EC2X&AG35-QuecOpen
数据拨号应用指导

LTE 系列

版本：EC2X&AG35-QuecOpen_数据拨号应用指导_V2.5

日期：2018-04-08

状态：临时文件

上海移远通信技术股份有限公司始终以为客户提供最及时、最全面的服务为宗旨。如需任何帮助，请随时联系我司上海总部，联系方式如下：

上海移远通信技术股份有限公司
上海市徐汇区虹梅路 1801 号宏业大厦 7 楼　　邮编：200233
电话：+86 21 51086236　　邮箱：info@quectel.com

或联系我司当地办事处，详情请登录：
http://quectel.com/cn/support/sales.htm

如需技术支持或反馈我司技术文档中的问题，可随时登陆如下网址：
http://quectel.com/cn/support/technical.htm
或发送邮件至：support@quectel.com

前言

上海移远通信技术股份有限公司提供该文档内容用以支持其客户的产品设计。客户须按照文档中提供的规范、参数来设计其产品。由于客户操作不当而造成的人身伤害或财产损失，本公司不承担任何责任。在未声明前，上海移远通信技术股份有限公司有权对该文档进行更新。

版权申明

本文档版权属于上海移远通信技术股份有限公司，任何人未经我司允许而复制转载该文档将承担法律责任。

版权所有 ©上海移远通信技术股份有限公司 2018，保留一切权利。
Copyright © Quectel Wireless Solutions Co., Ltd. 2018.

文档历史

修订记录

版本	日期	作者	变更表述
1.0	2017-08-15	钱润生	初始版本
1.1	2017-12-04	钱润生	增加 DNS 解析章节
1.2	2017-12-05	钱润生	增加 MODEM 状态检查步骤
2.0	2017-12-12	钱润生	新的数据拨号接口
2.1	2017-12-26	钱润生	补充 DNS 解析常见问题
2.2	2018-01-25	钱润生	增加关于 CDMA 网络拨号注意事项
2.3	2018-03-10	钱润生	更改文档《多路拨号快速解决方案》为《数据拨号应用指导》 校正 API 接口规范。 增加 APN 新接口。
2.4	2018-03-19	钱润生	对 3GPP2 的 HDPR 和 HDPR2 网络拨号添加注意事项说明
2.5	2018-04-08	钱润生	增加多种网络应用场景解决方案。

目录

文档历史 2
目录 3
表格索引 5
图片索引 6
1 介绍 7
2 网卡连接 8
3 AP 侧拨号 9
3.1. 设备检查 9
3.2. APN 配置 9
3.3. 单路拨号 9
3.4. 多路拨号 9
3.4.1. 拨号 9
3.4.2. 路由等参数设置 10
3.5. 3GPP2/CDMA 网络拨号 10
4 场景 1：单路 12
5 场景 2：单路+ECM 设备 13
6 场景 3：多路 14
6.1. 规则清空 14
6.2. 默认路由 14
6.3. 默认 DNS 设置 15
6.4. 访问某个 sever 通过 APN2 15
6.4.1. DNS 路由配置 15
6.4.2. 域名解析 15
6.4.3. 路由配置 15
7 场景 4：多路+ECM（1） 16
8 场景 5：多路+ECM（2） 17
8.1.1. 域名解析 17
8.1.2. 路由设置 17
9 场景 6：多路+ECM（3） 19
9.1. 规则清空 19
9.2. AP 侧路由设置 19
9.3. AP 侧 DNS 设置 20
9.4. 转发表设置 20
9.5. 策略路由设置 20
9.6. 上位机 DNS 服务器设置 20
9.6.1. dnsmasq 配置 21
9.6.2. APN1 的 DNS 路由设置 21

10 场景 7：多路+ECM（4） **22**
10.1. 基本设置 22
10.2. AP 侧访问某个 sever 通过 APN1 22
10.2.1. 域名解析 22
10.2.2. 路由设置 23
10.3. 上位机访问某个 sever 通过 APN2 23
10.3.1. 域名解析 23
10.3.2. 路由设置 23
11 域名解析补充说明 **24**
12 数据业务 API **25**
12.1. 数据类型 25
12.1.1. ql_data_call_error_e 25
12.1.2. ql_data_call_state_e 25
12.1.3. ql_data_call_ip_family_e 25
12.1.4. ql_apn_pdp_type_e 25
12.1.5. ql_apn_auth_proto_e 25
12.1.6. v4_address_status 26
12.1.7. v6_address_status 26
12.1.8. ql_data_call_state_s 26
12.1.9. ql_data_call_s 26
12.1.10. pkt_stats 27
12.1.11. v4_info 27
12.1.12. v6_info 27
12.1.13. ql_data_call_info_s 28
12.1.14. ql_apn_info_s 28
12.1.15. ql_apn_add_s 28
12.1.16. ql_apn_info_list_s 28
12.2. 函数 29
12.2.1. QL_Data_Call_Init 29
12.2.2. QL_Data_Call_Destroy 29
12.2.3. QL_Data_Call_Start 29
12.2.4. QL_Data_Call_Stop 30
12.2.5. QL_Data_Call_Info_Get 30
12.2.6. QL_APN_Set 30
12.2.7. QL_APN_Get 31
12.2.8. QL_APN_Add 31
12.2.9. QL_APN_Del 31
12.2.10. QL_APN_Get_Lists 32
13 常见问题 **33**
14 附录 A 参考文档及术语缩写 **34**

表格索引

表 1:参考文档 34
表 2:术语缩写 34

图片索引

图 1:场景 1 12
图 2:场景 2 13
图 3:场景 3 14
图 4:场景 4 16
图 5:场景 5 17
图 6:场景 6 19
图 7:场景 7 22

