

# **EC2x&EG9x&EG25-G**

## **QuecOpen Data Call User Guide**

**LTE Standard Module Series**

Rev. EC2x&EG9x&EG25-G\_QuecOpen\_Data\_Call\_User\_Guide\_V1.0

Date: 2020-05-31

Status: Released

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

**Quectel Wireless Solutions Co., Ltd.**

Building 5, Shanghai Business Park Phase III (Area B), No.1016 Tianlin Road, Minhang District, Shanghai, 200233, China

Tel: +86 21 5108 6236

Email: [info@quectel.com](mailto: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](mailto: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**

THE INFORMATION CONTAINED HERE IS PROPRIETARY TECHNICAL INFORMATION OF QUECTEL WIRELESS SOLUTIONS CO., LTD. TRANSMITTING, REPRODUCTION, DISSEMINATION AND EDITING OF THIS DOCUMENT AS WELL AS UTILIZATION OF THE CONTENT WITHOUT PERMISSION ARE FORBIDDEN. OFFENDERS WILL BE HELD LIABLE FOR PAYMENT OF DAMAGES. ALL RIGHTS ARE RESERVED IN THE EVENT OF A PATENT GRANT OR REGISTRATION OF A UTILITY MODEL OR DESIGN.

***Copyright © Quectel Wireless Solutions Co., Ltd. 2020. All rights reserved.***

# About the Document

## Revision History

Version	Date	Author	Description
1.0	2020-05-31	Running QIAN/ Tyler KUANG/ Baron QIAN/ Mike ZHOU	Initial

---

## Contents

About the Document .....	2
Contents .....	3
Table Index .....	6
Figure Index .....	7
<b>1 Introduction .....</b>	<b>8</b>
1.1. Applicable Modules .....	8
<b>2 Network Card Connection .....</b>	<b>9</b>
<b>3 Data Call on AP Side .....</b>	<b>10</b>
3.1. Device Check .....	10
3.2. APN Configuration .....	10
3.3. Single Data Call .....	11
3.4. Multiplex Data Call .....	11
3.4.1. Data Call .....	11
3.4.2. Routing and Other Parameters Configuration .....	12
3.5. 3GPP2/CDMA Data Call .....	12
<b>4 Configuration of Different Scenarios .....</b>	<b>14</b>
4.1. Scenario 1: Single Data Call .....	14
4.2. Scenario 2: Single Data Call with ECM Device .....	15
4.3. Scenario 3: Multiplex Data Call .....	16
4.3.1. Rule Deletion .....	16
4.3.2. Default Routing .....	16
4.3.3. Default DNS Configuration .....	17
4.3.4. Access a Server via APN2 .....	17
4.3.4.1. DNS & Routing Configuration .....	17
4.3.4.2. Domain Name Resolution .....	17
4.3.4.3. Routing Configuration .....	18
4.4. Scenario 4: Multiplex Data Call with ECM (1) .....	18
4.5. Scenario 5: Multiplex Data Call with ECM (2) .....	19
4.5.1. Domain Name Resolution .....	19
4.5.2. Routing Settings .....	20
4.6. Scenario 6: Multiplex Data Call with ECM (3) .....	20
4.6.1. Rule Deletion .....	21
4.6.2. Routing Settings on AP Side .....	21
4.6.3. DNS Settings on AP Side .....	21
4.6.4. FIB Configuration .....	21
4.6.5. Policy-based Routing Configuration .....	22
4.6.6. Host DNS Server Settings .....	22
4.6.6.1. dnsmasq Configuration .....	22
4.6.6.2. DNS Routing Settings of APN1 .....	23

4.7.	Scenario 7: Multiplex Data Call with ECM (4).....	24
4.7.1.	Basic Settings.....	24
4.7.2.	Access a Server via APN1 on AP Side .....	24
4.7.2.1.	Domain Name Resolution .....	24
4.7.2.2.	Routing Settings.....	25
4.7.3.	Access a Server via APN2 by Host.....	25
4.7.3.1.	Domain Name Resolution .....	25
4.7.3.2.	Routing Settings .....	25
4.8.	Scenario 8: Multiplex Data Call with ECM (5).....	26
4.8.1.	Routing and DNS Settings on AP Side .....	26
4.8.2.	Host Policy-based Routing Settings .....	26
4.8.3.	DNS Resolution Settings .....	27
<b>5</b>	<b>Supplementary Instructions .....</b>	<b>28</b>
5.1.	APIs.....	28
5.2.	Commands.....	28
5.2.1.	PING.....	28
5.2.2.	NSLOOKUP .....	28
<b>6</b>	<b>Data Call APIs.....</b>	<b>29</b>
6.1.	Data Types .....	29
6.1.1.	ql_data_call_error_e .....	29
6.1.2.	ql_data_call_state_e .....	29
6.1.3.	ql_data_call_ip_family_e.....	29
6.1.4.	ql_apn_pdp_type_e.....	29
6.1.5.	ql_apn_auth_proto_e .....	30
6.1.6.	v4_address_status .....	30
6.1.7.	v6_address_status .....	30
6.1.8.	ql_data_call_state_s .....	30
6.1.9.	ql_data_call_s .....	31
6.1.10.	pkt_stats .....	31
6.1.11.	v4_info.....	31
6.1.12.	v6_info.....	31
6.1.13.	ql_data_call_info_s .....	32
6.1.14.	ql_apn_info_s.....	32
6.1.15.	ql_apn_add_s.....	32
6.1.16.	ql_apn_info_list_s .....	32
6.2.	Functions.....	33
6.2.1.	QL_Data_Call_Init.....	33
6.2.2.	QL_Data_Call_Destroy .....	33
6.2.3.	QL_Data_Call_Start .....	34
6.2.4.	QL_Data_Call_Stop .....	34
6.2.5.	QL_Data_Call_Info_Get.....	35
6.2.6.	QL_APN_Set.....	35
6.2.7.	QL_APN_Get .....	36

6.2.8.	QL_APN_Add.....	36
6.2.9.	QL_APN_Del.....	37
6.2.10.	QL_APN_Get_Lists.....	37
6.2.11.	QL_Data_Call_Init_Precondition.....	37
<b>7</b>	<b>Common Problems .....</b>	<b>39</b>
7.1.	Capture PCAP Logs.....	39
7.2.	Check iptables Table .....	39
7.3.	3GPP2/CDMA Cannot Dial-up Due to Authentication .....	39
7.4.	Data Call Failure .....	40
7.5.	Problem of Sending DNS Server IPv6 Address Request .....	40
<b>8</b>	<b>Appendix A References.....</b>	<b>41</b>

## Table Index

Table 1: Applicable Modules.....	8
Table 2: Related Documents .....	41
Table 3: Terms and Abbreviations .....	41

## Figure Index

Figure 1: Scenario 1 (Single Data Call) .....	14
Figure 2: Scenario 2 (Single Data Call with ECM Device).....	15
Figure 3: Scenario 3 (Multiplex Data Call) .....	16
Figure 4: Scenario 4 (Multiplex Data Call with ECM (1)) .....	18
Figure 5: Scenario 5 (Multiplex Data Call with ECM (2)) .....	19
Figure 6: Scenario 6 (Multiplex Data Call with ECM (3)) .....	20
Figure 7: Scenario 7 (Multiplex Data Call with ECM (4)) .....	24
Figure 8: Scenario 8 (Multiplex Data Call with ECM (5)) .....	26



# 1 Introduction

Quectel LTE Standard modules support QuecOpen® solution. This document mainly introduces the process of establishing the wireless data service, namely the process of data call of LTE Standard QuecOpen modules. The entire calling process needs to be performed in strict accordance with the sequence described in **Chapter 3**, especially for customers who are new to wireless service.

