EM7590

libQMI User Guide

41114537
Rev 1

SIERRA WIRELESS®

Important Notice

Due to the nature of wireless communications, transmission and reception of data can never be guaranteed. Data may be delayed, corrupted (i.e., have errors) or be totally lost. Although significant delays or losses of data are rare when wireless devices such as the Sierra Wireless modem are used in a normal manner with a well-constructed network, the Sierra Wireless modem should not be used in situations where failure to transmit or receive data could result in damage of any kind to the user or any other party, including but not limited to personal injury, death, or loss of property. Sierra Wireless accepts no responsibility for damages of any kind resulting from delays or errors in data transmitted or received using the Sierra Wireless modem, or for failure of the Sierra Wireless modem to transmit or receive such data.

Safety and Hazards

Do not operate the Sierra Wireless modem in areas where blasting is in progress, where explosive atmospheres may be present, near medical equipment, near life support equipment, or any equipment which may be susceptible to any form of radio interference. In such areas, the Sierra Wireless modem **MUST BE POWERED OFF**. The Sierra Wireless modem can transmit signals that could interfere with this equipment.

Do not operate the Sierra Wireless modem in any aircraft, whether the aircraft is on the ground or in flight. In aircraft, the Sierra Wireless modem **MUST BE POWERED OFF**. When operating, the Sierra Wireless modem can transmit signals that could interfere with various onboard systems.

Note: Some airlines may permit the use of cellular phones while the aircraft is on the ground and the door is open. Sierra Wireless modems may be used at this time.

The driver or operator of any vehicle should not operate the Sierra Wireless modem while in control of a vehicle. Doing so will detract from the driver or operator's control and operation of that vehicle. In some states and provinces, operating such communications devices while in control of a vehicle is an offence.

Limitation of Liability

The information in this manual is subject to change without notice and does not represent a commitment on the part of Sierra Wireless. SIERRA WIRELESS AND ITS AFFILIATES SPECIFICALLY DISCLAIM LIABILITY FOR ANY AND ALL DIRECT, INDIRECT, SPECIAL, GENERAL, INCIDENTAL, CONSEQUENTIAL, PUNITIVE OR EXEMPLARY DAMAGES INCLUDING, BUT NOT LIMITED TO, LOSS OF PROFITS OR REVENUE OR ANTICIPATED PROFITS OR REVENUE ARISING OUT OF THE USE OR INABILITY TO USE ANY SIERRA WIRELESS PRODUCT, EVEN IF SIERRA WIRELESS AND/OR ITS AFFILIATES HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES OR THEY ARE FORESEEABLE OR FOR CLAIMS BY ANY THIRD PARTY.

Notwithstanding the foregoing, in no event shall Sierra Wireless and/or its affiliates aggregate liability arising under or in connection with the Sierra Wireless product, regardless of the number of events, occurrences, or claims giving rise to liability, be in excess of the price paid by the purchaser for the Sierra Wireless product.

Patents

This product may contain technology developed by or for Sierra Wireless Inc. This product includes technology licensed from QUALCOMM®. This product is manufactured or sold by Sierra Wireless Inc. or its affiliates under one or more patents licensed from InterDigital Group and MMP Portfolio Licensing.

Copyright

Trademarks

Sierra Wireless®, AirLink®, AirVantage® and the Sierra Wireless logo are registered trademarks of Sierra Wireless, Inc.

Windows® is a registered trademark of Microsoft Corporation.

QUALCOMM® is a registered trademark of QUALCOMM Incorporated. Used under license.

Other trademarks are the property of their respective owners.

Contact Information

Sales information and technical support, including warranty and returns	Web: sierrawireless.com/company/contact-us/ Global toll-free number: 1-877-687-7795 6:00 am to 6:00 pm PST
Corporate and product information	Web: sierrawireless.com

Revision History

Revision number	Release date	Changes
1	April 2023	Document creation

Contents

Introduction 6

Purpose 6

Installation 7

Confirm the EM7590 is mounted 7

Install libqmi 8

Install with apt-get 8

Install manually 8

libqmi APIs 10

DMS Service APIs 10

NAS Service APIs 15

WDS Service APIs 25

PBM Service APIs 30

PDC Service APIs 31

UIM Service APIs 32

SAR Service APIs 35

WDA Service APIs 36

LOC Service APIs 36

DSD Service APIs 37

QOS Service APIs 38

Link Management APIs 38

Qmiwwan APIs 39

Test Cases 40

PDN connection 40

libqmi connection testing with multiple PDNs 41

References 44

Sierra Wireless Documents 44

List of Tables

Table 3-1: DMS Service APIs 10
Table 3-2: NAS Service APIs 15
Table 3-3: WDS Service APIs 25
Table 3-4: PBM Service APIs 30
Table 3-5: PDC Service APIs 31
Table 3-6: UIM Service APIs 32
Table 3-7: SAR Service APIs 35
Table 3-8: WDA Service APIs 36
Table 3-9: LOC Service APIs 36
Table 3-10: DSD Service APIs 37
Table 3-11: QOS Service APIs 38
Table 3-12: Link Management APIs 38
Table 3-13: Qmmiwan Service APIs 39

1: Introduction

1.1 Purpose

This document is a guide on how to use libqmi with EM7590 on Linux platforms.

For more information, refer to the libqmi web page at https://www.freedesktop.org/wiki/Software/libqmi/

2: Installation

To install libqmi on the host platform:

1. Confirm the EM7590 is mounted
2. Install libqmi

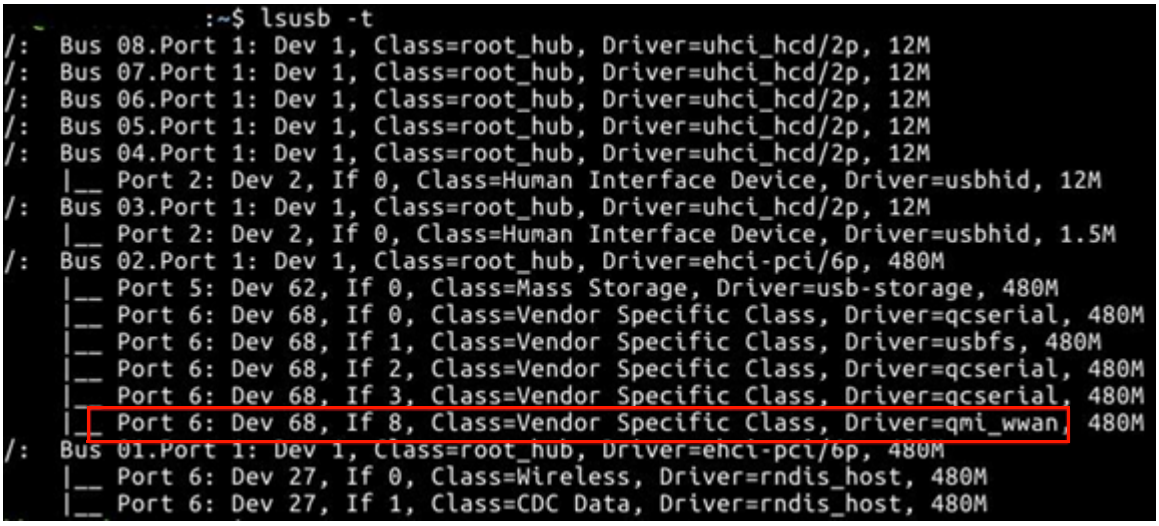
2.1 Confirm the EM7590 is mounted

Before installing libqmi on the host platform, make sure the EM7590 mounted successfully:

1. Display the USB interface details:

   ```
   $ lsusb -t
   ```

   For example:



2. If no response is displayed (i.e., the command returns immediately to the shell prompt), the USB interface has not enumerated. Some possible reasons include problems with the physical USB connection or the host platform Host Controller Interface (HCI). Resolve the problem, then go back to Step 1.
3. Locate the "`If 8`" line in the command response (as shown in the red outline above):
   - If the line includes "`Driver=qmi_wwan`", the QMI driver is hooked.
   - If the line includes "`Driver=`", the QMI driver has not been built. Refer to *[2] EM7590 Linux User Guide (Doc# 4134930)* for build instructions and then install the driver.
   - If there is no "`If 8`" line, reconfigure the USB interface composition:
     i. Use the **!USBCOMP** AT command to set the USB interface composition to RMNET0—refer to *[1] EM7590 AT Command Reference (Doc# 41114426)*.
     ii. Reset the module:

     AT!RESET

     iii. Go back to Step 1.