1 介绍

本篇文档主要描述无线数据业务的建立过程，即 Data Call。整个拨号过程，请严格按照文档的标题顺序，特别对于初次接触无线业务的客户。

第 2 章简单介绍 USB-ECM 网卡的使用

第 3 章主要描述拨号，不去描述路由、转发表和 DNS 配置。

第 4 章至第 10 章去描述各个应用场景的路由、转发表和 DNS 配置。

2 网卡连接

（1） 关于 USB-ECM 网卡连接，请直接参考 《Quectel_EC2X&AG35-QuecOpen_ECM_使用说明》。

（2） 关于 USB-RNDIS 网络连接，请直接参考《Quectel_EC2X&AG35-QuecOpen_RNDIS_使用说明》。

3 AP 侧拨号

本部分基于 openlinux SDK API 实现

3.1. 设备检查

拨号之前客户需要做一系列基本检查，开判断模块是否处在基本的正常工作状态，具体步骤如下：
（1）PC 通过 USB 转串口线连上模块的 MAIN UART 口或者 USB AT 口
（2）请插上 SIM 卡和天线，并上电
（3）通过 API 按顺序检查如下状态

```
检测 SIM 卡:    QL_SIM_GetState()
检测信号强度：QL_NW_GetRegState()
检测模块注网：QL_NW_GetRegState()
查询运营商：    QL_NW_GetNetworkNameMccMnc()
查询网络接入技术：QL_NW_GetRegState()
```

3.2. APN 配置

一般来说，对于多路拨号的使用场景，经常需要设置一些特殊 APN 用于专网访问。各路 APN 的参数查询和配置可以使用 Ql_Apn_Get()和 Ql_Apn_Set()，例子请参照 example_apn_v2.c。

备注

（1）APN 参数设置一定要在拨号之前设置，同时设置完自动保存，下次开机还是有效。
（2）通常公网 APN 不需要设置 usrname 和 password 以及鉴权参数
（3）通常专网 APN 是否需要设置 usrname 和 password 以及鉴权参数，需要咨询运营商
（4）本模块对于 CDMA 的 eHRPD 网络的 apn 设置，强制 profile_id 为 0，usrname 和 password 在拨号接口传入。本参数的写入也不影响 HPRD 网络的拨号。

3.3. 单路拨号

（1） 请设置 QL_Data_Call_Set_Default_Profile ()的值跟请求拨号的 profile id 值一样
（2） 参照 3.4 执行一次拨号。
（3） 不需要考虑路由、转发表和 DNS 的配置。

3.4. 多路拨号

3.4.1. 拨号

```
//初始化并注册回调函数;
Ql_Data_Call_Init(user_callback)
```

```
//不使用自动配置默认路由和默认转发
QL_Data_Call_Set_Default_Profile (8)
//建立第 1 路数据业务通道

Int err_code1;
ql_data_call_s    data1_call_paras;

data1_call_paras.profile_idx = 1;//第 1 路
data1_call_paras.ip_family = QL_DATA_CALL_IPV4;//只拨 IPV4
data1_call_paras.reconnet = true; //打开自动重连

Ql_Data_Call_Start(&data1_call_paras,    & err_code1)//…当前 Linux 系统会出现 rmnet_data0

//建立第 2 路数据业务通道
Int err_code2;
ql_data_call_s    data2_call_paras;

data2_call_paras.profile_idx = 2;// //第 2 路
data2_call_paras.ip_family = QL_DATA_CALL_IPV4;
data2_call_paras.reconnet = true;

Ql_Data_Call_Start(&data2_call_paras,    & err_code2)//…当前 Linux 系统会出现 rmnet_data1

//建立第 3 路数据业务通道
Int err_code3;
ql_data_call_s    data3_call_paras;

data3_call_paras.profile_idx = 3; //第 3 路
data3_call_paras.ip_family = QL_DATA_CALL_IPV4;
data3_call_paras.reconnet = true;

Ql_Data_Call_Start(&data3_call_paras,    & err_code3)//…当前 Linux 系统会出现 rmnet_data2
```

备注

多路拨号需要注意 profile_idx 值不要 QL_Data_Call_Get_Default_Profile ()一样。

3.4.2. 路由等参数设置

本部分牵涉的不同应用场景，设置步骤有所不同，请参照后面的各种应用场景章节。

3.5. 3GPP2/CDMA 网络拨号

如何启用 CDMA 网络拨号：

（1）确定所用的 SIM 卡运营商是否支持 CDMA 网络。
（2）拨号之前，调用 QL_NW_GetRegState()先查询当前模块注册在 LTE 网络还是 CDMA 网络。
（3）profile_id 强制为 0，同时 CDMA 网络不支持多路拨号。
（4）不用关心当前是 HPRD 还是 eHRPD 网络。

```
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;//必须为 0
data1_call_paras.ip_family = QL_DATA_CALL_IPV4;//只拨 IPV4
data1_call_paras.reconnet = true; //打开自动重连
data1_call_paras.cdma_username = usrname;
data1_call_paras.cdma_password = passwd;
Ql_Data_Call_Start(&data1_call_paras,    & err_code1)//…当前 Linux 系统会出现 rmnet_data0
```