## 1.1. Applicable Modules

Table 1: Applicable Modules

Module Series	Module
EC2x series QuecOpen	EC25 series QuecOpen
	EC21 series QuecOpen
	EC20 R2.1 QuecOpen
EG9x series QuecOpen	EG95 series QuecOpen
	EG91 series QuecOpen
EG25-G QuecOpen	EG25-G QuecOpen

## 2 Network Card Connection

This chapter mainly introduces the use of USB-ECM and USB-RNDIS network cards.

- For USB-ECM network card connection, please refer to **document [1]**.
- For USB-RNDIS network card connection, please refer to **document [2]**.

## 3 Data Call on AP Side

The content introduced in this chapter is implemented based on the APIs included in the OpenLinux SDK.

### 3.1. Device Check

Before calling, a series of basic checks are required to determine whether the module is in a basic normal working state. The specific steps are as follows:

1. Connect the PC to the main UART port or the USB port of the module through a USB-to-serial cable.
2. Insert a (U)SIM card and an antenna into the device, and then power it on.
3. Check the device status in sequence by the following APIs.
  - a) Test (U)SIM card: `QL_MCM_SIM_GetCardStatus ()`
  - b) Detect signal strength: `QL_MCM_NW_GetSignalStrength ()`
  - c) Check network registration status: `QL_MCM_NW_GetRegStatus()`
  - d) Query operator: `QL_MCM_NW_GetRegStatus()`
  - e) Query network access technology: `QL_MCM_NW_GetRegStatus()`
  - f) Query call service status: `QL_Data_Call_Init_Precondition()`

### 3.2. APN Configuration

In general, for multiplex data call application scenarios, it is often necessary to set some special APNs for private network access. The parameters of each APN can be queried and configured by `QL_Apn_Get()` and `QL_Apn_Set()`. For more details, please refer to the `example_apn_v2.c` in OpenLinux SDK.

#### NOTES

1. The APN parameter must be configured before starting a data call. The configured parameters will be saved automatically and remain valid after rebooting.
2. Generally, there is no need to configure the parameters of username, password and authentication for public network APN.
3. Generally, whether the parameters of username, password and authentication for the private network APN needs to be set requires consulting the operator.

4. For APN settings under eHRPD network of CDMA, configure profile\_id as 0 forcibly, and transmit the APN username and password via the calling interface. The entering of profile\_id does not affect the data call under the HRPD network.

### 3.3. Single Data Call

This chapter mainly introduces the steps of single data call, without considering the configuration of route, FIB and DNS:

1. Set the value of `QL_Data_Call_Get_Default_Profile()` to be the same as that of profile\_id.
2. Perform one data call as **Chapter 3.4** and establish only one APN data channel.

### 3.4. Multiplex Data Call

#### 3.4.1. Data Call

```
//Initial and register callback function.
QL_Data_Call_Init(user_callback)

//Do not use the default routing and default FIB configured automatically.
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 multiplex data call, please note that the profile\_idx value does not need to be the same as that of *QL\_Data\_Call\_Get\_Default\_Profile ()*.

### 3.4.2. Routing and Other Parameters Configuration

The steps for setting parameters such as routing in different application scenarios involved in this sub-chapter are different. Please refer to **Chapter 4** for details.

## 3.5. 3GPP2/CDMA Data Call

Please follow the following steps to enable CDMA network data call:

1. Confirm whether the operator of the (U)SIM card used supports CDMA network.
2. Before calling, call *QL\_MCM\_NW\_GetRegStatus()* first to query whether the module is registered on the LTE or the CDMA network currently.
3. Configure profile\_id as 0 forcibly, and the multiplex data call is not supported by CDMA network.

