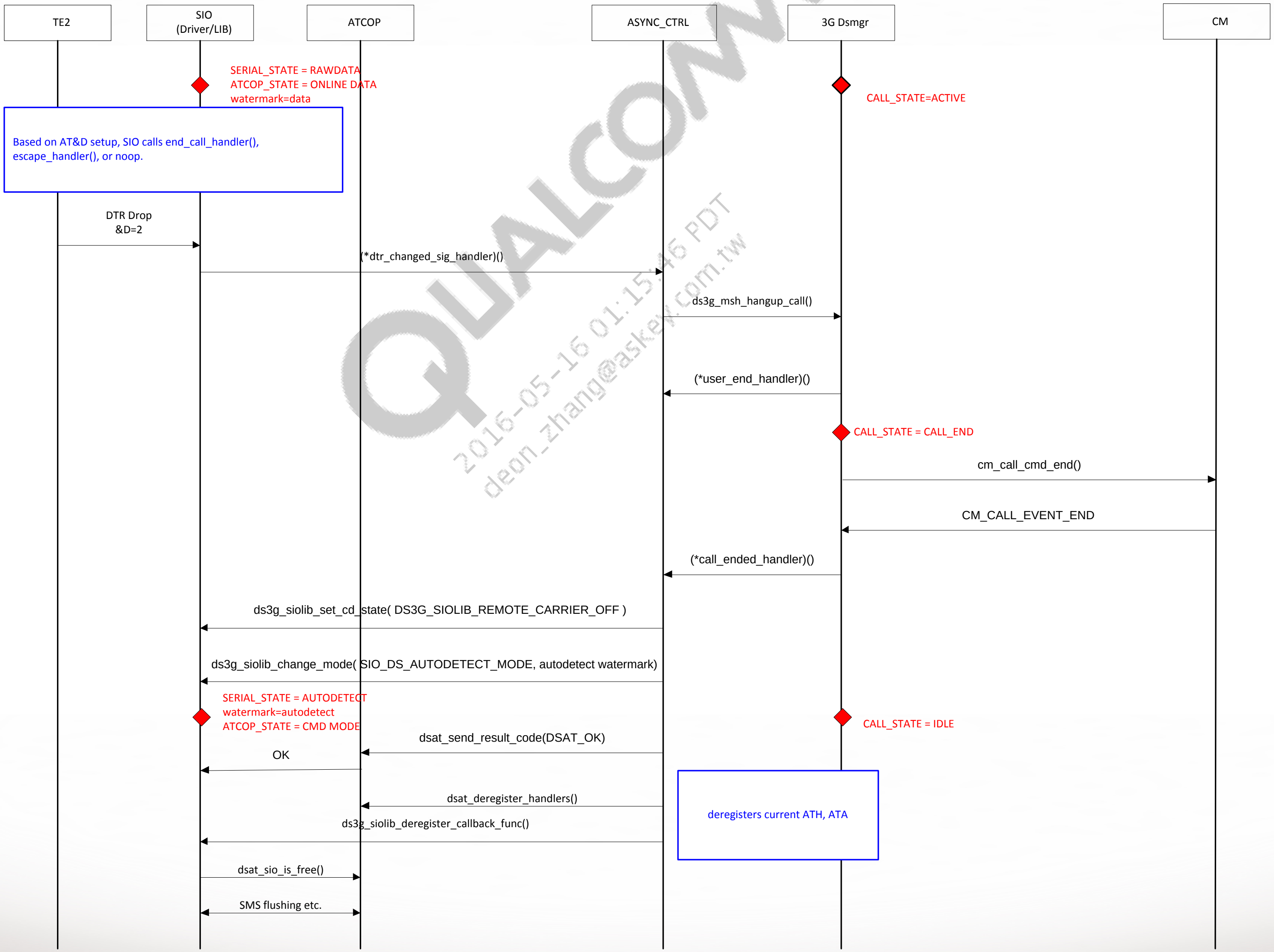
dsat_send_result_code(DSAT_CONNECT)

CONNECT

ds3gsiolib_rearm_autodetect()is always called by dsat_send_result_code()

Call Release by UE (DTR Drop)

CS Data Call Release by Mobile (DTR drop)



TE2

SIO (Driver/LIB)

ATCOP

ASYNC_CTRL

3G Dsmgr

CM

SERIAL_STATE = RAWDATA
ATCOP_STATE = ONLINE DATA
watermark=data

CALL_STATE=ACTIVE

Based on AT&D setup, SIO calls end_call_handler(), escape_handler(), or noop.

DTR Drop
&D=2

(*dtr_changed_sig_handler)()

ds3g_msh_hangup_call()

(*user_end_handler)()

CALL_STATE = CALL_END

cm_call_cmd_end()

CM_CALL_EVENT_END

(*call_ended_handler)()

ds3g_siolib_set_cd_state(DS3G_SIOLIB_REMOTE_CARRIER_OFF)

ds3g_siolib_change_mode(SIO_DS_AUTODETECT_MODE, autodetect watermark)

SERIAL_STATE = AUTODETECT
watermark=autodetect
ATCOP_STATE = CMD MODE

CALL_STATE = IDLE

dsat_send_result_code(DSAT_OK)

OK

dsat_deregister_handlers()

deregisters current ATH, ATA

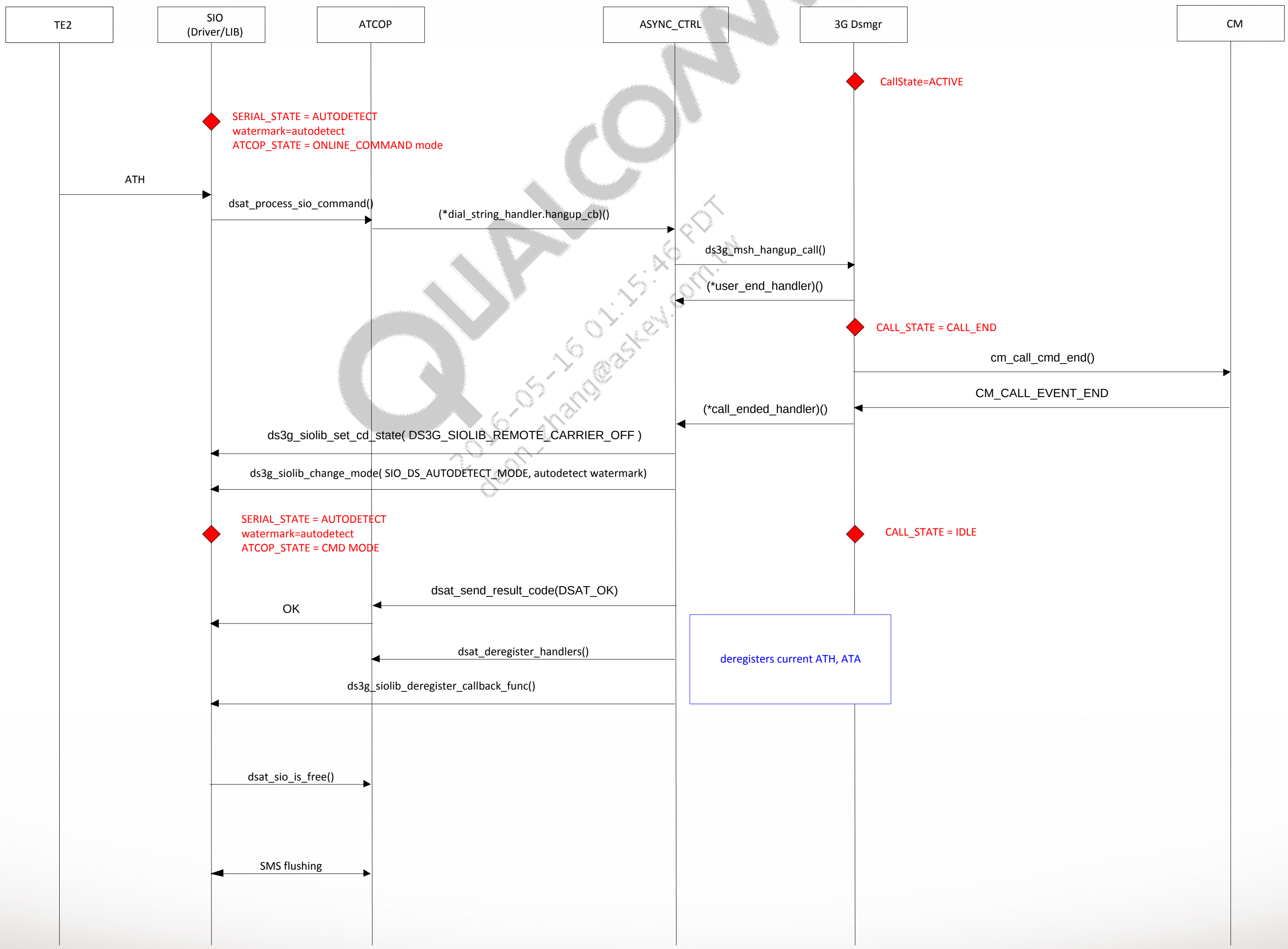
ds3g_siolib_deregister_callback_func()

dsat_sio_is_free()

SMS flushing etc.

Call Release by UE (ATH)

CS Data Call Release by Mobile (ATH)



TE2

SIO (Driver/LIB)

ATCOP

ASYNC_CTRL

3G Dsmgr

CM

CallState=ACTIVE

SERIAL_STATE = AUTODETECT
watermark=autodetect
ATCOP_STATE = ONLINE_COMMAND mode

ATH

dsat_process_sio_command()

(*dial_string_handler.hangup_cb)()

ds3g_msh_hangup_call()

(*user_end_handler)()

CALL_STATE = CALL_END

cm_call_cmd_end()

CM_CALL_EVENT_END

(*call_ended_handler)()

ds3g_siolib_set_cd_state(DS3G_SIOLIB_REMOTE_CARRIER_OFF)

ds3g_siolib_change_mode(SIO_DS_AUTODETECT_MODE, autodetect watermark)

SERIAL_STATE = AUTODETECT
watermark=autodetect
ATCOP_STATE = CMD MODE

CALL_STATE = IDLE

dsat_send_result_code(DSAT_OK)

OK

dsat_deregister_handlers()

deregisters current ATH, ATA

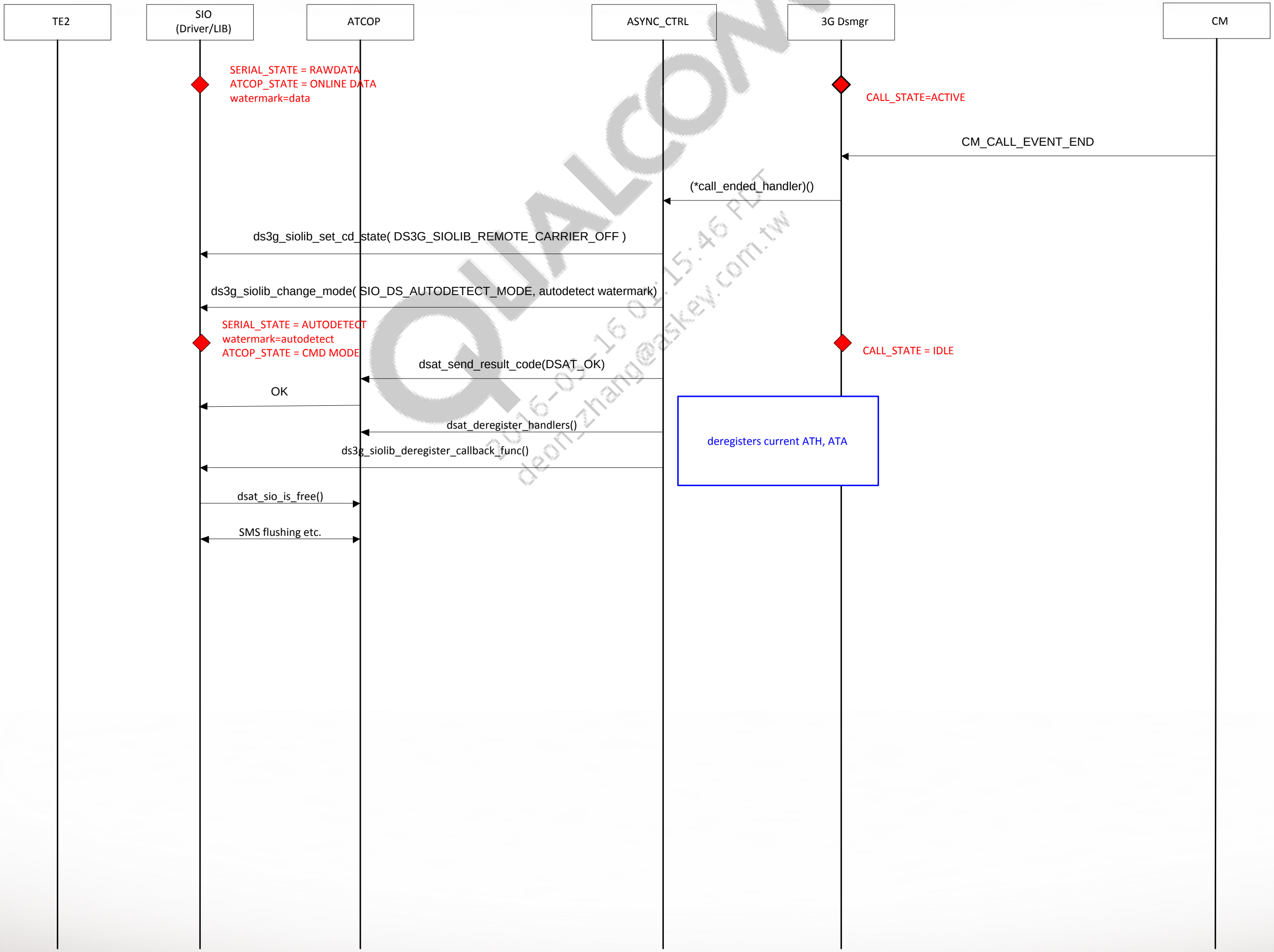
ds3g_siolib_deregister_callback_func()

dsat_sio_is_free()

SMS flushing

CS Call Release by Network

CS Data Call Release by Network
(or User Presses the End Key)


TE2
SIO
(Driver/LIB)
ATCOP
ASYNC_CTRL
3G Dsmgr
CM
SERIAL_STATE = RAWDATA
ATCOP_STATE = ONLINE DATA
watermark=data
CALL_STATE=ACTIVE
CM_CALL_EVENT_END
(*call_ended_handler)()
ds3g_siolib_set_cd_state(DS3G_SIOLIB_REMOTE_CARRIER_OFF)
ds3g_siolib_change_mode(SIO_DS_AUTODETECT_MODE, autodetect watermark)
SERIAL_STATE = AUTODETECT
watermark=autodetect
ATCOP_STATE = CMD MODE
CALL_STATE = IDLE
dsat_send_result_code(DSAT_OK)
OK
dsat_deregister_handlers()
deregisters current ATH, ATA
ds3g_siolib_deregister_callback_func()
dsat_sio_is_free()
SMS flushing etc.

Supported CS Data Configurations

	Radio Access		Information Transfer Capability				Rate Adaption					Attributes						
										H.223/		Trans/	Sync/	FNUR	WAIUR	Modem	V.42 Data	
+CBST Setting	GSM	UMTS	3.1kHz	UDI	RDI	Fax	V.110	V.120	X.31 FS	H.245	None	Ntrans	Async	[kbps]	[kbps]	Type	Compression	Comment
134,1,0		✓		✓						✓		T	S	64.0	Undef	None		Multimedia
116,1,0		✓		✓							✓	T	S	64.0	Undef	None		Bit Transparent
7,1,0	✓		✓			✓					✓	T	S	Undef	Undef	None		Group3 Class1
84,0,1		✓		✓					✓			NT	A	64.0	57.6	None	✓	Frame Tunneling Mode (FTM)
83,x,1		✓		✓	✓				✓			NT	A	56.0	57.6	None	✓	FTM, x: 0=UDI, 4=RDI
81,0,1		✓		✓			✓					NT	A	38.4	57.6	None	✓	
80,0,1		✓		✓			✓					NT	A	28.8	28.8	None	✓	
75,0,1	✓	✓		✓			✓					NT	A	14.4	14.4	None	✓	
71,0,1	✓			✓			✓					NT	A	Undef	Undef	None	✓	
51,0,1		✓		✓				✓				NT	A	56.0	57.6	None	✓	
48,0,1		✓		✓				✓				NT	A	28.8	28.8	None	✓	
43,0,1	✓	✓		✓				✓				NT	A	14.4	14.4	None	✓	
39,0,1	✓			✓				✓				NT	A	Undef	Undef	None	✓	
17,0,1		✓	✓								✓	NT	A	33.6	57.6	V.34	✓	
16,0,1		✓	✓								✓	NT	A	28.8	28.8	V.34	✓	
14,0,1	✓	✓	✓								✓	NT	A	14.4	14.4	V.34	✓	
12,0,1	✓		✓								✓	NT	A	9.6	9.6	V.34	✓	
7,0,1	✓		✓								✓	NT	A	Undef	Undef	V.32	✓	
0,0,1		✓	✓								✓	NT	A	NA	57.6	AutoBaud	✓	
0,0,1	✓		✓								✓	NT	A	9.6	Undef	AutoBaud	✓	

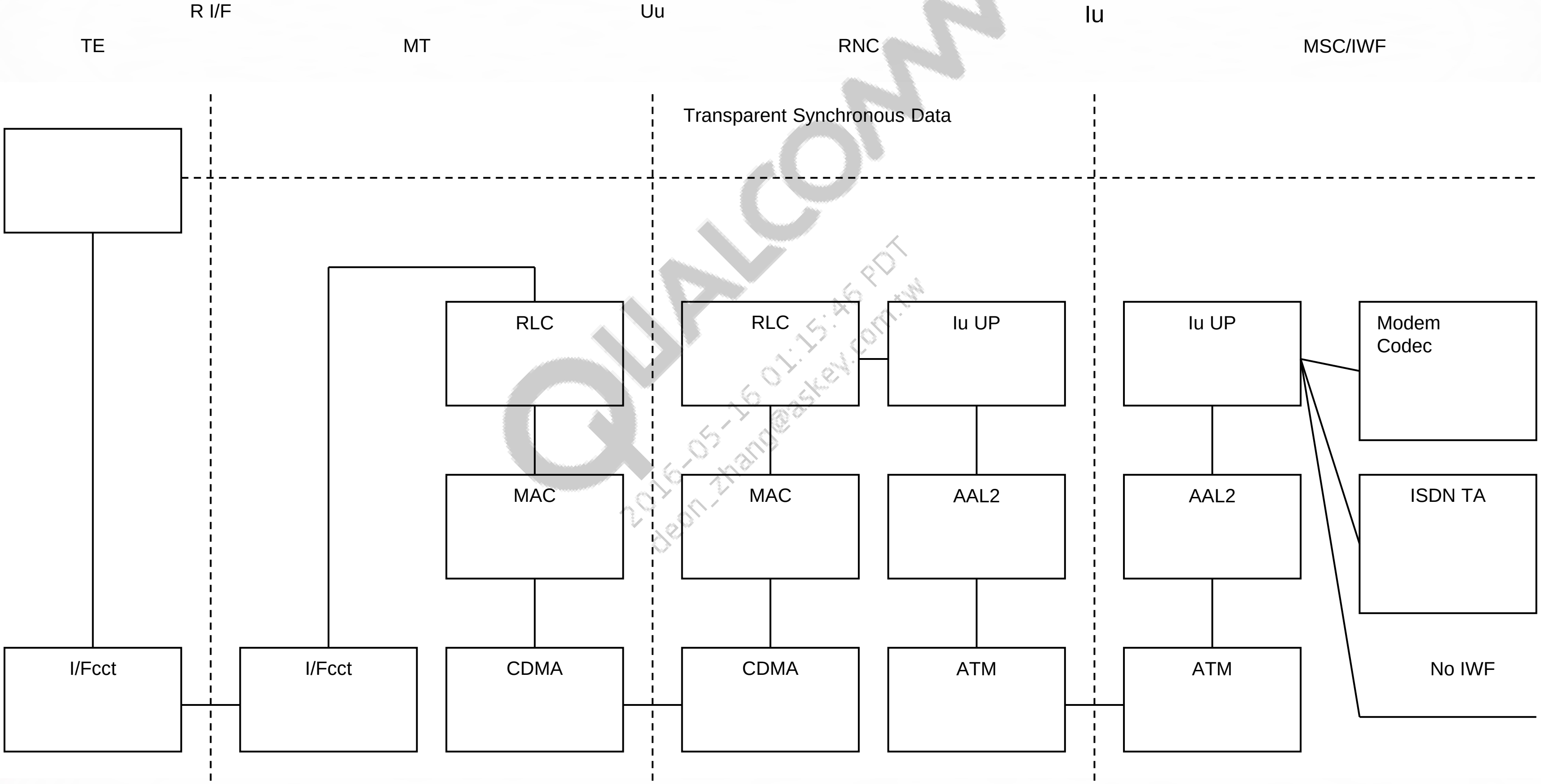
WCDMA CS Data Bearer Services

- UDI using the transparent/synchronous connection type is supported.
- UDI and RDI using the nontransparent connection type is supported for asynchronous CS data calls.
- Frame Tunneling mode FTM signaling supported for 56 kbps (UDI/RDI) and 64 kbps (UDI only).
- RLC is used in Transparent Mode (TM) for all CS Data Bearer services.
- Bearer capability settings for 64 kbps synchronous transparent CS data calls are per Appendix B.1.3.1.5 in [S1].
- Support for 14.4, 28.8, 56.0, and 64.0 kbps asynchronous nontransparent CS data calls are per Appendix B.1.2 in [S1].

WCDMA Synchronous Transparent CS Data

- Protocol configuration for synchronous transparent CS data is shown on the following slide.
- For synchronous CS data, the application must guarantee a constant data stream.
- Data transfer on the uplink is performed as follows:
  - The mobile buffers incoming data bytes from the TE.
  - For each TTI, if enough data is available, the data for the TTI is transmitted, otherwise, no data is transmitted.

WCDMA Synchronous Transparent CS Data (cont.)

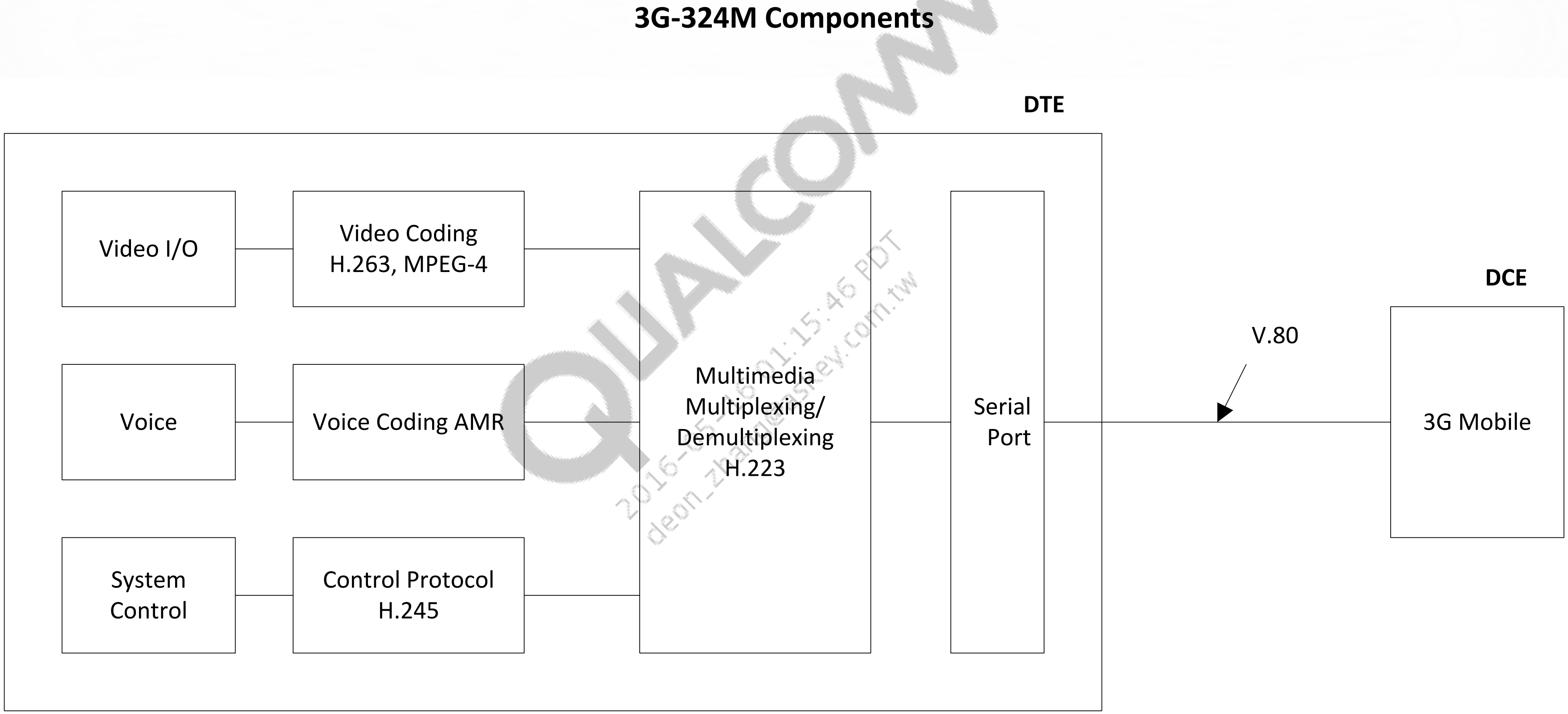


V.80 Background

- What is 3G-324M?
  - Defined by 3GPP to enable multimedia communication over CS networks
  - Based on the ITU-T H.324M standard, 3GPP added voice/video codecs and mandatory requirements for bit-error handling that was defined as optional by ITU-T
  - 3G-324M consists of video codec (H.263/MPEG-4), audio codec (AMR), control protocol (H.245), and multiplexing protocol (H.223)
    - H.223 multiplexing protocol generates synchronous HDLC frames
- Why do we need V.80?
  - It supports DTE-based 3G-324M videotelephony clients, e.g., laptop, iPAQ.
  - V.80 is the recommended method (see [S2]) for solving the rate adaptation problem, i.e., converting the asynchronous data stream from the DTE to a synchronous data stream over the air.

Note: V.80 is used only between the DTE and the phone, i.e., it does not affect interoperability.

V.80 Background (cont.)



V.80 Overview

- What is V.80?
  - ITU-T recommendation for In-band DCE Control and Synchronous Data modes for asynchronous DTE
  - Defines procedures for exchanging in-band commands and responses between the DTE and DCE while in Online Data mode
    - Escape character for in-band commands is EM or 19h/99h
    - In-band commands consist of <EM> followed by one or more command characters
    - For transparency, in-band commands are used to represent instances of user data with the same ordinal value as <EM>
    - The DCE must recognize in-band commands, remove them from the user data, execute them if possible, and generate in-band commands toward the DTE
  - Defines various synchronous operation modes; the only one relevant for 3G-324M is Synchronous Access mode, which consists of two submodes
    - Transparent submode – When a Tx underrun condition occurs, the DCE shall Tx one or more SYN sequences. The SYN bit pattern is configured by the DTE by means of the +ESA AT command.
    - Framed submode – This mode is not applicable to 3G-324M.

V.80 Functionality

- Support for V.80 Synchronous Access mode, Transparent submode
- All character transparency commands in Table 9/V.80 are supported
- Rate command 0xBE is returned after CONNECT
- Currently, all other in-band commands are not supported and are stripped from the bitstream
- AT commands for V.80 support
  - +ES – Synchronous modes enable
  - +ESA – Synchronous Access mode configuration
- +ES
  - Synchronous modes enabled
  - Table 7/V.80; Section 8.1/V.80
  - +ES = 6, ,8 supported only
  - +ES power-on default is undefined
  - +ES = 6, ,8 before ATDT will set up a V.80 call
  - +ES<CR> will set back to power-on default

V.80 Functionality (cont.)

- +ESA
  - Synchronous Access mode configuration
  - Table 8/V.80; Section 8.2/V.80
  - +ESA = 0, , , ,0,0,(0 to 255), supported only
  - +ESA power-on default is 0, , , ,0,0,255
  - +ESA is used for setting the SYN flag

WCDMA CS RAB Configurations

- The following RAB configurations are supported (see TS 34.108 v3.2.0)
  - Streaming/Unknown/UL:14.4/DL:14.4 Kbps/CS RAB + UL:3.4 DL 3.4 Kbps SRBs for DCCH
  - Conversational/Unknown/UL:28.8/DL:28.8 Kbps/CS RAB + UL:3.4 DL 3.4 Kbps SRBs for DCCH
  - Streaming/Unknown/UL:57.6/DL:57.6 Kbps/CS RAB + UL:3.4 DL 3.4 Kbps SRBs for DCCH
  - Conversational/Unknown/UL:64/DL:64 Kbps/CS RAB + UL:3.4 DL:3.4 Kbps SRBs for DCCH (TTI = 20 ms or 40 ms)

UMTS CS File Structure

- Common header file
  - dsucsd.h – Common data declarations
- Bearer capability
  - dsucsdbc.h, dsucsdbc.c – Bearer capability interface/implementation
- Mode-specific handlers
  - dsucsdhdlr.h, dsucsdhdlr.c – Mode-specific handler interface/implementation
- Application interface
  - dsucsdappif.h, dsucsdappif.c – Application interface interface/implementation
- Nontransparent data
  - dsucsdnt.h, dsucsdnt.c – Top-level control and state machines
  - dsucsddlpdu.c – Downlink RLP/L2RCOP queue management
  - dsucsdulpdu.c – Uplink RLP/L2RCOP queue management
  - dsucsdrlp.h, dsucsdrlp.c – RLP/L2RCOP protocol functions and state machine
  - dsucsdrlphw.c – Low-level PDU processing
  - dsucsdremap.c – RLP handover processing
  - dsucsdtimer.h, dsucsdtimer.c – Timer management
  - dsucsdlog.h, dsucsdlog.c – PDU logging

UMTS CS File Structure (cont.)