4 场景 1：单路

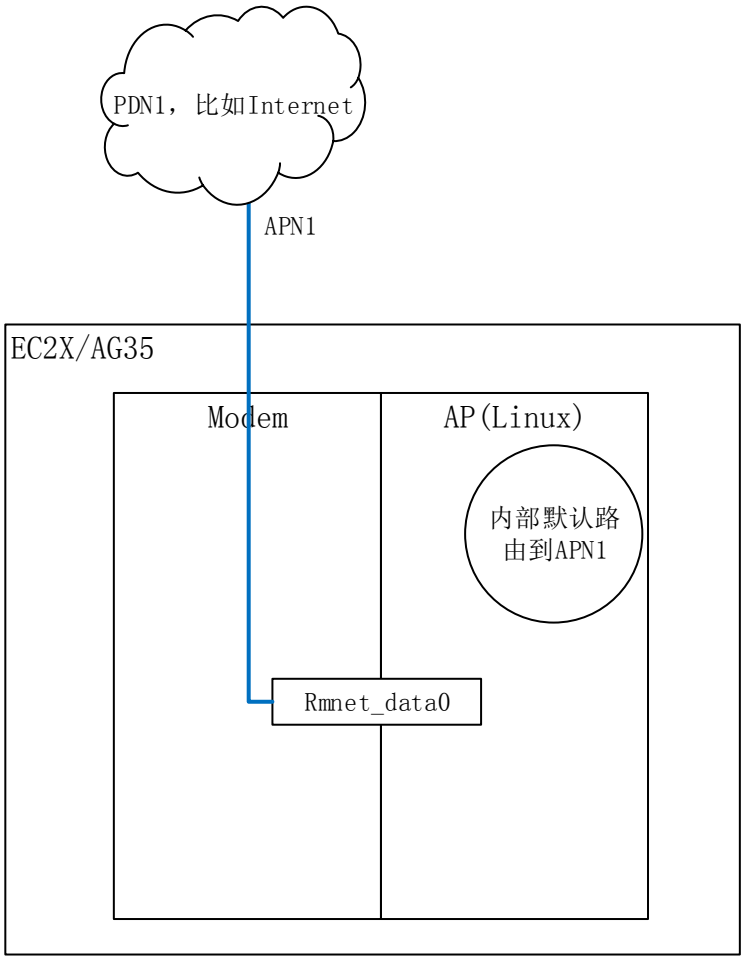


图1:场景 1

此应用场景，客户无需配置任何路由、转发表和 DNS 配置，只需要按照 3.3 节拨号步骤正确执行即可。

5 场景 2：单路+ECM 设备

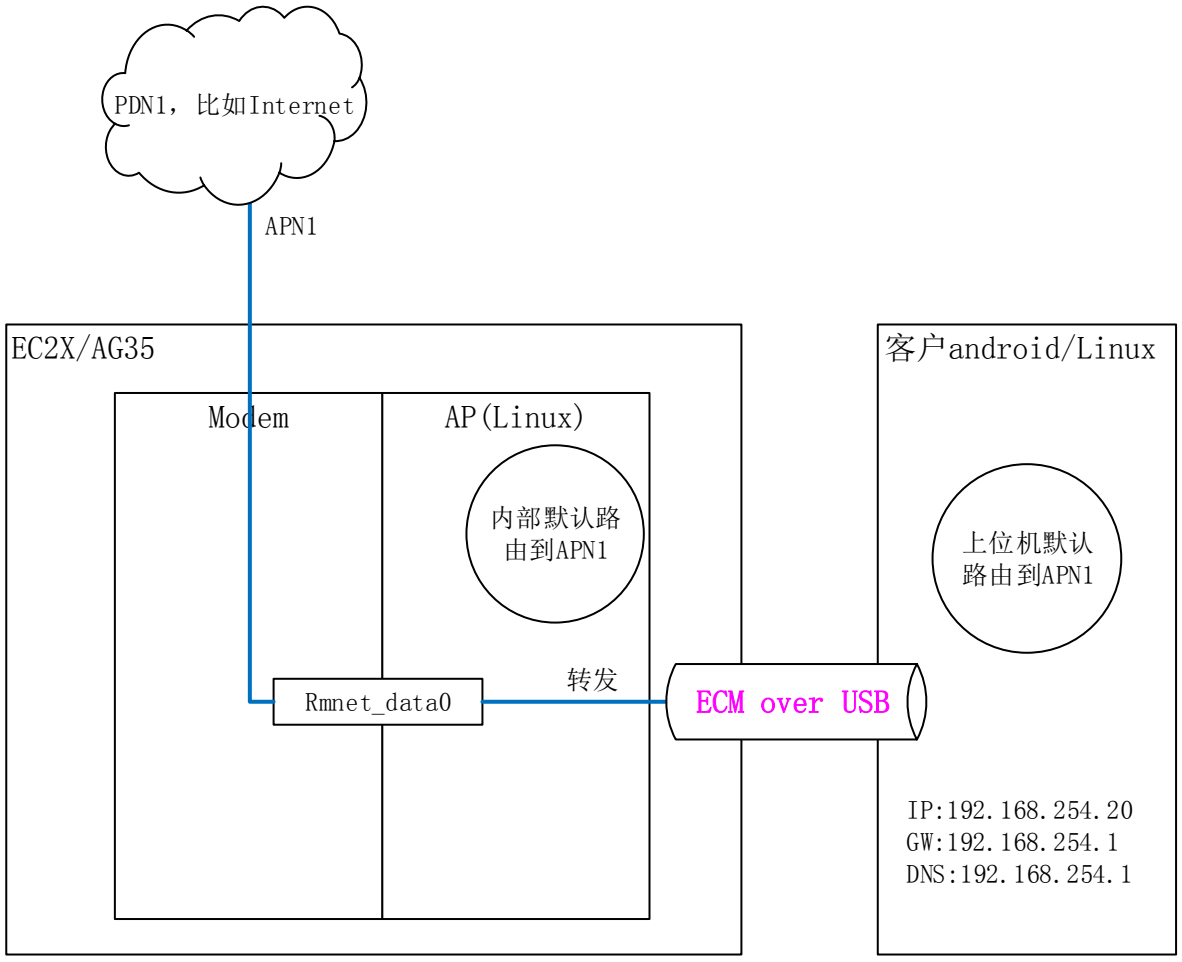


图2:场景 2

此应用场景，客户无需配置任何路由、转发表和 DNS 配置，只需要按照第 3 章和 3.3 节拨号步骤正确执行即可。

6 场景 3：多路

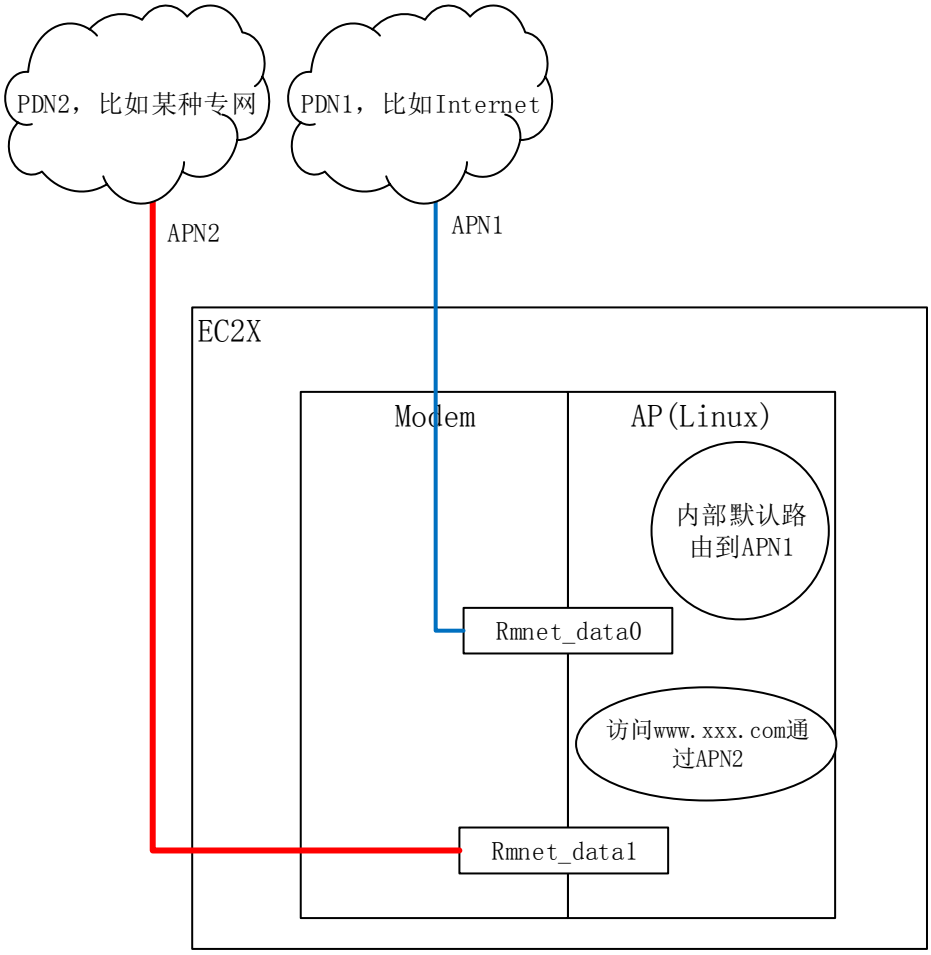


图3:场景 3

6.1. 规则清空

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

6.2. 默认路由

```
//获取 APN1 网关地址
QI_Data_Call_Info_Get(2, &info);

//设置默认路由，假设获得的地址为 10.112.7.176
route add default dev rmnet_data0
或者
```

```
route add –net 10.112.7.0/24 dev rmnet_data0
route add default gw 10.112.7.176
```

6.3. 默认 DNS 设置

```
//获取 APN1 DNS 地址（API）
Ql_Data_Call_Info_Get(1, &info);

//设置默认 DNS 服务器(命令)，假设获得的地址为 primary: 211.138.180.2 和 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. 访问某个 sever 通过 APN2

6.4.1. DNS 路由配置

```
//获取 APN2 DNS 地址（API）
Ql_Data_Call_Info_Get(2, &info);

//设置 DNS 路由(命令), 假设获得的地址为 primary: 121.158.280.8 和 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. 域名解析

```
//SDK API，用法如下
QL_nslookup( www.xxx.com, 121.158.280.8, IPV4, resolved_output)，其中 121.158.280.8 是 APN2 的
DNS 地址
```

6.4.3. 路由配置

```
route add –net 47.88.189.189/32 gw 10.32.80.46 dev rmnet_data1
其中 47.88.189.189 是 www.xxx.com 地址, 10.32.80.46 是 APN2 的网关(Ql_Data_Call_Info_Get 获得)
```

