EC2x&AG35-Quecopen Data Dial Application Guide

LTE Module Series

Rev. Quectel_EC2X&AG35-QuecOpen_Data_Call_Application_Guide

Date: 2018-10-20

Status: Preliminary

Our aim is to provide customers with timely and comprehensive service. For any assistance, please contact our company headquarters:

Quectel Wireless Solutions Co., Ltd.
7th Floor, Hongye Building, No.1801 Hongmei Road, Xuhui District, Shanghai 200233, China
Tel: +86 21 5108 6236
Email: info@quectel.com

Or our local office. For more information, please visit:
http://www.quectel.com/support/sales.htm

For technical support, or to report documentation errors, please visit:
http://www.quectel.com/support/technical.htm
Or email to: support@quectel.com

GENERAL NOTES

QUECTEL OFFERS THE INFORMATION AS A SERVICE TO ITS CUSTOMERS. THE INFORMATION PROVIDED IS BASED UPON CUSTOMERS' REQUIREMENTS. QUECTEL MAKES EVERY EFFORT TO ENSURE THE QUALITY OF THE INFORMATION IT MAKES AVAILABLE. QUECTEL DOES NOT MAKE ANY WARRANTY AS TO THE INFORMATION CONTAINED HEREIN, AND DOES NOT ACCEPT ANY LIABILITY FOR ANY INJURY, LOSS OR DAMAGE OF ANY KIND INCURRED BY USE OF OR RELIANCE UPON THE INFORMATION. ALL INFORMATION SUPPLIED HEREIN IS SUBJECT TO CHANGE WITHOUT PRIOR NOTICE.

COPYRIGHT

About the Document

History

Revision	Date	Author	Description
1.0	2017-08-15-	Running QIAN	Initial
1.1	2017-12-04	Running QIAN	Added DNS resolve ext appendix
1.2	2017-12-05	Running QIAN	Added checking modem status before data call
2.0	2017-12-12	Running QIAN	Added new data call API
2.1	2017-12-26	Running QIAN	Updated DNS common problems
2.2	2018-01-25	Running QIAN	Added CDMA network data call note in the FAQ
2.3	2018-03-10	Running QIAN	1. Modified document name from ***Multipath APN Data Call Quick Solution*** to ***Data Call Application Guide*** 2. Corrected API interface specification 3. Added new API interface
2.4	2018-03-19	Running QIAN	Added notes for 3GPP2's HDPR and HDPR2 network dial-up
2.5	2018-04-08	Running QIAN	Added varieties of network application scenario solutions.
2.6	2018-05-11	Running QIAN	Added the application scenario of multi-ip of host
2.7	2018-08-23	Tyler KUANG	Updated NW API
2.8	2018-10-20	Mike Zhou	Added interface function to query service status.
2.8	2019-3-5	Baron Qian	Added a common problem item

Contents

Table Index

Figure Index

1 Introduction

This document mainly introduces the process of establishing a wireless data business, i.e. Data Call. During the entire calling process, please strictly follow the title sequence to operate, especially for customers who are first exposed to wireless business.

Chapter 2 mainly introduces the use of USB-ECM network card.
Chapter 3 mainly introduces the calling process.
Chapter 4-10 mainly introduce the configuration of route, FIB and DNS for each application scenarios.

2 Network Card Connection

(1) For USB-ECM network card connection, please refer to below document:
Quectel_EC2X&AG35-QuecOpen_ECM_User_Guide
(2) For USB-RNDIS network card connection, please refer to below document:
Quectel_EC2X&AG35-QuecOpen_RNDIS_User_Guide

3 AP Side Data Call

This Chapter is based on the Openlinux SDK API to implement.

3.1 Device Check

Before data call, the customer need do a series of basic checks to judge whether the module is in the basic normal working state. The specific steps are as follows:

(1) Connect PC to the MAIN UART port or the USB AT port of the module via the USB turn serial port cable.
(2) Insert the SIM card and antenna, then power on.
(3) Check the following status through API in order

Test SIM card: QL_MCM_SIM_GetCardStatus ()
Detect signal strength: QL_MCM_NW_GetSignalStrength ()
Test module injection network: QL_MCM_NW_GetRegStatus()
Query operator: QL_MCM_NW_GetRegStatus()
Query network access technology: QL_MCM_NW_GetRegStatus()
Query call service status: QL_Data_Call_Init_Precondition()

3.2 APN Configuration

In general, for multiplex data call application scenarios, it is necessary to set up special APN to access private network. For each APNs, the parameters can be queried and configured by Ql_Apn_Get() and Ql_Apn_Set().

NOTES

1. The APN parameter setting must be done before calling, and set AutoSave at the same time. It still takes effect when next power on.
2. Generally for the public network APN, usrname, password and authentication parameters are not need to set.
3. Generally for the private network APN, whether usrname, password, and authentication parameters need to set or not, please consult with the operator.

4. For APN setting of eHRPD of CDMA, enforces profile_id to 0, and incoming usrname and password in the calling interface. The writing of this parameter does not affect the calling of the HPRD network.

(1) Please set QL_Data_Call_Get_Default_Profile() value same as profile_id.
(2) Please refer to Chapter 3.4 to call.
(3) Please ignore the configuration of route, forwarding and DNS.

3.3 Single Path Data Call

3.4 Mutiple Path Data Call

3.4

```
// Initial and register callback;
Ql_Data_Call_Init(user_callback)

// Automatic configuration default routing and default forwarding are not required.
QL_Data_Call_Set_Default_Profile (8)
//Establish APN1 data channel.

Int err_code1;
ql_data_call_s    data1_call_paras;

data1_call_paras.profile_idx = 1;// APN1
data1_call_paras.ip_family = QL_DATA_CALL_IPV4;// Only call IPV4
data1_call_paras.reconnet = true; // Turn on automatic connection

Ql_Data_Call_Start(&data1_call_paras,    & err_code1)//…The current Linux system will show
“rmnet_data0”

//Establish APN2 data channel.
Int err_code2;
ql_data_call_s    data2_call_paras;

data2_call_paras.profile_idx = 2;// APN2
data2_call_paras.ip_family = QL_DATA_CALL_IPV4;
data2_call_paras.reconnet = true;

Ql_Data_Call_Start(&data2_call_paras,    & err_code2)//…The current Linux system will
show“rmnet_data1”

//Establish APN3 data channel.
```

```
Int err_code3;
ql_data_call_s    data3_call_paras;

data3_call_paras.profile_idx = 3; // APN3
data3_call_paras.ip_family = QL_DATA_CALL_IPV4;
data3_call_paras.reconnet = true;

Ql_Data_Call_Start(&data3_call_paras,    & err_code3)//…The current Linux system will
show“rmnet_data2”
```

NOTE

For multiple data call, please make sure the value of profile_idx is different from QL_Data_Call_Get_Default_Profile ().

3.4.2 Routing and Other Parameter Configurations

For different application scenarios, the setup steps are different, please refer to the later chapters.

3.5 3GPP2/CDMA Call

How to enable CDMA Network Calling:

(1) To confirm whether the SIM card operator can support CDMA network.

(2) Before calling, call QL_MCM_NW_GetRegStatus () first to query the current module is registered in the LTE network or the CDMA network.

(3) profile_id is enforced to be 0, meanwhile CDMA network can’t support multiple APN data call.