- V.42bis compression
  - dsucsdv42.h,dsucsdv42.c – Top-level control interface/implementation
  - dsucsdv42c.c – Compression routines
  - dsucsdv42d.c – Decompression routines
  - dsucsdv42buf.c – V42bis buffer routines

WCDMA CS File Structure

- Common header file
  - dswcsd.h – Common data declarations
- Uplink task – Synchronous/transparent UDI
  - dswcsdultask.c, dswcsdultask.h – Functions for uplink task
- Uplink processing – Synchronous/transparent UDI
  - dswcsdul.c, dswcsdul.h – Functions for uplink processing
- Downlink task – Synchronous/transparent UDI
  - dswcsddltask.c, dswcsddltask.h – Functions for downlink task
- Downlink processing – Synchronous/transparent UDI
  - dswcsddl.c, dswcsddl.h – Functions for downlink processing
- Mode-specific handler interface
  - dsucsdmshif.c, dsucsdmshif.v – Functions for WCDMA mode-specific handlers
- RLP interface
  - dswcsdntrlpif.c, dswcsdntrlpif.h – Nontransparent call processing
- Videotelephony support
  - dswcsdv80.c, dswcsdv80.h – V.80 implementation

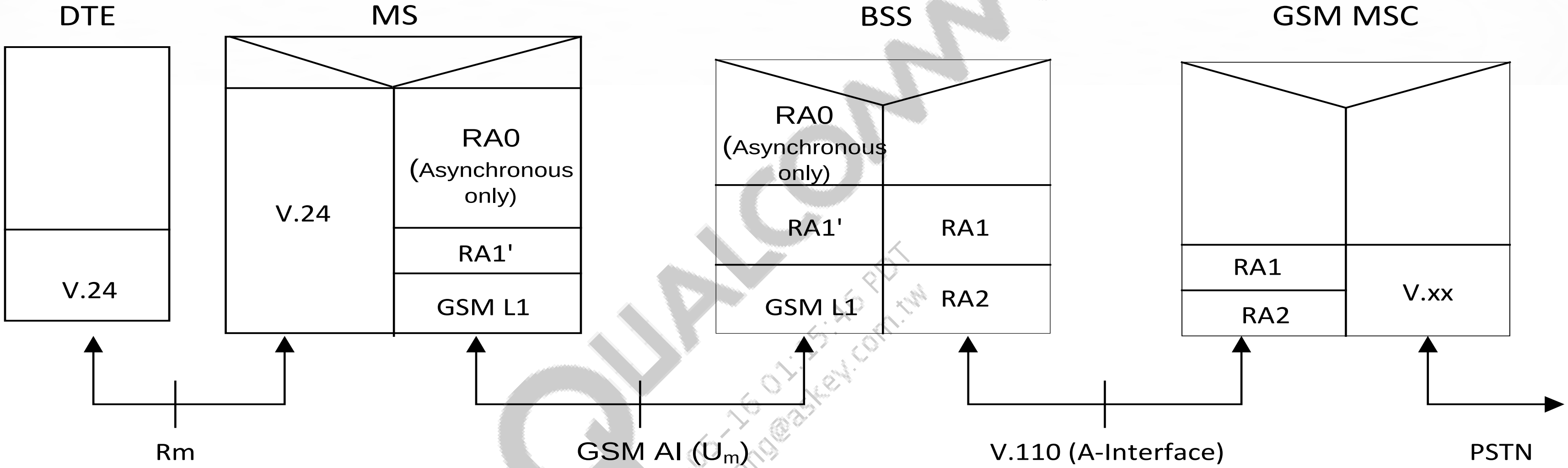
GSM CS Data Bearer Services

- Implements origination, verification, and negotiation of bearer-capabilities parameters
- Supports UDI, 3.1 kHz audio and fax connection types
- Supports both transparent and nontransparent transport protocols
- Supports asynchronous data
- Supports 9.6 Kbps and 14.4 Kbps connection speeds

GSM Circuit-Switched Transparent Data Protocol

- Protocol configuration for transparent calls is depicted on the following slide
- RA0 protocol is used to convert asynchronous data to/from synchronous data
  - Adds/removes start/stop/parity bits
  - Used only for asynchronous data
  - Not used for fax
- RA1’ protocol multiplexes status data (including modem status information) with the user data into a modified V.110 frame
- For additional information, see [S4]

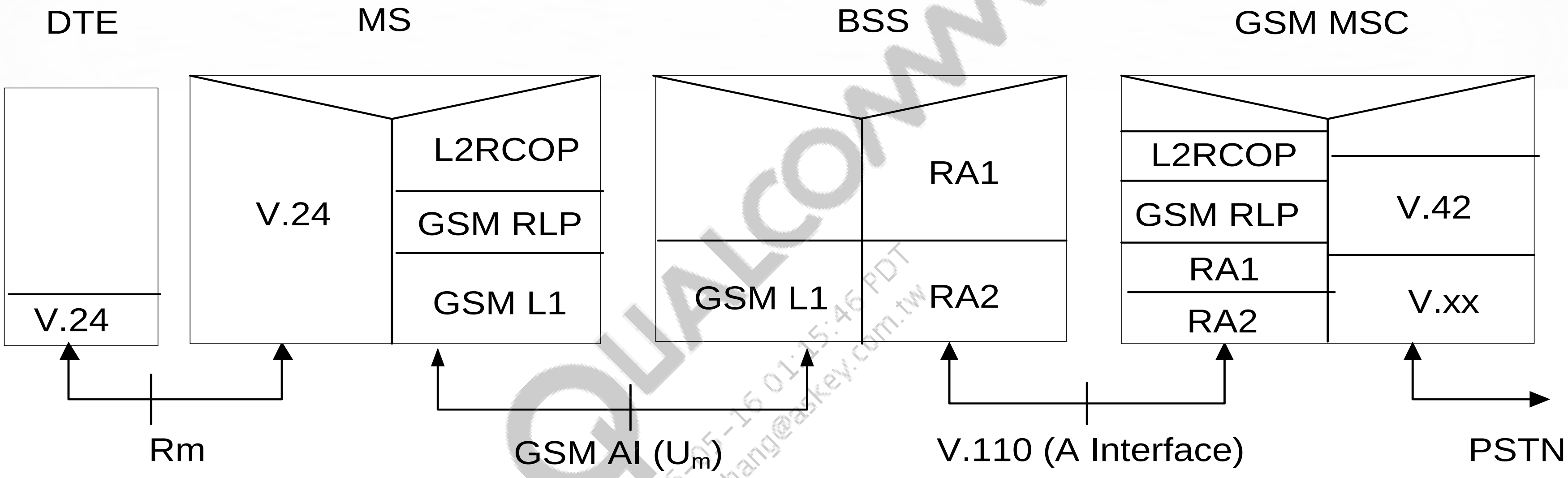
GSM CS Transparent Data Call



UMTS CS Nontransparent Data Protocol

- The protocol configuration for nontransparent data calls is depicted on the following slide.
- RLP is a sliding windows protocol used to ensure that data is delivered reliably to/from the mobile and the network.
- RLP negotiation, which occurs at the start of the call, can negotiate that user octets be coded using V.42bis compression.
- Layer 2 Character-Oriented Protocol (L2RCOP) multiplexes status data (including modem status indications) and user data into the L2RCOP frame.
- Common L2RCOP/RLP implementation is shared by both WCDMA and GSM protocols.
- RLP is described in [S5] and L2RCOP is described in Annex A of [S6].

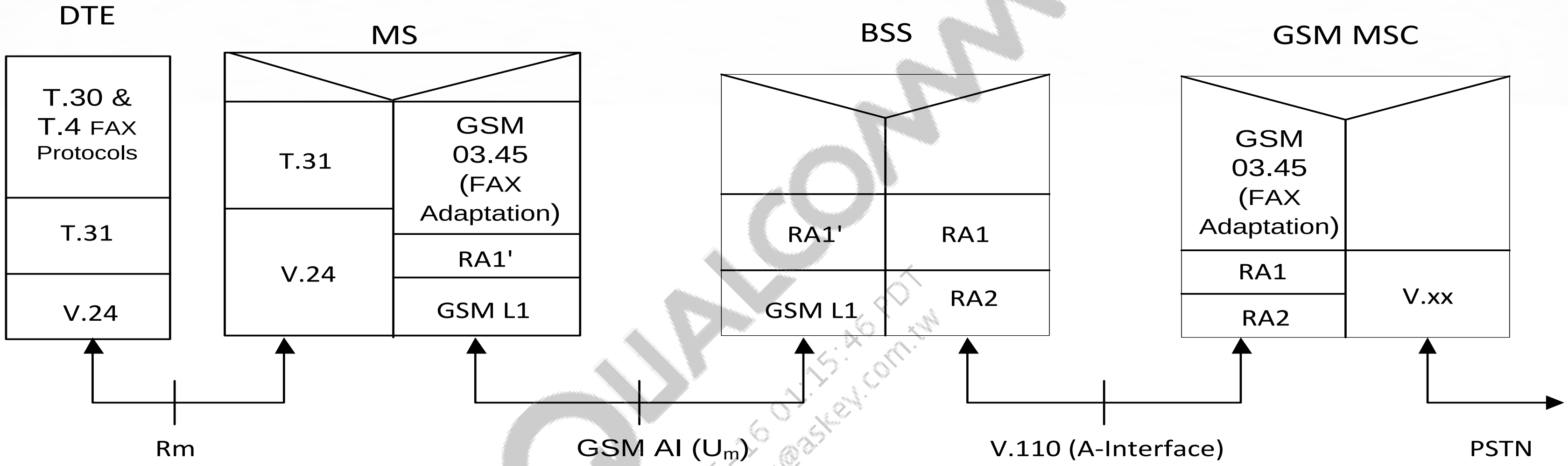
GSM CS Nontransparent Data Call



GSM Class 1 Fax Protocol

- Protocol configuration for a GSM Class 1 fax call is depicted on the following slide.
- Supports 7200 bps and 9600 bps transmission speeds.
- Class 1 fax calls utilize transparent, synchronous CS data as the transport mechanism.
- T.30 (fax session) and T.4 (page data encoding) fax protocols are originated/terminated in DTE and the far-end fax machine connected to the network.
- The T.31 protocol is used to transfer data to/from DTE and the mobile, i.e., the modem.
- The GSM 03.45 protocol is used to transfer data to/from the mobile and the GSM MSC (IWF).
- For more information, see [S11] and [S12].

GSM Class 1 Fax Call



GSM CS Data and Fax File Structure

- GCSD task
  - dsgcsd.h/dsgcsd.c – GCSD task interface/implementation
  - dsgcsdi.h – Function prototypes and data types that are used internally for the GCSD subsystem
- Transparent data (including RA0 and RA1')
  - dsgcsdtdata.c – Top-level control
  - dsgcsdra0.c – RA0 protocol
  - dsgcsdra1.c – RA1' protocol
- Utility files
  - dsgcsdsio.c – GCSD CS data interfaces to SIO
  - dsgcsdl1if.c – GCSD task interface to Layer 1
  - dsgcsdbuf.c – Buffer manipulation routines
- Class 1 fax
  - dsfps.h/dsfps.c – Top-level interface to the fax layer
  - dsgcsdfa.h/dsgcsdfa.c, dsgcsdfatrans.h/dsgcsdfatrans.c – GSM 03.45 Fax adaptation interface/implementation
  - dst31.h/dst31.c, dst31hdlc.h/dst31hdlc.c, dst31i.h, dst31sop.c, dst31states.c – T.31 implementation

Packet Data Bearer Services

- PDP-type PPP using transparent negotiation is supported (WCDMA only).
- PDP-type IP is supported.
- TE+MT mode of operation is supported over both PDP-type IP and PDP-type PPP.
- MT-only mode of operation, i.e., sockets, is supported over PDP-type IP only.
- Multiple PDP contexts are supported.
  - Up to three primary PDP contexts can be set up.
  - PDP contexts can be used for TE+MT or MT-only mode of operation.

Packet Data Bearer Services (cont.)

- PDP profile definitions are supported.
  - A PDP profile is a collection of parameters that define the parameters for a PDP context.
  - Up to 16 ATCoP profiles for TE+MT calls can be defined.
  - Up to 16 sockets profiles for embedded calls can be defined.
  - Profile definitions are retained across power-cycles.
  - Parameters of a PDP profile include:
    - Context definition
    - Authentication information
    - UMTS QoS parameters
    - GPRS QoS parameters
    - DNS parameters
  - The PDP type (PDP-IP or PDP-PPP) is specified as the context definitions for the profile.
  - Parameters for ATCoP profiles are specified using AT commands +CGDCONT, $QCPDPP, +CGEQREQ, +CGEQMIN, +CGQREQ, and +CDQMIN.
  - Parameters (subset) for sockets profiles are specified using UI menus.
  - The DNS address can only be specified using the UI, as it applies only to a sockets call.

Packet Data Bearer Services (cont.)

- PDP profile definitions are supported (cont.)
  - During call origination (TE+MT), the context and associated parameters to be used are specified in the dial string, e.g., ATD∗98<cid># (UMTS) or ATD∗99***<cid>#.
  - For sockets calls, the profile number to be used can be specified explicitly by the application. If the application does not specify the profile number, the default profile specified in EFS is activated. The default PDP profile number can be changed using the UI menu.
- PDP context procedures
  - PDP context activation (by the mobile or the network) and PDP context deactivation (by the mobile or network) are supported.
  - PDP context preservation procedures are supported.
  - Network-requested PDP context activation is not supported.
  - PDP context modification is not supported.
- Simultaneous PDP contexts
  - Secondary PDP contexts are supported.

Packet Data Bearer Services (cont.)

- PS data continuity
  - Preservation of existing PDP-type IP contexts during the course of intersystem changes between GSM and UMTS Radio Access Technologies is supported.
- Quality of Service (QoS)
  - The mobile will request the QoS parameters specified by the user by means of +CGEQREQ or +CGQREQ AT commands.
  - The mobile will request the subscribed QoS parameters in the Activate PDP Context Request message if the user has not specified the QoS parameters.
  - The user can specify the minimum QoS required for the PDP context using AT+CGQMIN or AT+CGEQMIN AT commands.
  - If the QoS assigned by the network is less than the minimum QoS requested by the user, the PDP context is released.
- Multiple PDP contexts
  - Up to three primary PDP contexts are supported.
  - Context can be set up for TE+MT or MT-only mode of operation.
  - The PDP profile parameters to be used for context setup is passed by the TE or by the MT (sockets stack).

Note: This feature is currently not supported for PDP-type PPP calls. All existing PDP-type PPP contexts are terminated when an intersystem change from UMTS to GSM occurs.

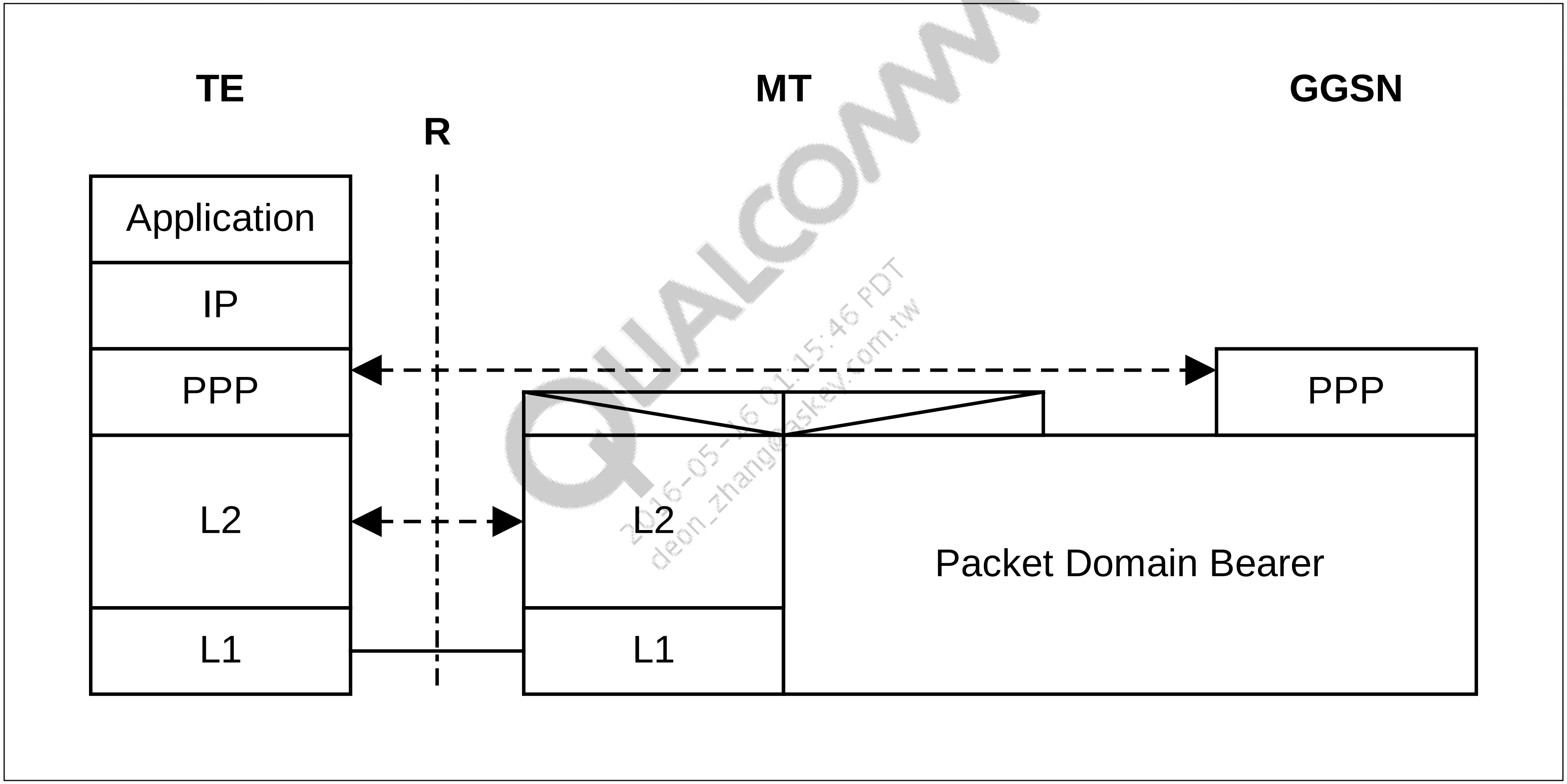
Packet Data Bearer Services (cont.)

- Secondary PDP contexts
  - Support for secondary PDP context activation for data applications using AMSS Sockets API is provided.
  - This allows the UE to request bandwidth resources for different IP flows with different QoS requirements while using the same PDP-type, IP address, and APN associated with existing primary PDP context.
  - Context can be set up for MT-only mode of operation.
  - One secondary context per primary context is supported.
- Mobile-Terminated PDP
  - MT PDP is only supported for embedded applications.
  - An application that is interested in MT context registers with the mode handler using dss_iface_ioctl on a specific APN.
  - When a page is received requesting a PDP Context Activation, the application that has registered for the request originates a call.
  - After the call origination, the call is treated like any MO call.

PDP-Type PPP

- For PDP-type PPP:
  - HDLC framing is added to PPP packets on the downlink.
  - HDLC framing is removed from PPP packets on the uplink.
  - All PPP negotiations are end-to-end (between the TE and GGSN).
- Protocol configuration for PDP-type PPP using transparent negotiation is shown on the following slide.

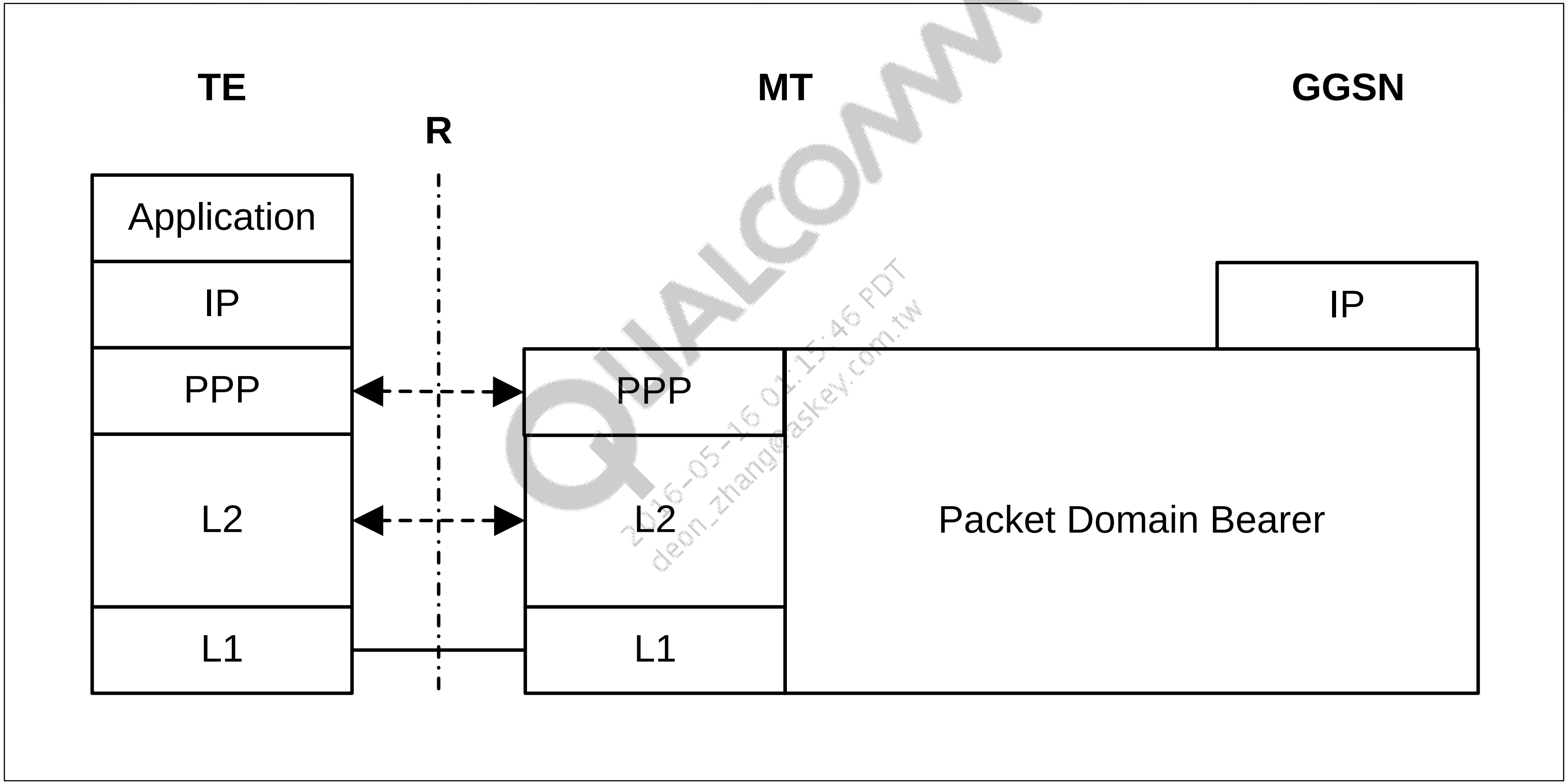
PDP-Type PPP (Transparent Negotiation)



PDP-Type IP

- For PDP-type IP TE+MT calls:
  - PPP negotiations are between the TE and the UE.
  - Authentication parameters are specified in the TE, e.g., Windows® dial-up networking configuration.
  - The primary/secondary DNS server address and the gateway address are provided to the TE, upon request, through the IPCP options.
- Protocol configuration for PDP-type IP is shown on the following slide.

PDP-Type IP (cont.)



Authentication Support for PDP-IP

- PAP and CHAP authentication are supported for PDP-IP calls.
- PAP/CHAP authentication packets are included in the Activate PDP Context request.
- For TE+MT calls, PAP/CHAP authentication packets that are received from the TE are included in the PDP Context Request message.
- The AT$QCPDPP command allows PAP/CHAP parameters to be specified on the UE for MT-only calls.
  - The UE generates the PAP/CHAP packets on the TE’s behalf and includes them in the Activate PDP Context request.
- $QCPDPP command format
  - AT$QCPDPP=[<cid>,<auth_type>,[<passwd>,[<user_name>]]]
    - cid – PDP context identifier to which the PPP parameters are applicable
    - auth_type – Authentication type; values are 0-none, 1-pap, 2-chap
    - passwd – ASCII string representing the following:
      - If auth_type is PAP(1) – Password
      - If auth_type is CHAP(2) – Shared secret
    - user_name – Represents the user name and is applicable only if auth_type is PAP(1)

Authentication Support for PDP-IP (cont.)

- Examples

Command	Response
AT$QCPDPP=1,0	OK
AT$QCPDPP=1,1,”pap_secret”,”pap_user “	OK
AT$QCPDPP=1,2,”chap_secret“	OK
AT$QCPDPP?	$QCPDPP : 1,2,””
AT$QCPDPP=?	$QCPDPP : 1,0-2,,

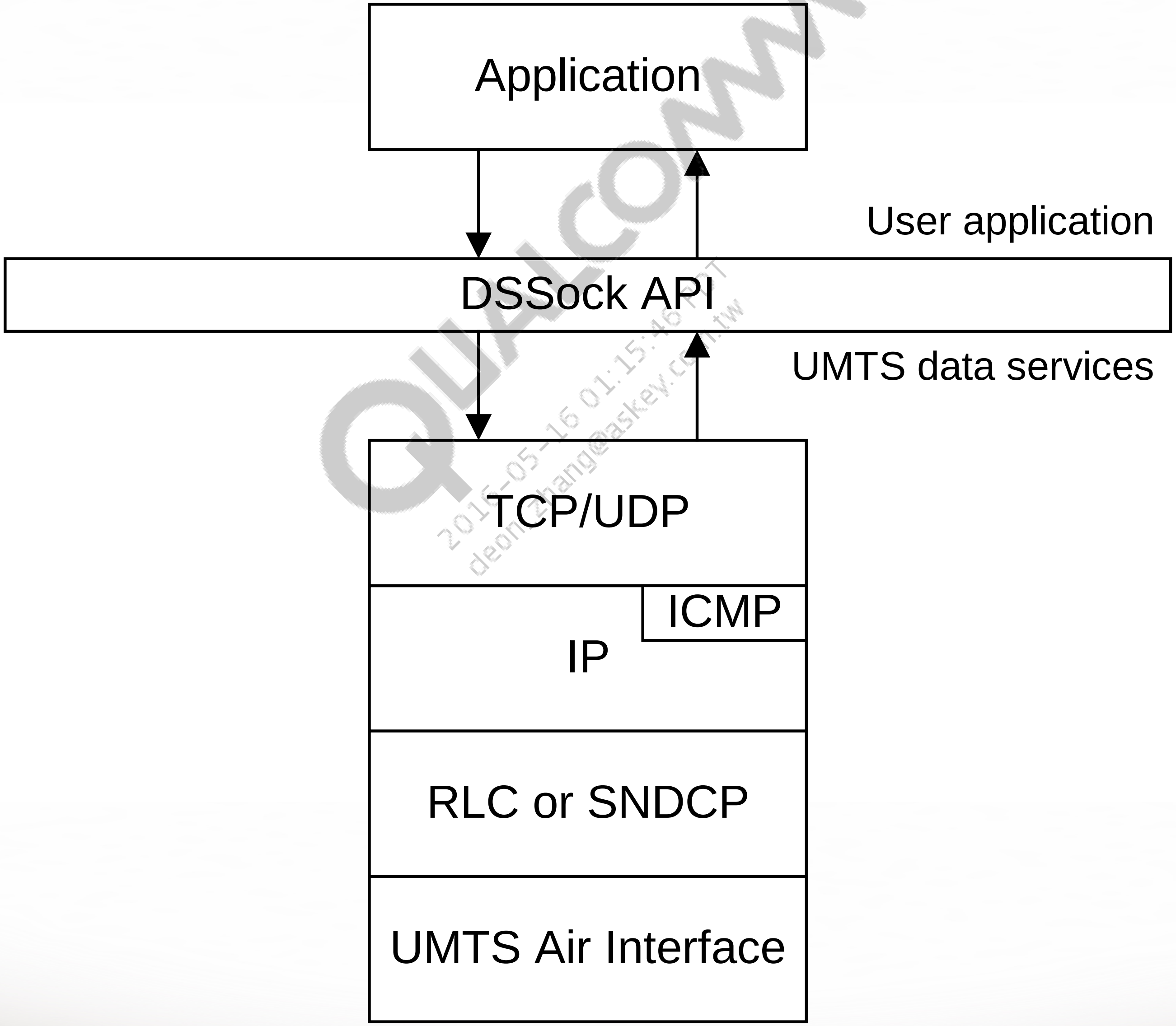
DNS Support for PDP-IP

- The UE includes IPCP packets (DNS Address request) in the PCO field of the Activate PDP Context request.
  - Both primary and secondary DNS addresses are requested.
- The network may respond with primary and/or secondary DNS addresses in the Activate PDP Context Accept.
- If no DNS addresses are specified by the network, preprovisioned DNS addresses are used.
  - For TE+MT calls, the TE does not advertise any DNS addresses if the network does not provide them in the PDP Context Accept message.
  - For MT calls, the TE uses the DNS addresses preprovisioned on the phone if the network does not provide them in the PDP Context Accept message.
    - DNS addresses may be preprovisioned for each PDP profile using UI menu items.
- IPCP options received on the UE, other than the DNS options, are discarded.
- For PDP-type IP, there is no IPCP negotiation on the Uu interface.

Sockets

- Only Sockets Over PDP-type IP is supported; Sockets Over PDP-type PPP is not supported.
- The protocol configuration for Sockets Over PDP-type IP is shown on the following slide.

