BG95-QuecOpen Extended QAPI Application Note

LPWA Module Series

Rev. BG95-QuecOpen_Extended_QAPI_Application_Note_V1.1

Date: 2019-09-03

Status: Preliminary

www.quectel.com

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, China 200233
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

About the Document

History

Revision	Date	Author	Description
1.0	2019-07-13	Elvis SUN/ Sherlock ZHAO/ Walker HAN/ Hyman DING	Initial
1.1	2019-09-02	Walker HAN Hyman Ding	Optimization for some QAPIs

Contents

Table Index

1 Introduction

This document mainly introduces the extended QAPIs designed and implemented by Quectel BG95-QuecOpen module. Working with module through QAPIs rather than AT commands allows customers to design their own QuecOpen applications more flexibly and efficiently.

2 Extended QAPIs

2.1. System APIs

Quectel provides some QAPIs for customers to perform system-level operations on module, including module shutdown, restart, and other configuration functions.

This chapter describes the following QAPIs:

```
qapi_QT_Reset_Device
qapi_QT_Shutdown_Device
qapi_QT_Sahara_Mode_Get
qapi_QT_Sahara_Mode_Set
qapi_QT_USB_Sio_Open*
qapi_QT_USB_Sio_Close*
qapi_QT_USB_Sio_Transmit*
qapi_QT_MP_FW_Ver_Get
qapi_QT_AP_FW_Ver_Get
qapi_QT_IMEI_Get
qapi_QT_MP_Core_Info_Get
qapi_QT_AP_Core_Info_Get
qapi_QT_Manufacturer_Info_Get
```

NOTE

$^{*)}$ means the QAPI is not support now

2.1.1. Data Structure

2.1.1.1. Enumeration Type

2.1.1.1.1. Enum qapi_QT_FATAL_ERR_MODE_e

This enumeration is used in *qapi_QT_Sahara_Mode_Set* function.

```
typedef enum {
      QT_FATAL_ERR_RESET = 0,
      QT_FATAL_ERR_SAHARA = 1,

      QT_FATAL_ERR_MAX
} qapi_QT_FATAL_ERR_MODE_e;
```

- **Parameters**

Parameter	Description
QAPI_FATAL_ERR_RESET	Set the module into reset mode.
QAPI_FATAL_ERR_SAHARA	Set the module into Sahara dump mode.

2.1.2. API Functions

2.1.2.1. qapi_QT_Reset_Device

This function is used to reset the module.

- **Prototype**

```
qapi_Status_t qapi_QT_Reset_Device(uint16_t mode)
```

- **Parameters**

mode:
[in] Must be 0.

- **Return Value**

QAPI_QT_ERR_OK on success, and others on error.

2.1.2.2. qapi_QT_Shutdown_Device

This function is used to shut down the module.

- **Prototype**

```
qapi_Status_t qapi_QT_Shutdown_Device(void);
```

- **Parameters**

None.

- **Return Value**

QAPI_QT_ERR_OK on success, and others on errors.

2.1.2.3. qapi_QT_Sahara_Mode_Get

When the module meets a fatal error, it may enter into either of the following two modes: one is the normal reset mode, and the other is Sahara dump mode in which the RAM log files can be collected to help analyze the crash issue.

This function is used to get the NV item value of Sahara mode setting.

- **Prototype**

```
qapi_Status_t qapi_QT_Sahara_Mode_Get(qapi_QT_FATAL_ERR_MODE_e* mode)
```

- **Parameters**

mode:
[out] Pointer, store the Sahara setting value which are read from the NV item.

- **Return Value**

QAPI_QT_ERR_OK on success, and others on errors.

2.1.2.4. qapi_QT_Sahara_Mode_Set

This function is used to set the NV item value of Sahara mode setting. If the module is expected to be reset automatically when crash occurs, set the mode as *QT_FATAL_ERR_RESET*. If RAM log files are expected to be collected when crash occurs, set the mode as *QT_FATAL_ERR_SAHARA*. The settings will take effect after the module is restarted.

- **Prototype**

```
qapi_Status_t qapi_QT_Sahara_Mode_Set(qapi_QT_FATAL_ERR_MODE_e mode)
```

- **Parameters**

mode:

[in] Set the module behavior when it meets a fatal error.

- **Return Value**

QAPI_QT_ERR_OK on success, and others on errors.

2.1.2.5. qapi_QT_USB_Sio_Open*

This function is used to open USB NMEA port for output customer application log which is used for debugging purpose.

- **Prototype**

```
qapi_Status_t qapi_QT_USB_Sio_Open(void);
```

- **Parameters**

None.

- **Return Value**

QAPI_QT_ERR_OK on success, and others on errors.

2.1.2.6. qapi_QT_USB_Sio_Close*

This function is used to close USB NMEA port if customer does not need it.

- **Prototype**

```
qapi_Status_t qapi_QT_USB_Sio_Close(void);
```

- **Parameters**

None.

- **Return Value**

QAPI_QT_ERR_OK on success, and others on errors.

2.1.2.7. qapi_QT_USB_Sio_Transmit*

This function is used to output customer's application debug log.

- **Prototype**

```
qapi_Status_t qapi_QT_USB_Sio_Transmit(char *log)
```

- **Parameters**

log:
[in] Pointer. Store the log data which the customer needs to output. Maximum support for output 149 characters each time.

- **Return Value**

QAPI_QT_ERR_OK on success, and others on errors.

2.1.2.8. qapi_QT_MP_FW_Ver_Get

This function is used to get module kernel modem side version number.

- **Prototype**

```
qapi_QT_Status_t qapi_QT_MP_FW_Ver_Get(char*  version, uint16* len)
```

- **Parameters**

version:
[out] Pointer. Used to store kernel modem side version number. A minimum buffer of 64 bytes is required.
len:
[out] Pointer: The length of the module version number string.

- **Return Value**

QAPI_QT_ERR_OK on success, and others on errors.

2.1.2.9. qapi_QT_AP_FW_Ver_Get

This function is used to get module kernel application side version number.

- **Prototype**

```
qapi_QT_Status_t qapi_QT_AP_FW_Ver_Get(char*  version, uint16* len)
```

- **Parameters**

version:
[out] Pointer. Used to store kernel application side version number. A minimum buffer of 64 bytes is required.
len:
[out] Pointer: The length of the application version number string.

- **Return Value**

QAPI_QT_ERR_OK on success, and others on errors.

2.1.2.10. qapi_QT_IMEI_Get

This function is used to get module IMEI number.

- **Prototype**

```
qapi_QT_Status_t qapi_QT_IMEI_Get(char* imei, uint16* len)
```

- **Parameters**

imei:
[out] Pointer. Used to store module IMEI number. A minimum buffer of 16 bytes is required.
len:
[out] Pointer: The length of the IMEI number string.

- **Return Value**

QAPI_QT_ERR_OK on success, and others on errors.

2.1.2.11. qapi_QT_MP_Core_Info_Get

This function is used to get Qualcomm MP release information.

- **Prototype**

```
qapi_QT_Status_t qapi_QT_MP_Core_Info_Get(char* info, uint16* len)
```

- **Parameters**

info:
[out] Pointer. Store the Qualcomm release information of MP. A minimum buffer of 64 bytes is required.
len:

[out] Pointer: The length of the Qualcomm release information of MP string.

- **Return Value**

QAPI_QT_ERR_OK on success, and others on errors.

2.1.2.12. qapi_QT_AP_Core_Info_Get

This function is used to get Qualcomm AP release information.

- **Prototype**

```
qapi_QT_Status_t qapi_QT_AP_Core_Info_Get(char* info, uint16* len)
```

- **Parameters**

info:
[out] Pointer. Store the Qualcomm release information of AP. A minimum buffer of 64 bytes is required.
len:
[out] Pointer: The length of the Qualcomm release information of AP string.

- **Return Value**

QAPI_QT_ERR_OK on success, and others on errors.

2.1.2.13. qapi_QT_Manufacturer_Info_Get

This function is used to get module manufacture information.

- **Prototype**

```
qapi_QT_Status_t qapi_QT_Manufacturer_Info_Get(char *info, uint16* len)
```

- **Parameters**

info:
[out] Pointer. Store the module manufacturer information. A minimum buffer of 32 bytes is required.
len:
[out] Pointer: The length of the manufacturer information string.

- **Return Value**

QAPI_QT_ERR_OK on success, and others on errors.

2.2. Network APIs

Quectel provides some QAPIs for customers to set the configuration which related to the RATs, frequency bands, RAT order, APN, PSM, eDRX functions.