2.2 Install libqmi

Install libqmi on the host platform using the appropriate method:

- Install with apt-get (Only if the OS is Ubuntu 16.04 (or later) or Debian, and libqmi is in the apt/sources.list)
- Install manually

2.2.1 Install with apt-get

If the OS (Debian/Ubuntu) supports apt-get, and libqmi is in the apt/sources.list (Ubuntu 16.04 or later), use apt-get to install the required libraries:

1. Download package information:

   ```
   $ sudo apt-get update
   ```

2. Install libqmi:

   ```
   $ sudo apt-get install libqmi-utils
   ```

2.2.2 Install manually

If the OS does not support apt-get, or libqmi is not in the apt source list, libqmi must be installed manually.

2.2.2.1 Install manually on normal Linux OS

To install libqmi on a normal (non-embedded) Linux system:

1. Pre-install packages:
   - Debian/Ubuntu systems:

     ```
     $ sudo apt-get install libglib2.0-dev
     ```

   - Redhat/CentOS/Fedora systems:

     ```
     $ sudo yum makecache
     $ sudo yum -y install glib2-devel
     ```

2. Remove any old versions of libqmi. Note—In the following commands, go to the appropriate platform-specific folders (e.g., these examples remove files for an x86 platform. Pathnames will be different for ARM or other platforms):

   ```
   cd /lib/x86_64-linux-gnu/
   rm -rf libqmi*.*
   cd /usr/lib/x86_64-linux-gnu/
   rm -rf libqmi*.*
   ```

3. Download the libqmi source code:
   a. Go to https://www.freedesktop.org/software/libqmi/
   b. Select the libqmi version to download (e.g., libqmi-1.28.8.tar.xz).
   c. Extract the file to your working directory.
4. Configure libqmi:
   a. In your working directory, run the following command, which configures the libqmi makefile for your platform:

      ```
      $ ./configure
      ```

b. Check the response for any missing components (e.g., udev is missing in the example below):

```
checking for pkg-config... /usr/bin/pkg-config
checking pkg-config is at least version 0.9.0... yes
checking for GLIB... yes
checking for python... /usr/bin/python
checking for python version... 2.7
checking for python platform... linux2
checking for python script directory... ${prefix}/lib/python2.7/dist-packages
checking for python extension module directory... ${exec_prefix}/lib/python2.7/dist-packages
checking for gobject-introspection... no
checking how to run the C preprocessor... gcc -E
checking for GUDEV... no
configure: error: Couldn't find gudev >= 147. Install it, or otherwise configure using --without-udev to disable udev support.
```

- If any required components (e.g., python) are missing, install them and then repeat Step 4a.
- If any optional components (e.g., udev) are missing, repeat Step 4 using the indicated arguments to ignore them (e.g., "`--without-udev`").

5. Build and install libqmi:

```
$ make
$ make install
```

2.2.2.2 Install manually on embedded Linux OS

An embedded Linux system should have an SDK loaded that includes the libqmi package and the required toolchain (e.g., cross-compiler and libraries).

To install libqmi on an embedded Linux system (this example uses the Openwrt SDK):

1. Configure the Kernel.

```
$ cd openwrt      ← Go to the openwrt main directory
$ make menuconfig
```

The openwrt menu appears.

2. Select Utilities.

3. Highlight qmi-utils and press the spacebar to select it—the flag at the start of the line shows "<*>" if the option is selected.

4. Press <ESC> to save the configuration.

5. Re-build the Openwrt package.

6. Download the image to the device.

3: libqmi APIs

3.1 DMS Service APIs

Table 3-1: DMS Service APIs

libqmi command	Function	Example
--dms-get-ids	Get IDs	**$ sudo qmicli -p --device=/dev/cdc-wdm0 --dms-get-ids** [/dev/cdc-wdm0] Device IDs retrieved: ESN: '0' IMEI: 'unknown' MEID: 'unknown'
--dms-get-capabilities	Get capabilities	**$ sudo qmicli -p --device=/dev/cdc-wdm0 --dms-get-capabilities** [/dev/cdc-wdm0] Device capabilities retrieved: Max TX channel rate: '50000000' Max RX channel rate: '100000000' Data Service: 'non-simultaneous-cs-ps' SIM: 'supported' Networks: 'umts, lte'
--dms-get-manufacturer	Get manufacturer	**$ sudo qmicli -p --device=/dev/cdc-wdm0 --dms-get-manufacturer** [/dev/cdc-wdm0] Device manufacturer retrieved: Manufacturer: 'QUALCOMM INCORPORATED'
--dms-get-model	Get model	**$ sudo qmicli -p --device=/dev/cdc-wdm0 --dms-get-model** [/dev/cdc-wdm0] Device model retrieved: Model: '0'
--dms-get-revision	Get revision	**$ sudo qmicli -p --device=/dev/cdc-wdm0 --dms-get-revision** [/dev/cdc-wdm0] Device revision retrieved: Revision: 'MPSS.AT.2.5.1.c9-00173-SDX12_GENNTCH_PACK-1 1 [Dec 20 2021 05:00:00]'
--dms-get-msisdn	Get MSISDN	**$ sudo qmicli -p --device=/dev/cdc-wdm0 --dms-get-msisdn** [/dev/cdc-wdm0] Device MSISDN retrieved: MSISDN: '0928535572'
--dms-get-power-state	Get power state	**$ sudo qmicli -p --device=/dev/cdc-wdm0 --dms-get-power-state** [/dev/cdc-wdm0] Device power state retrieved: Power state: 'external-source' Battery level: '0 %'
--dms-get-hardware-revision	Get the HW revision	**$ sudo qmicli -p --device=/dev/cdc-wdm0 --dms-get-hardware-revision** [/dev/cdc-wdm0] Hardware revision retrieved: Revision: '10001'

Table 3-1: DMS Service APIs (Continued)

libqmi command	Function	Example
--dms-get-operating-mode	Get the device operating mode	**$ sudo qmicli -p --device=/dev/cdc-wdm0 --dms-get-operating-mode** [/dev/cdc-wdm0] Operating mode retrieved: Mode: 'online' HW restricted: 'no'
--dms-set-operating-mode=[(Operating mode)]	Set the device operating mode	**$ sudo qmicli -p --device=/dev/cdc-wdm0 --dms-set-operating-mode=online** [/dev/cdc-wdm0] Operating mode set successfully
--dms-get-time	Get the device time	**$ sudo qmicli -p --device=/dev/cdc-wdm0 --dms-get-time** [/dev/cdc-wdm0] Time retrieved: Time count: '1064124053246 (x 1.25ms): 2013-09-25 06:00:53' Time source: 'device' System time: '1330155066558 (ms): 2022-03-01 07:31:06' User time: '1673905 (ms): 1980-01-06 00:27:53'
--dms-read-eri-file	Read ERI file	**$ sudo qmicli -p --device=/dev/cdc-wdm0 --dms-read-eri-file** [/dev/cdc-wdm0] ERI file read: Size: '0' bytes Contents:
--dms-restore-factory-defaults=[(Service Programming Code)]	Restore factory defaults	**$ sudo qmicli -p --device=/dev/cdc-wdm0 --dms-restore-factory-defaults=000000** [/dev/cdc-wdm0] Factory defaults restored Device needs to get power-cycled for reset to take effect.
--dms-validate-service-programming-code=[(Service Programming Code)]	Validate the Service Programming Code	**$ sudo qmicli -p --device=/dev/cdc-wdm0 $ sudo qmicli -p --device=/dev/cdc-wdm0 --dms-validate-service-programming-code=000000** [/dev/cdc-wdm0] Service Programming Code validated
--dms-get-band-capabilities	Get band capabilities	**$ sudo qmicli -p --device=/dev/cdc-wdm0 $ sudo qmicli -p --device=/dev/cdc-wdm0 --dms-get-band-capabilities** [/dev/cdc-wdm0] Device band capabilities retrieved: Bands: 'wcdma-2100, wcdma-pcs-1900, wcdma-1700-us, wcdma-850-us, wcdma-800, wcdma-900, wcdma-1700-japan, wcdma-850-japan' LTE bands: '1, 2, 3, 4, 5, 7, 8, 12, 13, 14, 18, 19, 20, 25, 26, 28, 29, 32, 38, 39, 40, 41, 42, 43' LTE bands (extended): '1, 2, 3, 4, 5, 7, 8, 12, 13, 14, 18, 19, 20, 25, 26, 28, 29, 32, 38, 39, 40, 41, 42, 43, 48, 66, 71'
--dms-get-factory-sku	Get factory stock keeping unit	**$ sudo qmicli -p --device=/dev/cdc-wdm0 --dms-get-factory-sku** [/dev/cdc-wdm0] Device factory SKU retrieved: SKU: '10'