Sockets Over PDP-Type IP

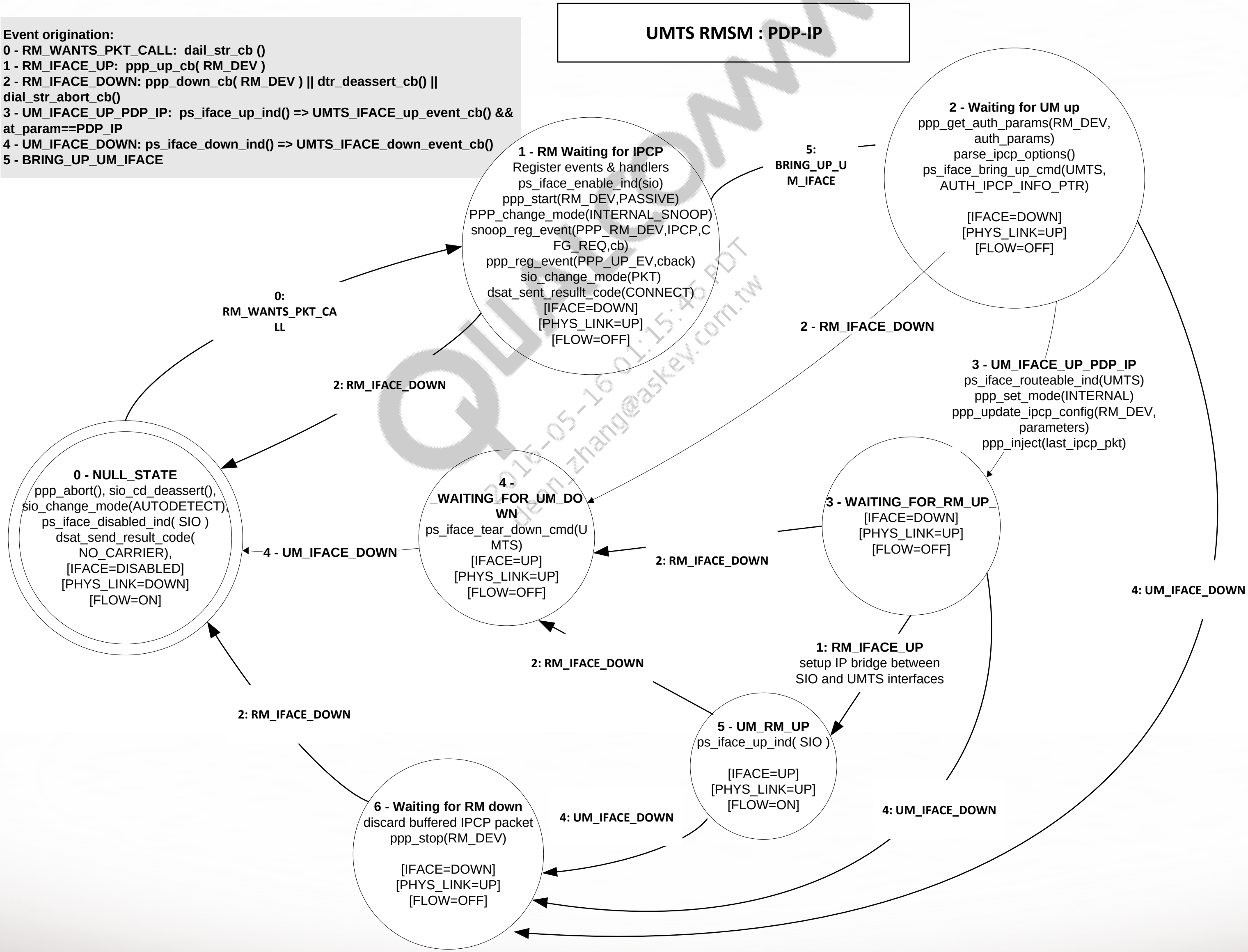


Mode-Specific Handler Interfaces

- The mode-specific handler interfaces to:
  - PS_IFACE
    - Uses PS_IFACE function ps_iface_create() to create an interface for each PDP context at startup
    - Registers callbacks with PS_IFACE to bring up and tear down interfaces and the traffic channel
    - Uses PS_IFACE functions ps_iface_enable_flow() and ps_iface_disable_flow() to enable/disable flow on itself
    - Sets all the configuration parameters in the interface using PS_IFACE
  - Rm State Machine (RMSM)
    - Picks up the correct Um interface to bring up a call and calls ps_iface_bring_up_cmd() to bring up the interface
    - Gets the PDP profile number from the dial string and passes it to the Um interface
    - Handles flow control events from the Um interface to flow control the Rm interface
    - For PDP-PPP calls, it bridges Um and Rm PPP devices and configures PPP on the Um interface
    - Brings down the Um interface when the user ends the call by calling ps_iface_tear_down_cmd()

RMSM State Machine

Event origination:
0 - RM_WANTS_PKT_CALL: dail_str_cb ()
1 - RM_IFACE_UP: ppp_up_cb(RM_DEV)
2 - RM_IFACE_DOWN: ppp_down_cb(RM_DEV) || dtr_deassert_cb() || dial_str_abort_cb()
3 - UM_IFACE_UP_PDP_IP: ps_iface_up_ind() => UMTS_IFACE_up_event_cb() && at_param==PDP_IP
4 - UM_IFACE_DOWN: ps_iface_down_ind() => UMTS_IFACE_down_event_cb()
5 - BRING_UP_UM_IFACE



UMTS RMSM : PDP-IP

0 - NULL_STATE
ppp_abort(), sio_cd_deassert(),
sio_change_mode(AUTODETECT),
ps_iface_disabled_ind(SIO)
dsat_send_result_code(
NO_CARRIER),
[IFACE=DISABLED]
[PHYS_LINK=DOWN]
[FLOW=ON]

0: RM_WANTS_PKT_CALL

1 - RM Waiting for IPCP
Register events & handlers
ps_iface_enable_ind(sio)
ppp_start(RM_DEV,PASSIVE)
PPP_change_mode(INTERNAL_SNOOP)
snoop_reg_event(PPP_RM_DEV,IPCP,CFG_REQ,cb)
ppp_reg_event(PPP_UP_EV,cback)
sio_change_mode(PKT)
dsat_sent_resullt_code(CONNECT)
[IFACE=DOWN]
[PHYS_LINK=UP]
[FLOW=OFF]

5: BRING_UP_UM_IFACE

2 - Waiting for UM up
ppp_get_auth_params(RM_DEV,
auth_params)
parse_ipcp_options()
ps_iface_bring_up_cmd(UMTS,
AUTH_IPCP_INFO_PTR)

[IFACE=DOWN]
[PHYS_LINK=UP]
[FLOW=OFF]

2: RM_IFACE_DOWN

2 - RM_IFACE_DOWN

3 - UM_IFACE_UP_PDP_IP
ps_iface_routeable_ind(UMTS)
ppp_set_mode(INTERNAL)
ppp_update_ipcp_config(RM_DEV,
parameters)
ppp_inject(last_ipcp_pkt)

4 - _WAITING_FOR_UM_DOWN
ps_iface_tear_down_cmd(UMTS)
[IFACE=UP]
[PHYS_LINK=UP]
[FLOW=OFF]

4 - UM_IFACE_DOWN

3 - WAITING_FOR_RM_UP_
[IFACE=DOWN]
[PHYS_LINK=UP]
[FLOW=OFF]

2: RM_IFACE_DOWN

4: UM_IFACE_DOWN

2: RM_IFACE_DOWN

1: RM_IFACE_UP
setup IP bridge between
SIO and UMTS interfaces

2: RM_IFACE_DOWN

5 - UM_RM_UP
ps_iface_up_ind(SIO)

[IFACE=UP]
[PHYS_LINK=UP]
[FLOW=ON]

6 - Waiting for RM down
discard buffered IPCP packet
ppp_stop(RM_DEV)

[IFACE=DOWN]
[PHYS_LINK=UP]
[FLOW=OFF]

4: UM_IFACE_DOWN

4: UM_IFACE_DOWN

Mode-Specific Handler Interfaces

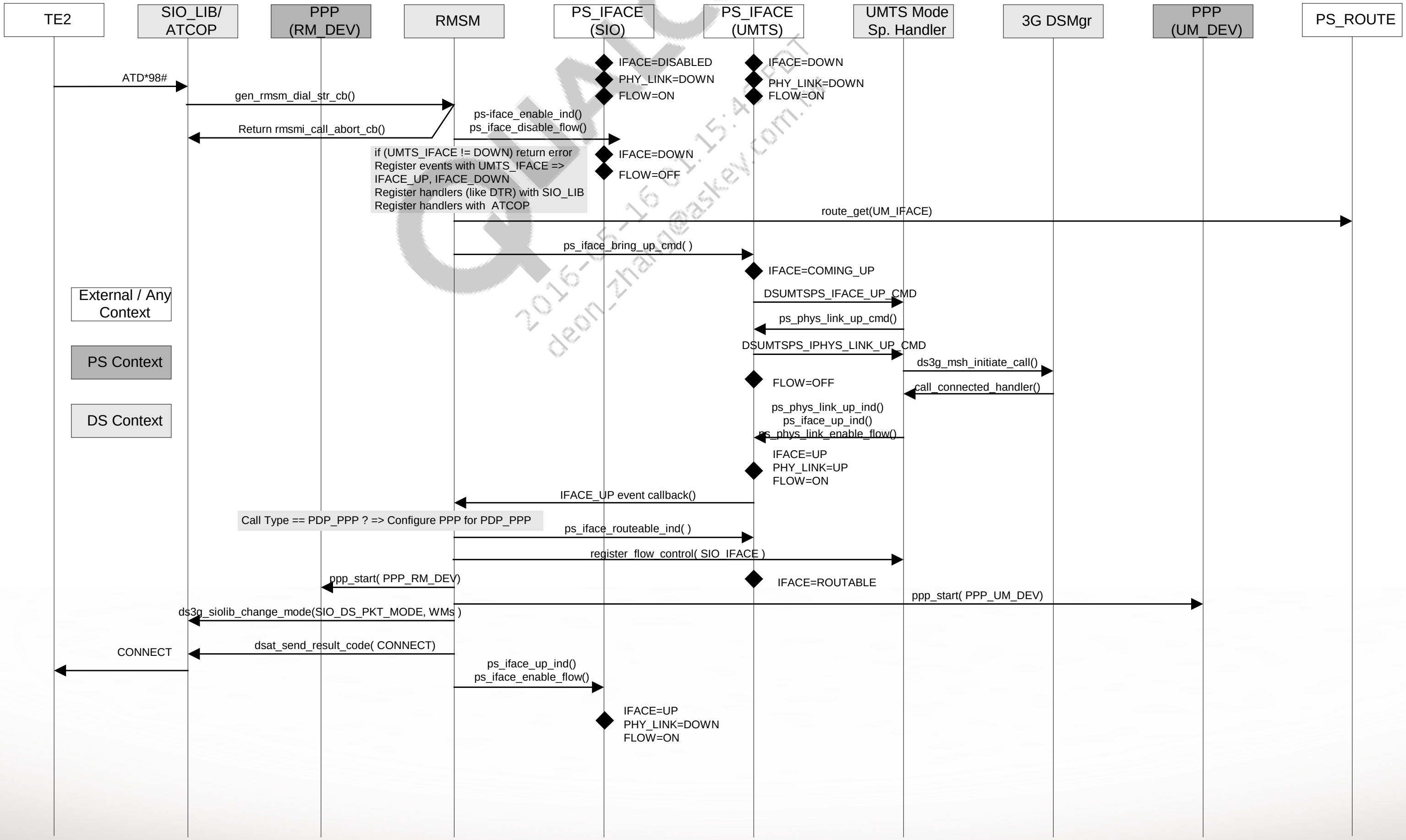
- The mode-specific handler interfaces to (cont.):
  - 3G dsmgr
    - Registers mode-specific handler at powerup
    - As part of bringing up the UMTS interface, calls ds3g_initiate_call() to set up the traffic channel, which leads to the origination-related handlers, i.e., orig_handler(), call_id_handler(), and call_connected_handler(), registered at startup being invoked
    - As part of tearing up the UMTS interface, calls ds3g_hangup_call() to tear down the traffic channel, which leads to the call-end-related handlers, i.e., user_end_handler(), and call_ended_handler(), registered at startup being invoked
    - Calls ds3g_msh_call_rab_reestab_req() to bring the interface out of dormancy

MO PDP-PPP Call Setup

PDP-PPP TE2 Call Setup

NOTES

- UMTS IFACE created and initialized at powerup
- UMTS Interface is enabled when service becomes available, default route is added
- ROUTABLE state denies service to application sockets
- Default operational mode configured for Sockets
- Data from and to SIO_IFACE is denied by default
- Each RMSM instance is initialized with a UMTS iface pointer



TE2 | SIO_LIB/ ATCOP | PPP (RM_DEV) | RMSM | PS_IFACE (SIO) | PS_IFACE (UMTS) | UMTS Mode Sp. Handler | 3G DSMgr | PPP (UM_DEV) | PS_ROUTE

IFACE=DISABLED
PHY_LINK=DOWN
FLOW=ON

IFACE=DOWN
PHY_LINK=DOWN
FLOW=ON

ATD*98#

gen_rmsm_dial_str_cb()

Return rmsmi_call_abort_cb()

ps-iface_enable_ind()
ps_iface_disable_flow()

if (UMTS_IFACE != DOWN) return error
Register events with UMTS_IFACE =>
IFACE_UP, IFACE_DOWN
Register handlers (like DTR) with SIO_LIB
Register handlers with ATCOP

IFACE=DOWN

FLOW=OFF

route_get(UM_IFACE)

ps_iface_bring_up_cmd( )

IFACE=COMING_UP

DSUMTSPS_IFACE_UP_CMD

ps_phys_link_up_cmd()

DSUMTSPS_IPHYS_LINK_UP_CMD

ds3g_msh_initiate_call()

FLOW=OFF

call_connected_handler()

ps_phys_link_up_ind()
ps_iface_up_ind()
ps_phys_link_enable_flow()

IFACE=UP
PHY_LINK=UP
FLOW=ON

IFACE_UP event callback()

Call Type == PDP_PPP ? => Configure PPP for PDP_PPP

ps_iface_routeable_ind( )

register_flow_control(SIO_IFACE)

ppp_start(PPP_RM_DEV)

IFACE=ROUTABLE

ppp_start(PPP_UM_DEV)

ds3g_siolib_change_mode(SIO_DS_PKT_MODE, WMs)

CONNECT

dsat_send_result_code(CONNECT)

ps_iface_up_ind()
ps_iface_enable_flow()

IFACE=UP
PHY_LINK=DOWN
FLOW=ON

External / Any Context

PS Context

DS Context

MO PDP-IP TE+MT Call Setup

PDP-IP TE2 Call Setup

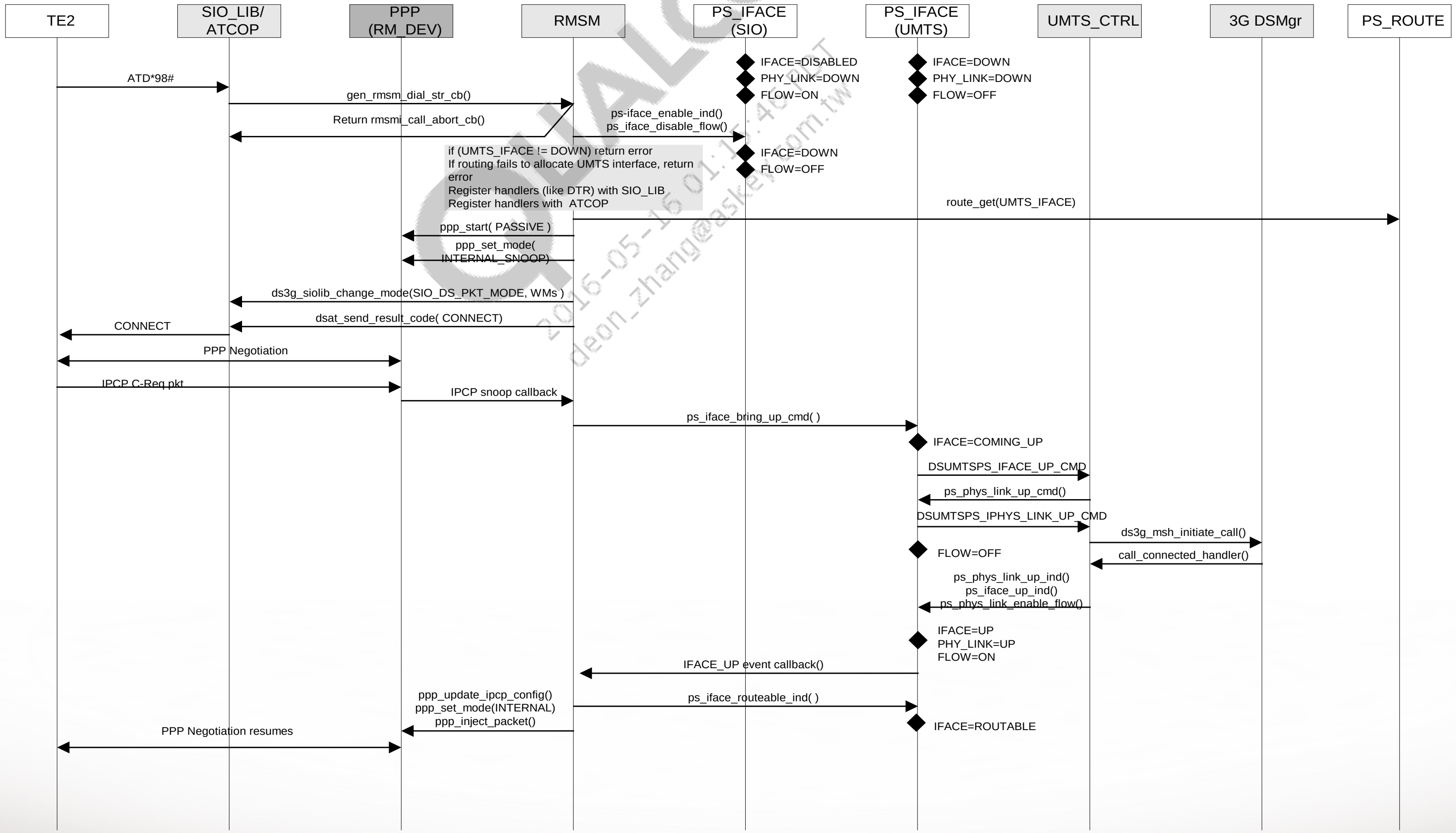
PS Context

DS Context

External / Any Context

NOTES

- UMTS IFACE created and initialized at powerup
- UMTS Interface is enabled when service becomes available, default route is added
- ROUTABLE state denies service to application sockets
- Default operational mode configured for Sockets
- Data from and to SIO_IFACE is denied by default
- Each RMSM instance is initialized with a UMTS iface pointer



TE2 | SIO_LIB/ ATCOP | PPP (RM_DEV) | RMSM | PS_IFACE (SIO) | PS_IFACE (UMTS) | UMTS_CTRL | 3G DSMgr | PS_ROUTE

IFACE=DISABLED
PHY_LINK=DOWN
FLOW=ON

IFACE=DOWN
PHY_LINK=DOWN
FLOW=OFF

ATD*98#

gen_rmsm_dial_str_cb()

Return rmsmi_call_abort_cb()

ps-iface_enable_ind()
ps_iface_disable_flow()

IFACE=DOWN
FLOW=OFF

if (UMTS_IFACE != DOWN) return error
If routing fails to allocate UMTS interface, return error
Register handlers (like DTR) with SIO_LIB
Register handlers with ATCOP

route_get(UMTS_IFACE)

ppp_start(PASSIVE)

ppp_set_mode(INTERNAL_SNOOP)

ds3g_siolib_change_mode(SIO_DS_PKT_MODE, WMs)

dsat_send_result_code(CONNECT)

CONNECT

PPP Negotiation

IPCP C-Req pkt

IPCP snoop callback

ps_iface_bring_up_cmd( )

IFACE=COMING_UP

DSUMTSPS_IFACE_UP_CMD

ps_phys_link_up_cmd()

DSUMTSPS_IPHYS_LINK_UP_CMD

ds3g_msh_initiate_call()

FLOW=OFF

call_connected_handler()

ps_phys_link_up_ind()
ps_iface_up_ind()
ps_phys_link_enable_flow()

IFACE=UP
PHY_LINK=UP
FLOW=ON

IFACE_UP event callback()

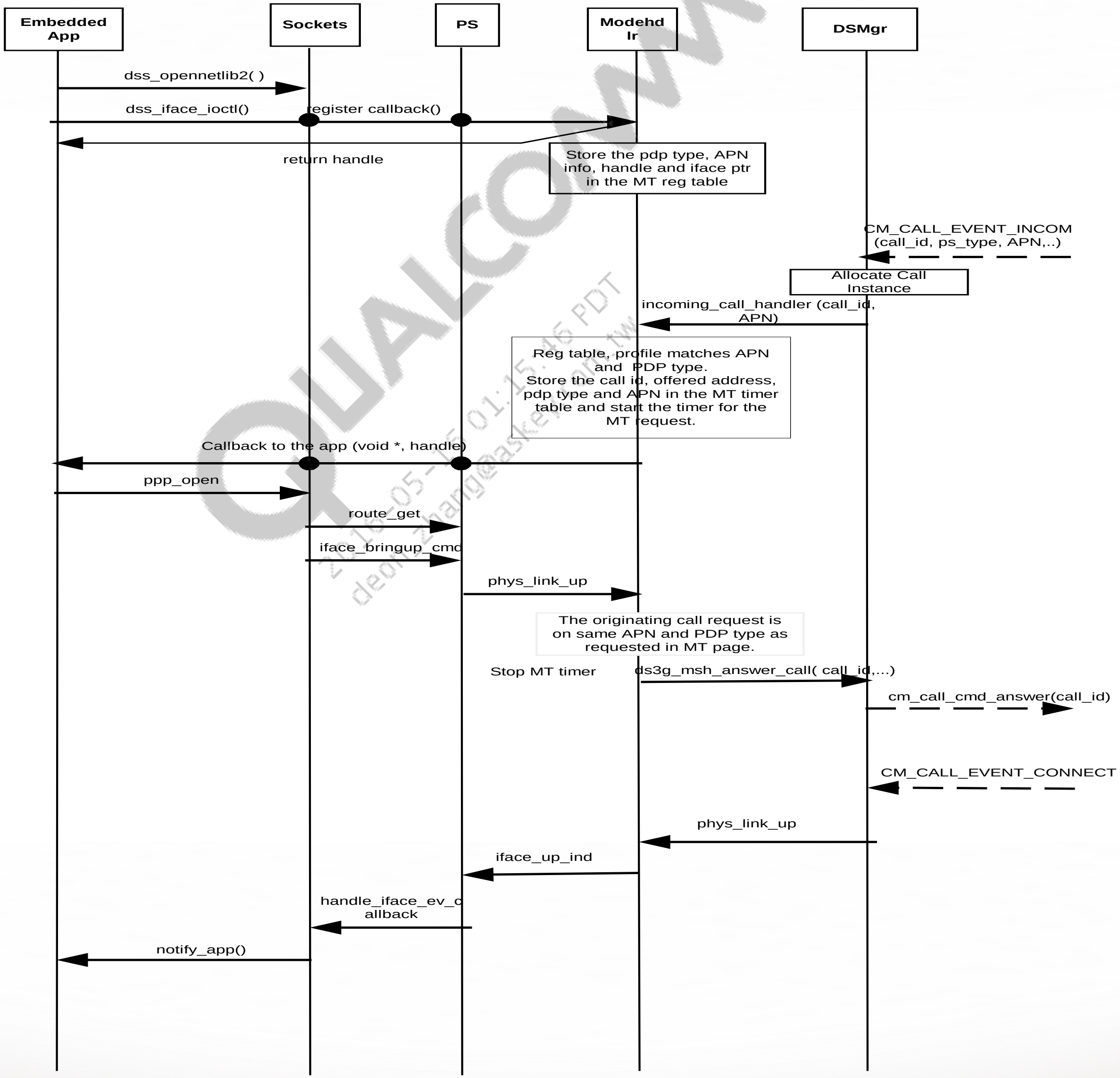
ppp_update_ipcp_config()
ppp_set_mode(INTERNAL)
ppp_inject_packet()

ps_iface_routeable_ind( )

IFACE=ROUTABLE

PPP Negotiation resumes

MT PDP-IP Call Setup



Embedded App

Sockets

PS

Modehd lr

DSMgr

dss_opennetlib2( )

dss_iface_ioctl()

register callback()

return handle

Store the pdp type, APN info, handle and iface ptr in the MT reg table

CM_CALL_EVENT_INCOM (call_id, ps_type, APN,..)

Allocate Call Instance

incoming_call_handler (call_id, APN)

Reg table, profile matches APN and PDP type. Store the call id, offered address, pdp type and APN in the MT timer table and start the timer for the MT request.

Callback to the app (void *, handle)

ppp_open

route_get

iface_bringup_cmd

phys_link_up

The originating call request is on same APN and PDP type as requested in MT page.

Stop MT timer

ds3g_msh_answer_call(call_id,...)

cm_call_cmd_answer(call_id)

CM_CALL_EVENT_CONNECT

phys_link_up

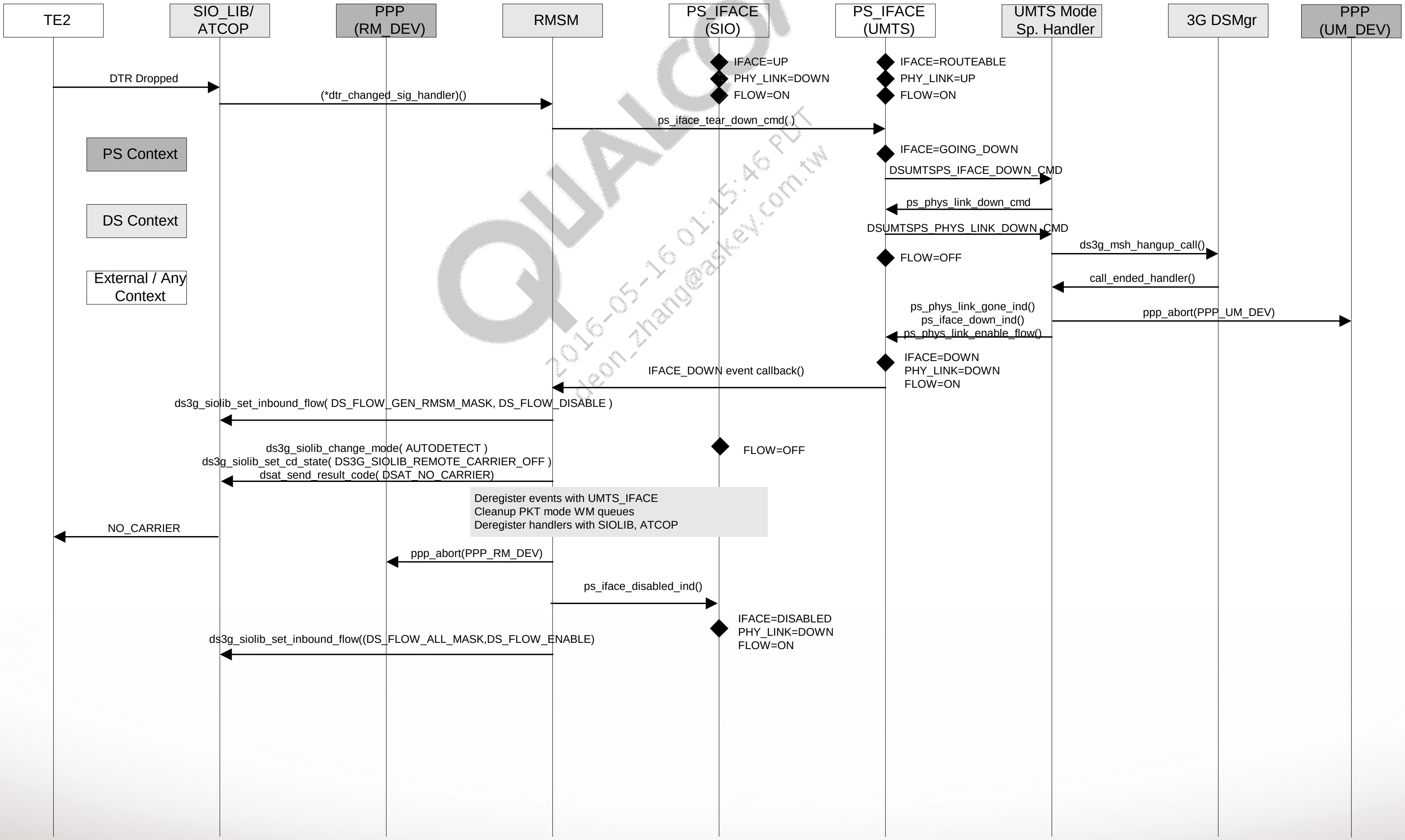
iface_up_ind

handle_iface_ev_callback

notify_app()

PDP-PPP UE Call Release

PDP-PPP TE2 Call Release



TE2

SIO_LIB/ ATCOP

PPP (RM_DEV)

RMSM

PS_IFACE (SIO)

PS_IFACE (UMTS)

UMTS Mode Sp. Handler

3G DSMgr

PPP (UM_DEV)

PS Context

DS Context

External / Any Context

IFACE=UP
PHY_LINK=DOWN
FLOW=ON

IFACE=ROUTEABLE
PHY_LINK=UP
FLOW=ON

DTR Dropped

(*dtr_changed_sig_handler)()

ps_iface_tear_down_cmd( )

IFACE=GOING_DOWN

DSUMTSPS_IFACE_DOWN_CMD

ps_phys_link_down_cmd

DSUMTSPS_PHYS_LINK_DOWN_CMD

ds3g_msh_hangup_call()

FLOW=OFF

call_ended_handler()

ps_phys_link_gone_ind()
ps_iface_down_ind()
ps_phys_link_enable_flow()

ppp_abort(PPP_UM_DEV)

IFACE=DOWN
PHY_LINK=DOWN
FLOW=ON

IFACE_DOWN event callback()

ds3g_siolib_set_inbound_flow(DS_FLOW_GEN_RMSM_MASK, DS_FLOW_DISABLE)

FLOW=OFF

ds3g_siolib_change_mode(AUTODETECT)
ds3g_siolib_set_cd_state(DS3G_SIOLIB_REMOTE_CARRIER_OFF)
dsat_send_result_code(DSAT_NO_CARRIER)

Deregister events with UMTS_IFACE
Cleanup PKT mode WM queues
Deregister handlers with SIOLIB, ATCOP

NO_CARRIER

ppp_abort(PPP_RM_DEV)