```
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; //The value of profile_idx 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 = username;  
data1_call_paras.cdma_password = passwd;  
QI_Data_Call_Start(&data1_call_paras, & err_code1) //Current Linux system will show rmnet_data0.
```

**NOTE**

Whether the current network is HRPD or eHRPD does not affect 3GPP2/CDMA data call.

# 4 Configuration of Different Scenarios

## 4.1. Scenario 1: Single Data Call

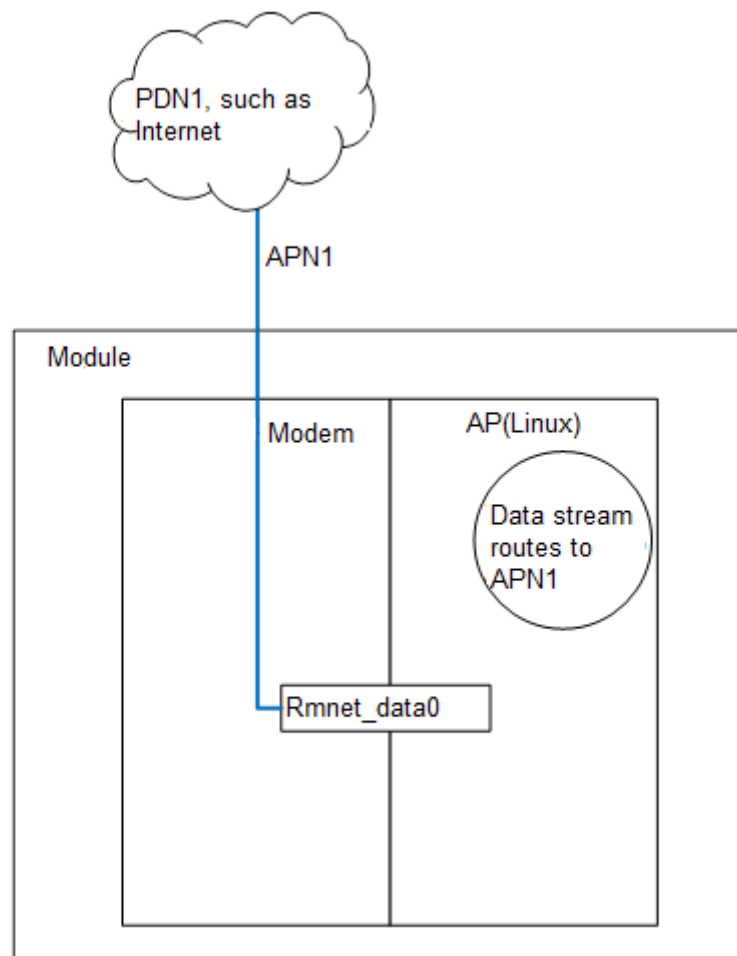
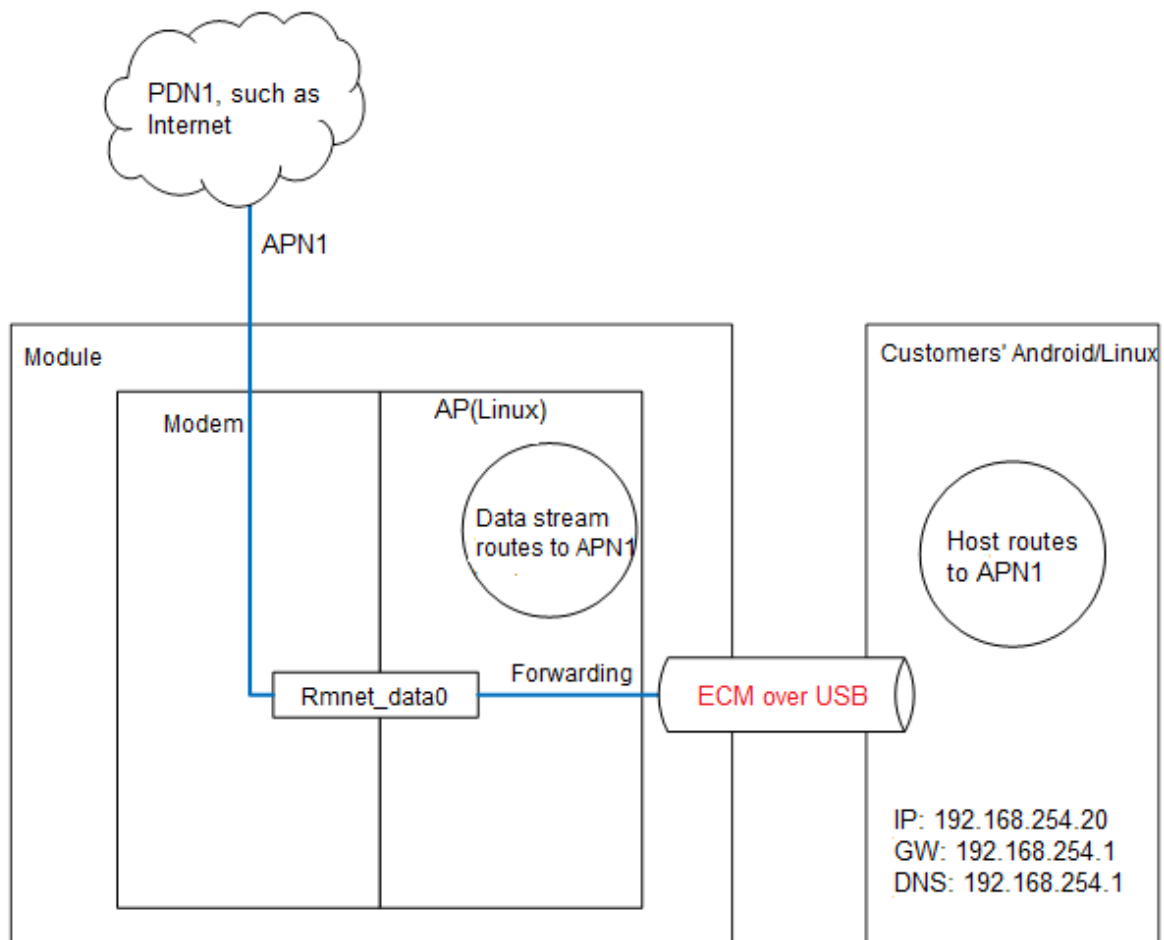


Figure 1: Scenario 1 (Single Data Call)

In this application scenario, there is no need for any routing, FIB and DNS configuration, and just perform correctly by following the calling steps described in **Chapter 3.3**.

## 4.2. Scenario 2: Single Data Call with ECM Device



**Figure 2: Scenario 2 (Single Data Call with ECM Device)**

In this application scenario, there is no need for any routing, FIB and DNS configuration, and just perform correctly by following the calling steps described in **Chapter 3** and **3.3**.



### 4.3. Scenario 3: Multiplex Data Call

This chapter mainly introduces the routing, FIB and DNS configuration in the multiplex data call application scenario.

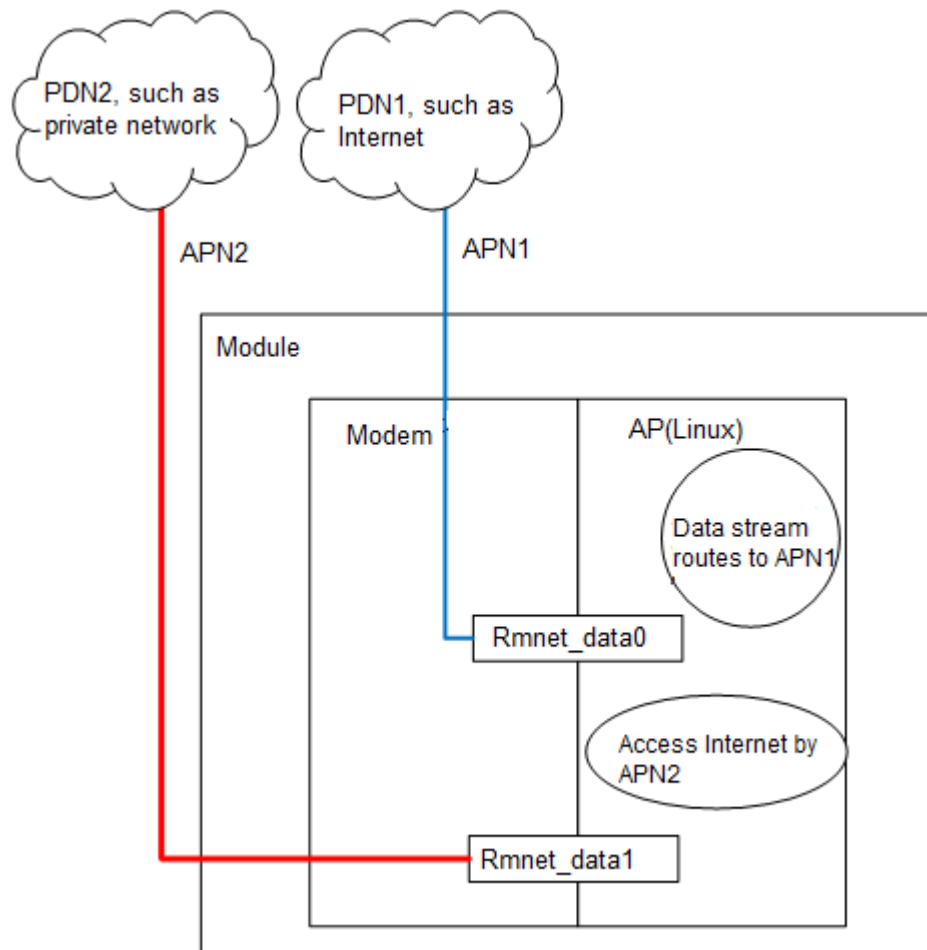


Figure 3: Scenario 3 (Multiplex Data Call)

#### 4.3.1. Rule Deletion

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

#### 4.3.2. Default Routing

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