Table 3-1: DMS Service APIs (Continued)

libqmi command	Function	Example
--dms-get-software-version	Get software version	**$ sudo qmicli -p --device=/dev/cdc-wdm0 --dms-get-software-version** [/dev/cdc-wdm0] Software version: .1.c9-00173-SDX12_GENNTCH_PACK-1
--dms-swi-get-current-firmware	Get current firmware version	**$ sudo qmicli -d /dev/cdc-wdm0 --dms-swi-get-current-firmware** [/dev/cdc-wdm0] Successfully retrieved current firmware: Model: EM7590 Boot version: SWIX12C_01.01.00 AMSS version: SWIX12C_01.01.00 SKU ID: unknown Package ID: pakage_id Carrier ID: carrier_id Config version: CONFIG_V
--dms-get-firmware-preference	Get firmware preference	**$ sudo qmicli -d /dev/cdc-wdm0 --dms-get-firmware-preference** firmware preference successfully retrieved: [image 0] Image type: 'modem' Unique ID: '?_?' Build ID: '01.01.07.00_?' [image 1] Image type: 'pri' Unique ID: '001.002_001' Build ID: '01.01.07.00_GENERIC'

Table 3-1: DMS Service APIs (Continued)

libqmi command	Function	Example
--dms-list-stored-images	Get all stored images	**sudo qmicli -d /dev/cdc-wdm0 --dms-list-stored-images** [/dev/cdc-wdm0] Device list of stored images retrieved: [0] Type: 'modem' Maximum: '3' >>>>>>>>> [CURRENT] <<<<<<<<<< [modem0] Unique ID: '?_?' Build ID: '01.01.07.00_?' Storage index: '1' Failure count: '0' [modem1] Unique ID: '?_?' Build ID: '01.01.07.84_?' Storage index: '2' Failure count: '0' [modem2] Unique ID: '?_?' Build ID: '01.01.07.88_?' Storage index: '3' Failure count: '0' [1] Type: 'pri' Maximum: '25' >>>>>>>>> [CURRENT] <<<<<<<<<< [pri0] Unique ID: '001.002_001' Build ID: '01.01.07.00_GENERIC' Storage index: '1' Failure count: '0' [pri1] Unique ID: '001.001_001' Build ID: '01.01.07.00_PTCRB' Storage index: '2' Failure count: '0' [pri2] Unique ID: '001.004_000' Build ID: '01.01.07.00_VERIZON' Storage index: '3' Failure count: '0' [pri3] Unique ID: '001.002_003' Build ID: '01.01.07.00_ATT' Storage index: '4' Failure count: '0'

Table 3-1: DMS Service APIs (Continued)

libqmi command	Function	Example
		[pri4] Unique ID: '001.002_001' Build ID: '01.01.07.00_TMO' Storage index: '5' Failure count: '0'
--dms-select-stored-image=[modem(x),pri(x)]	Select stored image	**$ sudo qmicli -d /dev/cdc-wdm0 --dms-select-stored-image=modem1,pri1** [/dev/cdc-wdm0] Firmware preference successfully selected You may want to power-cycle the modem now, or just set it offline and reset it: $> sudo qmicli ... --dms-set-operating-mode=offline $> sudo qmicli ... --dms-set-operating-mode=reset After reset, the modem will wait in QDL mode for new firmware. Images to download: 'modem'
--dms-delete-stored-image=[modem(x),pri(x)]	Delete stored image	**$ sudo qmicli -d /dev/cdc-wdm0 --dms-delete-stored-image=pri1** [/dev/cdc-wdm0] Stored image successfully deleted

3.2 NAS Service APIs

Table 3-2: NAS Service APIs

libqmi command	Function	Example
--nas-get-signal-strength	Get signal strength	**$ sudo qmicli -p --device=/dev/cdc-wdm0 --nas-get-signal-strength** [/dev/cdc-wdm0] Successfully got signal strength Current: Network 'lte': '-55 dBm' RSSI: Network 'lte': '-55 dBm' ECIO: Network 'lte': '-2.5 dBm' IO: '-106 dBm' SINR (8): '9.0 dB' RSRQ: Network 'lte': '-3 dB' SNR: Network 'lte': '2.0 dB' RSRP: Network 'lte': '-84 dBm'
--nas-get-signal-info	Get signal info	**$ sudo qmicli -p --device=/dev/cdc-wdm0 --nas-get-signal-info** [/dev/cdc-wdm0] Successfully got signal info LTE: RSSI: '-65 dBm' RSRQ: '-13 dB' RSRP: '-85 dBm' SNR: '6.4 dB'
--nas-get-tx-rx-info=[(Radio Interface)]	Get TX/RX info	**$ sudo qmicli -p --device=/dev/cdc-wdm0 --nas-get-tx-rx-info=lte** [/dev/cdc-wdm0] Successfully got TX/RX info RX Chain 0: Radio tuned: 'yes' Power: '-52.7 dBm' ECIO: '-15.0 dB' RSRP: '-84.6 dBm' Phase: '0.00 degrees' RX Chain 1: Radio tuned: 'yes' Power: '-55.4 dBm' ECIO: '-13.8 dB' RSRP: '-86.2 dBm' Phase: '0.00 degrees' TX: In traffic: 'no'

Table 3-2: NAS Service APIs (Continued)

libqmi command	Function	Example
--nas-get-home-network	Get home network	**$ sudo qmicli -p --device=/dev/cdc-wdm0 --nas-get-home-network** [/dev/cdc-wdm0] Successfully got home network: Home network: 0MCC: '466' MNC: '1' Description: 'FET' Network name source: se13
--nas-get-serving-system	Get serving system	**$ sudo qmicli -p --device=/dev/cdc-wdm0 --nas-get-serving-system** [/dev/cdc-wdm0] Successfully got serving system: Registration state: 'registered' CS: 'attached' PS: 'attached' Selected network: '3gpp' Radio interfaces: '1' [0]: 'lte' Roaming status: 'off' Data service capabilities: '1' [0]: 'lte' Current PLMN: MCC: '466' MNC: '1' Description: 'FET' Roaming indicators: '1' [0]: 'off' (lte) 3GPP time zone offset: '480' minutes 3GPP daylight saving time adjustment: '0' hours 3GPP location area code: '65534' 3GPP cell ID: '26095129' Detailed status: Status: 'available' Capability: 'cs-ps' 0HDR Status: 'none' HDR Hybrid: 'no' Forbidden: 'no' LTE tracking area code: '19103' Full operator code info: MCC: '466' MNC: '1' MNC with PCS digit: 'no'

Table 3-2: NAS Service APIs (Continued)

libqmi command	Function	Example
--nas-get-system-info	Get system info	**$ sudo qmicli -p --device=/dev/cdc-wdm0 --nas-get-system-info** [/dev/cdc-wdm0] Successfully got system info: WCDMA service: Status: 'none' True Status: 'none' Preferred data path: 'no' LTE service: Status: 'available' True Status: 'available' Preferred data path: 'no' Domain: 'cs-ps' Service capability: 'cs-ps' Roaming status: 'off' Forbidden: 'no' Cell ID: '26095129' MCC: '466' MNC: '01' Tracking Area Code: '19103' Voice support: 'yes' IMS voice support: 'yes' eMBMS coverage info support: 'no' eMBMS coverage info trace ID: '65535' Cell access: 'all-calls' Registration restriction: 'unrestricted' Registration domain: 'not-applicable' SIM reject info: 'available'
--nas-get-technology-preference	Get technology preference	**$ sudo qmicli -p --device=/dev/cdc-wdm0 --nas-get-technology-preference** [/dev/cdc-wdm0] Successfully got technology preference Active: '3gpp, cdma-or-wcdma, lte', duration: 'permanent'

