ACKNOWLEDGEMENT

By utilizing this website and/or documentation, I hereby acknowledge as follows:

Effective October 1, 2012, QUALCOMM Incorporated completed a corporate reorganization in which the assets of certain of its businesses and groups, as well as the stock of certain of its direct and indirect subsidiaries, were contributed to Qualcomm Technologies, Inc. (QTI), a wholly-owned subsidiary of QUALCOMM Incorporated that was created for purposes of the reorganization.

Qualcomm Technology Licensing (QTL), the Company's patent licensing business, continues to be operated by QUALCOMM Incorporated, which continues to own the vast majority of the Company's patent portfolio. Substantially all of the Company's products and services businesses, including QCT, as well as substantially all of the Company's engineering, research and development functions, are now operated by QTI and its direct and indirect subsidiaries[1]. Neither QTI nor any of its subsidiaries has any right, power or authority to grant any licenses or other rights under or to any patents owned by QUALCOMM Incorporated.

No use of this website and/or documentation, including but not limited to the downloading of any software, programs, manuals or other materials of any kind or nature whatsoever, and no purchase or use of any products or services, grants any licenses or other rights, of any kind or nature whatsoever, under or to any patents owned by QUALCOMM Incorporated or any of its subsidiaries. A separate patent license or other similar patent-related agreement from QUALCOMM Incorporated is needed to make, have made, use, sell, import and dispose of any products or services that would infringe any patent owned by QUALCOMM Incorporated in the absence of the grant by QUALCOMM Incorporated of a patent license or other applicable rights under such patent.

Any copyright notice referencing QUALCOMM Incorporated, Qualcomm Incorporated, QUALCOMM Inc., Qualcomm Inc., Qualcomm or similar designation, and which is associated with any of the products or services businesses or the engineering, research or development groups which are now operated by QTI and its direct and indirect subsidiaries, should properly reference, and shall be read to reference, QTI.

[1] The products and services businesses, and the engineering, research and development groups, which are now operated by QTI and its subsidiaries include, but are not limited to, QCT, Qualcomm Mobile & Computing (QMC), Qualcomm Atheros (QCA), Qualcomm Internet Services (QIS), Qualcomm Government Technologies (QGOV), Corporate Research & Development, Qualcomm Corporate Engineering Services (QCES), Office of the Chief Technology Officer (OCTO), Office of the Chief Scientist (OCS), Corporate Technical Advisory Group, Global Market Development (GMD), Global Business Operations (GBO), Qualcomm Ventures, Qualcomm Life (QLife), Quest, Qualcomm Labs (QLabs), Snaptracs/QCS, Firethorn, Qualcomm MEMS Technologies (QMT), Pixtronix, Qualcomm Innovation Center (QuIC), Qualcomm iSkoot, Qualcomm Poole and Xiam.

QUALCOMM® MSM™ Interface (QMI)

QUALCOMM CDMA Technologies

80-VA615-1 B

QUALCOMM® Proprietary

QUALCOMM Incorporated
5775 Morehouse Drive
San Diego, CA 92121-1714
U.S.A.

Revision History

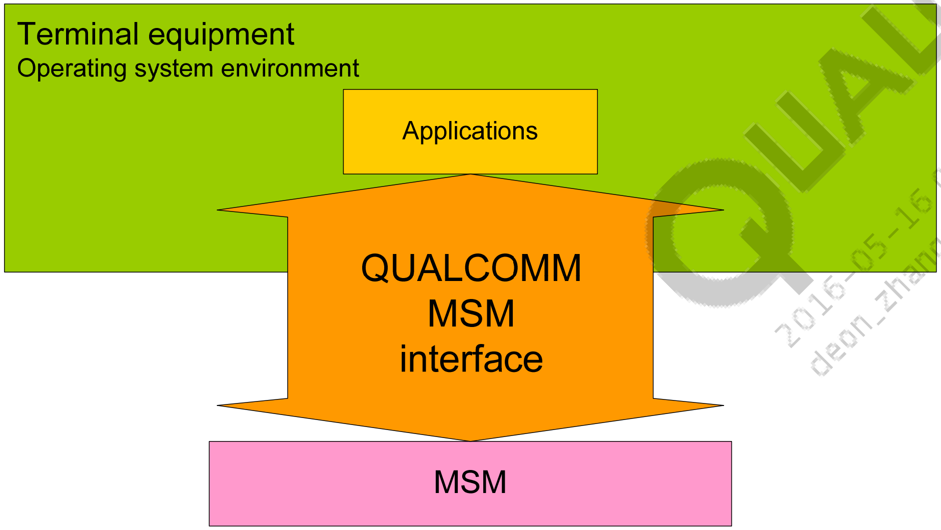
Version	Date	Description
A	Nov 2005	Initial release
B	Apr 2006	Minor clarifications and changed title

Contents

- Introduction
- QMI Architecture
- R_m Network Interface (R_m Net) QMI Logical Device
- Questions?

Introduction

QMI Overview



- QMI
  - Defines the interface between MSM™ and attached Terminal Equipment (TE)
  - A framework to export multitude of services from MSM to TE including:
    - » Interconnection guidelines
    - » Logical device enumeration
    - » QMI service protocol message definitions
  - Fully extensible, scalable, and portable architecture

    Note: R_m network interface is a new logical device in the QMI framework

QMI Design Goals

- Hardware form factor independent
  - Supports any TE form factor
    - » Example – PC, notebook, laptop, PDA, SmartPhone, etc.
  - Supports any MSM form factor
    - » Example – Embedded module, removable data card, tethered phone, etc.
- Interconnect independent framework
  - Portable to various MSM-TE interconnects
    - » Example – USB, Bluetooth® (BT), serial bus, shared memory, IPC, etc.
- Operating system independent
  - Extensible to any OS environment on TE

QMI Usage Scenarios

- PC users
  - TE
    - Laptops/desktops running third-party OS
      - Example – Windows® XP, Longhorn, Linux, MacOS
  - MSM
    - Phone tethered via USB
    - Data card via USB (PCI express)
    - Phone connected via BT
- PDAs and SmartPhone
  - Various combinations of TE and MSM
    - Standalone MSM with external applications processor (TE)
    - Standalone MSM with no applications processor (tethered data service)
    - Combined MSM and applications processor
      - Apps processor running AMSS or third-party OS