This chapter describes the following QAPIs:

```
qapi_QT_Phone_Func_Set
qapi_QT_Phone_Func_Get
qapi_QT_Real_Time_Clock_Set
qapi_QT_Real_Time_Clock_Get
qapi_QT_NW_Band_Pref_Set
qapi_QT_NW_Band_Pref_Get
qapi_QT_NW_Extend_Band_Pref_Set
qapi_QT_NW_Extend_Band_Pref_Get*
qapi_QT_NW_Rat_Pref_Set
qapi_QT_NW_Rat_Pref_Get
qapi_QT_NW_Rat_Scan_Pre_Set
qapi_QT_NW_Rat_Scan_Pre_Get
qapi_QT_NW_Srv_Domain_Pref_Set
qapi_QT_NW_Srv_Domain_Pref_Get
qapi_QT_NW_PDP_Cfg_Set
qapi_QT_NW_PDP_Cfg_Get
qapi_QT_NW_GSM_Meas_Info_Get
qapi_QT_NW_LTE_Meas_Info_Get
qapi_QT_NW_PSM_Cfg_Set
qapi_QT_NW_PSM_Cfg_Get
qapi_QT_NW_eDRX_Cfg_Set
qapi_QT_NW_eDRX_Cfg_Set
```

NOTE

*) means the QAPI is not support now

2.2.1. Data Structure

2.2.1.1. Enumeration Type

2.2.1.1.1. Enum qapi_QT_NW_RAT_PREF_e

```
typedef enum {
      QT_NW_EMTC = 0,
      QT_NW_NB_IOT = 1,

      QT_NW_RAT_Max,
}qapi_QT_NW_RAT_e;
```

- **Parameters**

Parameter	Description
QT_NW_EMTC	The preferential RAT is eMTC.
QT_NW_NB_IOT	The preferential RAT is NB-IoT.
QT_NW_PREF_RAT_MAX	Invalid preferential RAT.

2.2.1.1.2. Enum qapi_QT_NW_RAT_PREF_e

```
typedef enum {
      QT_NW_PREF_GSM = 0,
      QT_NW_PREF_CATM = 1,
      QT_NW_PREF_GSM_CATM = 2,
      QT_NW_PREF_CATNB = 3,
      QT_NW_PREF_GSM_CATNB = 4,
      QT_NW_PREF_CATM_CATNB = 5,
      QT_NW_PREF_GSM_CATM_CATNB = 6,

      QT_NW_PREF_RAT_MAX
} qapi_QT_NW_RAT_PREF_e;
```

- **Parameters**

Parameter	Description
QT_NW_PREF_GSM	The preferential mode is GSM.

QT_NW_PREF_CATM	The preferential mode is Cat M1.
QT_NW_PREF_GSM_CATM	The preferential mode is GSM and Cat M1.
QT_NW_PREF_CATNB	The preferential mode is Cat NB2.
QT_NW_PREF_GSM_CATNB	The preferential mode is GSM and Cat NB2.
QT_NW_PREF_CATM_CATNB	The preferential mode is Cat M1 and Cat NB2.
QT_NW_PREF_GSM_CATM_CATNB	The preferential mode is GSM, Cat M1 and Cat NB2.
QT_NW_PREF_RAT_MAX	Invalid preferential mode.

2.2.1.1.3. Enum qapi_QT_NW_RAT_SCAN_ORDER_e

```
typedef enum {
      QT_NW_PREF_SCAN_CATM_CATNB_GSM  = 0,
      QT _NW_PREF_SCAN_CATM_GSM_CATNB  = 1,
      QT _NW_PREF_SCAN_CATNB_CATM_GSM  = 2,
      QT _NW_PREF_SCAN_CATNB_GSM_CATM  = 3,
      QT _NW_PREF_SCAN_GSM_CATM_CATNB  = 4,
      QT _NW_PREF_SCAN_GSM_CATNB_CATM  = 5,

      QT_NW_PREF_RAT_SCAN_ORDER_MAX
} qapi_QT_NW_RAT_SCAN_ORDER_e;
```

- **Parameters**

Parameter	Description
QT_NW_PREF_SCAN_CATM_CATNB_GSM	The priority of scanning RAT is Cat M1, Cat NB2, GSM.
QT_NW_PREF_SCAN_CATM_GSM_CATNB	The priority of scanning RAT is Cat M1, GSM, Cat NB2.
QT _NW_PREF_SCAN_CATNB_CATM_GSM	The priority of scanning RAT is Cat NB2, Cat M1, GSM.
QT _NW_PREF_SCAN_CATNB_GSM_CATM	The priority of scanning RAT is Cat NB2, GSM, Cat M1.
QT _NW_PREF_SCAN_GSM_CATM_CATNB	The priority of scanning RAT is GSM, Cat M1, Cat NB2.
QT _NW_PREF_SCAN_GSM_CATNB_CATM	The priority of scanning RAT is GSM, Cat NB2, Cat M1.
QT_NW_PREF_RAT_SCAN_ORDER_MAX	Invalid priority of scanning RAT.

2.2.1.1.4. Enum qapi_QT_NW_SRV_DOMAIN_PREF_e

```
typedef enum {
        QT_NW_PREF_CS_ONLY = 0,
        QT_NW_PREF_PS_ONLY = 1,
        QT_NW_PREF_CS_PS = 2,

        QT_NW_PREF_SRV_DOMAIN_MAX
} qapi_QT_NW_SRV_DOMAIN_PREF_e;
```

- **Parameters**

Parameter	Description
QT_NW_PREF_CS_ONLY	The preferential service domain is only CS.
QT_NW_PREF_PS_ONLY	The preferential service domain is only PS.
QT_NW_PREF_CS_PS	The preferential service domain is CS and PS.
QT_NW_PREF_SRV_DOMAIN_MAX	Invalid preferential service domain.

2.2.1.1.5. Enum qapi_QT_GSM_BAND_e

```
typedef enum {
        QT_NW_GSM_BAND_EGSM = 0,
        QT_NW_GSM_BAND_PGSM = 1,
        QT_NW_GSM_BAND_PCS_1900  = 2,
        QT_NW_GSM_BAND_DCS_1800  = 3,
        QT_NW_GSM_BAND_CELL_850  = 4,

        QT_NW_GSM_BAND_MAX
} qapi_QT_GSM_BAND_e;
```

- **Parameters**

Parameter	Description
QT_NW_GSM_BAND_EGSM	The band is EGSM 900.
QT_NW_GSM_BAND_PGSM	The band is PGSM 900.
QT_NW_GSM_BAND_PCS_1900	The band is PCS 1900.

QT_NW_GSM_BAND_DCS_1800	The band is DCS 1800.
QT_NW_GSM_BAND_CELL_850	The band is 850.
QT_NW_GSM_BAND_MAX	Invalid band.

2.2.1.1.6. Enum qapi_QT_NW_MODE_SEL_e

```
typedef enum {
      QT_NW_GSM_MODE = 0,
      QT_NW_CATM_MODE = 1,
      QT_NW_CATNB_MODE = 2,

      QT_NW_MAX_MODE
} qapi_QT_NW_MODE_SEL_e;
```

- **Parameters**

Parameter	Description
QT_NW_GSM_MODE	The selected mode is GSM.
QT_NW_CATM_MODE	The selected mode is Cat M1.
QT_NW_CATNB_MODE	The selected mode is Cat NB2.
QT_NW_MAX_MODE	Invalid selected mode.

2.2.1.1.7. Enum qapi_QT_NW_DS_PROFILE_PDP_TYPE_e

```
typedef enum
{
      QT_NW_DS_PROFILE_PDP_IPV4 = 0,
      QT_NW_DS_PROFILE_PDP_IPV6 = 1,
      QT_NW_DS_PROFILE_PDP_IPV4V6 = 2,

      QT_NW_DS_PROFILE_PDP_MAX
} qapi_QT_NW_DS_PROFILE_PDP_TYPE_e;
```

- **Parameters**

Parameter	Description

QT_NW_DS_PROFILE_PDP_IPV4	The PDP type is IPv4.
QT_NW_DS_PROFILE_PDP_IPV6	The PDP type is IPv6.
QT_NW_DS_PROFILE_PDP_IPV4V6	The PDP type is IPv4v6.
QT_NW_DS_PROFILE_AUTH_TYPE_MAX	Invalid PDP type.

