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 CDMA Technologies

REDEFINING MOBILITY

QMI IDL/QCCI/QCSI Overview

80-N4863-1 C

Qualcomm Confidential and Proprietary

Restricted Distribution. Not to be distributed to anyone who is not an employee of either Qualcomm or a subsidiary of Qualcomm without the express approval of Qualcomm's Configuration Management.

Not to be used, copied, reproduced in whole or in part, nor its contents revealed in any manner to others without the express written permission of Qualcomm.

Qualcomm reserves the right to make changes to the product(s) or information contained herein without notice. No liability is assumed for any damages arising directly or indirectly by their use or application. The information provided in this document is provided on an "as is" basis.

This document contains Qualcomm confidential and proprietary information and must be shredded when discarded.

QUALCOMM is a registered trademark of QUALCOMM Incorporated in the United States and may be registered in other countries. Other product and brand names may be trademarks or registered trademarks of their respective owners. CDMA2000 is a registered certification mark of the Telecommunications Industry Association, used under license. ARM is a registered trademark of ARM Limited. QDSP is a registered trademark of QUALCOMM Incorporated in the United States and other countries.

This technical data may be subject to U.S. and international export, re-export, or transfer ("export") laws. Diversion contrary to U.S. and international law is strictly prohibited.

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

Revision History

Version	Date	Description
A	Mar 2011	Initial release
B	Jun 2011	Updated to Qualcomm standards
C	Oct 2011	Added slides 28-33

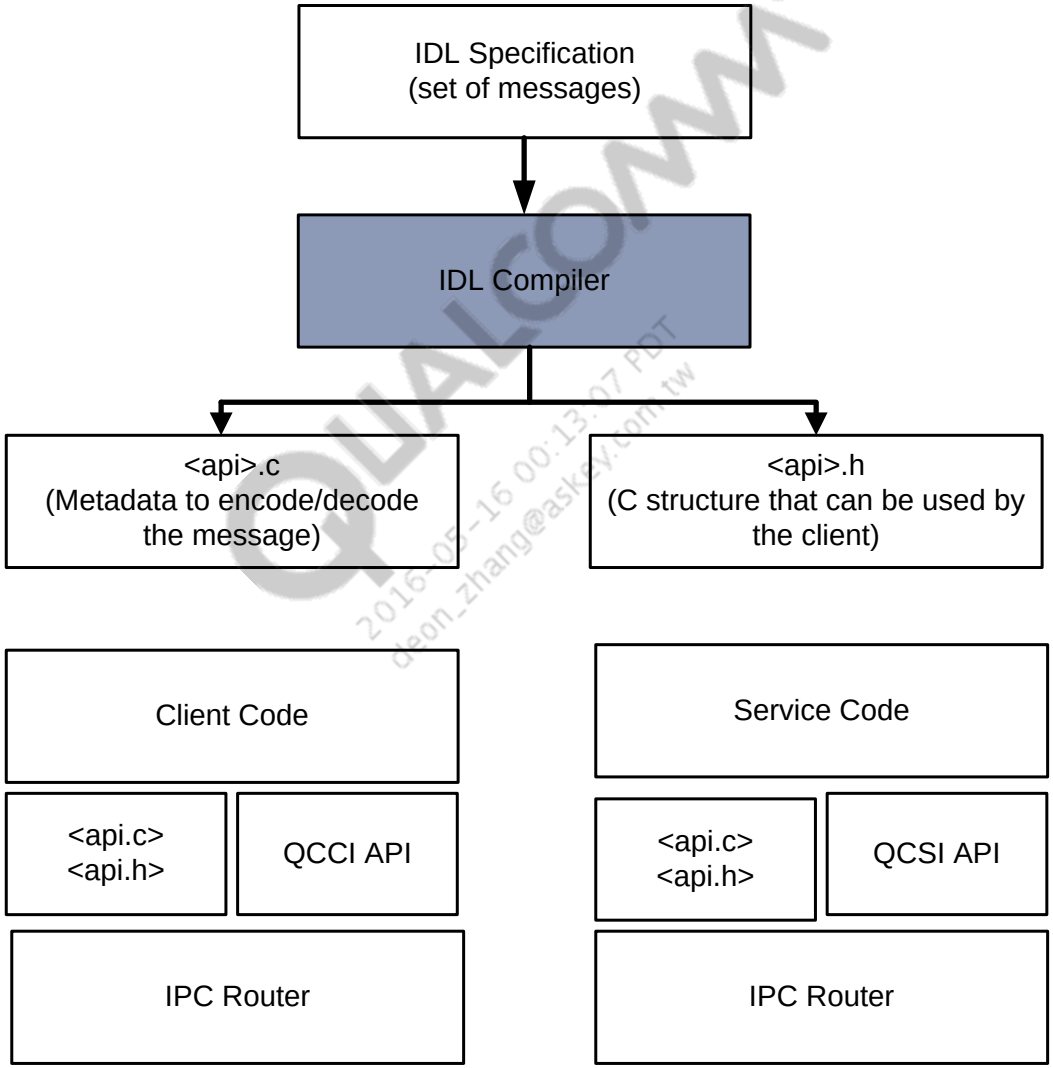
Contents

- Introduction
- IDL QMI
- QCCI
- QCSI
- Backward Compatible with Legacy Services
- QMI Core Server Framework APIs
- References
- Questions?

Introduction

REDEFINING MOBILITY

Introduction of IDL/QCCI/QCSI



Note: This picture is for new services developed using the new tools.

Introduction of IDL/QCCI/QCSI (cont.)

