Qualcomm
TECHNOLOGIES, INC

Data Services Profile Registry API

80-N4766-1 B

October 24, 2014

QUALCOMM®
2015-05-16 00:13:38 PDT
dawn_zhang@askey.com.tw

Confidential and Proprietary – Qualcomm Technologies, Inc.

NO PUBLIC DISCLOSURE PERMITTED: Please report postings of this document on public servers or websites to: DocCtrlAgent@qualcomm.com.

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

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

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 confidential and proprietary information and must be shredded when discarded.

Qualcomm is a trademark of QUALCOMM Incorporated, registered in the United States and other countries. All QUALCOMM Incorporated trademarks are used with permission. Other product and brand names may be trademarks or registered trademarks of their respective owners.

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

© 2011, 2014 Qualcomm Technologies, Inc.
All rights reserved.

Contents

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

Figures

Tables

QUALCOMM®
2016-05-16 00:13:38 PDT
deon_zhang@askey.com.tw

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

Revision History

Revision	Date	Description
A	Feb 2011	Initial release.
B	Oct 2014	Added Register/deregister for callbacks (Section 3.1.6). Updated Sections 3.2.1, 3.2.2, and3.2.3. Added several new functions in Chapter 4. Added Chapter 6, Usage of DS PROFILE APIs

QUALCOMM
2016-05-16 00:13:38 PDT
deon_zhang@askey.com.tw

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

1 Introduction

1.1 Purpose

This document describes the new Profile Registry Software Library to be used by software entities for Data Services Profile management. The Profiles reside in persistent storage on Modem/Apps Processor. This library provides a consistent programming interface to manage these Profiles in both single and dual processor solutions for all radio technologies and HLOS.

1.2 Scope

This document is intended for developers who are using the Data Services Profile Registry API.

1.3 Conventions

Function declarations, function names, type declarations, and code samples appear in a different font, e.g., `#include`.

Code variables appear in angle brackets, e.g., `<number>`.

Parameter types are indicated by arrows:

- → Designates an input parameter
- ← Designates an output parameter
- ↔ Designates a parameter used for both input and output

Shading indicates content that has been added or changed in this revision of the document.

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

1.4 References

Reference documents are listed in Table 1-1. Reference documents that are no longer applicable are deleted from this table; therefore, reference numbers may not be sequential.

Table 1-1 Reference documents and standards

Ref.	Document	
Qualcomm Technologies		
Q1	*Application Note: Software Glossary for Customers*	CL93-V3077-1
Q2	*QMI Master Document* *(Used to look up the appropriate QMI WDS (QMI Wireless Data Servcie) Spec document)*	80-NK255-1
Standards		
S1	*Mobile radio interface Layer 3 specification; Core network protocols; Stage 3*	3GPP TS 24.008 V6.9.0

1.5 Technical Assistance

For assistance or clarification on information in this document, submit a case to Qualcomm Technologies, Inc. (QTI) at https://support.cdmatech.com/.

If you do not have access to the CDMATech Support website, register for access or send email to support.cdmatech@qti.qualcomm.com.

1.6 Acronyms

For definitions of terms and abbreviations, refer to [Q1]. Table 1-2 lists terms that are specific to this document.

Table 1-2 Definition of document-specific terms

Term	Definition
Record	Refers to a profile
Parameters	Individual fields with the profile
Registry	A collection of records

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

2 Overview/Requirements

The requirement is to provide software a programming interface for applications to access the data services profile registry. The mode handlers use the profile registry as persistent storage for data call settings. In both Qualcomm Technology, Inc. (QTI)'s single processor and dual processor architectures, the profiles reside on the modem processor. The information contained within the profile is used for data call setup. This API is to be used by software entities on both the application and modem processor.

The salient features are:

- Same programming interface for all radio technologies by hiding interaction with mode handler-specific interfaces, e.g., UMTS profile registry, CDMA profile registry, EPC profile registry
- Provides a synchronous interface on the modem and apps processors over the Qualcomm Messaging Interface (QMI); it serves as the method of communication between the apps and modem processors
- New parameter addition is easier

QUALCOMM
2016-05-16 00:13:32 PDT
deon_zhang@askey.com

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

3 New Data Services Profile Registry API

3.1 Basic operations

The API is intended to be used by clients to perform the following:

- Create a profile
- Delete a profile
- Get a list of profiles for a radio technology type
- Get/set individual profile parameters
- Get/set default profile number
- Register/deregister for callbacks on Profile changes.

A profile is uniquely identified by the tuple: [Profile ID, Tech Type]. Examples of technology family are UMTS, CDMA1x, etc. In case of subscription-specific get/set APIs, Subscription ID is a mandatory parameter.

3.1.1 Create profile

Create a profile according to parameters provided by user. The create operation will return a Profile ID to the user. At a minimum, the Tech Type must be specified.

3.1.2 Delete profile

Delete a profile identified by the tuple: [Profile ID, Tech Type].

3.1.3 Get list of profiles

Get a list of valid [Profile IDs, Tech Types]. This operation is expected to return Profile IDs for the specified Tech Type.

3.1.4 Get/set profile parameters

Get or Set Profile parameters individually. The tuple [Profile ID, Tech Type] must be provided for this operation to work.

3.1.5 Get/set default profile number

Provide ability to mark or unmark a profile as default for that call type. The supported call types are Embedded, RmNet, and ATCoP.

Subscription specific get/set default profile number APIs provides the ability to mark or unmark a profile as default for that call type on a particular subscription. The supported call types are Embedded and Tethered.

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

3.1.6 Register/deregister for callbacks

NOTE: This section was added to this document revision.

Enables the user to register or deregister for callbacks on changes to a particular Profile or all Profiles of a specific technology

Provides a list of profile numbers with the techtypes that have changed, which is conveyed as an indication to clients (e.g., QMI)

3.2 Profile registry programming interface

3.2.1 UMTS interface specification

3.2.1.1 ds_profile_3gpp_ip_version_enum_type

```
typedef enum {
   DS_PROFILE_3GPP_IP_V4   = 4,
   DS_PROFILE_3GPP_IP_V6   = 6,
   DS_PROFILE_3GPP_IP_V4V6     = 10,
   DS_PROFILE_3GPP_IP_MAX  = 0xff
}  ds_profile_3gpp_ip_version_enum_type
```

3.2.1.2 ds_profile_3gpp_pdp_access_control_e_type

```
typedef enum {
   DS_PROFILE_3GPP_PDP_ACCESS_CONTROL_NONE         = 0x0,
   DS_PROFILE_3GPP_PDP_ACCESS_CONTROL_REJECT       = 0x1,
   DS_PROFILE_3GPP_PDP_ACCESS_CONTROL_PERMISSION   = 0x2,
   DS_PROFILE_3GPP_PDP_ACCESS_CONTROL_MAX          = 0xff
}  ds_profile_3gpp_pdp_access_control_e_type
```

3.2.1.3 ds_profile_3gpp_pdp_type_enum_type

```
typedef enum {
   DS_PROFILE_3GPP_PDP_IP                    = 0x0,  /* PDP type IP */
   DS_PROFILE_3GPP_PDP_PPP,                      /* PDP type PPP */
   DS_PROFILE_3GPP_PDP_IPV6,                     /* PDP type IPV6 */
   DS_PROFILE_3GPP_PDP_IPV4V6,                   /* PDP type IPV4V6 */
   DS_PROFILE_3GPP_PDP_MAX                       = 0xff
}  ds_profile_3gpp_pdp_type_enum_type
```


MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

3.2.1.4 ds_profile_3gpp_auth_pref_type

```
typedef enum {
   DS_PROFILE_3GPP_AUTH_PREF_PAP_CHAP_NOT_ALLOWED  = 0x0,  /* No
                                                   authentication       */
   DS_PROFILE_3GPP_AUTH_PREF_PAP_ONLY_ALLOWED      = 0x1,  /* PAP
                                                   authentication      */
   DS_PROFILE_3GPP_AUTH_PREF_CHAP_ONLY_ALLOWED     = 0x2,  /* CHAP
                                                   authentication     */
   DS_PROFILE_3GPP_AUTH_PREF_PAP_CHAP_ALLOWED      = 0x3,  /* PAP/CHAP
                                                   authentication */
   DS_PROFILE_3GPP_AUTH_PREF_MAX                   = 0xff
}  ds_profile_3gpp_auth_pref_type
```

3.2.1.5 ds_profile_3gpp_traffic_class_type

```
typedef enum {
   DS_PROFILE_3GPP_TC_SUBSCRIBED         = 0x0, /* Subscribed */
   DS_PROFILE_3GPP_TC_CONVERSATIONAL     = 0x1, /* Conversational */
   DS_PROFILE_3GPP_TC_STREAMING          = 0x2, /* Streaming */
   DS_PROFILE_3GPP_TC_INTERACTIVE        = 0x3, /* Interactive */
   DS_PROFILE_3GPP_TC_BACKGROUND         = 0x4, /* Background  */
   DS_PROFILE_3GPP_TC_RESERVED           = 0xff
}  ds_profile_3gpp_traffic_class_type
```

3.2.1.6 ds_profile_3gpp_qos_delivery_order_type

```
typedef enum {
   DS_PROFILE_3GPP_DO_SUBSCRIBED     = 0x0, /* Subscribed */
   DS_PROFILE_3GPP_DO_ON             = 0x1, /* With delivery order */
   DS_PROFILE_3GPP_DO_OFF            = 0x2, /* Without delivery order */
   DS_PROFILE_3GPP_DO_RESERVED       = 0xff
}  ds_profile_3gpp_qos_delivery_order_type
```

QUALCOMM® 2015-05-18 00:15:38 PDT deon.cheng@askey.com.tw

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

3.2.1.7 ds_profile_3gpp_sdu_err_ratio_type

```
typedef enum {
   DS_PROFILE_3GPP_SDU_ERR_RATIO_SUBSCRIBE          = 0x0,   /* Subscribed  */
     DS_PROFILE_3GPP_SDU_ERR_RATIO_1ENEG2           = 0x1,   /* 1E-2  */
     DS_PROFILE_3GPP_SDU_ERR_RATIO_7ENEG3           = 0x2,   /* 7E-3  */
     DS_PROFILE_3GPP_SDU_ERR_RATIO_1ENEG3           = 0x3,   /* 1E-3  */
     DS_PROFILE_3GPP_SDU_ERR_RATIO_1ENEG4           = 0x4,   /* 1E-4  */
     DS_PROFILE_3GPP_SDU_ERR_RATIO_1ENEG5           = 0x5,   /* 1E-5  */
     DS_PROFILE_3GPP_SDU_ERR_RATIO_1ENEG6           = 0x6,   /* 1E-4  */
     DS_PROFILE_3GPP_SDU_ERR_RATIO_1ENEG1           = 0x7,   /* 1E-1  */
   DS_PROFILE_3GPP_SDU_ERR_RATIO_MAX                = 0x8,   /* Max Val     */
   DS_PROFILE_3GPP_SDU_ERR_RESERVED                 = 0xff
}  ds_profile_3gpp_sdu_err_ratio_type
```

3.2.1.8 ds_profile_3gpp_residual_ber_ratio_type

```
typedef enum {
   DS_PROFILE_3GPP_RESIDUAL_BER_SUBSCRIBE     = 0x0,   /* Subscribed */
   DS_PROFILE_3GPP_RESIDUAL_BER_5ENEG2        = 0x1,   /* 5E-2 */
   DS_PROFILE_3GPP_RESIDUAL_BER_1ENEG2        = 0x2,   /* 1E-2 */
   DS_PROFILE_3GPP_RESIDUAL_BER_5ENEG3        = 0x3,   /* 5E-3 */
   DS_PROFILE_3GPP_RESIDUAL_BER_4ENEG3        = 0x4,   /* 4E-3 */
   DS_PROFILE_3GPP_RESIDUAL_BER_1ENEG3        = 0x5,   /* 1E-3 */
   DS_PROFILE_3GPP_RESIDUAL_BER_1ENEG4        = 0x6,   /* 1E-4 */
   DS_PROFILE_3GPP_RESIDUAL_BER_1ENEG5        = 0x7,   /* 1E-5 */
   DS_PROFILE_3GPP_RESIDUAL_BER_1ENEG6        = 0x8,   /* 1E-6 */
   DS_PROFILE_3GPP_RESIDUAL_BER_6ENEG8        = 0x9,   /* 6E-8 */
                                              = 0xff
   DS_PROFILE_3GPP_RESIDUAL_BER_RESERVED
}  ds_profile_3gpp_residual_ber_ratio_type
```