2.2.1.1.8. Enum qapi_QT_NW_DS_PROFILE_AUTH_TYPE_e

```
typedef enum {
    QT_NW_DS_PROFILE_AUTH_PAP = 0,
    QT_NW_DS_PROFILE_AUTH_CHAP = 1,
    QT_NW_DS_PROFILE_AUTH_PAP_CHAP = 2,

    QT_NW_DS_PROFILE_AUTH_TYPE_MAX
} qapi_QT_NW_DS_PROFILE_AUTH_TYPE_e;
```

- **Parameters**

Parameter	Description
QT_NW_DS_PROFILE_AUTH_PAP	Password Authentication Protocol
QT_NW_DS_PROFILE_AUTH_CHAP	Challenge Handshake Authentication Protocol
QT_NW_DS_PROFILE_AUTH_PAP_CHAP	PAP and CHAP
QT_NW_DS_PROFILE_AUTH_TYPE_MAX	Invalid authentication

2.2.1.1.9. Enum qapi_QT_NW_CFUN_MODE_e

```
typedef enum {
    QT_NW_CFUN_MIN_FUNC = 0,
    QT_NW_CFUN_FUNN_FUNC = 1,
    QT_NW_CFUN_SHUT_DOWN = 2,
    QT_NW_CFUN_RESET = 3,
    QT_NW_CFUN_FTM = 4,

    QT_NW_CFUN_MAX
} qapi_QT_NW_CFUN_MODE_e;
```

- **Parameters**

Parameter	Description
QT_NW_CFUN_MIN_FUNC	Minimum functionality
QT_NW_CFUN_FUNN_FUNC	Full functionality
QT_NW_CFUN_SHUT_DOWN	Shut down
QT_NW_CFUN_RESET	Reset
QT_NW_CFUN_FTM	Factory Test Mode
QT_NW_CFUN_MAX	Invalid parameter.

2.2.1.2. Definition Type

```
#define QT_DS_PROFILE_MAX_APN_STRING_LEN        (101)
#define QT_DS_PROFILE_MAX_USERNAME_LEN          (128)
#define QT_DS_PROFILE_MAX_PASSWORD_LEN          (128)
```

2.2.1.3. Structure Type

2.2.1.3.1. Struct qapi_QT_NW_Band_Params_t

```
typedef struct {
      uint8_t gsm_band;
      uint64_t catm_band_low;
      uint64_t nb_band_low;
}qapi_QT_NW_Band_Params_t;
```

- **Parameters**

Type	Parameter	Description
Uint8_t	*gsm_band*	Preferred GSM band
Uint64_t	*catm_band_low*	Preferred eMTC band from B1 to B64
Uint64_t	*Nb_band_low*	Preferred NB-IoT band from B1 to B64

2.2.1.3.2. Struct qapi_QT_NW_DS_Profile_PDP_Context_t

```
typedef struct {
        qapi_QT_NW_DS_PROFILE_PDP_TYPE_e  pdp_type;
        uint8_t apn[QT_DS_PROFILE_MAX_APN_STRING_LEN+1];
        uint8_t user_name[QT_DS_PROFILE_MAX_USERNAME_LEN+1];
        uint8_t pass_word[QT_DS_PROFILE_MAX_PASSWORD_LEN+1];
        qapi_QT_NW_DS_PROFILE_AUTH_TYPE_e  auth_type;
} qapi_QT_NW_DS_Profile_PDP_Context_t;
```

- **Parameters**

Type	Parameter	Description
qapi_QT_NW_DS_PROFILE_PDP_TYPE_e	*pdp_type*	The PDP protocol type.
uint8_t	*apn*	The access point name.
uint8_t	*user_name*	The name of user.
uint8_t	*pass_word*	The password.
qapi_QT_NW_DS_PROFILE_AUTH_TYPE_e	*auth_type*	The authentication methods.

2.2.1.3.3. Struct qapi_QT_NW_GSM_Meas_Info_t

```
typedef struct {
        uint16_t arfcn;
        uint16_t mcc;
        uint16_t mnc;
        uint16_t lac;
        uint32_t cell_id;
        qapi_QT_GSM_BAND_e  band;
        uint8_t bsic;
        uint8_t rxlev;
        uint16_t drx;
        int32_t c1;
        int32_t c2;
} qapi_QT_NW_GSM_Meas_Info_t;
```

- **Parameters**

Type	Parameter	Description
uint32_t	*arfcn*	Absolute Radio Frequency Channel Number
uint16_t	*mcc*	Mobile Country Code
uint16_t	*mnc*	Mobile Network Code
uint16_t	*lac*	Tracking Area Code
uint32_t	*cell_id*	Cell Identification
qapi_QT_GSM_BAND_e	*band*	Frequency Band
uint8_t	*bsic*	Base Station Identification Code
uint8_t	*rxlev*	RX level value for base station selection
uint16_t	*drx*	Discontinuous reception cycle length
Uint32_t	*c1*	Cell selection criterion
Uint32_t	*c2*	Cell reselection criterion

2.2.1.3.4. Struct qapi_QT_NW_LTE_Meas_Info_t

```
typedef struct {
        uint32_t earfcn;
        uint16_t mcc;
        uint16_t mnc;
        uint16_t tac;
        uint32_t cell_id;
        uint8_t freq_band;
        uint16_t pci;
        uint16_t rsrp;
        uint16_t rsrq;
        uint16_t rssi;
        uint16_t sinr;
} qapi_QT_NW_LTE_Meas_Info_t;
```

- **Parameters**

Type	Parameter	Description
uint32_t	*earfcn*	E-UTRA Absolute Radio Frequency Channel Number
uint16_t	*mcc*	Mobile Country Code
uint16_t	*mnc*	Mobile Network Code
uint16_t	*tac*	Tracking Area Code
uint32_t	*cell_id*	Cell Identification
uint8_t	*freq_band*	Frequency Band
uint16_t	*pci*	Physical Cell Identification
uint16_t	*rsrp*	Reference Signal Receiving Power
uint16_t	*rsrq*	Reference Signal Receiving Quality
uint16_t	*rssi*	Received Signal Strength Indicator
uint16_t	*sinr*	Signal to Interference plus Noise Ratio

2.2.1.3.5. Struct qapi_QT_NW_Req_PSM_Cfg_t

```
typedef struct {
      bool req_psm_enable;
      uint32_t req_active_timer_value;
      uint32_t req_periodic_tau_timer_value;
} qapi_QT_NW_Req_PSM_Cfg_t;
```

- **Parameters**

Type	Parameter	Description
bool	*req_psm_enabled*	Request to disable or enable the use of PSM.
uint32_t	*req_active_timer_value*	Requested active time value 1).
uint32_t	*req_periodic_tau_timer_value*	Requested extended periodic TAU value 2).

NOTES

1. [1] active_timer (in seconds). Valid values are below:
   0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 120, 180, 240, 300, 360, 420, 480, 540, 600, 660, 720, 780, 840, 900, 960, 1020, 1080, 1140, 1200, 1260, 1320, 1380, 1440, 1500, 1560, 1620, 1680, 1740, 1800, 1860, 2160, 2520, 2880, 3240, 3600, 3960, 4320, 4680, 5040, 5400, 5760, 6120, 6480, 6840, 7200, 7560, 7920, 8280, 8640, 9000, 9360, 9720, 10080, 10440, 10800, 11160.

2. [2] periodic_update_timer (in seconds). Valid values are below:
   0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 90, 120, 150, 180, 210, 240, 270, 300, 330, 360, 390, 420, 450, 480, 510, 540, 570, 600, 630, 660, 690, 720, 750, 780, 810, 840, 870, 900, 930, 960, 1020, 1080, 1140, 1200, 1260, 1320, 1380, 1440, 1500, 1560, 1620, 1680, 1740, 1800, 1860, 2400, 3000, 3600, 4200, 4800, 5400, 6000, 6600, 7200, 7800, 8400, 9000, 9600, 10200, 10800, 11400, 12000, 12600, 13200, 13800, 14400, 15000, 15600, 16200, 16800, 17400, 18000, 18600, 21600, 25200, 28800, 32400, 36000, 39600, 43200, 46800, 50400, 54000, 57600, 61200, 64800, 68400, 72000, 75600, 79200, 82800, 86400, 90000, 93600, 97200, 100800, 104400, 108000, 111600, 144000, 180000, 216000, 252000, 288000, 324000, 360000, 396000, 432000, 468000, 504000, 540000, 576000, 612000, 648000, 684000, 720000, 756000, 792000, 828000, 864000, 900000, 936000, 972000, 1008000, 1044000, 1080000, 1116000, 1152000, 2304000, 3456000, 4608000, 5760000, 6912000, 8064000, 9216000, 10368000, 11520000, 12672000, 13824000, 14976000, 16128000, 17280000, 18432000, 19584000, 20736000, 21888000, 23040000, 24192000, 25344000, 26496000, 27648000, 28800000, 29952000, 31104000, 32256000, 33408000, 34560000, 35712000

2.2.1.3.6. Struct qapi_QT_NW_Alloc_PSM_Cfg_t

```
typedef struct {
      bool alloc_psm_enabled;
      uint32_t alloc_active_timer_value;
      uint32_t alloc_periodic_tau_timer_value;
} qapi_QT_NW_Alloc_PSM_Cfg_t;
```

● **Parameters**

Type	Parameter	Description
bool	*alloc_psm_enable*	Allocate to disable or enable the use of PSM.
uint32_t	*alloc_active_timer_value*	Allocated active time value [1].
uint32_t	*alloc_periodic_tau_timer_value*	Allocated extended periodic TAU value [2].