```
//Set default routing, and take the address 10.112.7.176 as an example.
```

```
Method 1:
```

```
route add default dev rmnet_data0
```

```
Method 2:
```

```
route add -net 10.112.7.0/24 dev rmnet_data0
```

```
route add default gw 10.112.7.176
```

### 4.3.3. Default DNS Configuration

```
//Get APN1 DNS address.
```

```
QI_Data_Call_Info_Get(1, &info);
```

```
//Set default DNS server, and take the obtained addresses of primary: 211.138.180.2 and secondary:  
211.138.180.3 as example.
```

```
echo "nameserver 211.138.180.2" > /etc/resolv.conf
```

```
echo "nameserver 211.138.180.3" >> /etc/resolv.conf
```

### 4.3.4. Access a Server via APN2

#### 4.3.4.1. DNS & Routing Configuration

```
//Get APN2 DNS address.
```

```
QI_Data_Call_Info_Get(2, &info);
```

```
//Set default DNS server, and take the addresses of primary: 121.158.280.8 and secondary:  
121.158.280.9 as example.
```

```
ip route add 121.158.200.8/32 dev rmnet_data1
```

```
ip route add 121.158.200.9/32 dev rmnet_data1
```

#### 4.3.4.2. Domain Name Resolution

```
//Resolve domain name.
```

```
QL_nslookup( www.xxx.com, 121.158.280.8, IPV4, resolved_output)
```

121.158.280.8 is the DNS address of APN2.

#### 4.3.4.3. Routing Configuration

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

47.88.189.18 is the address of www.xxx.com, and 10.32.80.46 is the gateway of APN2 (obtained by `QI_Data_Call_Info_Get`)

### 4.4. Scenario 4: Multiplex Data Call with ECM (1)

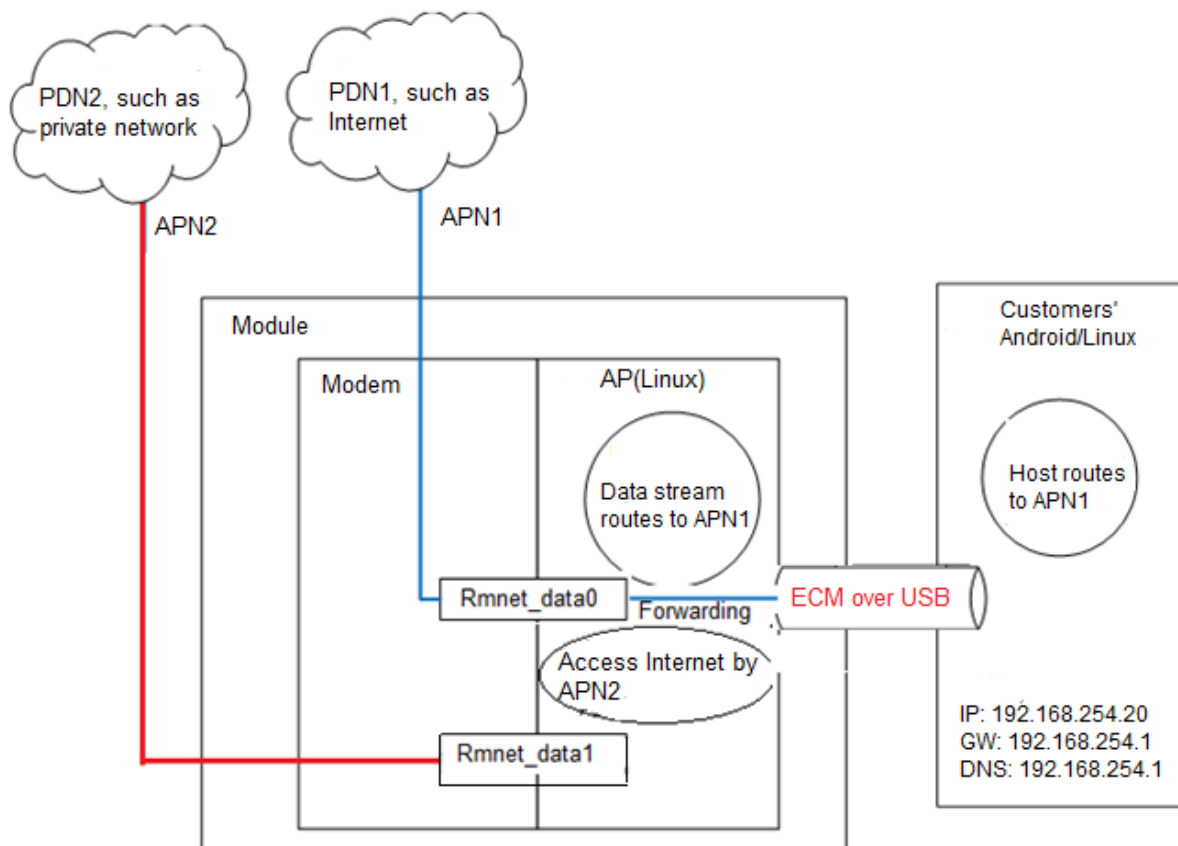


Figure 4: Scenario 4 (Multiplex Data Call with ECM (1))

The basic steps in this application scenario are basically the same as that of scenario 3 in **Chapter 4.3**, and only the default FIB needs to be opened before performing **Chapter 4.3.4**.

```
//Enable default forwarding.
```

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