QUALCOMM® 2016-05-06 10:13:38 PDT leon_zhang@sky.com.tw

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

3.2.1.9 ds_profile_3gpp_deliver_err_sdu_type

```
typedef enum {
   DS_PROFILE_3GPP_DELIVER_ERR_SDU_          = 0x0, /* Subscribed */
   SUBSCRIBE
   DS_PROFILE_3GPP_DELIVER_ERR_SDU_NO_       = 0x1, /* No detection */
   DETECT
     DS_PROFILE_3GPP_DELIVER_ERR_SDU_        = 0x2, /* Erroneous SDU is
   DELIVER                                   delivered  */
   DS_PROFILE_3GPP_DELIVER_ERR_SDU_NO_       = 0x3, /* Erroneous SDU not
   DELIVER                                   delivered */
     DS_PROFILE_3GPP_DELIVER_ERR_SDU_        = 0xff
   RESERVED
}  ds_profile_3gpp_deliver_err_sdu_type
```

3.2.1.10 ds_profile_3gpp_pdp_addr_type_ipv6

```
typedef PACKED struct PACKED_POST {
   PACKED union PACKED_POST {
   uint8                                     u6_addr8[16];
   uint16                                    u6_addr16[8];
   uint32                                    u6_addr32[4];
   uint64                                    u6_addr64[2];
   }in6_u;
}  ds_profile_3gpp_pdp_addr_type_ipv6
```

3.2.1.11 ds_profile_3gpp_pdp_addr_type

```
typedef PACKED struct PACKED_POST {
   PACKED struct PACKED_POST {
   uint32                                ds_profile_3gpp_pdp_addr_ipv4;
   ds_profile_3gpp_pdp_addr_type_ipv6    ds_profile_3gpp_pdp_addr_ipv6;
   }ds_profile_3gpp_pdp_addr;            u6_addr64[2];
   #define                               ds_profile_3gpp_pdp_addr.ds_
   ds_profile_3gpp_pdp_addr_ipv4         profile_3gpp_pdp_addr_ipv4
   #define                               ds_profile_3gpp_pdp_addr.ds_
   ds_profile_3gpp_pdp_addr_ipv6         profile_3gpp_pdp_addr_ipv6
}  ds_profile_3gpp_pdp_addr_type
```

QUALCOMM 2016-05-16 00:13:38 PDT deon_zhang@askey.com.tw

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

3.2.1.12 ds_profile_3gpp_pdp_header_comp_e_type

```
typedef PACKED enum {
   DS_PROFILE_3GPP_PDP_HEADER_COMP_OFF    = 0,    /* PDP header compression
                                          is OFF.      */
   DS_PROFILE_3GPP_PDP_HEADER_COMP_ON     = 1,    /* Manufacturer preferred
                                          compression. */
   DS_PROFILE_3GPP_PDP_HEADER_COMP_RFC1144       /* PDP header compression
                                                    based on rfc 1144.    */
   DS_PROFILE_3GPP_PDP_HEADER_COMP_RFC2507       /* PDP header compression
                                                    based on rfc 2507.    */
   DS_PROFILE_3GPP_PDP_HEADER_COMP_RFC3095       /* PDP header compression
                                                    based on rfc 3095.    */
   DS_PROFILE_3GPP_PDP_HEADER_COMP_MAX           =0xFF
}  PACKED_POST ds_profile_3gpp_pdp_header_comp_e_type
```

3.2.1.13 ds_profile_3gpp_pdp_data_comp_e_type

```
typedef PACKED enum {
   DS_PROFILE_3GPP_PDP_DATA_COMP_OFF      = 0,     /* PDP Data compression
                                          is OFF                          */
   DS_PROFILE_3GPP_PDP_DATA_COMP_ON       = 1,     /* Manufacturer preferred
                                          compression.                    */
   DS_PROFILE_3GPP_PDP_DATA_COMP_V42_BIS  /* V.42BIS data compression     */
   DS_PROFILE_3GPP_PDP_DATA_COMP_V44      /* V.44 data compression        */
   DS_PROFILE_3GPP_PDP_DATA_COMP_MAX      = 0xFF
}  PACKED_POST ds_profile_3gpp_pdp_data_comp_e_type
```

3.2.1.14 ds_profile_3gpp_pdp_ipv4_addr_alloc_e_type

```
typedef PACKED enum {
   DS_PROFILE_3GPP_PDP_IPV4_ADDR_ALLOC_NAS         = 0,    /* Addr alloc
                                                      using NAS           */
   DS_PROFILE_3GPP_PDP_IPV4_ADDR_ALLOC_DHCPV4      = 1,    /* Addr alloc
                                                      using DHCPv4        */
   DS_PROFILE_3GPP_PDP_IPV4_ADDR_ALLOC_MAX         = 0xFF
}  PACKED_POST
   ds_profile_3gpp_pdp_ipv4_addr_alloc_e_type
```


MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

3.2.1.15 ds_profile_3gpp_lte_qci_e_type

```
typedef PACKED enum {
    DS_PROFILE_3GPP_LTE_QCI_0                    = 0,
    DS_PROFILE_3GPP_LTE_QCI_1                    = 1,
    DS_PROFILE_3GPP_LTE_QCI_2                    = 2,
    DS_PROFILE_3GPP_LTE_QCI_3                    = 3,
    DS_PROFILE_3GPP_LTE_QCI_4                    = 4,
    DS_PROFILE_3GPP_LTE_QCI_5                    = 5,
    DS_PROFILE_3GPP_LTE_QCI_6                    = 6,
    DS_PROFILE_3GPP_LTE_QCI_7                    = 7,
    DS_PROFILE_3GPP_LTE_QCI_8                    = 8,
    DS_PROFILE_3GPP_LTE_QCI_9                    = 9,
    DS_PROFILE_3GPP_LTE_QCI_INVALID
    DS_PROFILE_3GPP_LTE_QCI_MAX                  = 0xFF
}   PACKED_POST ds_profile_3gpp_lte_qci_e_type
```

3.2.1.16 ds_profile_3gpp_pdp_auth_type

```
typedef PACKED struct PACKED_POST {
    ds_profile_3gpp_auth_pref_type      auth_type;
                                        /* Authentication type      */
    byte                                password[DS_PROFILE_3GPP_MAX_QCPDP_
                                        STRING_LEN+1]; /* Passw/secret string
                                        */
    byte                                username[DS_PROFILE_3GPP_MAX_QCPDP_
                                        STRING_LEN+1]; /* Username string
                                        */
}   ds_profile_3gpp_pdp_auth_type
```

QUALCOMM® 2016-05-26 10:13:38 PDT deon_zhang@askey.com.tw

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

3.2.1.17 ds_profile_3gpp_3gpp_qos_params_type

```
typedef PACKED struct PACKED_POST {
    ds_profile_3gpp_traffic_class_type          traffic_class;   /* Traffic
                                                Class         */
    uint32                                      max_ul_bitrate; /* Maximum
                                                UL bitrate    */
    uint32                                      max_dl_bitrate; /* Maximum
                                                DL bitrate    */
    uint32                                      gtd_ul_bitrate; /*
                                                Guaranteed UL bitrate */
    uint32                                      gtd_dl_bitrate; /*
                                                Guaranteed DL bitrate */
    ds_profile_3gpp_qos_delivery_order_type     dlvry_order;     /* SDU
                                                delivery order    */
    uint32                                      max_sdu_size;   /* Maximum
                                                SDU size      */
    ds_profile_3gpp_sdu_err_ratio_type          sdu_err;         /* SDU
                                                error ratio index */
    ds_profile_3gpp_residual_ber_ratio_type     res_biterr;      /* Residual
                                                bit err index*/
    ds_profile_3gpp_deliver_err_sdu_type        dlvr_err_sdu;    /* Delivery
                                                of err SDU   */
    uint32                                      trans_delay;    /* Transfer
                                                Delay        */
    uint32                                      thandle_prio;    /* Traffic
                                                handling prio */
    boolean                                     sig_ind;        /* Signaling
                                                Indication Flag    */
}   ds_profile_3gpp_3gpp_qos_params_type
```

3.2.1.18 ds_profile_3gpp_gprs_qos_params_type

```
typedef PACKED struct PACKED_POST {
    uint32                                      precedence;  /* Precedence
                                                class             */
    uint32                                      delay;       /* Delay class
                                                */
    uint32                                      reliability; /* Reliability
                                                class             */
    uint32                                      peak;        /* Peak
                                                throughput class        */
    uint32                                      mean;        /* Mean
                                                throughput class        */
}   ds_profile_3gpp_gprs_qos_params_type
```


MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

3.2.1.19 ds_profile_3gpp_lte_qos_params_type

```
typedef PACKED struct PACKED_POST {
    ds_profile_3gpp_lte_qci_e_type          qci; /* QCI Value */
    uint32                                  g_dl_bit_rate;   /* Guaranteed DL
                                            bit rate       */
    uint32                                  max_dl_bit_rate; /* Maximum DL
                                            bit rate          */
    uint32                                  g_ul_bit_rate;   /* Guaranteed UL
                                            bit rate       */
    uint32                                  max_ul_bit_rate; /* Maximum UL
                                            bit rate          */
}   ds_profile_3gpp_lte_qos_params_type
```

3.2.1.20 ds_profile_3gpp_dns_addresses_type

```
typedef PACKED struct PACKED_POST {
    ds_profile_3gpp_pdp_addr_type           primary_dns_addr;   /* primary
                                            DNS address   */
    ds_profile_3gpp_pdp_addr_type           secondary_dns_addr; /* secondary
                                            DNS address */
}   ds_profile_3gpp_dns_addresses_type
```

3.2.1.21 ds_profile_3gpp_tft_addr_type

```
typedef PACKED struct PACKED_POST {
    PACKED union PACKED_POST  {
    uint32                                      ds_profile_3gpp_tft_addr_
                                                ipv4;
    ds_profile_3gpp_pdp_addr_type_ipv6          ds_profile_3gpp_tft_addr_
                                                ipv6;
    }ds_profile_3gpp_tft_addr;
    #define ds_profile_3gpp_tft_addr_ipv4      ds_profile_3gpp_tft_addr.
                                                ds_profile_3gpp_tft_addr_
                                                ipv4
    #define ds_profile_3gpp_tft_addr_ipv6      ds_profile_3gpp_tft_addr.
                                                ds_profile_3gpp_tft_addr_
                                                ipv6
}   ds_profile_3gpp_tft_addr_type
```

QUALCOMM® 2016-05-16 00:13:38 PDT dedez.zhang@alkey.com.tw

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

3.2.1.22 ds_profile_3gpp_address_mask_type

```
typedef PACKED struct PACKED_POST {
    ds_profile_3gpp_tft_addr_type    address;  /* IPV4 or IPV6 Address        */
    uint8                            mask;     /* IPV4 or IPV6 Subnet mask    */
}   ds_profile_3gpp_address_mask_type
```

3.2.1.23 ds_profile_3gpp_port_range_type

```
typedef PACKED struct PACKED_POST {
    uint16                           from; /* Range lower limit */
    uint16                           to;   /* Range upper limit */
}   ds_profile_3gpp_port_range_type
```

3.2.1.24 ds_profile_3gpp_tft_params_type

```
typedef PACKED struct PACKED_POST {
    byte                                filter_id;       /* Filter identifier           */
    byte                                eval_prec_id;    /* Evaluation precedence index */
    ds_profile_3gpp_ip_version_enum_type ip_version;     /* IP version for address      */
    ds_profile_3gpp_address_mask_type   src_addr;   /* Source address & mask       */
    byte                                prot_num;      /* Protocol number => next_header in IPv6*/
    ds_profile_3gpp_port_range_type     dest_port_range; /* Destination port range      */
    ds_profile_3gpp_port_range_type     src_port_range;  /* Source port range           */
    uint32                              ipsec_spi;       /* Security parameter index    */
    uint16                              tos_mask;      /* Type of srvc & mask => tclass in  IPv6*/
}   ds_profile_3gpp_tft_params_type
```