NOTES

1. [1] active_timer (in seconds). Valid values are below:
   0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 120, 180, 240, 300, 360, 420, 480, 540, 600, 660, 720, 780, 840, 900, 960, 1020, 1080, 1140, 1200, 1260, 1320, 1380, 1440, 1500, 1560, 1620, 1680, 1740, 1800, 1860, 2160, 2520, 2880, 3240, 3600, 3960, 4320, 4680, 5040, 5400, 5760, 6120, 6480, 6840, 7200, 7560, 7920, 8280, 8640, 9000, 9360, 9720, 10080, 10440, 10800, 11160.

2. [2] periodic_update_timer (in seconds). Valid values are below:
   0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 90, 120, 150, 180, 210, 240, 270, 300, 330, 360, 390, 420, 450, 480, 510, 540, 570, 600, 630, 660, 690, 720, 750, 780, 810, 840, 870, 900, 930, 960, 1020, 1080, 1140, 1200, 1260, 1320, 1380, 1440, 1500, 1560, 1620, 1680, 1740, 1800, 1860, 2400, 3000, 3600, 4200, 4800, 5400, 6000, 6600, 7200, 7800, 8400, 9000, 9600, 10200, 10800, 11400, 12000, 12600, 13200, 13800, 14400, 15000, 15600, 16200, 16800, 17400, 18000, 18600, 21600, 25200, 28800, 32400, 36000, 39600, 43200, 46800, 50400, 54000, 57600, 61200, 64800, 68400, 72000, 75600, 79200, 82800, 86400, 90000, 93600, 97200, 100800, 104400, 108000, 111600, 144000, 180000, 216000, 252000, 288000, 324000, 360000, 396000, 432000, 468000, 504000, 540000, 576000, 612000, 648000, 684000, 720000, 756000, 792000, 828000, 864000, 900000, 936000, 972000, 1008000, 1044000, 1080000, 1116000, 1152000, 2304000, 3456000, 4608000, 5760000, 6912000, 8064000, 9216000, 10368000, 11520000, 12672000, 13824000, 14976000, 16128000, 17280000, 18432000, 19584000, 20736000, 21888000, 23040000, 24192000, 25344000, 26496000, 27648000, 28800000, 29952000, 31104000, 32256000, 33408000, 34560000, 35712000

2.2.1.3.7. Struct qapi_QT_NW_Req_eDRX_Cfg_t

```
typedef struct {
    bool req_edrx_enable;
    qapi_QT_NW_RAT_e  rat_mode;
    uint8_t req_ptw_cycle;
    uint8_t req_edrx_cycle;
} qapi_QT_NW_Req_eDRX_Cfg_t;
```

- **Parameters**

Type	Parameter	Description
bool	*req_edrx_enable*	Request to disable or enable the use of eDRX
qapi_QT_NW_RAT_e	*rat_mode*	Selected Radio Access Technology
uint8_t	*req_ptw_cycle*	Requested PTW cycle length for eDRX (0~15).
uint8_t	*req_edrx_cycle*	Requested eDRX cycle length for eDRX (0~15).

2.2.1.3.8. Struct qapi_QT_NW_Alloc_eDRX_Cfg_t

```
typedef struct {
        bool alloc_edrx_enable;
        uint8_t alloc_ptw_cycle;
        uint8_t alloc_edrx_cycle;
} qapi_QT_NW_Alloc_eDRX_Cfg_t;
```

- **Parameters**

Type	Parameter	Description
bool	alloc_edrx_enable	Allocate to disable or enable the use of eDRX
uint8_t	alloc_ptw_cycle	Allocated PTW cycle lenth for eDRX (0~15).
uint8_t	alloc_edrx_cycle	Allocated eDRX cycle length for eDRX (0~15).

2.2.1.3.9. Struct qapi_QT_Real_Time_Cfg_Params_t

```
typedef struct {
        uint16_t year;
        uint8_t month;
        uint8_t day;
        uint8_t hour;
        uint8_t minute;
        uint8_t second;
        uint8_t time_zone;
} qapi_QT_Real_Time_Cfg_Params_t;
```

- **Parameters**

Type	Parameter	Description
Uint16_t	*year*	Year.
uint8_t	*month*	Month.
uint8_t	*day*	Day.
uint8_t	*hour*	Hour.
uint8_t	*minute*	Minute.
uint8_t	*second*	Second

uint8_t	*time_zone*	Time zone.

2.2.2. API Functions

2.2.2.1. qapi_QT_Phone_Func_Set

This function is used to set phone functionality.

- **Prototype**

```
qapi_QT_Status_t qapi_QT_Phone_Func_Set(qapi_QT_NW_CFUN_MODE_e  *fun);
```

- **Parameters**

fun:
[in] Pointer, used to set the module functionality. Please refer to the enumeration *qapi_QT_NW_CFUN_MODE_e*.

- **Return Value**

QAPI_QT_ERR_OK on success, and others on errors.

2.2.2.2. qapi_QT_Phone_Func_Get

This function is used to get phone functionality.

- **Prototype**

```
qapi_QT_Status_t qapi_QT_Phone_Func_Get(uint8_t* fun);
```

- **Parameters**

fun:
[in] Pointer, used to store the current functionality configuration.

- **Return Value**

QAPI_QT_ERR_OK on success, and others on errors.

2.2.2.3. qapi_QT_Real_Time_Clock_Set

This function is used to set module real time.

- **Prototype**

```
qapi_QT_Status_t qapi_QT_Real_Time_Clock_Set(qapi_QT_Real_Time_Cfg_Params_t* time);
```

- **Parameters**

time:
[in] Pointer, used to store the time setting values.

- **Return Value**

QAPI_QT_ERR_OK on success, and others on errors.

2.2.2.4. qapi_QT_Real_Time_Clock_Get*

This function is used to get module real time.

- **Prototype**

```
qapi_QT_Status_t qapi_QT_Real_Time_Clock_Get(qapi_QT_Real_Time_Cfg_Params_t* time);
```

- **Parameters**

time:
[in] Pointer, used to store the time values get form modem. Same with AT+CCLK result.

- **Return Value**

QAPI_QT_ERR_OK on success, and others on errors.

2.2.2.5. qapi_QT_NW_Band_Pref_Get

This function is used to get preferred band.

- **Prototype**

```
qapi_QT_Status_t qapi_QT_NW_Band_Pref_Get(qapi_QT_NW_Band_Params_t *band_pref);
```

- **Parameters**

band_pref:
[out] Pointer, used to store the preferred band.

- **Return Value**

QAPI_QT_ERR_OK on success, and others on errors.

2.2.2.6. qapi_QT_NW_Band_Pref_Set

This function is used to set preferred band.

- **Prototype**

```
qapi_QT_Status_t qapi_QT_NW_Band_Pref_Set(qapi_QT_NW_Band_Params_t  *band_pref);
```

- **Parameters**

band_pref:
[in] Pointer, used to set the preferred band.

- **Return Value**

QAPI_QT_ERR_OK on success, and others on errors.

2.2.2.7. qapi_QT_NW_Rat_Pref_Get

This function is used to get preferred RAT (Radio Access Technology).

- **Prototype**

```
qapi_QT_Status_t qapi_QT_NW_Rat_Pref_Get(qapi_QT_NW_RAT_PREF_e  *type);
```

- **Parameters**

mode:
[out] Pointer, used to store the current configuration of preferred RAT.

- **Return Value**

QAPI_QT_ERR_OK on success, and others on errors.