- IDL language and IDL Compiler
  - The syntax of QCT IDL is similar to the C programming language; the IDL Compiler generates the source file and header file
- The QMI Common Client Interface (QCCI)
  - C library that includes interfaces for all QMI services autogenerated from IDL
  - Registers clients with a service
  - Performs encoding/decoding of QMI messages
  - Exchanges messages
    - Synchronous and asynchronous command/response
    - Asynchronous indications
  - Registers clients with a service
  - IPC Router used for new services

QMI Common Service Interface (QCSI)

- QCSI wraps the QMI framework and provides service interface analogous to QCCI
- Performs encoding/decoding of QMI messages
- Includes autogenerated files for all QMI service interfaces
- C language library with functions to:
  - Register a service
  - Inform the service of client registration/deregistration
  - Deliver a request from a client and send a response
  - Send asynchronous indications
- Pairs up the service response with the request transaction

IDL QMI

REDEFINING MOBILITY

IDL

- QMI interfaces are written in an Interface Definition Language (IDL)
- Autogenerated files
  - IDL Compiler generates the <api>.c and <api>.h
    - Header file containing data types for all QMI messages
    - Source file containing meta-data needed by the encode/decode library
  - Interface documentation
- Encode/decode library
  - C routines translate between the C data structures and the wire message format
  - Library is independent and decoupled from autogenerated files
  - Meta information to encode/decode the messages would be generated in <api>.c as Service_Object
  - During Service/Client initiation, the meta information would be registered to QCSI/QCCI for message encode/decode
  - Clients/Service can send/receive messages by using QCCI/QCSI API

IDL Example

Message definition

```
revision 1
```
← Revision Statement

```
//! @MSG      QMI_PING_REQ
//! @TYPE     Request
//! @SENDER   Control point
//-------------------------------------------
------------------------------
message {
            //! Ping
            mandatory char ping[4];
} ping_req_msg;

//! @MSG      QMI_PING_DATA_RESP
//! @TYPE     Response
//! @SENDER   Service
//-------------------------------------------
------------------------------

message {
  //! Response Data
  mandatory uint8 data<PING_MAX_DATA_SIZE>;
  //! Result Code
  mandatory qmi_response_type resp;
//!< Standard response type.
} ping_data_resp_msg;
```

Service definition

```
service ping {
            //! @ID QMI_PING
            ping_req_msg
QMI_PING_REQ,
            ping_resp_msg
QMI_PING_RESP,
            ping_ind_msg
QMI_PING_IND;
            //! @ID QMI_PING_DATA
            ping_data_req_msg
QMI_PING_DATA_REQ,
            ping_data_resp_msg
QMI_PING_DATA_RESP;
            //! @QMI_PING_DATA_IND_REG
            ping_data_ind_reg_req_msg
QMI_PING_DATA_IND_REG_REQ,
            ping_data_ind_reg_resp_msg
QMI_PING_DATA_IND_REG_RESP;
            //! @QMI_PING_DATA_INDICATION
            ping_data_ind_msg
QMI_PING_DATA_IND;
} = 0x55;
```
← Service ID

.h Example

- Message definition

```
typedef struct {

  /* Mandatory */
  /*  Ping */
  char ping[4];
}ping_req_msg_v01;  /* Message */

typedef struct {

  /* Mandatory */
  /*  Pong */
  char pong[4];

  /* Mandatory */
  /*  Result Code */
  qmi_response_type_v01 resp;  /*  Standard
response type.  */
}ping_resp_msg_v01;  /* Message */
```

- Service definition

```
/*Service Message Definition*/
#define QMI_PING_REQ_V01 0x0001
#define QMI_PING_RESP_V01 0x0001
#define QMI_PING_IND_V01 0x0001
#define QMI_PING_DATA_REQ_V01 0x0002
#define QMI_PING_DATA_RESP_V01 0x0002
#define QMI_PING_DATA_IND_REG_REQ_V01 0x0003
#define QMI_PING_DATA_IND_REG_RESP_V01 0x0003
#define QMI_PING_DATA_IND_V01 0x0004

/* Service Object Accessor */
qmi_idl_service_object_type
ping_get_service_object_internal_v01
 ( int32_t idl_maj_version, int32_t
idl_min_version, int32_t library_version );

#define ping_get_service_object_v01( ) \
         ping_get_service_object_internal_v01(
\
            PING_V01_IDL_MAJOR_VERS,
PING_V01_IDL_MINOR_VERS, \
            PING_V01_IDL_TOOL_VERS )
```

← Message ID

← Service Object

.c Example

- Message meta information

```
/*Type Definitions*/
/*Message Definitions*/
static const uint8_t ping_req_msg_data_v01[] =
{
  QMI_IDL_TLV_FLAGS_LAST_TLV | 0x01,
  QMI_IDL_FLAGS_IS_ARRAY |
QMI_IDL_GENERIC_1_BYTE,
  QMI_IDL_OFFSET8(ping_req_msg_v01, ping),
  4
};

static const uint8_t ping_resp_msg_data_v01[] =
{
  0x01,
  QMI_IDL_FLAGS_IS_ARRAY |
QMI_IDL_GENERIC_1_BYTE,
  QMI_IDL_OFFSET8(ping_resp_msg_v01, pong),
  4,

  QMI_IDL_TLV_FLAGS_LAST_TLV | 0x02,
  QMI_IDL_AGGREGATE,
  QMI_IDL_OFFSET8(ping_resp_msg_v01, resp),
  0, 1
};
```

- Service object

```
/* Service Object Accessor */
qmi_idl_service_object_type
ping_get_service_object_internal_v01
 ( int32_t idl_maj_version, int32_t
idl_min_version, int32_t library_version ){
  if ( PING_V01_IDL_MAJOR_VERS !=
idl_maj_version || PING_V01_IDL_MINOR_VERS !=
idl_min_version
       || PING_V01_IDL_TOOL_VERS !=
library_version)
  {
    return NULL;
  }
  return
(qmi_idl_service_object_type)&ping_qmi_idl_serv
ice_object_v01;
}
```