7 场景 4：多路+ECM（1）

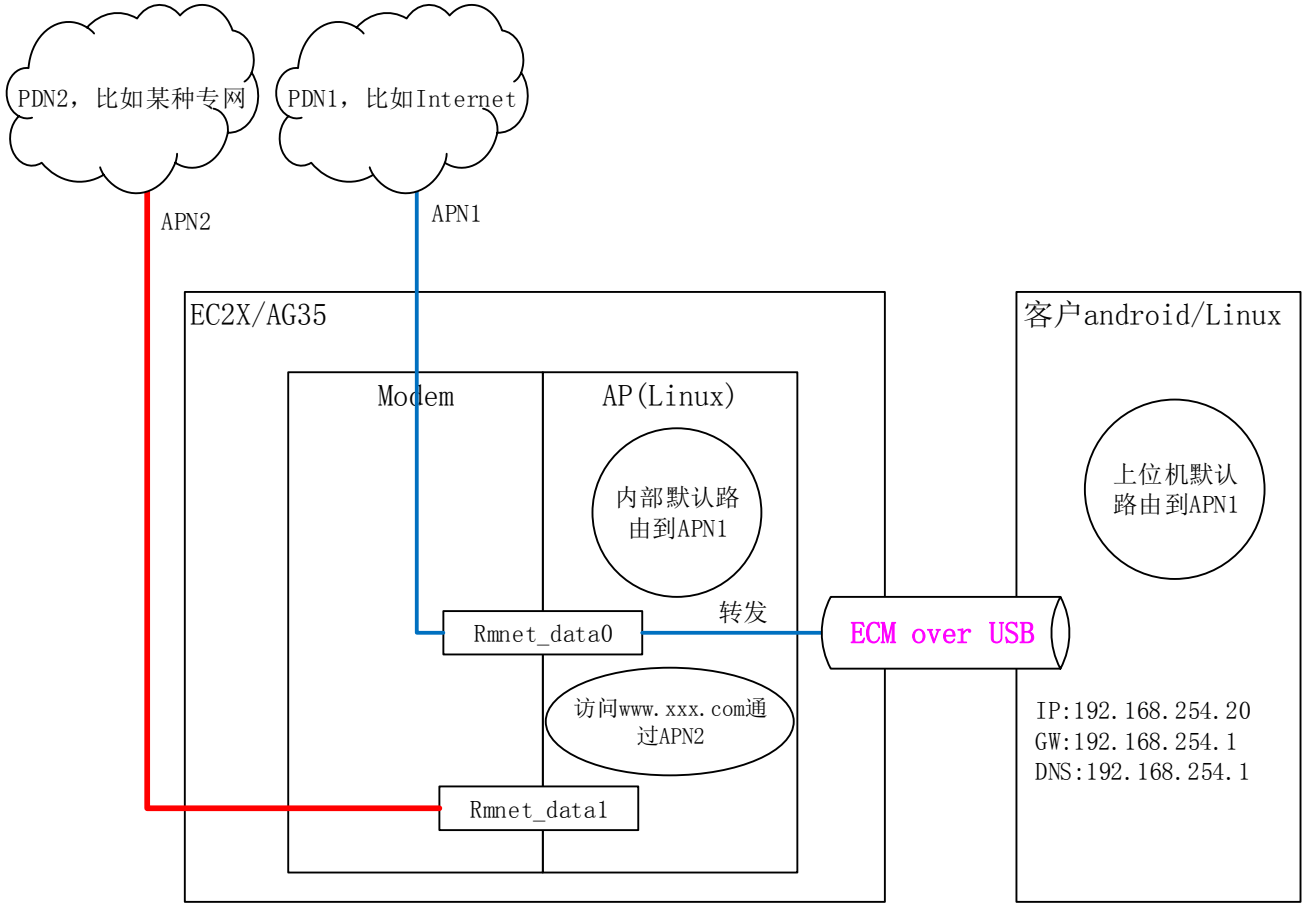


图4:场景 4

此应用场景基本步骤跟第六章场景 3 基本一样，只需要在 6.4 节之前打开默认转发表

```
//开启默认转发
echo 1 > /proc/sys/net/ipv4/ip_forward

//设置转发表
iptables -t nat -A POSTROUTING -o rmnet_data0 -j MASQUERADE --random
```

8 场景 5：多路+ECM（2）

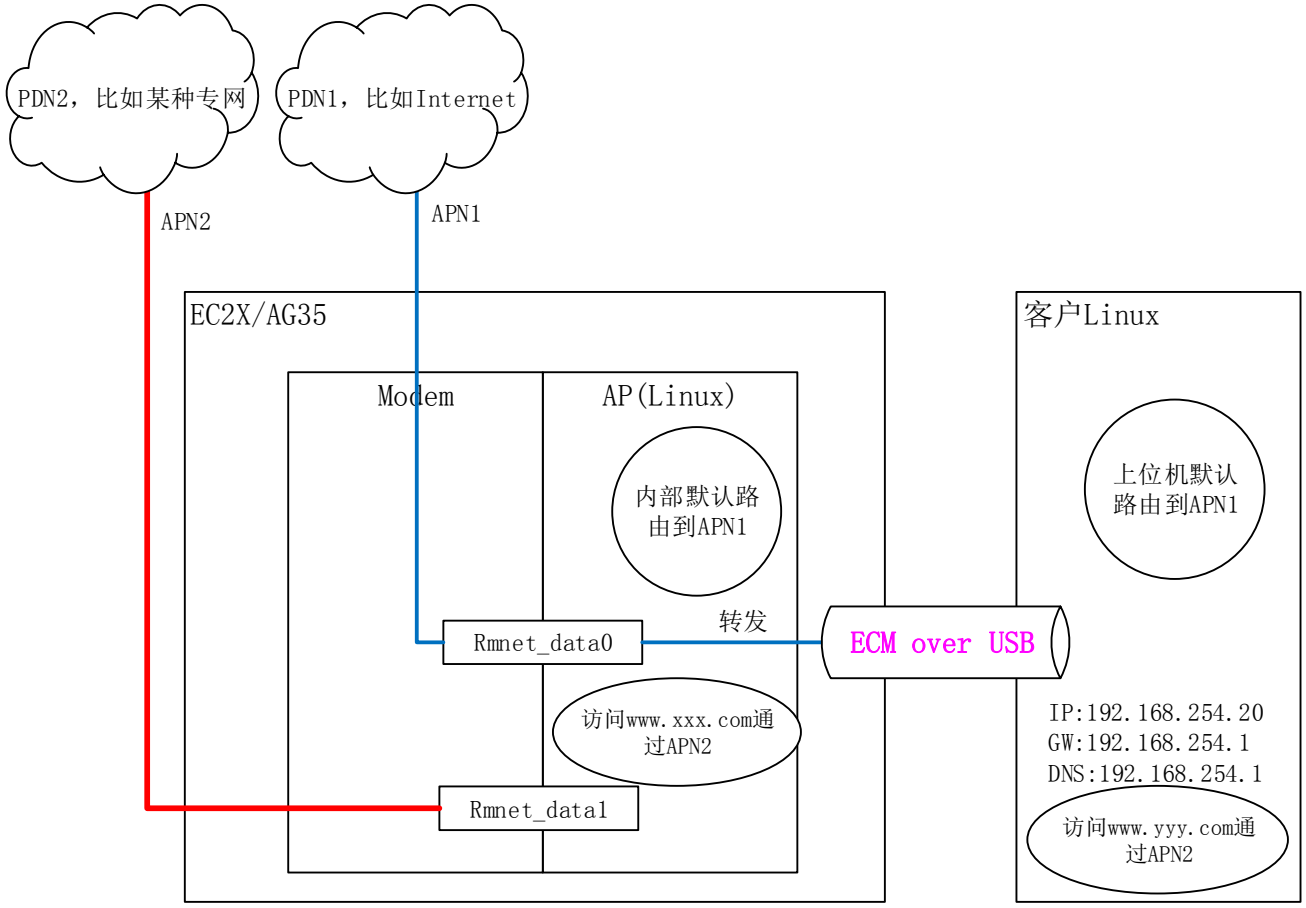


图5:场景 5

这种应用场景的基本步骤跟场景 4 一样。上位机的访问某些 sever 通过 APN2，参考设计方案：

8.1.1. 域名解析

因为上位机的域名解析全部是由模块 AP 侧的 dnsmasq 通过 APN1 的 DNS 解析，所以上位机想通过域名访问必须在 AP 侧建立一个 server（一个 socket）来完成 dns 解析和转发表配置。