2.2.2.8. qapi_QT_NW_Rat_Pref_Set

This function is used to set preferred RAT (Radio Access Technology).

- **Prototype**

```
qapi_Status_t qapi_QT_NW_Rat_Pref_Set(qapi_QT_NW_RAT_PREF_e   *mode)
```

- **Parameters**

mode:
[in] Pointer, used to set the configuration of preferred RAT.

- **Return Value**

QAPI_QT_ERR_OK on success, and others on errors.

2.2.2.9. qapi_QT_NW_Rat_Scan_Pre_Get

This function is used to get configuration of preference of RAT scan.

- **Prototype**

```
qapi_Status_t qapi_QT_NW_Rat_Scan_Pre_Get(qapi_QT_NW_RAT_SCAN_ORDER_e*   mode)
```

- **Parameters**

mode:
[out] Pointer, used to store the current configuration of preference of RAT scan.

- **Return Value**

QAPI_QT_ERR_OK on success, and others on errors.

2.2.2.10. qapi_QT_NW_Rat_Scan_Pre_Set

This function is used to set configuration of preference of RAT scan.

- **Prototype**

```
qapi_Status_t qapi_QT_NW_Rat_Scan_Pre_Get(qapi_QT_NW_RAT_SCAN_ORDER_e*   mode)
```

- **Parameters**

mode:
[in] Pointer, used to set the configuration of preference of RAT scan.

- **Return Value**

QAPI_QT_ERR_OK on success, and others on errors.

2.2.2.11. qapi_QT_NW_Srv_Domain_Pref_Get

This function is used to get configuration of preferred service domain.

- **Prototype**

```
qapi_Status_t qapi_QT_NW_Srv_Domain_Pref_Get(qapi_QT_NW_Srv_Domain_Pref_e* mode)
```

- **Parameters**

mode:
[out] Pointer, used to store the current configuration of preferred service domain.

- **Return Value**

QAPI_QT_ERR_OK on success, and others on errors.

2.2.2.12. qapi_QT_NW_Srv_Domain_Pref_Set

This function is used to set configuration of preferred service domain.

- **Prototype**

```
qapi_Status_t qapi_QT_NW_Srv_Domain_Pref_Set(qapi_QT_NW_Srv_Domain_Pref_e* mode)
```

- **Parameters**

mode:
[in] Pointer, used to set the configuration of preferred service domain.

- **Return Value**

QAPI_QT_ERR_OK on success, and others on errors.

2.2.2.13. qapi_QT_NW_PDP_Cfg_Get

This function is used to get configuration of specific pdp context number.

- **Prototype**

```
qapi_Status_t qapi_QT_NW_PDP_Cfg_Get(uint8_t *pdp_context_number,
qapi_QT_NW_DS_Profile_PDP_Context_t* profile)
```

- **Parameters**

pdp_context_number:
[in] Pointer, used to indicates specific PDP context number which need to set PDP context.

profile:
[out] Pointer, used to store the configuration of specific PDP context number.

- **Return Value**

QAPI_QT_ERR_OK on success, and others on errors.

2.2.2.14. qapi_QT_NW_PDP_Cfg_Set

This function is used to set configuration of specific PDP context number.

- **Prototype**

```
qapi_Status_t qapi_QT_NW_PDP_Cfg_Get(uint8_t *pdp_context_number,
qapi_QT_NW_DS_Profile_PDP_Context_t* profile)
```

- **Parameters**

pdp_context_number:
[in] Pointer, used to indicate specific PDP context number which need to get PDP context.

profile:
[in] Pointer, used to set the configuration of specific PDP context number.

- **Return Value**

QAPI_QT_ERR_OK on success, and others on errors.

2.2.2.15. qapi_QT_NW_GSM_Meas_Info_Get

This function is used to get information of measurement under GSM.

- **Prototype**

```
qapi_Status_t qapi_QT_NW_GSM_Meas_Info_Get(qapi_QT_NW_GSM_Meas_Info_t* meas_info)
```

- **Parameters**

meas_info:
[out] Pointer, used to store information of measurement under GSM.

- **Return Value**

QAPI_QT_ERR_OK on success, and others on errors.

2.2.2.16. qapi_QT_NW_LTE_Meas_Info_Get

This function is used to get information of measurement under LTE.

- **Prototype**

```
qapi_Status_t qapi_QT_NW_LTE_Meas_Info_Get(qapi_QT_NW_LTE_Meas_Info_t* meas_info)
```

- **Parameters**

meas_info:
[out] Pointer, used to store information of measurement under LTE.

- **Return Value**

QAPI_QT_ERR_OK on success, and others on errors.

2.2.2.17. qapi_QT_NW_PSM_Cfg_Set

This function is used to set configuration of PSM of UE.

- **Prototype**

```
qapi_Status_t qapi_QT_NW_PSM_Cfg_Set(qapi_QT_NW_Req_PSM_CFG_t* psm_cfg)
```

- **Parameters**

psm_cfg:
[in] Pointer, used to set the configuration of parameter of PSM.

- **Return Value**

QAPI_QT_ERR_OK on success, and others on errors.

2.2.2.18. qapi_QT_NW_PSM_Cfg_Get

This function is used to get parameters of PSM form network.

- **Prototype**

```
qapi_Status_t qapi_QT_NW_PSM_Cfg_Set(qapi_QT_NW_Alloc_PSM_Cfg_t* psm_cfg)
```

- **Parameters**

psm_cfg:
[out] Pointer, used to store that network allocates parameters of PSM

- **Return Value**

QAPI_QT_ERR_OK on success, and others on errors.

2.2.2.19. qapi_QT_NW_eDRX_Cfg_Set

This function is used to set configuration of eDRX of UE.

- **Prototype**

```
qapi_Status_t qapi_QT_NW_eDRX_Cfg_Set (qapi_QT_NW_Req_eDRX_Cfg_t* edrx_cfg)
```

- **Parameters**

edrx_cfg:
[in] Pointer, used to set the configuration of eDRX of UE.

- **Return Value**

QAPI_QT_ERR_OK on success, and others on errors.

2.2.2.20. qapi_QT_NW_eDRX_Cfg_Get

This function is used to get parameters of eDRX form network.

- **Prototype**

```
qapi_Status_t qapi_QT_NW_eDRX_Cfg_Set (qapi_QT_NW_RAT_e  *rat_mode,
qapi_QT_NW_Alloc_eDRX_Cfg_t* edrx_cfg)
```

- **Parameters**

rat_mode:
[in] Pointer, RAT that needs to get the parameter of eDRX.

edrx_cfg:
[out] Pointer, used to store that network allocates parameters of eDRX

- **Return Value**

QAPI_QT_ERR_OK on success, and others on errors.

2.3. (U)SIM APIs*

Quectel provides some QAPIs for customers to check (U)SIM card status and get related information.

This chapter describes the following QAPIs:

```
qapi_QT_SIM_RDY_Check*
qapi_QT_SIM_IMSI_Get*
qapi_QT_SIM_MSISDN_Get*
qapi_QT_SIM_CCID_Get*
```

NOTE

*) means the QAPI is not support now

2.3.1. API Functions

2.3.1.1. qapi_QT_SIM_RDY_Check*

This function is used to query SIM status.

- **Prototype**

```
qapi_Status_t qapi_QT_SIM_RDY_Check(char* status);
```

- **Parameters**

status:
[out] char: Pointer, get the (U)SIM status.

- **Return Value**

QAPI_QT_ERR_OK on success, and others on errors.

2.3.1.2. qapi_QT_SIM_IMSI_Get*

This function is used to query (U)SIM card IMSI.

- **Prototype**

```
qapi_Status_t qapi_QT_SIM_IMSI_Get(char* imsi);
```

- **Parameters**

imsi:
[out] char: pointer, get (U)SIM card’s IMSI.

- **Return Value**

QAPI_QT_ERR_OK on success, and others on errors.

2.3.1.3. qapi_QT_SIM_MSISDN_Get*

This function is used to query (U)SIM card MSISDN.

- **Prototype**

```
qapi_Status_t qapi_QT_SIM_MSISDN_Get(char* msisdn);
```

- **Parameters**

msisdn:
[out] char: Pointer, get (U)SIM card MSISDN.

- **Return Value**

QAPI_QT_ERR_OK on success, and others on errors.

2.3.1.4. qapi_QT_SIM_CCID_Get*

This function is used to query (U)SIM card ICCID.