← Service Object

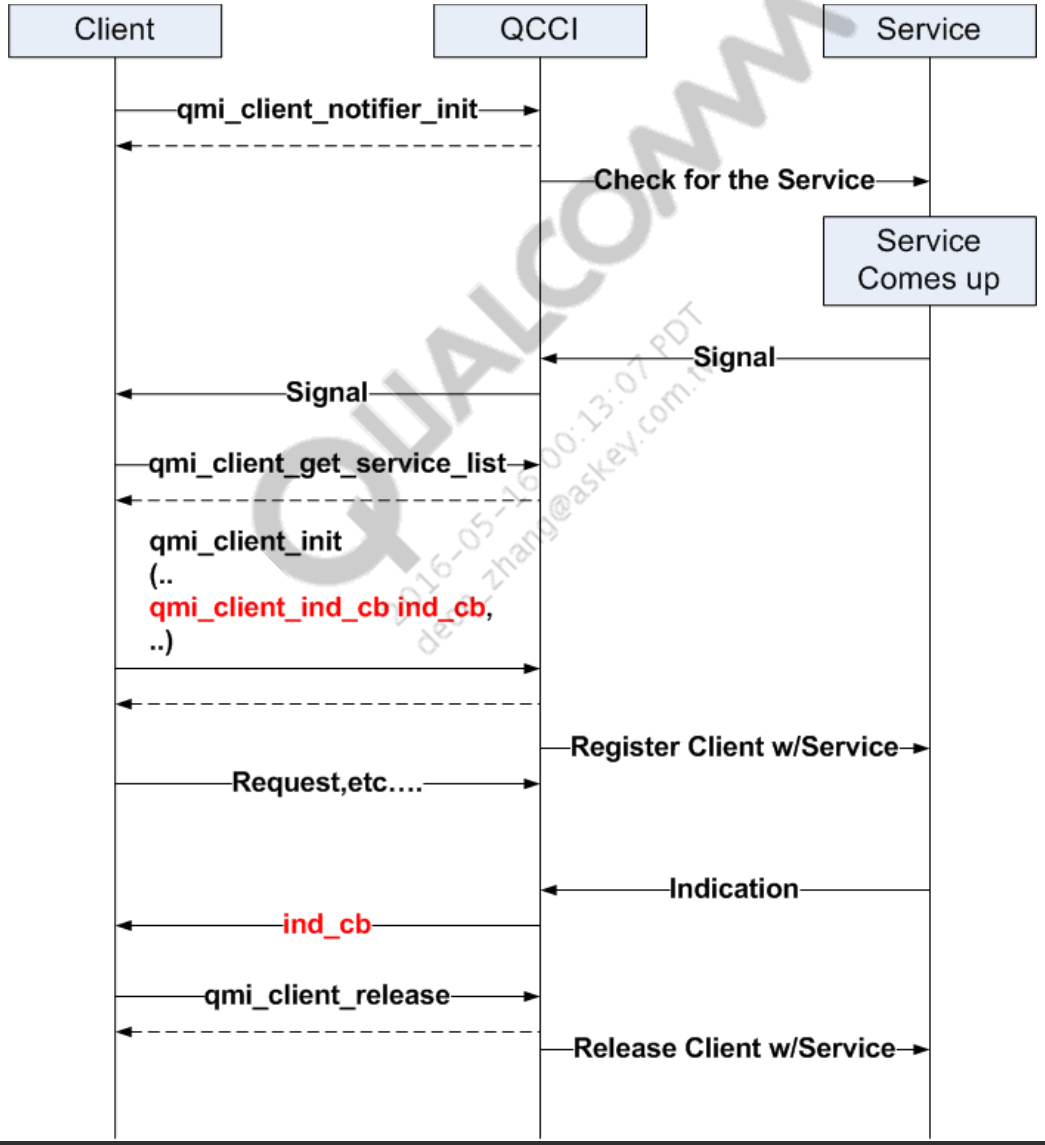
QCCI

REDEFINING MOBILITY

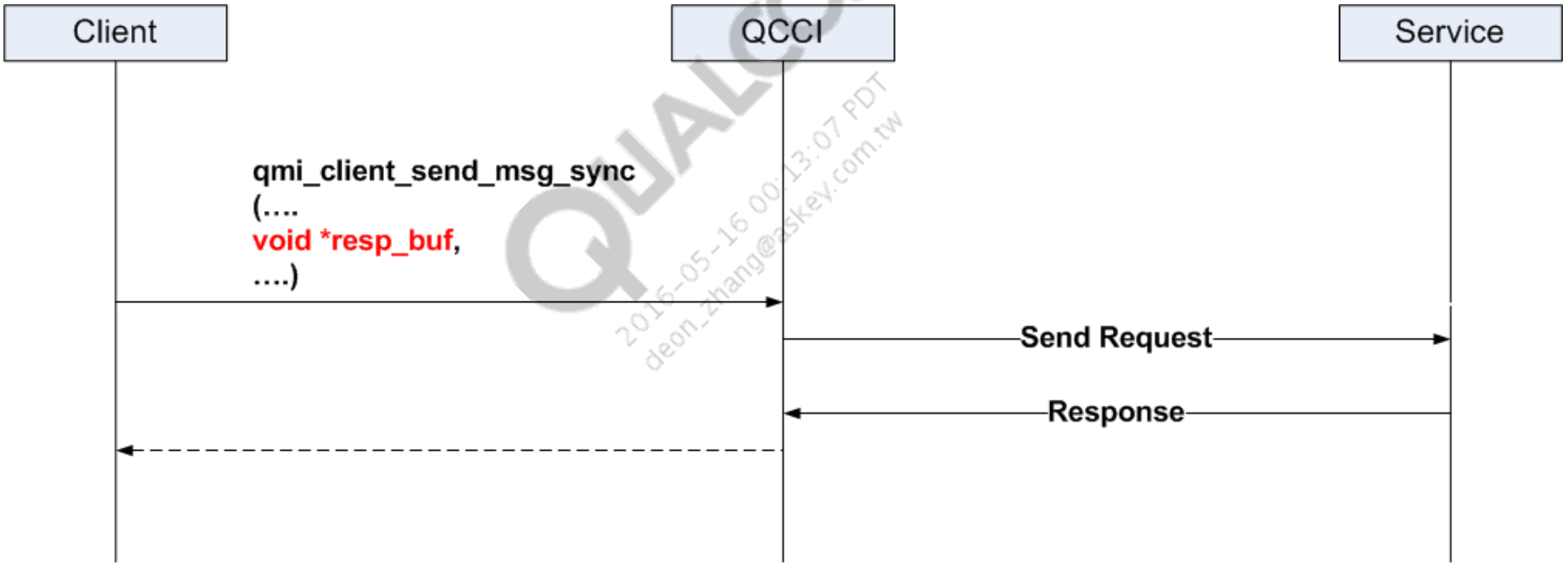
QCCI

- QCCI is a set of APIs that allow the client to send and receive QMI messages to and from the server.
- QCCI APIs
  - Callback function prototypes
    - qmi_client_recv_raw_msg_async_cb
    - qmi_client_recv_msg_async_cb
    - qmi_client_ind_cb
  - Connection APIs
    - qmi_client_notifier_init
    - qmi_client_init
    - qmi_client_get_service_list
  - Message sending APIs
    - qmi_client_send_raw_msg_async/qmi_client_send_raw_msg_sync
    - qmi_client_send_msg_async/ qmi_client_send_msg_sync
    - qmi_client_delete_async_txn
  - Release connection APIs
    - qmi_client_release
  - Encode and decode APIs
    - qmi_client_message_encode
    - qmi_client_message_decode