8.1.2. 路由设置

（2）如果上位机直接基于 IP 地址访问，则只需要在 AP 侧配置一条转发规则，如下

```
//假设 www.yyy.com 的 ip 地址为 47.88.189.189
//转发设置
iptables -t nat -A POSTROUTING -d 47.88.189.189 -o rmnet_data1 -j MASQUERADE
或
iptables -t nat -A POSTROUTING -o rmnet_data1 -j MASQUERADE --random
```

```
//路由设置
ip route add 47.88.189.189/32 dev rmnet_data1
```

9 场景 6：多路+ECM（3）

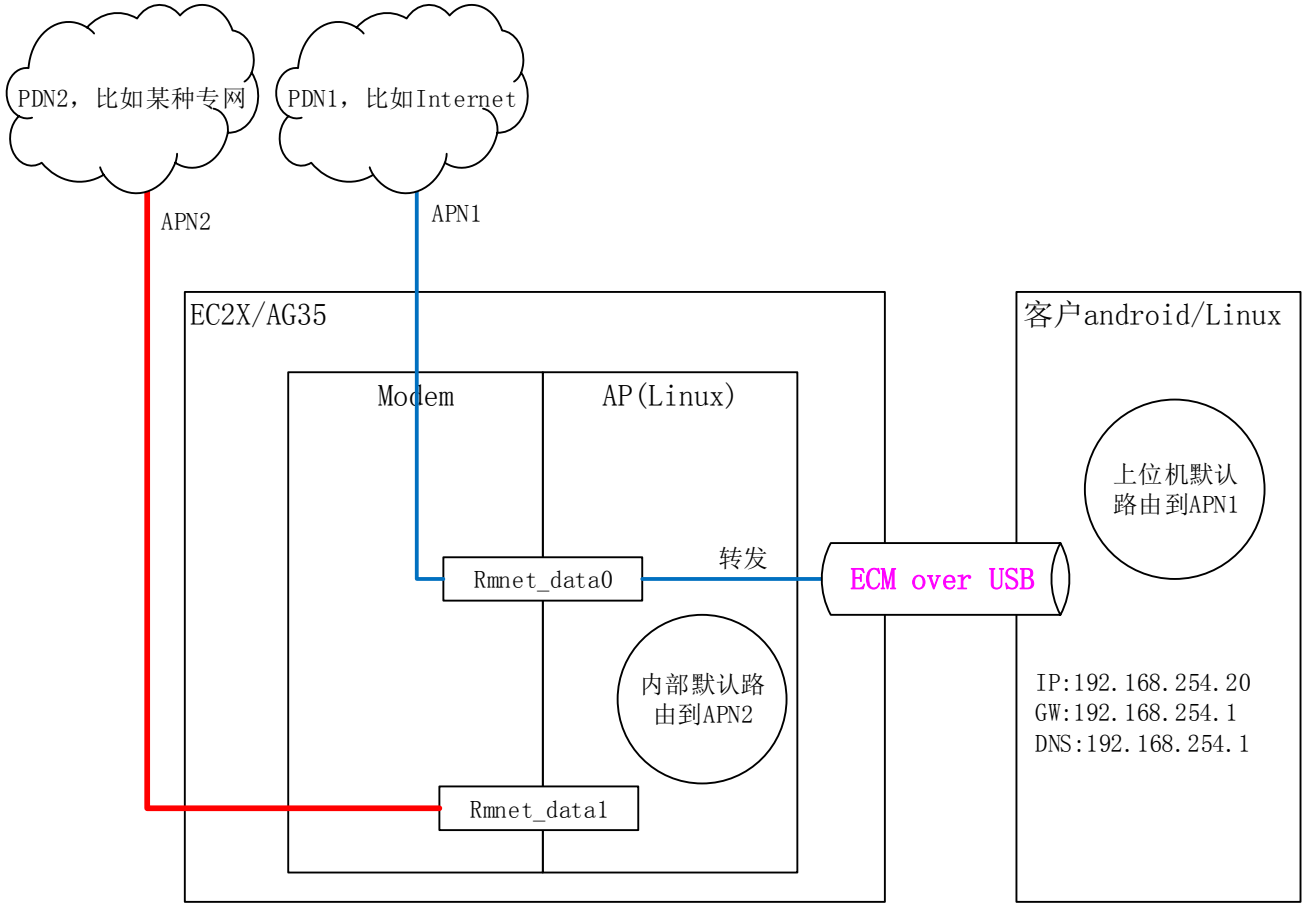


图6:场景 6

9.1. 规则清空

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

9.2. AP 侧路由设置

```
//获取 APN2 网关地址
QI_Data_Call_Info_Get(2, &info);
//设置默认路由，假设获得的地址为 10.32.80.46
route add default dev rmnet_data2
或者
route add –net 10.32.80.0/24 dev rmnet_data1
route add default gw 10.32.80.46
```

9.3. AP 侧 DNS 设置

这个步骤实现的模块内部默认 DNS 设置。

```
//获取 APN2 DNS 地址（API）
Ql_Data_Call_Info_Get(2, &info);

//设置默认 DNS 服务器(命令)，假设获得的地址为 primary: 121.158.280.8  和 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. 转发表设置

设置转发表是实现上位机访问外网的关键步骤。

```
//开启默认转发
echo 1 > /proc/sys/net/ipv4/ip_forward
//设置转发表
iptables -t nat -A POSTROUTING -o rmnet_data0 -j MASQUERADE --random
```

9.5. 策略路由设置

下面的操作还是在模块 AP 侧操作。

（1）创建路由表

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

（2）设置策略路由

```
ip rule add from 192.168.225.0/24 table 200 //192.168.225.0 为 ECM 设备的网络地址
```

（3）添加路由规则