(4) The current network is HPRD or eHRPD is not matter.

```
Int err_code1;
ql_data_call_s    data1_call_paras;
char username[] = {“this is example”};
char passwd[] = {“this is example”};

data1_call_paras.profile_idx = 0;    //Must be 0
data1_call_paras.ip_family = QL_DATA_CALL_IPV4;// Only call IPV4
data1_call_paras.reconnet = true; // Enable automatic reconnection
data1_call_paras.cdma_username = usrname;
data1_call_paras.cdma_password = passwd;
Ql_Data_Call_Start(&data1_call_paras, & err_code1)//Current Linux system will appear rmnet_data0
```

4 Scenario 1: Single Path

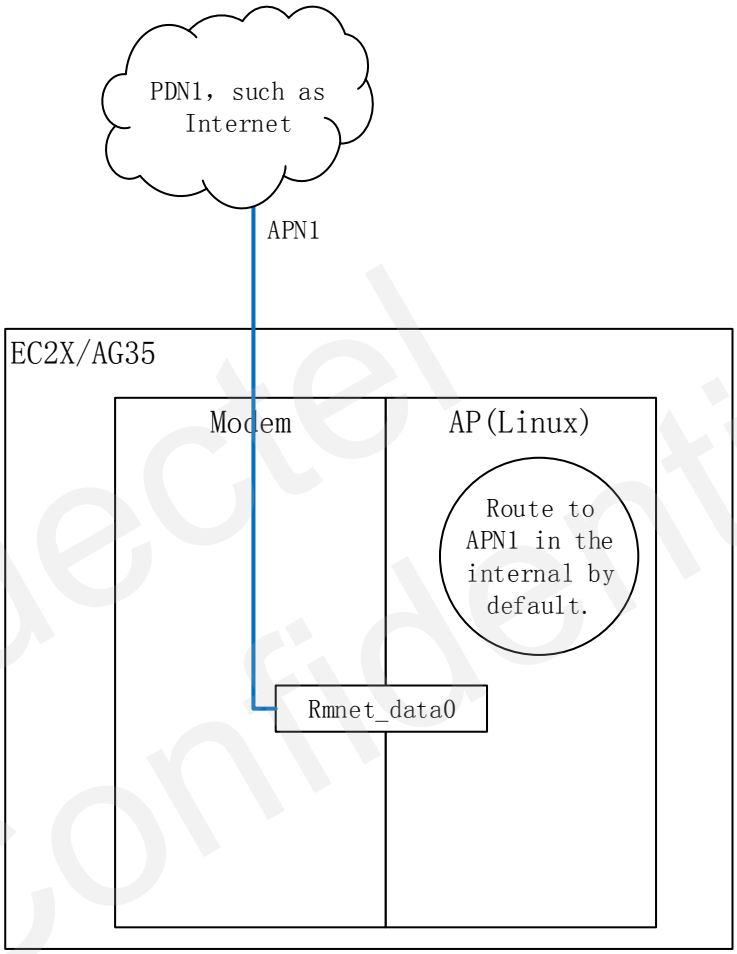

Figure 1: Scenario 1

Under this scenario, any configuration of route, FIB and DNS is not required, only need to operate follow Chapter 3.3.

5 Scenario 2:Single Path+ECM Device

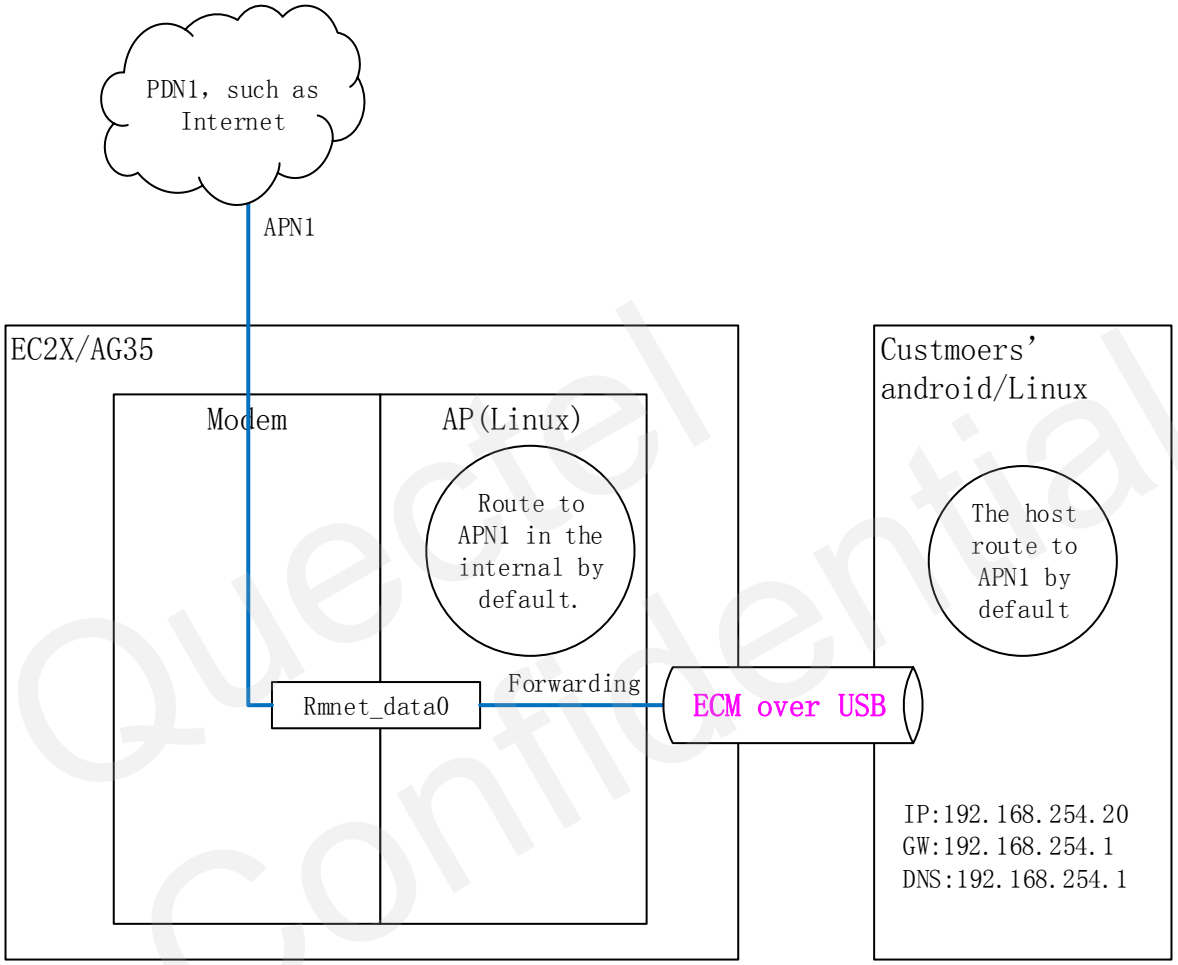

Figure 2: Scenario 2

Under this scenario, any configuration of route, FIB and DNS is not required, only need to operate follow Chapter 3 and Chapter 3.3.