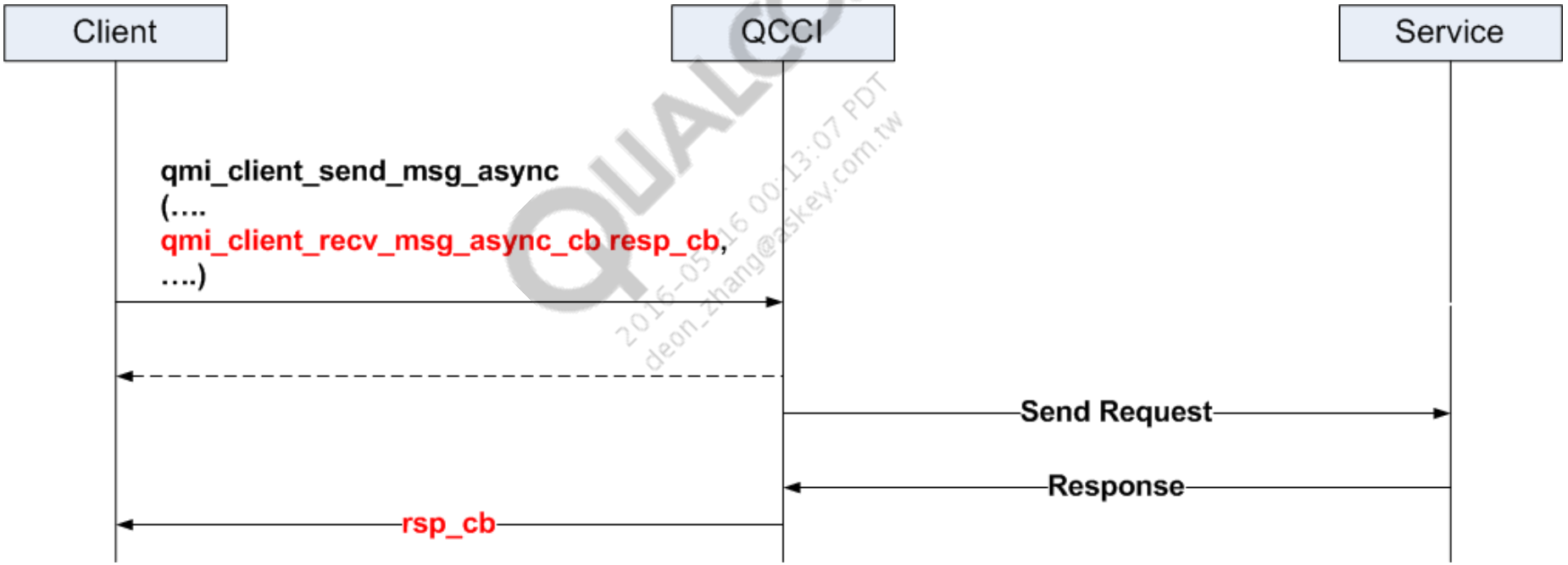
QCCI – Initializing a Client



QCCI – Sending a Message Synchronous



QCCI – Sending a Message Asynchronous



QCSI

REDEFINING MOBILITY

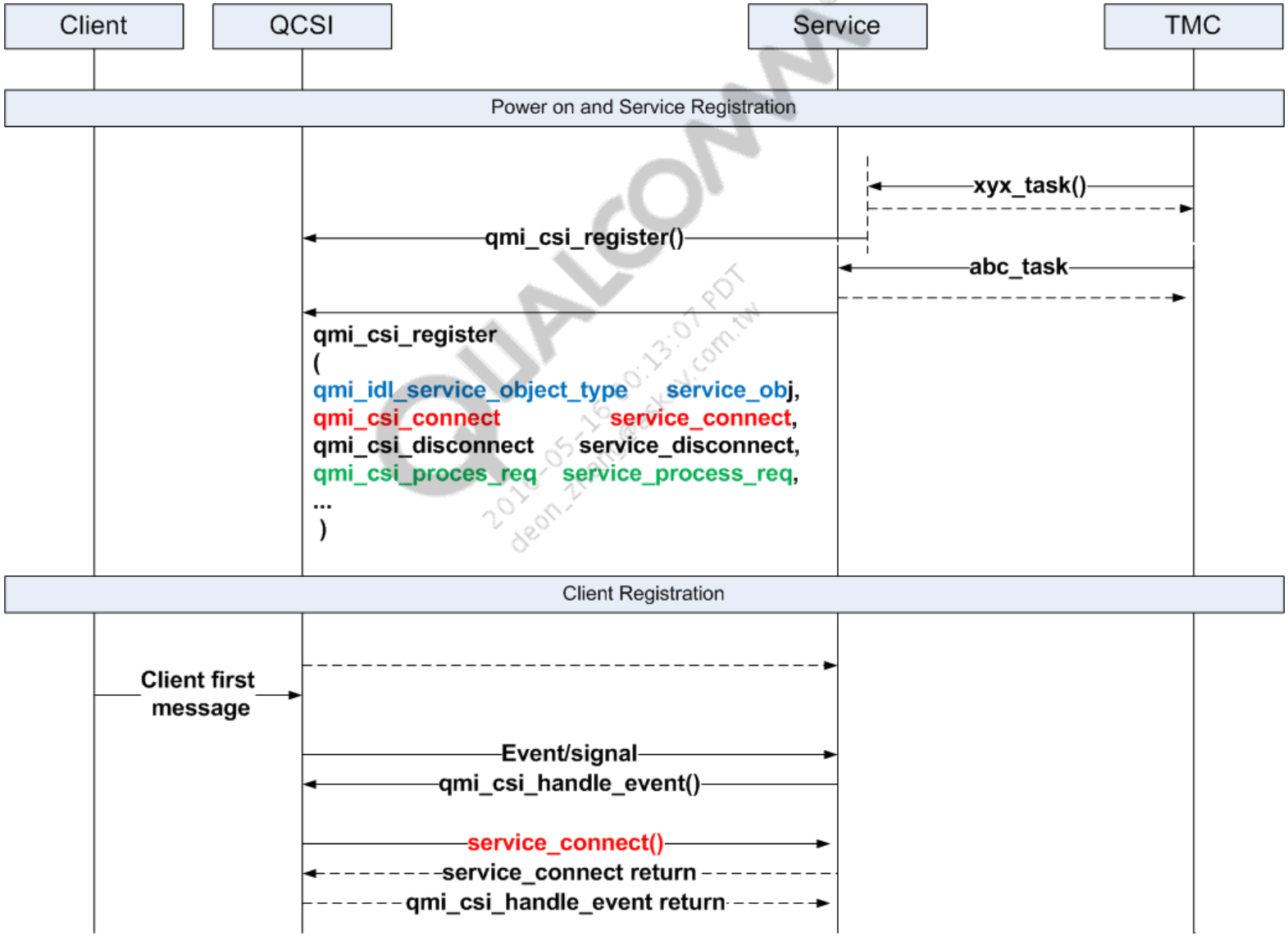
QCSI – QMI Common Service Interface

- Used with the autogenerated files from the QMI IDL compiler to write a service that can receive and respond to messages from a client, as well as send indication
- Simplifies the process of writing a QMI service by a set of APIs and callbacks for:
  - Handling requests
  - Sending responses and indications
  - Handling client connect and disconnect event
  - Encoding and decoding of messages

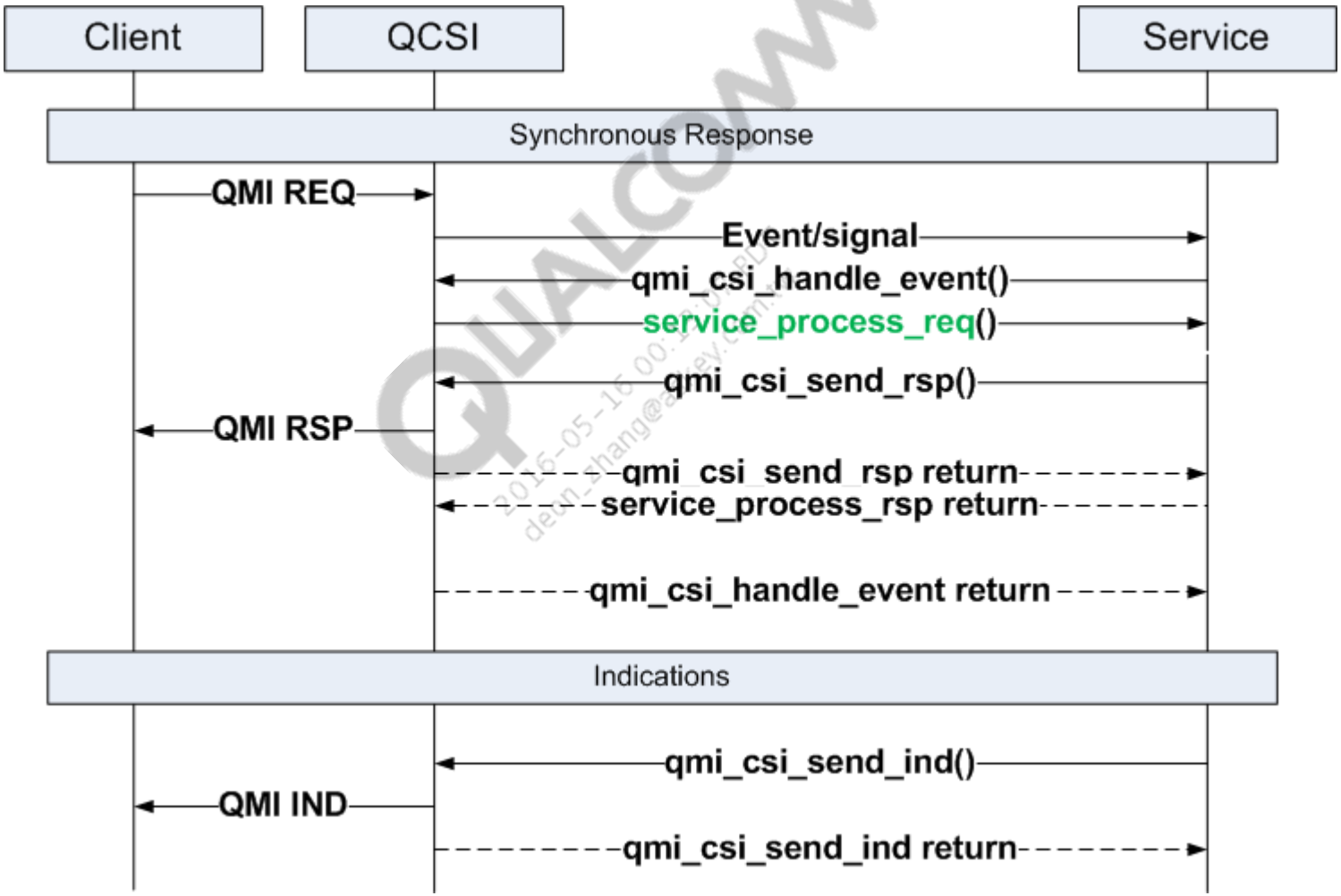
QCSI APIs

- Callback function prototypes
  - qmi_csi_connect
  - qmi_csi_disconnect
  - qmi_csi_process_req
- Registration APIs
  - qmi_csi_register
  - qmi_csi_unregister
- Message sending APIs
  - qmi_csi_send_resp
  - qmi_csi_send_ind
  - qmi_csi_send_broadcast_ind
- Event handling APIs
  - qmi_csi_handle_event

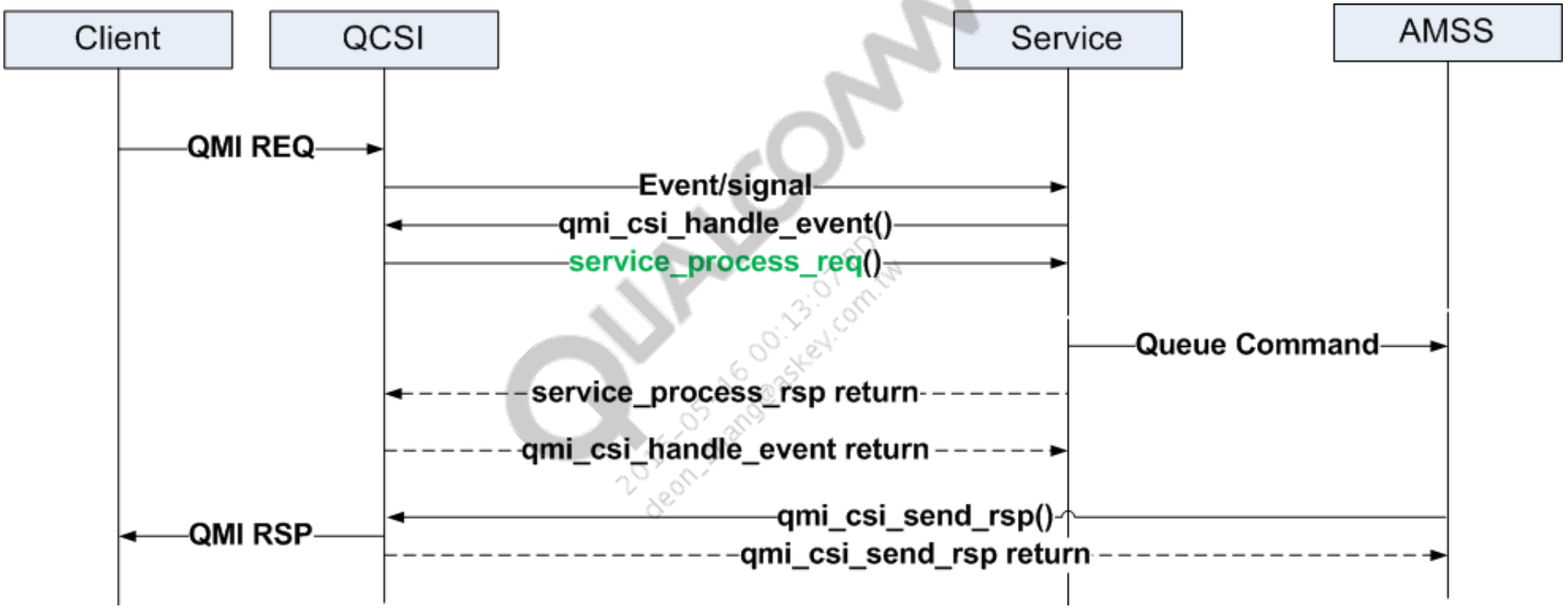
QCSI – Service Registration and Client Registration



QCCI – Synchronous Response and Indication



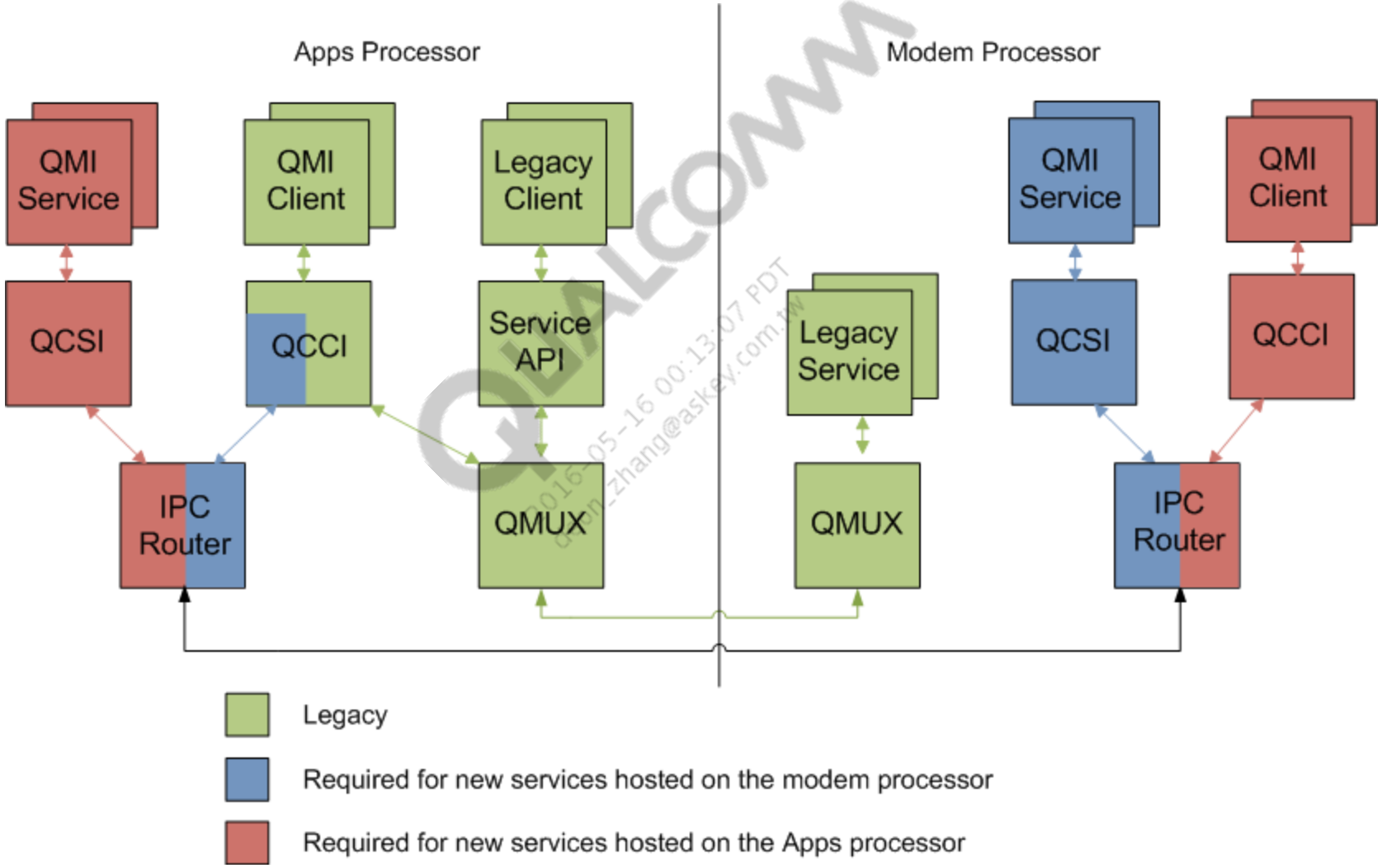
QCCI – Asynchronous Response



Backward Compatible with Legacy Services

REDEFINING MOBILITY

Backward Compatible with Legacy Services



Backward Compatible with Legacy Services (cont.)

- QMI Link Layer
  - QCCI/QCSI decides to use Qmux or IPC Router, based on Service ID
  - IPC Router used for new services
  - QCCI/QCSI interface is the same even though Qmux and IPC Router interfaces are different
- New QCSI/QCCI exists in both the modem side and app side.
  - Services located in the modem side use the QCSI in the modem side, e.g., Universal Location Provider (ULP) Service
  - Services located in the app side use the QCSI in the apps side, e.g., EFS Service

QMI Core Server Framework APIs

REDEFINING MOBILITY

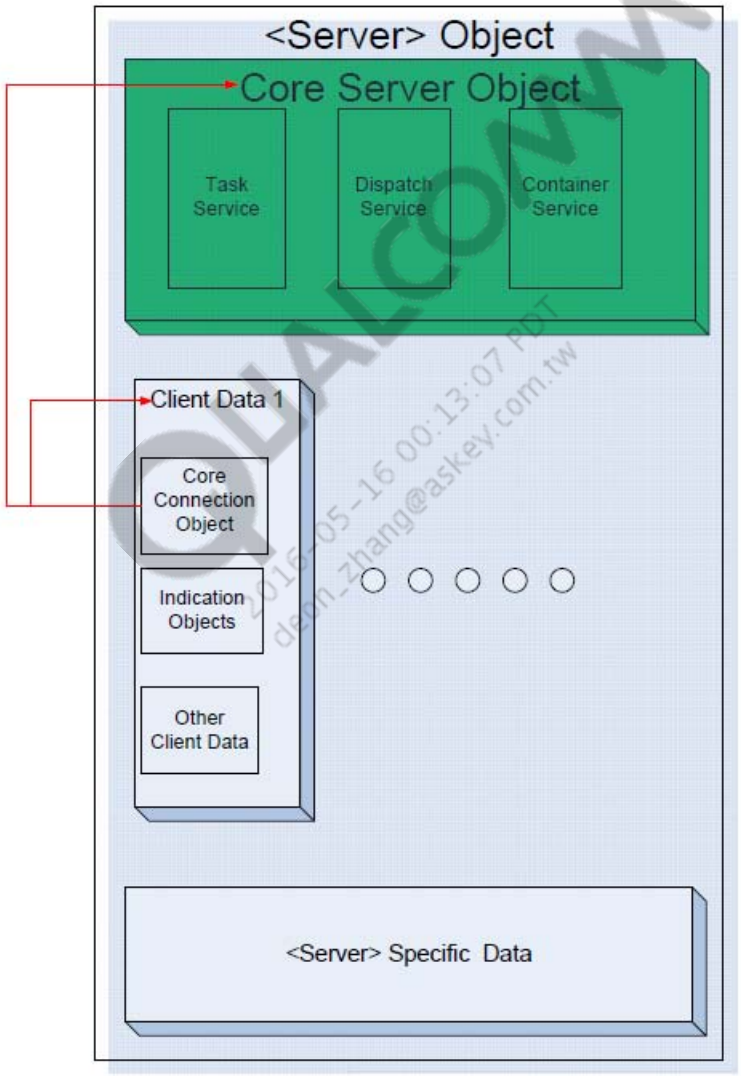
QMI Core Server Framework APIs

- These APIs are built on top of the QCSI framework
- The primary purpose is to facilitate the clients to implement a server with less effort
- The framework allows the users to write an object-based server that extends the core server object. The core server object provides generic functionality that every service needs

QMI Core Server Framework APIs

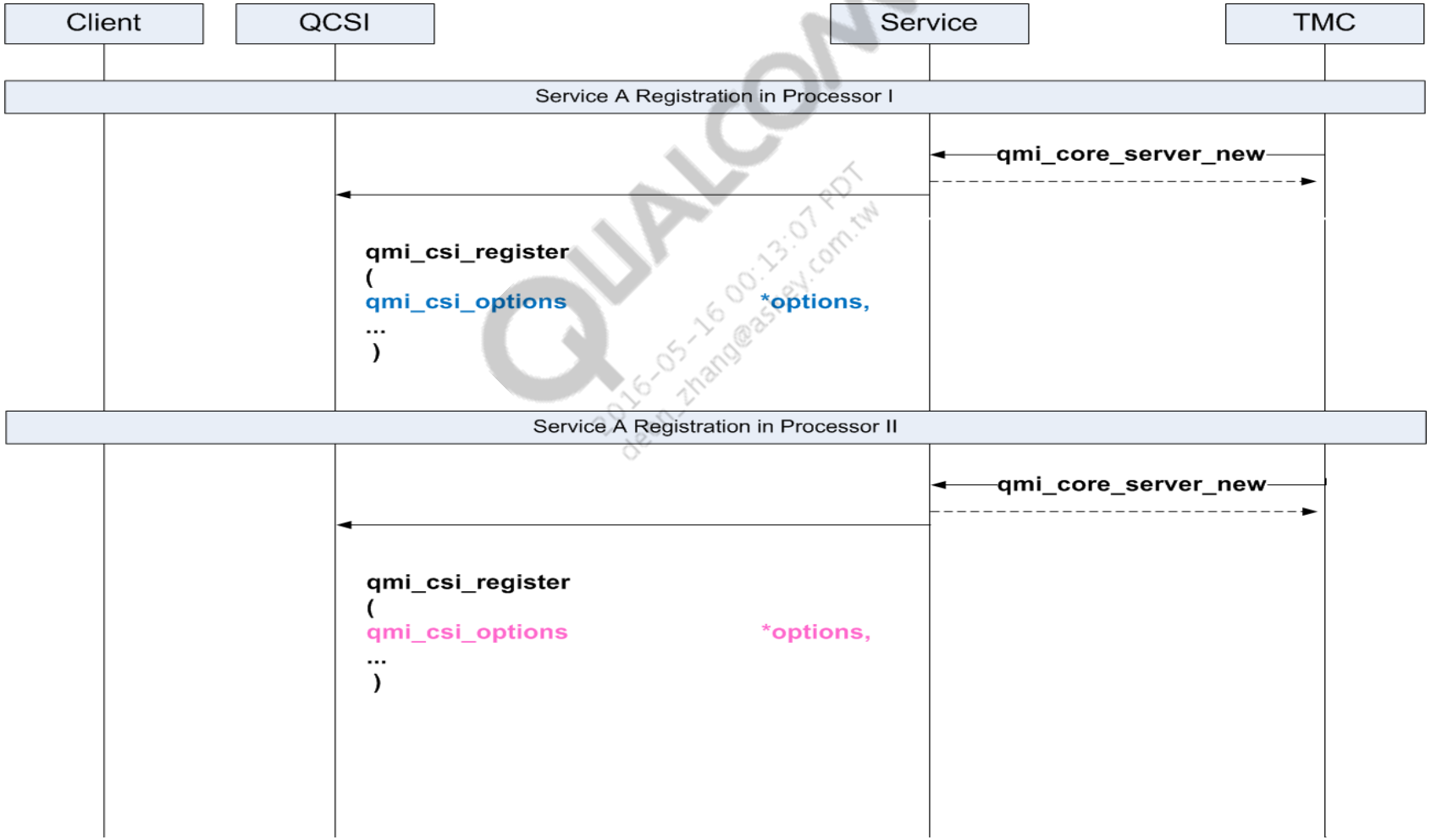
- Callback Function Prototypes
  - qmi_csi_connect
  - qmi_csi_disconnect
  - qmi_csi_process_req
- Constructor/Destructor APIs
  - qmi_core_server_new
  - qmi_core_server_delete
- Registration APIs
  - qmi_core_server_register
  - qmi_core_server_unregister
  - qmi_core_server_register_client
  - qmi_core_server_unregister_client
- Message Dispatch APIs
  - qmi_core_server_dispatch_msg
  - qmi_core_server_send_ind
  - qmi_core_server_send_resp
- Event Handling APIs
  - qmi_core_server_handle_event
- Utility APIs
  - qmi_core_server_start_server
  - qmi_core_server_check_valid_object
  - qmi_core_server_get_client_data
  - qmi_core_server_get_os_params