```
ip route add dev rmnet_data0 table 200
或
ip route add via 10.112.7.176 table 200 //10.112.7.176 为网关地址
```

9.6. 上位机 DNS 服务器设置

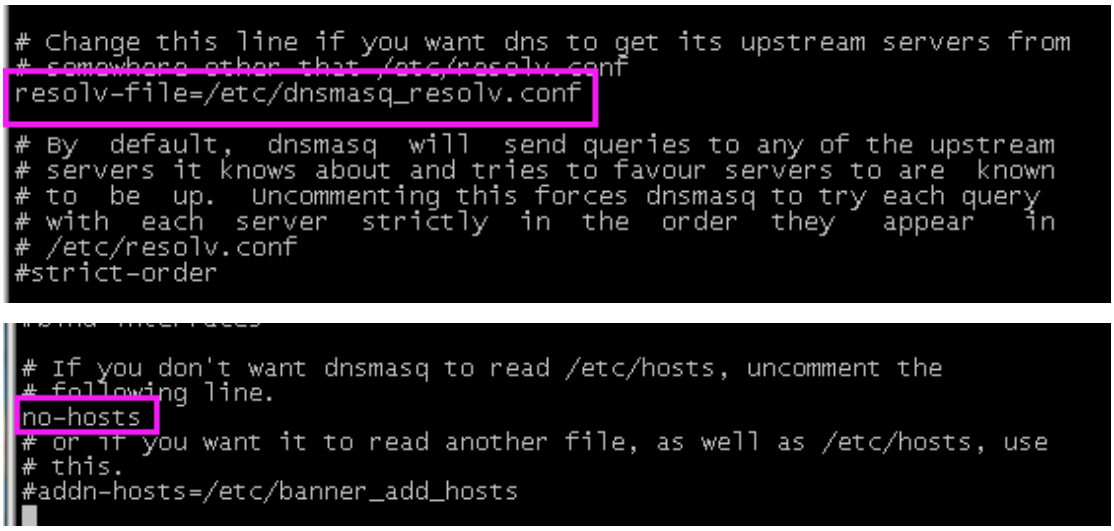
备注

（1） 如果上位机 DNS 地址是模块下发，即借助模块使用基站下发 DNS 地址解析域名，则需要继续执行如下步骤。

（2） 如果上位机设定已知的公共 DNS 服务器，比如 8.8.8.8、114.114.114.114 等，则不需要执行如下步骤。

9.6.1. dnsmasq 配置

（1）修改 /etc/dnsmasq.conf 文件



Sync，重启模块

（2）创建/etc/dnsmasq_resolv.conf 文件，添加 APN1 的 DNS 服务器地址

```
//获取 APN1 DNS 地址（API）
Ql_Data_Call_Info_Get(1, &info);
//假设获得的地址为 primary: 211.138.180.2 和 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. APN1 的 DNS 路由设置

```
//获取 APN1 DNS 地址（API）
Ql_Data_Call_Info_Get(1, &info);

//设置 DNS 路由(命令), 假设获得的地址为 primary: 211.138.180.2 和 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
```