QUALCOMM® 2016-05-16 03:13:38 PDT deon_zhang@askey.com.tw

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

3.2.1.25 Profile parameter identifiers

ds_profile_3gpp_param_enum_type

```
typedef enum {
    DS_PROFILE_3GPP_PROFILE_PARAM_INVALID             = 0,
    DS_PROFILE_3GPP_PROFILE_PARAM_MIN                 = 0x10,
    DS_PROFILE_3GPP_PROFILE_PARAM_PROFILE_NAME        = 0x10,
    DS_PROFILE_3GPP_PROFILE_PARAM_PDP_CONTEXT_        = 0x11,
    PDP_TYPE
    DS_PROFILE_3GPP_PROFILE_PARAM_PDP_CONTEXT_H_COMP  = 0x12,
    DS_PROFILE_3GPP_PROFILE_PARAM_PDP_CONTEXT_D_COMP  = 0x13,
    DS_PROFILE_3GPP_PROFILE_PARAM_PDP_CONTEXT_APN     = 0x14,
    DS_PROFILE_3GPP_PROFILE_PARAM_DNS_ADDR_V4_        = 0x15,
    PRIMARY
    DS_PROFILE_3GPP_PROFILE_PARAM_DNS_ADDR_V4_        = 0x16,
    SECONDARY
    /* Values 0x17 and 0x18 are reserved for
    internal use */
    DS_PROFILE_3GPP_PROFILE_PARAM_UMTS_REQ_QOS_EXTEN  = 0x17,
    DED
    DS_PROFILE_3GPP_PROFILE_PARAM_UMTS_MIN_QOS_EXTEN  = 0x18,
    DED
    DS_PROFILE_3GPP_PROFILE_PARAM_GPRS_REQ_QOS        = 0x19,
    DS_PROFILE_3GPP_PROFILE_PARAM_GPRS_MIN_QOS        = 0x1A,
    DS_PROFILE_3GPP_PROFILE_PARAM_AUTH_USERNAME       = 0x1B,
    DS_PROFILE_3GPP_PROFILE_PARAM_AUTH_PASSWORD       = 0x1C,
    DS_PROFILE_3GPP_PROFILE_PARAM_AUTH_TYPE           = 0x1D,
    DS_PROFILE_3GPP_PROFILE_PARAM_PDP_CONTEXT_PDP_    = 0x1E,
    ADDR_V4
    DS_PROFILE_3GPP_PROFILE_PARAM_PCSCF_REQ_FLAG      = 0x1F,
    DS_PROFILE_3GPP_PROFILE_PARAM_PDP_CONTEXT_        = 0x20,
    TE_MT_ACCESS_CTRL_FLAG
    DS_PROFILE_3GPP_PROFILE_PARAM_PCSCF_DHCP_         = 0x21,
    REQ_FLAG
    DS_PROFILE_3GPP_PROFILE_PARAM_IM_CN_FLAG          = 0x22,
    DS_PROFILE_3GPP_PROFILE_PARAM_TFT_FILTER_ID1      = 0x23,
    DS_PROFILE_3GPP_PROFILE_PARAM_TFT_FILTER_ID2      = 0x24,
    DS_PROFILE_3GPP_PROFILE_PARAM_PDP_CONTEXT_NUMBER  = 0x25,
    DS_PROFILE_3GPP_PROFILE_PARAM_PDP_CONTEXT_        = 0x26,
    SECONDARY_FLAG
    DS_PROFILE_3GPP_PROFILE_PARAM_PDP_CONTEXT_        = 0x27,
    PRIMARY_ID
    DS_PROFILE_3GPP_PROFILE_PARAM_PDP_CONTEXT_        = 0x28,
    PDP_ADDR_V6

    DS_PROFILE_3GPP_PROFILE_PARAM_UMTS_REQ_QOS        = 0x29,
    DS_PROFILE_3GPP_PROFILE_PARAM_UMTS_MIN_QOS        = 0x2A,
    DS_PROFILE_3GPP_PROFILE_PARAM_DNS_ADDR_           = 0x2B,
    V6_PRIMARY
```


MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

```
    DS_PROFILE_3GPP_PROFILE_PARAM_DNS_ADDR_          = 0x2C,
    V6_SECONDARY
    DS_PROFILE_3GPP_PROFILE_PARAM_IPV4_ADDR_ALLOC    = 0x2D,
    DS_PROFILE_3GPP_PROFILE_PARAM_LTE_REQ_QOS        = 0x2E,
    DS_PROFILE_3GPP_PROFILE_PARAM_LINGER_PARAMS      = 0x2F,
    DS_PROFILE_3GPP_PROFILE_PARAM_INACTIVITY_TIMER_  = 0x30,
    VAL
    DS_PROFILE_3GPP_PROFILE_PARAM_APN_CLASS          = 0x31,
    DS_PROFILE_3GPP_PROFILE_PARAM_LINGER_PARAMS      = 0x32,
    DS_PROFILE_3GPP_PROFILE_PARAM_SRC_STAT_DESC_REQ  = 0x33,
    DS_PROFILE_3GPP_PROFILE_PARAM_SRC_STAT_DESC_MIN  = 0x34,
    DS_PROFILE_3GPP_PROFILE_PARAM_APN_BEARER         = 0x35,
    DS_PROFILE_3GPP_PROFILE_PARAM_EMERGENCY_CALLS_   = 0x36,
    SUPPORTED
    DS_PROFILE_3GPP_PROFILE_PARAM_OPERATOR_RESERVED_ = 0x37,
    PCO
    DS_PROFILE_3GPP_PROFILE_PARAM_MCC                = 0x38,
    DS_PROFILE_3GPP_PROFILE_PARAM_MNC                = 0x39,
    DS_PROFILE_3GPP_PROFILE_MAX_PDN_CONN_PER_BLOCK   = 0x3A,
    DS_PROFILE_3GPP_PROFILE_MAX_PDN_CONN_TIMER       = 0x3B,
    DS_PROFILE_3GPP_PROFILE_PDN_REQ_WAIT_TIMER       = 0x3C,
    DS_PROFILE_3GPP_PROFILE_USER_APP_DATA            = 0x3D,
    DS_PROFILE_3GPP_PROFILE_PARAM_ROAMING_DISALLOWED = 0x3E,
    DS_PROFILE_3GPP_PROFILE_PARAM_PDN_DISCON_WAIT_   = 0x3F,
    TIME
    /*Internal use only*/
    DS_PROFILE_3GPP_PROFILE_PARAM_SUBS_ID            = 0x40,

    DS_PROFILE_3GPP_PROFILE_PARAM_MAX                =
                                                     DS_PROFILE_3GPP_PROFILE
                                                     _PARAM_SUBS_ID

}   ds_profile_3gpp_param_enum_type
```

3.2.2 CDMA1x interface specification

ds_profile_3gpp2_pdn_type_enum_type

```
typedef enum {
    DS_PROFILE_3GPP2_PDN_TYPE_V4          = 0,
    DS_PROFILE_3GPP2_PDN_TYPE_V6          = 1,
    DS_PROFILE_3GPP2_PDN_TYPE_V4_V6       = 2,
    DS_PROFILE_3GPP2_PDN_TYPE_UNSPEC      = 3,
    DS_PROFILE_3GPP2_PDN_TYPE_MAX         = 0xFF
}   ds_profile_3gpp2_pdn_type_enum_type
```


MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

ds_profile_3gpp2_rat_type_enum_type

```
typedef enum {
    DS_PROFILE_3GPP2_RAT_TYPE_HRPD          = 0,
    DS_PROFILE_3GPP2_RAT_TYPE_EHRPD         = 1,
    DS_PROFILE_3GPP2_RAT_TYPE_HRPD_EHRPD    = 2,
    DS_PROFILE_3GPP2_RAT_TYPE_MAX           = 0xFF
}   ds_profile_3gpp2_rat_type_enum_type
```

ds_profile_3gpp2_in_addr_type

```
typedef struct ds_profile_3gpp2_in_addr {
    uint32                                  ds_profile_3gpp2_s_addr;
}   ds_profile_3gpp2_in_addr_type
```

ds_profile_3gpp2_in6_addr_type

```
typedef struct ds_profile_3gpp2_in6_addr {
    union {
    uint8                                   ds_profile_3gpp2_u6_addr8[16];
    uint16                                  ds_profile_3gpp2_u6_addr16[8];
    uint32                                  ds_profile_3gpp2_u6_addr32[4];
    uint64                                  ds_profile_3gpp2_u6_addr64[2];
    } ds_profile_3gpp2_in6_u;
    #define ds_profile_3gpp2_s6_addr        ds_profile_3gpp2_in6_u.ds_
                                            profile_3gpp2_u6_addr8
    #define ds_profile_3gpp2_s6_addr16      ds_profile_3gpp2_in6_u.ds_
                                            profile_3gpp2_u6_addr16
    #define ds_profile_3gpp2_s6_addr32      ds_profile_3gpp2_in6_u.ds_
                                            profile_3gpp2_u6_addr32
    #define ds_profile_3gpp2_s6_addr64      ds_profile_3gpp2_in6_u.ds_
                                            profile_3gpp2_u6_addr64
}   ds_profile_3gpp2_in6_addr_type
```

QUALCOMM®
2016-05-16 00:13:35 PDT
dennis_zhang@askey.com.tw

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

3.2.2.1 Profile parameter identifiers

```
typedef enum {
    DS_PROFILE_3GPP2_PROFILE_PARAM_INVALID                 = 0x0,
    DS_PROFILE_3GPP2_PROFILE_PARAM_MIN                     = 0x10,
    DS_PROFILE_3GPP2_PROFILE_PARAM_NEGOTIATE_DNS_SERVER    = 0x10,
                                                           = 0x11,
    DS_PROFILE_3GPP2_PROFILE_PARAM_SESSION_CLOSE_TIMER_DO
                                                           = 0x12,
    DS_PROFILE_3GPP2_PROFILE_PARAM_SESSION_CLOSE_TIMER_1X
    DS_PROFILE_3GPP2_PROFILE_PARAM_ALLOW_LINGER            = 0x13,
    DS_PROFILE_3GPP2_PROFILE_PARAM_LCP_ACK_TIMEOUT         = 0x14,
    DS_PROFILE_3GPP2_PROFILE_PARAM_IPCP_ACK_TIMEOUT        = 0x15,
    DS_PROFILE_3GPP2_PROFILE_PARAM_AUTH_TIMEOUT            = 0x16,
    DS_PROFILE_3GPP2_PROFILE_PARAM_LCP_CREQ_RETRY_COUNT    = 0x17,
                                                           = 0x18,
    DS_PROFILE_3GPP2_PROFILE_PARAM_IPCP_CREQ_RETRY_COUNT
    DS_PROFILE_3GPP2_PROFILE_PARAM_AUTH_RETRY_COUNT        = 0x19,
    DS_PROFILE_3GPP2_PROFILE_PARAM_AUTH_PROTOCOL           = 0x1A,
    DS_PROFILE_3GPP2_PROFILE_PARAM_USER_ID                 = 0x1B,
    DS_PROFILE_3GPP2_PROFILE_PARAM_AUTH_PASSWORD           = 0x1C,
    DS_PROFILE_3GPP2_PROFILE_PARAM_DATA_RATE               = 0x1D,
    DS_PROFILE_3GPP2_PROFILE_PARAM_DATA_MODE               = 0x1F,
    DS_PROFILE_3GPP2_PROFILE_PARAM_APP_TYPE                = 0x1E,
    DS_PROFILE_3GPP2_PROFILE_PARAM_APP_PRIORITY            = 0x20,
    DS_PROFILE_3GPP2_PROFILE_PARAM_APN_STRING              = 0x21,
    DS_PROFILE_3GPP2_PROFILE_PARAM_PDN_TYPE                = 0x22,
    DS_PROFILE_3GPP2_PROFILE_PARAM_IS_PCSCF_ADDR_NEEDED    = 0x23,
    DS_PROFILE_3GPP2_PROFILE_PARAM_V4_DNS_ADDR_PRIMARY     = 0x24,
                                                           = 0x25,
    DS_PROFILE_3GPP2_PROFILE_PARAM_V4_DNS_ADDR_SECONDARY
    DS_PROFILE_3GPP2_PROFILE_PARAM_V6_DNS_ADDR_PRIMARY     = 0x26,
                                                           = 0x27,
    DS_PROFILE_3GPP2_PROFILE_PARAM_V6_DNS_ADDR_SECONDARY
    DS_PROFILE_3GPP2_PROFILE_PARAM_RAT_TYPE                = 0x28,
    DS_PROFILE_3GPP2_PROFILE_PARAM_LINGER_PARAMS           = 0x29,
    DS_PROFILE_3GPP2_PROFILE_PARAM_APN_ENABLED             = 0x2A,
    DS_PROFILE_3GPP2_PROFILE_PARAM_PDN_INACTIVITY_TIMEOUT  = 0x2B,
    DS_PROFILE_3GPP2_PROFILE_PARAM_LINGER_PARAMS           = 0x2C,
    DS_PROFILE_3GPP2_PROFILE_PARAM_PDN_LEVEL_AUTH_         = 0x2D,
    PROTOCOL
    DS_PROFILE_3GPP2_PROFILE_PARAM_PDN_LEVEL_USER_ID       = 0x2E,
    DS_PROFILE_3GPP2_PROFILE_PARAM_PDN_LEVEL_AUTH_         = 0x2F,
    PASSWORD
    DS_PROFILE_3GPP2_PROFILE_PARAM_PDN_LABEL               = 0x30,
    DS_PROFILE_3GPP2_PROFILE_PARAM_FAILURE_TIMER_1         = 0x31,
    DS_PROFILE_3GPP2_PROFILE_PARAM_FAILURE_TIMER_2         = 0x32,
    DS_PROFILE_3GPP2_PROFILE_PARAM_FAILURE_TIMER_3         = 0x33,
    DS_PROFILE_3GPP2_PROFILE_PARAM_FAILURE_TIMER_4         = 0x34,
```


MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

```
    DS_PROFILE_3GPP2_PROFILE_PARAM_FAILURE_TIMER_5          = 0x35,
    DS_PROFILE_3GPP2_PROFILE_PARAM_FAILURE_TIMER_6          = 0x36,
    DS_PROFILE_3GPP2_PROFILE_PARAM_DISALLOW_TIMER_1         = 0x37,
    DS_PROFILE_3GPP2_PROFILE_PARAM_DISALLOW_TIMER_2         = 0x38,
    DS_PROFILE_3GPP2_PROFILE_PARAM_DISALLOW_TIMER_3         = 0x39,
    DS_PROFILE_3GPP2_PROFILE_PARAM_DISALLOW_TIMER_4         = 0x3A,
    DS_PROFILE_3GPP2_PROFILE_PARAM_DISALLOW_TIMER_5         = 0x3B,
    DS_PROFILE_3GPP2_PROFILE_PARAM_DISALLOW_TIMER_6         = 0x3C,
    DS_PROFILE_3GPP2_PROFILE_PARAM_OP_PCO_ID                = 0x3D,
    DS_PROFILE_3GPP2_PROFILE_PARAM_MCC                      = 0x3E,
    DS_PROFILE_3GPP2_PROFILE_PARAM_MNC                      = 0x3F,
    DS_PROFILE_3GPP2_PROFILE_PARAM_FAILURE_TIMERS           = 0x40,
    DS_PROFILE_3GPP2_PROFILE_PARAM_DISALLOW_TIMERS          = 0x41,
    DS_PROFILE_3GPP2_PROFILE_PARAM_USER_APP_DATA            = 0x42,
    DS_PROFILE_3GPP2_PROFILE_PARAM_MAX                      =
                                                            DS_PROFILE_3GPP2_P
                                                            ROFILE_PARAM_USER_
                                                            APP_DATA

    ds_profile_3gpp2_param_enum_type
}
```

QUALCOMM®
2016-05-16 00:13:38 PDT
deon_zhang@askey.com.tw

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

3.2.3 Common interface specification

3.2.3.1 Operation result codes

```
DS_PROFILE_REG_RESULT_SUCCESS    =       0,  /* Successful operation       */
DS_PROFILE_REG_RESULT_FAIL,                  /* General failure in the lib */

DS_PROFILE_REG_RESULT_ERR_INVAL_HNDL,        /* Invalid profile handle     */
DS_PROFILE_REG_RESULT_ERR_INVAL_OP,          /* Operation not supported    */

DS_PROFILE_REG_RESULT_ERR_INVAL_PROFILE_TYPE,/* Invalid tech type          */
DS_PROFILE_REG_RESULT_ERR_INVAL_PROFILE_NUM, /* Invalid profile number     */
DS_PROFILE_REG_RESULT_ERR_INVAL_IDENT,       /* Invalid identifier         */
DS_PROFILE_REG_RESULT_ERR_INVAL,             /* other invalid arg          */

DS_PROFILE_REG_RESULT_ERR_LIB_NOT_INITED,    /* lib not initialized        */

DS_PROFILE_REG_RESULT_ERR_LEN_INVALID,       /* for get_param, buff size
                                                cannot be less than max
                                                for set_param buff size
                                                cannot be greater than max */

DS_PROFILE_REG_RESULT_LIST_END,              /* End of list reached, return
                                                value for _itr_next        */

DS_PROFILE_REG_RESULT_ERR_INVALID_SUBS_ID,   /* Invalid subscription ID    */
DS_PROFILE_REG_INVAL_PROFILE_FAMILY,         /* Invalid profile family     */
DS_PROFILE_REG_PROFILE_VERSION_MISMATCH,     /**< Profile Version Mismatch */
DS_PROFILE_REG_RESULT_ERR_OUT_OF_MEMORY,     /**< Out of memory */
DS_PROFILE_REG_RESULT_ERR_FILE_ACCESS,       /**< Error accessing the embedded file
                                                system */
DS_PROFILE_REG_RESULT_ERR_EOF,               /**< Error end of file. */
DS_PROFILE_REG_RESULT_ERR_VALID_FLAG_NOT_SET,/**< Profile valid flag is not set.  */
DS_PROFILE_REG_ERR_OUT_OF_PROFILES,          /**< No profiles are available while
                                                creating a new profile. */
DS_PROFILE_REG_NO_EMERGENCY_PDN_SUPPORT,     /**< Emergency PDN Feature disabled */

DS_PROFILE_REG_NOT_SUPPORTED,                /**< Operation is not supported */

DS_PROFILE_REG_3GPP_SPEC_MIN = 0x1000,       /* Offset for 3GPP Tech
                                                specific errors            */
DS_PROFILE_REG_3GPP_INVAL_PROFILE_FAMILY,
DS_PROFILE_REG_3GPP_ACCESS_ERR,
DS_PROFILE_REG_3GPP_CONTEXT_NOT_DEFINED,
DS_PROFILE_REG_3GPP_VALID_FLAG_NOT_SET,
DS_PROFILE_REG_3GPP_READ_ONLY_FLAG_SET,
DS_PROFILE_REG_3GPP_SPEC_MAX = 0x10FF,

DS_PROFILE_REG_3GPP2_SPEC_MIN = 0x1100,       /* Offset for 3GPP2 Tech
                                                specific errors            */
```


MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

```
  DS_PROFILE_REG_3GPP2_ERR_INVALID_IDENT_FOR_PROFILE, /* To specify that
                                                         identifier is not valid for
                                                         the profile               */
  DS_PROFILE_REG_3GPP2_SPEC_MAX = 0x11FF,

  DS_PROFILE_REG_RESULT_MAX    = 0xFFFF
```

QUALCOMM®
2016-05-16 00:13:38 PDT
deon_zhang@askey.com.tw

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

3.2.3.2 Common data types

#define DS_PROFILE_EMBEDDED_PROFILE_FAMILY 0

#define DS_PROFILE_TETHERED_PROFILE_FAMILY 1

```
typedef void*                ds_profile_hndl_type
typedef uint16               ds_profile_num_type
typedef uint32               ds_profile_identifier_type
typedef void*                ds_profile_itr_type
```

ds_profile_info_type;

```
typedef struct {
    uint32                              len
    void*                               buf
}   ds_profile_info_type;
```

ds_profile_tech_e_type

```
typedef enum {
    DS_PROFILE_TECH_MIN
    DS_PROFILE_TECH_3GPP
    DS_PROFILE_TECH_3GPP2
    DS_PROFILE_TECH_COMMON
    DS_PROFILE_TECH_MAX
}   ds_profile_tech_e_type
```

ds_profile_trn_e_type

```
typedef enum {
    DS_PROFILE_TRN_MIN
    DS_PROFILE_TRN_READ
    DS_PROFILE_TRN_RW
    DS_PROFILE_TRN_MAX
}   ds_profile_trn_e_type
```

QUALCOMM®
2016-05-16 00:13:38 PDT
deon_zhang@askey.com.tw

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

ds_profile_action_e_type

```
typedef enum {
    DS_PROFILE_ACTION_COMMIT
    DS_PROFILE_ACTION_CANCEL
}   ds_profile_action_e_type
```

ds_profile_subs_e_type

```
typedef enum {
    DS_PROFILE_ACTIVE_SUBSCRIPTION_NONE =
    0x00
    DS_PROFILE_ACTIVE_SUBSCRIPTION_1
    DS_PROFILE_ACTIVE_SUBSCRIPTION_2
    DS_PROFILE_ACTIVE_SUBSCRIPTION_MAX  =
    0xff
}    ds_profile_subs_e_type
```

ds_profile_linger_params_type

```
typedef PACKED struct PACKED_POST {
    boolean                              allow_linger_flag
    uint8                                linger_timeout_val
}   ds_profile_linger_params_type
```

ds_profile_config_type

```
typedef struct
{
  ds_profile_config_mask_type           config_mask;
  boolean                               is_persistent;
  sys_modem_as_id_e_type                subs_id;
  ds_profile_num_type                   num;
} ds_profile_config_type;
```

QUALCOMM®
2016-05-16 00:13:38 PDT
deon_zhang@askey.com.tw

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

3.2.3.3 List service data types

The following data types are defined in list.h.

list_link_type

```
typedef struct list_link_struct {
    struct list_link_struct *               next_ptr
}   list_link_type;
```

list_type

```
typedef struct {
    list_link_type *                        front_ptr
    list_link_type *                        back_ptr
    list_size_type                          size
}   list_type;
```

3.2.3.4 Iterator service data types

ds_profile_list_dfn_etype

```
typedef enum {
    GET_ALL_PROFILES
    SEARCH_PROFILES
}   ds_profile_list_dfn_etype
```

ds_profile_list_type

```
typedef struct {
    ds_profile_list_dfn_etype               dfn
    ds_profile_identifier_type              ident
    ds_profile_info_type                    info
}   ds_profile_list_type;
```

ds_profile_list_info_type

```
typedef struct {
    ds_profile_num_type                     num
    ds_profile_info_type *                  name
}   ds_profile_list_info_type;
```

QUALCOMM® 2016-05-16 00:13:38 PDT deon_zhang@askey.com.tw

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

3.2.3.5 Callback

NOTE: This section was added to this document revision.

ds_profile_event_etype

```
typedef enum
{
  DS_PROFILE_INVALID_EVENT             =   0,
  DS_PROFILE_CREATE_PROFILE_EVENT      =   1,
  DS_PROFILE_DELETE_PROFILE_EVENT      =   2,
  DS_PROFILE_MODIFY_PROFILE_EVENT      =   3,
  DS_PROFILE_RESET_PROFILE_EVENT       =   5,
  DS_PROFILE_SUBSCRIPTION_CHANGE_EVENT =   6,
  DS_PROFILE_MAX_EVENT                 =   0xff
} ds_profile_event_etype;
```

ds_profile_cb_type

```
typedef void (*ds_profile_cb_type) (
  ds_profile_event_etype               event,
  unsigned int                         profile_count,
  ds_profile_changes_info*             profiles_changes,
  void*                                user_data
);
```

QUALCOMM®
2016-05-16 00:13:38 PDT
deon_zhang@askey.com.tw

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

4 New API Declaration

4.1 DS_PROFILE_INIT_LIB

This function initializes the DS profile software library. On modem processor and AMSS-AMSS this function is called during initialization. On application processor this routine must be called by every client of this library which is being executed on a unique process domain.

```
ds_profile_status_etype ds_profile_init_lib (
)
```

Return value

This function returns:

- DS_PROFILE_REG_RESULT_SUCCESS – On successful operation
- DS_PROFILE_REG_RESULT_FAIL – On general errors. This return code provides blanket coverage

Caveats on API use

- None

QMI message REQ/RSP

- None

QUALCOMM
2016-05-16 00:13:16 PDT
deon_zhang@askey.com.tw

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

4.2 DS_PROFILE_BEGIN_TRANSACTION

This function returns a handle that the clients of this software library can use for subsequent profile operations. The handle returned is of requested transaction type. Currently, handles of only one type (READ or READ+WRITE) are supported. All profile operations using this handle require that DS_PROFILE_END_TRANSACTION be called at the end. In dual processor architecture a copy of profile is fetched across processor boundaries for every open handle. In single processor architecture a local copy of profile is maintained in library for every open handle.

```
ds_profile_status_etype
ds_profile_begin_transaction (
→  ds_profile_trn_etype                  trn
→  ds_profile_tech_etype                 tech
→  ds_profile_num_type                   num
←  ds_profile_hndl_type *                hndl
)
```

Return value

This function returns:

- DS_PROFILE_REG_RESULT_SUCCESS – On successful operation
- DS_PROFILE_REG_RESULT_ERR_INVALID_PROFILE_NUM – In case of profile number specified being invalid
- DS_PROFILE_REG_RESULT_ERR_INVALID_PROFILE_TYPE – In case of profile type specified being invalid
- DS_PROFILE_REG_RESULT_ERR_LIB_NOT_INITED – When library was not initialized
- DS_PROFILE_REG_RESULT_FAIL – On general errors

Caveats on API use

- None

QMI message REQ/RSP

- QMI_WDS_GET_PROFILE_SETTINGS

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

4.3 DS_PROFILE_BEGIN_TRANSACTION_PER_SUB

NOTE: This section was added to this document revision.