```
//Set FIB.
```

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

## 4.5. Scenario 5: Multiplex Data Call with ECM (2)

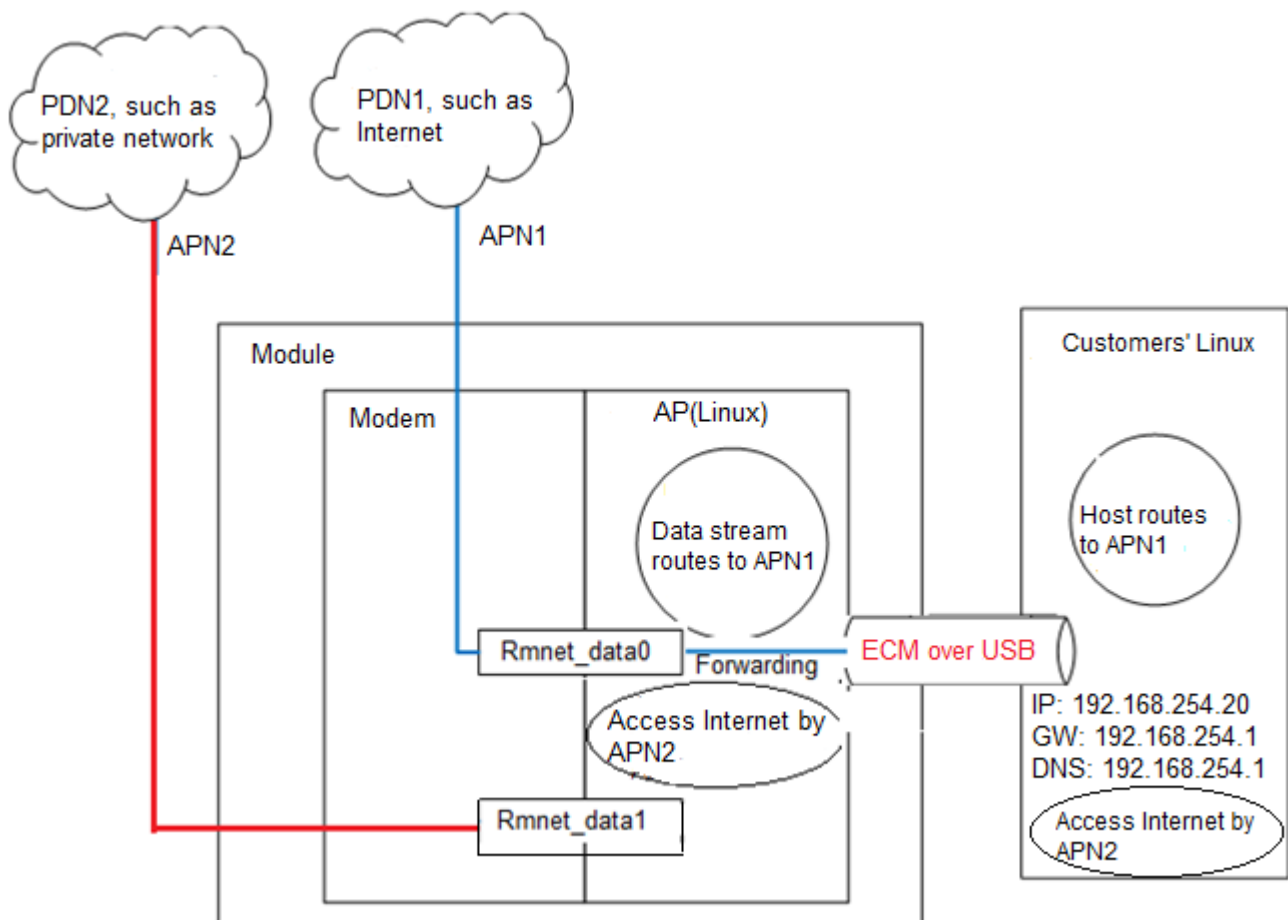


Figure 5: Scenario 5 (Multiplex Data Call with ECM (2))

The basic steps in this application scenario are basically the same as that of scenario 4 in **Chapter 4**. The difference is that the host accesses some servers through APN2. **Chapter 4.5.1** and **4.5.2** are reference design solutions.

### 4.5.1. Domain Name Resolution

Since the domain name resolution of the host is all resolved by the dnsmasq on module AP side through the DNS of APN1, the host needs to set up a server (a socket) to complete the DNS resolution and FIB configuration if the host accesses the server through domain name.

### 4.5.2. Routing Settings

If the host accesses a server through IP address directly, then configuring a forwarding rule on AP side as follows is needed only.

```
//If IP address of www.yyy.com is 47.88.189.189
//Set forwarding.
Method 1:
iptables -t nat -A POSTROUTING -d 47.88.189.189 -o rmnet_data1 -j MASQUERADE
Method 2:
iptables -t nat -A POSTROUTING -o rmnet_data1 -j MASQUERADE --random

//Set routing.
ip route add 47.88.189.189/32 dev rmnet_data1
```

### 4.6. Scenario 6: Multiplex Data Call with ECM (3)

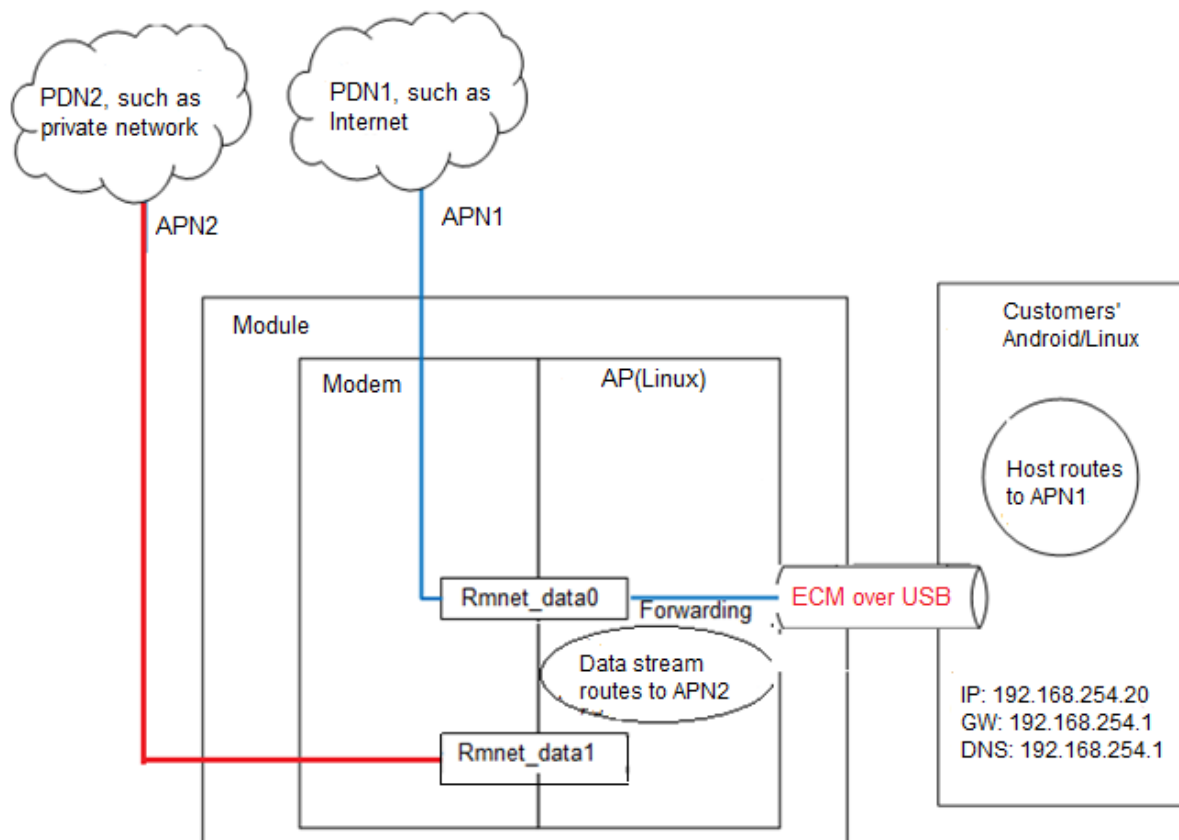


Figure 6: Scenario 6 (Multiplex Data Call with ECM (3))

#### 4.6.1. Rule Deletion

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

#### 4.6.2. Routing Settings on AP Side

```
//Get gateway address of APN2.
Ql_Data_Call_Info_Get(2, &info);
//Set default routing, and take the address 10.32.80.46 as an example.
Method 1:
route add default dev rmnet_data1
Method 2:
route add -net 10.32.80.0/24 dev rmnet_data1
route add default gw 10.32.80.46
```

#### 4.6.3. DNS Settings on AP Side

This step can realize the internal default DNS settings of the module.

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

//Set default DNS server (command), and take the obtained addresses of primary: 121.158.280.8 and
secondary: 121.158.280.9 as example.
echo "nameserver 121.158.280.8" > /etc/resolv.conf
echo "nameserver 121.158.280.9" >> /etc/resolv.conf
```

#### 4.6.4. FIB Configuration

FIB configuration is the key step for the host to access the external network.