QMI Architecture

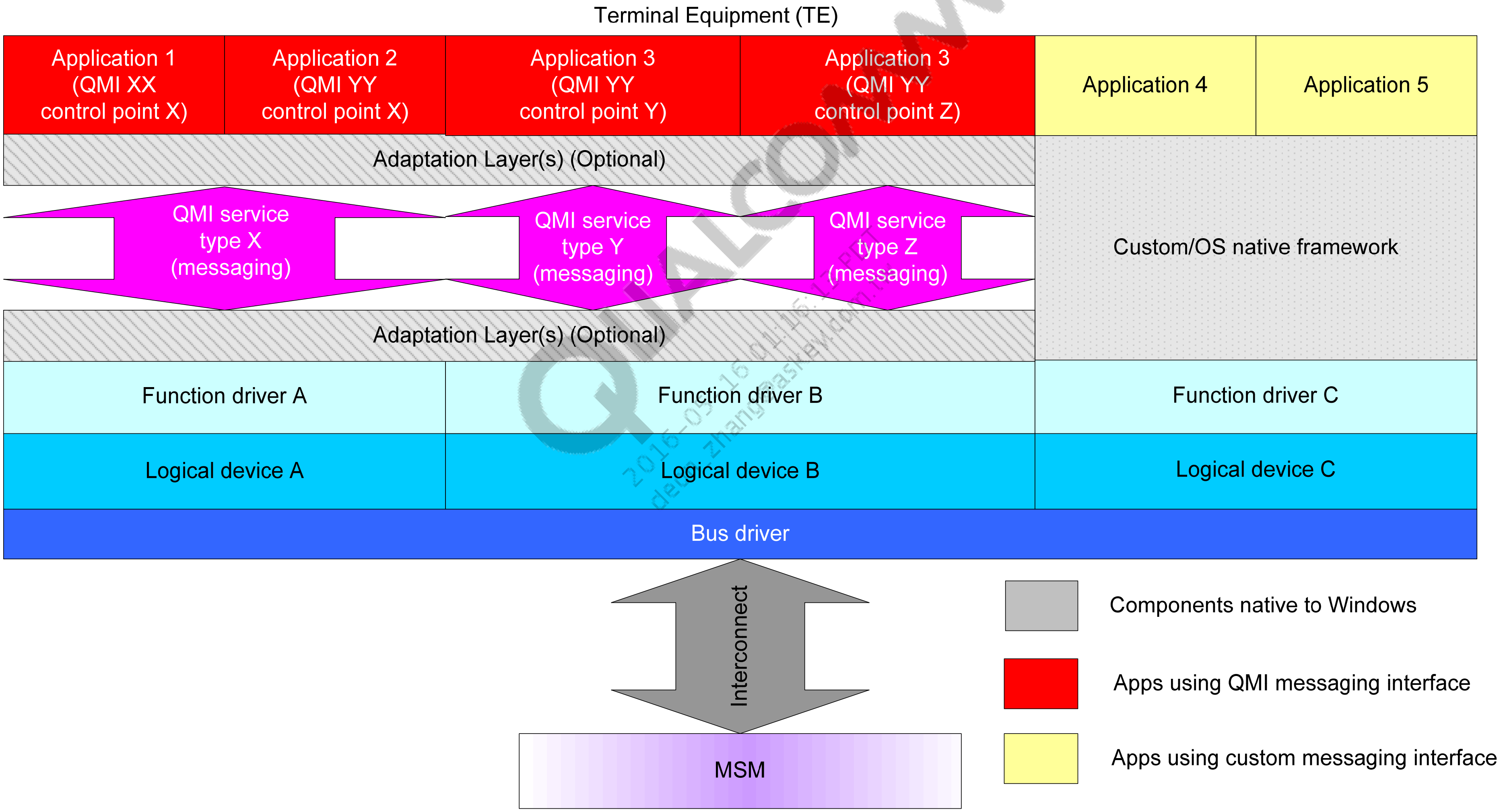
QMI Architecture

- QMI architecture defines three components
  - QMI device connectivity framework
    - MSM-TE connectivity and logical device configuration
  - QMI service model
    - Client server architecture
  - QMI control messaging interface
    - Communication endpoints and protocol stack

QMI Interconnect Requirements

- Supports one or more logical devices
  - Each has control and/or data channel
- Control channel
  - Bidirectional
  - Reliable
  - Optional for custom logical devices
  - Provides packet framing function
- Data channel
  - Bidirectional
  - Best effort
  - Provides packet framing function