Table 3-2: NAS Service APIs (Continued)

libqmi command	Function	Example
--nas-get-system-selection-preference	Get system selection preference	**$ sudo qmicli -p --device=/dev/cdc-wdm0 --nas-get-system-selection-preference** [/dev/cdc-wdm0] Successfully got system selection preference Emergency mode: 'no' Mode preference: 'umts, lte' Disabled modes: '(null)' Band preference: 'bc-0-a-system, bc-0-b-system, bc-1-all-blocks, bc-2, bc-3-a-system, bc-4-all-blocks, bc-5-all-blocks, gsm-dcs-1800, gsm-900-extended, gsm-900-primary, bc-6, bc-7, bc-8, bc-9, bc-10, bc-11, gsm-450, gsm-480, gsm-750, gsm-850, gsm-900-railways, gsm-pcs-1900, wcdma-2100, wcdma-pcs-1900, wcdma-dcs-1800, wcdma-1700-us, wcdma-850-us, wcdma-800, bc-12, bc-14, bc-15, wcdma-2600, wcdma-900, wcdma-1700-japan, bc-16, bc-17, bc-18, bc-19, wcdma-850-japan, wcdma-1500' LTE band preference: '1, 2, 3, 4, 12, 13, 14, 18, 19, 20, 28, 29, 32, 38, 39, 40, 41, 42, 43' LTE band preference (extended): '1, 2, 3, 4, 12, 13, 14, 18, 19, 20, 28, 29, 32, 38, 39, 40, 41, 42, 43, 48, 66, 71' TD-SCDMA band preference: 'a, b, c, d, e, f' Roaming preference: 'any' Network selection preference: 'automatic' Service domain preference: 'cs-ps' GSM/WCDMA acquisition order preference: 'wcdma' Usage preference: 'voice-centric' Voice domain preference: 'cs-preferred' Registration restriction: 'unrestricted' Acquisition order preference: 'cdma-1x, gsm, umts, cdma-1xevdo, lte, td-scdma'
--nas-set-system-selection-preference=[cdma-1x|cdma-1xevdo|gsm|umts|lte|td-scdma][,[automatic|manual=MCCMNC]]	Set system selection preference	**$ sudo qmicli -p --device=/dev/cdc-wdm0 --nas-set-system-selection-preference=lte** [/dev/cdc-wdm0] System selection preference set successfully; replug your device.

Table 3-2: NAS Service APIs (Continued)

libqmi command	Function	Example
--nas-network-scan	Scan networks	**$ sudo qmicli -p --device=/dev/cdc-wdm0 --nas-network-scan** [/dev/cdc-wdm0] Successfully scanned networks Network [0]: MCC: '466' MNC: '1' Status: 'current-serving, home, not-forbidden, preferred' Description: 'FET' Network [1]: MCC: '466' MNC: '1' Status: 'available, home, not-forbidden, preferred' Description: 'FET' Network [2]: MCC: '466' MNC: '5' Status: 'available, roaming, forbidden, not-preferred' Description: 'APT' Network [3]: MCC: '466' MNC: '97' Status: 'available, roaming, forbidden, not-preferred' Description: 'TWM' Network [4]: MCC: '466' MNC: '92' Status: 'available, roaming, forbidden, not-preferred' Description: 'Chunghwa' Network [5]: MCC: '466' MNC: '12' Status: 'available, roaming, not-forbidden, not preferred' Description: 'APT' Network [6]: MCC: '466' MNC: '92' Status: 'available, roaming, forbidden, not-preferred' Description: 'Chunghwa' Network [7]: MCC: '466' MNC: '97' Status: 'available, roaming, forbidden, not-preferred' Description: 'TWM' Network [8]: MCC: '466' MNC: '89' Status: 'available, roaming, forbidden, not-preferred' Description: 'T Star' Network [0]: MCC: '466' MNC: '1' RAT: 'lte'

Table 3-2: NAS Service APIs (Continued)

libqmi command	Function	Example
		Network [1]: MCC: '466' MNC: '1' RAT: 'umts' Network [2]: MCC: '466' MNC: '5' RAT: 'lte' Network [3]: MCC: '466' MNC: '97' RAT: 'lte' Network [4]: MCC: '466' MNC: '92' RAT: 'umts' Network [5]: MCC: '466' MNC: '12' RAT: 'lte' Network [6]: MCC: '466' MNC: '92' RAT: 'lte' Network [7]: MCC: '466' MNC: '97' RAT: 'umts' Network [8]: MCC: '466' MNC: '89' RAT: 'umts' Network [0]: MCC: '466' MNC: '1' MCC with PCS digit: 'no' Network [1]: MCC: '466' MNC: '1' MCC with PCS digit: 'no' Network [2]: MCC: '466' MNC: '5' MCC with PCS digit: 'no' Network [3]: MCC: '466' MNC: '97' MCC with PCS digit: 'no' Network [4]: MCC: '466' MNC: '92' MCC with PCS digit: 'no' Network [5]:

Table 3-2: NAS Service APIs (Continued)

libqmi command	Function	Example
		MCC: '466' MNC: '12' MCC with PCS digit: 'no' Network [6]: MCC: '466' MNC: '92' MCC with PCS digit: 'no' Network [7]: MCC: '466' MNC: '97' MCC with PCS digit: 'no' Network [8]: MCC: '466' MNC: '89' MCC with PCS digit: 'no' Network scan result: success

Table 3-2: NAS Service APIs (Continued)

libqmi command	Function	Example
--nas-get-cell-location-info	Get Cell Location Info	**$ sudo qmicli -p --device=/dev/cdc-wdm0 --nas-get-cell-location-info** [/dev/cdc-wdm0] Successfully got cell location info Intrafrequency LTE Info UE In Idle: 'yes' PLMN: '46601' Tracking Area Code: '19103' Global Cell ID: '26095129' EUTRA Absolute RF Channel Number: '50' (E-UTRA band 1: 2100) Serving Cell ID: '337' Cell Reselection Priority: '6' S Non Intra Search Threshold: '42' Serving Cell Low Threshold: '0' S Intra Search Threshold: '62' Cell [0]: Physical Cell ID: '337' RSRQ: '-11.4' dB RSRP: '-83.6' dBm RSSI: '-55.1' dBm Cell Selection RX Level: '38' Cell [1]: Physical Cell ID: '89' RSRQ: '-12.3' dB RSRP: '-86.0' dBm RSSI: '-62.2' dBm Cell Selection RX Level: '36' Interfrequency LTE Info UE In Idle: 'yes' Frequency [0]: EUTRA Absolute RF Channel Number: '1550' (E-UTRA band 3: 1800+) Selection RX Level Low Threshold: '0' Cell Selection RX Level High Threshold: '20' Cell Reselection Priority: '6' Cell [0]: Physical Cell ID: '89' RSRQ: '-14.4' dB RSRP: '-85.5' dBm RSSI: '-63.4' dBm Cell Selection RX Level: '36' Cell [1]: Physical Cell ID: '337' RSRQ: '-15.3' dB RSRP: '-86.4' dBm RSSI: '-63.4' dBm Cell Selection RX Level: '35' Cell [2]: Physical Cell ID: '436' RSRQ: '-20.0' dB RSRP: '-96.6' dBm RSSI: '-63.4' dBm

Table 3-2: NAS Service APIs (Continued)

libqmi command	Function	Example
		Cell Selection RX Level: '25' Frequency [1]: EUTRA Absolute RF Channel Number: '9360' (E-UTRA band 28: 700 APT) Selection RX Level Low Threshold: '0' Cell Selection RX Level High Threshold: '4' Cell Reselection Priority: '5' Cell [0]: 00Physical Cell ID: '337' RSRQ: '-8.2' dB RSRP: '-81.9' dBm RSSI: '-66.0' dBm Cell Selection RX Level: '40' Cell [1]: Physical Cell ID: '436' RSRQ: '-11.1' dB RSRP: '-84.8' dBm RSSI: '-66.0' dBm Cell Selection RX Level: '37' LTE Info Neighboring GSM UE In Idle: 'yes' LTE Info Neighboring WCDMA UE In Idle: 'yes' LTE Timing Advance: 'unavailable'
--nas-force-network-search	Force network search	**$ sudo qmicli -p --device=/dev/cdc-wdm0 --nas-force-network-search** [/dev/cdc-wdm0] Successfully forced network search
--nas-get-operator-name	Get operator name data	**$ sudo qmicli -p --device=/dev/cdc-wdm0 --nas-get-operator-name** [/dev/cdc-wdm0] Successfully got operator name data NITZ information: Long Name: '' Short Name: '' Country: 'initials-do-not-add'
--nas-get-plmn-name=[mccmnc]	Get plmn name data	**$ sudo qmicli -p --device=/dev/cdc-wdm0 --nas-get-plmn-name=46601** [/dev/cdc-wdm0] Successfully got plmn name data 3GPP EONS PLMN Name: Long Name: 'Far EasTone' Short Name: 'FET' Service Name: '' Country: 'initials-do-not-add' 3GPP EONS PLMN Name with Language ID: 0: '??' ('??')Country: 'zh-trad' 1: '??' ('??')Country: 'zh-simp'