QMI Core Server Framework APIs – Service Object

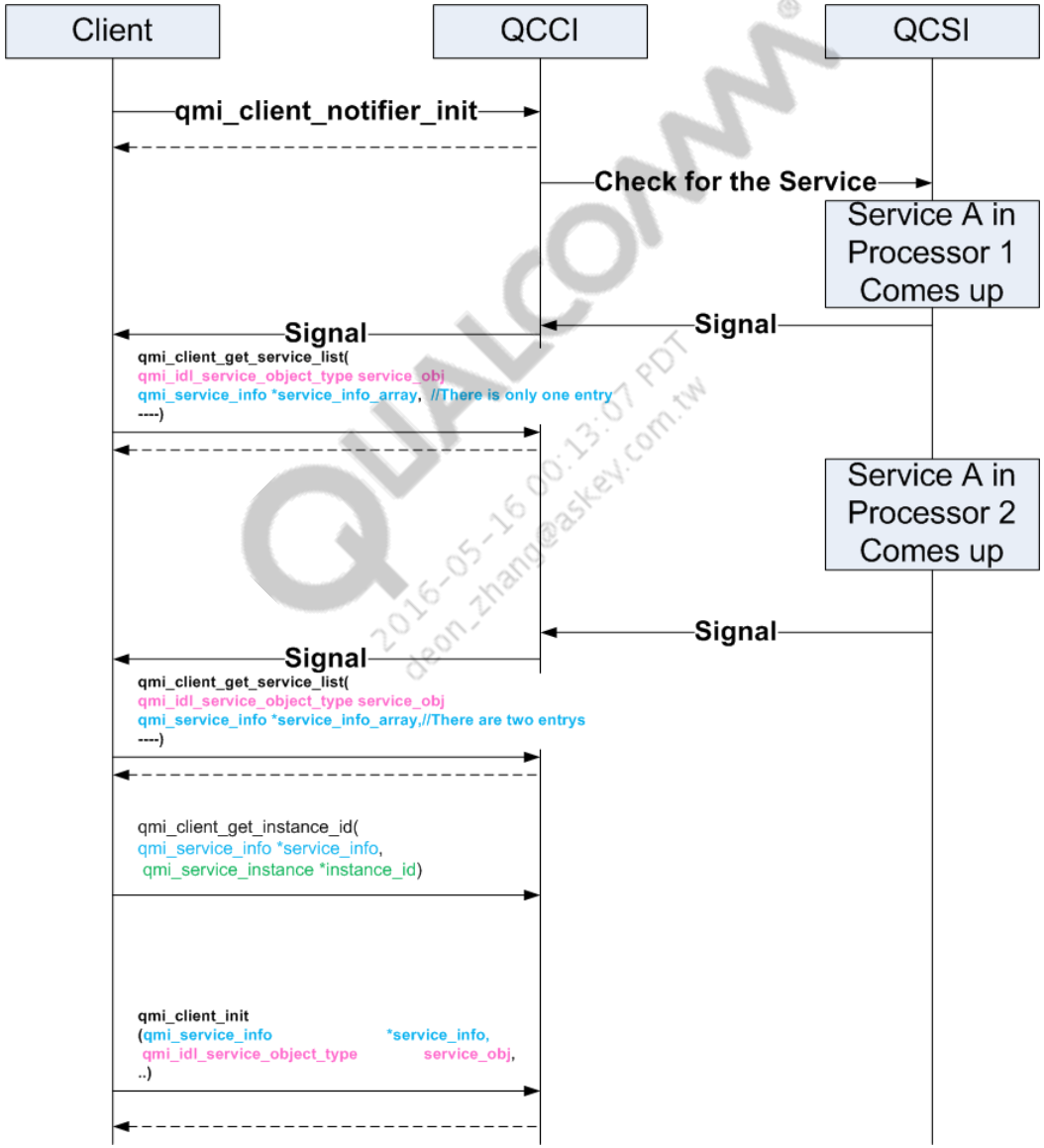


Example – Service A Running in Two Processors

- There is instance_id in qmi_csi_options
- <service id> : <version> : <instance> should be unique



Example – Client Initialization with Service A Running in Processors 1 and 2



References

Ref.	Document	
Qualcomm		
Q1	*Application Note: Software Glossary for Customers*	CL93-V3077-1
Q2	*QMI Client API Reference Guide*	80-N1123-1
Q3	*QMI Common Service APIs Reference Guide*	80-N1123-2
Q4	*QCT API Guidelines and Tools*	80-N0846-1

Questions?

https://support.cdmatech.com

REDEFINING MOBILITY