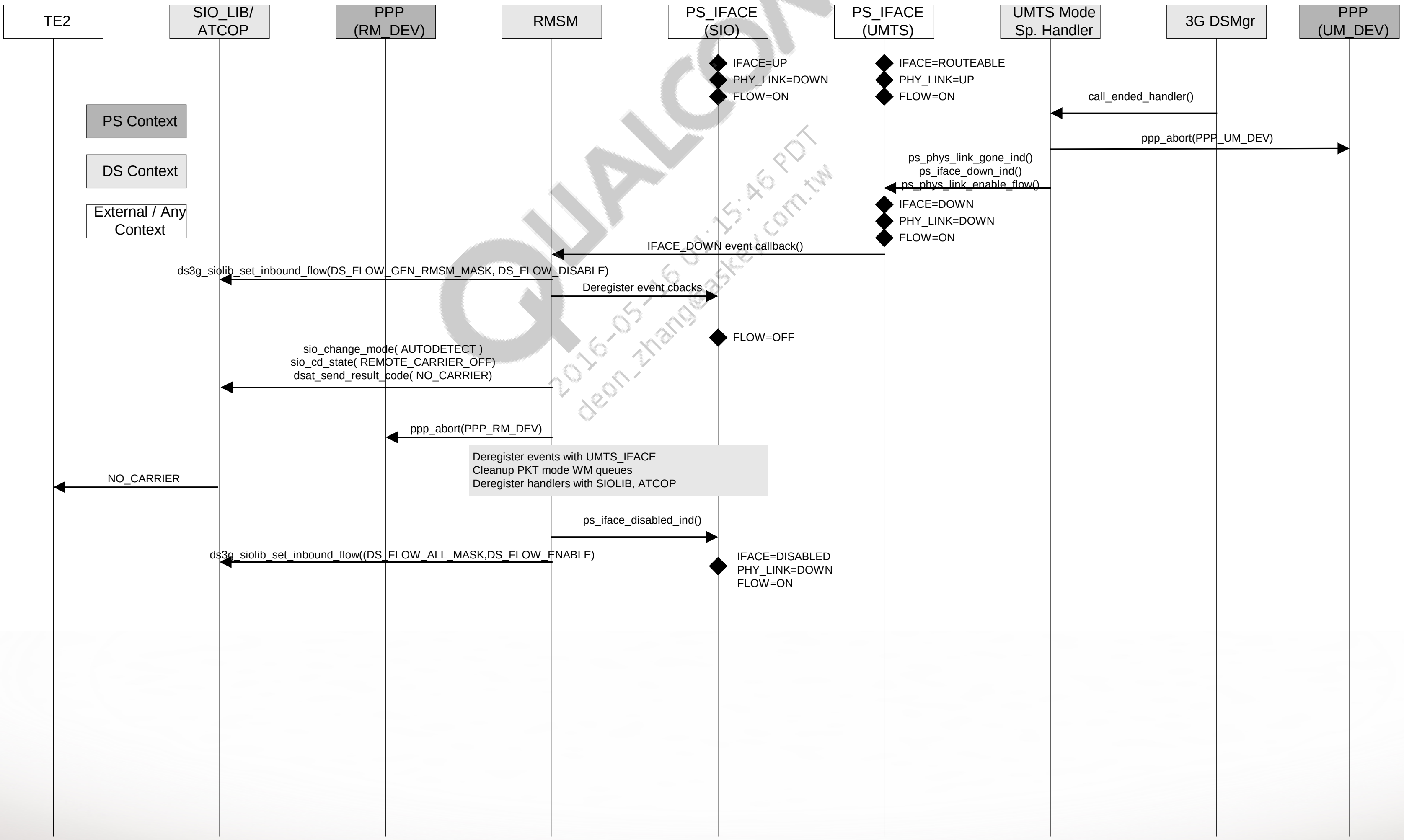
ps_iface_disabled_ind()

IFACE=DISABLED
PHY_LINK=DOWN
FLOW=ON

ds3g_siolib_set_inbound_flow((DS_FLOW_ALL_MASK,DS_FLOW_ENABLE)

PDP-PPP Network Call Release

PDP-PPP Network Call Release



TE2

SIO_LIB/ ATCOP

PPP (RM_DEV)

RMSM

PS_IFACE (SIO)

PS_IFACE (UMTS)

UMTS Mode Sp. Handler

3G DSMgr

PPP (UM_DEV)

PS Context

DS Context

External / Any Context

IFACE=UP
PHY_LINK=DOWN
FLOW=ON

IFACE=ROUTEABLE
PHY_LINK=UP
FLOW=ON

call_ended_handler()

ppp_abort(PPP_UM_DEV)

ps_phys_link_gone_ind()
ps_iface_down_ind()
ps_phys_link_enable_flow()

IFACE=DOWN
PHY_LINK=DOWN
FLOW=ON

IFACE_DOWN event callback()

ds3g_siolib_set_inbound_flow(DS_FLOW_GEN_RMSM_MASK, DS_FLOW_DISABLE)

Deregister event cbacks

FLOW=OFF

sio_change_mode(AUTODETECT)
sio_cd_state(REMOTE_CARRIER_OFF)
dsat_send_result_code(NO_CARRIER)

ppp_abort(PPP_RM_DEV)

Deregister events with UMTS_IFACE
Cleanup PKT mode WM queues
Deregister handlers with SIOLIB, ATCOP

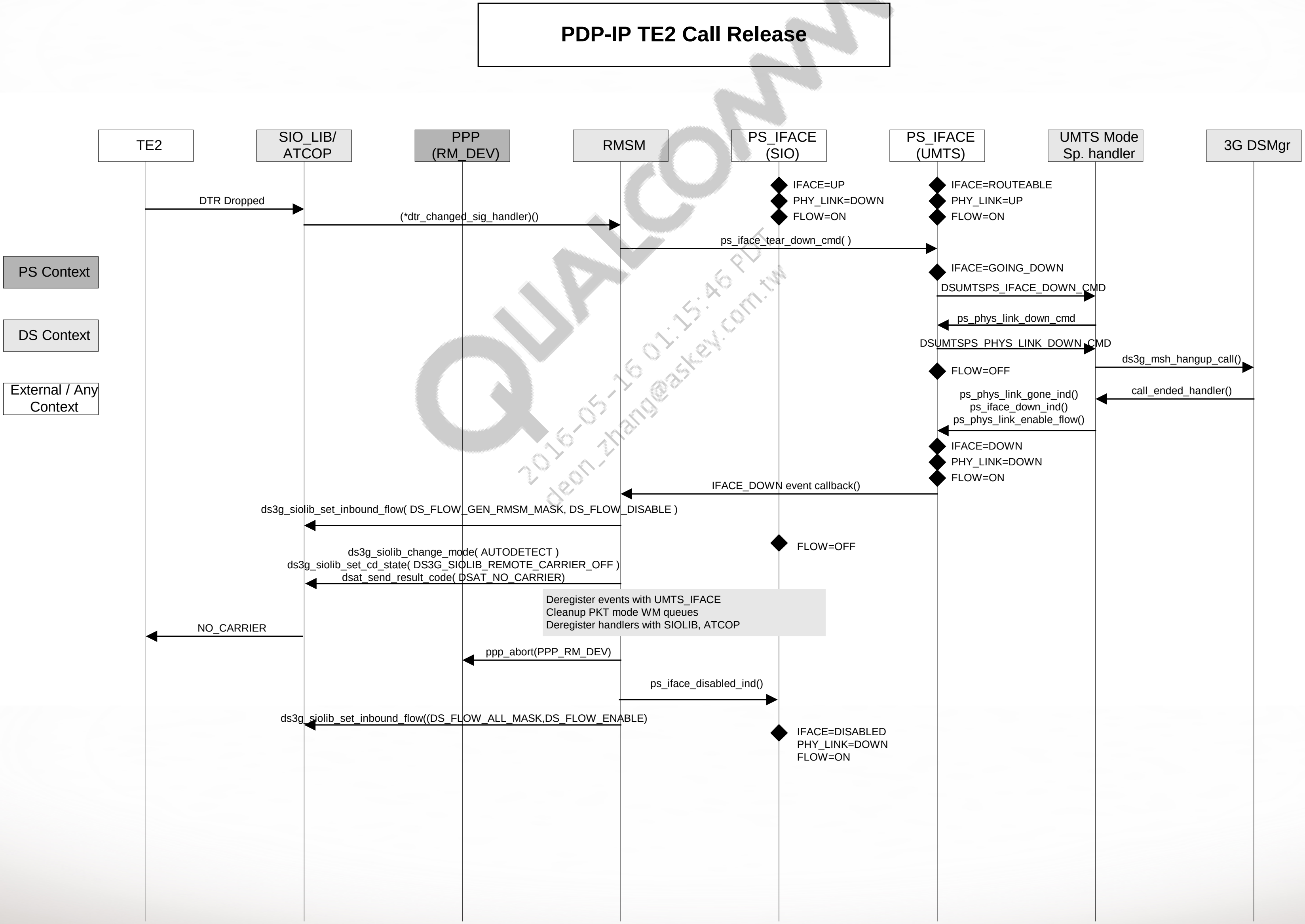
NO_CARRIER

ps_iface_disabled_ind()

ds3g_siolib_set_inbound_flow((DS_FLOW_ALL_MASK,DS_FLOW_ENABLE)

IFACE=DISABLED
PHY_LINK=DOWN
FLOW=ON

PDP-IP UE-Initiated Call Release



PDP-IP TE2 Call Release

TE2

SIO_LIB/ ATCOP

PPP (RM_DEV)

RMSM

PS_IFACE (SIO)

PS_IFACE (UMTS)

UMTS Mode Sp. handler

3G DSMgr

PS Context

DS Context

External / Any Context

IFACE=UP
PHY_LINK=DOWN
FLOW=ON

IFACE=ROUTEABLE
PHY_LINK=UP
FLOW=ON

DTR Dropped

(*dtr_changed_sig_handler)()

ps_iface_tear_down_cmd( )

IFACE=GOING_DOWN

DSUMTSPS_IFACE_DOWN_CMD

ps_phys_link_down_cmd

DSUMTSPS_PHYS_LINK_DOWN_CMD

ds3g_msh_hangup_call()

FLOW=OFF

call_ended_handler()

ps_phys_link_gone_ind()
ps_iface_down_ind()
ps_phys_link_enable_flow()

IFACE=DOWN
PHY_LINK=DOWN
FLOW=ON

IFACE_DOWN event callback()

ds3g_siolib_set_inbound_flow(DS_FLOW_GEN_RMSM_MASK, DS_FLOW_DISABLE)

FLOW=OFF

ds3g_siolib_change_mode(AUTODETECT)
ds3g_siolib_set_cd_state(DS3G_SIOLIB_REMOTE_CARRIER_OFF)
dsat_send_result_code(DSAT_NO_CARRIER)

Deregister events with UMTS_IFACE
Cleanup PKT mode WM queues
Deregister handlers with SIOLIB, ATCOP

NO_CARRIER

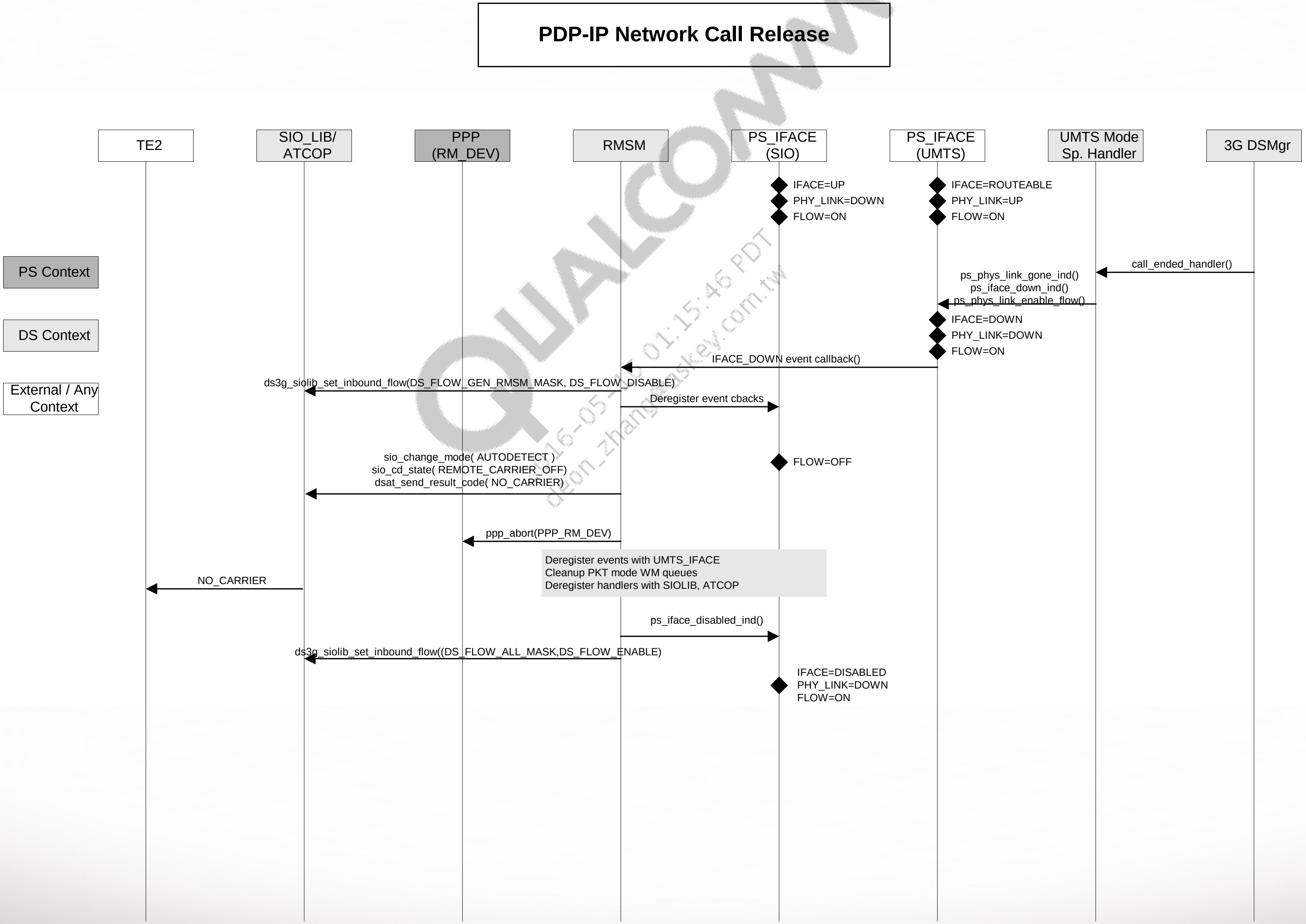
ppp_abort(PPP_RM_DEV)

ps_iface_disabled_ind()

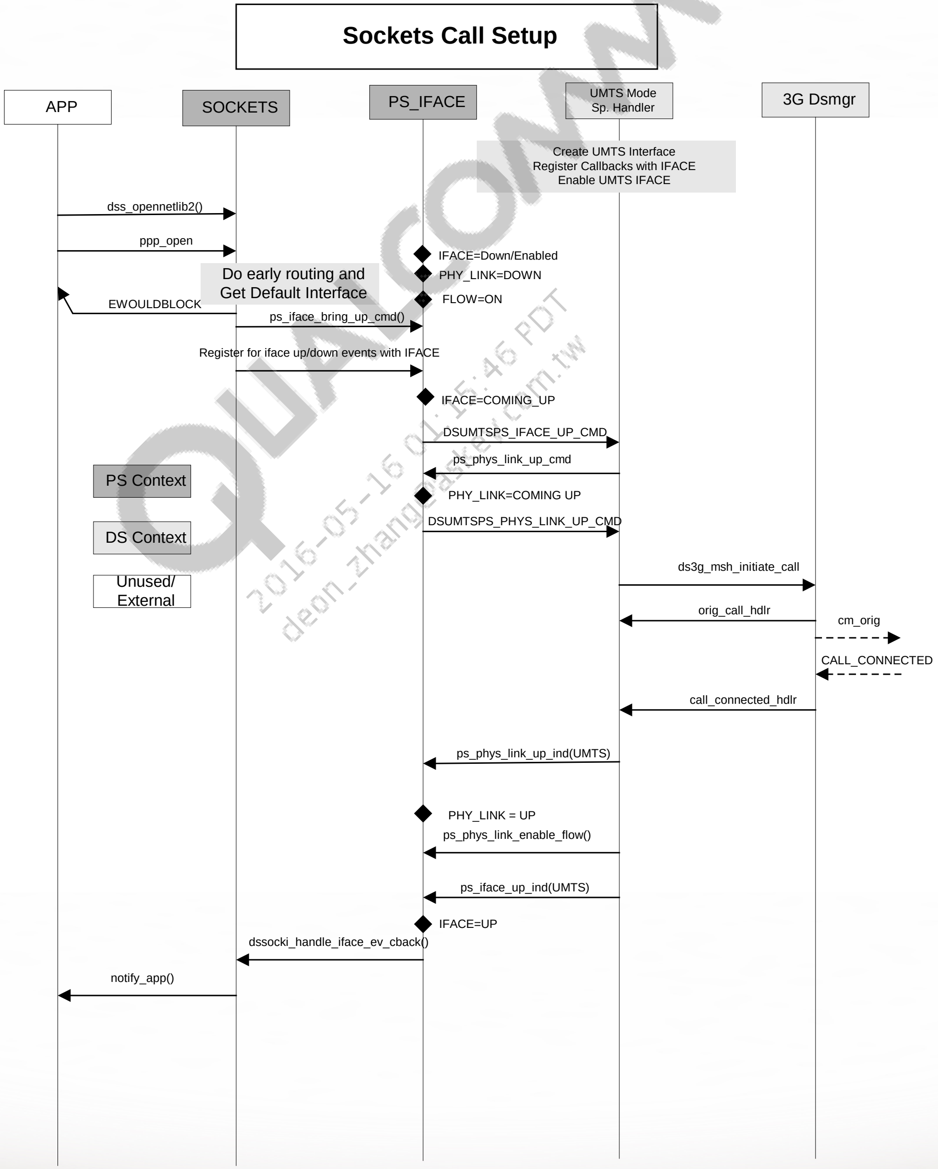
ds3g_siolib_set_inbound_flow((DS_FLOW_ALL_MASK,DS_FLOW_ENABLE)

IFACE=DISABLED
PHY_LINK=DOWN
FLOW=ON

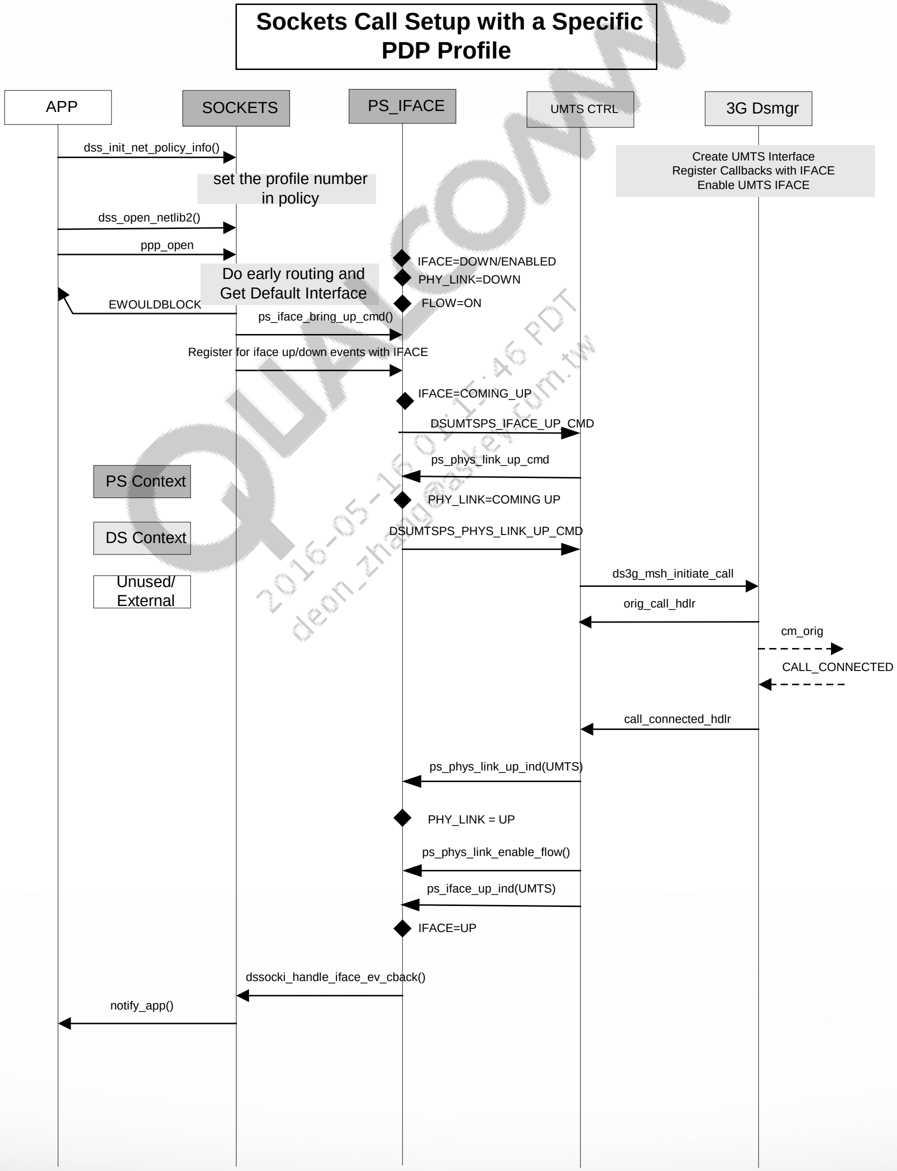
PDP-IP Network Call Release



Sockets Primary PDP Context Setup (PDP-IP)



PDP-IP – Application Specifies Profile Number



Sockets Call Setup with a Specific PDP Profile

APP

SOCKETS

PS_IFACE

UMTS CTRL

3G Dsmgr

Create UMTS Interface
Register Callbacks with IFACE
Enable UMTS IFACE

dss_init_net_policy_info()

set the profile number in policy

dss_open_netlib2()

ppp_open

IFACE=DOWN/ENABLED

PHY_LINK=DOWN

Do early routing and Get Default Interface

EWOULDBLOCK

FLOW=ON

ps_iface_bring_up_cmd()

Register for iface up/down events with IFACE

IFACE=COMING_UP

DSUMTSPS_IFACE_UP_CMD

ps_phys_link_up_cmd

PS Context

PHY_LINK=COMING UP

DS Context

DSUMTSPS_PHYS_LINK_UP_CMD

ds3g_msh_initiate_call

Unused/ External

orig_call_hdlr

cm_orig

CALL_CONNECTED

call_connected_hdlr

ps_phys_link_up_ind(UMTS)

PHY_LINK = UP

ps_phys_link_enable_flow()

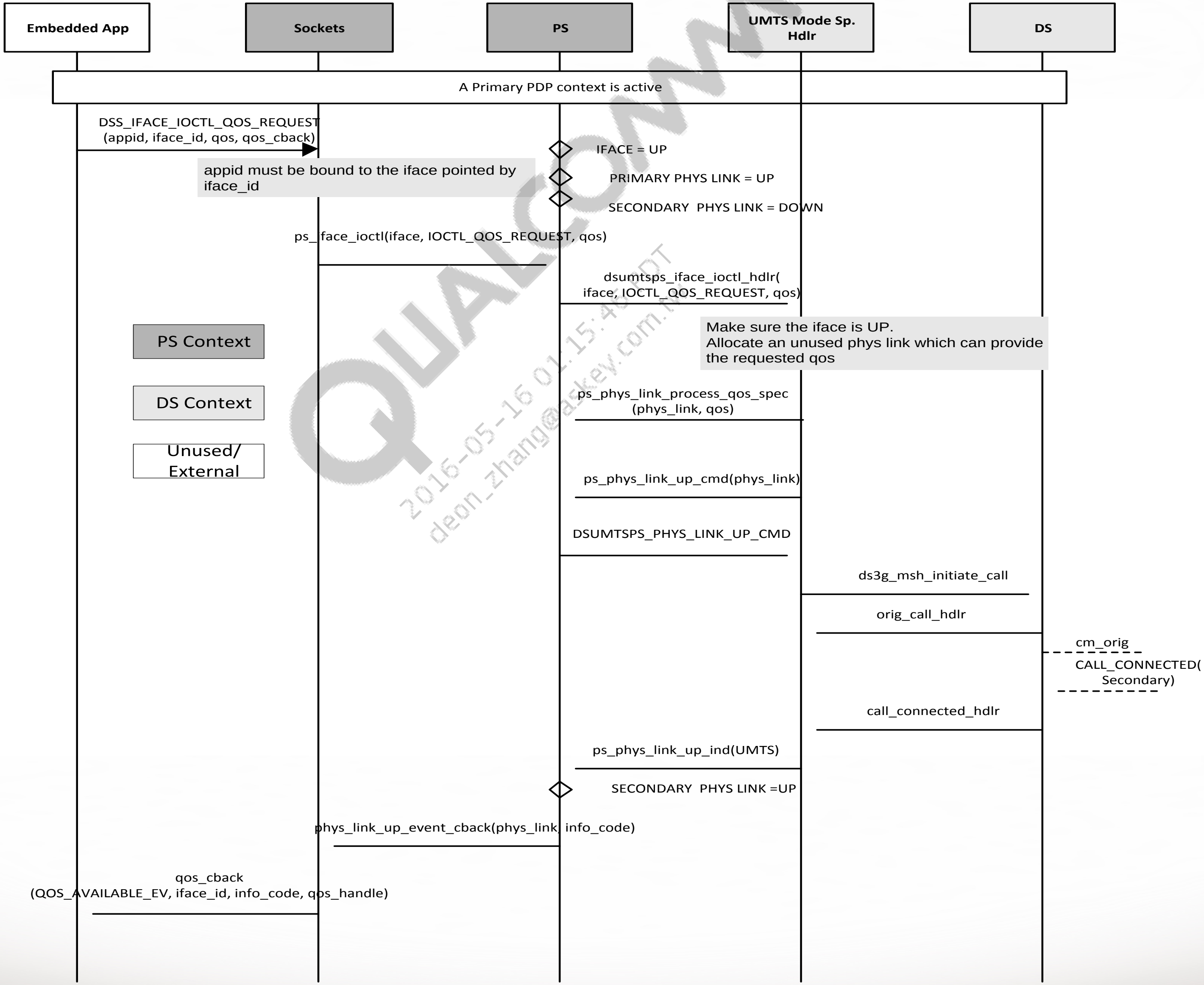
ps_iface_up_ind(UMTS)

IFACE=UP

dssocki_handle_iface_ev_cback()

notify_app()

Sockets Secondary PDP Context Setup



Embedded App

Sockets

PS

UMTS Mode Sp. Hdlr

DS

A Primary PDP context is active

DSS_IFACE_IOCTL_QOS_REQUEST (appid, iface_id, qos, qos_cback)

appid must be bound to the iface pointed by iface_id

IFACE = UP

PRIMARY PHYS LINK = UP

SECONDARY PHYS LINK = DOWN

ps_iface_ioctl(iface, IOCTL_QOS_REQUEST, qos)

dsumtsps_iface_ioctl_hdlr(iface, IOCTL_QOS_REQUEST, qos)

Make sure the iface is UP.
Allocate an unused phys link which can provide the requested qos

PS Context

DS Context

Unused/ External

ps_phys_link_process_qos_spec (phys_link, qos)

ps_phys_link_up_cmd(phys_link)

DSUMTSPS_PHYS_LINK_UP_CMD

ds3g_msh_initiate_call

orig_call_hdlr

cm_orig

CALL_CONNECTED(Secondary)

call_connected_hdlr

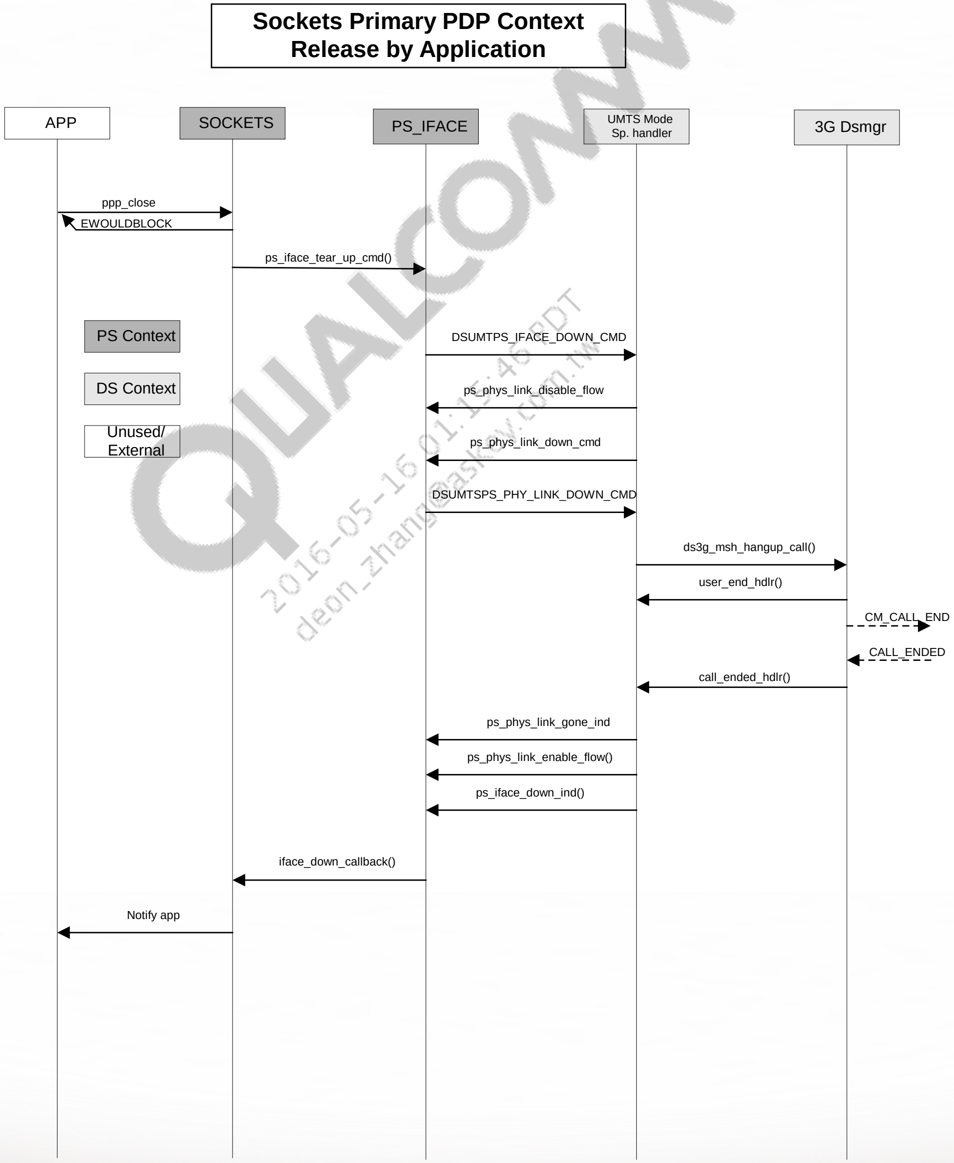
ps_phys_link_up_ind(UMTS)