- **Prototype**

```
qapi_Status_t qapi_QT_SIM_CCID_Get(char* ccid);
```

- **Parameters**

ccid:
[out] char: pointer, get (U)SIM card ICCID.

- **Return Value**

QAPI_QT_ERR_OK on success, and others on errors.

2.4. SMS APIs*

Quectel provides some QAPIs for customers to send/receive/delete SMS message and also other related operation.

This chapter describes the following QAPIs:

```
qapi_QT_SMS_CPMS_Set*
qapi_QT_SMS_CPMS_Get*
qapi_QT_SMS_Rcvd_Num*
qapi_QT_SMS_Message_Read*
qapi_QT_SMS_Message_Delete*
```

```
qapi_QT_SMS_Message_Send*
qapi_QT_SMS_Para_Set*
qapi_QT_SMS_Para_Get*
qapi_QT_SMS_Charset_Set*
qapi_QT_SMS_Charset_Get*
```

NOTE

*) means the QAPI is not support now

2.4.1. Data Structure

2.4.1.1. Enumeration Type

2.4.1.1.1. Enum qapi_QT_SMS_Mem_e

```
typedef enum {
        QT_WMS_MEMORY_STORE_NONE=0,
        QT_WMS_MEMORY_STORE_SM=1,
        QT_WMS_MEMORY_STORE_ME=2,
        QT_WMS_MEMORY_STORE_MT=3,

        QT_WMS_MEMORY_STORE_MAX
} qapi_QT_SMS_Mem_e;
```

- **Parameters**

Parameter	Description
QT_WMS_MEMORY_STORE_NONE	No memory storage
QT_WMS_MEMORY_STORE_SM	(U)SIM message storage
QT_WMS_MEMORY_STORE_ME	Mobile equipment message storage
QT_WMS_MEMORY_STORE_MT	Same as "*QT_WMS_MEMORY_STORE_ME*" storage
QT_WMS_MEMORY_STORE_MAX	Invalid message storage

2.4.1.1.2. Enum qapi_QT_SMS_Status_e

```
typedef enum {
        QT_SMS_REC_UNREAD=0,
        QT_SMS_REC_READ=1,
        QT_SMS_STO_UNSENT=2,
        QT_SMS_STO_SENT=3,
        QT_SMS_ALL=4,

        QT_SMS_STATUS_MAX
} qapi_QT_SMS_Status_e;
```

- **Parameters**

Parameter	Description
QT_SMS_REC_UNREAD	Received unread messages
QT_SMS_REC_READ	Received read messages
QT_SMS_STO_UNSENT	Stored unsent messages
QT_SMS_STO_SENT	Stored sent messages
QT_SMS_ALL	All messages
QT_SMS_STATUS_MAX	Invalid SMS status

2.4.1.1.3. Enum qapi_QT_SMS_Char_Set_e

```
typedef enum{
        QT_ALPHA_GSM=0,
        QT_ALPHA_IRA=1,
        QT_ALPHA_UCS2=2,

        QT_ALPHA_MAX
} qapi_QT_SMS_Char_Set_e;
```

- **Parameters**

Parameter	Description

QT_ALPHA_GSM	GSM default alphabet
QT_ALPHA_IRA	International reference alphabet
QT_ALPHA_UCS2	UCS2 alphabet
QT_ALPHA_MAX	Invalid alphabet

2.4.1.2. Structure Type

2.4.1.2.1. qapi_QT_SMS_Mem_Info_t

```
typedef struct {
        qapi_QT_SMS_Mem_e  SMS_mem;
        uint8_t used;
        uint8_t total;
} qapi_QT_SMS_Mem_Info_t;
```

- **Parameters**

Type	Parameter	Description
qapi_QT_SMS_Mem_e	*SMS_mem*	memory storages
uint8_t	*used*	Number of current messages in < *SMS_mem* >
uint8_t	*total*	Total number of messages which can be stored in < *SMS_mem* >

2.4.1.2.2. qapi_QT_SMS_Cpms_Set_t

```
typedef struct {
        qapi_QT_SMS_Mem_e  mem1;
        qapi_QT_SMS_Mem_e  mem2;
        qapi_QT_SMS_Mem_e  mem3;
} qapi_QT_SMS_Cpms_Set_t;
```

- **Parameters**

Type	Parameter	Description
qapi_QT_SMS_Mem_e	*mem1*	Messages to be read and deleted from this memory storage

qapi_QT_SMS_Mem_e	*mem2*	Messages will be written and sent to this memory storage
qapi_QT_SMS_Mem_e	*mem3*	Received messages will be placed in this memory storage if not route to PC.

2.4.1.2.3. qapi_QT_SMS_Cpms_Query_t

```
typedef struct {
       qapi_QT_SMS_Mem_Info_t mem1;
       qapi_QT_SMS_Mem_Info_t mem2;
       qapi_QT_SMS_Mem_Info_t mem3;
}qapi_QT_SMS_Cpms_Query_t;
```

- **Parameters**

Type	Parameter	Description
qapi_QT_SMS_Mem_Info_t	*mem1*	Messages storage type and status which to be read and deleted
qapi_QT_SMS_Mem_Info_t	*mem2*	Messages storage type and status which to be written and sent
qapi_QT_SMS_Mem_Info_t	*mem3*	Messages storage type and status which Received messages will be placed if not route to PC.

2.4.1.2.4. qapi_QT_SMS_Message_Content_t

```
typedef struct {
       char *address;
       char *message;
       size_t len;
} qapi_QT_SMS_Message_Content_t;
```

- **Parameters**

Type	Parameter	Description
char	*address*	Originating or destination address
char	*message*	SMS message content
size_t	*len*	SMS message content length

2.4.1.2.5. qapi_QT_SMS_Message_Rcvd_t

```
typedef struct{
  time_t time;
  qapi_QT_SMS_Status_e status;
  qapi_QT_SMS_Message_Info_t sms_info;
}qapi_QT_SMS_Message_Rcvd_t;
```

- **Parameters**

Type	Parameter	Description
time_t	*time*	Service center time stamp
qapi_QT_SMS_Status_e	*status*	Received message status(read or unread)
qapi_QT_SMS_Message_Info_t	*sms_info*	Received message information

2.4.1.2.6. qapi_QT_SMS_Para_t

```
typedef struct {
    uint8 fo;
    uint8 vp;
    uint8 pid;
    uint8 dcs;
} qapi_QT_SMS_Para_t;
```

- **Parameters**

Type	Parameter	Description
uint8	*fo*	First octet, refer to 3GPP TS 23.040
uint8	*vp*	Validity period, refer to 3GPP TS 23.040
uint8	*pid*	Protocol identifier, refer to 3GPP TS 23.040
uint8	*dcs*	Data coding scheme, refer to 3GPP TS 23.038

2.4.2. API Functions

2.4.2.1. qapi_QT_SMS_CPMS_Set*

This function is used to set preferred message storage.

- **Prototype**

```
qapi_QT_Status_t qapi_QT_SMS_CPMS_Set(qapi_QT_SMS_Cpms_Set_t* para);
```

- **Parameters**

para:
[in] Pointer, set module's preferred message storage.

- **Return Value**

QAPI_QT_ERR_OK on success, and others on errors.

2.4.2.2. qapi_QT_SMS_CPMS_Get*

This function is used to get preferred message storage.

- **Prototype**

```
qapi_QT_Status_t qapi_QT_SMS_CPMS_Get(qapi_QT_SMS_Cpms_Query_t* para);
```

- **Parameters**

para:
[out] Pointer, get module's preferred message storage.

- **Return Value**

QAPI_QT_ERR_OK on success, and others on errors.

2.4.2.3. qapi_QT_SMS_Rcvd_Num*

This function is used to get received message number.

- **Prototype**

```
qapi_QT_Status_t qapi_QT_SMS_Rcvd_Num(uint8 * rec_sms_number);
```

- **Parameters**

rec_sms_number:
[out] uint8: Pointer, get the received SMS number.

- **Return Value**

QAPI_QT_ERR_OK on success, and others on errors.

2.4.2.4. qapi_QT_SMS_Message_Read*

This function is used to read the specified index SMS.

- **Prototype**

```
qapi_QT_Status_t qapi_QT_SMS_Message_Read(uint16_t index, qapi_QT_SMS_Message_Rcvd_t*
info)
```

- **Parameters**

index:
[in] SMS index which need to read.

Info:
[out] Pointer, store the SMS content.

- **Return Value**

QAPI_QT_ERR_OK on success, and others on errors.

2.4.2.5. qapi_QT_SMS_Message_Numseq_Read*

This function is used to read SMS through sequence number.