6 Scenario 3: Multi Path

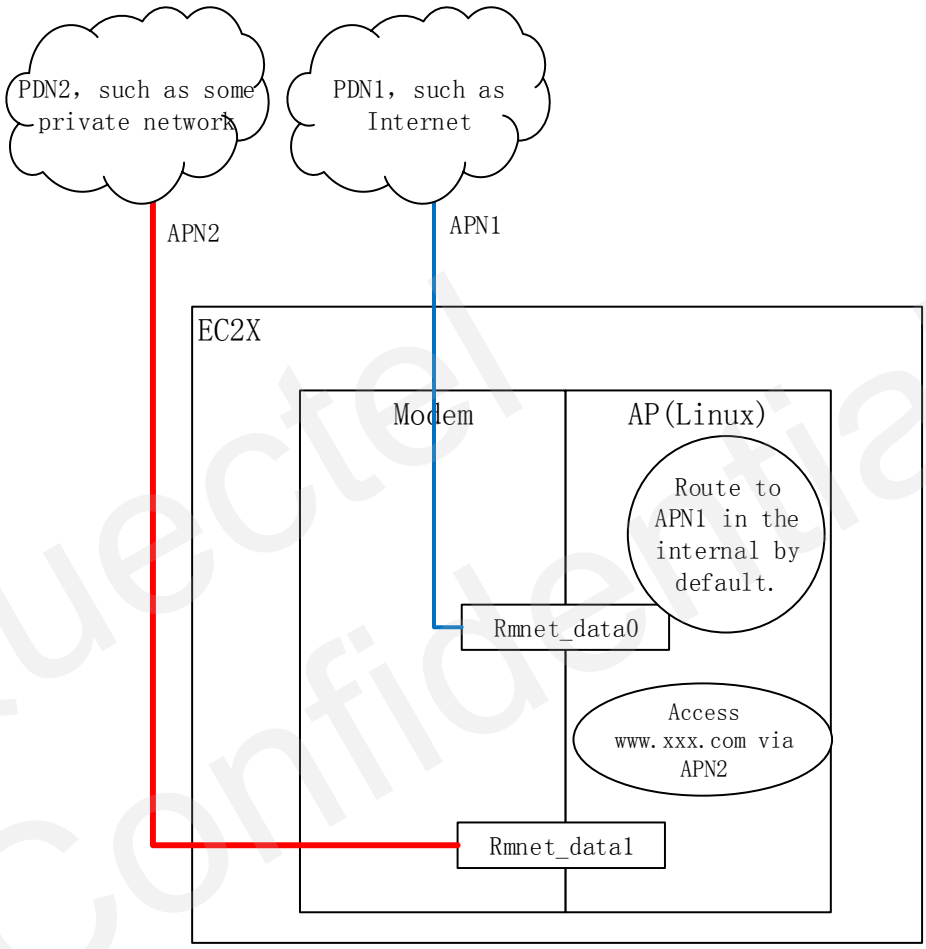

Figure 3: Scenario 3

6.1 Delete Rules

```
route del default
iptables -t filter -F
iptables -t nat -F
```

6.2 Defaut Route

```
//Get APN1 Gateway address
Ql_Data_Call_Info_Get(2, &info);

// Set default route, assumed address is 10.112.7.176
route add default dev rmnet_data0
//or
route add –net 10.112.7.0/24 dev rmnet_data0
route add default gw 10.112.7.176
```

```
//Get APN1 DNS address(API)
Ql_Data_Call_Info_Get(1, &info);

//Set default DNS server (command), assumed address is primary: 211.138.180.2 or secdnary:
211.138.180.3
echo “nameserver 211.138.180.2” > /etc/resolv.conf
echo “nameserver 211.138.180.3” >> /etc/resolv.conf
```

6.4 Access a Server via APN2

6.4.1 DNS & Route Configuration

```
// Get APN2 DNS address(API)
Ql_Data_Call_Info_Get(2, &info);

//Set default DNS server (command), assumed address is primary: 121.158.280.8 or secdnary:
121.158.280.9
ip route add 121.158.200.8/32 dev rmnet_data1
ip route add 121.158.200.9/32 dev rmnet_data1
```

6.4.2 Domain Name Resolution

6.4.3 Route Configuration

```
//SDK API, usage is as follows:
QL_nslookup( www.xxx.com, 121.158.280.8, IPV4, resolved_output), in which 121.158.280.8 is DNS
address of APN2.
```

```
route add –net 47.88.189.189/32 gw 10.32.80.46 dev rmnet_data1
```

In which 47.88.189.18 is the address of www.xxx.com, 10.32.80.46 is the gateway of APN2 (Gotten from Ql_Data_Call_Info_Get)

7 Scenario 4: Multi Path + ECM (1)

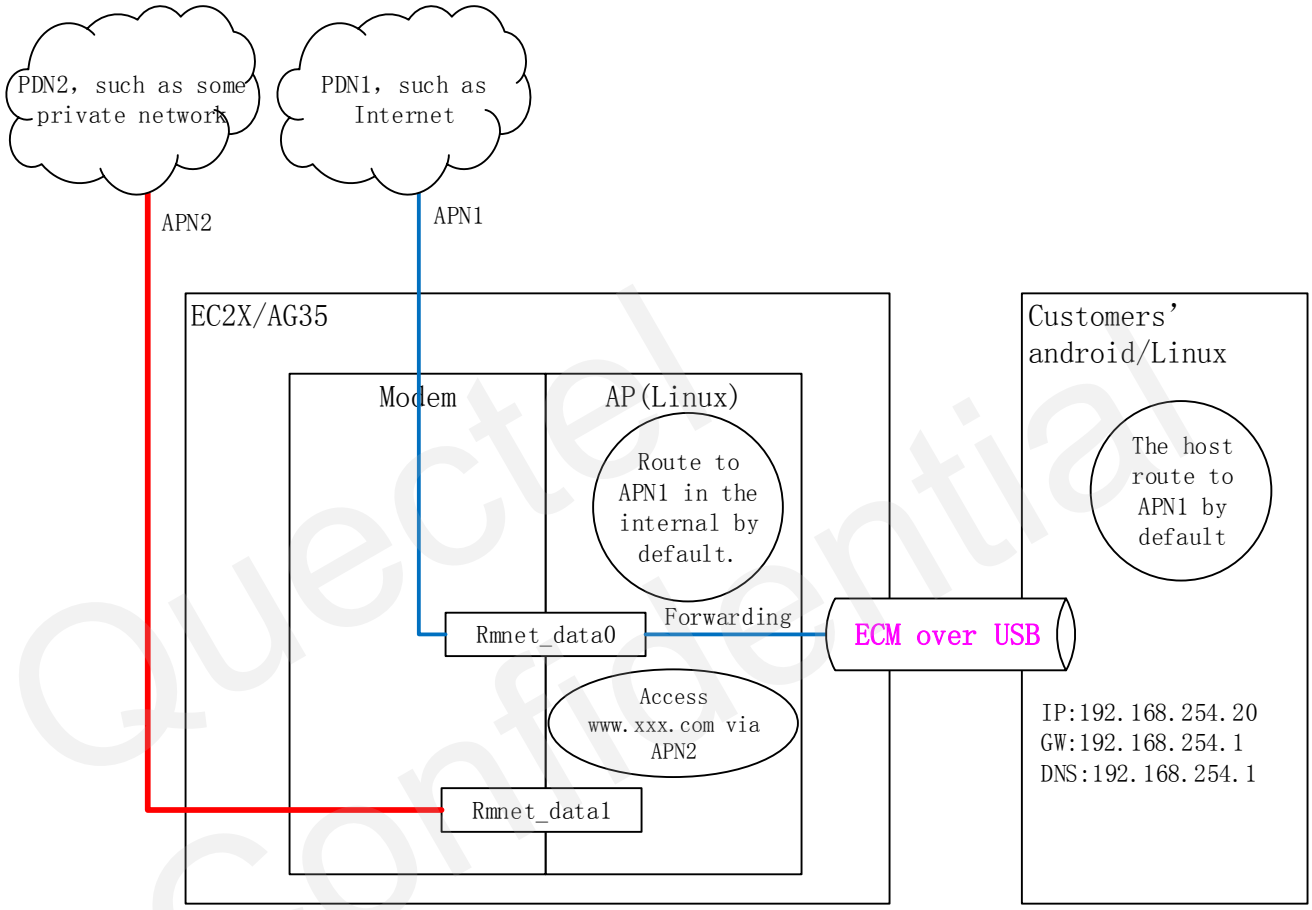

Figure 4: Scenario 4

The basic steps are similar as the one in Chapter 6, except turn on default FIB before performing Chapter 6.4.

```
//Turn on default FIB.
echo 1 > /proc/sys/net/ipv4/ip_forward

//Set FIB
iptables -t nat -A POSTROUTING -o rmnet_data0 -j MASQUERADE --random
```

