## 1. 介绍

本文档主要概述了openlinux数据拨号和网络参数获取接口，包含了编程接口API，数据类型等定义和说明。本接口支持多路APN拨号，自动维护重拨，支持异步事件通知，无需用户实时查询。

**注意**：此接口在一个Linux系统上只能由一个进程调用

## 2. 编程接口

### bool Ql\_wwan\_init(void)

完成库的初始化，成功返回TRUE，错误返回FALSE

**Linux RmNet网卡启动和查询接口**

### bool Ql\_wwan\_rm\_up(Ql\_wwan\_data\_call\_params\_t \*call\_params)

发起某一路拨号，成功返回TRUE,错误返回FALSE

### bool Ql\_wwan\_rm\_down(unsigned int profile\_id)

释放某一路，成功返回TRUE,错误返回FALSE

### bool Ql\_wwan\_rm\_info\_get(int profile\_id, Ql\_wwan\_rm\_info\_t\* rm\_info)

获取某一路的网卡的运行参数信息

**Modem 空口基本动作和参数获取**

### bool Ql\_wwan\_um\_attach(void)

强制modem注册网络，成功返回TRUE,返回FALSE

### bool Ql\_wwan\_um\_deattach(void)

强制modem反注册，成功返回TRUE,返回FALSE

### bool Ql\_wwan\_um\_info\_get(Ql\_wwan\_um\_info\_t \*um\_info)

获取当前空口基本网络参数

## 3. 数据类型

### Ql\_wwan\_attach\_state\_t

typedef enum {

QL\_WWAN\_UNKNOWN, //-- Unknown or not applicable

QL\_WWAN\_ATTACHED, //-- Attached

QL\_WWAN\_DETACHED //-- Detached

} Ql\_wwan\_attach\_state\_t;

### Ql\_wwan\_registration\_state\_t

typedef enum {

QL\_WWAN\_NOT\_REGISTERED, /\* Not registered; mobile is not currently searching

for a new network to provide service \*/

QL\_WWAN\_REGISTERED, /\* Registered with a network \*/

QL\_WWAN\_NOT\_REGISTERED\_SEARCHING, /\* Not registered, but mobile is currently

searching for a new network to provide service

\*/

QL\_WWAN\_REGISTRATION\_DENIED, /\* Registration denied by the visible network \*/

QL\_WWAN\_REGISTRATION\_UNKNOWN /\* Registration state is unknown \*/

} Ql\_wwan\_registration\_state\_t

### Ql\_wwan\_network\_t

typedef enum {

QL\_WWAN\_UNKONWN,

QL\_WWAN\_3GPP, //Connect the WAN on a UMTS network

QL\_WWAN\_3GPP2 //Connect the WAN on a CDMA network

} Ql\_wwan\_network\_t;

### Ql\_wwan\_bear\_type\_t

typedef enum {

QL\_WWAN\_RAT\_NULL\_BEARER,

QL\_WWAN\_RAT\_3GPP\_WCDMA,

QL\_WWAN\_RAT\_3GPP\_GERAN,

QL\_WWAN\_RAT\_3GPP\_LTE,

QL\_WWAN\_RAT\_3GPP\_TDSCDMA,

QL\_WWAN\_RAT\_3GPP\_WLAN,

QL\_WWAN\_RAT\_3GPP\_MAX,

QL\_WWAN\_RAT\_3GPP2\_1X,

QL\_WWAN\_RAT\_3GPP2\_HRPD,

QL\_WWAN\_RAT\_3GPP2\_EHRPD,

QL\_WWAN\_RAT\_3GPP2\_WLAN,

QL\_WWAN\_RAT\_3GPP2\_MAX,

QL\_WWAN\_RAT\_WLAN,

QL\_WWAN\_RAT\_WLAN\_MAX

} Ql\_wwan\_bear\_type\_t;

### Ql\_wwan\_call\_method\_t

typedef enum {

QL\_WWAN\_CALL\_IF\_DSI,

QL\_WWAN\_CALL\_IF\_QCMAP

} Ql\_wwan\_call\_method\_t;