Table 3-2: NAS Service APIs (Continued)

libqmi command	Function	Example
--nas-get-rf-band-info	Get RF Band Info	**$ sudo qmicli -p --device=/dev/cdc-wdm0 --nas-get-rf-band-info** [/dev/cdc-wdm0] Successfully got RF band info Band Information: Radio Interface: 'lte' Active Band Class: 'eutran-1' Active Channel: '50' Band Information (Extended): Radio Interface: 'lte' Active Band Class: 'eutran-1' Active Channel: '50' Bandwidth: Radio Interface: 'lte' Bandwidth: '10
--nas-get-drx	Get DRX	**$ sudo qmicli -p --device=/dev/cdc-wdm0 --nas-get-drx** [/dev/cdc-wdm0] Successfully got DRX: unknown
--nas-swi-get-status	Get NAS status	**$ sudo qmicli -d /dev/cdc-wdm0 --nas-swi-get-status** [/dev/cdc-wdm0] Successfully got status: Common Info: Temperature: '25' Modem mode: 'online' System mode: 'wcdma' IMS registration state: 'no-srv' Packet service state: 'detached'

3.3 WDS Service APIs

Table 3-3: WDS Service APIs

libqmi command	Function	Example
--wds-start-network=["key=value,..."]	Start network (allowed keys: apn, 3gpp-profile, 3gpp2-profile, auth (PAP|CHAP|BOTH), username, password, autoconnect=yes, ip-type (4|6))	**$ sudo qmicli -p --device=/dev/cdc-wdm0 --wds-start-network="ip-type=4,apn=internet" --client-no-release-cid** [/dev/cdc-wdm0] Network started Packet data handle: '2404856720' [/dev/cdc-wdm0] Client ID not released: Service: 'wds' CID: '31' **$ sudo qmicli -p --device=/dev/cdc-wdm0 --set-expected-data-format=raw-ip** [/dev/cdc-wdm0] expected data format set to: raw-ip **$ sudo ip link set dev wwan0 up** **$ sudo udhcpc -q -f -n -i wwan0** udhcpc (v1.22.1) started Sending discover... Sending select for 100.111.17.32... Lease of 100.111.17.32 obtained, lease time 7200 deleting routers SIOCDELRT: DNS= 139.175.1.2 210.241.208.1 adding dns 139.175.1.2 adding dns 210.241.208.1 **$ ping 8.8.8.8** PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data. 64 bytes from 8.8.8.8: icmp_seq=1 ttl=52 time=26.4 ms 64 bytes from 8.8.8.8: icmp_seq=2 ttl=52 time=34.6 ms 64 bytes from 8.8.8.8: icmp_seq=3 ttl=52 time=32.9 ms
--wds-follow-network	Follow the network status until disconnected. Use with `--wds-start-network	**'$ sudo qmicli -p --device=/dev/cdc-wdm0 --wds-start-network="ip-type=4,apn=internet" --client-no-release-cid --wds-follow-network** [/dev/cdc-wdm0] Network started Packet data handle: '2405005728' Ctrl+C will stop the network **^C**cancelling the operation... Network cancelled... releasing resources Network cancelled... releasing resources error: operation failed: operation is cancelled [/dev/cdc-wdm0] Client ID not released: Service: 'wds' CID: '30'
--wds-stop-network=[Packet data handle] OR [disable-autoconnect]	Stop network	**$ sudo qmicli -p --device=/dev/cdc-wdm0 --wds-stop-network=2404856720 --client-cid=31** Network cancelled... releasing resources [/dev/cdc-wdm0] Network stopped

Table 3-3: WDS Service APIs (Continued)

libqmi command	Function	Example
--wds-get-current-settings	Get current settings	**$ sudo qmicli -p --device=/dev/cdc-wdm0 --wds-get-current-settings** [/dev/cdc-wdm0] Current settings retrieved: IP Family: IPv4 IPv4 address: 10.80.29.114 IPv4 subnet mask: 255.255.255.252 IPv4 gateway address: 10.80.29.113 IPv4 primary DNS: 210.241.208.1 IPv4 secondary DNS: 139.175.1.2 MTU: 1500 Domains: none
--wds-get-packet-service-status	Get packet service status	**$ sudo qmicli -p --device=/dev/cdc-wdm0 --wds-get-packet-service-status** [/dev/cdc-wdm0] Connection status: 'connected'
--wds-get-packet-statistics	Get packet statistics	**$ sudo qmicli -p --device=/dev/cdc-wdm0 --wds-get-packet-statistics** [/dev/cdc-wdm0] Connection statistics: TX packets OK: 0 RX packets OK: 0 TX packets dropped: 0 RX packets dropped: 0 TX bytes OK: 0 RX bytes OK: 0
--wds-get-current-data-bearer-technology	Get current data bearer technology	**$ sudo qmicli -p --device=/dev/cdc-wdm0 --wds-get-current-data-bearer-technology** [/dev/cdc-wdm0] Data bearer technology (current): Network type: '3gpp' Radio Access Technology: 'lte'
--wds-go-dormant	Make the active data connection go dormant	**$ sudo qmicli -p --device=/dev/cdc-wdm0 --wds-go-dormant**
--wds-go-active	Make the active data connection go active	**$ sudo qmicli -p --device=/dev/cdc-wdm0 --wds-go-active**
--wds-get-dormancy-status	Get the dormancy status of the active data connection	**$ sudo qmicli -p --device=/dev/cdc-wdm0 --wds-get-dormancy-status** [/dev/cdc-wdm0] Dormancy Status: 'traffic-channel-dormant'

Table 3-3: WDS Service APIs (Continued)

libqmi command	Function	Example
--wds-create-profile=["(3gpp|3gpp2)[,key=value,...]"]	Create new profile using first available profile index (optional keys: name, apn, pdp-type (IP|PPP|IPV6|IPV4V6), auth (NONE|PAP|CHAP|BOTH), username, password, context-num, no-roaming=yes, disabled=yes)	**$ sudo qmicli -p --device=/dev/cdc-wdm0 --wds-create-profile=3gpp,name=test,apn=testapn,pdp-type=IPV4V6** New profile created: Profile type: '3gpp' Profile index: '3'
--wds-modify-profile=["(3gpp|3gpp2),#,key=value,..."]	Modify existing profile (optional keys: name, apn, pdp-type (IP|PPP|IPV6|IPV4V6), auth (NONE|PAP|CHAP|BOTH), username, password, context-num, no-roaming=yes, disabled=yes)	**$ sudo qmicli -p --device=/dev/cdc-wdm0 --wds-modify-profile="3gpp,3,apn=internet,pdp-type=ipv4"** Profile successfully modified.
--wds-delete-profile=[(3gpp|3gpp2),#]	Delete existing profile	**$ sudo qmicli -p --device=/dev/cdc-wdm0 --wds-delete-profile="3gpp,3"** Profile successfully deleted.
--wds-get-profile-list=[3gpp|3gpp2]	Get profile list	**$ sudo qmicli -p --device=/dev/cdc-wdm0 --wds-get-profile-list=3gpp** Profile list retrieved: [1] 3gpp - APN: '' PDP type: 'ipv4-or-ipv6' PDP context number: '1' Username: '' Password: '' Auth: 'none' No roaming: 'no' APN disabled: 'no' [2] 3gpp - APN: 'internet' 0PDP type: 'ipv4-or-ipv6' PDP context number: '2' Username: '' Password: '' Auth: 'none' No roaming: 'no' APN disabled: 'no'
--wds-get-default-profile-number=[3gpp|3gpp2]	Get default profile number	**$ sudo qmicli -p --device=/dev/cdc-wdm0 --wds-get-default-profile-number=3gpp** Default profile number retrieved: Default profile number: '1'

Table 3-3: WDS Service APIs (Continued)

libqmi command	Function	Example
--wds-set-default-profile-number=[(3gpp|3gpp2),#]	Set default profile number	**$ sudo qmicli -p --device=/dev/cdc-wdm0 --wds-set-default-profile-number=3gpp,1** Default profile number updated
--wds-get-default-settings=[3gpp|3gpp2]	Get default settings	**$ sudo qmicli -p --device=/dev/cdc-wdm0 --wds-get-default-settings=3gpp** Default settings retrieved: APN: '' PDP type: 'ipv4-or-ipv6' Username: '' Password: '' Auth: 'none'
--wds-get-autoconnect-settings	Get autoconnect settings	**$ sudo qmicli -p --device=/dev/cdc-wdm0 --wds-get-autoconnect-settings** Autoconnect settings retrieved: Status: 'disabled'
--wds-set-autoconnect-settings=[(enabled|disabled|paused)[,(roaming-allowed|home-only)]]	Set autoconnect settings (roaming settings optional)	**$ sudo qmicli -p --device=/dev/cdc-wdm0 --wds-set-autoconnect-settings=enabled** Autoconnect settings updated
--wds-reset	Reset the service state	**$ sudo qmicli -p --device=/dev/cdc-wdm0 --wds-reset** [/dev/cdc-wdm0] Successfully performed WDS service reset
--wds-bind-mux-data-port=["key=value,..."]	Bind qmux data port to controller device (allowed keys: mux-id, ep-type (undefined|hsusb|pcie|embedded), ep-iface-number) to be used with `--client-no-release-cid	**$ sudo qmicli -d /dev/cdc-wdm1 --wds-bind-mux-data-port="mux-id=1,ep-iface-number=5" --client-no-release-cid** [/dev/cdc-wdm1] Client ID not released: Service: 'wds' CID: '30'
--wds-set-ip-family=[4|6]	Set IP family	**$ sudo qmicli -p -d /dev/cdc-wdm0 -wds-set-ip-family=4**
--wds-get-channel-rates	Get channel data rates	**$ sudo qmicli -p -d /dev/cdc-wdm0 --wds-get-channel-rates** Channel data rates: Current TX rate: n/a Current RX rate: n/a Max TX rate: 150000000bps Max RX rate: 400000000bps
--wds-get-lte-attach-parameters	Get LTE attach parameters	**$ sudo qmicli -p -d /dev/cdc-wdm0 --wds-get-lte-attach-parameters** LTE attach parameters successfully retrieved: APN: internet IP support type: ipv4v6 OTA attach performed: yes