- **Prototype**

```
qapi_QT_Status_t qapi_QT_SMS_Message_Numseq_Read(uint16_t numseq, uint16_t *index,
qapi_QT_SMS_Message_Rcvd_t* info)
```

- **Parameters**

numseq:
[in] SMS number sequence which need to read.

index:
[out] Pointer, store the SMS index.

Info:
[out] Pointer, store the SMS content.

- **Return Value**

QAPI_QT_ERR_OK on success, and others on errors.

2.4.2.6. qapi_QT_SMS_Message_Delete*

This function is used to delete the specified index SMS.

- **Prototype**

```
qapi_QT_Status_t qapi_QT_SMS_Message_Delete(uint16_t index);
```

- **Parameters**

index:
[in] uint16_t: SMS index which need to delete.

- **Return Values**

QAPI_QT_ERR_OK on success, and others on errors.

2.4.2.7. qapi_QT_SMS_Message_Send*

This function is used to send out SMS.

- **Prototype**

```
qapi_QT_Status_t qapi_QT_SMS_Message_Send(qapi_QT_SMS_Message_Content_t* message);
```

- **Parameters**

message:
[out] SMSMessageContent_t: Pointer, the SMS message which need to send out.

- **Return Values**

QAPI_QT_ERR_OK on success, and others on errors.

2.4.2.8. qapi_QT_SMS_Para_Set*

This function is used to set SMS text mode parameters.

- **Prototype**

```
qapi_QT_Status_t qapi_QT_SMS_Para_Set(qapi_QT_SMS_Para_t* para);
```

- **Parameters**

para:
[in] the text mode parameter which need to set.

- **Return Values**

QAPI_QT_ERR_OK on success, and others on errors.

2.4.2.9. qapi_QT_SMS_Para_Get*

This function is used to get SMS text mode parameters.

- **Prototype**

```
qapi_QT_Status_t qapi_QT_SMS_Para_Get(qapi_QT_SMS_Para_t* para);
```

- **Parameters**

para:
[out] Pointer, get the text mode parameter.

- **Return Values**

QAPI_QT_ERR_OK on success, and others on errors.

2.4.2.10. qapi_QT_SMS_Charset_Set*

This function is used to inform the module which character set is used. This enables the UE to convert character strings correctly.

- **Prototype**

```
qapi_QT_Status_t qapi_QT_SMS_Charset_Set(qapi_QT_SMS_Char_Set_t* para);
```

- **Parameters**

para:
[in] set character set.

- **Return Values**

QAPI_QT_ERR_OK on success, and others on errors.

2.4.2.11. qapi_QT_SMS_Charset_Get*

This function is used to get module's current character set.

- **Prototype**

```
qapi_QT_Status_t qapi_QT_SMS_Charset_Get(qapi_QT_SMS_Char_Set_t* para);
```

- **Parameters**

para:
[out] Get character set.

- **Return Values**

QAPI_QT_ERR_OK on success, and others on errors.

2.5. FTP APIs*

Quectel provides some QAPIs to support FTP client function.

This chapter describes the following QAPIs:

```
qapi_QT_Net_FTPc_Start*
qapi_QT_Net_FTPc_Stop*
qapi_QT_Net_FTPc_New_sess*
qapi_QT_NET_FTPc_Free_sess*
qapi_QT_NET_FTPc_Set_Param*
qapi_QT_NET_FTPc_Get_Param*
qapi_QT_NET_FTPc_Conn*
qapi_QT_NET_FTPc_Disc*
```

```
qapi_QT_NET_FTPc_Cmd*
```

NOTE

*) means the QAPI is not support now

2.5.1. Data Structure

2.5.1.1. Enumeration Type

2.5.1.1.1. Enum qapi_Net_FTPc_Parameter_e

FTP setting type.

```
typedef enum {
      QAPI_NET_FTP_PARAM_USERNAME,              /* set username  */
      QAPI_NET_FTP_PARAM_PASSWD,                /* set password */
      QAPI_NET_FTP_PARAM_RESP_TIMEOUT,          /* response timeout */

      QAPI_NET_FTP_ PARAM _MAX                  /* Don't exceed this value */
} qapi_Net_FTPc_Parameter_e;
```

- **Parameters**

Parameter	**Description**
QAPI_NET_FTP_PARAM_USERNAME	The username of FTP server account.
QAPI_NET_FTP_PARAM_PASSWD	The password of FTP server account.
QAPI_NET_FTP_PARAM_RESP_TIMEOUT	Response timeout value.

2.5.1.1.2. Enum qapi_Net_FTPc_Command_e

FTP request command type.

```
typedef enum {
      QAPI_NET_FTP_CMD_ASCII,  /* set ASCII mode */
      QAPI_NET_FTP_CMD_BIN,    /* set Binary mode */
      QAPI_NET_FTP_CMD_CWD,    /* change directory */
      QAPI_NET_FTP_CMD_GET,    /* Download file which in ftp server */
```

```
    QAPI_NET_FTP_CMD_PUT,    /* Upload file to server */
    QAPI_NET_FTP_CMD_DEL,    /* Delete file from server */
    QAPI_NET_FTP_CMD_PWD,    /* List current working directory */
    QAPI_NET_FTP_CMD_MKDIR, /* Create a new directory */
    QAPI_NET_FTP_CMD_RMDIR, /* Remove a new directory */

    QAPI_NET_FTP_CMD_MAX    /* Don't exceed this value */
} qapi_Net_FTPc_Command_e;
```

- **Parameters**

Parameter	Description
QAPI_NET_FTP_CMD_ASCII	Set FTP transport mode to ASCII mode.
QAPI_NET_FTP_CMD_BIN	Set FTP transport mode to Binary mode.
QAPI_NET_FTP_CMD_CWD	Enter into the directory on the FTP server.
QAPI_NET_FTP_CMD_GET	Download resource from FTP server.
QAPI_NET_FTP_CMD_PUT	Upload resource to FTP server.
QAPI_NET_FTP_CMD_DEL	Delete the directory on the FTP server.
QAPI_NET_FTP_CMD_PWD	View the current working directory on the FTP server.
QAPI_NET_FTP_CMD_MKDIR	Create a directory on the FTP server.
QAPI_NET_FTP_CMD_RMDIR	Delete a directory on the FTP server.

2.5.1.2. Definition and Typedef Type

2.5.1.2.1. Definition Type

FTP client session basic macro.

```
#define QAPI_NET_FTP_USRNAME_MAX_LEN        (64)
#define QAPI_NET_FTP_PASSWD_MAX_LEN         (64)
#define QAPI_NET_FTP_SRV_MAX_LEN            (200)
```

2.5.1.2.2. typedef void * qapi_Net_FTPc_handle_t handle

The handle to FTP client session.

```
typedef void * qapi_Net_FTPc_handle_t handle;
```

2.5.1.2.3. typedef void (*qapi_Net_FTPc_CB_t)(int32_t resp_code, void *user_data)

FTP response user callback registered during qapi_Net_FTPc_New_sess().

- **Prototype**

```
typedef void (*qapi_Net_FTPc_CB_t)(int32_t protocol_code, void *user_data);
```

- **Parameters**

protocol_code:
[out] The FTP protocol code which from FTP server.

user_data:
[out] The FTP response data information.

- **Return Value**

None.

2.5.1.3. Structure Type

2.5.1.3.1. qapi_Net_FTPc_Context_t

Structure to configure an FTP client session.

```
typedef struct {
    uint8_t server[QAPI_NET_FTP_SRV_MAX_LEN];
    uint16_t port;

    uint8_t username[QAPI_NET_FTP_USRNAME_MAX_LEN];
    uint8_t password[QAPI_NET_FTP_PASSWD_MAX_LEN];

    /* Security mode */
    uint8_t security_mode;

    qapi_Net_SSL_Obj_Hdl_t  sslCtx;
    qapi_Net_SSL_Con_Hdl_t  ssl;
    qapi_Net_SSL_Config_t  config;
    qapi_Net_SSL_Role_t  role;
    qapi_Net_FTPc_CB_t  cb;
    void *user_data;
} qapi_Net_FTPc_Context_t;
```

- **Parameters**

Type	Parameter	Description
uint8_t	*server*	The FTP server address.
uint16_t	*port*	The FTP server port.
uint8_t	*username*	The username of FTP server account.
uint8_t	*password*	The password of FTP server account.
uint8_t	*security_mode*	Security mode.
qapi_Net_SSL_Obj_Hdl_t	*sslCtx*	Handle to an SSL object.
qapi_Net_SSL_Con_Hdl_t	*ssl*	Handle to an SSL connection
qapi_Net_SSL_Config_t	*config*	Structure to configure an SSL connection.
qapi_Net_SSL_Role_t	*role*	SSL object role.
qapi_Net_FTPc_CB_t	*cb*	FTP response user callback.
void	*user_data*	User data payload to be returned by the callback function.