8 Scenario 5: Multi Path + ECM (2)

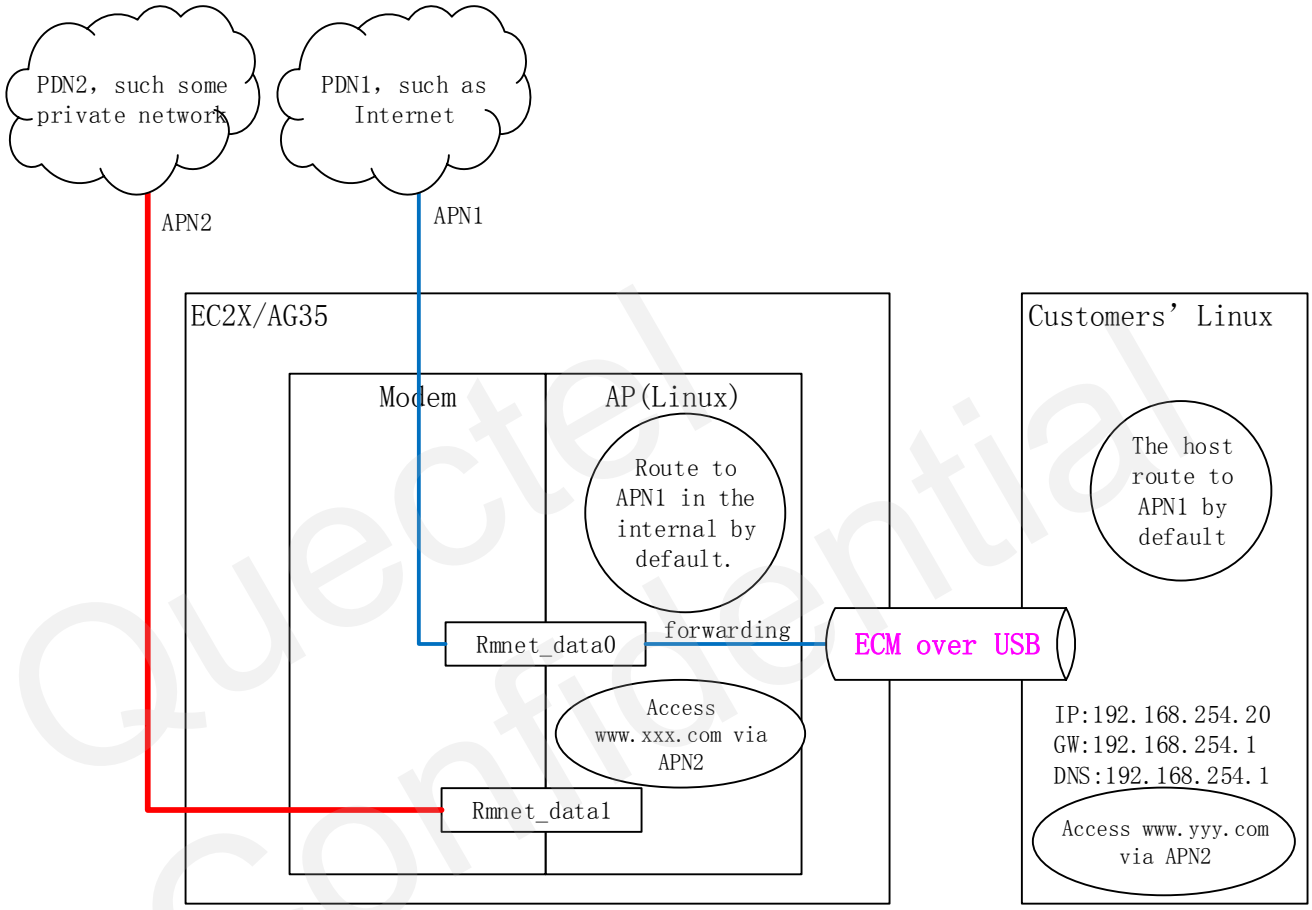

Figure 5: Scenario 5

The basic steps are similar as the one in Scenario 4, except the host accesses some server through APN2.

8.1 Domain Name Resolution

Because the domain name resolution of the host is all done by the module AP side dnsmasq through the DNS of APN1, if the host wants to access the server via domain name, it must set up a server (a socket) on the AP side to complete the DNS resolution and FIB configuration.

8.2 Route Settings

If host accesses server via IP address directly, only need configure a forwarding rule on AP side, shown

as followings.

```
//If IP address of www.yyy.com is 47.88.189.189
//Forwarding Setting
iptables -t nat -A POSTROUTING -d 47.88.189.189 -o rmnet_data1 -j MASQUERADE
//or
iptables -t nat -A POSTROUTING -o rmnet_data1 -j MASQUERADE --random

//Route Setting
ip route add 47.88.189.189/32 dev rmnet_data1
```