SECONDARY PHYS LINK =UP

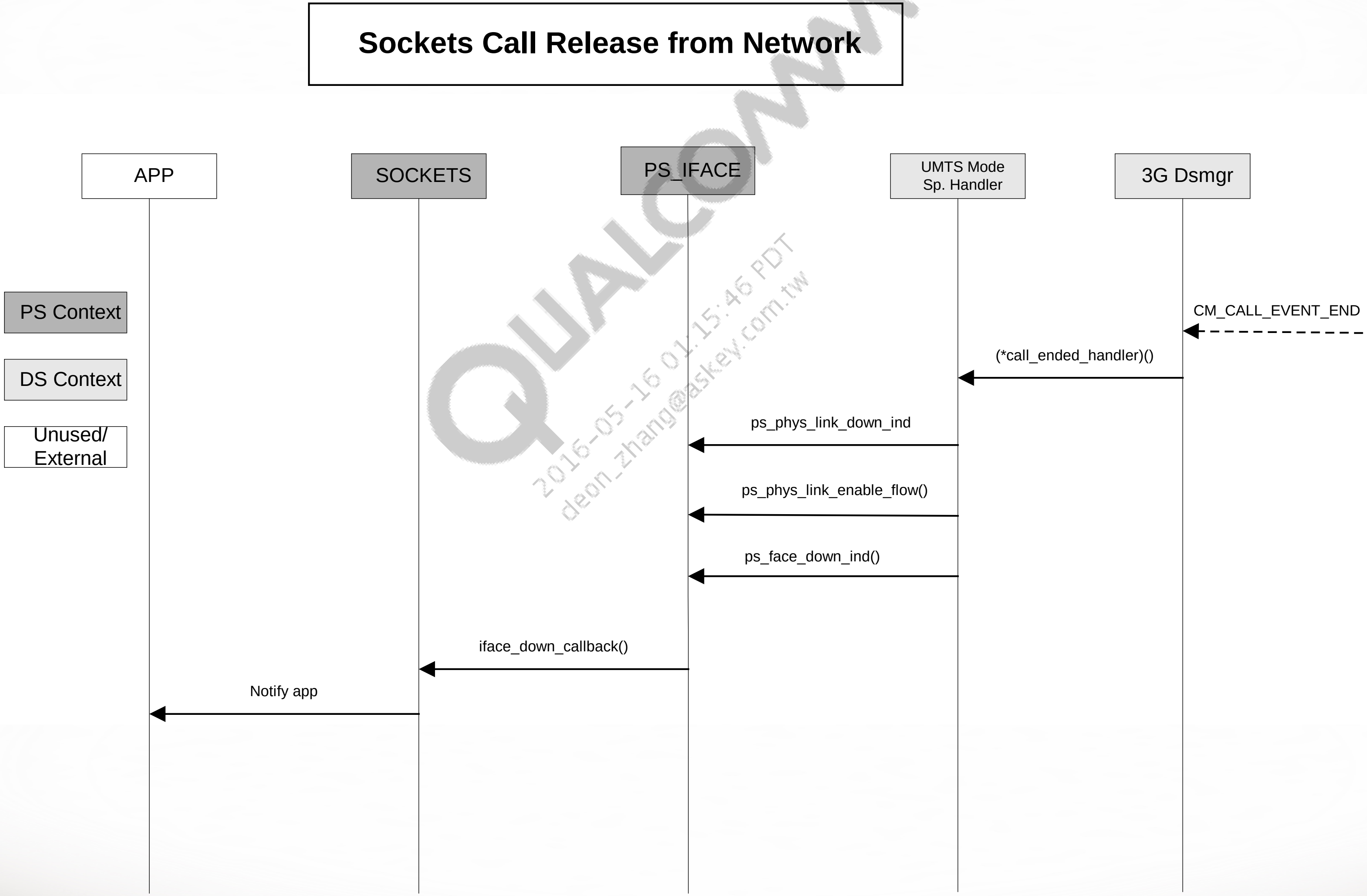
phys_link_up_event_cback(phys_link, info_code)

qos_cback (QOS_AVAILABLE_EV, iface_id, info_code, qos_handle)

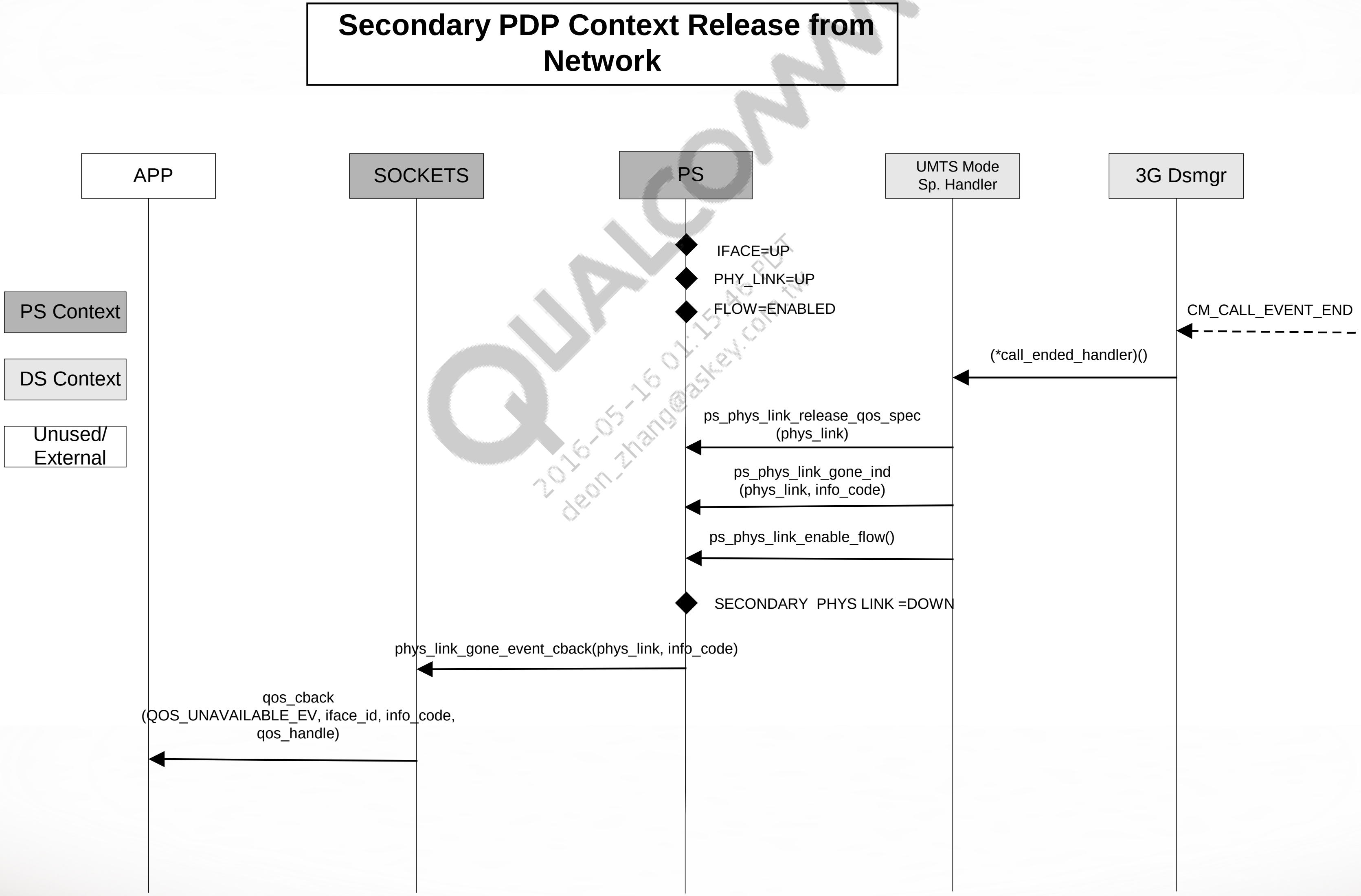
Sockets Primary PDP Context Release by Application



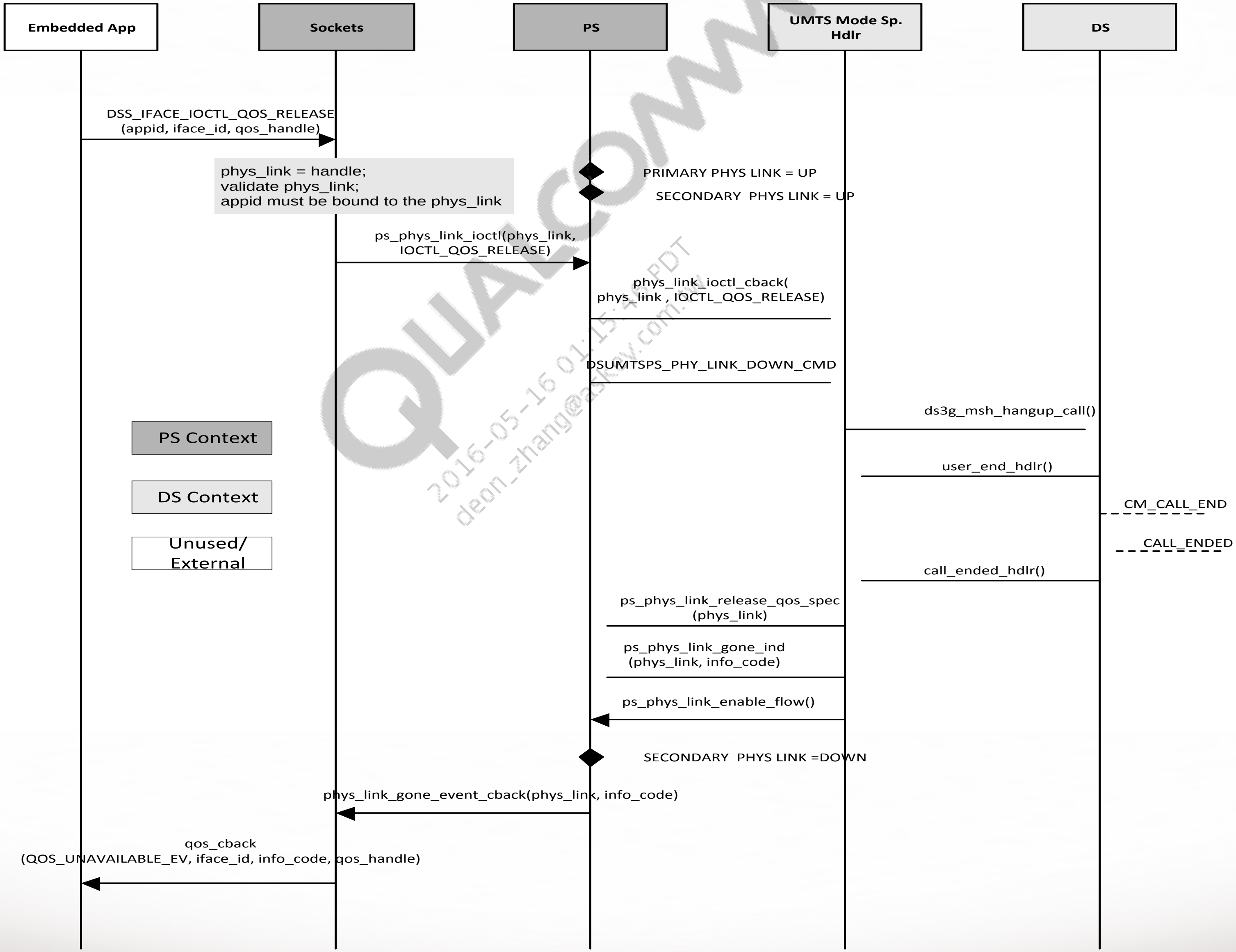
Sockets Primary PDP Context Release from Network



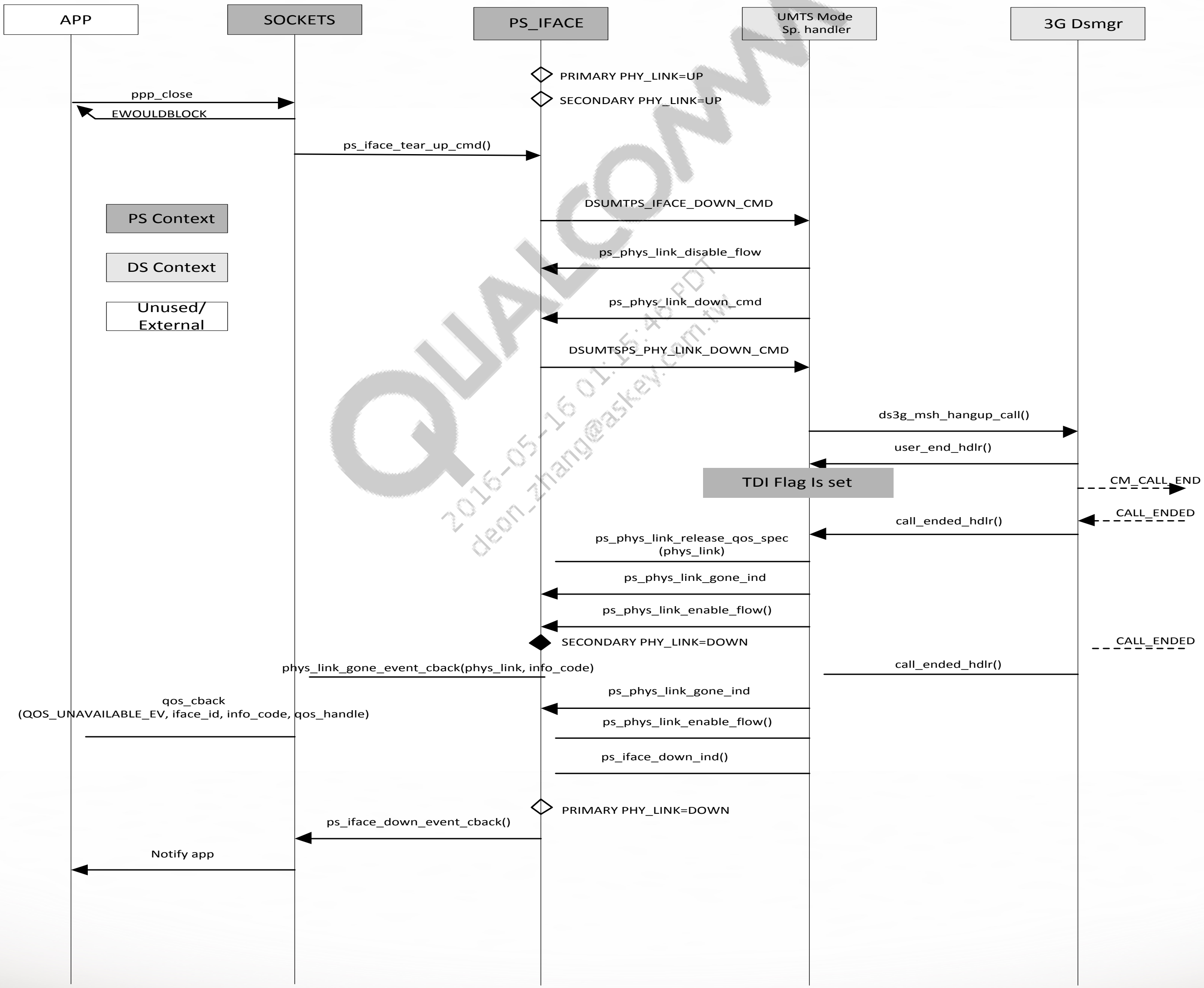
Secondary PDP Context Release from Network


Secondary PDP Context Release from Network
APP
SOCKETS
PS
UMTS Mode Sp. Handler
3G Dsmgr
PS Context
DS Context
Unused/ External
IFACE=UP
PHY_LINK=UP
FLOW=ENABLED
CM_CALL_EVENT_END
(*call_ended_handler)()
ps_phys_link_release_qos_spec (phys_link)
ps_phys_link_gone_ind (phys_link, info_code)
ps_phys_link_enable_flow()
SECONDARY PHYS LINK =DOWN
phys_link_gone_event_cback(phys_link, info_code)
qos_cback (QOS_UNAVAILABLE_EV, iface_id, info_code, qos_handle)

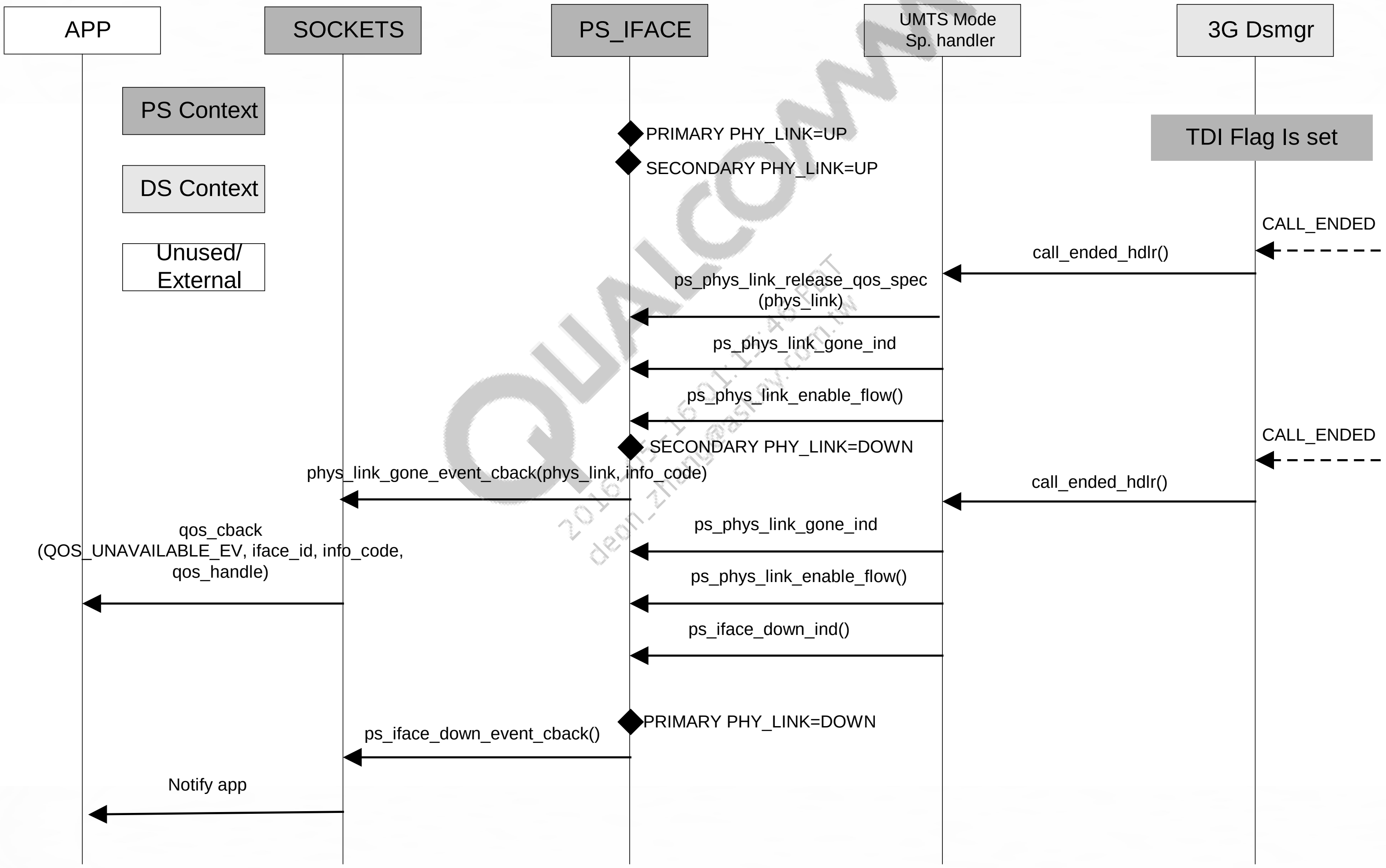
Sockets Secondary PDP Context Release by Application


Embedded App
Sockets
PS
UMTS Mode Sp. Hdlr
DS
DSS_IFACE_IOCTL_QOS_RELEASE (appid, iface_id, qos_handle)
phys_link = handle;
validate phys_link;
appid must be bound to the phys_link
PRIMARY PHYS LINK = UP
SECONDARY PHYS LINK = UP
ps_phys_link_ioctl(phys_link, IOCTL_QOS_RELEASE)
phys_link_ioctl_cback(phys_link , IOCTL_QOS_RELEASE)
DSUMTSPS_PHY_LINK_DOWN_CMD
ds3g_msh_hangup_call()
user_end_hdlr()
CM_CALL_END
CALL_ENDED
call_ended_hdlr()
ps_phys_link_release_qos_spec (phys_link)
ps_phys_link_gone_ind (phys_link, info_code)
ps_phys_link_enable_flow()
SECONDARY PHYS LINK =DOWN
phys_link_gone_event_cback(phys_link, info_code)
qos_cback (QOS_UNAVAILABLE_EV, iface_id, info_code, qos_handle)
PS Context
DS Context
Unused/ External

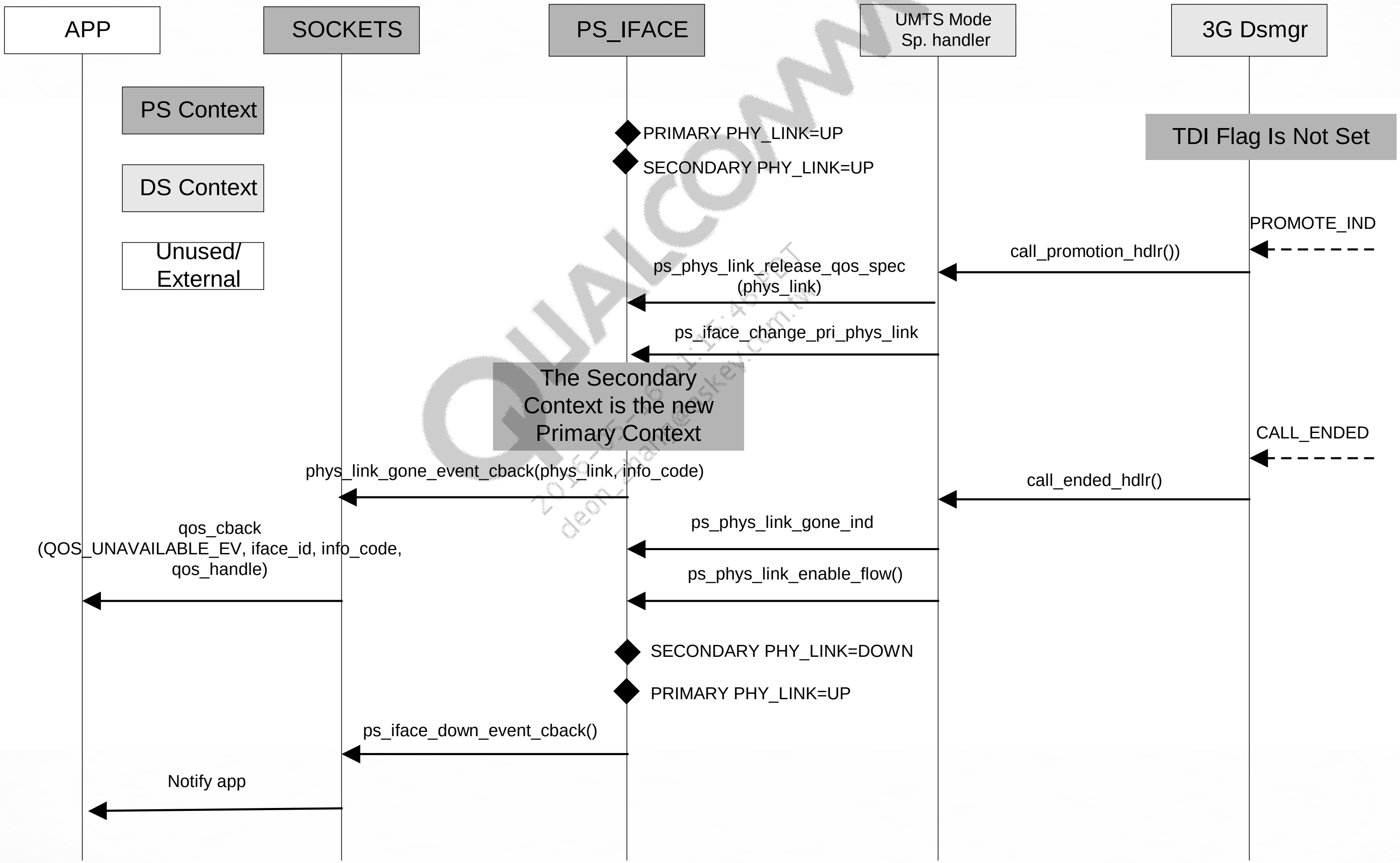
Sockets Primary PDP Context Release by Application with Active Secondary



Sockets Primary PDP Context Network Release with Active Secondary (TDI Set)



Sockets Primary PDP Context Network Release with Active Secondary (TDI Not Set)


APP
SOCKETS
PS_IFACE
UMTS Mode Sp. handler
3G Dsmgr
PS Context
DS Context
Unused/ External
TDI Flag Is Not Set
PRIMARY PHY_LINK=UP
SECONDARY PHY_LINK=UP
PROMOTE_IND
call_promotion_hdlr())
ps_phys_link_release_qos_spec (phys_link)
ps_iface_change_pri_phys_link
The Secondary Context is the new Primary Context
CALL_ENDED
phys_link_gone_event_cback(phys_link, info_code)
call_ended_hdlr()
ps_phys_link_gone_ind
qos_cback (QOS_UNAVAILABLE_EV, iface_id, info_code, qos_handle)
ps_phys_link_enable_flow()
SECONDARY PHY_LINK=DOWN
PRIMARY PHY_LINK=UP
ps_iface_down_event_cback()
Notify app

WCDMA Packet Data Bearer Services

- RLC is used in Acknowledged mode for all WCDMA Packet Data services
- PDP context preservation
  - RAB release and reestablishment during the lifetime of a PDP context is supported.
  - Network-initiated RAB release procedure is supported.
  - Network- and UE-initiated RAB reestablishment procedure is supported.

WCDMA PS RAB Configurations

- Supported RAB configurations (see [S3])
  - Interactive or Background/UL:64 DL:64 kbps/PS RAB + UL:3.4 DL 3.4 kbps SRBs for DCCH
  - Interactive or Background/UL:64 DL:128 kbps/PS RAB + UL:3.4 DL 3.4 kbps SRBs for DCCH
  - Interactive or Background/UL:64 DL:256 kbps/PS RAB + UL:3.4 DL 3.4 kbps SRBs for DCCH
  - Interactive or Background/UL:64 DL:384 kbps/PS RAB + UL:3.4 DL:3.4 kbps SRBs for DCCH
  - Interactive or background/UL:64 DL:64 kbps/PS RAB + Interactive or background/UL:64 DL:64 kbps/PS RAB + UL:3.4 DL:3.4 kbps SRBs for DCCH
  - Interactive or background/UL:384 DL:384 kbps/PS RAB + UL 3.4 DL 3.4 kbps SRBs for DCCH

GPRS Packet Data

- Mode of operation depends on the reliability class (negotiated during PDP Context Activation procedure)
  - Reliability Class 1 and 2 – LLC Acknowledged (Ack), LLC data protected, RLC Ack
  - Reliability Class 3 – LLC Unacknowledged (Unack), LLC data protected, RLC Ack
  - Reliability Class 4 – LLC Unack, LLC data protected, RLC Unack
  - Reliability Class 5 – LLC Unack, LLC data unprotected, RLC Unack
- All four types of GPRS channel encoding schemes are supported, resulting in the following data rates:
  - CS1/9.05 Kbps, CS2/13.4 Kbps, CS3/15.6 Kbps, and CS4/21.4 Kbps
- Multislot Class 1 to Class 10 are supported resulting in the following data rate range:
  - Minimum data rate (CS1 and Multislot Class 1) 9.05 Kbps DL, 9.05 Kbps UL
  - Maximum data rate (CS4 and Multislot Class 10) 85.6 Kbps DL, 21.4 Kbps UL (or 64.2 Kbps DL, 42.8 Kbps UL depending on the UL/DL slot allocation)

File Structure

- Common header file
  - dsumtsps.h – Common data declarations
- Mode-specific handlers
  - dsumtspshdlr.c, dsumtspshdlr.h – Top-level functions for mode-specific handler
  - dswpsdhdlr.c, dswpsdhdlr.h – WCDMA-specific handler functions
  - dsgpsdhdlr.c, dsgpsdhdlr.h – GPRS-specific handler functions
- Protocol stack data processing
  - dswpsdllif.c, dswpsdllif.h – Functions to handle data from/to WCDMA air interface
  - dsgpsdllif.c, dsgpsdllif.h – Functions to handle data from/to GPRS air interface

File Structure (cont.)

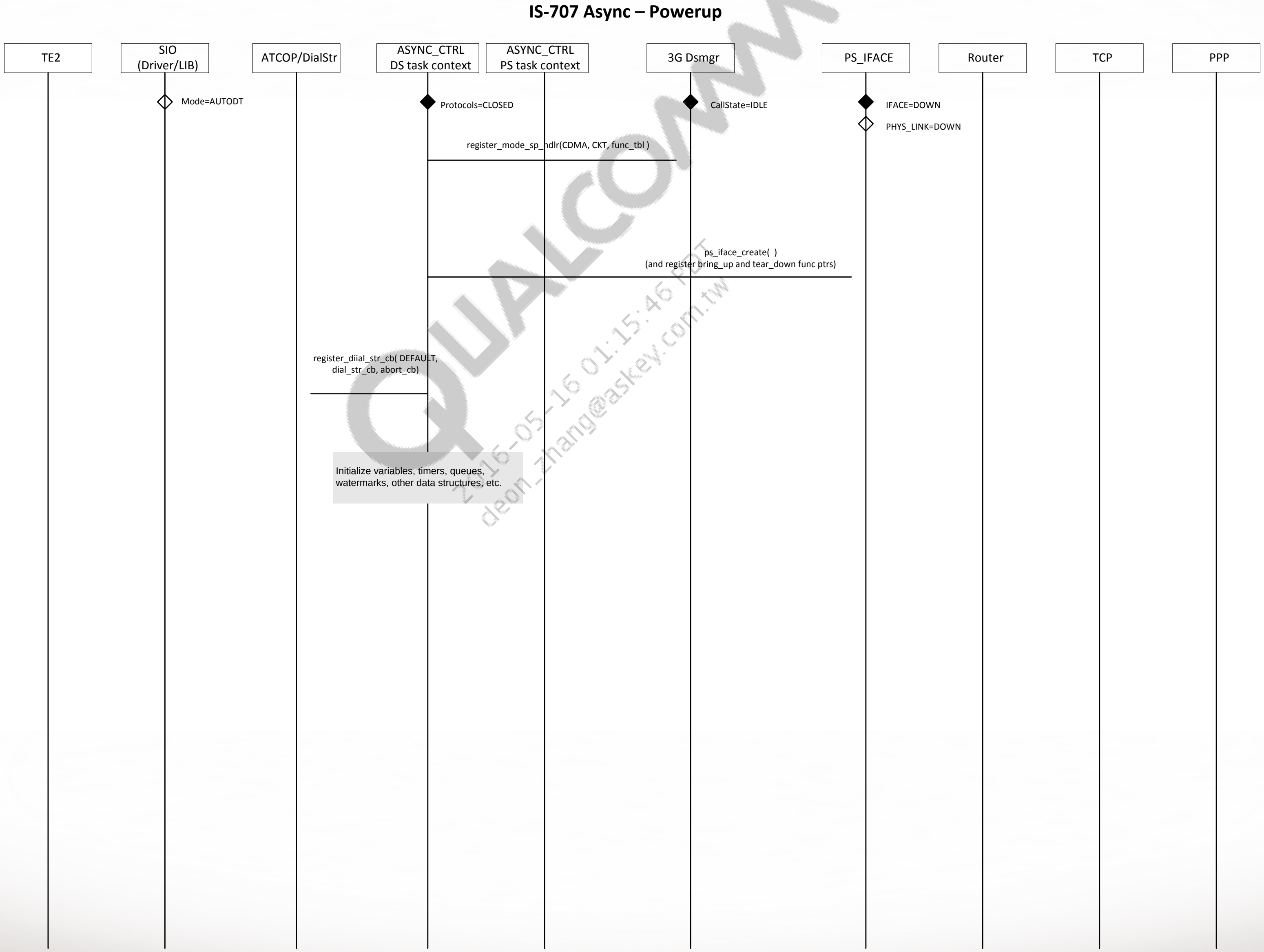
- Other
  - dsumtspspco.c, dsumtspspco.h – Functions to encode/decode protocol configuration packets in PDP context messages
  - dsumtspsmisc, dsumtspsmisc.h – Utility functions
  - dsumtsps_rt_acl.c, dsumtsps_rt_acl.h – UMTS-specific routing ACL used to enable policy and address-based routing across the UMTS interface
  - dsgen_rmsm.c, dsgen_rmsm.h – Functions to manage the Rm and Um interfaces for packet data calls initiated from TE
  - dsumtspsqos.c, dsumtspsqos.h – Functions for QoS parameter processing for secondary PDP contexts
  - dsumtspsmthdlr.c, dsumtspsmthdlr.h – Functions for MT PDP context setup for PDP-type IP

Feature Usage

- To include support for the WCDMA CS Data UDI Bearer Services, FEATURE_DATA_WCDMA_CS must be defined.
- To include GSM CS data support, FEATURE_DATA_GCSD must be defined.
- To include GSM fax support, FEATURE_DATA_GCSD_FAX must be defined.
- To include UMTS packet-data support, FEATURE_DATA_PS and FEATURE_DS_NET_MODEL must be defined.
- To include WCDMA packet-data support, FEATURE_DATA_WCDMA_PS must be defined.
- To include GPRS packet data support, FEATURE_GSM_GPRS must be defined.

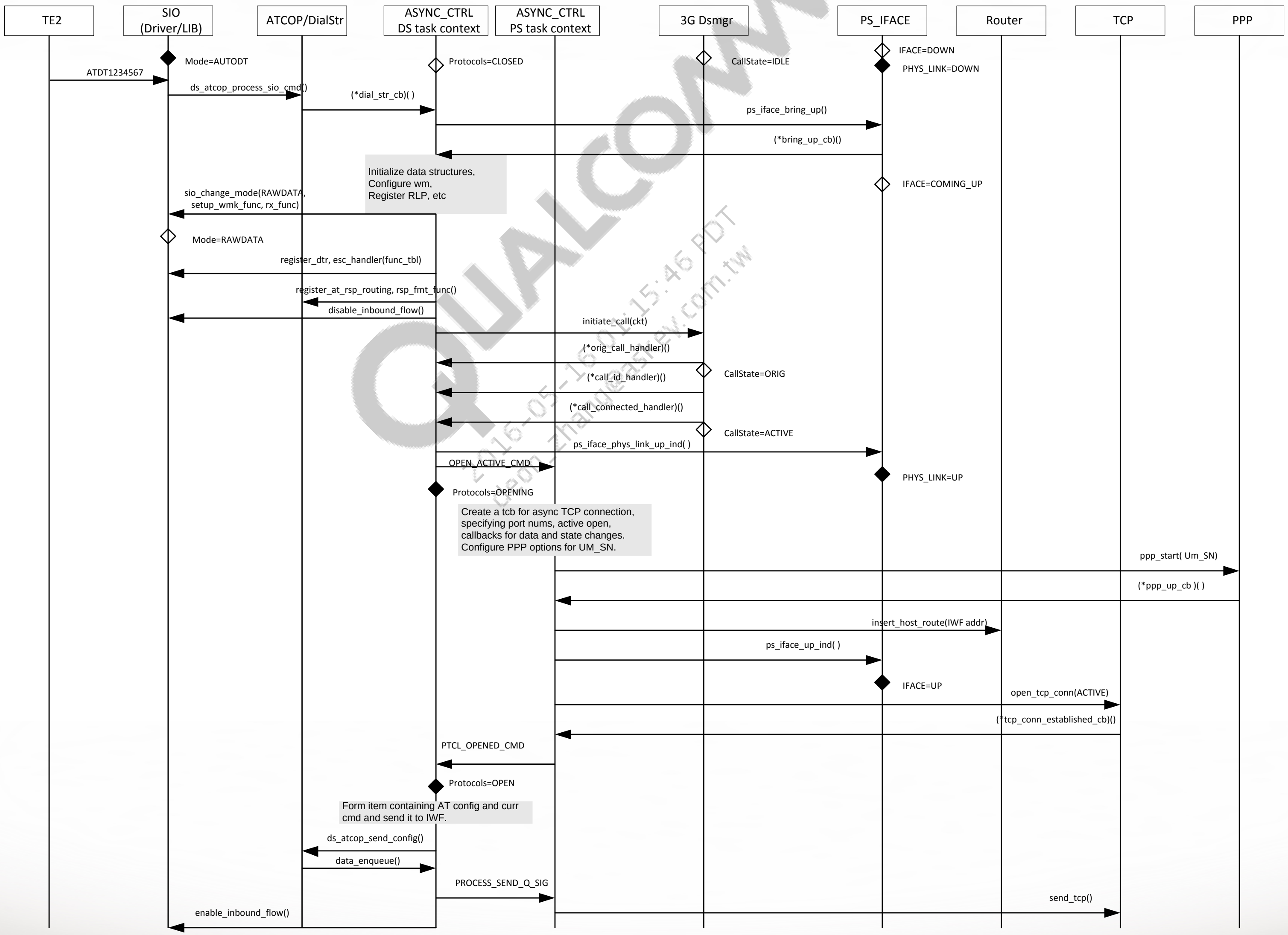
Appendix A – IS-707 Async and 1X Packet Call Flows

IS-707 Async – Powerup


IS-707 Async – Powerup
TE2
SIO (Driver/LIB)
ATCOP/DialStr
ASYNC_CTRL DS task context
ASYNC_CTRL PS task context
3G Dsmgr
PS_IFACE
Router
TCP
PPP
Mode=AUTODT
Protocols=CLOSED
CallState=IDLE
IFACE=DOWN
PHYS_LINK=DOWN
register_mode_sp_hdlr(CDMA, CKT, func_tbl)
ps_iface_create( )
(and register bring_up and tear_down func ptrs)
register_diial_str_cb(DEFAULT,
dial_str_cb, abort_cb)
Initialize variables, timers, queues, watermarks, other data structures, etc.

IS-707 Async – Call Setup (MO)

IS-707 Async – Call Setup (Mobile-Originated)



TE2

SIO (Driver/LIB)

ATCOP/DialStr

ASYNC_CTRL DS task context

ASYNC_CTRL PS task context

3G Dsmgr

PS_IFACE

Router

TCP

PPP

Mode=AUTODT

Protocols=CLOSED

CallState=IDLE

IFACE=DOWN

PHYS_LINK=DOWN

ATDT1234567

ds_atcop_process_sio_cmd()

(*dial_str_cb)( )

ps_iface_bring_up()

(*bring_up_cb)()

Initialize data structures,
Configure wm,
Register RLP, etc

IFACE=COMING_UP

sio_change_mode(RAWDATA, setup_wmk_func, rx_func)

Mode=RAWDATA

register_dtr, esc_handler(func_tbl)

register_at_rsp_routing, rsp_fmt_func()

disable_inbound_flow()

initiate_call(ckt)

(*orig_call_handler)()

CallState=ORIG

(*call_id_handler)()

(*call_connected_handler)()

CallState=ACTIVE

ps_iface_phys_link_up_ind( )

OPEN_ACTIVE_CMD

PHYS_LINK=UP

Protocols=OPENING

Create a tcb for async TCP connection, specifying port nums, active open, callbacks for data and state changes.
Configure PPP options for UM_SN.

ppp_start(Um_SN)

(*ppp_up_cb)( )

insert_host_route(IWF addr)

ps_iface_up_ind( )

IFACE=UP

open_tcp_conn(ACTIVE)

(*tcp_conn_established_cb)()

PTCL_OPENED_CMD

Protocols=OPEN

Form item containing AT config and curr cmd and send it to IWF.

ds_atcop_send_config()

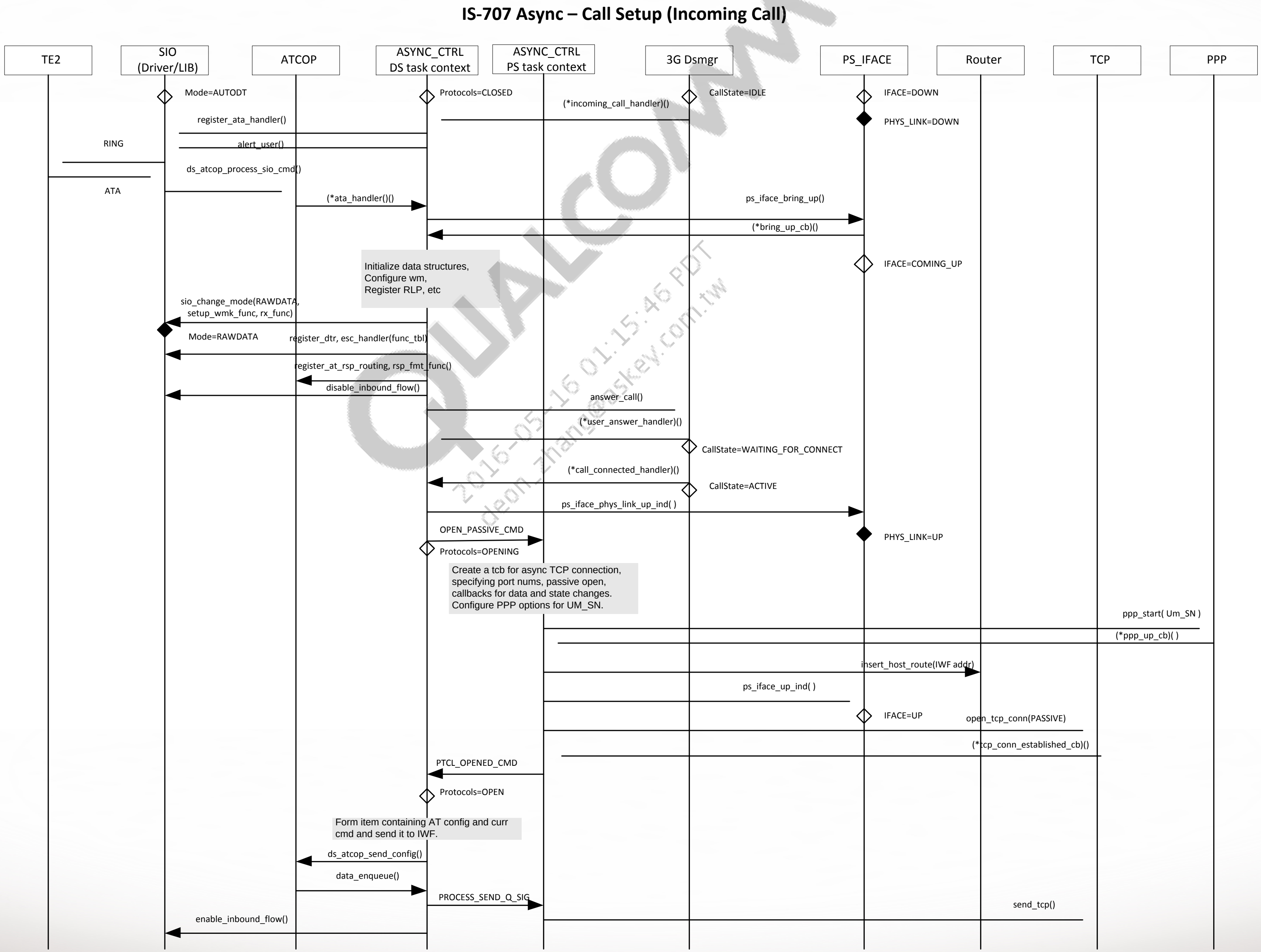
data_enqueue()

PROCESS_SEND_Q_SIG

send_tcp()

enable_inbound_flow()

IS-707 Async – Call Setup (Incoming Call)



IS-707 Async – Call Setup (Incoming Call)

TE2

SIO (Driver/LIB)

ATCOP

ASYNC_CTRL DS task context

ASYNC_CTRL PS task context

3G Dsmgr

PS_IFACE

Router

TCP

PPP

Mode=AUTODT

Protocols=CLOSED

(*incoming_call_handler)()

CallState=IDLE

IFACE=DOWN

register_ata_handler()

PHYS_LINK=DOWN

RING

alert_user()

ds_atcop_process_sio_cmd()

ATA

(*ata_handler()()

ps_iface_bring_up()

(*bring_up_cb)()

IFACE=COMING_UP

Initialize data structures,
Configure wm,
Register RLP, etc

sio_change_mode(RAWDATA,
setup_wmk_func, rx_func)

Mode=RAWDATA

register_dtr, esc_handler(func_tbl)

register_at_rsp_routing, rsp_fmt_func()

disable_inbound_flow()

answer_call()

(*user_answer_handler)()

CallState=WAITING_FOR_CONNECT

(*call_connected_handler)()

CallState=ACTIVE

ps_iface_phys_link_up_ind( )

OPEN_PASSIVE_CMD

PHYS_LINK=UP

Protocols=OPENING

Create a tcb for async TCP connection,
specifying port nums, passive open,
callbacks for data and state changes.
Configure PPP options for UM_SN.

ppp_start(Um_SN)

(*ppp_up_cb)( )

insert_host_route(IWF addr)

ps_iface_up_ind( )

IFACE=UP

open_tcp_conn(PASSIVE)

(*tcp_conn_established_cb)()

PTCL_OPENED_CMD

Protocols=OPEN

Form item containing AT config and curr
cmd and send it to IWF.

ds_atcop_send_config()

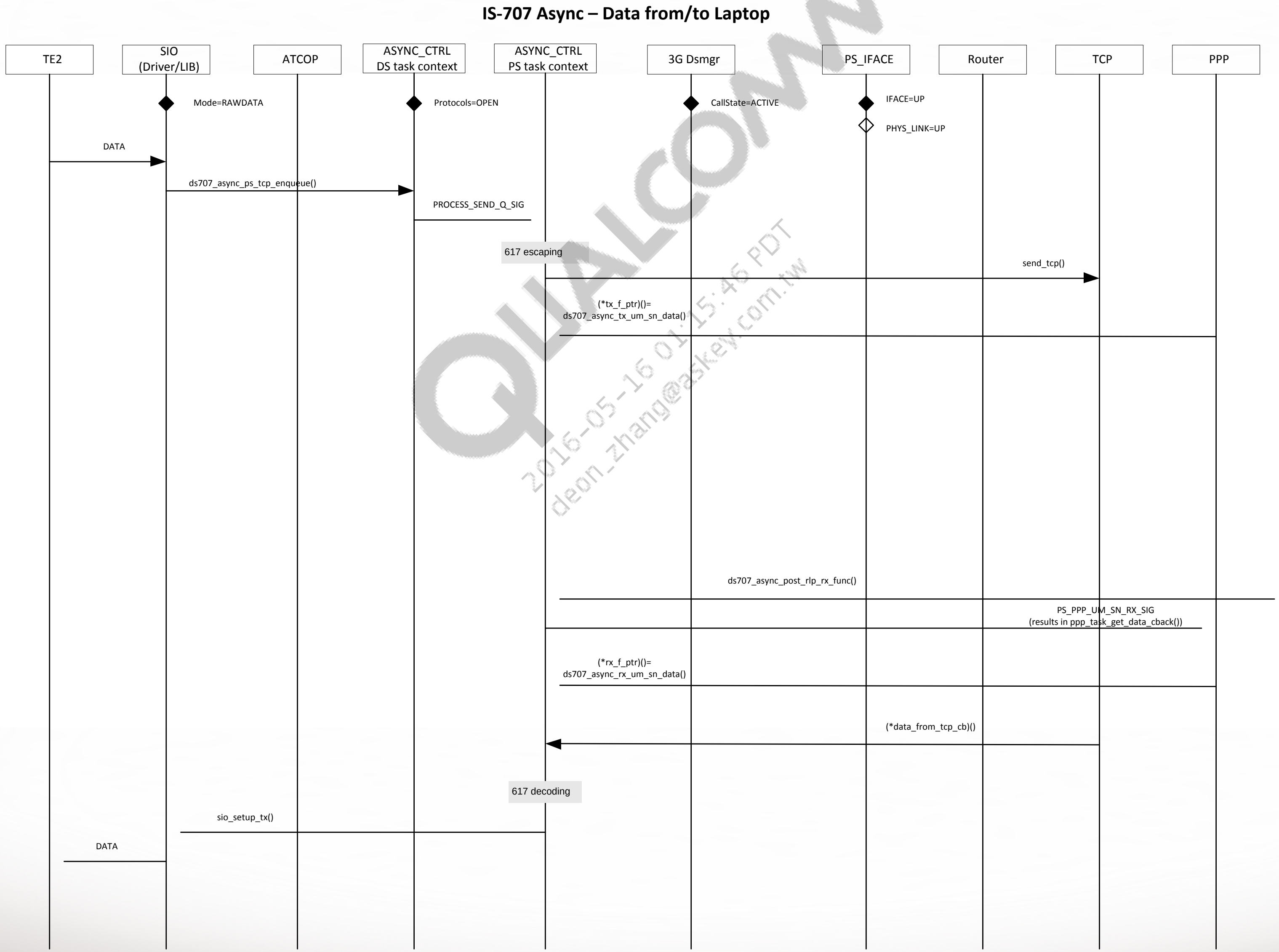
data_enqueue()

PROCESS_SEND_Q_SIG

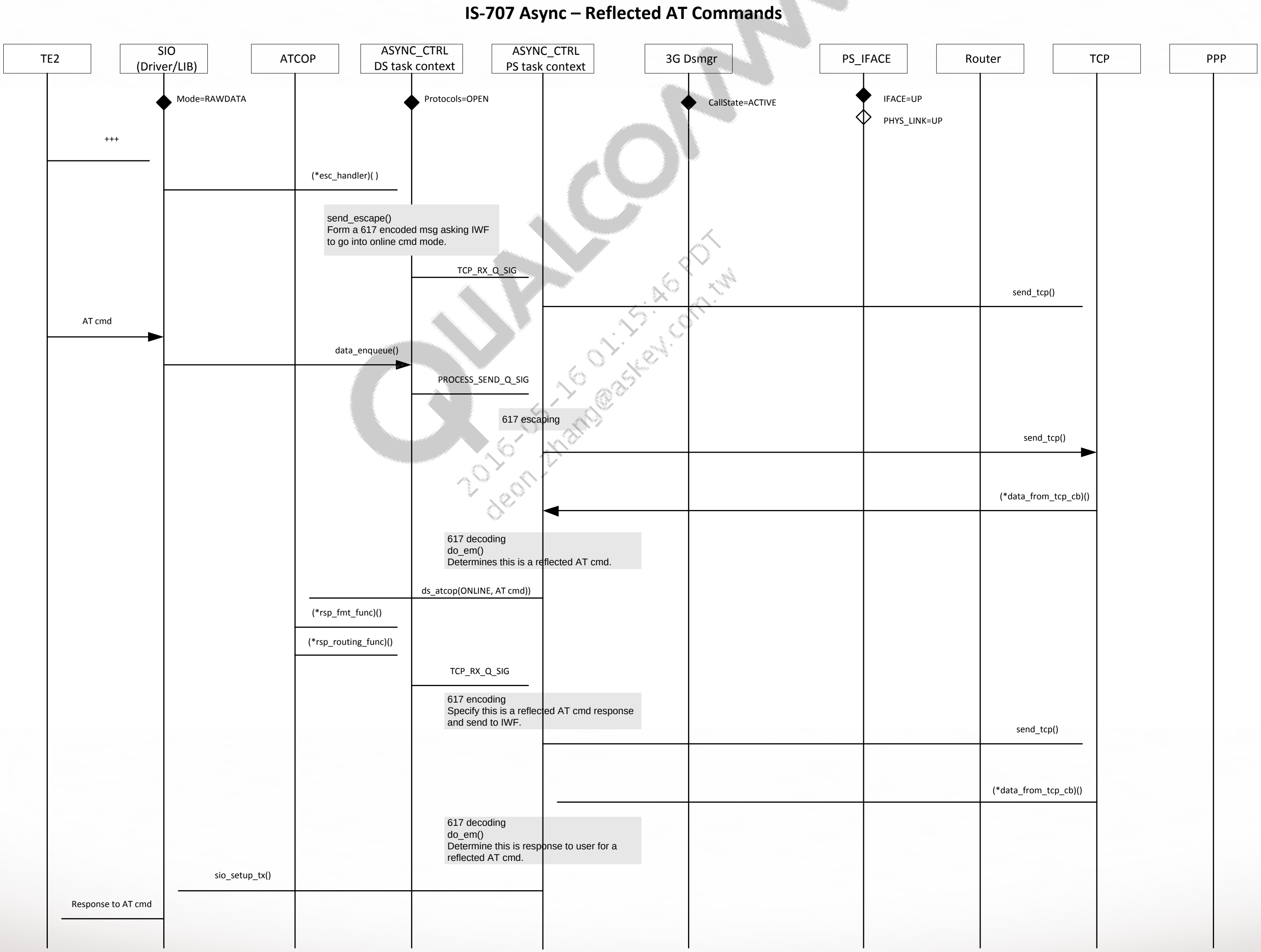
send_tcp()

enable_inbound_flow()

IS-707 Async – Data from/to Laptop



IS-707 Async – Reflected AT Commands



IS-707 Async – Reflected AT Commands

TE2

SIO (Driver/LIB)

ATCOP

ASYNC_CTRL DS task context

ASYNC_CTRL PS task context

3G Dsmgr

PS_IFACE

Router

TCP

PPP

Mode=RAWDATA

Protocols=OPEN

CallState=ACTIVE

IFACE=UP

PHYS_LINK=UP

+++

(*esc_handler)( )

send_escape()
Form a 617 encoded msg asking IWF to go into online cmd mode.

TCP_RX_Q_SIG

send_tcp()

AT cmd

data_enqueue()

PROCESS_SEND_Q_SIG

617 escaping

send_tcp()

(*data_from_tcp_cb)()

617 decoding
do_em()
Determines this is a reflected AT cmd.

ds_atcop(ONLINE, AT cmd))

(*rsp_fmt_func)()

(*rsp_routing_func)()

TCP_RX_Q_SIG

617 encoding
Specify this is a reflected AT cmd response and send to IWF.

send_tcp()

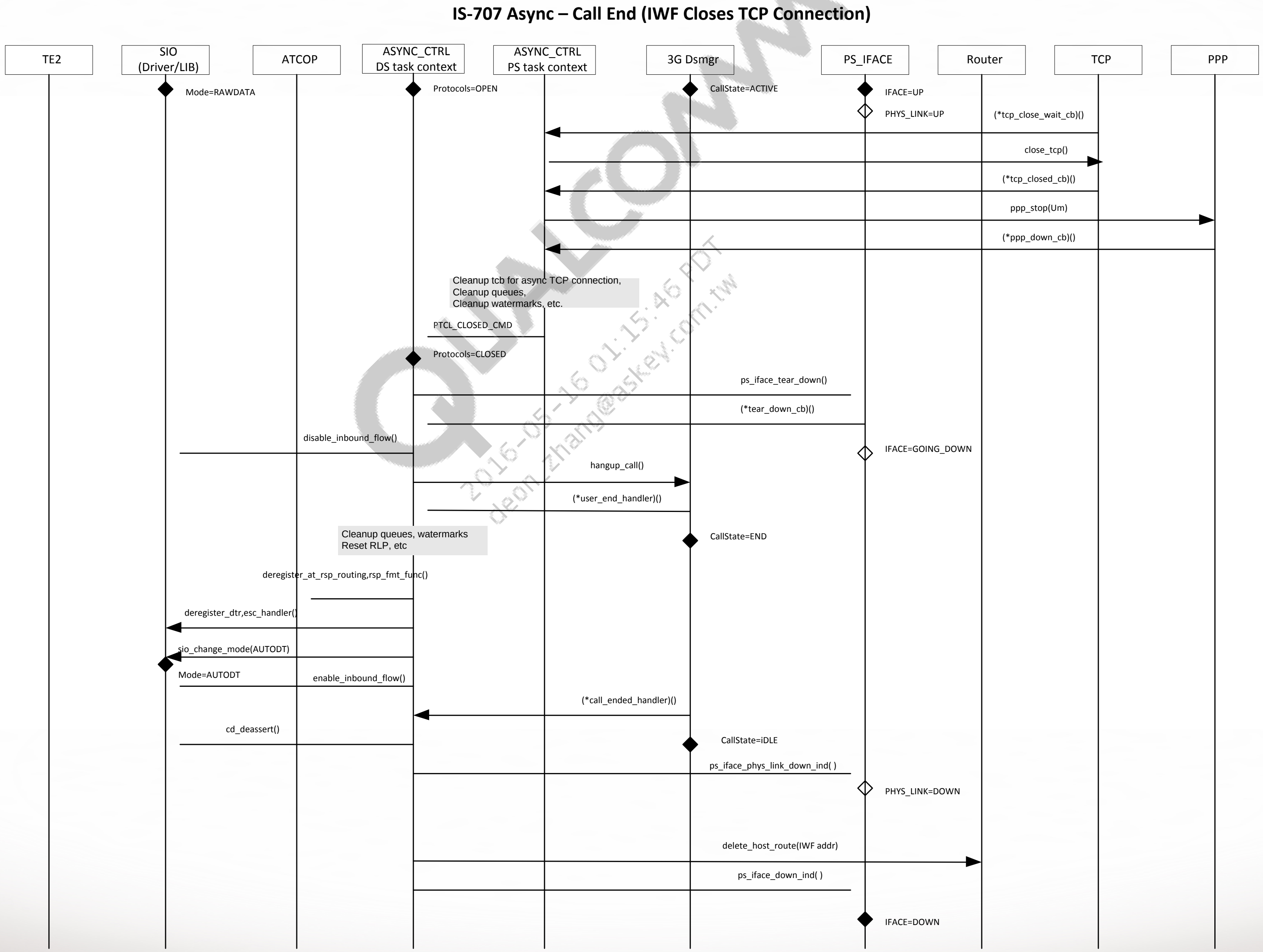
(*data_from_tcp_cb)()

617 decoding
do_em()
Determine this is response to user for a reflected AT cmd.

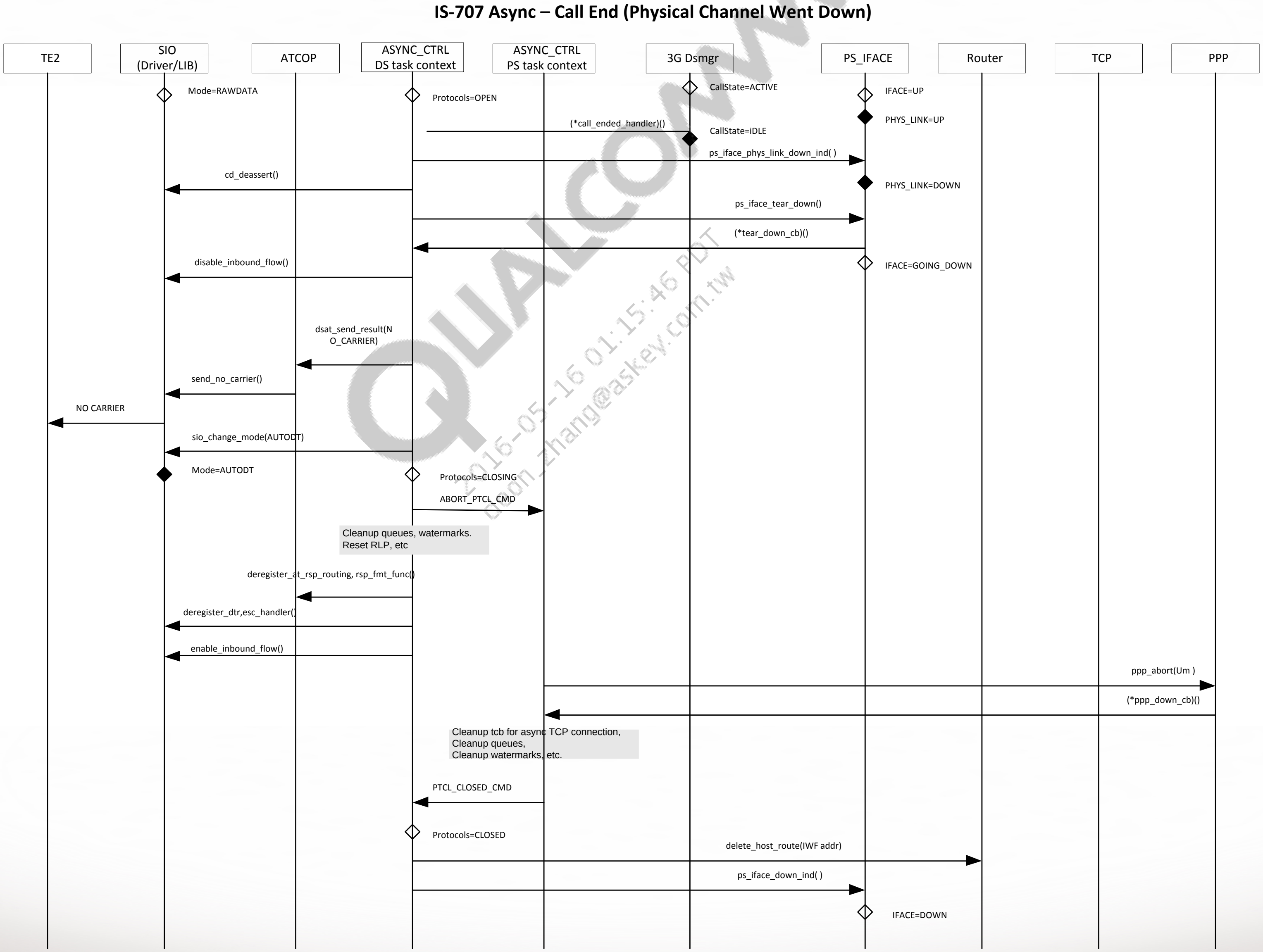
sio_setup_tx()

Response to AT cmd

IS-707 Async – Call End (IWF Closes TCP Connection)

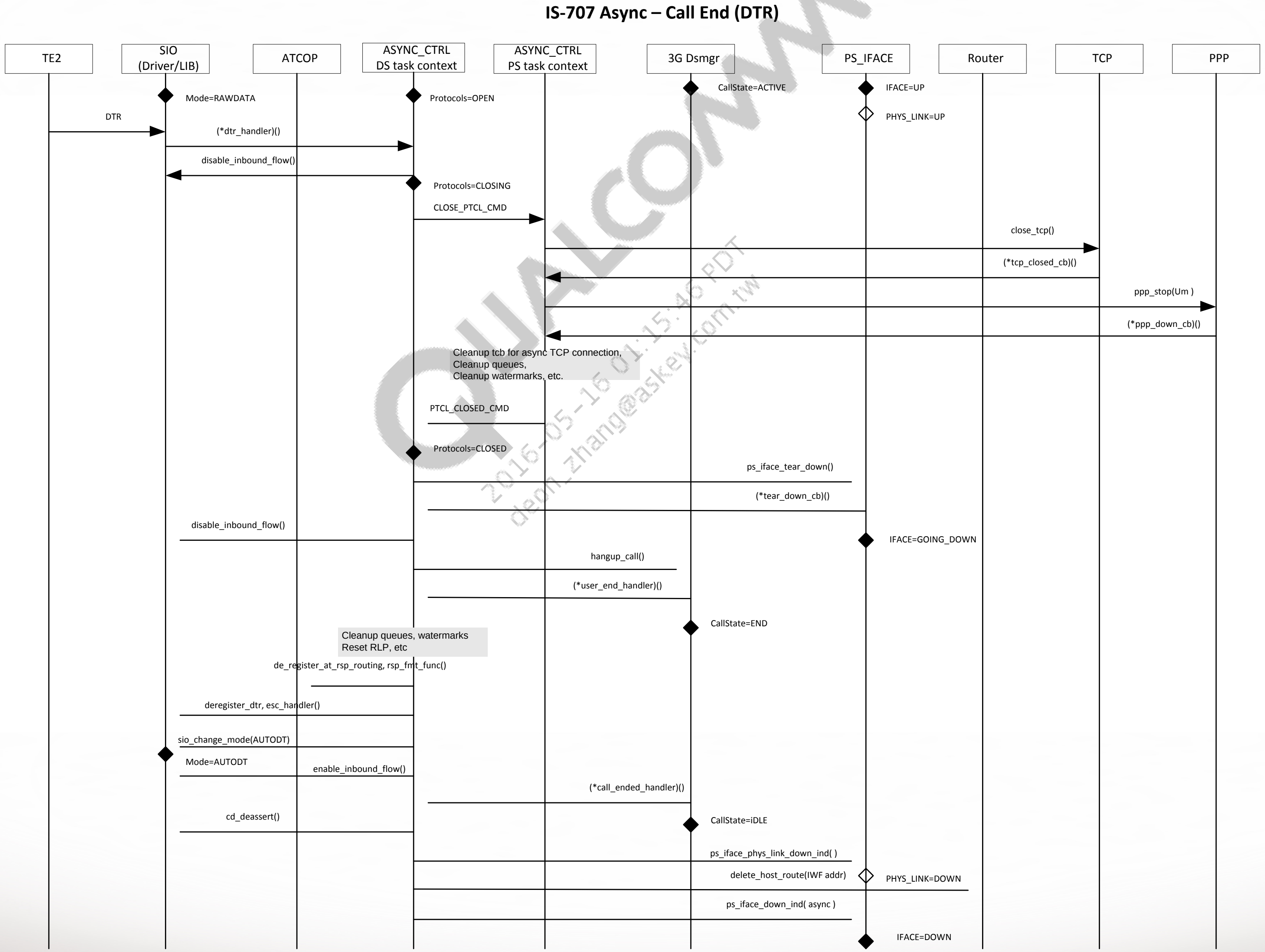


IS-707 Async – Call End (Physical Channel Went Down)



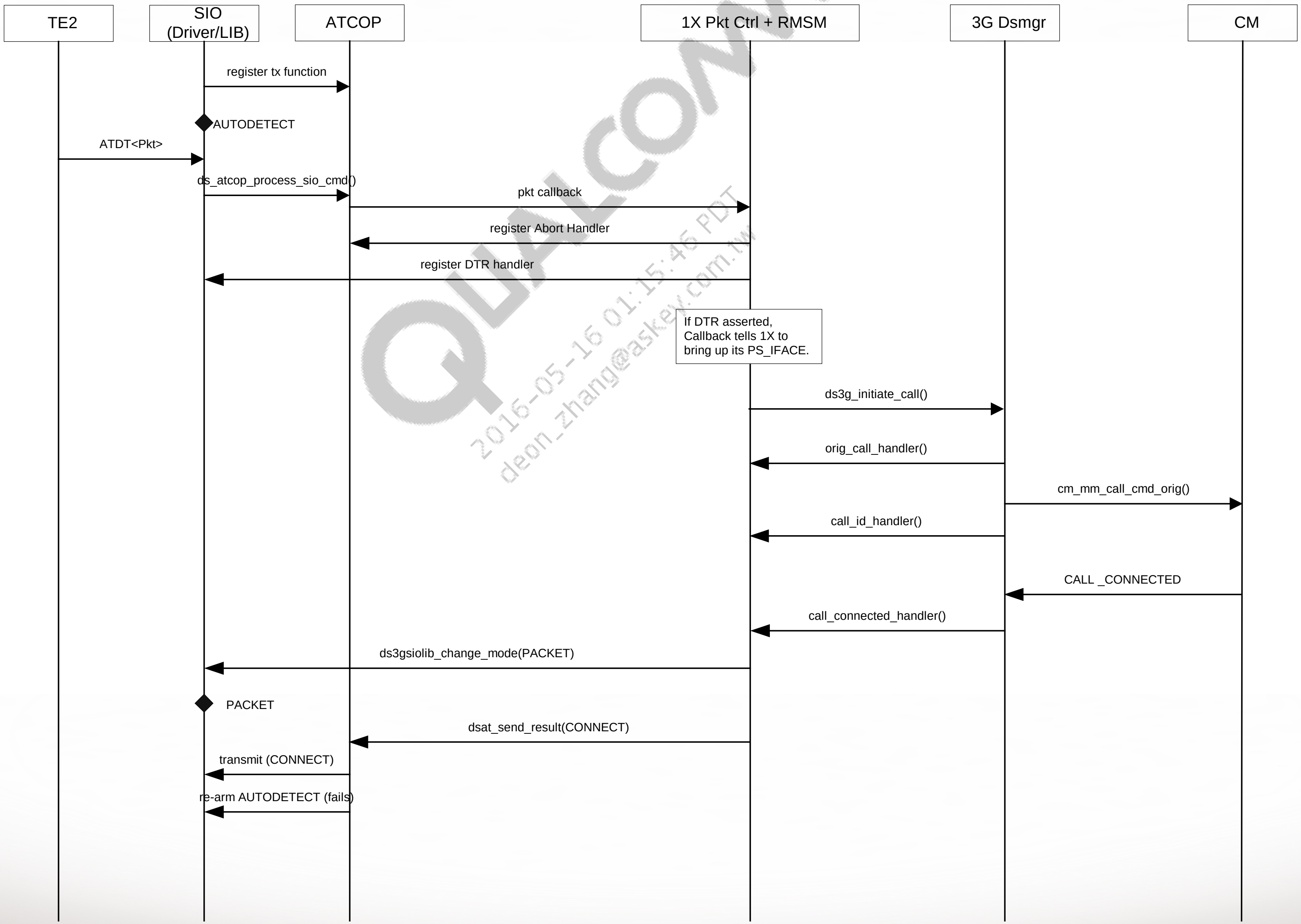
IS-707 Async – Call End (Physical Channel Went Down)

IS-707 Async – Call End (DTR)



1X Packet Origination (TE)

1X Packet Origination (TE)



TE2

SIO (Driver/LIB)

ATCOP

1X Pkt Ctrl + RMSM

3G Dsmgr

CM

register tx function

AUTODETECT

ATDT<Pkt>

ds_atcop_process_sio_cmd()

pkt callback

register Abort Handler

register DTR handler

If DTR asserted, Callback tells 1X to bring up its PS_IFACE.

ds3g_initiate_call()

orig_call_handler()

cm_mm_call_cmd_orig()

call_id_handler()

CALL _CONNECTED

call_connected_handler()

ds3gsiolib_change_mode(PACKET)

PACKET

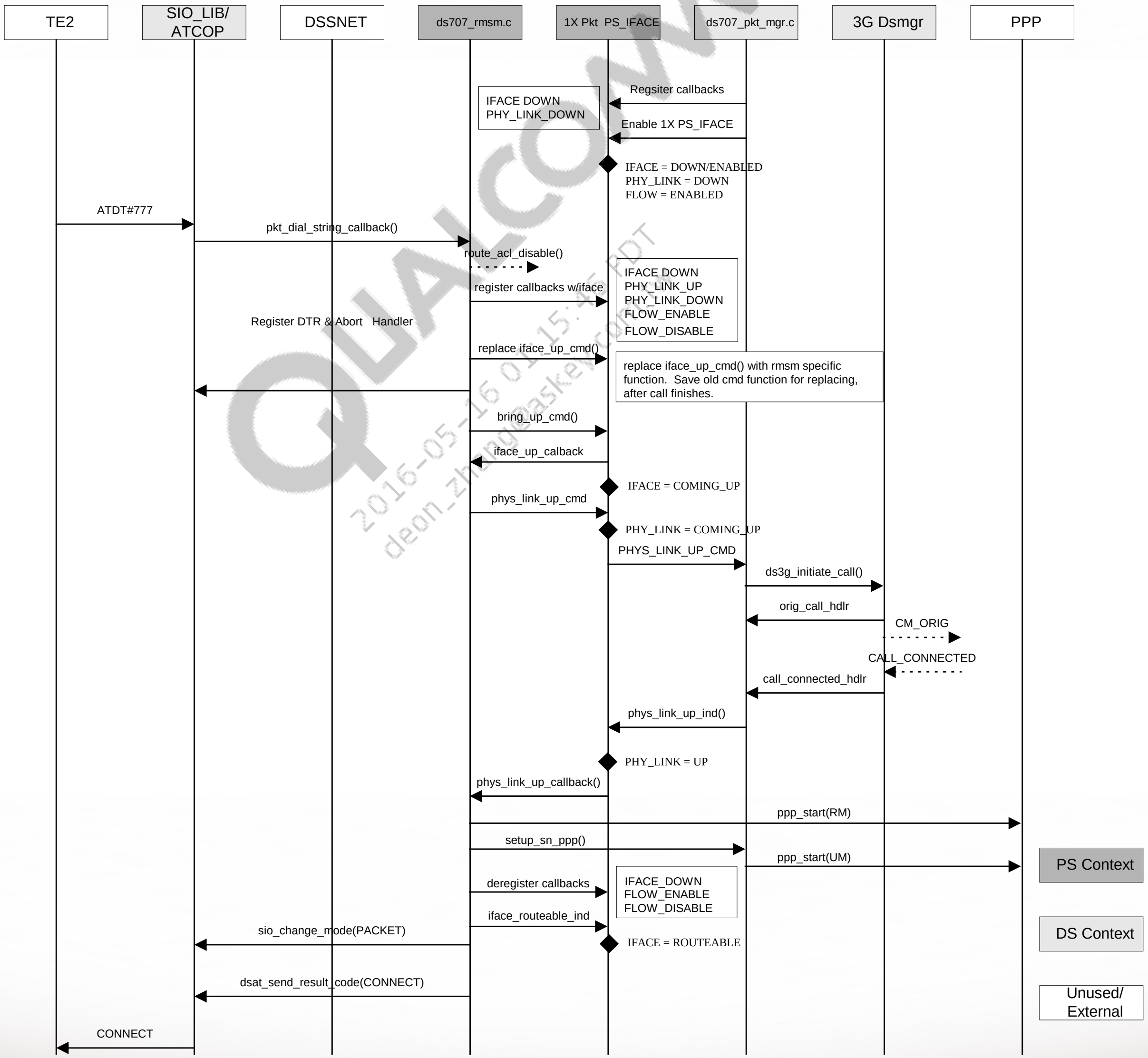
dsat_send_result(CONNECT)

transmit (CONNECT)

re-arm AUTODETECT (fails)

1X Packet Origination (TE) – PS Interface Detailed

1X Packet Origination (TE) – PS IFACE Detailed



TE2
SIO_LIB/ ATCOP
DSSNET
ds707_rmsm.c
1X Pkt PS_IFACE
ds707_pkt_mgr.c
3G Dsmgr
PPP

IFACE DOWN
PHY_LINK_DOWN

Regsiter callbacks
Enable 1X PS_IFACE

IFACE = DOWN/ENABLED
PHY_LINK = DOWN
FLOW = ENABLED

ATDT#777
pkt_dial_string_callback()
route_acl_disable()
register callbacks w/iface

IFACE DOWN
PHY_LINK_UP
PHY_LINK_DOWN
FLOW_ENABLE
FLOW_DISABLE

Register DTR & Abort Handler
replace iface_up_cmd()

replace iface_up_cmd() with rmsm specific function. Save old cmd function for replacing, after call finishes.

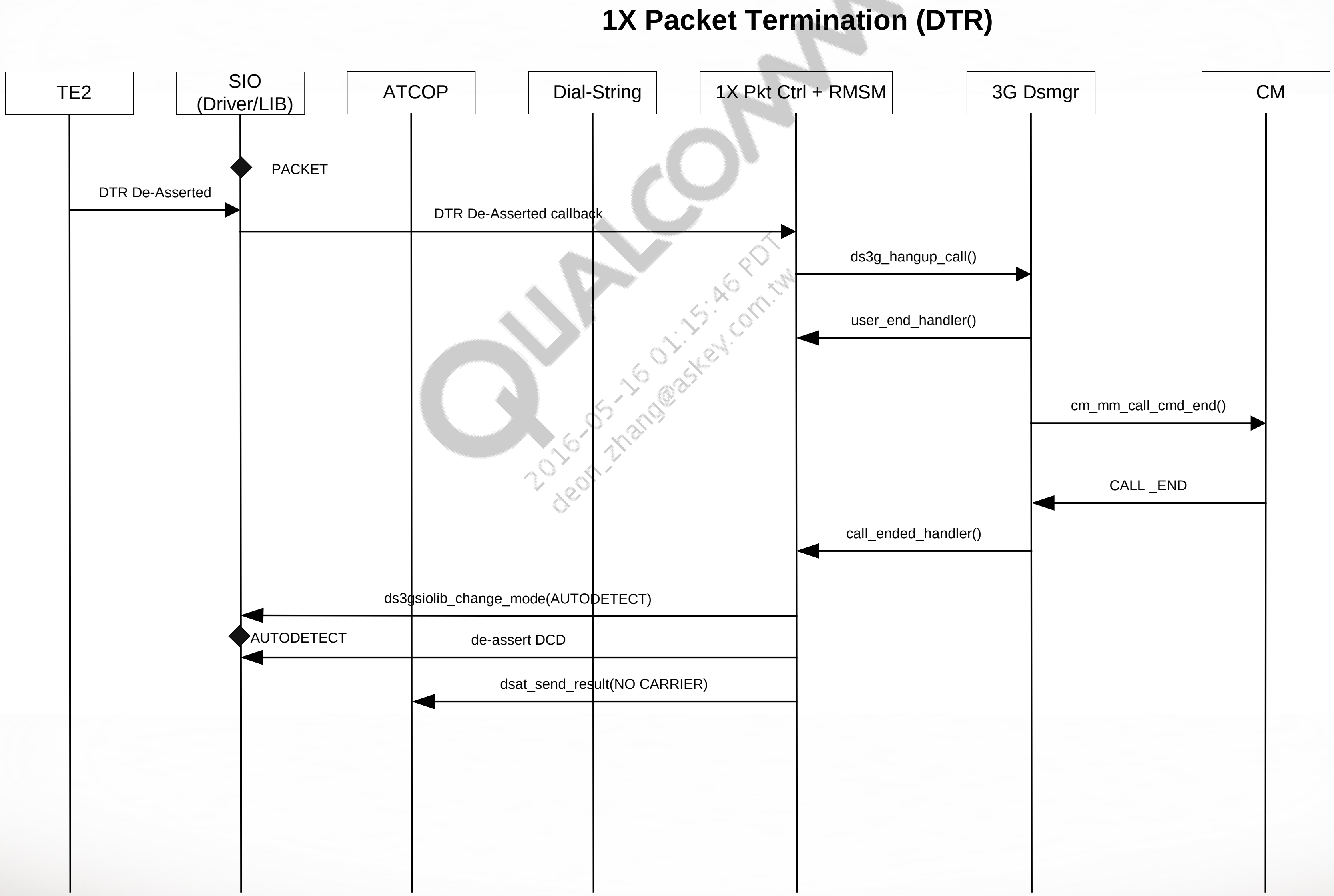
bring_up_cmd()
iface_up_calback
IFACE = COMING_UP
phys_link_up_cmd
PHY_LINK = COMING_UP
PHYS_LINK_UP_CMD
ds3g_initiate_call()
orig_call_hdlr
CM_ORIG
CALL_CONNECTED
call_connected_hdlr
phys_link_up_ind()
PHY_LINK = UP
phys_link_up_callback()
ppp_start(RM)
setup_sn_ppp()
ppp_start(UM)
deregister callbacks

IFACE_DOWN
FLOW_ENABLE
FLOW_DISABLE

iface_routeable_ind
sio_change_mode(PACKET)
IFACE = ROUTEABLE
dsat_send_result_code(CONNECT)
CONNECT

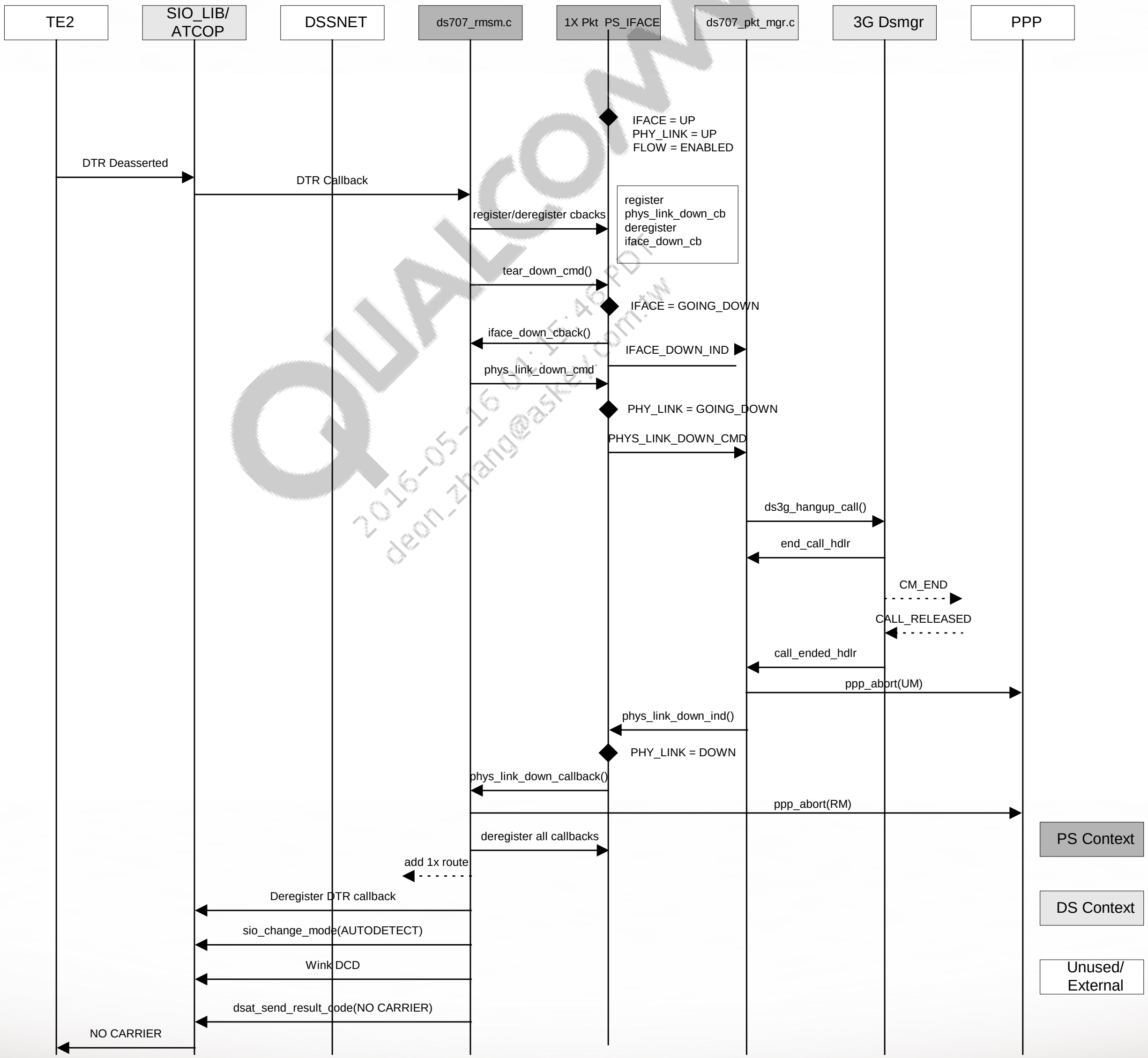
PS Context
DS Context
Unused/ External

1X Packet Termination



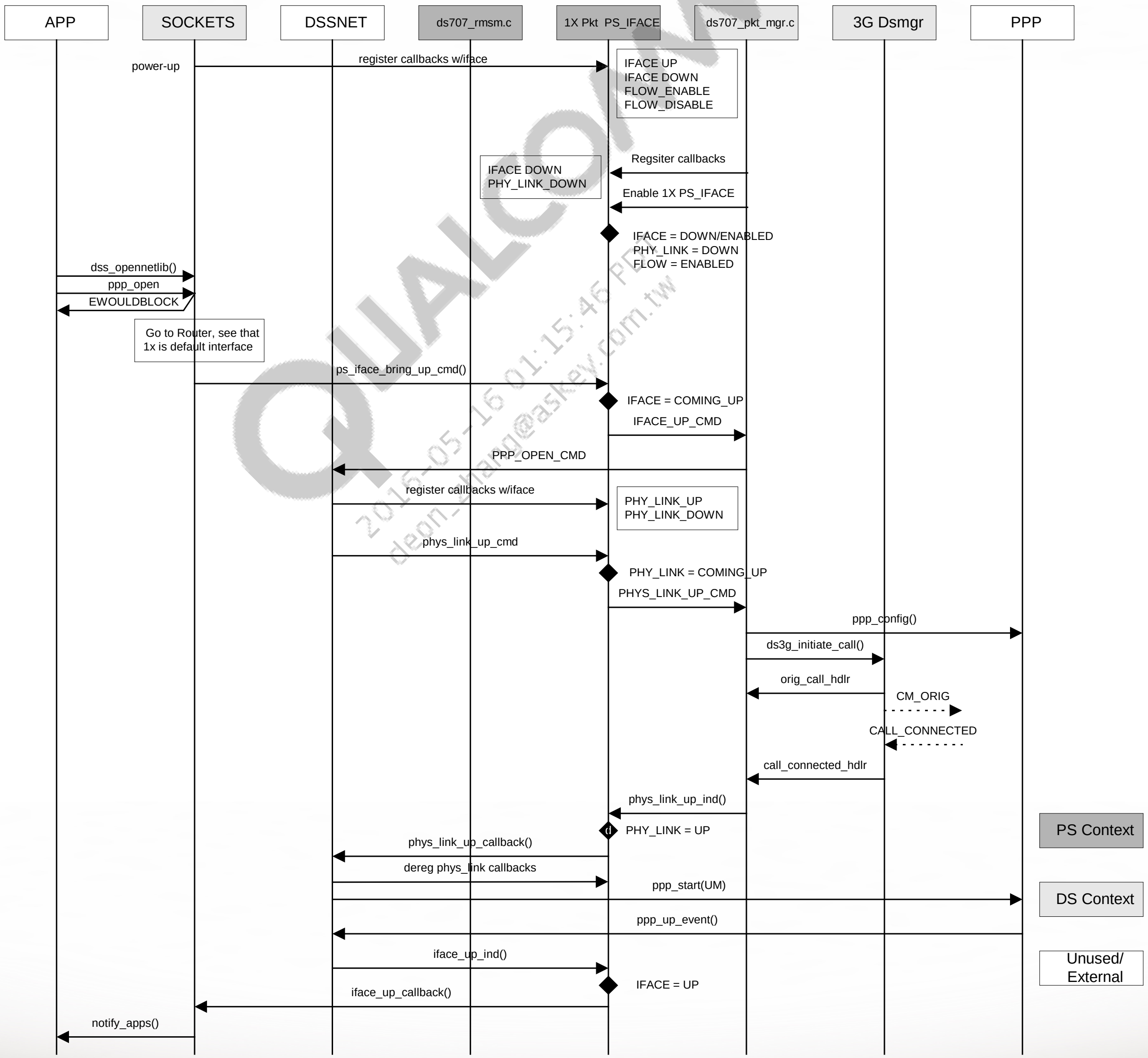
1X Packet Termination (TE) – PS Interface Detailed

1X Packet Termination (TE) – PS IFACE Detailed



1X Packet Sockets Call Origination – PS Interface Detailed

1X Packet Sockets Call Origination – PS IFACE Detailed



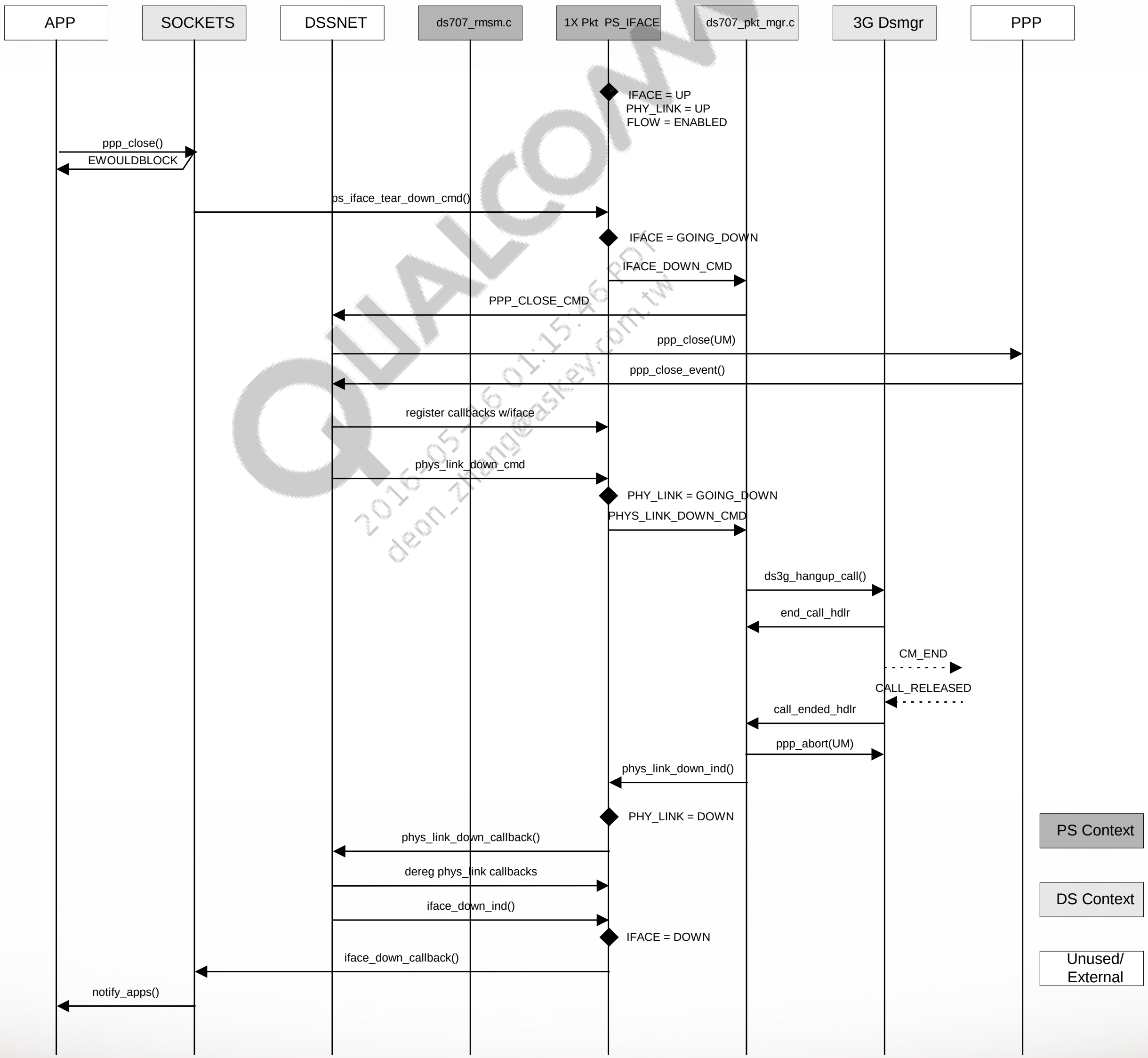
APP
SOCKETS
DSSNET
ds707_rmsm.c
1X Pkt PS_IFACE
ds707_pkt_mgr.c
3G Dsmgr
PPP

power-up
register callbacks w/iface
IFACE UP
IFACE DOWN
FLOW_ENABLE
FLOW_DISABLE
Regsiter callbacks
IFACE DOWN
PHY_LINK_DOWN
Enable 1X PS_IFACE
IFACE = DOWN/ENABLED
PHY_LINK = DOWN
FLOW = ENABLED
dss_opennetlib()
ppp_open
EWOULDBLOCK
Go to Router, see that
1x is default interface
ps_iface_bring_up_cmd()
IFACE = COMING_UP
IFACE_UP_CMD
PPP_OPEN_CMD
register callbacks w/iface
PHY_LINK_UP
PHY_LINK_DOWN
phys_link_up_cmd
PHY_LINK = COMING_UP
PHYS_LINK_UP_CMD
ppp_config()
ds3g_initiate_call()
orig_call_hdlr
CM_ORIG
CALL_CONNECTED
call_connected_hdlr
phys_link_up_ind()
PHY_LINK = UP
phys_link_up_callback()
dereg phys_link callbacks
ppp_start(UM)
ppp_up_event()
iface_up_ind()
IFACE = UP
iface_up_callback()
notify_apps()

PS Context
DS Context
Unused/
External

1X Packet Sockets Call Release – PS Interface Detailed

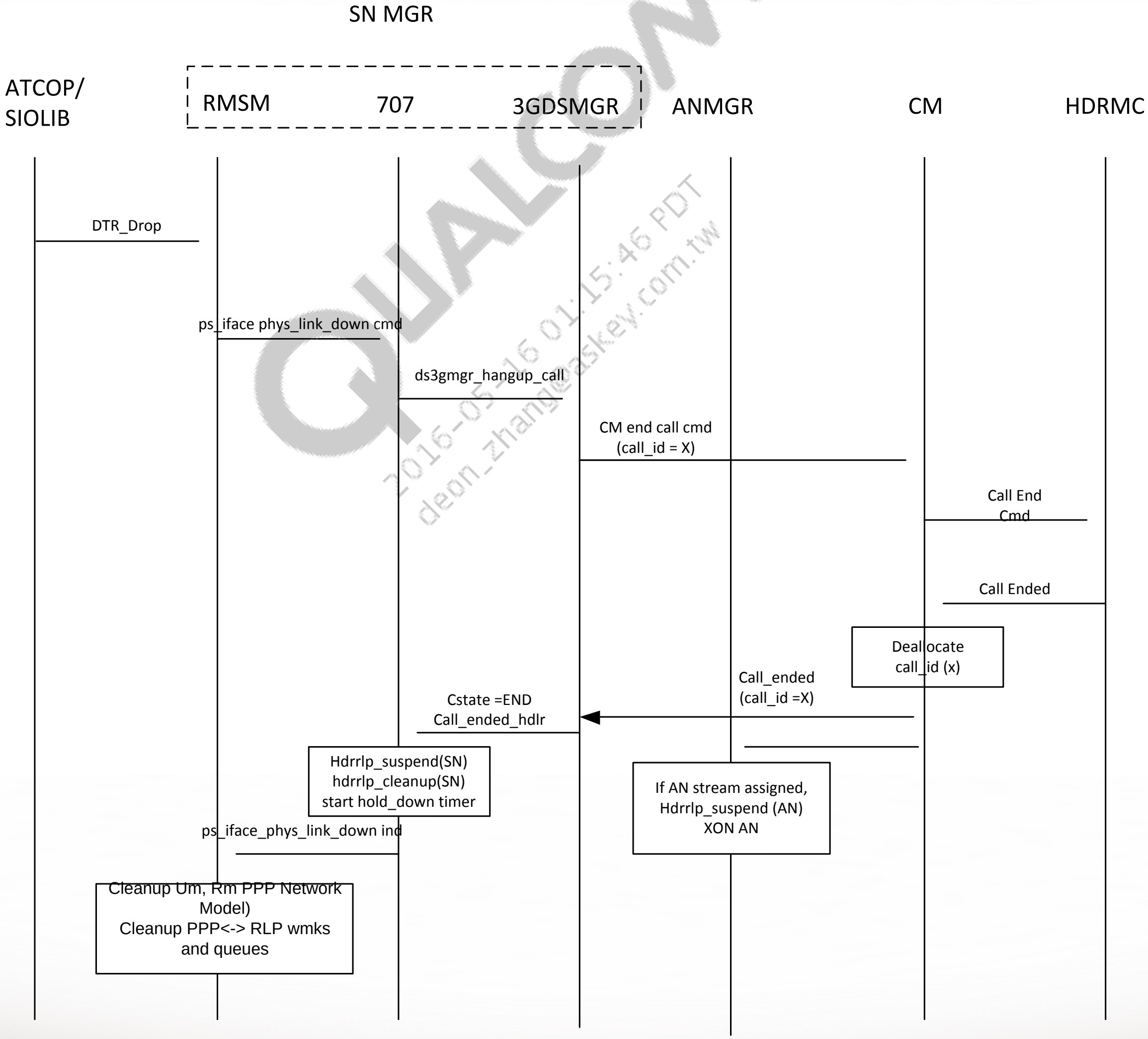
1X Packet Sockets Call Release – PS IFACE Detailed



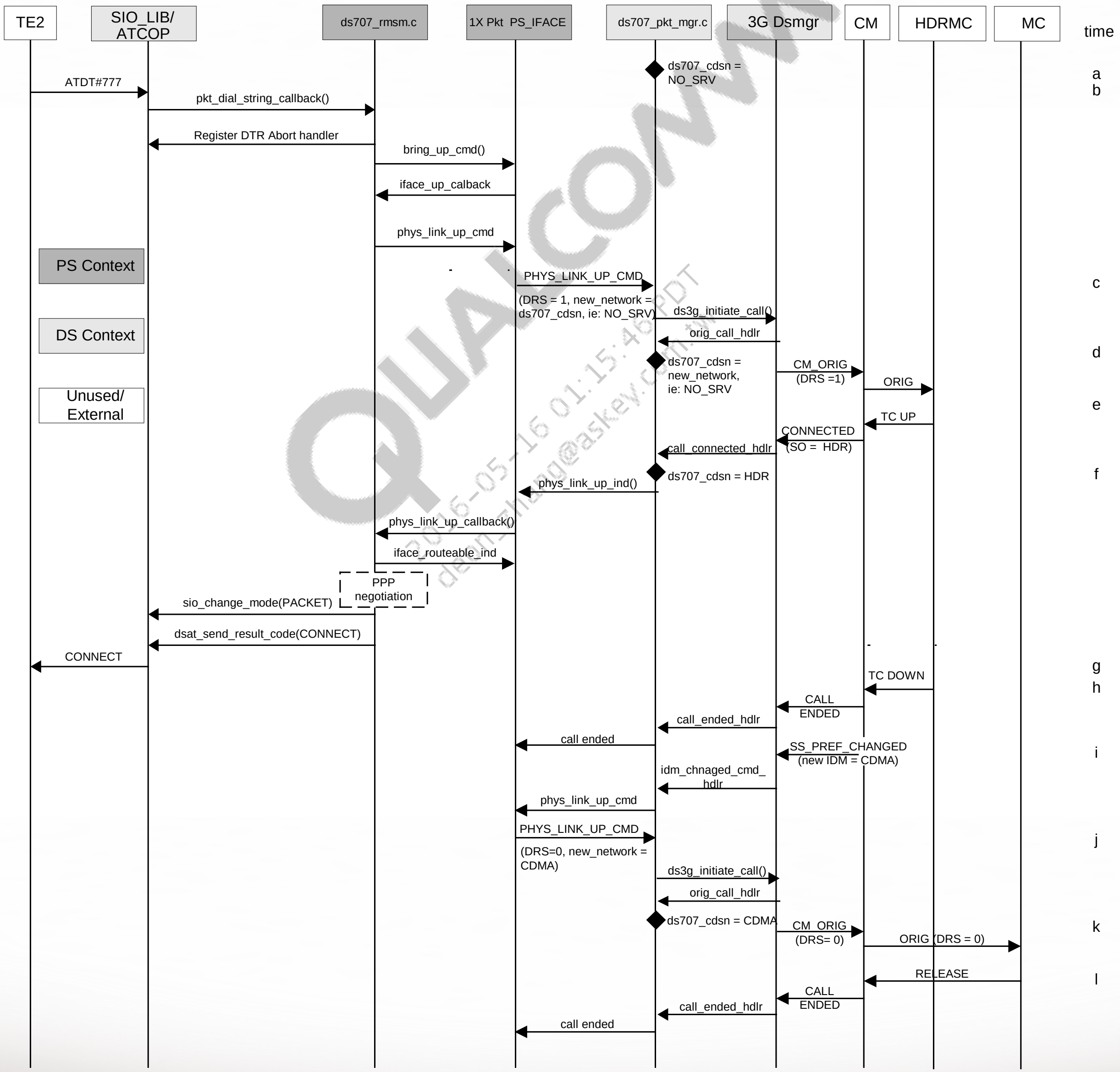
Appendix B – HDR Call Flows

HDR Packet Termination (TE)

LAPTOP CALL TERMINATION



HDR→CDMA Dormant Handoff



HDR→CDMA Dormant Notes

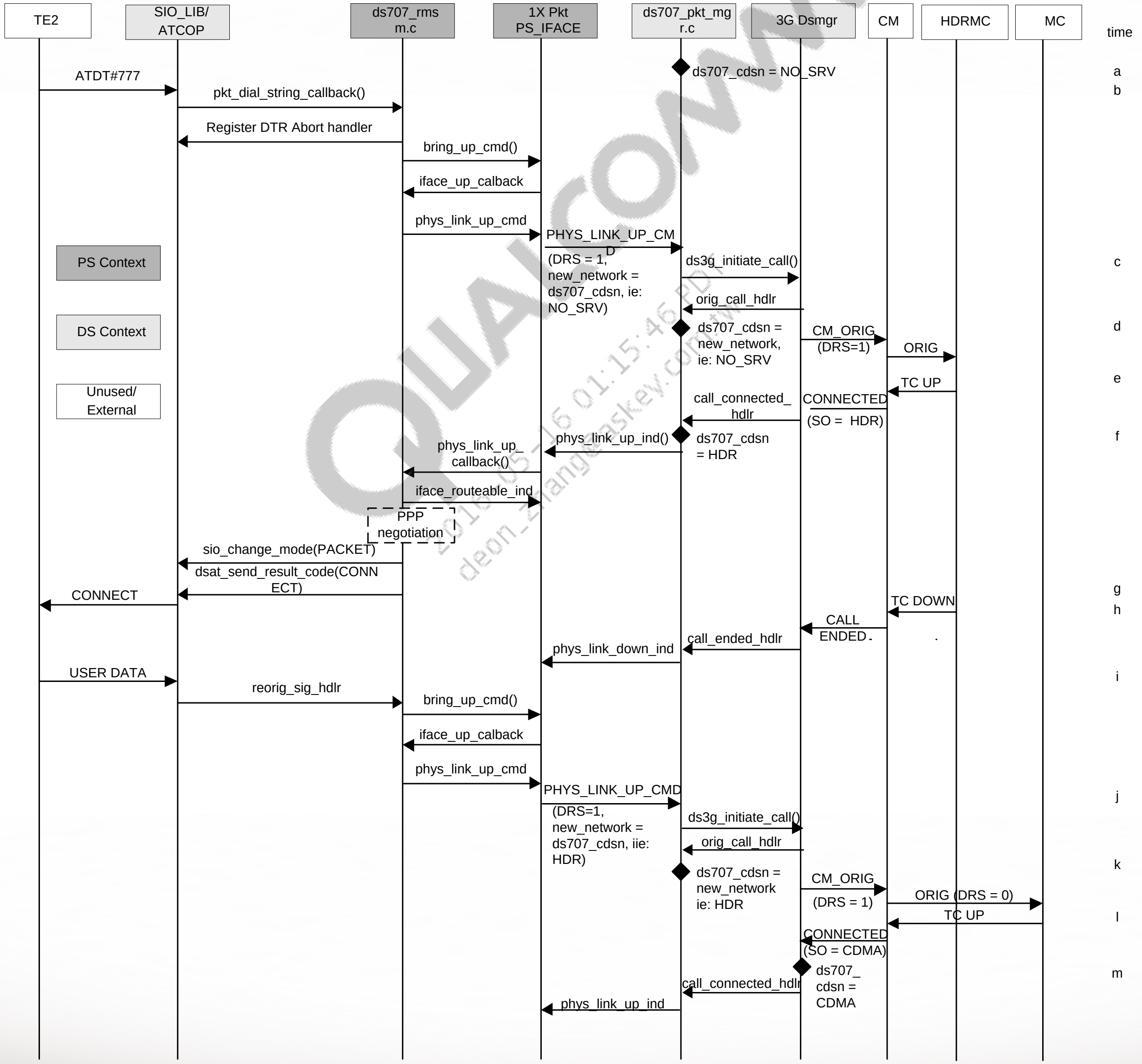
- Initially, the current data session network is NO_SRV.
- The user originates a packet data call.
- The Phys Link Up command is called with DRS = 1, i.e., the MS has user data to send and the new network, passed in the command, is the current data session network, which happens to be NO_SRV.
- Immediately after populating the parameters for the origination command to CM, the current data session network is updated to the new network, which is NO_SRV.
- CM and lower layers bring up the call on HDR.
- Upon call connection, the current data session is updated to HDR, based on the SO passed in the Call Connected command.
- After PPP negotiation, the laptop is informed that the call has been connected.
- Sometime after the call is connected, the HDR signal seen by the MS fades and the packet data call goes dormant.

HDR-CDMA Dormant Notes (cont.)

- Once the lower layers have determined that HDR is lost and CDMA is available, CM sends an S_PREF_CHANGED command to DS with the new Idle Digital mode set to CDMA.
- In response to Idle Digital mode command, DS initiates an origination with DRS = 0, i.e., it is an origination purely to notify the CDMA base station to move the PPP session to CDMA and that there is no real user data to send. Since this origination is the result of Idle Digital mode changing to CDMA, the new network is explicitly specified as CDMA in the Phys Link Up command.
- Immediately after populating the parameters for the origination command to CM, the current data session network is updated to the new network, which is CDMA.
- Since this was an origination with DRS = 0, the base station releases the call and does not set up the traffic channel.

Note: The call flow for a dormant CDMA→HDR data session is similar. The Idle Digital Mode Changed command will be given by CM in the case when HDR is acquired during dormancy.

HDR→CDMA Inter-System Call Setup



HDR→CDMA Active Notes

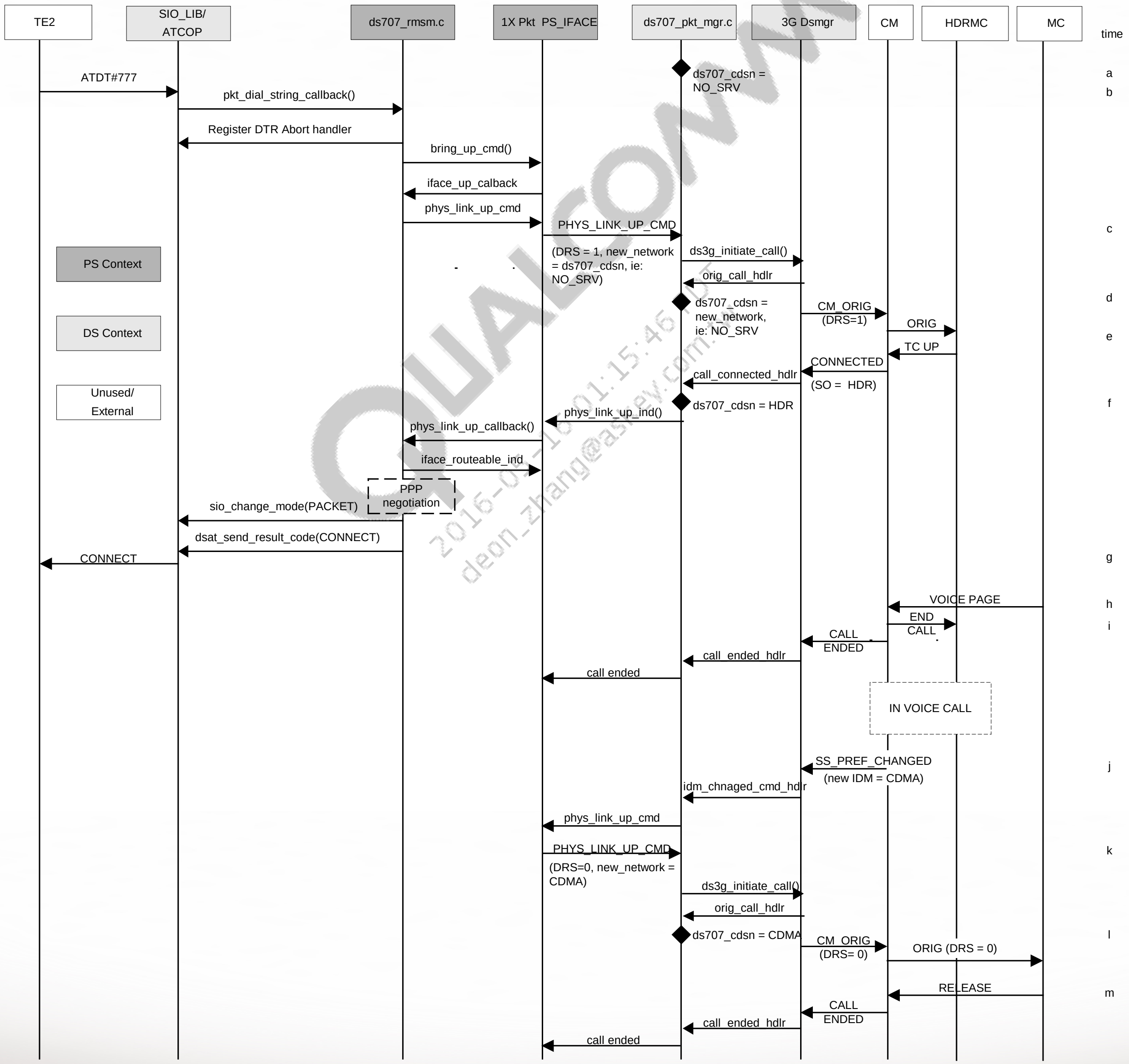
- Initially, the current data session network is NO_SRV.
- The user originates a packet data call.
- The Phys Link Up command is called with DRS = 1, i.e., the MS has user data to send and the new network, passed in the command, is the current data session network, which happens to be NO_SRV.
- Immediately after populating parameters for the origination command to CM, the current data session network is updated to the new network, which is NO_SRV.
- CM and lower layers bring up the call on HDR.
- Upon call connection, the current data session is updated to HDR, based on the SO passed in the Call Connected command.
- After PPP negotiation, the laptop is informed that the call has been connected.
- Sometime after the call is connected, the HDR signal seen by the MS fades and the packet data call goes dormant.

HDR→CDMA Active Notes (cont.)

- The user has data to send and the DS reoriginates the call from dormancy.
- The Phys Link Up command is called with DRS = 1, i.e., the MS has user data to send and the new network passed in the command is the current data session network, which happens to be HDR.
- Immediately after populating the parameters for the origination command to CM, the current data session network is updated to the new network, which is HDR.
- CM and lower layers bring up the call on CDMA.
- On getting a call connected, the current data session is updated to CDMA, based on the SO passed in the Call Connected command.

Note: The callflow for CDMA→HDR intersystem call setup is similar; this will happen if HDR is just acquired when the user has data to send.

Tune-Away to CDMA for a Voice Call and Stay In CDMA



TE2
SIO_LIB/ ATCOP
ds707_rmsm.c
1X Pkt PS_IFACE
ds707_pkt_mgr.c
3G Dsmgr
CM
HDRMC
MC
time

PS Context
DS Context
Unused/ External

a ds707_cdsn = NO_SRV
b ATDT#777
pkt_dial_string_callback()
Register DTR Abort handler
bring_up_cmd()
iface_up_calback
phys_link_up_cmd
c PHYS_LINK_UP_CMD
(DRS = 1, new_network = ds707_cdsn, ie: NO_SRV)
ds3g_initiate_call()
orig_call_hdlr
d ds707_cdsn = new_network, ie: NO_SRV
CM_ORIG (DRS=1)
e ORIG
TC UP
CONNECTED (SO = HDR)
call_connected_hdlr
f ds707_cdsn = HDR
phys_link_up_ind()
phys_link_up_callback()
iface_routeable_ind
PPP negotiation
sio_change_mode(PACKET)
dsat_send_result_code(CONNECT)
g CONNECT
h VOICE PAGE
i END CALL
CALL ENDED
call_ended_hdlr
call ended
IN VOICE CALL
j SS_PREF_CHANGED (new IDM = CDMA)
idm_chnaged_cmd_hdlr
phys_link_up_cmd
k PHYS_LINK_UP_CMD
(DRS=0, new_network = CDMA)
ds3g_initiate_call()
orig_call_hdlr
l ds707_cdsn = CDMA
CM_ORIG (DRS= 0)
ORIG (DRS = 0)
m RELEASE
CALL ENDED
call_ended_hdlr
call ended

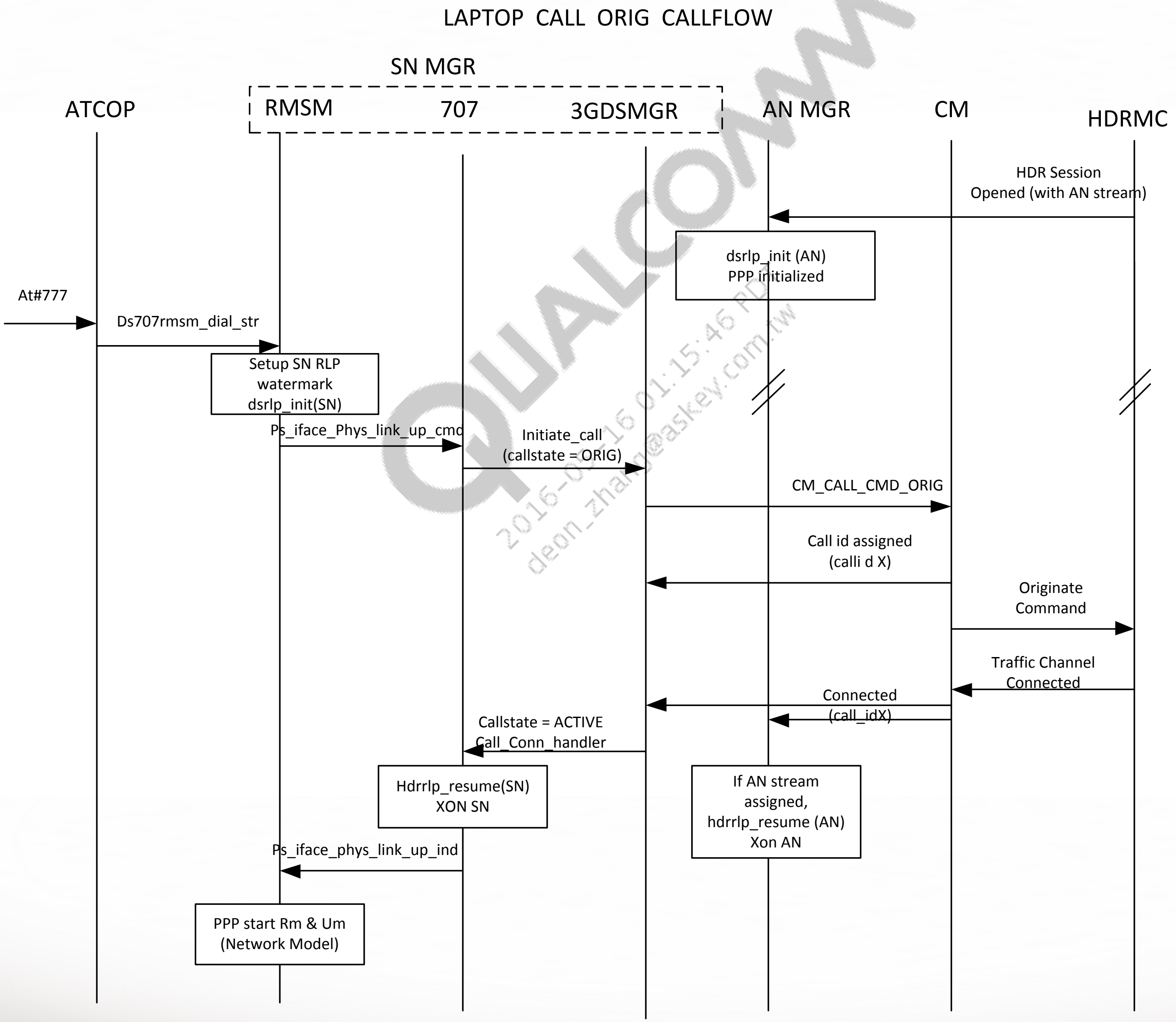
CDMA Voice Tune-Away Notes

- Initially, the current data session network is NO_SRV.
- The user originates a packet data call.
- The Phys Link Up command is called with DRS = 1, i.e., the MS has user data to send and the new network, passed in the command, is the current data session network, which happens to be NO_SRV.
- Immediately after populating parameters for the origination command to CM, the current data session network is updated to the new network, which is NO_SRV.
- CM and lower layers bring up the call on HDR.
- Upon call connection, the current data session is updated to HDR, based on the SO passed in the Call Connected command.
- After PPP negotiation, the laptop is informed that the call has been connected.
- During HDR traffic, the MS tunes away to CDMA and receives a voice page.
- The CM sends a call end to HDRMC and DS for the data call, and accepts the voice call; the data call goes dormant.

CDMA Voice Tune-Away Notes (cont.)

- After the voice call ends, the lower layers try to acquire HDR and fail. Once the lower layers have determined that HDR is lost and CDMA is available, the CM sends a SS_PREF_CHANGED command to DS with the new Idle Digital mode set to CDMA.
- In response to the Idle Digital Mode command, DS initiates an origination with DRS = 0, i.e., it is an origination purely to notify the CDMA base station to move the PPP session to CDMA and that there is no real user data to send. Since this origination is the result of Idle Digital mode changing to CDMA, the new network is explicitly specified as CDMA in the Phys Link Up command.
- Immediately after populating the parameters for the origination command to CM, the current data session network is updated to the new network, which is CDMA.
- Since this was an origination with DRS = 0, the base station releases the call and does not set up the traffic channel.

HDR Packet Origination (TE)



LAPTOP CALL ORIG CALLFLOW
SN MGR
ATCOP
RMSM
707
3GDSMGR
AN MGR
CM
HDRMC
HDR Session Opened (with AN stream)
dsrlp_init (AN) PPP initialized
At#777
Ds707rmsm_dial_str
Setup SN RLP watermark dsrlp_init(SN)
Ps_iface_Phys_link_up_cmd
Initiate_call (callstate = ORIG)
CM_CALL_CMD_ORIG
Call id assigned (calli d X)
Originate Command
Traffic Channel Connected
Connected (call_idX)
Callstate = ACTIVE Call_Conn_handler
Hdrrlp_resume(SN) XON SN
If AN stream assigned, hdrrlp_resume (AN) Xon AN
Ps_iface_phys_link_up_ind
PPP start Rm & Um (Network Model)

Multimode Data Key Feature Support on Various Air Interfaces

Software	Version	Multimode data component	Feature support effective date
DMSS 6100	3.0	1X, MIP	Mar 2003
AMSS 6200	2.0	UMTS CS data	Dec 2002
	2.0	UMTS PS data	Dec 2002
	2.0	GSM CS data	Dec 2002
AMSS 6300	2.0	1X, MIP	Mar 2003
	3.0	GSM/GPRS	Jun 2003
AMSS 6500	1.0	1X, 1xEV-DO	Jun 2003
	3.0	GSM/GPRS	Jun 2004

References

Ref.	Document	
Qualcomm Technologies		
Q1	*Application Note: Software Glossary for Customers*	CL93-V3077-1
Q2	*Data Services API Interface Specification and Operational Description (ISOD)*	80-V6415-1
Q3	*Presentation: Data Services Software Components*	80-VE662-1
Q4	*Presentation: IS-856 Data Services Overview*	80-V3436-1
Standards		
S1	*General on Terminal Adaptation Functions (TAF) for Mobile Stations (MS)*	3GPP TS 27.001 v3.7.0
S2	*Codec(s) for Circuit-Switched (CS) Multimedia Telephony Service; Terminal Implementor's Guide*	3GPP TR 26.911
S3	*Common Test Environments for User Equipment (UE); Conformance Testing*	3GPP TS 34.108 v3.2.0
S4	*Rate Adaption on the Mobile Station - Base Station System (MS-BSS) Interface*	3GPP TS 04.21
S5	*Radio Link Protocol (RLP) for Circuit-Switched Bearer and Teleservices*	3GPP TS 24.022
S6	*Terminal Adaptation Functions (TAF) for Services Using Asynchronous Bearer Capabilities*	3GPP TS 27.002
S7	*Data Service Options for Wideband Spread Spectrum Systems*	TIA/EIA/IS-707-A
S8	*Data Transmission Systems and Equipment In-Band DCE Control*	TIA/EIA-617
S9	*Asynchronous Facsimile DCE Control Standard*	TIA/EIA-592
S10	*Facsimile DCE-DTE Packet Protocol Standard*	TIA/EIA-605
S11	*T.31 : Asynchronous facsimile DCE control - Service Class 1*	ITU-T T.31
S12	*3GPP Specification Change Request records for: 04.35*	GSM 04.35

Questions?

https://support.cdmatech.com