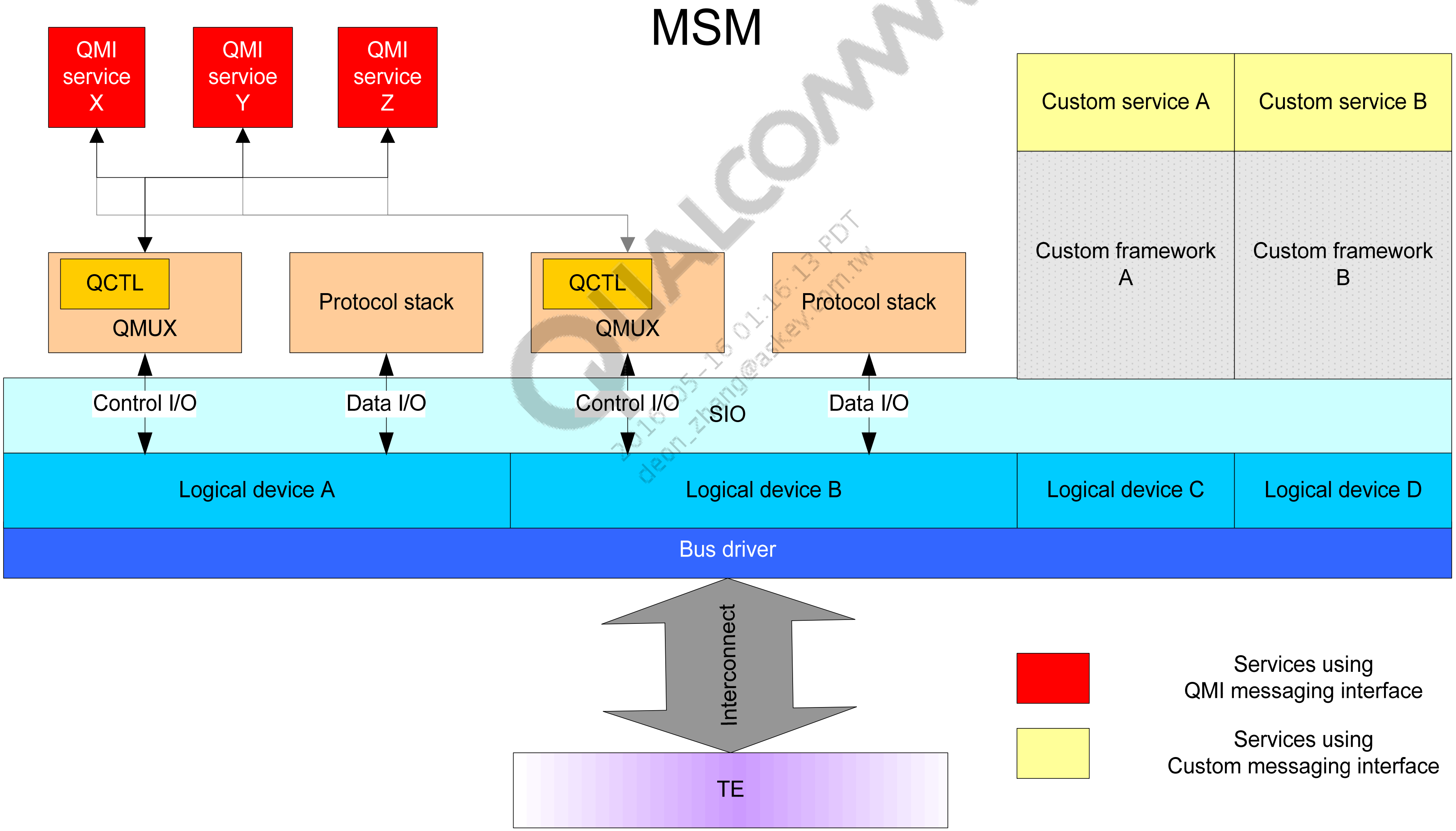
QMI Connectivity Framework – TE


Terminal Equipment (TE)
Application 1 (QMI XX control point X)
Application 2 (QMI YY control point X)
Application 3 (QMI YY control point Y)
Application 3 (QMI YY control point Z)
Application 4
Application 5
Adaptation Layer(s) (Optional)
QMI service type X (messaging)
QMI service type Y (messaging)
QMI service type Z (messaging)
Custom/OS native framework
Adaptation Layer(s) (Optional)
Function driver A
Function driver B
Function driver C
Logical device A
Logical device B
Logical device C
Bus driver
Interconnect
MSM
Components native to Windows
Apps using QMI messaging interface
Apps using custom messaging interface

QMI Framework Components

- Bus driver
  - Driver for the bus controller
  - Mostly provided natively by OS vendor
- Logical devices
  - Exposed through bus enumeration
    - » With plug-and-play, logical device enumeration may be dynamic
    - » Without plug-and-play, device enumeration is statically defined
  - QMI framework defines logical devices that are exposed
- Function drivers
  - Exports specific functionality to applications
  - Upper layer exposes OS native and/or custom API to applications
  - Lower layer interfaces to MSM via QMI messaging
    - » Uses one or more QMI services
    - » May expose zero or more QMI service APIs directly to applications

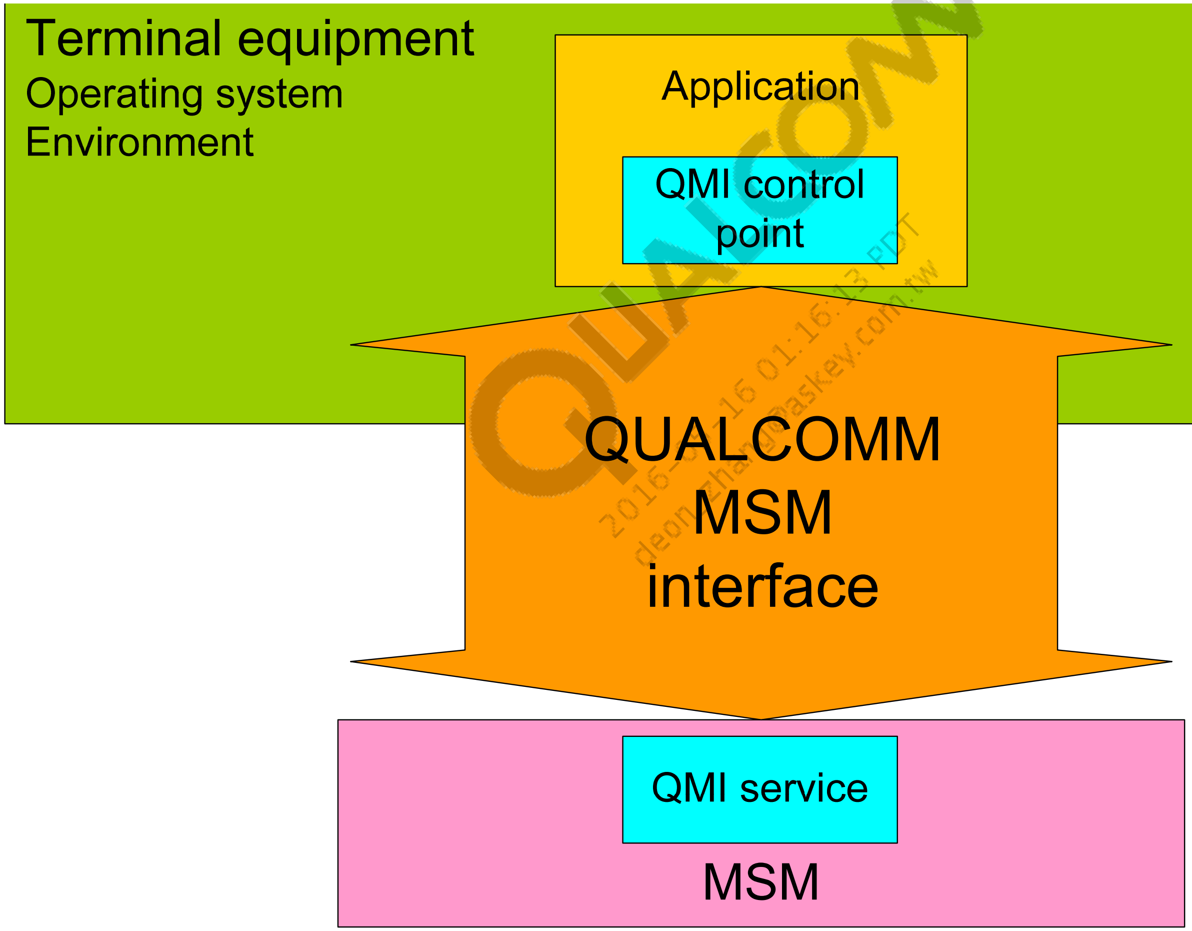
QMI Connectivity Framework – MSM



🟥 Services using QMI messaging interface

🟨 Services using Custom messaging interface

QMI Service Model



QMI Service Model (cont.)

- QMI supports two types of services
  - QMI (native) service
    - » Uses QMI messaging via control I/O path of one or more logical devices
    - » Addressed by a unique identifier (QMI service type)
    - » Service-specific protocol defined in associated QMI service ISOD
      - QMI vendor-defined services are allowed
  - Custom service
    - » Does not use QMI messaging interface
    - » Uses custom data format on control and/or data channels
    - » Includes legacy services, e.g., MSM Diag
    - » Suitable when data format is already defined, e.g., MSM Diag, SyncML