Table 3-3: WDS Service APIs (Continued)

libqmi command	Function	Example
--wds-get-max-lte-attach-pdn-num	Get the maximum number of LTE attach PDN	**$ sudo qmicli -p -d /dev/cdc-wdm0 --wds-get-max-lte-attach-pdn-num** Maximum number of LTE attach PDN: 56
--wds-get-lte-attach-pdn-list	Get the list of LTE attach PDN	**$ sudo qmicli -p -d /dev/cdc-wdm0 --wds-get-lte-attach-pdn-list** Attach PDN list retrieved: Current list: '2' Pending list: n/a
--wds-set-lte-attach-pdn-list=[#,#,...]	Set the list of LTE attach PDN	**$ sudo qmicli -p -d /dev/cdc-wdm0 --wds-set-lte-attach-pdn-list=2** Attach PDN list update successfully requested

3.4 PBM Service APIs

Table 3-4: PBM Service APIs

libqmi command	Function	Example
--pbm-get-all-capabilities	Get all phonebook capabilities	**$ sudo qmicli -p -d /dev/cdc-wdm0 --pbm-get-all-capabilities** [/dev/cdc-wdm0] Phonebook capabilities: Capability basic information: [gw-primary]: [fdn]: Used records: 0 Maximum records: 3 Maximum number length: 40 Maximum name length: 16 [msisdn]: Used records: 0 Maximum records: 3 Maximum number length: 40 Maximum name length: 12 [mbdn]: Used records: 0 Maximum records: 2 Maximum number length: 40 Maximum name length: 10 [sdn]: Used records: 0 Maximum records: 5 Maximum number length: 40 Maximum name length: 12 [global-phonebook-slot-1]: [adn]: Used records: 0 Maximum records: 250 Maximum number length: 40 Maximum name length: 16 Group capability: [global-phonebook-slot-1]: Maximum groups: 4 Maximum group tag length: 12 Additional number capability: [global-phonebook-slot-1]: Maximum additional numbers: 1 Maximum additional number length: 40 Maximum additional number tag length: 12 Email capability: [global-phonebook-slot-1]: Maximum emails: 1 Maximum email address length: 38 Second name capability: [global-phonebook-slot-1]: Supported: no Alpha string capability: [global-phonebook-slot-1]: Maximum records: 10

Table 3-4: PBM Service APIs (Continued)

libqmi command	Function	Example
		Used records: 0 Maximum string length: 12 Additional number alpha string capability: [global-phonebook-slot-1]: Maximum records: 5 Used records: 0 Maximum string length: 12

3.5 PDC Service APIs

Table 3-5: PDC Service APIs

libqmi command	Function	Example
--pdc-list-configs=[(platform|software)]	List all configs	**$ sudo qmicli -p -d /dev/cdc-wdm0 --pdc-list-configs=platform** Total configurations: 3 Configuration 1: Description: SS-WD-SDX12 Type: platform Size: 12436 Status: Inactive Version: 0x8000E00 ID: BA:9F:2F:D9:E4:C4:10:85:A6:F4:60:E9:CA:BE:BB:AA:68:9B:A4:41 Configuration 2: Description: DSSA-WD-SDX12 Type: platform Size: 12792 Status: Inactive Version: 0x8007900 ID: 60:93:8D:37:F3:66:66:F6:88:56:46:96:15:5B:61:29:70:57:EA:41 Configuration 3: Description: SS-LE-SDX12 Type: platform Size: 9704 Status: Inactive Version: 0x8008000 ID: 0C:86:E4:52:48:C0:24:45:04:0A:4C:F0:CF:BD:9F:3B:C9:8A:4D:3F
--pdc-delete-config=[(platform|software),ConfigId]	Delete config	**$ sudo qmicli -p -d /dev/cdc-wdm0 --pdc-delete-config=software,25** [/dev/cdc-wdm0] Successfully deleted config

Table 3-5: PDC Service APIs (Continued)

libqmi command	Function	Example
--pdc-activate-config=[(platform|software),ConfigId]	Activate config	**$ sudo qmicli -p -d /dev/cdc-wdm0 --pdc-activate-config="software,C5:4E:D7:27:B1:29:78:8B:94:89:12:39:7E:00:4F:4A:83:4F:47:50"** [/dev/cdc-wdm0] Successfully requested config activation
--pdc-deactivate-config=[(platform|software),ConfigId]	Deactivate config	**$ sudo qmicli -p -d /dev/cdc-wdm0 --pdc-deactivate-config="software,C5:4E:D7:27:B1:29:78:8B:94:89:12:39:7E:00:4F:4A:83:4F:47:50"** [/dev/cdc-wdm0] Successfully requested config deactivation

3.6 UIM Service APIs

Table 3-6: UIM Service APIs

libqmi command	Function	Example
--uim-set-pin-protection=[(PIN1|PIN2|UPIN),(disable|enable),(current PIN)]	Set PIN protection	**$ sudo qmicli -p -d /dev/cdc-wdm0 --uim-set-pin-protection=PIN1,enable,8888** [/dev/cdc-wdm0] PIN protection updated
--uim-verify-pin=[(PIN1|PIN2|UPIN),(current PIN)]	Verify PIN	**$ sudo qmicli -p -d /dev/cdc-wdm0 --uim-verify-pin=PIN1,8888** [/dev/cdc-wdm0] PIN verified successfully
--uim-unblock-pin=[(PIN1|PIN2|UPIN),(PUK),(new PIN)]	Unblock PIN	**$ sudo qmicli -p -d /dev/cdc-wdm0 --uim-unblock-pin=PIN1,62217604,8888** [/dev/cdc-wdm0] PIN unblocked successfully
--uim-change-pin=[(PIN1|PIN2|UPIN),(old PIN),(new PIN)]	Change PIN	**$ sudo qmicli -p -d /dev/cdc-wdm0 --uim-change-pin=PIN1,0000,8888** [/dev/cdc-wdm0] PIN changed successfully
--uim-read-transparent=[0xNNNN,0xNNNN,...]	Read a transparent file given the file path	**$ sudo qmicli -p --device=/dev/cdc-wdm0 --uim-read-transparent="0x3F00,0x7FFF,0x6F07"** [/dev/cdc-wdm0] Successfully read information from the UIM: Card result: SW1: '0x90' SW2: '0x00' Read result: 08:49:66:10:21:10:16:83:26

Table 3-6: UIM Service APIs (Continued)

libqmi command	Function	Example
--uim-get-file-attributes=[0xNNNN,0xNNNN,...]	Get the attributes of a given file	**$ sudo qmicli -p --device=/dev/cdc-wdm0 --uim-get-file-attributes="0x3F00,0x7FFF,0x6F07"** [(null)] Successfully got file '/dev/cdc-wdm0' attributes from the UIM: Card result: SW1: '0x90' SW2: '0x00' File attributes: File size: 9 File ID: 28423 File type: transparent Record size: 0 Record count: 0 Read security attributes: (single) pin1 Write security attributes: (single) adm Increase security attributes: (always) (null) Deactivate security attributes: (single) adm Activate security attributes: (single) adm Raw: 62:1F:82:02:41:21:83:02:6F:07:A5:06:C0:01:00:DE:01:20:8A:01:05:8B:03:6F:06:04: 80:02:00:09:88:01:38