9 Scenario 6: Multi Path + ECM (3)

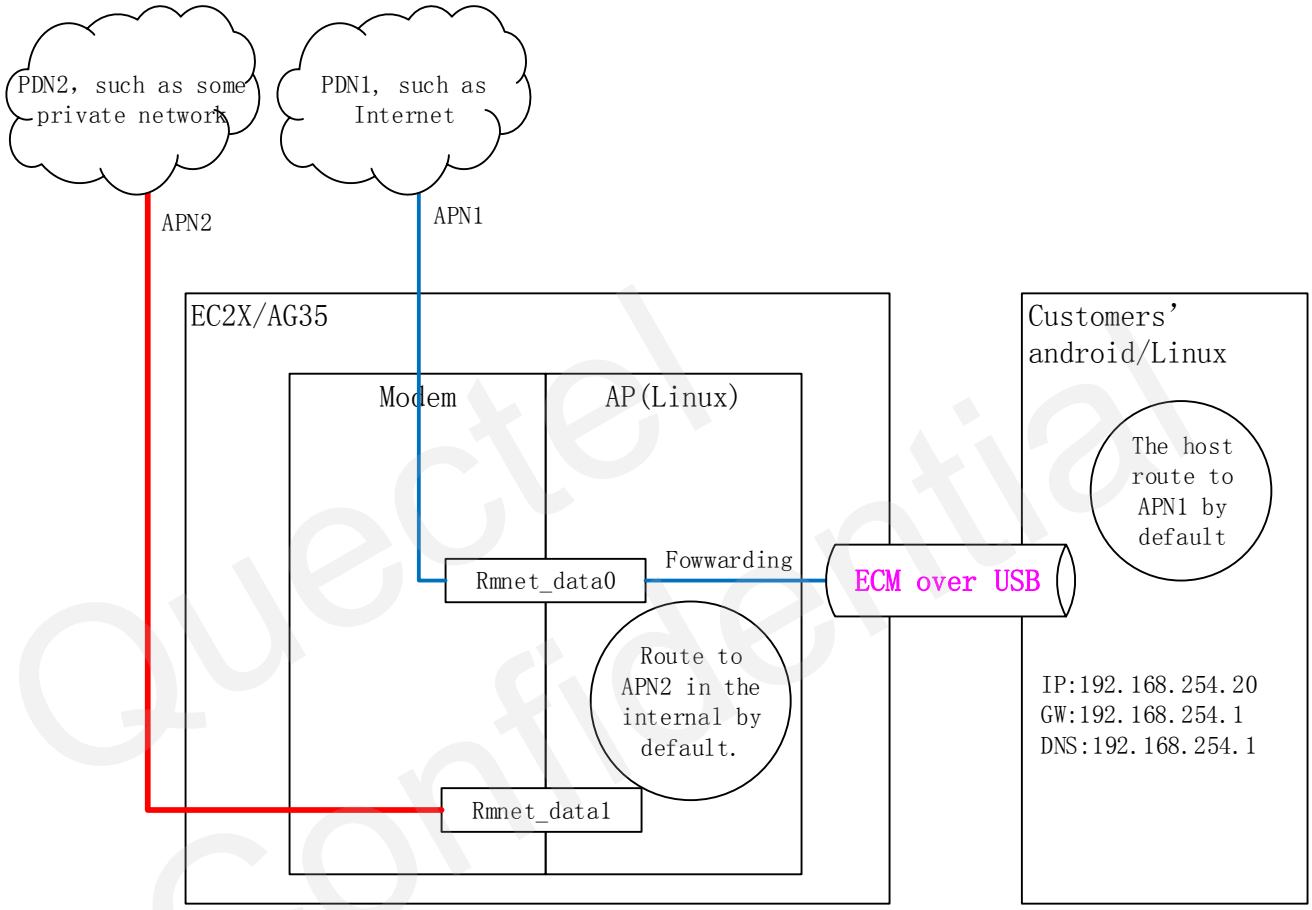

Figure 6: Scenario 6

9.1 Delete Rules

```
route del default
iptables -t filter -F
iptables -t nat -F
```

9.2 Route Settings on AP Side

```
//Get gateway address of APN2
Ql_Data_Call_Info_Get(2, &info);
//Set default route, assumed address is 10.32.80.46
```

```
route add default dev rmnet_data2
//or
route add –net 10.32.80.0/24 dev rmnet_data1
route add default gw 10.32.80.46
```

DNS Settings on AP Side

```
//Get DNS address of APN2( API)
Ql_Data_Call_Info_Get(2, &info);
```

9.3

```
// Set default DNS server (command), assumed address is primary: 121.158.280.8 and secdnary:
121.158.280.9
echo “nameserver 121.158.280.8” > /etc/resolv.conf
echo “nameserver 121.158.280.9” >> /etc/resolv.conf
```

9.4 FIB Configuration

FIB configuration is the key step to realize host access outer network.

```
//Turn on default forwarding
echo 1 > /proc/sys/net/ipv4/ip_forward
//Set FIB
iptables -t nat -A POSTROUTING -o rmnet_data0 -j MASQUERADE –random
```

9.5 Policy Based Route Configuration

Below steps are performed on AP side.

(1) Create RIB

```
echo "200     rmnet_data_apn1 " >> /etc/iproute2/rt_tables       //EC2X Module
echo "200     rmnet_data_apn1" >> /data/iproute2/rt_tables       //AG35 Module
```

(2) Set policy base route

```
ip rule add from 192.168.225.0/24 table 200          //192.168.225.0 is the network address of ECM
device.
```

(3) Add route rules

```
ip route add dev rmnet_data0 table 200
//or
ip route add via 10.112.7.176 table 200              //10.112.7.176 is the gateway address
```

NOTES

1. If host DNS address is sent by module, please perform below steps.
2. If the host sets up a known public DNS server, such as 8.8.8.8, 114.114.114.114, etc., the following steps are not required.

9.6.1 dnsmasq Configuration

(1) Modify file /etc/dnsmasq.conf

```
# Change this line if you want dns to get its upstream servers from
# somewhere other that /etc/resolv.conf
resolv-file=/etc/dnsmasq_resolv.conf

# By  default,  dnsmasq  will  send queries to any of the upstream
# servers it knows about and tries to favour servers to are  known
# to  be  up.  Uncommenting this forces dnsmasq to try each query
# with  each  server  strictly  in  the  order  they  appear  in
# /etc/resolv.conf
#strict-order
```

```
# If you don't want dnsmasq to read /etc/hosts, uncomment the
# following line.
no-hosts
# or if you want it to read another file, as well as /etc/hosts, use
# this.
#addn-hosts=/etc/banner_add_hosts
```

Sync, restart module.

(2) Create file /etc/dnsmasq_resolv.conf, add DNS server address of APN1

```
//Get APN1 DNS address(API)
QI_Data_Call_Info_Get(1, &info);
// Set default DNS server (command), assumed address is primary: 211.138.180.2 and secdnary: 211.138.180.3
echo “nameserver 211.138.180.2” > /etc/ dnsmasq_resolv.conf
echo “nameserver 211.138.180.3” >> /etc/dnsmasq_resolv.conf
```

9.6.2 DNS Route Settings of APN1

```
//Get APN1 DNS address(API)
QI_Data_Call_Info_Get(1, &info);

// Set default DNS server (command), assumed address is primary: 211.138.180.2 and secdnary: 211.138.180.3
ip route add 211.138.180.2/32 dev rmnet_data0
ip route add 211.138.180.3/32 dev rmnet_data0
```