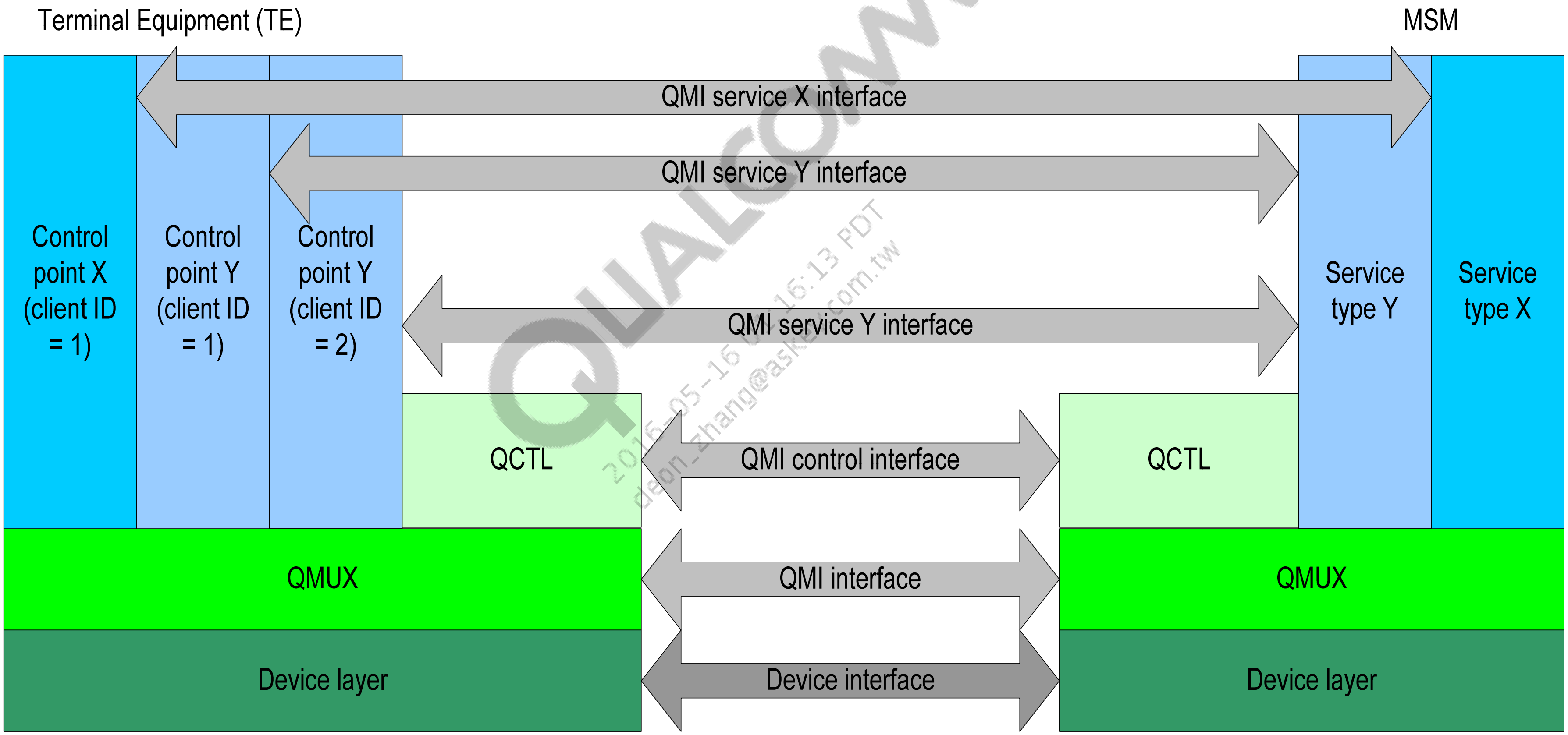
QMI Service Model (cont.)

- QMI messaging defines two communication endpoints
  - Service entity
    - » Exports a specific QMI native service from the MSM
    - » Separate service entities for each defined QMI service
    - » Supports a certain number of control points, e.g., one if service only wants one controller to avoid contention
  - Control point
    - » Customer of a QMI native service
    - » Identified by a 1 byte client_id assigned by service entity
    - » Typically resides in TE as part of a User mode application/module
      - May exist in function driver
    - » Application may own multiple control points
      - Associated with different services

QMI Service Model (cont.)

- QMI services defined today
  - QMI Control Service (QMI_CTL)
  - QMI Wireless Data Service (QMI_WDS)
  - QMI Device Management Service (QMI_DMS)
  - QMI Network Access Service (QMI_NAS)
- Future services
  - QMI Quality of Service (QMI_QOS)
  - QMI Location-Based Services (QMI_LBS)

QMI Messaging Interface



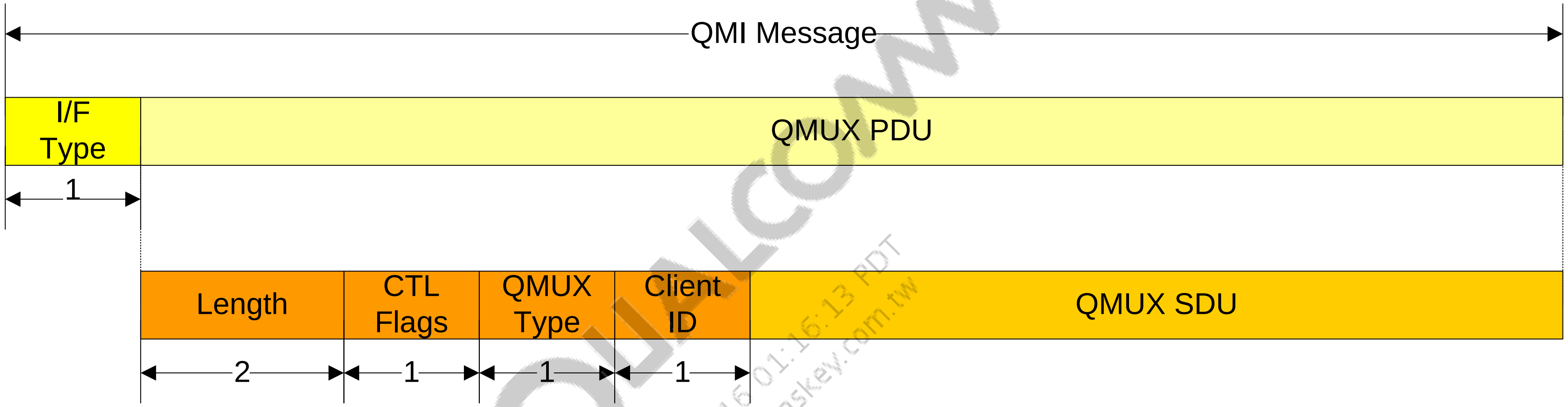
QMI Messaging Interface (cont.)

- QMUX
  - QMI control channel multiplexing and routing layer
  - All QMI messages are transported via this layer
  - Routes QMI messages to appropriate service/control point entities
  - One QMUX instance per logical device
  - For each logical device, allows:
    - » Multiple services to be exported
    - » Multiple control points per service

QMI Messaging Interface (cont.)