This function returns a handle that the clients of this software library can use for subsequent profile operations for a particular subscription. The handle returned is of requested transaction type. Currently, handles of only one type (READ or READ+WRITE) are supported. All profile operations using this handle require that DS_PROFILE_END_TRANSACTION be called at the end. In dual processor architecture a copy of profile is fetched across processor boundaries for every open handle. In single processor architecture a local copy of profile is maintained in library for every open handle.

```
ds_profile_status_etype
ds_profile_begin_transaction_per_sub (
→   ds_profile_trn_etype                trn
→   ds_profile_tech_etype               tech
→   ds_profile_num_type                 num
→   ds_profile_subs_etype               subs_id
←   ds_profile_hndl_type *              hndl
)
```

Return value

This function returns:

- DS_PROFILE_REG_RESULT_SUCCESS – On successful operation
- DS_PROFILE_REG_RESULT_ERR_INVALID_PROFILE_NUM – In case of profile number specified being invalid
- DS_PROFILE_REG_RESULT_ERR_INVALID_PROFILE_TYPE – In case of profile type specified being invalid
- DS_PROFILE_REG_RESULT_ERR_LIB_NOT_INITED – When library was not initialized
- DS_PROFILE_REG_RESULT_FAIL – On general errors

Caveats on API use

- None

QMI message REQ/RSP

- QMI_WDS_QUERY_PROFILE_SETTINGS

QUALCOMM 2012-05-16 00:13:38 PDT

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

4.4 DS_PROFILE_SET_PARAM

Clients can use this routine to set identified Profile parameters. Input function parameter info points to the identified data element and its length. The prefetched Profile is modified and requires that END_TRANSACTION be called at the end.

```
ds_profile_status_etype
ds_profile_set_param (
→   ds_profile_hndl_type                  profile_hndl
→   ds_profile_identifier_type            identifier
→   ds_profile_info_type *                info
)
```

Return value

This function returns:

- DS_PROFILE_REG_RESULT_SUCCESS on successful operation
- DS_PROFILE_REG_RESULT_FAIL on general errors. This return code provides blanket coverage.
- DS_PROFILE_REG_ERR_INVAL_HNDL
- DS_PROFILE_REG_ERR_INVAL_IDENT
- DS_PROFILE_REG_ERR_INVAL_LEN
- DS_PROFILE_REG_ERR_INVAL_PROFILE_TYPE
- DS_PROFILE_REG_ERR_INVAL_PROFILE_NUM

Caveats on API use

- Client does memory management
- END_TRANSACTION needs to be called to actually write the Profile on Modem

QMI message REQ/RSP

- None

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

4.5 DS_PROFILE_GET_PARAM

This routine is used to get Profile fields identified by the identifier. The data elements are read from the prefetched Profile and info is returned with value and length.

```
ds_profile_status_etype
ds_profile_get_param (
→  ds_profile_hndl_type                  profile_hndl
→  ds_profile_identifier_type            identifier
←  ds_profile_info_type *                info
)
```

Return value

This function returns:

- DS_PROFILE_REG_RESULT_SUCCESS on successful operation
- DS_PROFILE_REG_RESULT_FAIL on general errors. This return code provides blanket coverage.
- DS_PROFILE_REG_ERR_INVAL_HNDL
- DS_PROFILE_REG_ERR_INVAL_IDENT
- DS_PROFILE_REG_ERR_INVAL_LEN
- DS_PROFILE_REG_ERR_INVAL_PROFILE_TYPE
- DS_PROFILE_REG_ERR_INVAL_PROFILE_NUM

Caveats on API use

- Client does memory management

QMI message REQ/RSP

- None

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

4.6 DS_PROFILE_END_TRANSACTION

This routine commits the pre-fetched modified Profile to the persistent storage of the Modem. This routine also invokes the cleanup routines for the Profile Handle specified. On return the Handle becomes unusable. The act parameter specifies the action that the client has requested.

```
ds_profile_status_etype
ds_profile_end_transaction (
→  ds_profile_hndl_type                    hndl
→  ds_profile_action_etype                 act
)
```

Return value

This function returns:

- DS_PROFILE_REG_RESULT_SUCCESS – On successful operation
- DS_PROFILE_REG_RESULT_ERR_LIB_NOT_INITED
- DS_PROFILE_REG_RESULT_ERR_INVAL_HNDL
- DS_PROFILE_REG_RESULT_FAIL – On general errors. This return code provides blanket coverage.

Caveats on API use

- None

QMI message REQ/RSP

- QMI_WDS_MODIFY_PROFILE_SETTINGS

QUALCOMM®
2016-05-16 00:13:08 PDT
leon_zhang@askey.com.tw

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

4.7 DS_PROFILE_RESET_PARAM_TO_INVALID

This routine resets profile fields identified by the identifier. The profile field is reset to default values and marked as invalid. This operation is currently allowed only on UMTS QoS and TFT container data structures.

```
ds_profile_status_etype
ds_profile_reset_param (
→   ds_profile_tech_etype                  tech
→   ds_profile_num_type                    num
→   ds_profile_identifier_type             ident
)
```

Return value

This function returns:

- DS_PROFILE_REG_RESULT_SUCCESS – On successful operation
- DS_PROFILE_REG_RESULT_INVALID_OP – If operation is not allowed
- DS_PROFILE_REG_RESULT_FAIL – On general errors. This return code provides blanket coverage
- DS_PROFILE_REG_ERR_INVAL_PROFILE_TYPE
- DS_PROFILE_REG_ERR_INVAL_PROFILE_NUM
- DS_PROFILE_REG_ERR_INVAL_OP

Caveats on API use

- None

QMI message REQ/RSP

- New message is required

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

4.8 DS_PROFILE_RESET_PARAM_TO_INVALID_PER_SUB

NOTE: This section was added to this document revision.

This routine resets profile fields identified by the identifier. This function should be used to have reset profile fields for a particular subscription. The profile field is reset to default values and marked as invalid. This operation is currently allowed only on UMTS QoS and TFT container data structures.

```
ds_profile_status_etype
ds_profile_reset_param_per_sub (
→   ds_profile_tech_etype                  tech
→   ds_profile_num_type                    num
→   ds_profile_identifier_type             ident
→   ds_profile_subs_etype                  subs_id
)
```

Return value

This function returns:

- DS_PROFILE_REG_RESULT_SUCCESS – On successful operation
- DS_PROFILE_REG_RESULT_INVALID_OP – If operation is not allowed
- DS_PROFILE_REG_RESULT_FAIL – On general errors. This return code provides blanket coverage
- DS_PROFILE_REG_ERR_INVAL_PROFILE_TYPE
- DS_PROFILE_REG_ERR_INVAL_PROFILE_NUM
- DS_PROFILE_REG_ERR_INVAL_OP

Caveats on API use

- None

QMI message REQ/RSP

- New message is required

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

4.9 DS_PROFILE_CREATE

This function creates a profile on the modem EFS. All profile parameters are initialized. The profile number starting from 100 is returned.

```
ds_profile_status_etype ds_profile_create
(
→  ds_profile_tech_etype                  tech
←  ds_profile_num_type *                  num
)
```

Return value

This function returns:

- DS_PROFILE_REG_RESULT_SUCCESS – On successful operation
- DS_PROFILE_REG_RESULT_ERR_INVALID_PROFILE_TYPE – In case of profile type specified being invalid
- DS_PROFILE_REG_RESULT_FAIL – On general errors. This return code provides blanket coverage
- DS_PROFILE_REG_ERR_INVAL_PROFILE_TYPE
- DS_PROFILE_REG_ERR_INVAL_PROFILE_NUM
- DS_PROFILE_REG_ERR_INVAL_OP

Caveats on API use

- None

QMI message REQ/RSP

- QMI_WDS_CREATE_PROFILE

QUALCOMM® 2016-05-16 10:13:38 PDT deon_zhang@asus.com.tw

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

4.10 DS_PROFILE_CREATE_EX

NOTE: This section was added to this document revision.

This function creates a profile on the modem EFS. All profile parameters are initialized.

```
ds_profile_status_etype
ds_profile_create_ex (
→   ds_profile_tech_etype                  tech,
→   ds_profile_config_type                 config_ptr,
←   ds_profile_num_type *                  Num
)
```

Return value

This function returns:

- DS_PROFILE_REG_RESULT_SUCCESS – On successful operation
- DS_PROFILE_REG_RESULT_ERR_INVALID_PROFILE_TYPE – In case of profile type specified being invalid
- DS_PROFILE_REG_RESULT_FAIL – On general errors. This return code provides blanket coverage
- DS_PROFILE_REG_ERR_INVAL_PROFILE_TYPE
- DS_PROFILE_REG_ERR_INVAL_PROFILE_NUM
- DS_PROFILE_REG_ERR_INVAL_OP

Caveats on API use

- None

QMI message REQ/RSP

- QMI_WDS_CREATE_PROFILE

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

4.11 DS_PROFILE_RESET_PROFILE_TO_DEFAULT

This function resets a profile to default on modem EFS. All profile parameters are reset to default values. For UMTS this resets the profile to IPv4 type.

```
ds_profile_status_etype
ds_profile_reset_profile_to_default (
→  ds_profile_tech_etype                    tech
→  ds_profile_num_type                      num
)
```

Return value

This function returns:

- DS_PROFILE_REG_RESULT_SUCCESS – On successful operation
- DS_PROFILE_REG_RESULT_ERR_INVALID_PROFILE_TYPE – In case of profile tech type specified being invalid
- DS_PROFILE_REG_RESULT_FAIL – On general errors. This return code provides blanket coverage

Caveats on API use

- None

QMI message REQ/RSP

- New QMI message is needed
- New QMI MSG Lib API is needed for dual processor solutions

QUALCOMM®
2016-05-16 00:13:38 PDT
leon_zhang@askey.com.tw

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

4.12 DS_PROFILE_RESET_PROFILE_TO_DEFAULT_PER_SUB

NOTE: This section was added to this document revision.

This function resets a profile to default on the modem EFS for a particular subscription. All profile parameters are reset to default values. For UMTS, this resets the profile to IPv4 type.

```
ds_profile_status_etype
ds_profile_reset_profile_to_default_per_sub
(
→  ds_profile_tech_etype                    tech
→  ds_profile_num_type                      num
→  ds_profile_subs_etype                    subs_id
)
```

Return value

This function returns:

- DS_PROFILE_REG_RESULT_SUCCESS – On successful operation
- DS_PROFILE_REG_RESULT_ERR_INVALID_PROFILE_TYPE – In case of profile tech type specified being invalid
- DS_PROFILE_REG_RESULT_FAIL – On general errors. This return code provides blanket coverage

Caveats on API use

- None

QMI message REQ/RSP

- New QMI message is needed
- New QMI MSG Lib API is needed for dual processor solutions

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

4.13 DS_PROFILE_SET_DEFAULT_PROFILE_NUM

This function sets a specified profile as default for the specified technology and family.

```
ds_profile_status_etype
ds_profile_set_default_profile_num (
→   ds_profile_tech_etype                  tech
→   uint32                                 family
→   ds_profile_num_type                    num
)
```

Return value

This function returns:

- DS_PROFILE_REG_RESULT_SUCCESS – On successful operation
- DS_PROFILE_REG_RESULT_ERR_INVALID_PROFILE_TYPE – In case of profile tech type specified being invalid
- DS_PROFILE_REG_RESULT_FAIL – On general errors. This return code provides blanket coverage

Caveats on API use

- None

QMI message REQ/RSP

- New QMI message is needed
- New QMI MSG Lib API is needed for dual processor solutions

QUALCOMM®
2016-05-16 00:13:38 PDT
ardeun_zhang@askey.com.tw

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

4.14 DS_PROFILE_GET_DEFAULT_PROFILE_NUM

This function gets the default profile for the specified technology and family.

```
ds_profile_status_etype
ds_profile_get_default_profile_num (
→  ds_profile_tech_etype                  tech
→  uint32                                 family
←  ds_profile_num_type *                  num
)
```

Return value

This function returns:

- DS_PROFILE_REG_RESULT_SUCCESS – On successful operation
- DS_PROFILE_REG_RESULT_ERR_INVALID_PROFILE_TYPE – In case of profile tech type specified being invalid
- DS_PROFILE_REG_RESULT_FAIL – On general errors. This return code provides blanket coverage

Caveats on API use

- None

QMI message REQ/RSP

- New QMI message is needed
- New QMI MSG Lib API is needed for dual processor solutions

QUALCOMM® 2016-05-16 00:13:38 PDT ardeun_zhang@askey.com.tw

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

4.15 DS_PROFILE_SET_DEFAULT_PROFILE_NUM_PER_SUBS

This API sets a specified profile as default for a particular subscription, technology, and family.

```
ds_profile_status_etype
ds_profile_set_default_profile_num_per_sub
s (
→  ds_profile_tech_e_type                  Tech
→  uint32                                  family
→  ds_profile_subs_e_type                  subs_id
→  ds_profile_num_type                     Num
)
```

Return value

This function returns:

- DS_PROFILE_REG_RESULT_SUCCESS – On successful operation
- DS_PROFILE_REG_RESULT_ERR_INVALID_PROFILE_TYPE – In case of profile tech type specified being invalid
- DS_PROFILE_REG_RESULT_FAIL – On general errors. This return code provides blanket coverage
- DS_PROFILE_REG_RESULT_ERR_INVALID_SUBS_ID – In case of invalid subscription id.

Caveats on API use

- None

QMI message REQ/RSP

- New QMI message is needed
- New QMI MSG Lib API is needed for dual processor solutions

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

4.16 DS_PROFILE_GET_DEFAULT_PROFILE_NUM_PER_SUBS

This API retrieves the profile number set for a particular subscription, technology, and family.

```
ds_profile_status_etype
ds_profile_get_default_profile_num_per_subs
(
→  ds_profile_tech_e_type                  tech
→  uint32                                  family
→  ds_profile_subs_e_type                  subs_id
←  ds_profile_num_type *                   num
)
```

Return value

This function returns:

- DS_PROFILE_REG_RESULT_SUCCESS – On successful operation
- DS_PROFILE_REG_RESULT_ERR_INVALID_PROFILE_TYPE – In case of profile tech type specified being invalid
- DS_PROFILE_REG_RESULT_FAIL – On general errors. This return code provides blanket coverage
- DS_PROFILE_REG_RESULT_ERR_INVALID_SUBS_ID – In case of invalid subscription ID.