NOTE

File dnsmasq.conf of AG35 module is under path /data

10 Scenario 7: Multi Path + ECM (4)

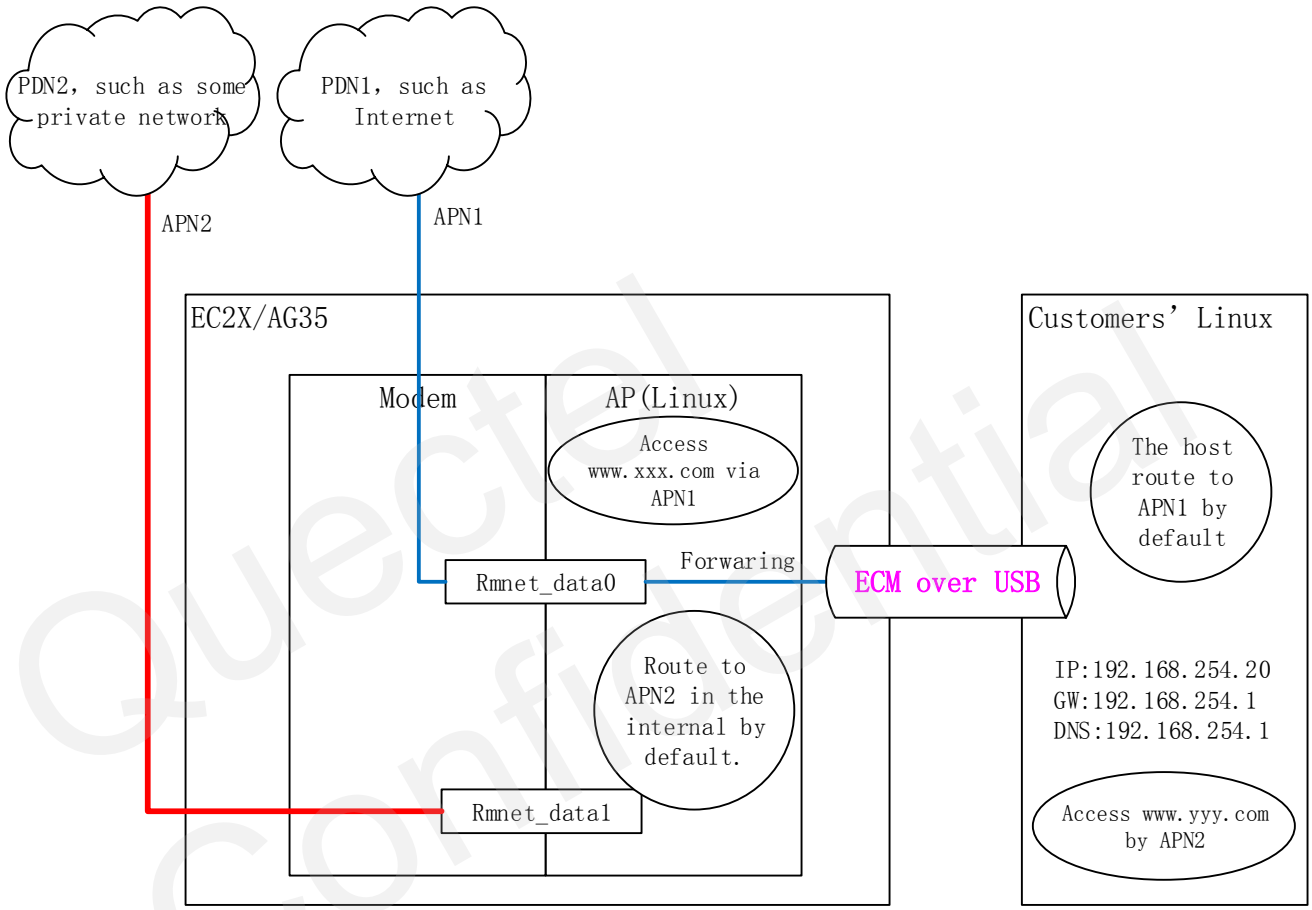

Figure 7: Scenario 7

10.1 The Basic Settings

Please refer to Chapter 9 to complete all basic settings.

10.2 Access a Server via APN1

10.2.1 Domain Name Resolution

10.2.2 Route Settings

//SDK API, the usage is shown as following
QL_nslookup(www.xxx.com, 211.138.180.2, IPV4, resolved_output), in which 211.138.180.2 is the DNS address of APN1.

```
//Settings of access rules in the module
route add –net 47.88.189.189/32 gw 10.112.7.176 dev rmnet_data0
In which 47.88.189.189 is the address of www.xxx.com, 10.112.7.176 is the gateway APN1(Gotten from Ql_Data_Call_Info_Get)
```

10.3.1 Domain Resolution Name

Because the domain name resolution of the host is all done by the module AP side dnsmasq through the DNS of APN1, if we want to access the server via domain name, it must set up a server (a socket) on the AP side to complete the DNS resolution and FIB configuration.

10.3 Access a Server via APN2

10.3.2 Route Settings

If host accesses server via IP address directly, only need configure a forwarding rule on AP side, shown as followings.

```
//Assumed that IP address of www.yyy.com is 47.88.189.189
//Forwarding Settings
iptables -t nat -A POSTROUTING -d 47.88.189.189 -o rmnet_data1 -j MASQUERADE
//or
iptables -t nat -A POSTROUTING -o rmnet_data1 -j MASQUERADE --random

//Route Settings
ip route add 47.88.189.189/32 dev rmnet_data1 table 200
//or based on policy base settings
ip rule add to 47.88.189.189 table main
```

11 Scenario 8: Multi Path + ECM (5)

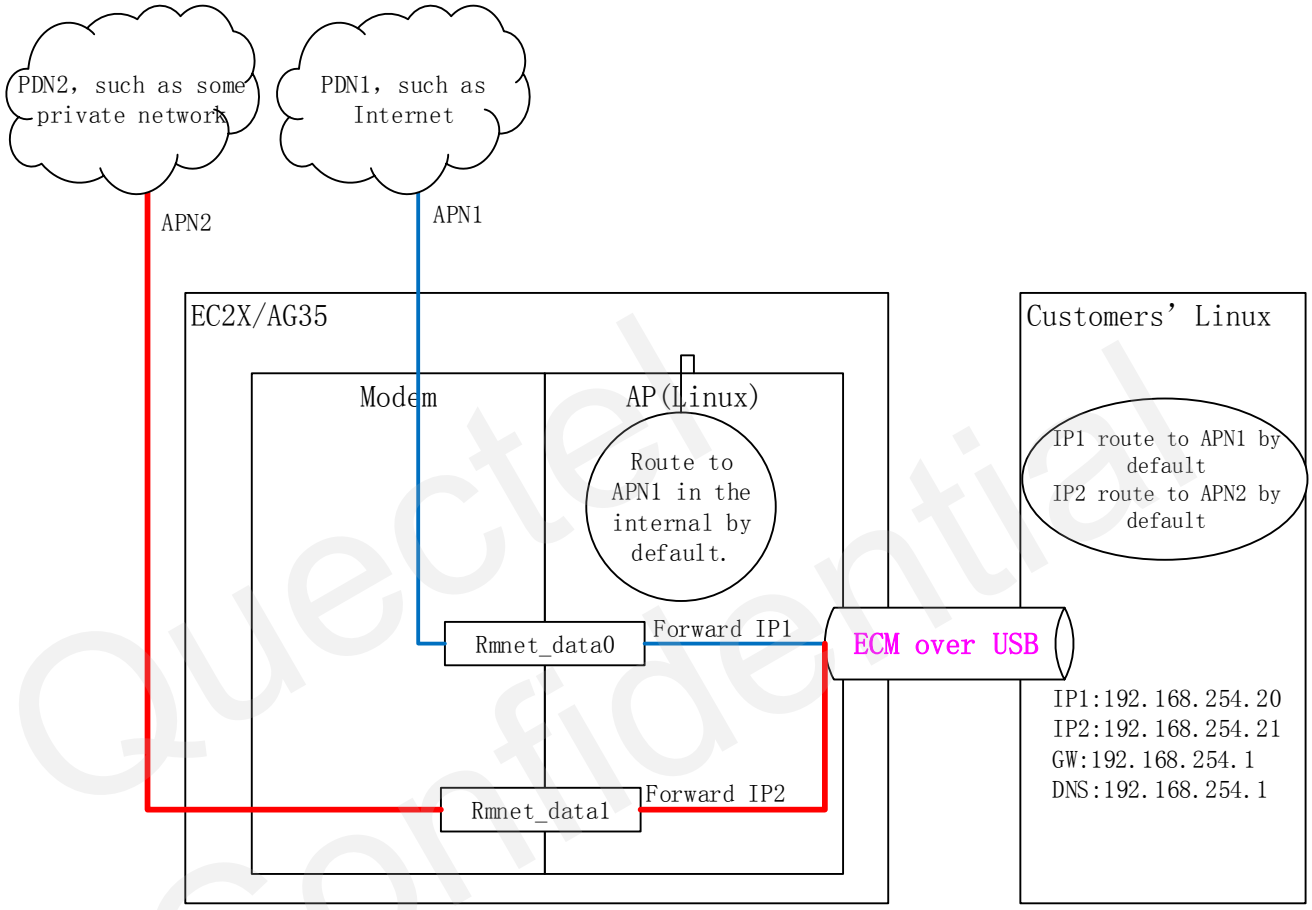

Figure 8: Scenario 8

11.1 Route and DNS Settings on AP Side

11.2 Policy Based Settings

Please refer to Chapter 9.1 & 9.3.