Table 3-6: UIM Service APIs (Continued)

libqmi command	Function	Example
--uim-get-card-status	Get card status	**$ sudo qmicli -p -d /dev/cdc-wdm0 --uim-get-card-status** [/dev/cdc-wdm0] Successfully got card status Provisioning applications: Primary GW: slot '1', application '1' Primary 1X: session doesn't exist Secondary GW: session doesn't exist Secondary 1X: session doesn't exist Slot [1]: Card state: 'present' UPIN state: 'not-initialized' UPIN retries: '0' UPUK retries: '0' Application [1]: Application type: 'usim (2)' Application state: 'ready' Application ID: A0:00:00:00:87:10:02:FF:47:F0:01:89:00:00:01:FF Personalization state: 'ready' UPIN replaces PIN1: 'no' 0PIN1 state: 'enabled-verified' PIN1 retries: '3' PUK1 retries: '10' PIN2 state: 'enabled-not-verified' PIN2 retries: '3' PUK2 retries: '10' Application [2]: Application type: 'isim (5)' Application state: 'detected' Application ID: 00A0:00:00:00:87:10:04:FF:49:FF:FF:89:08:15:00:FF Personalization state: 'unknown' UPIN replaces PIN1: 'no' PIN1 state: 'enabled-verified' PIN1 retries: '3' PUK1 retries: '10' PIN2 state: 'not-initialized' PIN2 retries: '0' PUK2 retries: '0'
--uim-sim-power-on=[(slot number)]	Power on SIM card	**$ sudo qmicli -p -d /dev/cdc-wdm0 --uim-sim-power-on=1** [/dev/cdc-wdm0] Successfully performed SIM power on
--uim-sim-power-off=[(slot number)]	Power off SIM card	**$ sudo qmicli -p -d /dev/cdc-wdm0 --uim-sim-power-off=1** [/dev/cdc-wdm0] Successfully performed SIM power off
--uim-change-provisioning-session=["key=value,..."]	Change provisioning session (allowed keys: session-type, activate, slot, aid)	**$ sudo qmicli -p -d /dev/cdc-wdm0 --uim-change-provisioning-session='activate=no,session-type=primary-gw-provisioning'** [/dev/cdc-wdm0] Successfully changed provisioning session

Table 3-6: UIM Service APIs (Continued)

libqmi command	Function	Example
--uim-get-slot-status	Get slot status	**$ sudo qmicli -p -d /dev/cdc-wdm0 --uim-get-slot-status** [/dev/cdc-wdm0] Successfully got slots status [/dev/cdc-wdm0] 2 physical slots found: Physical slot 1: Card status: present Slot status: active Logical slot: 1 ICCID: 89886012137816178620 Protocol: uicc Num apps: 0 Is eUICC: no Physical slot 2: Card status: absent Slot status: inactive
--uim-switch-slot=[(slot number)]	Switch active physical slot	**$ sudo qmicli -p -d /dev/cdc-wdm0 --uim-switch-slot=1** [/dev/cdc-wdm0] Successfully switched slots
--uim-reset	Reset the service state	**$ sudo qmicli -p -d /dev/cdc-wdm0 --uim-reset** [/dev/cdc-wdm0] Successfully performed UIM service reset
--uim-monitor-refresh-all	Watch for REFRESH events for any file	**$ sudo qmicli -p --device=/dev/cdc-wdm0 --uim-monitor-refresh-all**

3.7 SAR Service APIs

Table 3-7: SAR Service APIs

libqmi command	Function	Example
--sar-rf-get-state	Get RF state	**$ sudo qmicli -p --device=/dev/cdc-wdm0 --sar-rf-get-stat** [/dev/cdc-wdm0] Successfully got SAR RF state: 0
--sar-rf-set-state=[(state number)]	Set RF state	**$ sudo qmicli -p --device=/dev/cdc-wdm0 --sar-rf-set-state=0** [/dev/cdc-wdm0] RF state set successfully

3.8 WDA Service APIs

Table 3-8: WDA Service APIs

libqmi command	Function	Example
--wda-set-data-format=["key=value,..."]	Set data format (allowed keys: link-layer-protocol (802-3|raw-ip), ul-protocol (disabled|tlp|qc-ncm|mbim|rndis|qmap|qmapv5), dl-protocol (disabled|tlp|qc-ncm|mbim|rndis|qmap|qmapv5), dl-datagram-max-size, dl-max-datagrams, ep-type (undefined|hsusb|pcie|embedded), ep-iface-number)	**$ sudo qmicli -p -d /dev/cdc-wdm0 --wda-set-data-format="link-layer-protocol=raw-ip,ul-protocol=qmap,dl-protocol=qmap,dl-max-datagrams=32,dl-datagram-max-size=32768,ep-type=hsusb,ep-iface-number=8" --client-no-release-cid** [/dev/cdc-wdm0] Successfully set data format QoS flow header: no Link layer protocol: 'raw-ip' Uplink data aggregation protocol: 'qmap' Downlink data aggregation protocol: 'qmap' NDP signature: '0' Downlink data aggregation max datagrams: '32' Downlink data aggregation max size: '16384' [/dev/cdc-wdm0] Client ID not released: Service: 'wda' CID: '2'
--wda-get-data-format=["key=value,..."]	Get data format (allowed keys: ep-type (undefined|hsusb|pcie|embedded), ep-iface-number); also allows empty key list	**$ sudo qmicli -p -d /dev/cdc-wdm0 --wda-get-data-format="ep-type=hsusb,ep-iface-number=8"** [/dev/cdc-wdm0] Successfully got data format QoS flow header: no Link layer protocol: 'raw-ip' Uplink data aggregation protocol: 'qmap' Downlink data aggregation protocol: 'qmap' NDP signature: '0' Downlink data aggregation max datagrams: '32' Downlink data aggregation max size: '16384'

3.9 LOC Service APIs

Table 3-9: LOC Service APIs

libqmi command	Function	Example
--loc-session-id=[ID]	Session ID for the LOC session	**$ sudo qmicli -p -d /dev/cdc-wdm0 --loc-session-id=0 --loc-stop** [/dev/cdc-wdm0] Successfully stopped location tracking (session id 0)
--loc-start	Start location gathering	**$ sudo qmicli -p -d /dev/cdc-wdm0 --loc-start** [/dev/cdc-wdm0] Successfully started location tracking (session id 0)
--loc-stop	Stop location gathering	**$ sudo qmicli -p -d /dev/cdc-wdm0 --loc-stop** [/dev/cdc-wdm0] Successfully stopped location tracking (session id 0)

Table 3-9: LOC Service APIs (Continued)

libqmi command	Function	Example
--loc-timeout=[SECS]	Maximum time to wait for information in `--loc-get-position-report' and `--loc-get-gnss-sv-info' (default 30s)	**$ sudo qmicli -p -d /dev/cdc-wdm0 --loc-timeout=10 --loc-get-position-report** error: operation failed: timeout
--loc-delete-assistance-data	Delete positioning assistance data	**$ sudo qmicli -p -d /dev/cdc-wdm0 --loc-delete-assistance-data** Successfully deleted assistance data
--loc-get-nmea-types	Get list of enabled NMEA traces	**$ sudo qmicli -p -d /dev/cdc-wdm0 --loc-get-nmea-types** Successfully retrieved NMEA types: gga, rmc, gsv, gsa, vtg, pqxfi
--loc-set-nmea-types=[type1|type2|type3...]	Set list of enabled NMEA traces	**$ sudo qmicli -p -d /dev/cdc-wdm0 --loc-set-nmea-types="gga|rmc|gsv|gsa|vtg"** Successfully set NMEA types
--loc-get-operation-mode	Get operation mode	**$ sudo qmicli -p -d /dev/cdc-wdm0 --loc-get-operation-mode** Successfully retrieved operation mode: standalone
--loc-set-operation-mode=[default|msb|msa|standalone|cellid|wwan]	Set operation mode	**$ sudo qmicli -p -d /dev/cdc-wdm0 --loc-set-operation-mode=default** Successfully set operation mode
--loc-get-engine-lock	Get engine lock status	**$ sudo qmicli -p -d /dev/cdc-wdm0 --loc-get-engine-lock** Successfully retrieved engine lock: mt
--loc-set-engine-lock=[none|mi|mt|all]	Set engine lock status	**$ sudo qmicli -p -d /dev/cdc-wdm0 --loc-set-engine-lock=mt** Successfully set engine lock

3.10 DSD Service APIs

Table 3-10: DSD Service APIs

libqmi command	Function	Example
--dsd-set-apn-type=[(name),(type1|type2|type3...)]	Sets the types associated to a given APN name	**$ sudo qmicli -p -d /dev/cdc-wdm0 --dsd-set-apn-type="test,default"** APN type set