### Ql\_wwan\_call\_state\_t

typedef enum {

QL\_WWAN\_CALL\_IDLE,

QL\_WWAN\_CALL\_CONNECTING,

QL\_WWAN\_CALL\_CONNECTED, //get ip addr

QL\_WWAN\_CALL\_RECONNECT,

QL\_WWAN\_CALL\_DISCONNECTING,

QL\_WWAN\_CALL\_DISCONNECTED,

} Ql\_wwan\_call\_state\_t;

### Ql\_wwan\_v4\_conf\_t

typedef struct {

struct in\_addr public\_ip; /\*\* Public IP address. \*/

struct in\_addr gw\_addr; /\*\* Gateway IP address. \*/

struct in\_addr pri\_dns\_addr; /\*\* Primary Domain Name Service (DNS) IP address. \*/

struct in\_addr sec\_dns\_addr; /\*\* Secondary Domain Name Service (DNS) IP address. \*/

} Ql\_wwan\_v4\_conf\_t;

### Ql\_wwan\_v6\_conf\_t

typedef struct {

struct in6\_addr public\_ip; /\*\* Public IPv6 address. \*/

struct in6\_addr gw\_addr; /\*\* Gateway IP address. \*/

struct in6\_addr pri\_dns\_addr; /\*\* Primary Domain Name Service (DNS) IPv6 address. \*/

struct in6\_addr sec\_dns\_addr; /\*\*Secondary Domain Name Service (DNS) IPv6 address. \*/

} Ql\_wwan\_v6\_conf\_t;

### Ql\_wwan\_nw\_params\_t

typedef struct {

/\*\* IPv4 configuration. \*/

int v4\_valid; //0:invalid, 1:valid

Ql\_wwan\_v4\_conf\_t v4\_conf;

/\*\* IPv6 configuration. \*/

int v6\_valid; //0:invalid, 1:valid

Ql\_wwan\_v6\_conf\_t v6\_conf;

} Ql\_wwan\_nw\_params\_t;

### Ql\_wwan\_evt\_t

typedef struct {

unsigned int profile\_id;

Ql\_wwan\_call\_state\_t state;

} Ql\_wwan\_evt\_t;

### Ql\_wwan\_call\_evt\_cbfun\_t

typedef void (\*Ql\_wwan\_call\_evt\_cbfun\_t)(Ql\_wwan\_evt\_t);

### Ql\_wwan\_data\_call\_params\_t

typedef struct {

unsigned int profile\_id;//0-4, 0 means APN1

Ql\_wwan\_ip\_type\_t ip\_type;

bool autoconnect;

Ql\_wwan\_call\_evt\_cbfun\_t call\_evt\_cb;

} Ql\_wwan\_data\_call\_params\_t;

### Ql\_wwan\_rm\_info\_t

typedef struct {

bool enable;//false:down, true:up

Ql\_wwan\_call\_state\_t sm;//state machine

Ql\_wwan\_call\_method\_t call\_if;

Ql\_wwan\_ip\_type\_t call\_ip\_type;

bool autoconnect;

/\* profile Params \*/

char apn[16];

unsigned int profile\_id;

/\* Rm iface Params \*/

char rm\_name[16]; //iface name

Ql\_wwan\_nw\_params\_t rm\_conf;

} Ql\_wwan\_rm\_info\_t;//rmnet info

### Ql\_wwan\_um\_info\_t

typedef struct {

Ql\_wwan\_registration\_state\_t reg\_state;

Ql\_wwan\_attach\_state\_t attach\_state;

Ql\_wwan\_network\_t network\_type;

Ql\_wwan\_bear\_type\_t bear\_type;

} Ql\_wwan\_um\_info\_t;//umnet info

## 4. 拨号流程

大致流程如下，具体可参照test：

进程守护

通过qwwan\_rm\_info\_get获取网络参数，设置系统网络信息，例如路由 DNS

qwwan\_init

qwwan\_rm\_up

等待回调函数收到CONNECTED事件