11.3 DNS Configuration

Please refer to Chapter 9.4 & 9.5.

For DNS settings, the DNS address should correspond to the each respective APN, and the following

routing rules can be created in policy routing table.

(1) Access the DNS server address delivered by APN1, DNS server address is routed to APN1 gateway.

(2) Access the DNS server address delivered by APN2, DNS server address is routed to APN2 gateway.

12 Supplementary Instructions

This chapter is mainly about some additional instructions for the DNS that supported by Linux system, as well as guidelines for private network domain name resolution.

12.1 API

(1) The Linux standard function gethostbyname(), resolve the IPV4 address through the DNS address in resolve.conf, or directly through /etc/hosts.
(2) The Linux standard function getaddrinfo(), resolve IPV4 and IPV6 address through the DNS address in resolve.conf at the same time, or directly through /etc/hosts.
(3) The Quectel function QL_nslookup(), resolve IPV4 and IPV6 addresses separately through specified DNS address.

NOTE

All above API can't guarantee a one-time resolving success, need to call 2-3 times repeatedly, this is because the DNS transport layer is to take the UDP protocol, losing packet is normal. Besides, the network connectivity of the DNS server is also important.

12.2 Command

(1) ping, returned domain name resolution is based on Linux system default DNS.
(2) nslookup, in module only can resolve domain name through resolve.conf default DNS, which is different from on PC.

13 Data Call API

Data Call Type

13.1.1 ql_data_call_error_e

```
typedef enum {
    QL_DATA_CALL_ERROR_NONE = 0,
    QL_DATA_CALL_ERROR_INVALID_PARAMS,
} ql_data_call_error_e;
```

13.1.2 ql_data_call_state_e

```
typedef enum {
    QL_DATA_CALL_DISCONNECTED = 0,          /*!< call is disconnected */
    QL_DATA_CALL_CONNECTED,                 /*!< call is connected */
} ql_data_call_state_e;
```

13.1.3 ql_data_call_ip_family_e

```
typedef enum {
    QL_DATA_CALL_TYPE_IPV4 = 0,     /*!< IPv4 call. */
    QL_DATA_CALL_TYPE_IPV6,         /*!< IPv6 call. */
    QL_DATA_CALL_TYPE_IPV4V6,       /*!< IPv4 and IPv6 call (Only used call start or stop). */
} ql_data_call_ip_family_e;
```

13.1.4 ql_apn_pdp_type_e

```
typedef enum {
    QL_APN_PDP_TYPE_IPV4 = 0,
    QL_APN_PDP_TYPE_PPP,
    QL_APN_PDP_TYPE_IPV6,
    QL_APN_PDP_TYPE_IPV4V6,
} ql_apn_pdp_type_e;
```

13.1.5 ql_apn_auth_proto_e

```
typedef enum {
    QL_APN_AUTH_PROTO_DEFAULT = 0,
    QL_APN_AUTH_PROTO_NONE,
```

```
    QL_APN_AUTH_PROTO_PAP,
    QL_APN_AUTH_PROTO_CHAP,
    QL_APN_AUTH_PROTO_PAP_CHAP,
} ql_apn_auth_proto_e;
```

13.1.6 v4_address_status

```
struct v4_address_status {
    struct in_addr ip;                  /*!< Public IPv4 address. */
    struct in_addr gateway;             /*!< Public IPv4 gateway. */
    struct in_addr pri_dns;             /*!< Primary Domain Name Service IP address. */
    struct in_addr sec_dns;             /*!< Secondary Domain Name Service IP address. */
};
```

13.1.7 v6_address_status

```
struct v6_address_status {
    struct in6_addr ip;                 /*!< Public IPv6 address. */
    struct in6_addr gateway;            /*!< Public IPv6 gateway. */
    struct in6_addr pri_dns;            /*!< Primary Domain Name Service IPv6 address. */
    struct in6_addr sec_dns;            /*!< Secondary Domain Name Service IPv6 address. */
};
```

13.1.8 ql_data_call_state_s

```
typedef struct {
    char profile_idx;                           /*!< UMTS/CMDA profile ID. */
    char name[16];                              /*!< Interface Name. */
    ql_data_call_ip_family_e ip_family;         /*!< IP version. */
    ql_data_call_state_e state;                 /*!< The dial status. */
    ql_data_call_error_e err;                   /*!< The Reason code after data call disconnected. */

union {
        struct v4_address_status v4;            /*!< IPv4 information. */
        struct v6_address_status v6;            /*!< IPv6 information. */
    };
} ql_data_call_state_s;
```

13.1.9 ql_data_call_s

```
/*
 *!< Client callback function used to post event indications. */
typedef void (*ql_data_call_evt_cb_t)(ql_data_call_state_s *state);

typedef struct {
```

```
    char profile_idx;                       /*!< UMTS/CMDA profile ID. */
    bool reconnect;                         /*!< Whether to re-dial after disconnecting the network. */
    ql_data_call_ip_family_e ip_family;     /*!< IP version. */

    char cdma_username[127];
    char cdma_password[127];
} ql_data_call_s;
```

13.1.10 pkt_stats

```
struct pkt_stats {
    unsigned long pkts_tx;              /*!< Number of packets transmitted. */
    unsigned long pkts_rx;              /*!< Number of packets received. */
    long long bytes_tx;                 /*!< Number of bytes transmitted. */
    long long bytes_rx;                 /*!< Number of bytes received. */
    unsigned long pkts_dropped_tx;      /*!< Number of transmit packets dropped. */
    unsigned long pkts_dropped_rx;      /*!< Number of receive packets dropped. */
};
```

13.1.11 v4_info

```
struct v4_info {
    char name[16];                      /*!< Interface Name. */
    ql_data_call_state_e state;         /*!< The dial status. */
    bool reconnect;                     /*!< re-dial flag. */
    struct v4_address_status addr;      /*!< IPv4 IP Address information. */
    struct pkt_stats stats;             /*!< IPv4 statics */
};
```

13.1.12 v6_info

```
struct v6_info {
    char name[16];                      /*!< Interface Name. */
    ql_data_call_state_e state;         /*!< The dial status. */
    bool reconnect;                     /*!< re-dial flag. */
    struct v6_address_status addr;      /*!< IPv6 IP Address information. */
    struct pkt_stats stats;             /*!< IPv6 statics */
};
```

13.1.13 ql_data_call_info_s

```
typedef struct {
    char profile_idx;                       /*!< UMTS/CDMA profile ID. */
    ql_data_call_ip_family_e ip_family;     /*!< IP version. */
    struct v4_info v4;                      /*!< IPv4 information */
```

```
    struct v6_info v6;                          /*!< IPv6 information */
} ql_data_call_info_s;
```