2.5.2. API Functions

2.5.2.1. qapi_QT_Net_FTPc_Start*

Start or restart FTP client module.

This function is invoked to start or restart the FTP client after it is stopped via a call to qapi_QT_Net_FTPc_Stop().

- **Prototype**

```
qapi_Status_t qapi_QT_Net_FTPc_Start(void);
```

- **Parameters**

None.

- **Return Value**

QAPI_QT_ERR_OK on success, and others on errors.

2.5.2.2. qapi_QT_Net_FTPc_Stop*

Stop FTP client module.

This function is invoked to stop the FTP client after it was started via a call to *qapi_QT_Net_FTPc_Start*.

- **Prototype**

```
qapi_Status_t qapi_QT_Net_FTPc_Stop(void);
```

- **Parameters**

None.

- **Return Value**

QAPI_QT_ERR_OK on success, and others on errors.

2.5.2.3. qapi_QT_Net_FTPc_New_sess*

Creates a new FTP client session.

To create a client session, the caller must invoke this function and the handle to the newly created context is returned if successful. As part of the function call, a user callback function is registered with the FTP client module that gets invoked for that particular session if there is some response data from the FTP server. Passing in the SSL context information ensures that a secure session is created.

- **Prototype**

```
qapi_Net_FTPc_handle_t*                    qapi_QT_Net_FTPc_New_sess(qapi_Net_SSL_Obj_Hdl_t
ssl_Object_Handle, qapi_Net_FTPc_CB_t callback, void *userData);
```

- **Parameters**

ssl_Object_handle:
[in] SSL context for FTPs connect (zero for no FTPs session support).

callback:
[in] Register a callback function; NULL for no support for a callback.

userData:
[in] User data payload to be returned by the callback function.

- **Return Value**

QAPI_QT_ERR_OK on success, and others on errors.

2.5.2.4. qapi_QT_Net_FTPc_Free_sess*

An FTP client session that is connected to the FTP server is disconnected before releasing the resources associated with that session.

- **Prototype**

```
qapi_Status_t qapi_QT_Net_FTPc_Free_sess(qapi_Net_FTPc_handle_t handle);
```

- **Parameters**

handle:
[in] Handle to FTP session.

- **Return Value**

QAPI_QT_ERR_OK on success, and others on errors.

2.5.2.5. qapi_QT_Net_FTPc_Set_Param*

Sets FTP client session parameter.

Multiple invocations of this function will result in appending the parameter key-value pair information to the internal data buffer.

- **Prototype**

```
qapi_Status_t qapi_QT_Net_FTPc_Set_Param(qapi_Net_FTPc_handle_t handle,
qapi_Net_FTPc_Parameter_e key, const char *value);
```

- **Parameters**

handle:
[in] Handle to FTP client session.

key:
[in] The FTP key related information.

value:
[in] The FTP value associated with the key.

- **Return Value**

QAPI_QT_ERR_OK on success, and others on errors.

2.5.2.6. qapi_QT_Net_FTPc_Conn*

Connects FTP client session to the FTP server.

- **Prototype**

```
qapi_Status_t qapi_QT_Net_FTPc_Conn(qapi_Net_FTPc_handle_t handle, const char *url, uint16_t
port);
```

- **Parameters**

handle:
[in] Handle to FTP client session.
url:
[in] The FTP server URL information.

port:
[in] The FTP server port information.

- **Return Value**

QAPI_QT_ERR_OK on success, and others on errors.

2.5.2.7. qapi_QT_Net_FTPc_Disc*

Disconnects the FTP client session from FTP server.

The FTP client session that is connected to the FTP server is disconnected from the FTP server.

- **Prototype**

```
qapi_Status_t qapi_QT_Net_FTPc_Disc(qapi_Net_FTPc_handle_t handle);
```

- **Parameters**

handle:
[in] Handle to FTP client session.

- **Return Value**

QAPI_QT_ERR_OK on success, and others on errors.

2.5.2.8. qapi_FTPc_Cmd*

Send FTP command to the FTP server via FTP client session.

- **Prototype**

```
qapi_Status_t   qapi_FTPc_Cmd(qapi_Net_FTPc_handle_t   handle,   qapi_Net_FTPc_Command_e
*cmd, const char *args);
```

- **Parameters**

handle:
[in] Handle to FTP client session.

cmd:
[in] The FTP request command information.

args:
[in] The user data associated with FTP request command.

- **Return Value**

QAPI_QT_ERR_OK on success, and others on errors.

2.6. URC APIs*

Quectel provides some QAPIs for customers to process URC information.

NOTE

$^{*)}$ means the QAPI is not support now

2.6.1. Data Structure

2.6.1.1. Enumeration Type

2.6.1.1.1. Enum qapi_QT_URC_MASK_e

```
typedef enum {
    QT_URC_MASK_NONE  = 0,
    QT_URC_MASK_POWER_ON_REASON  = 1,
    QT_URC_MASK_SMS_RCVD  = 2,
    QT_URC_MASK_SMS_SENT = 3,
    QT_URC_MASK_SIM_HOTSWAP = 4,

    QT_URC_MASK_OTHERS  = 999,
    QT_URC_MASK_MAX
} qapi_QT_URC_MASK_e;
```

- **Parameters**

Parameter	Description
QT_URC_MASK_POWER_ON_REASON	Module power on reason.
QT_URC_MASK_SMS_RCVD	Module received a new SMS.
QT_URC_MASK_SMS_SENT	Module sent a new SMS.
QT_URC_MASK_SIM_HOTSWAP	Module has been plugged and unplugged.

2.6.1.1.2. Enum qapi_QT_SIM_PLUG_e

```
typedef enum {
    QT_SIM_PLUG_NONE  = 0,
    QT_SIM_PLUG_IN = 1,
    QT_SIM_PLUG_OUT  = 2,

    QT_SIM_PLUG_MAX
} qapi_QT_SIM_PLUG_e;
```

- **Parameters**

Parameter	Description

QT_URC_MASK_POWER_ON_REASON	Module power on reason.
QT_URC_MASK_SMS_RCVD	Module received a new SMS.
QT_URC_MASK_SMS_SENT	Module sent a new SMS.
QT_URC_MASK_SIM_HOTSWAP	Module has been plugged and unplugged.

2.6.1.2. Typedef Type

Callback function. Handle URC reports.

```
typedef void (*qapi_URC_CB_t)(int16_t mask, void* info);
```

2.6.1.3. Structure Type

Structure is used for SMS related URC operations.

```
typedef struct {
      uint8_t status;
      uint16_t ret;
} qapi_QT_URC_SMS_t;
```

- **Parameters**

Type	Parameter	Description
Uint8_t	*status*	Success or failure in sending or receiving SMS.
uint16_t	*ret*	Result code for sending or receiving a SMS.

2.6.2. API Functions

2.6.2.1. qapi_QT_Reg_URC_CB_Hdlr*

This function is used to register a callback function to the kernel space for process URC information.

- **Prototype**

```
qapi_QT_Status_t qapi_QT_Reg_URC_CB_Hdlr(qapi_URC_CB_t cb);
```

- **Parameters**

cb:
[in] callback function. See *qapi_URC_CB_t* definition.

- **Return Value**

QAPI_QT_ERR_OK on success, and others on errors.

3 References

Table 1: Related Documents

SN	Document Name	Remark
[1]	Quectel_BG95_AT_Commands_Manual	BG95 AT Commands Manual
[2]	Quectel_BG95-QuecOpen_Hardware_Design	BG95-QuecOpen Hardware Design
[3]	80-P8101-32 Qualcomm Application Programming Interface Specification	Qualcomm QAPI introduction

Table 2: Terms and Abbreviations

Abbreviation	Description
API	Qualcomm Application Programming Interface
AP	Application Processor
ICCID	Integrated Circuit Card ID
FTP	File Transfer Protocol
IMEI	International Mobile station Equipment Identity
MP	Modem Processor
MSISDN	Mobile Subscriber International ISDN
RAM	Security Socket Layer
RAT	Radio Access Type
SIM	Subscriber Identity Module