Caveats on API use

- None

QMI message REQ/RSP

- New QMI message is needed
- New QMI MSG Lib API is needed for dual processor solutions

QUALCOMM®
2016-05-16 09:13:32 PDT
deon_zhang@asky.com.cn

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

4.17 DS_PROFILE_DELETE

This function deletes a configured profile on the modem. Default profiles can also be deleted.

```
ds_profile_status_etype ds_profile_delete
(
→  ds_profile_tech_etype                    tech
→  ds_profile_num_type                      num
)
```

Return value

This function returns:

- DS_PROFILE_REG_RESULT_SUCCESS – On successful operation
- DS_PROFILE_REG_RESULT_ERR_INVALID_PROFILE_TYPE – In case of profile type specified being invalid
- DS_PROFILE_REG_RESULT_FAIL – On general errors. This return code provides blanket coverage.

Caveats on API use

- None

QMI message REQ/RSP

- QMI_WDS_DELETE_PROFILE

QUALCOMM®
2011-05-16 00:13:38 PDT
deon_zhang@askey.com.tw

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

4.18 DS_PROFILE_DELETE_PER_SUB

NOTE: This section was added to this document revision.

This function deletes a configured profile on the modem belonging to a particular subscription.

```
ds_profile_status_etype
ds_profile_delete_per_sub (
→   ds_profile_tech_etype                  tech
→   ds_profile_num_type                    num
→   ds_profile_subs_etype                  subs_id
)
```

Return value

This function returns:

- DS_PROFILE_REG_RESULT_SUCCESS – On successful operation
- DS_PROFILE_REG_RESULT_ERR_INVALID_PROFILE_TYPE – In case of profile type specified being invalid
- DS_PROFILE_REG_RESULT_FAIL – On general errors. This return code provides blanket coverage.

Caveats on API use

- None

QMI message REQ/RSP

- QMI_WDS_DELETE_PROFILE

QUALCOMM® 2016-05-16 00:13:38 PDT deon_zhang@askey.com.tw

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

4.19 DS_PROFILE_GET_LIST_ITR

This function gets an iterator of a list. The input list type determines the operation to be performed. Currently supported operations are enumerated as list definitions in ds_profile_list_dfn_etype. The list definition is technology-agnostic and defined in common header. Definition specifies the information to be fetched. The return information is always a list of nodes where each node has the tuple [Profile Number, Profile Name]. This function returns an Iterator to that list. The Iterator is passed as an argument to DS_PROFILE_ITR_NEXT and DS_PROFILE_ITR_FIRST. Clients can extract information from Iterator using DS_PROFILE_GET_INFO_BY_ITR. After traversal is complete, the client is expected to call DS_PROFILE_ITR_DESTROY.

```
ds_profile_status_etype
ds_profile_get_list_itr (
→   ds_profile_tech_etype                Tech
→   ds_profile_list_type *               List
←   ds_profile_itr_type *                Itr
)
```

Return value

This function returns:

- DS_PROFILE_REG_RESULT_SUCCESS – On successful operation
- DS_PROFILE_REG_RESULT_FAIL – On general errors. This return code provides blanket coverage.

Caveats on API use

- None

QMI message REQ/RSP

- QMI_WDS_GET_PROFILE_LIST

QUALCOMM 2016-04-10 20:13:38 PDT deon_zhang@asus.com.tw

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

4.20 DS_PROFILE_GET_LIST_ITR_PER_SUB

NOTE: This section was added to this document revision.

This function gets an Iterator of a list of profiles belonging to a particular subscription. The input list type determines the operation to be performed. Currently supported operations are enumerated as list definitions in ds_profile_list_dfn_etype. The list definition is technology-agnostic and defined in the common header. The definition specifies the information to be fetched. The return information is always a list of nodes where each node has the tuple [Profile Number, Profile Name]. This function returns an Iterator to that list. The Iterator is passed as an argument to DS_PROFILE_ITR_NEXT and DS_PROFILE_ITR_FIRST. Clients can extract information from the Iterator using DS_PROFILE_GET_INFO_BY_ITR. After traversal is complete, the client is expected to call DS_PROFILE_ITR_DESTROY.

```
ds_profile_status_etype
ds_profile_get_list_itr_per_sub (
→  ds_profile_tech_etype                  Tech
→  ds_profile_list_type *                 List
←  ds_profile_itr_type *                  Itr
→  ds_profile_subs_etype                  subs_id
)
```

Return value

This function returns:

- DS_PROFILE_REG_RESULT_SUCCESS – On successful operation
- DS_PROFILE_REG_RESULT_FAIL – On general errors. This return code provides blanket coverage.

Caveats on API use

- None

QMI message REQ/RSP

- QMI_WDS_GET_PROFILE_LIST

QUALCOMM 2016-05-16 00:13:38 PDT cjon.chang@askey.com.tw

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

4.21 DS_PROFILE_GET_INFO_BY_ITR

This function gets a pointer to a blob of information. The information is again technology specific and depends upon the [Tech type, definition] used in the DS_PROFILE_GET_LIST_ITR routine.

```
ds_profile_status_etype ds_profile_itr_curr (
→  ds_profile_itr_type                          itr
←  ds_profile_list_info_type *                  info
)
```

Return value

This function returns:

- DS_PROFILE_REG_REG_RESULT_SUCCESS – On successful operation
- DS_PROFILE_REG_REG_RESULT_FAIL – On general errors. This return code provides blanket coverage

Caveats on API use

- None

QMI message REQ/RSP

- None

QUALCOMM®
2016-05-16 00:13:38 PDT
deon_zhang@askey.com.tw

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

4.22 DS_PROFILE_ITR_NEXT

This routine advances the iterator to the next element.

```
ds_profile_status_etype
ds_profile_itr_next (
→  ds_profile_itr_type                    itr
)
```

Return value

This function returns:

- DS_PROFILE_REG_REG_RESULT_SUCCESS – On successful operation
- DS_PROFILE_REG_REG_RESULT_FAIL – On general errors. This return code provides blanket coverage

Caveats on API use

- None

QMI message REQ/RSP

- None

QUALCOMM® 2016-05-16 00:13:38 PDT deon_zhang@askey.com.tw

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

4.23 DS_PROFILE_ITR_FIRST

This routine resets the iterator to the beginning of the list.

```
ds_profile_status_etype ds_profile_itr_first (
→  ds_profile_itr_type                          itr
)
```

Return value

This function returns:

- DS_PROFILE_RESULT_SUCCESS – On successful operation
- DS_PROFILE_RESULT_FAIL – On general errors. This return code provides blanket coverage

Caveats on API use

- None

QMI message REQ/RSP

- None

QUALCOMM®
2016-05-16 00:13:38 PDT
deon_zhang@askey.com.tw

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

4.24 DS_PROFILE_ITR_DESTROY

This routine destroys the iterator.

```
ds_profile_status_etype ds_profile_itr_destroy (
→  ds_profile_itr_type *                          itr
)
```

Return value

This function returns:

- DSPRF_RESULT_SUCCESS – On successful operation
- DSPRF_RESULT_FAIL – On general errors. This return code provides blanket coverage

Caveats on API use

- None

QMI message REQ/RSP

- None

QUALCOMM®
2016-05-16 00:13:38 PDT
deon_zhang@askey.com.tw

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

4.25 DS_PROFILE_GET_MAX_NUM

This function returns the maximum number of profiles possible for a given technology type.

```
ds_profile_status_etype
ds_profile_get_max_num (
→   ds_profile_tech_etype                  tech
←   uint32 *                              max_num_profiles
)
```

Return value

This function returns:

- DS_PROFILE_REG_RESULT_SUCCESS – On successful operation
- DS_PROFILE_REG_RESULT_ERR_INVALID_PROFILE_TYPE – In case of profile type specified being invalid
- DS_PROFILE_REG_RESULT_FAIL – On general errors. This return code provides blanket coverage

Caveats on API use

- None

QMI message REQ/RSP

- None

QUALCOMM®
2016-05-16 00:13:38 PDT
deon_zhang@askey.com.tw

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

4.26 DS_PROFILE_GET_SUPPORTED_TYPE

This function returns the number of Tech Types supported and an array of values for the same. The returned array is indexed [0, num – 1]

```
ds_profile_status_etype
ds_profile_get_supported_type (
  ←  uint32                                *num
  ←  ds_profile_tech_type                  *tech
)
```

Return value

This function returns:

- DS_PROFILE_REG_RESULT_SUCCESS – On successful operation
- DS_PROFILE_REG_RESULT_ERR_INVALID_PROFILE_TYPE – In case of profile type specified being invalid
- DS_PROFILE_REG_RESULT_FAIL – On general errors. This return code provides blanket coverage

QUALCOMM®
2016-05-16 00:13:38 PDT
deon_zhang@askey.com.tw

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

4.27 DS_PROFILE_CLOSE_LIB

This function is used to close the DS PROFILE software library. It closes all open handles and all non-committed data are discarded.

```
ds_profile_status_etype ds_profile_init_lib (
)
```

Return value

This function returns:

- DS_PROFILE_REG_REG_RESULT_SUCCESS – On successful operation
- DS_PROFILE_REG_REG_RESULT_FAIL – On general errors. This return code provides blanket coverage

Caveats on API use

- None

QUALCOMM®
2016-05-16 00:13:38 PDT
deon_zhang@askey.com.tw

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

4.28 DS_PROFILE_GET_PARAM_IN_USE

NOTE: This section was added to this document revision.

This routine is used to get the Profile fields that will be used during call bring-up, identified by the Common Tech identifier. The data elements are read from the prefetched profile, and information is returned with that value and length. The prefetched profile should be of type EPC.

```
ds_profile_status_etype
ds_profile_get_param_in_use (
→   ds_profile_hndl_type                *num,
→   ds_profile_identifier_type          Identifier,
←   ds_profile_info_type                *info,
→   ds_profile_tech_etype               tech_type
)
```

Return value

This function returns:

- DS_PROFILE_REG_REG_RESULT_SUCCESS – On successful operation
- DS_PROFILE_REG_REG_RESULT_FAIL – On general errors. This return code provides blanket coverage
- DS_PROFILE_REG_ERR_INVAL_HNDL
- DS_PROFILE_REG_ERR_INVAL_IDENT
- DS_PROFILE_REG_ERR_INVAL_LEN
- DS_PROFILE_REG_ERR_INVAL_PROFILE_TYPE
- DS_PROFILE_REG_ERR_INVAL_PROFILE_NUM

Caveats on API use

- None

QMI message REQ/RSP

- None

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

4.29 DS_PROFILE_GET_PARAM_IN_USE_PER_SUB

NOTE: This section was added to this document revision.

This routine is used to get the Profile fields that will be used during call bring-up, identified by the Common Tech identifier for a given subscription. The data elements are read from the prefetched profile, and information is returned with that value and length. The prefetched profile should be of type EPC.

```
ds_profile_status_etype
ds_profile_get_param_in_use_per_sub (
→  ds_profile_hndl_type                  *num,
→  ds_profile_identifier_type            identifier,
←  ds_profile_info_type                  *info,
→  ds_profile_tech_etype                 tech_type
→  ds_profile_subs_etype                 subs_id
)
```

Return value

This function returns:

- DS_PROFILE_REG_REG_RESULT_SUCCESS – On successful operation
- DS_PROFILE_REG_REG_RESULT_FAIL – On general errors. This return code provides blanket coverage
- DS_PROFILE_REG_ERR_INVAL_HNDL
- DS_PROFILE_REG_ERR_INVAL_IDENT
- DS_PROFILE_REG_ERR_INVAL_LEN
- DS_PROFILE_REG_ERR_INVAL_PROFILE_TYPE
- DS_PROFILE_REG_ERR_INVAL_PROFILE_NUM

Caveats on API use

- None

QMI message REQ/RSP

- None

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

4.30 DS_PROFILE_GET_PERSISTENCE_FROM_PROFILE_NUM

NOTE: This section was added to this document revision.

This routine is used to check if the profile exists in the persistent storage of the modem. The profile is identified by the profile number specified by the client.

```
ds_profile_status_etype
ds_profile_get_persistence_from_profile_num
(
→   ds_profile_num_type                    profile_num
←   boolean                                *is_persistent
)
```

Return value

This function returns:

- DS_PROFILE_REG_REG_RESULT_SUCCESS – On successful operation
- DS_PROFILE_REG_REG_RESULT_FAIL – On general errors. This return code provides blanket coverage
- DS_PROFILE_REG_RESULT_ERR_OUT_OF_MEMORY
- DS_PROFILE_REG_ERR_INVAL_PROFILE_NUM

Caveats on API use

- Client does memory management

QMI message REQ/RSP

- None

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

4.31 DS_PROFILE_GET_PERSISTENCE_FROM_PROFILE_NUM_PER_SUB

NOTE: This section was added to this document revision.