13.1.14 ql_apn_info_s

```
typedef struct {
    unsigned char profile_idx;          /*!< UMTS/CDMA profile ID. */
ql_apn_pdp_type_e pdp_type;
ql_apn_auth_proto_e auth_proto;         /*!< Authentication Protocol. */
char apn_name[QL_APN_NAME_SIZE];
char username[QL_APN_USERNAME_SIZE];    /*!< Username used during data network
authentication. */
    char password[QL_APN_PASSWORD_SIZE];    /*!< Password to be used during data network
authentication. */
} ql_apn_info_s;
```

13.1.15 ql_apn_add_s

```
typedef struct {
ql_apn_pdp_type_e pdp_type;
ql_apn_auth_proto_e auth_proto;             /*!< Authentication Protocol. */
char apn_name[QL_APN_NAME_SIZE];
char username[QL_APN_USERNAME_SIZE];    /*!< Username used during data network authentication.
*/
char password[QL_APN_PASSWORD_SIZE];    /*!< Password to be used during data network
authentication. */
} ql_apn_add_s;
```

13.1.16 ql_apn_info_list_s

```
typedef struct {
    int cnt;
    ql_apn_info_s apn[QL_APN_MAX_LIST];
} ql_apn_info_list_s;
```

13.2 Function

13.2.1 QL_Data_Call_Init

```
/**
 * Initialization data call module, and callback function registered.
 *
 * @param [in] evt_cb                   callback fucntion
 *
```

```
 * @return
 *     On success, 0 is returned.    On error, -1 is returned.
 *
 */
int QL_Data_Call_Init (ql_data_call_evt_cb_t evt_cb)
```

13.2.2 QL_Data_Call_Destroy

```
/**
 * Destroy data call module, and unregister callback funciton
 *
 * @param
 *     None
 *
 * @return
 *     On success, 0 is returned.    On error, -1 is returned.
 *
 */
void QL_Data_Call_Destroy (void)
```

13.2.3 QL_Data_Call_Start

```
/**
 * Starts a data call.
 *
 * @param [in] data_call            The data call parameters
 * @param [out] error               Error code returned by data call
 *
 * @return
 *     On success, 0 is returned.    On error, -1 is returned.
 *
 */
int QL_Data_Call_Start (ql_data_call_s *data_call, ql_data_call_error_e *err)
```

13.2.4 QL_Data_Call_Stop

```
/**
 * Stop a data call.
 *
 * @param [in] profile_idx          UMTS/CDMA profile ID
 * @param [in] ip_family            IP Version
 * @param [out] error               Error code returned by data call
 *
 * @return
 *     On success, 0 is returned.    On error, -1 is returned.
```

```
 *
 */
int QL_Data_Call_Stop (char profile_idx, ql_data_call_ip_family_e ip_family, ql_data_call_error_e *err)
```

13.2.5 QL_Data_Call_Info_Get

```
/**
 * Get a data call information.
 *
 * @param [in] profile_idx          UMTS/CDMA profile ID
 * @param [in] ip_family             IP Version
 * @param [out] info                  The Data Call information
 * @param [out] error                 Error code returned by data call
 *
 * @return
 *     On success, 0 is returned.   On error, -1 is returned.
 *
 */
int QL_Data_Call_Info_Get (
char profile_idx,
ql_data_call_ip_family_e ip_family,
ql_data_call_info_s   *info,
ql_data_call_error_e   *err)
```

13.2.6 QL_APN_Set

```
/**
 * Changes the settings in a configured profile. if this profile is not exist, will creates a configured *
*   profile with specified settings
 *
 * @param [in] apn          the profile information.
 *
 * @return
 *     On success, 0 is returned.   On error, -1 is returned.
 */
int QL_APN_Set (ql_apn_info_s *apn)
```

13.2.7 QL_APN_Get

```
/**
 * Retrieves the settings from a configured profile.
 *
 * @param [in] profile_idx            UMTS/CDMA profile ID
 * @param [out] apn                       the profile information.
 *
```

```
 * @return
 *     On success, 0 is returned.    On error, -1 is returned.
 *
 */
int QL_APN_Get (unsigned char profile_idx, ql_apn_info_s *apn)
```

13.2.8 QL_APN_Add

```
/**
 * Retrieves the settings from a configured profile.
 *
 * @param [in] apn                    the profile information.
 * @param [out] profile_idx           UMTS/CDMA profile ID
 *
 * @return
 *     On success, 0 is returned.    On error, -1 is returned.
 *
 */
extern int QL_APN_Add(ql_apn_add_s *apn, unsigned char *profile_idx);
```

13.2.9 QL_APN_Del

```
/**
 * Delete a configured profile.
 *
 * @param [in] profile_idx            UMTS/CDMA profile ID
 *
 * @return
 *     On success, 0 is returned.    On error, -1 is returned.
 */
extern int QL_APN_Del(unsigned char profile_idx);
```

13.2.10 QL_APN_Get_Lists

```
/**
 * Retrieves the settings from a configured profile list.
 *
 * @param [out] apn_list              the profile list information.
 *
 * @return
 *     On success, 0 is returned.    On error, -1 is returned.
 *
 */
extern int QL_APN_Get_Lists(ql_apn_info_list_s *apn_list);
```

13.2.11 QL_Data_Call_Init_Precondition

```
/**
 * Get the running status of the quectel manager service.
 *
 * @param
 *     None
 *
 * @return
 *     If the service is working properly 0 is returned, otherwise -1 is returned.
 *
 */
```

14 Common Problem

14.1 Capture PCAP Log

(1) Enter the following Command
Capture all network interface
tcpdump -i any -p -vv -s 0 -w ./capture1.pcap &
Capture specified network interface
tcpdump -i rmnet_data0 -p -vv -s 0 -w ./capture1.pcap &
(2) Run the program
(3) Kill tcpdump command, upload capture1.pcap file.

14.2 Check iptables Table

(1) Check nat table
iptables -nvt nat -L
(2) Check fliter table
iptables -nvt filter -L

14.3 How to exclude when 3GPP2/CDMA cannot dial-up due to the authentication

(1) Check modem side
Set Parameters:
at+qctpwdcfg="<username>","<userpasswd>"
at+qcfg=”cdmaruim”,1
Call Testing
at+qiact=1// If failed return error

If failed, the username and password may be incorrect.

(2) Check AP side
If step 1 passed, check whether profile_id is 0, authentication parameters are correct, specifically please refer to **Chapter 4.2** and **Chapter 4.5**.

14.4 Data Call Failure

After module booting up and automatically running customer APP, sometimes the data calling failure will occur, but calling again will be successful. The main reason is when calling, its service process is not completed, the interface QL_Data_Call_Init_Precondition() can be used to check.

14.5 When the module dials up in IPv4&IPv6, may be have a problem of sending DNS Server IPv6 Address Request

When the module Use IPv4&IPv6 to dial-up, but the current network does not support IPv6, there may be a problem of DNS Server IPV6 Address Request. The current solution is to specify Only IPv4 to dial when dialing. Specifically, when calling QL_Data_Call_Start function, specify ip_family in ql_data_call_s as QL_DATA_CALL_TYPE_IPV4。

15 Appendix A References

Table 1: Related Documents

SN	Document Name	Remark
[1]	Quectel_EC2x&AG35-QuecOpen_Quick_Start	

Table 2: Terms and Abbreviations

Abbreviation	Description
APN	Access Point Name
ECM	Ethernet Networking Control Model
EPS	Evolved Packet System
PDP	Packet Data Protocol
HRPD	High Rate Packet Data