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

#### 4.6.5. Policy-based Routing Configuration

The steps for setting policy-based routing on the AP side of the module are as follows:

1. Execute the following command to create RIB.

```
echo "200    rmnet_data_apn1" >> /etc/iproute2/rt_tables
```

2. Execute the following command to set policy-based routing.

```
ip rule add from 192.168.254.0/24 table 200    //192.168.254.0 is the network address of ECM device.
```

3. Execute the following command to add routing rules.

Method 1:

```
ip route add dev rmnet_data0 table 200
```

Method 2:

```
ip route add via 10.112.7.176 table 200    //10.112.7.176 is the gateway address.
```

#### 4.6.6. Host DNS Server Settings

If host DNS address is sent by the module, which means resolving DNS address obtained from base station by module, please perform the following steps.

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.

##### 4.6.6.1. dnsmasq Configuration

1. Modify file */etc/dnsmasq.conf*. And restart the module after synchronization.

```
# Change this line if you want dns to get its upstream servers from
# somewhere other than /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
```

2. Create file `/etc/dnsmasq_resolv.conf`, and add DNS server address of APN1.

```
//Get APN1 DNS address.
QI_Data_Call_Info_Get(1, &info);
//Set default DNS server, and take the addresses of primary: 211.138.180.2 and secondary:
211.138.180.3 as example.
echo "nameserver 211.138.180.2" > /etc/dnsmasq_resolv.conf
echo "nameserver 211.138.180.3" >> /etc/dnsmasq_resolv.conf
```

#### 4.6.6.2. DNS Routing Settings of APN1

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

//Set default DNS server, and take the addresses of primary: 211.138.180.2 and secondary:
211.138.180.3 as example.
ip route add 211.138.180.2/32 dev rmnet_data0
ip route add 211.138.180.3/32 dev rmnet_data0
```



## 4.7. Scenario 7: Multiplex Data Call with ECM (4)

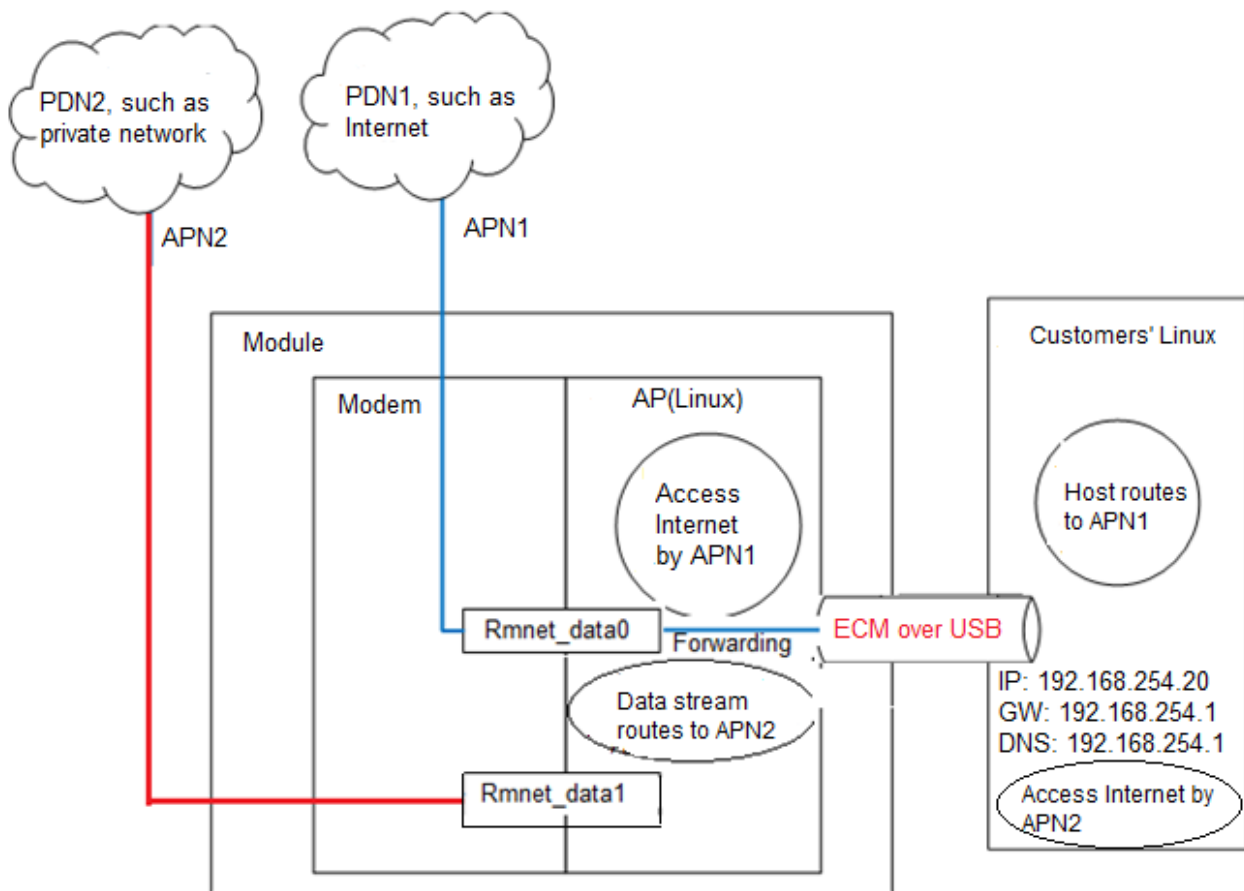


Figure 7: Scenario 7 (Multiplex Data Call with ECM (4))

### 4.7.1. Basic Settings

Please refer to scenario 6 in **Chapter 4.6** to complete all basic settings.

### 4.7.2. Access a Server via APN1 on AP Side

#### 4.7.2.1. Domain Name Resolution

```
//Resolve domain resolution.
QL_nslookup( www.xxx.com, 211.138.180.2, IPV4, resolved_output)
```

211.138.180.2 is the DNS address of APN1.

#### 4.7.2.2. Routing Settings

```
//Set the module's internal access rules.  
route add -net 47.88.189.189/32 gw 10.112.7.176 dev rmnet_data0
```

47.88.189.189 is the address of [www.xxx.com](http://www.xxx.com), and 10.112.7.176 is the gateway of APN1 (obtained by *Ql\_Data\_Call\_Info\_Get*).

#### 4.7.3. Access a Server via APN2 by Host

The following **Chapter 4.7.3.1** and **4.7.3.2** are reference design solution.

##### 4.7.3.1. Domain Name Resolution

The domain name resolution of the host is all resolved by the dnsmasq on the module AP side through the DNS of APN1, even if a public DNS server is used. If the host needs to access the network through a domain name, a server (a socket) must be established on the AP side to complete DNS resolution and FIB configuration.

##### 4.7.3.2. Routing Settings

If the host accesses a server through IP address directly, then configuring a forwarding rule on AP side as follows is needed only.