This routine is used to check if the profile exists in the persistent storage of the modem for a given subscription. The profile is identified by the profile number specified by the client.

```
ds_profile_status_etype
ds_profile_get_persistence_from_profile_num
_per_sub (
→   ds_profile_num_type                    profile_num
←   boolean                                *is_persistent
→   ds_profile_subs_etype                  subs_id
)
```

Return value

This function returns:

- DS_PROFILE_REG_REG_RESULT_SUCCESS – On successful operation
- DS_PROFILE_REG_REG_RESULT_FAIL – On general errors. This return code provides blanket coverage
- DS_PROFILE_REG_RESULT_ERR_OUT_OF_MEMORY
- DS_PROFILE_REG_ERR_INVAL_PROFILE_NUM

Caveats on API use

- Client does memory management

QMI message REQ/RSP

- None

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

4.32 DS_PROFILE_GET_TECH_TYPE_FROM_PROFILE_NUM

NOTE: This section was added to this document revision.

This routine is used to get the technology type of the profile specified by the profile number. Invalid Tech Type is returned if an invalid profile number is passed.

```
ds_profile_status_etype
ds_profile_get_tech_type_from_profile_num
(
→  ds_profile_num_type                    profile_num

)
```

Return value

This function returns:

- DS_PROFILE_TECH_3GPP if the profile is of 3GPP technology
- DS_PROFILE_TECH_3GPP2 if the profile is of 3GPP2 technology
- DS_PROFILE_TECH_EPC if the profile is of EPC technology
- DS_PROFILE_TECH_INVALID if the profile number is invalid

Caveats on API use

- None

QMI message REQ/RSP

- None

QUALCOMM® 2016-05-16 00:13:39 PDT deon_zhang@askey.com.tw

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

4.33 DS_PROFILE_GET_TECH_TYPE_FROM_PROFILE_NUM_PER_SUB

NOTE: This section was added to this document revision.

This routine is used to get the technology type of the profile specified by the profile number. Invalid Tech Type is returned if an invalid profile number is passed.

```
ds_profile_status_etype
ds_profile_get_tech_type_from_profile_num_
per_sub (
→   ds_profile_num_type                    profile_num
→   ds_profile_subs_etype                  subs_id
)
```

Return value

This function returns:

- DS_PROFILE_TECH_3GPP if the profile is of 3GPP technology
- DS_PROFILE_TECH_3GPP2 if the profile is of 3GPP2 technology
- DS_PROFILE_TECH_EPC if the profile is of EPC technology
- DS_PROFILE_TECH_INVALID if the profile number is invalid

Caveats on API use

- None

QMI message REQ/RSP

- None

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

4.34 DS_PROFILE_UPDATE_LTE_ATTACH_PDN_LIST_ PROFILES

NOTE: This section was added to this document revision.

This routine is used to update the profile parameters in the LTE Attach PDN list of the specified technology. Not all technology types support this operation.

```
ds_profile_status_etype
ds_profile_update_lte_attach_pdn_list_prof
iles
(
→  ds_profile_tech_etype                    tech
)
```

Return value

This function returns:

- DS_PROFILE_REG_RESULT_SUCCESS on successful operation.
- DS_PROFILE_REG_RESULT_FAIL on general errors. This return code provides blanket coverage.
- DS_PROFILE_REG_RESULT_ERR_INVAL_PROFILE_TYPE
- DS_PROFILE_REG_RESULT_ERR_INVAL_OP

Caveats on API use

- None

QMI message REQ/RSP

- QMI_WDS_UPDATE_LTE_ATTACH_PDN_LIST_PROFILES

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

4.35 DS_PROFILE_UPDATE_LTE_ATTACH_PDN_LIST_ PROFILES_PER_SUBS

NOTE: This section was added to this document revision.

This routine is used to update the profile parameters in the LTE Attach PDN list of the specified technology and specified subscription. Not all technology types support this operation.

```
ds_profile_status_etype
ds_profile_update_lte_attach_pdn_list_prof
iles_per_subs
(
→  ds_profile_tech_etype                    tech
→  ds_profile_subs_etype                    subs_id
)
```

Return value

This function returns:

- DS_PROFILE_REG_RESULT_SUCCESS on successful operation.
- DS_PROFILE_REG_RESULT_FAIL on general errors. This return code provides blanket coverage.
- DS_PROFILE_REG_RESULT_ERR_INVAL_PROFILE_TYPE
- DS_PROFILE_REG_RESULT_ERR_INVAL_OP

Caveats on API use

- None

QMI message REQ/RSP

- QMI_WDS_UPDATE_LTE_ATTACH_PDN_LIST_PROFILES

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

4.36 DS_PROFILE_UNREGISTER_CALLBACK

NOTE: This section was added to this document revision.

This routine is used to de register the clients from getting the callback notifications.

```
void
ds_profile_unregister_callback (
→  ds_profile_cb_handle_type                cb_handle
)
```

Return value

This function returns:

- DS_PROFILE_REG_RESULT_SUCCESS on successful operation.
- DS_PROFILE_REG_RESULT_FAIL on general errors. This return code provides blanket coverage.
- DS_PROFILE_REG_RESULT_ERR_LIB_NOT_INITED
- DS_PROFILE_REG_RESULT_ERR_INVAL_OP
- DS_PROFILE_REG_RESULT_ERR_INVAL

Caveats on API use

- None

QMI message REQ/RSP

- QMI_WDS_CONFIGURE_PROFILE_EVENT_LIST

QUALCOMM® 2016-05-16 01:13:38 PDT deon_zhang@asker.com.tw

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

4.37 DS_PROFILE_UNREGISTER_CALLBACK_PER_SUB

NOTE: This section was added to this document revision.

This routine is used to de register the clients from getting the callback notifications for a given subscription.

```
void
ds_profile_unregister_callback_per_sub (
→  ds_profile_cb_handle_type                cb_handle
→  ds_profile_subs_etype                     subs_id
)
```

Return value

This function returns:

- DS_PROFILE_REG_RESULT_SUCCESS on successful operation.
- DS_PROFILE_REG_RESULT_FAIL on general errors. This return code provides blanket coverage.
- DS_PROFILE_REG_RESULT_ERR_LIB_NOT_INITED
- DS_PROFILE_REG_RESULT_ERR_INVAL_OP
- DS_PROFILE_REG_RESULT_ERR_INVAL

Caveats on API use

- None

QMI message REQ/RSP

- QMI_WDS_CONFIGURE_PROFILE_EVENT_LIST

QUALCOMM® 2016-05-16 01:38 PDT deon_zhang@sky.com.tw

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

4.38 DS_PROFILE_REGISTER_CALLBACK

NOTE: This section was added to this document revision.

This routine is used to register for notifications triggered by changes in profiles. Clients can monitor either a specific profile or all profiles of a particular technology.

```
void
ds_profile_register_callback (
→  ds_profile_num_type                  num,
→  ds_profile_tech_etype                tech,
→  ds_profile_cb_type                   cback,
←  void*                                user_data
)
```

Return value

This function returns valid cb handle in case of success, 0 in case of failure.

Caveats on API use

- None

QMI message REQ/RSP

- QMI_WDS_CONFIGURE_PROFILE_EVENT_LIST

QUALCOMM® 2016-05-16 00:13:38 PDT deon_zhang@askey.com.tw

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

4.39 DS_PROFILE_REGISTER_CALLBACK_PER_SUB

NOTE: This section was added to this document revision.

This routine is used to register for notifications triggered by changes in profiles for a given subscription. Clients can monitor either a specific profile or all profiles of a particular technology.

```
void
ds_profile_register_callback_per_sub (
→  ds_profile_num_type                    num,
→  ds_profile_tech_etype                  tech,
→  ds_profile_cb_type                     cback,
←  void*                                  user_data,
→  ds_profile_subs_etype                  subs_id
)
```

Return value

This function returns valid cb handle in case of success, 0 in case of failure.

Caveats on API use

- None

QMI message REQ/RSP

- QMI_WDS_CONFIGURE_PROFILE_EVENT_LIST

QUALCOMM®
2015-05-16 00:13:38 PDT
deon_zhang@askey.com.tw

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

5 Architecture of the DS Profile Registry Library

This chapter describes the internals related to the operation of the Profile Registry Library. It addresses the following topics:

- Internal structure of the Profile Registry Library
- Interaction between the various internal modules

5.1 Internal structure

The internal structure of the Profile Registry Library consists of the following distinct entities:

- Profile Registry Operation Module (PRF) which implements the public interface. This module is platform and OS independent.
- Platform Manager (PLM) is aware of all Profile Tech Types supported. Currently, Profile Registry is physically located on the modem file-system. One of the objectives of this new Profile Registry API is to provide seamless access to Profile store irrespective of whether there exists a processor boundary or not. Thus PLM also provides abstraction for all communication between PRF and physical storage of Profile Registry.
- Tech Specific Operation (TSO) manages all technology specific operation. A TSO is required for every technology supported and is platform independent.
- Tech Specific Access (TSA) controls access to technology specific Profile Registry on the persistent storage. This module is platform aware and implements access methods to physical storage of Profile Registry.

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

5.2 Internal working

This section lists the possible module interactions.

Figure 5-1 illustrates the interaction between various modules.

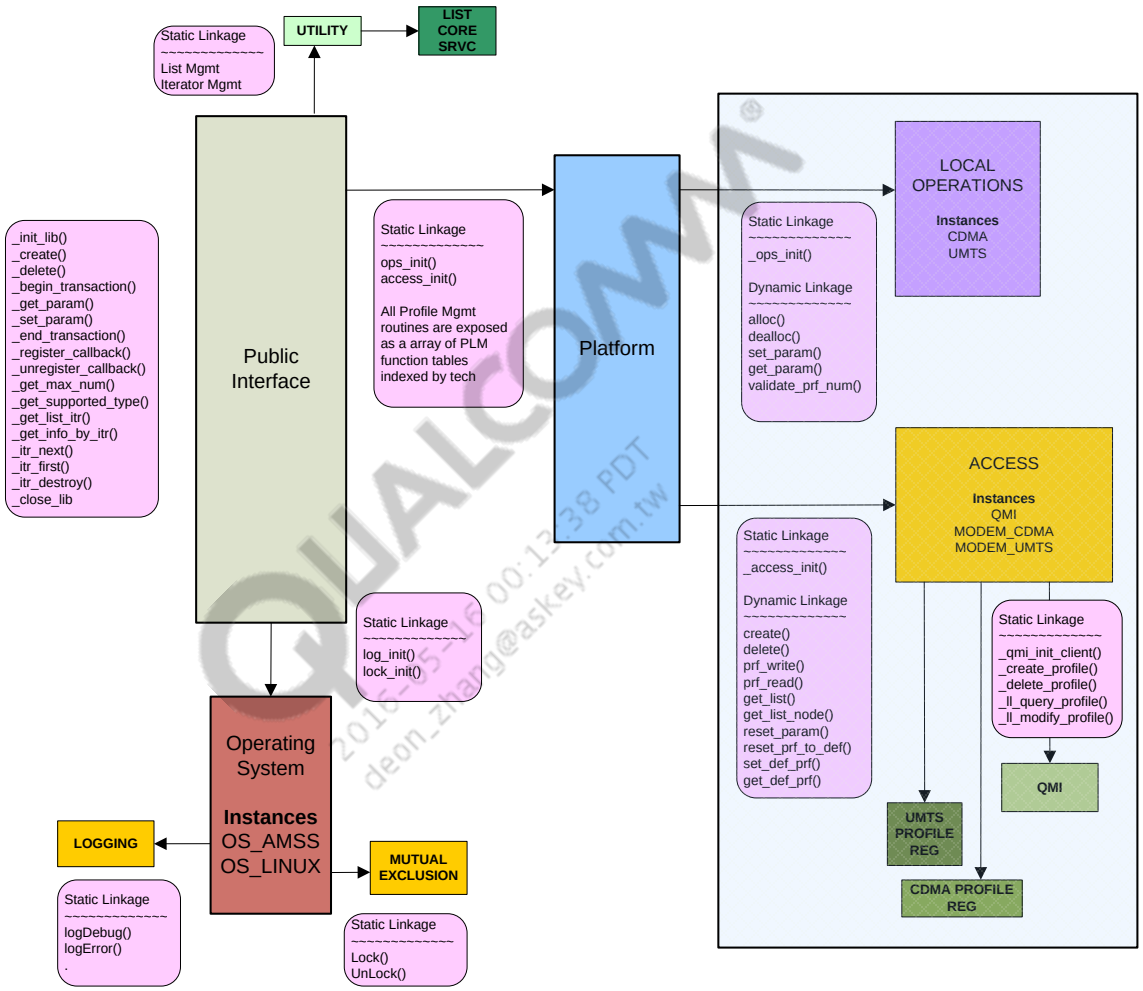

Figure 5-1 Module interaction

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

5.2.1 Overall use case

Figure 5-2 shows an overall use case of the profile registry.