备注

AG35 模块的 dnsmasq.conf 文件在/data 路径下面

10 场景 7：多路+ECM（4）

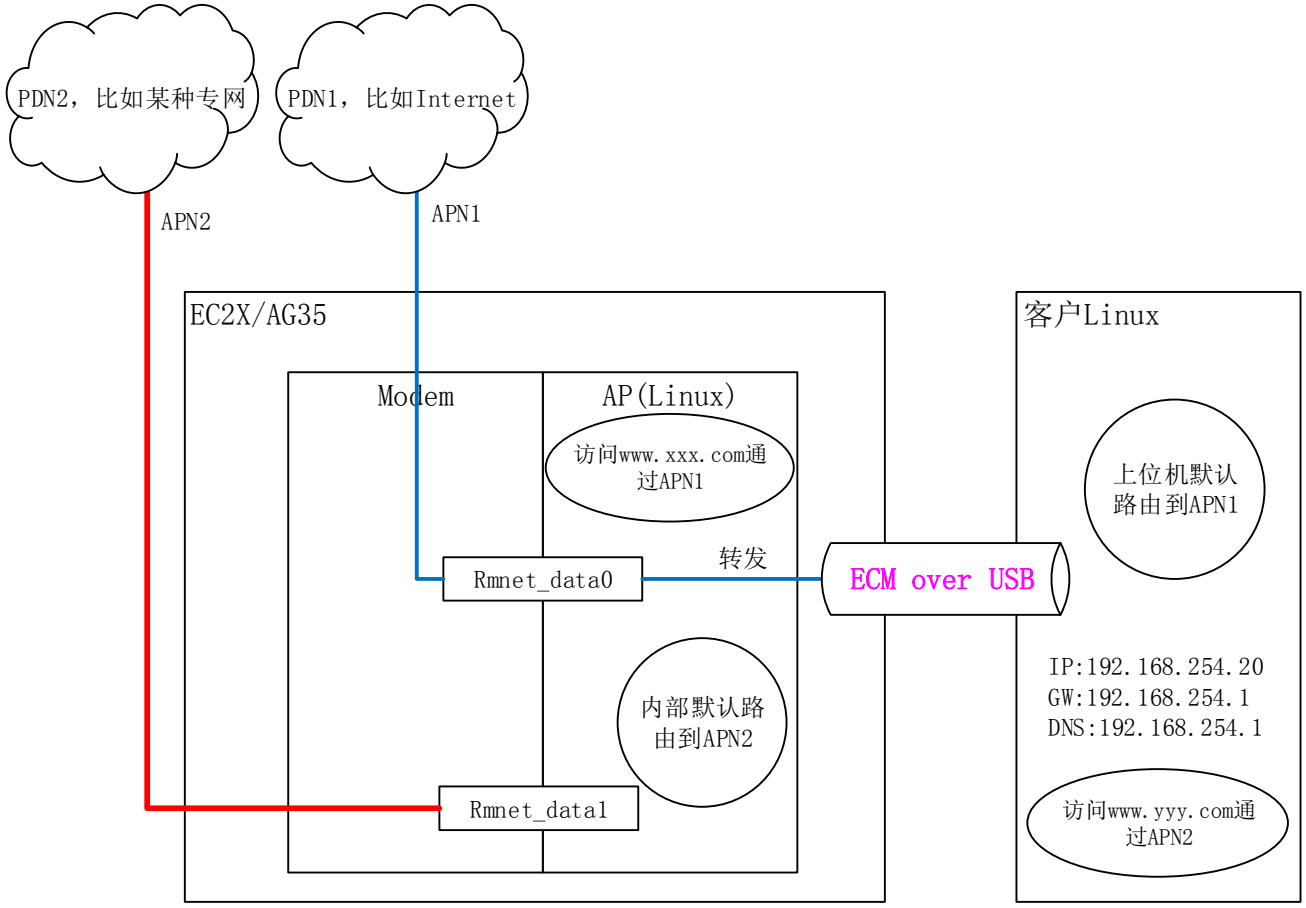


图7:场景 7

10.1. 基本设置

参照第 9 章场景 6 完成所有基本操作设置

10.2. AP 侧访问某个 sever 通过 APN1

10.2.1. 域名解析

```
//SDK API，用法如下
QL_nslookup( www.xxx.com, 211.138.180.2, IPV4, resolved_output)，其中 211.138.180.2 是 APN1 的 DNS 地址
```

10.2.2. 路由设置

```
//模块内部访问规则设置
route add –net 47.88.189.189/32 gw 10.112.7.176 dev rmnet_data0
其中 47.88.189.189 是 www.xxx.com 地址, 10.112.7.176 是 APN1 的网关(Ql_Data_Call_Info_Get 获得)
```

10.3. 上位机访问某个 sever 通过 APN2

参考设计方案：

10.3.1. 域名解析

因为上位机的域名解析全部是由模块 AP 侧的 dnsmasq 通过 APN1 的 DNS 解析，即使使用公共 DNS 服务器也是如此。所以上位机想通过域名访问必须在 AP 侧建立一个 server（一个 socket）来完成 dns 解析和转发表配置。

10.3.2. 路由设置

如果上位机直接基于 IP 地址访问，则只需要在 AP 侧配置一条转发规则，如下

```
//假设 www.yyy.com 的 ip 地址为 47.88.189.189
//转发设置
iptables -t nat -A POSTROUTING -d 47.88.189.189 -o rmnet_data1 -j MASQUERADE
或
iptables -t nat -A POSTROUTING -o rmnet_data1 -j MASQUERADE --random

//路由设置
ip route add 47.88.189.189/32 dev rmnet_data1 table 200
或基于策略路由
ip rule add to 47.88.189.189 table main
```

11 域名解析补充说明

本章对 Linux 系统的 DNS 解作出一些额外说明，以及对专网域名解析给出指导意见。

API

（1） Linux 标准函数 gethostbyname()，使用 resolve.conf 里面的 dns 地址进行 IPV4 地址解析，或者从 /etc/hosts 直接解析。

（2） Linux 标准函数 getaddrinfo()，使用 resolve.conf 里面的 dns 地址同时进行 IPV4 和 IPV6 地址解析，或者从 /etc/hosts 直接解析。

（3） Quectel 函数 QL_nslookup()，使用指定的 dns 地址，能够分别单独解析 IPV4 和 IPV6 地址。

需要指出的是：以上 API 都不能保证一次性解析成功，需要反复调用 2-3 次，这是因为 DNS 协议传输层走的是 UDP 协议，丢失报文是很正常的现象。另外 dns 服务器本身的网络连通性也是很重要。

命令

（1）ping，返回的域名解析基于 Linux 系统默认 DNS

（2）nslookup，模块里面只能使用 resolve.conf 默认的 DNS 进行解析，与 PC 上面使用该命令有差异。

12 数据业务 API

12.1. 数据类型

12.1.1. ql_data_call_error_e

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

12.1.2. 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;
```

12.1.3. 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;
```

12.1.4. 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;
```

12.1.5. 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;
```

12.1.6. 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. */
};
```

12.1.7. 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. */
};
```

12.1.8. 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;
```

12.1.9. 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;
```

12.1.10. 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. */
};
```

12.1.11. 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 */
};
```

12.1.12. 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 */
};
```

12.1.13. 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;
```

12.1.14. 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;
```

12.1.15. 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;
```

12.1.16. ql_apn_info_list_s

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

12.2. 函数

12.2.1. 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)
```

12.2.2. 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)
```

12.2.3. 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)
```

12.2.4. 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)
```

12.2.5. 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)
```

12.2.6. 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)
```

12.2.7. 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)
```

12.2.8. 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);
```

12.2.9. 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);
```

12.2.10. 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 常见问题

1. 抓取 pcap log

（1）输入如下命令
抓所有网口
tcpdump -i any -p -vv -s 0 -w ./capture1.pcap &
抓指定网口
tcpdump -i rmnet_data0 -p -vv -s 0 -w ./capture1.pcap &

（2）运行程序
（3）kill 掉 tcpdump 命令，上传 capture1.pcap 文件

2. 查看 iptables 的表

（1）查看 nat 表
iptables -nvt nat -L
（2）查看 filter 表
iptables -nvt filter -L

3. 3GPP2/CDMA 因为鉴权无法拨号上网，如何排除

（1）Modem 侧检查
参数设置：
at+qctpwdcfg="<username>","<userpasswd>"
at+qcfg=”cdmaruim”,1
拨号测试：
at+qiact=1//失败返回 error

如果失败，大多数原因都是用户名和密码不正确。
（2）AP 侧检查
如果第一步 OK，检查 profile_id 是否为 0，鉴权参数是否正确，具体参照 4.2 和 4.5 节。

14 附录 A 参考文档及术语缩写

表1: 参考文档

序号	文档名称	备注
[1]	Quectel_EC2x&AG35-QuecOpen_快速入门	软件开发入门

表2: 术语缩写

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