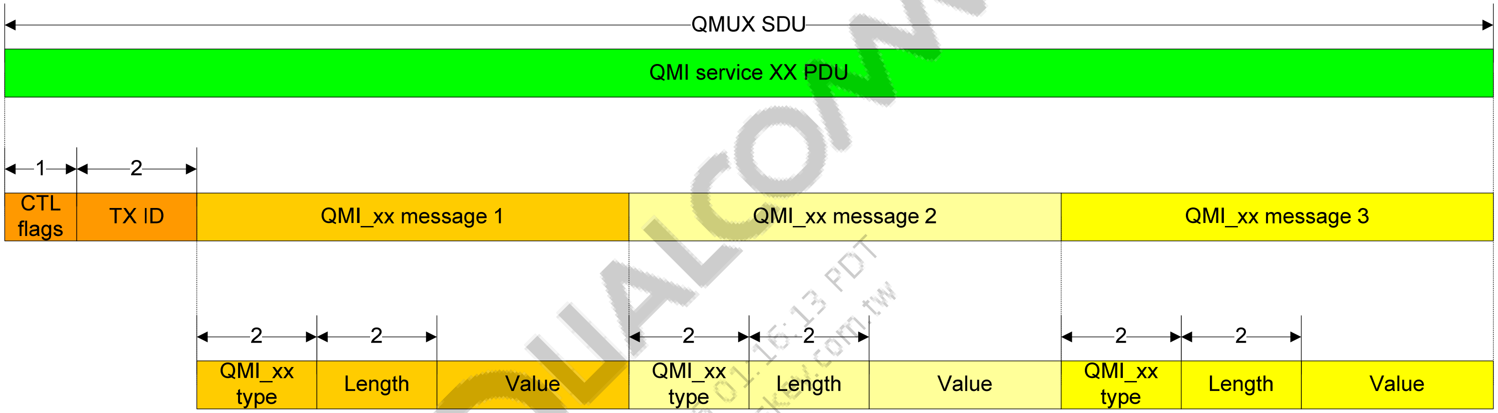
- QCTL – QMI control service
  - Internal QMI service with a single control point – QMUX layer on TE
    - Client ID of 0 reserved for QCTL control point
    - One QCTL instance per QMUX
  - Purpose
    - Client ID management for control points (applications)
    - Assigning QMUX instance identifier for associated QMUX layer
    - May be extended in future for other administrative/control tasks

QMI Message Format



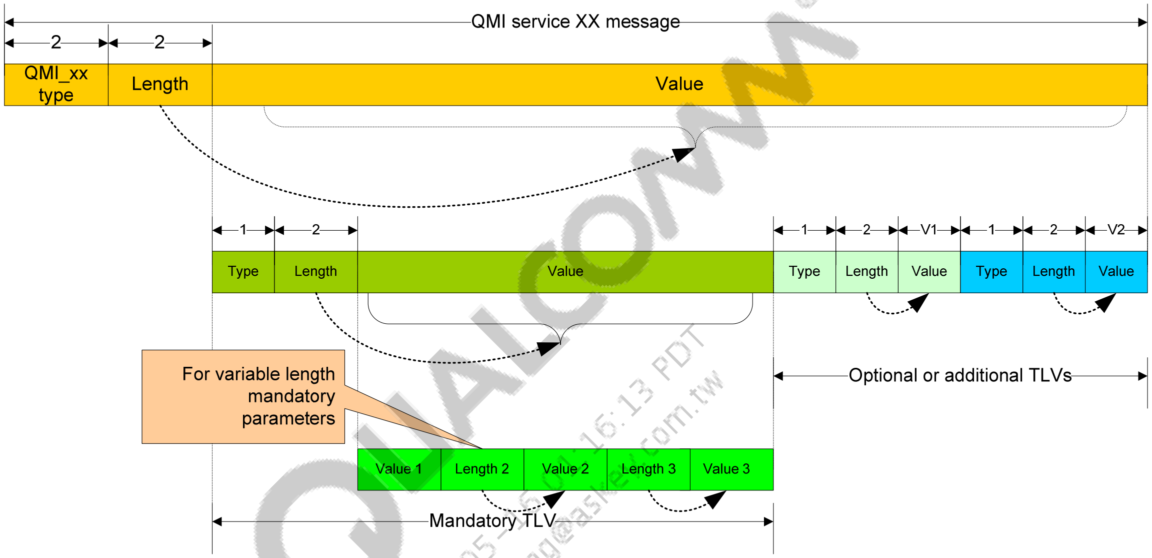
- I/F Type – Identifies the type of interface (1 = QMUX Ver 1.0)
- Length – Total length of QMUX PDU (including length itself)
- CTL – Flags specify sender type (control point or QMI service)
- QMUX Type – Identifies the QMI service to which this message belongs
- Client ID – Identifies the control point to which the message pertains
  - 0xFF = broadcast → to all Control Points
- QMUX SDU – Actual message content; these are defined by the appropriate QMI service specification

QMUX SDU Format



- Control flags
  - Compound message bit – Set if multiple messages are contained in this service PDU
  - OpCode bits – Indicate the message(s) type (command, response, or indication)
- Transaction ID
  - Identifier to correlate a response with a command
  - May be used as a sequence number in the case of indications (not required)
- Message bundling
  - Messages may be bundled in a single service PDU, indicated by compound message bit
  - All messages must belong to same QMI service
  - All messages must be of same OpCode (command, response, or indication)

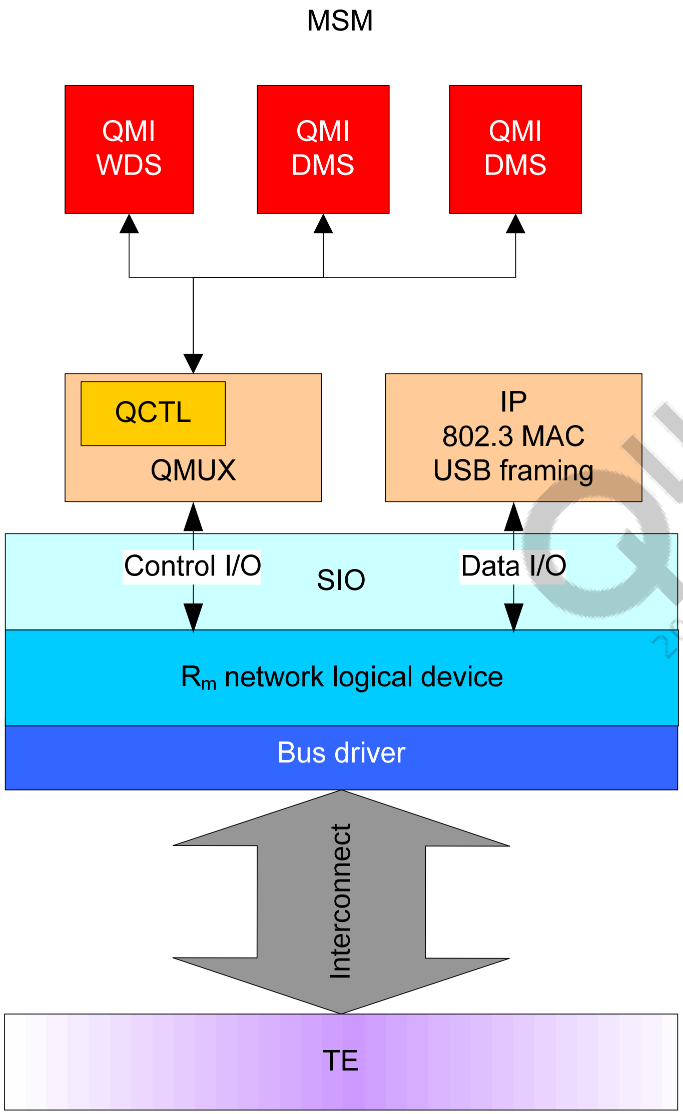
QMI Service Message Format



- Each service message consists of one or more TLVs
  - One mandatory TLV placed at the beginning of the message payload
  - Zero or more optional TLVs
  - Optional TLVs support backward compatibility within a major version
- QMI_xx type identifies the message type
  - Unique within the service type
- Overall message format itself is a TLV