```
//If IP address of www.yyy.com is 47.88.189.189  
//Set forwarding.  
Method 1:  
iptables -t nat -A POSTROUTING -d 47.88.189.189 -o rmnet_data1 -j MASQUERADE  
Method 2:  
iptables -t nat -A POSTROUTING -o rmnet_data1 -j MASQUERADE --random  
  
//Set routing.  
ip route add 47.88.189.189/32 dev rmnet_data1 table 200  
Or set routing based on policy-based routing.  
ip rule add to 47.88.189.189 table main
```

## 4.8. Scenario 8: Multiplex Data Call with ECM (5)

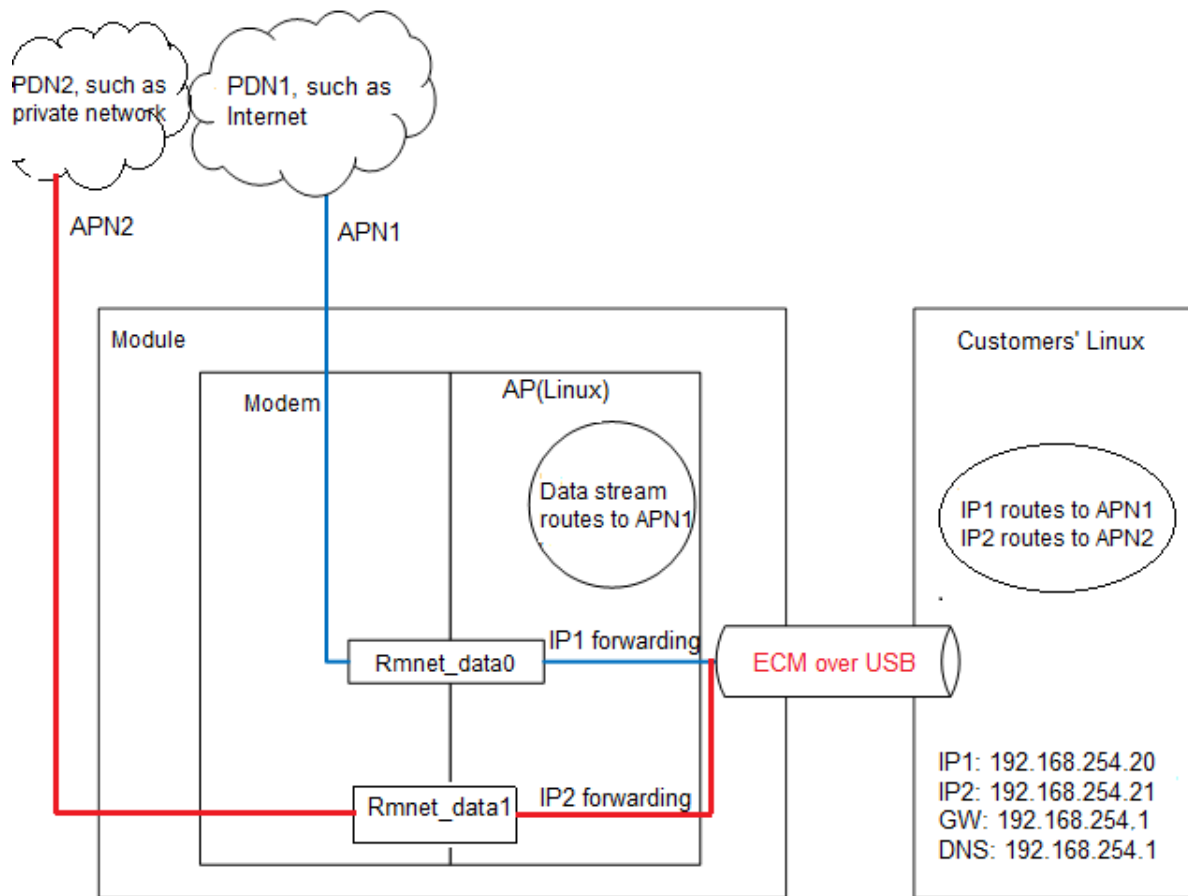


Figure 8: Scenario 8 (Multiplex Data Call with ECM (5))

### 4.8.1. Routing and DNS Settings on AP Side

Please refer to **Chapter 4.6.1** and **Chapter 4.6.3**.

### 4.8.2. Host Policy-based Routing Settings

Please refer to **Chapter 4.6.4** and **Chapter 4.6.5**.

### **4.8.3. DNS Resolution Settings**

DNS resolution settings should follow the DNS addresses delivered by the corresponding APNs, and the following routing rules can be created in the policy-based routing table:

1. When accessing the DNS server address delivered by APN1, the DNS server address is routed to the APN1 gateway.
2. When accessing the DNS server address delivered by APN2, the DNS server address is routed to the APN2 gateway.

# 5 Supplementary Instructions

This chapter provides additional instructions about the DNS of Linux system, and gives guidance on domain name resolution of private network.

## 5.1. APIs

1. The Linux standard function *gethostbyname()* uses the DNS address in *resolve.conf* to resolve IPv4 address or directly resolve the address from */etc/hosts*.
2. The Linux standard function *getaddrinfo()* uses the DNS address in *resolve.conf* to resolve both IPv4 and IPv6 addresses simultaneously or directly resolve the addresses from */etc/hosts*.
3. Quectel function *QL\_nslookup()* uses the specified DNS address to resolve IPv4 and IPv6 addresses separately.

### NOTE

None of the above APIs can guarantee a one-time resolving success. It needs to be called repeatedly 2 to 3 times. This is because the DNS protocol transport layer uses the UDP protocol, and it is normal for packets to be lost. In addition, the network connectivity of the DNS server itself is also very critical.

## 5.2. Commands

### 5.2.1. PING

After executing this command, the returned domain name resolution is based on the Linux system default DNS.

### 5.2.2. NSLOOKUP

After executing this command, the module can only use *resolve.conf*'s default DNS for resolution, which is different from using this command on the PC.

# 6 Data Call APIs

## 6.1. Data Types

### 6.1.1. ql\_data\_call\_error\_e

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

### 6.1.2. 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;
```

### 6.1.3. 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;
```

### 6.1.4. 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;
```

### 6.1.5. 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;
```

### 6.1.6. 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. */  
};
```

### 6.1.7. 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. */  
};
```

### 6.1.8. 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;
```

### 6.1.9. 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;
```

### 6.1.10. 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. */
};
```

### 6.1.11. 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 */
};
```

### 6.1.12. 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 */
};
```

#### 6.1.13. 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;
```

#### 6.1.14. 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;
```

#### 6.1.15. 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;
```

#### 6.1.16. ql\_apn\_info\_list\_s

```
typedef struct {
    int cnt;
```

```
    ql_apn_info_s apn[QL_APN_MAX_LIST];  
} ql_apn_info_list_s;
```

## 6.2. Functions

### 6.2.1. QL\_Data\_Call\_Init

This function is used to initial data call and register callback function.

- **Prototype**

```
int QL_Data_Call_Init (ql_data_call_evt_cb_t evt_cb)
```

- **Parameter**

*evt\_cb*:  
[In] callback function.

- **Return Value**

0: Success  
-1: Error

### 6.2.2. QL\_Data\_Call\_Destroy

This function is used to release data call and deregister callback function.

- **Prototype**

```
void QL_Data_Call_Destroy (void)
```

- **Parameter**

None

- **Return Value**

0: Success  
-1: Error

### 6.2.3. QL\_Data\_Call\_Start

This function is used to start a data call.

- **Prototype**

```
int QL_Data_Call_Start (ql_data_call_s *data_call, ql_data_call_error_e *err)
```

- **Parameter**

*data\_call:*

[In] Data call parameters.

*error:*

[Out] Error code returned by data call.

- **Return Value**

0: Success

-1: Error

### 6.2.4. QL\_Data\_Call\_Stop

This function is used to stop a data call.

- **Prototype**

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

- **Parameter**

*profile\_idx:*

[In] UMTS/CDMA profile ID.

*ip\_family:*

[In] IP version

*error:*

[Out] Error code returned by data call.

- **Return Value**

0: Success

-1: Error

### 6.2.5. QL\_Data\_Call\_Info\_Get

This function is used to get data call information.

- **Prototype**

```
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)
```

- **Parameter**

*profile\_idx:*

[In] UMTS/CDMA profile ID.

*ip\_family:*

[In] IP version.

*info:*

[Out] Data call information.

*error:*

[Out] Error code returned by data call.

- **Return Value**

0: Success

-1: Error

### 6.2.6. QL\_APN\_Set

This function is used to change the settings in a configured profile. If the profile does not exist, a new configuration file will be created with the settings configured using this API.

- **Prototype**

```
int QL_APN_Set (ql_apn_info_s *apn)
```

- **Parameter**

*apn:*

[In] Profile information.

- **Return Value**

0: Success

-1: Error

### 6.2.7. QL\_APN\_Get

This function is used to retrieve the settings from a configured profile.

- **Prototype**

```
int QL_APN_Get (unsigned char profile_idx, ql_apn_info_s *apn)
```

- **Parameter**

*profile\_idx*:

[In] UMTS/CDMA profile ID.

*apn*:

[Out] Profile information.

- **Return Value**

0: Success

-1: Error

### 6.2.8. QL\_APN\_Add

This function is used to add a new APN configured profile.

- **Prototype**

```
int QL_APN_Add(ql_apn_add_s *apn, unsigned char *profile_idx)
```

- **Parameter**

*profile\_idx*:

[Out] UMTS/CDMA profile ID.

*apn*:

[In] Profile information.

- **Return Value**

0: Success

-1: Error

### 6.2.9. QL\_APN\_Del

This function is used to delete a configured profile.

- **Prototype**

```
int QL_APN_Del(unsigned char profile_idx)
```

- **Parameter**

*profile\_idx:*

[In] UMTS/CDMA profile ID.

- **Return Value**

0: Success

-1: Error

### 6.2.10. QL\_APN\_Get\_Lists

This function is used to retrieve the settings from a configured profile list.

- **Prototype**

```
int QL_APN_Get_Lists(ql_apn_info_list_s *apn_list)
```

- **Parameter**

*apn\_list:*

[Out] Profile list information.

- **Return Value**

0: Success

-1: Error

### 6.2.11. QL\_Data\_Call\_Init\_Precondition

This function is used to get the running status of the Quectel Manager service.

- **Prototype**

```
QL_Data_Call_Init_Precondition
```

- **Parameter**

None

- **Return Value**

0: the service is working properly.

-1: the service is not working properly.

# 7 Common Problems

## 7.1. Capture PCAP Logs

1. Enter the following command to capture the needed log.

Capture the log of all network interfaces:

```
tcpdump -i any -p -vv -s 0 -w ./capture1.pcap &
```

Capture the log of specified network interface:

```
tcpdump -i rmnet_data0 -p -vv -s 0 -w ./capture1.pcap &
```

2. Run the program.
3. Interrupt tcpdump command execution, and upload *capture1.pcap* file.

## 7.2. Check iptables Table

1. Execute the following command to check nat table:

```
iptables -nvt nat -L
```

2. Execute the following command to check filter table:

```
iptables -nvt filter -L
```

## 7.3. 3GPP2/CDMA Cannot Dial-up Due to Authentication

Please follow the steps below to solve the problem that 3GPP2/CDMA cannot dial-up due to authentication.



1. Check modem side.

Set parameters as follows:

```
AT+QCTPWDCFG="<username>","<userpasswd>"  
AT+QCFG="cdmaruim",1
```

Test data call:

```
AT+QIACT=1 //ERROR will be returned if failed to test, and is mostly due to the  
            incorrect username and password.
```

2. Check AP side.

If there is no problem in the first step check, please check whether profile\_id is 0 and the authentication parameters are correct. For details, refer to **Chapter 3.2** and **3.5**.

## 7.4. Data Call Failure

The probabilistic data call failure may occur when customers' APP runs automatically after module's booting up. The inspection found that if all the items like the network registration is normal, the data call will be successful in another try.

The main reason for this problem is that the service process has not been started and completed when dialing. And whether the service has been started and completed can be queried by *QL\_Data\_Call\_Init\_Precondition()*.

## 7.5. Problem of Sending DNS Server IPv6 Address Request

When the module performs data call based on IPv4&IPv6, the problem of sending DNS server IPv6 address request may occur if the current network does not support IPv6.

The current solution to this problem is to specify that only IPv4 is used during a data call, which can be realized by specifying *ip\_family* in *ql\_data\_call\_s* as *QL\_DATA\_CALL\_TYPE\_IPV4* when calling function *QL\_Data\_Call\_Start*,

## 8 Appendix A References

**Table 2: Related Documents**

SN	Document Name	Remark
[1]	Quectel_EC2x&EG9x&EG25-G&_Series_QuecOpen_ECM_User_Guide	USB-ECM user guide applicable for EC2x&EG9x&EG25-G series QuecOpen
[2]	Quectel_EC2x&EG9x&EG25-G_Series_QuecOpen_RNDIS_User_Guide	USB-RNDIS user guide applicable for EC2x&EG9x&EG25-G series QuecOpen

**Table 3: Terms and Abbreviations**

Abbreviation	Description
3GPP2	3rd Generation Partnership Project 2
API	Application Programming Interface
APN	Access Point Name
APP	Application
CDMA	Code Division Multiple Access
DNS	Domain Name System
ECM	Ethernet Networking Control Model
eHRPD	Evolved High Rate Package Data
EPS	Evolved Packet System
FIB	Forward Information dataBase
HRPD	High Rate Packet Data
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6

---

LTE	Long Term Evolution
PDP	Packet Data Protocol
RNDIS	Remote Network Driver Interface Specification
SDK	Software Development Kit
(U)SIM	(Universal) Subscriber Identification Module
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus
UMTS	Universal Mobile Telecommunications System

---