3.11 QOS Service APIs

Table 3-11: QOS Service APIs

libqmi command	Function	Example
--qos-swi-read-data-stats=[profile id]	Get qos stats by profile id	**$ sudo qmicli -d /dev/cdc-wdm0 --qos-swi-read-data-stats=0** [/dev/cdc-wdm0] QoS data stats read APN ID: 0 TX packets: 0 TX packets dropped: 0 RX packets: 0 TX bytes: 0 TX bytes dropped: 0 RX bytes: 0

3.12 Link Management APIs

Table 3-12: Link Management APIs

libqmi command	Function	Example
--link-list=[IFACE]	List links created from a given interface	**$ sudo qmicli -p -d /dev/cdc-wdm0 --link-list=wwan0** [/dev/cdc-wdm0] found 0 links
--link-add=[iface=IFACE,prefix=PREFIX[,mux-id=N]]	Create new network interface link	**$ sudo qmicli -p -d /dev/cdc-wdm0 --link-add="iface=wwan0,prefix=mux01"** [/dev/cdc-wdm0] link successfully added: iface name: qmimux0 Mux-id: 1 **$ sudo qmicli -p -d /dev/cdc-wdm0 --link-add="iface=wwan0,prefix=mux,mux-id=2"** [/dev/cdc-wdm0] link successfully added: iface name: qmimux1 Mux-id: 2
--link-delete=[link-iface=IFACE][,[mux-id=N]]	Delete a given network interface link	**$ sudo qmicli -p -d /dev/cdc-wdm0 --link-delete="link-iface=qmimux1,mux-id=2"** [/dev/cdc-wdm0] link successfully deleted
--link-delete-all=[IFACE]	Delete all network interface links from the given interface	**$ sudo qmicli -p -d /dev/cdc-wdm0 --link-delete-all=wwan0** [/dev/cdc-wdm0] all links successfully deleted

3.13 Qmiwwan APIs

Table 3-13: Qmmiwan Service APIs

libqmi command	Function	Example
-w, --get-wwan-iface	Get the associated WWAN interface name	**$ sudo qmicli -p -d /dev/cdc-wdm0 --get-wwan-iface** Wwan0
-e, --get-expected-data-format	Get the expected data format for the WWAN interface	**$ sudo qmicli -p -d /dev/cdc-wdm0 --get-expected-data-format** Raw-ip
-E, --set-expected-data-format=[802-3|raw-ip|qmap-pass-through]	Set the expected data format in the WWAN interface	**$ sudo qmicli -p -d /dev/cdc-wdm0 --set-expected-data-format=raw-ip** [/dev/cdc-wdm0] expected data format set to: raw-ip

A: Test Cases

A.1 PDN connection

Use libqmi to connect the module and ping 8.8.8.8.

Commands	Notes
$ sudo qmicli -p --device=/dev/cdc-wdm0 --dms-set-operating-mode=online **$ sudo qmicli -p --device=/dev/cdc-wdm0 --wds-start-network="ip-type=4,apn=internet" --client-no-release-cid** [/dev/cdc-wdm0] Network started Packet data handle: '2404856720' [/dev/cdc-wdm0] Client ID not released: Service: 'wds' CID: '31'	
$ sudo qmicli --device=/dev/cdc-wdm0 --set-expected-data-format=raw-ip [/dev/cdc-wdm0] expected data format set to: raw-ip	
$ sudo ip link set dev wwan0 up	
$ sudo udhcpc -q -f -n -i wwan0 udhcpc (v1.22.1) started Sending discover... Sending select for 100.111.17.32... Lease of 100.111.17.32 obtained, lease time 7200 deleting routers DNS= 139.175.1.2 210.241.208.1 adding dns 139.175.1.2 adding dns 210.241.208.1	

$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=52 time=26.4 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=52 time=34.6 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=52 time=32.9 ms
	The ping command uses the default route for the WWAN interface, which udhcpc should have set up.
	If the ping command fails:
	1. Check the default route setting:
	`$ route`
	2. If the default route is missing, add it:
	`$ sudo route add default gw 100.111.17.32 dev wwan0`

A.2 libqmi connection testing with multiple PDNs

Commands	Notes
$ sudo qmicli -p --device=/dev/cdc-wdm0 --dms-set-operating-mode=online **$ sudo qmicli -p --device=/dev/cdc-wdm0 --set-expected-data-format=raw-ip** [/dev/cdc-wdm0] expected data format set to: raw-ip **$ sudo chmod 777 /sys/class/net/wwan0/qmi/add_mux** **$ sudo echo 1 > /sys/class/net/wwan0/qmi/add_mux** **$ sudo echo 2 > /sys/class/net/wwan0/qmi/add_mux** **$ sudo qmicli -p -d /dev/cdc-wdm0 --wda-set-data-format="link-layer-protocol=raw-ip,ul-protocol=qmap,dl-protocol=qmap,dl-max-datagrams=32,dl-datagram-max-size=32768,ep-type=hsusb,ep-iface-number=5" --client-no-release-cid** [/dev/cdc-wdm0] Successfully set data format QoS flow header: no Link layer protocol: 'raw-ip' Uplink data aggregation protocol: 'qmap' Downlink data aggregation protocol: 'qmap' NDP signature: '0' Downlink data aggregation max datagrams: '32' Downlink data aggregation max size: '16384' [/dev/cdc-wdm0] Client ID not released: Service: 'wda' CID: '1' **$ sudo qmicli -p -d /dev/cdc-wdm0 --wds-bind-mux-data-port="mux-id=1,ep-iface-number=8" --client-no-release-cid[/dev/cdc-wdm0] Client ID not released:** Service: 'wds' CID: '33' **$ sudo qmicli -p -d /dev/cdc-wdm0 --wds-bind-mux-data-port="mux-id=2,ep-iface-number=8" --client-no-release-cid** [/dev/cdc-wdm0] Client ID not released: Service: 'wds' CID: '34'	

Commands	Notes
$ sudo qmicli -p --device=/dev/cdc-wdm0 --wds-start-network="ip-type=4,apn=hos" --client-no-release-cid --client-cid=33 [/dev/cdc-wdm0] Network started Packet data handle: '3042382832' [/dev/cdc-wdm0] Client ID not released: Service: 'wds' CID: '33' **$ sudo qmicli -p --device=/dev/cdc-wdm0 --wds-start-network="ip-type=4,apn=internet" --client-no-release-cid --client-cid=34** [/dev/cdc-wdm0] Network started Packet data handle: '3042524624' [/dev/cdc-wdm0] Client ID not released: Service: 'wds' CID: '34'	*Note: In these two commands, "client-cid=33" and "client-cid=34" are not constants. These are the return values of wds-bind-mux-data-port.*
$ sudo ip link set qmimux0 mtu 1500 **$ sudo ip link set qmimux1 mtu 1500** **$ sudo ip link set dev wwan0 up** **$ sudo ip link set dev qmimux0 up** **$ sudo ip link set dev qmimux1 up**	
$ sudo udhcpc -q -f -n -i qmimux0 udhcpc (v1.22.1) started Sending discover... Sending select for 100.67.61.182... Lease of 100.67.61.182 obtained, lease time 7200 deleting routers DNS= 10.156.204.208 10.159.204.208 adding dns 10.156.204.208 adding dns 10.159.204.208 **$ sudo udhcpc -q -f -n -i qmimux1** udhcpc (v1.22.1) started Sending discover... Sending select for 10.160.14.247... Lease of 10.160.14.247 obtained, lease time 7200 deleting routers DNS= 168.95.1.1 168.95.192.1 adding dns 168.95.1.1 adding dns 168.95.192.1	

Commands	Notes
$ ping 8.8.8.8 -I qmimux1 PING 8.8.8.8 (8.8.8.8) from 10.160.14.247 qmimux1: 56(84) bytes of data. 64 bytes from 8.8.8.8: icmp_seq=1 ttl=55 time=44.8 ms 64 bytes from 8.8.8.8: icmp_seq=2 ttl=55 time=35.1 ms 64 bytes from 8.8.8.8: icmp_seq=3 ttl=55 time=53.6 ms **^C** --- 8.8.8.8 ping statistics --- 3 packets transmitted, 3 received, 0% packet loss, time 2002ms rtt min/avg/max/mdev = 35.189/44.561/53.602/7.524 ms **$ ping 8.8.8.8 -I qmimux2**	*Important: Chunghwa Telecom's HOS cannot transmit IP packets to Internet.*

B: References

B.1 Sierra Wireless Documents

Sierra Wireless documents are available from source.sierrawireless.com, or on request (subject to license agreements or NDAs) from your Sierra Wireless representative.

Sierra Wireless Documents on the Source

The following documents are available from source.sierrawireless.com:

[1] EM7590 AT Command Reference (Doc# 41114426)

[2] EM7590 Linux User Guide (Doc# 4134930)