R_m Network Interface (R_m Net) QMI Logical Device

R_m Network Interface



- A new QMI logical device for data service
- In accordance with QMI architecture uses:
  - Data I/O channel for IP data transfer
  - Control I/O channel for QMI messaging
    - Messaging supports functionality similar to AT commands
- May be used in place of legacy USB (PPP) modem interface
  - Legacy modem (AT/PPP) device still enumerated alongside R_m Net device

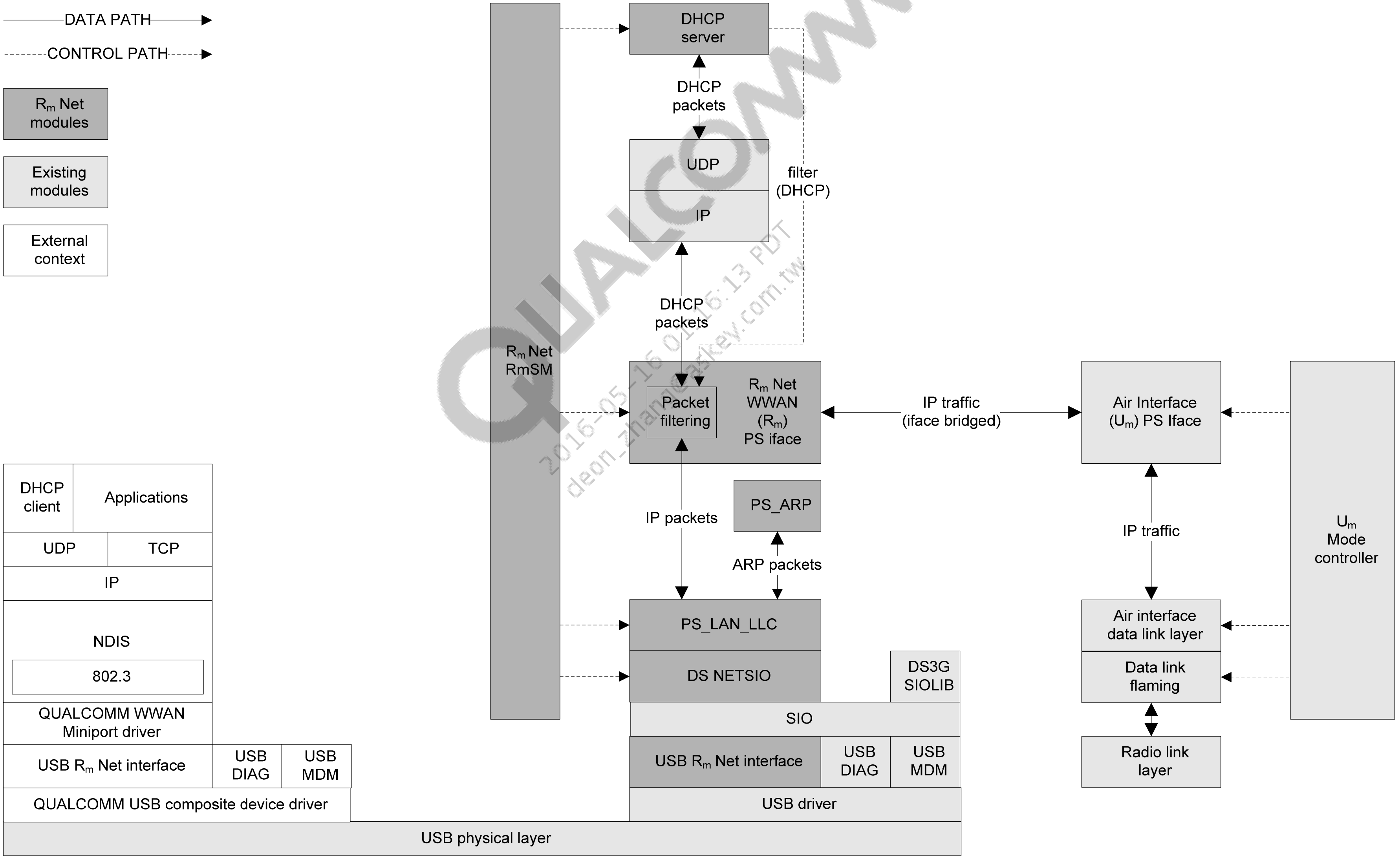
R_m Network Interface (cont.)

- Motivated by carrier demand for Connection Manager software
  - Connection Manager application controls data session on TE
  - Feature enriched – Configures and controls advanced data functionality, e.g.:
    - Authentication credentials
    - Data session profiles
    - Serving System information
    - Extensible to support new data features as they are incorporated, e.g., QoS
- Benefits
  - Asynchronous and binary control channel
  - Control interface is always available (even during data session)
  - Remove QMI control interface is always on (even during data session)
  - Eliminates R_m HDLC framing overhead
  - Will support new functionality that is not available through AT commands
  - NIC-like user experience

R_m Net Data I/O Channel

- Standard-based data link layer
  - 802.3 Ethernet data link layer (replaces PPP)
  - Address configuration via DHCP (IETF RFC 2131)
- Link layer framing implemented in USB drivers
  - End of packet delimited by short (or 0-length) USB frame
  - Reference USB drivers shipped with MSM software
- Auto-connect feature
  - Call can be established when USB cable is plugged in

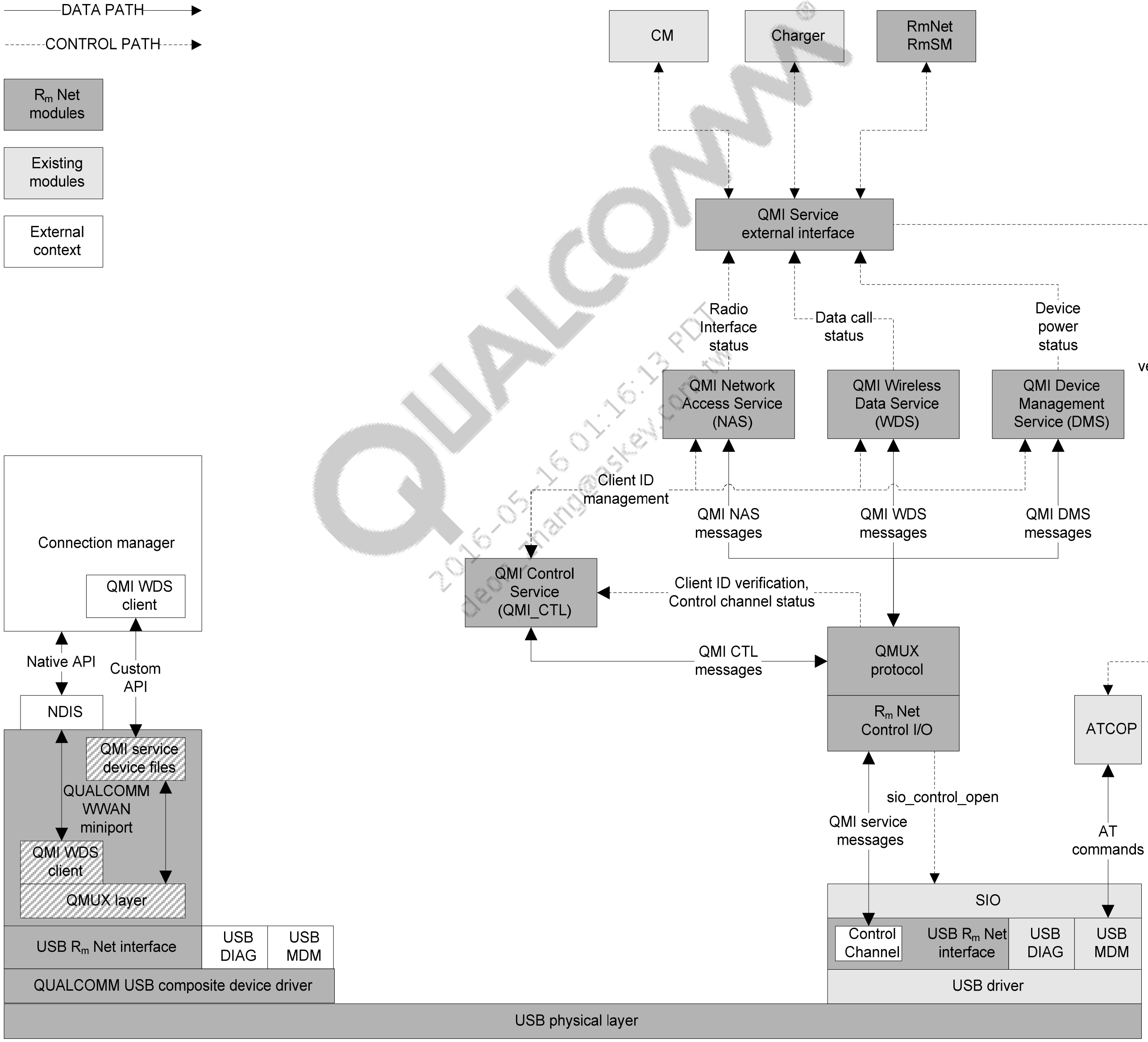
R_m Network Interface Data Path (Phase One – QMI)



R_m Net Control Channel

- Available during data connection (unlike AT commands)
- Exports functionality to be used by Connection Manager
  - QMI WDS
    - Start/stop network interface
    - Obtain bearer technology and data rate
    - Packet transfer statistics
    - Go active/go dormant
  - QMI NAS
    - Obtain serving system
    - Obtain signal strength
    - Perform network scan, register, attach
    - Configure home/preferred/forbidden networks
  - QMI DMS
    - Device identification
    - Device power source and charge level
- Additional services to be added based on licensee and carrier requirements

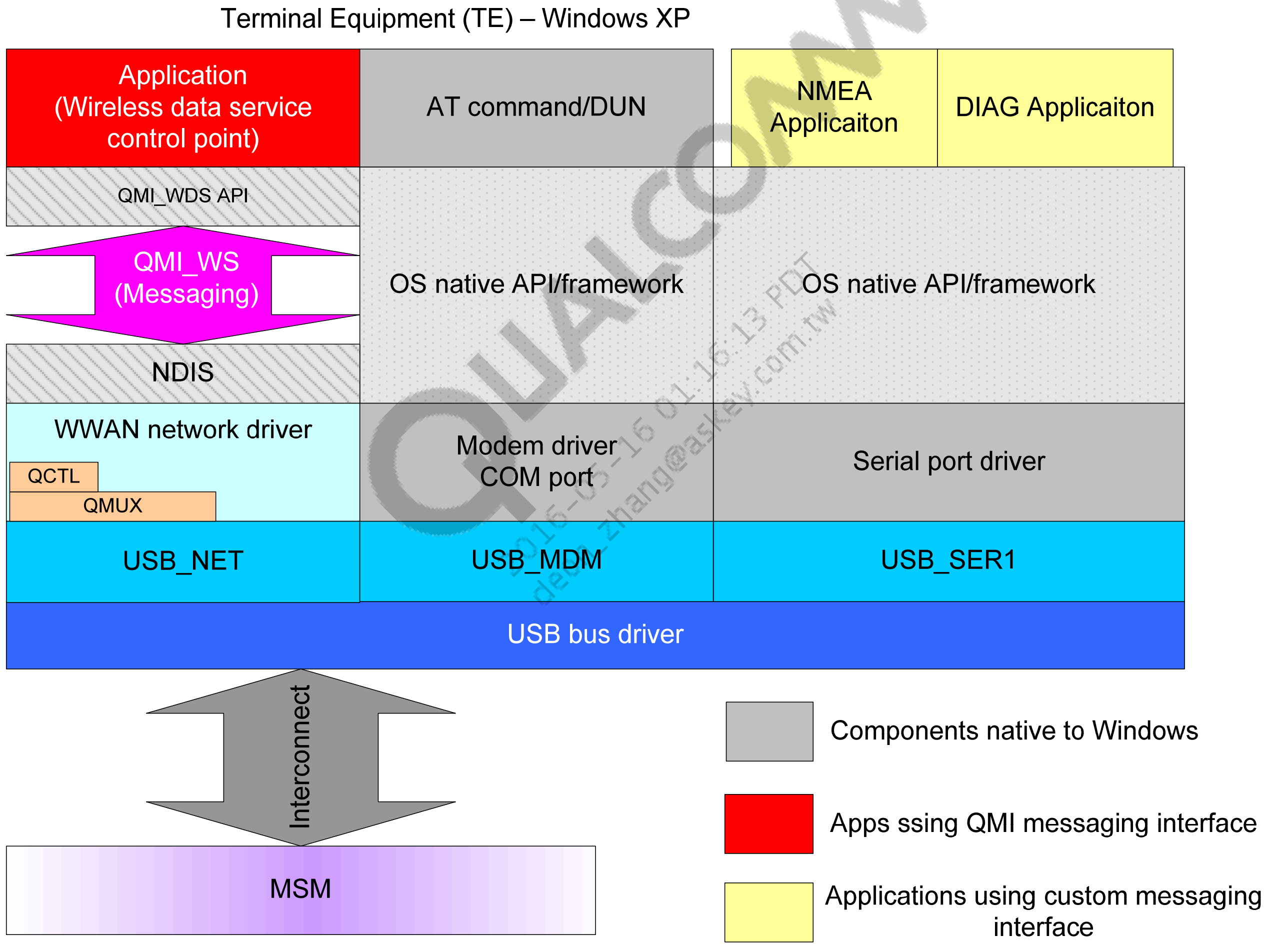
R_m Network Interface Control Path (Phase One – QMI)



R_m Net/Modem Logical Device Interaction

- Modem and R_m Net devices available simultaneously
  - PS calls supported on R_m Net logical device
  - CS/PS calls supported on modem logical device
  - AT commands continue to be supported on modem logical device
    - AT channel unavailable during CS calls
      - Only accessible to CS application (using Esc sequence/ONLINE_DATA mode)
- USB enumeration is configurable
  - R_m Net interface can be turned on/off via NV setting (2782)
    - For backward-compatibility, use legacy drivers

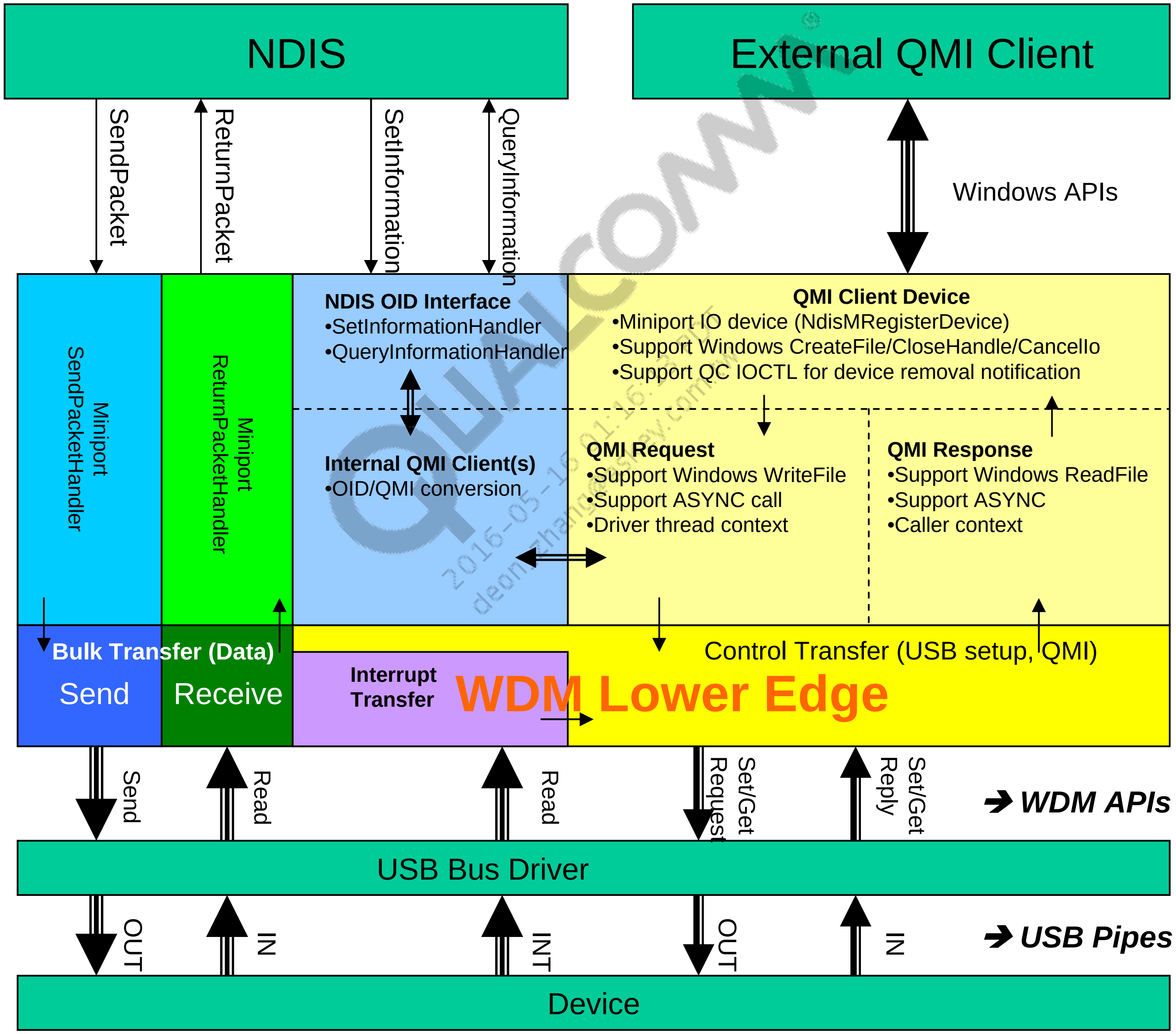
QMI TE Architecture – Windows XP



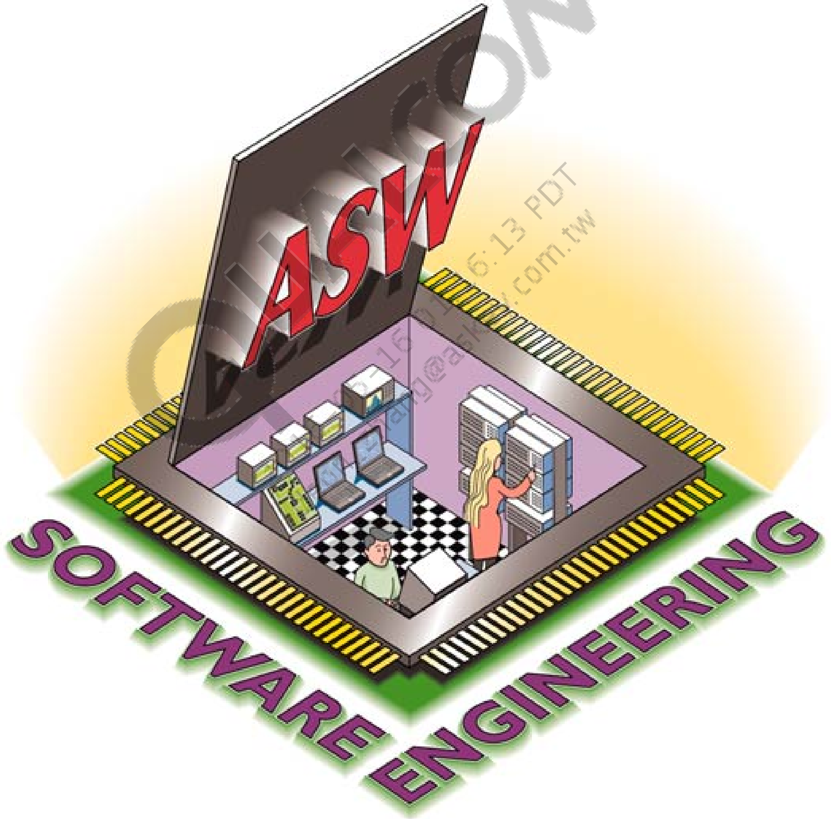
QMI Windows Reference Driver

- Supports Windows XP (NDIS 5.1)
  - Currently emulates Ethernet device
    - » 802.3 miniport
    - » Will work with Longhorn (MS claims Longhorn will be backward-compatible)
  - Portable to other Windows flavors like Windows CE, Windows 2000
- Supports USB interconnect
  - Adds a new USB interface – WWAN Network device (USB_NET)
  - Continues to enumerate legacy interfaces – Modem (USB_MDM), Serial (USB_SER
- Shipped as a reference to QUALCOMM licensees in source form only
  - Reference driver will pass WHQL tests with QC reference devices
- For commercialization, licensee:
  - Adds any proprietary customizations (if required)
  - Generates driver binaries (requires Windows DDK license)
  - WHQLs with commercial devices (if WHQL certification required)

NDIS 5.1 Miniport Driver for XP with WDM Lower Edge



Questions?



https://support.cdmatech.com/