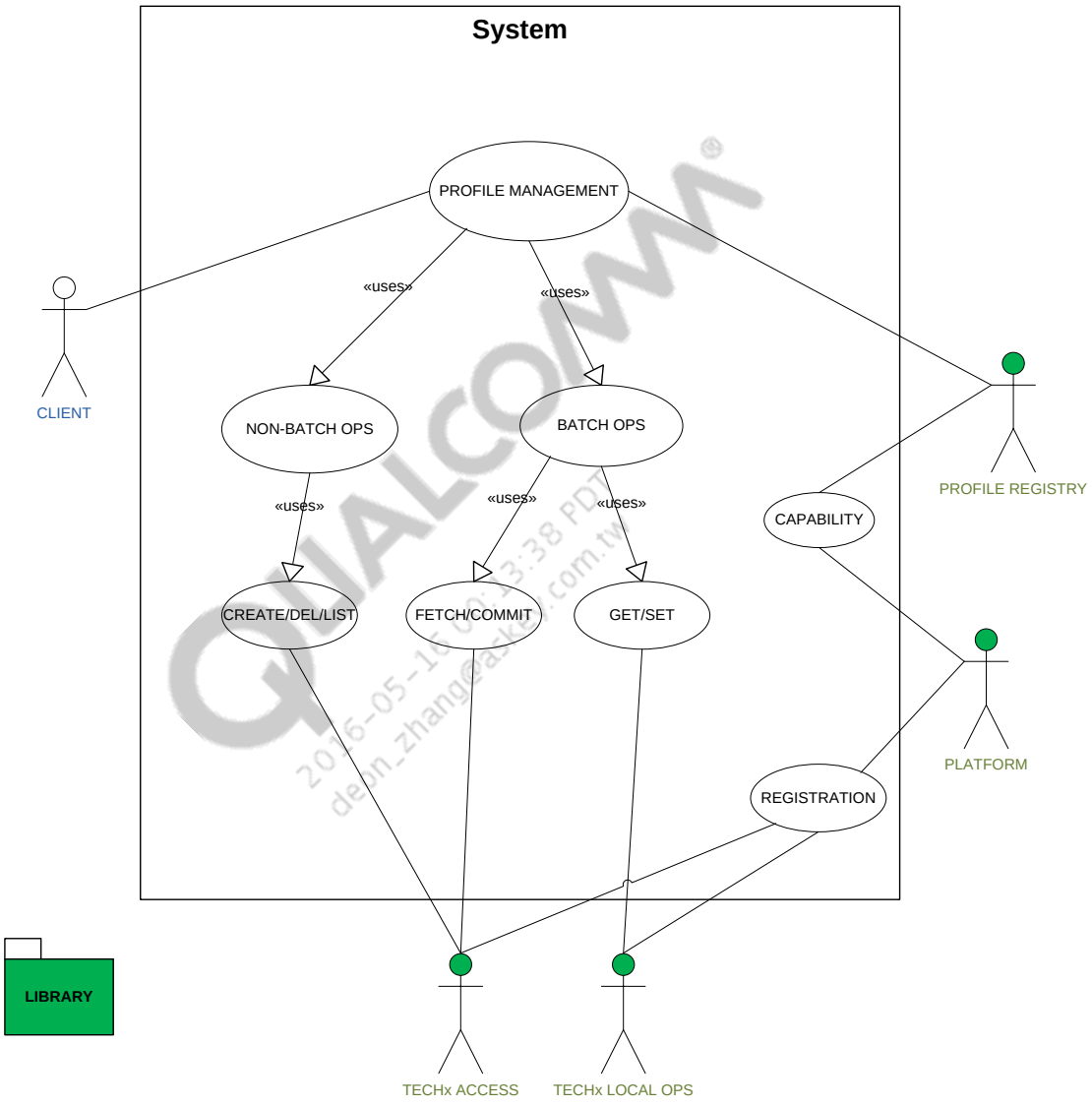

Figure 5-2 Overall use case

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

5.2.2 Typical use-case in Linux-AMSS

Figure 5-3 shows a typical use case of the DS Profile Library.

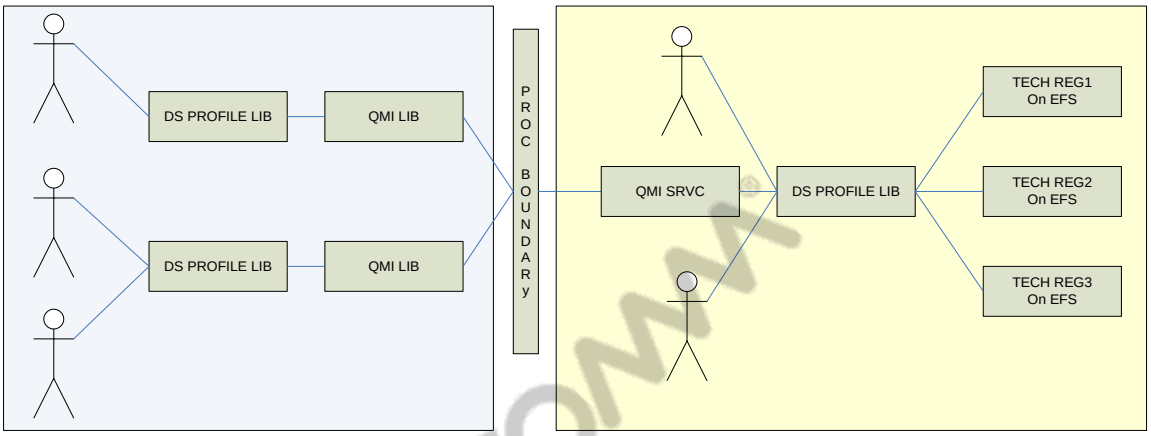

Figure 5-3 Typical use case of the DS Profile Library

5.2.3 Typical use-case in AMSS-AMSS

Figure 5-4 shows a typical use case in dual processor AMSS targets.

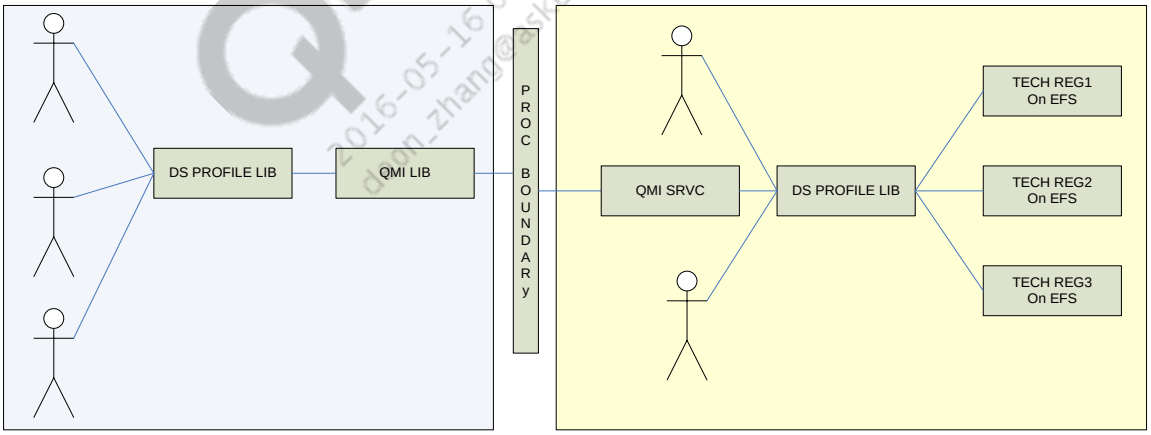

Figure 5-4 Typical use case in dual processor AMSS targets

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

5.2.4 Typical use-case in AMSS

Figure 5-5 shows a typical use case in single processor AMSS targets.

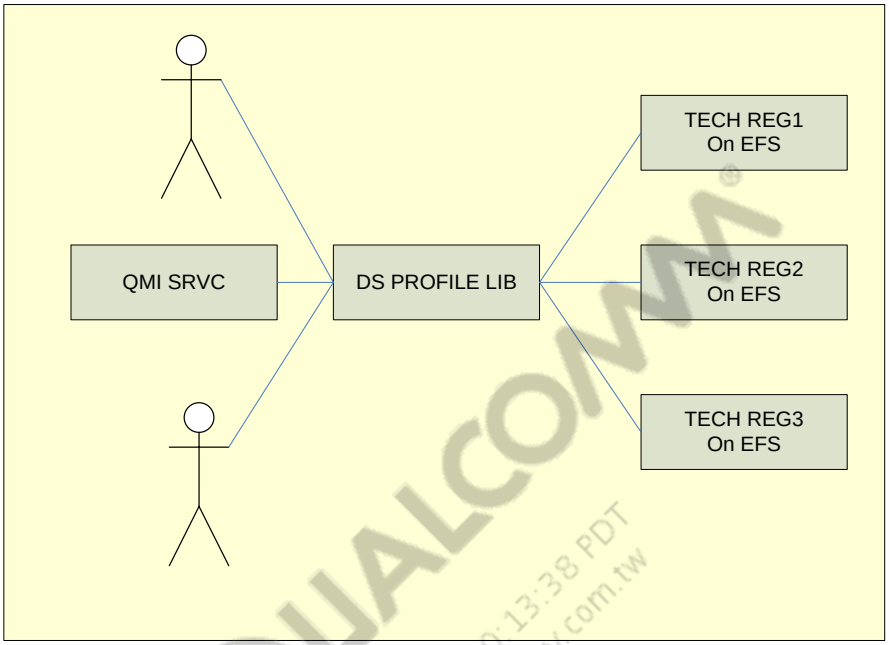

Figure 5-5 Typical use case in single processor AMSS targets

QUALCOMM
2016-05-16 00:13:38 PDT
deon_zhang@sky.com.tw

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

6 Usage of DS PROFILE APIs

NOTE: This chapter was added to this document revision.

6.1 DS PROFILE GET/SET PARAM

This module provides an example of how prefetched profile parameters can be modified or read.

6.1.1 Set Parameters

For setting a required parameter or identifier in the prefetched profile, a transaction must be created to get the handle of the profile. Later, the paramete information with the parameter value is provided as a part of Set Parameter. To commit these changes to persistent storage, the transaction must be ended.

```
ds_profile_hndl_type    profile_hndl =
NULL;
ds_profile_identifier_type ident;
ds_profile_info_type       info_write;
ds_profile_tech_etype   profile_type =
DS_PROFILE_TECH_3GPP;
ds_profile_status_etype status      =
DS_PROFILE_REG_RESULT_FAIL;
ds_profile_action_etype act         =
DS_PROFILE_ACTION_COMMIT;
char *apn_name                      =
“test_apn”
ident                               =
DS_PROFILE_3GPP_PROFILE_PARAM_PDP_CONTEXT_
APN;
act                                 =
DS_PROFILE_ACTION_COMMIT;
status                              =
ds_profile_begin_transaction( trn_type,
profile_type,
profile_num,
&profile_hndl );

/*Check for status before continuing*/
info_write.buf                      =
apn_name;
info_write.len                      =
strlen(apn_name);
status                              =
ds_profile_set_param(profile_hndl,
ident, &info_write);
/*Check for status before continuing*/
status                              =
ds_profile_end_transaction(profile_hndl,
act);
```


MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

6.1.2 Get Parameters

For getting a required parameter or identifier in the prefetched profile, a transaction must be created to get the handle of the profile. Later, the parameter buffer in which the parameter value is to be stored is provided by the client as a part of Get Parameter, followed by End Transaction.

```
ds_profile_hndl_type    profile_hndl =
NULL;
ds_profile_identifier_type ident;
ds_profile_info_type       info_read;
ds_profile_tech_etype   profile_type =
DS_PROFILE_TECH_3GPP;
ds_profile_status_etype status       =
DS_PROFILE_REG_RESULT_FAIL;
ds_profile_action_etype act          =
DS_PROFILE_TRN_READ;
char *apn_name                       =
“test_apn”

ident =
DS_PROFILE_3GPP_PROFILE_PARAM_PDP_CONTEXT_
APN;

status                               =
ds_profile_begin_transaction(trn_type,
profile_type,
profile_num,
&profile_hndl);
/*Check for status before continuing*/

info_read.len                        =
DS_PROFILE_DB_MAX_APN_NAME_LEN+1;

info_read.buf                        =
(void*)malloc(info_read.len*sizeof(byte));

status                               =
ds_profile_get_param(profile_hndl,
ident,
&info_read);

/*Check for status before continuing*/
status                               =
ds_profile_end_transaction(profile_hndl,
act);
```

QUALCOMM® 2015-05-16 00:13:38 PDT dean_zhang@askey.com.tw

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION