EC200U&EG91xU Series
SSL Application Note

LTE Standard Module Series

Version: 1.2

Date: 2023-09-15

Status: Released

At Quectel, our aim is to provide timely and comprehensive services to our customers. If you require any assistance, please contact our headquarters:

Quectel Wireless Solutions Co., Ltd.
Building 5, Shanghai Business Park Phase III (Area B), No.1016 Tianlin Road, Minhang District, Shanghai 200233, China
Tel: +86 21 5108 6236
Email: info@quectel.com

Or our local offices. 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 us at: support@quectel.com.

Legal Notices

We offer information as a service to you. The provided information is based on your requirements and we make every effort to ensure its quality. You agree that you are responsible for using independent analysis and evaluation in designing intended products, and we provide reference designs for illustrative purposes only. Before using any hardware, software or service guided by this document, please read this notice carefully. Even though we employ commercially reasonable efforts to provide the best possible experience, you hereby acknowledge and agree that this document and related services hereunder are provided to you on an “as available” basis. We may revise or restate this document from time to time at our sole discretion without any prior notice to you.

Use and Disclosure Restrictions

License Agreements

Documents and information provided by us shall be kept confidential, unless specific permission is granted. They shall not be accessed or used for any purpose except as expressly provided herein.

Copyright

Our and third-party products hereunder may contain copyrighted material. Such copyrighted material shall not be copied, reproduced, distributed, merged, published, translated, or modified without prior written consent. We and the third party have exclusive rights over copyrighted material. No license shall be granted or conveyed under any patents, copyrights, trademarks, or service mark rights. To avoid ambiguities, purchasing in any form cannot be deemed as granting a license other than the normal non-exclusive, royalty-free license to use the material. We reserve the right to take legal action for noncompliance with abovementioned requirements, unauthorized use, or other illegal or malicious use of the material.

Trademarks

Except as otherwise set forth herein, nothing in this document shall be construed as conferring any rights to use any trademark, trade name or name, abbreviation, or counterfeit product thereof owned by Quectel or any third party in advertising, publicity, or other aspects.

Third-Party Rights

This document may refer to hardware, software and/or documentation owned by one or more third parties ("third-party materials"). Use of such third-party materials shall be governed by all restrictions and obligations applicable thereto.

We make no warranty or representation, either express or implied, regarding the third-party materials, including but not limited to any implied or statutory, warranties of merchantability or fitness for a particular purpose, quiet enjoyment, system integration, information accuracy, and non-infringement of any third-party intellectual property rights with regard to the licensed technology or use thereof. Nothing herein constitutes a representation or warranty by us to either develop, enhance, modify, distribute, market, sell, offer for sale, or otherwise maintain production of any our products or any other hardware, software, device, tool, information, or product. We moreover disclaim any and all warranties arising from the course of dealing or usage of trade.

Privacy Policy

To implement module functionality, certain device data are uploaded to Quectel's or third-party's servers, including carriers, chipset suppliers or customer-designated servers. Quectel, strictly abiding by the relevant laws and regulations, shall retain, use, disclose or otherwise process relevant data for the purpose of performing the service only or as permitted by applicable laws. Before data interaction with third parties, please be informed of their privacy and data security policy.

Disclaimer

a) We acknowledge no liability for any injury or damage arising from the reliance upon the information.
b) We shall bear no liability resulting from any inaccuracies or omissions, or from the use of the information contained herein.
c) While we have made every effort to ensure that the functions and features under development are free from errors, it is possible that they could contain errors, inaccuracies, and omissions. Unless otherwise provided by valid agreement, we make no warranties of any kind, either implied or express, and exclude all liability for any loss or damage suffered in connection with the use of features and functions under development, to the maximum extent permitted by law, regardless of whether such loss or damage may have been foreseeable.
d) We are not responsible for the accessibility, safety, accuracy, availability, legality, or completeness of information, advertising, commercial offers, products, services, and materials on third-party websites and third-party resources.

Copyright © Quectel Wireless Solutions Co., Ltd. 2023. All rights reserved.

About the Document

Revision History

Version	Date	Author	Description
-	2021-05-19	Kruskal ZHU	Creation of the document
1.0	2021-07-19	Kruskal ZHU	First official release
1.1	2021-08-17	Kruskal ZHU	Added an applicable module series EG915U.
1.2	2023-09-15	Kruskal ZHU	1. Added an applicable module EG912U-GL. 2. Added the following commands: AT+QSSLCFG="session_cache" (Chapter 2.2.1), AT+QSSLCFG="dtlsversion" (Chapter 2.2.1), AT+QSSLRECV=<clientID>,0 (Chapter 2.2.4). 3. Deleted a command: AT+QSSLCFG="cacertex".

Contents

Table Index

1 Introduction

Quectel LTE Standard EC200U series and EG91xU family modules support SSL function.

SSL (Secure Sockets Layer) is a networking protocol designed for securing connections between web clients and web servers over an insecure network, such as the internet.

The SSL function is to ensure the privacy of communication. In some cases, the communication between the server and the client should be encrypted way to prevent data from being eavesdropped, tampered with or forged during the communication process.

1.1. Applicable Modules

Table 1: Applicable Modules

Module Family	Module
-	EC200U Series
EG91xU	EG912U-GL
	EG915U Series

1.2. SSL Version and Cipher Suite

The following SSL versions are applicable.

Table 2: SSL Versions

SSL Versions
SSL 3.0
TLS 1.2
TLS 1.1
TLS 1.0

The following table shows SSL cipher suites supported by EC200U and EG91xU family modules, and all the SSL cipher suites are supported by default. For detailed description of cipher suites, see *RFC 2246-The TLS Protocol Version 1.0*.

Table 3: Supported SSL Cipher Suites

Codes of Cipher Suites	Names of Cipher Suites
0X0035	TLS_RSA_WITH_AES_256_CBC_SHA
0X002F	TLS_RSA_WITH_AES_128_CBC_SHA
0X0005	TLS_RSA_WITH_RC4_128_SHA
0X0004	TLS_RSA_WITH_RC4_128_MD5
0X000A	TLS_RSA_WITH_3DES_EDE_CBC_SHA
0X003D	TLS_RSA_WITH_AES_256_CBC_SHA256
0XC002	TLS_ECDH_ECDSA_WITH_RC4_128_SHA
0XC003	TLS_ECDH_ECDSA_WITH_3DES_EDE_CBC_SHA
0XC004	TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA
0XC005	TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA

0XC007	TLS_ECDHE_ECDSA_WITH_RC4_128_SHA
0XC008	TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA
0XC009	TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
0XC00A	TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
0XC011	TLS_ECDHE_RSA_WITH_RC4_128_SHA
0XC012	TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA
0XC013	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
0XC014	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
0xC00C	TLS_ECDH_RSA_WITH_RC4_128_SHA
0XC00D	TLS_ECDH_RSA_WITH_3DES_EDE_CBC_SHA
0XC00E	TLS_ECDH_RSA_WITH_AES_128_CBC_SHA
0XC00F	TLS_ECDH_RSA_WITH_AES_256_CBC_SHA
0XC023	TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
0xC024	TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
0xC025	TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA256
0xC026	TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA384
0XC027	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
0XC028	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
0xC029	TLS_ECDH_RSA_WITH_AES_128_CBC_SHA256
0XC02A	TLS_ECDH_RSA_WITH_AES_256_CBC_SHA384
0XC02F	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
0XC030	MBEDTLS_TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
0XFFFF	Support all cipher suites above

1.3. The Process of Using SSL Function

Step 1: Configure **<APN>**, **<username>**, **<password>** and other parameters of a PDP context by **AT+QICSGP**. See ***document [1]*** for details.

Step 2: Activate the PDP context by **AT+QIACT**, then the assigned IP address can be queried by **AT+QIACT?**. See ***document [1]*** for details.

Step 3: Configure the SSL version, cipher suite, path of trusted CA certificate authentication mode, the path of the client certificate and private key, etc. for the specified SSL context by **AT+QSSLCFG**.

Step 4: Open an SSL socket to connect a remote server by **AT+QSSLOPEN**. **<SSL_ctxID>** is used to specify SSL context, and **<access_mode>** is used to specify data access mode.

Step 5: After the SSL connection has been established, data will be sent or received via the connection. For details about how to send and receive data in each access mode, please refer to ***Chapter 1.4***.

Step 6: Close an SSL connection by **AT+QSSLCLOSE**.

Step 7: Deactivate the PDP context by **AT+QIDEACT**. See ***document [1]*** for details.

1.4. Description of Data Access Modes

The SSL connection supports the following three kinds of data access modes:

- Buffer access mode
- Direct push mode
- Transparent access mode

When opening an SSL connection via **AT+QSSLOPEN**, the data access mode can be specified by the **<access_mode>**. After the SSL connection has been established, **AT+QISWTMD** can be used to switch the data access mode. See ***document [1]*** for details of **AT+QISWTMD**.

1. In buffer access mode, data can be sent via **AT+QSSLSEND**, and if the module has received data from the Internet, it will report a URC as **+QSSLURC: "recv",<clientID>**. In a such case, data can be retrieved via **AT+QSSLRECV**.

2. In direct push mode, data can be sent via **AT+QSSLSEND**, and if the module has received data from the Internet, the data will be outputted directly via UART/USB modem/USB AT port in the following format of **+QSSLURC: "recv",<clientID>,<currectrecvlength><CR><LF><data>**.

3. In transparent access mode, the corresponding port enters exclusive mode. The data received from COM port will be sent to the Internet directly, and the received data from Internet will be outputted to COM port directly. Use **+++** or DTR (executing **AT&D1** first) to exit transparent access mode. In transparent access mode, if any abnormal SSL disconnection happens, the module will report **NO CARRIER**. See ***document [3]*** for details of **AT&D**.

- **Exit transparent access mode**

To exit transparent access mode, **+++** or DTR (executing **AT&D1** first) can be used. To prevent the **+++** from being misinterpreted as data, follow the following sequence:

1) Do not input any other character within 1 s (at least) before inputting **+++**.
2) Input **+++** within 1 s, and no other characters can be inputted during the time.
3) Do not input any other character within 1 s after **+++** has been inputted.
4) Use **+++** or DTR (executing **AT&D1** first) to make the module exit transparent access mode, and wait until **OK** is returned.

- **Return to transparent access mode**

1) By **AT+QISWTMD**. Specify the **<access_mode>** as 2 when executing this command. If entering transparent access mode successfully, **CONNECT** will be returned.
2) By **ATO**. **ATO** will change the access mode of connection that exits from transparent access mode lately. If entering transparent access mode successfully, **CONNECT** will be returned. If there is no connection entering transparent access mode before, **ATO** will return **NO CARRIER**. See ***document [3]*** for details of **ATO**.

1.5. Validity Period Check of Certificate

To check whether a certificate is in the validity period, the certificate must be parsed, and compare the local time with the “Not before” and “Not after” of the certificate. If the local time is earlier than the time of “Not before” or later than the time of “Not after”, the certificate will be considered expired.

When validity period check of certificate is required (set **<ignore_ltime>** as 0 when executing **AT+QSSLCFG**), in order to avoid failure of certificate validity period check, execute **AT+CCLK** to configure the module time within the validity period of the certificate. See ***document [3]*** for details of **AT+CCLK**.

1.6. Server Name Indication

SNI (Server Name Indication) is desirable for clients to provide Server Host Name information to enhance secure connection with multiple virtual servers based on a single IP address. And this feature is only applicable for TLS protocol.

2 Description of SSL AT Commands

2.1. AT Command Introduction

2.1.1. Definitions

- **<CR>** Carriage return character.
- **<LF>** Line feed character.
- **<...>** Parameter name. Angle brackets do not appear on the command line.
- **[...]** Optional parameter of a command or an optional part of TA information response. Square brackets do not appear on the command line. When an optional parameter is not given in a command, the new value equals to its previous value or the default settings, unless otherwise specified.
- **Underline** Default setting of a parameter.

2.1.2. AT Command Syntax

All command lines must start with **AT** or **at** and end with **<CR>**. Information responses and result codes always start and end with a carriage return character and a line feed character: **<CR><LF><response><CR><LF>**. In tables presenting commands and responses throughout this document, only the commands and responses are presented, and **<CR>** and **<LF>** are deliberately omitted.

Table 4: Types of AT Command

Command Type	Syntax	Description
Test Command	**AT+<cmd>=?**	Test the existence of the corresponding command and return information about the type, value, or range of its parameter.
Read Command	**AT+<cmd>?**	Check the current parameter value of the corresponding command.
Write Command	**AT+<cmd>=<p1>[,<p2>[,<p3>[...]]]**	Set user-definable parameter value.
Execution Command	**AT+<cmd>**	Return a specific information parameter or perform a specific action.

2.1.3. Declaration of AT Command Examples

The AT command examples in this document are provided to help you familiarize with AT commands and learn how to use them. The examples, however, should not be taken as Quectel's recommendation or suggestions about how you should design a program flow or what status you should set the module into. Sometimes multiple examples may be provided for one AT command. However, this does not mean that there exists a correlation among these examples and that they should be executed in a given sequence.

2.2. Description of AT Commands

2.2.1. AT+QSSLCFG Configure Parameters of an SSL Context

The command configures the SSL version, cipher suite, path of trusted CA certificate, authentication mode, the path of the client certificate and private key, etc. for the specified SSL context. These parameters will be used in the handshake procedure.

<SSL_ctxID> is the index of the SSL context. The module supports 6 SSL contexts at most. On the basis of one SSL context, several SSL connections can be established. The settings such as the SSL version and the cipher suite are stored in the SSL context, and they will be applied to the new SSL connections associated with the SSL context.

AT+QSSLCFG Configure Parameters of an SSL Context	
Test Command **AT+QSSLCFG=?**	Response **+QSSLCFG: "sslversion",(**range of supported **<SSL_ctxID>**s**),(**range of supported **<SSL_version>**s**)** **+QSSLCFG: "dtls",(**range of supported **<SSL_ctxID>**s**),(**list of supported **<DTLS_enable>)** **+QSSLCFG: "ciphersuite",(**range of supported **<SSL_ctxID>**s**),(**list of supported **<cipher_suites>**s**)** **+QSSLCFG: "seclevel",(**range of supported **<SSL_ctxID>**s**),(**range of supported **<seclevel>**s**)** **+QSSLCFG: "cacert",(**range of supported **<SSL_ctxID>**s**),<cacertpath>** **+QSSLCFG: "clientcert",(**range of supported **<SSL_ctxID>**s**),<client_cert_path>** **+QSSLCFG: "clientkey",(**range of supported **<SSL_ctxID>**s**),<client_key_path>** **+QSSLCFG: "sni",(**range of supported **<SSL_ctxID>**s**),(**list of supported **<SNI>**s**)** **+QSSLCFG: "ignorelocaltime",(**range of supported **<SSL_ctxID>**s**),(**list of supported **<ignore_ltime>**s**)** **+QSSLCFG: "negotiatetime",(**range of supported **<SSL_**

	ctxID>s),(range of supported **<negotiate_time>s)** **+QSSLCFG: "ignoreinvalidcertsign",(**range of supported **<SSL_ctxID>s),(**list of supported **<ignore_invalid_certsign>s)** **+QSSLCFG: "ignorecertitem",(**range of supported **<SSL_ctxID>s),(**range of supported **<ignore_check_item>s)** **+QSSLCFG: "ignoremulticertchainverify",(**range of supported **<SSL_ctxID>s),(**list of supported **<ignore_multicertchain_verify>s)** **+QSSLCFG: "session_cache",(**range of supported **<SSL_ctxID>s),(**list of supported **<session_cache_enable>s)** **+QSSLCFG: "dtlsversion",(**range of supported **<SSL_ctxID>s),(**list of supported **<DTLS_version>s)** **OK**
Write Command Configure the SSL version for the specified SSL context: **AT+QSSLCFG="sslversion",<SSL_ctxID>[,<SSL_version>]**	Response If the optional parameter is omitted, query the SSL version for the specified SSL context: **+QSSLCFG: "sslversion",<SSL_ctxID>,<SSL_version>** **OK** If the optional parameter is specified, set the SSL version for the specified SSL context: **OK** Or **ERROR**
Write Command Configure the DTLS function for the specified SSL context: **AT+QSSLCFG="dtls",<SSL_ctxID>[,<DTLS_enable>]**	Response If the optional parameter is omitted, query whether the DTLS function is enabled for the specified SSL context: **+QSSLCFG: "dtls",<SSL_ctxID>,<DTLS_enable>** **OK** If the optional parameter is specified, enable/disable the DTLS function for the specified SSL context: **OK** Or **ERROR**
Write Command Configure the SSL cipher suites for the specified SSL context: **AT+QSSLCFG="ciphersuite",<SSL_ctxID>[,<cipher_suites>]**	Response If the optional parameter is omitted, query the SSL cipher suites for the specified SSL context: **+QSSLCFG: "ciphersuite",<SSL_ctxID>,<cipher_suites>**

	OK If the optional parameter is specified, set the SSL cipher suite for the specified SSL context: **OK** Or **ERROR**
Write Command Configure the authentication mode for the specified SSL context: **AT+QSSLCFG="seclevel",<SSL_ctxID>[,<seclevel>]**	Response If the optional parameter is omitted, query the authentication mode for the specified SSL context: **+QSSLCFG: "seclevel",<SSL_ctxID>,<seclevel>** **OK** If the optional parameter is specified, set the authentication mode for the specified SSL context: **OK** Or **ERROR**
Write Command Configure the path of trusted CA certificate for the specified SSL context: **AT+QSSLCFG="cacert",<SSL_ctxID>[,<cacertpath>]**	Response If the optional parameter is omitted, query the path of configured trusted CA certificate for the specified SSL context: **+QSSLCFG: "cacert",<SSL_ctxID>,<cacertpath>** **OK** If the optional parameter is specified, set the path of trusted CA certificate for the specified SSL context: **OK** Or **ERROR**
Write Command Configure the path of client certificate for the specified SSL context: **AT+QSSLCFG="clientcert",<SSL_ctxID>[,<client_cert_path>]**	Response If the optional parameter is omitted, query the path of client certificate for the specified SSL context: **+QSSLCFG: "clientcert",<SSL_ctxID>,<client_cert_path>** **OK** If the optional parameter is specified, set the path of client certificate for the specified SSL context: **OK** Or **ERROR**
Write Command	Response

Configure the path of client private key for the specified SSL context: **AT+QSSLCFG="clientkey",<SSL_ctxID>[,<client_key_path>]**	If the optional parameter is omitted, query the path of client private key for the specified SSL context: **+QSSLCFG: "clientkey",<SSL_ctxID>,<client_key_path>** **OK** If the optional parameter is specified, set the path of client private key for the specified SSL context: **OK** Or **ERROR**
Write Command Configure Server Name Indication feature for the specified SSL context: **AT+QSSLCFG="sni",<SSL_ctxID>[,<SNI>]**	Response If the optional parameter is omitted, query whether the Server Name Indication feature is enabled for the specified SSL context: **+QSSLCFG: "sni",<SSL_ctxID>,<SNI>** **OK** If the optional parameter is specified, disable/enable Server Name Indication feature for the specified SSL context: **OK** Or **ERROR**
Write Command Configure whether to ignore certificate validity period check for the specified SSL context: **AT+QSSLCFG="ignorelocaltime",<SSL_ctxID>[,<ignore_ltime>]**	Response If the optional parameter is omitted, query whether the certificate validity period check is ignored for the specified SSL context: **+QSSLCFG: "ignorelocaltime",<SSL_ctxID>,<ignore_ltime>** **OK** If the optional parameter is specified, set whether or not to ignore certificate validity check for the specified SSL context: **OK** Or **ERROR**
Write Command Configure the maximum timeout in SSL negotiation stage for the specified SSL context: **AT+QSSLCFG="negotiatetime",<SSL_ctxID>[,<negotiate_time>]**	Response If the optional parameter is omitted, query the maximum timeout in SSL negotiation stage for the specified SSL context: **+QSSLCFG: "negotiatetime",<SSL_ctxID>,<negotiate_time>**

	OK If the optional parameter is specified, set the maximum timeout in SSL negotiation stage for the specified SSL context: **OK** Or **ERROR**
Write Command Configure whether to ignore the invalid certificate signature for the specified SSL context: **AT+QSSLCFG="ignoreinvalidcertsign",<SSL_ctxID>[,<ignore_invalid_certsign>]**	Response If the optional parameter is omitted, query whether the invalid certificate signature is ignored for the specified SSL context: **+QSSLCFG: "ignoreinvalidcertsign",<SSL_ctxID>,<ignore_invalid_certsign>** **OK** If the optional parameter is specified, set whether or not to ignore the invalid certificate signature for the specified SSL context: **OK** Or **ERROR**
Write Command Configure whether the client ignores one or more checks specified in the certificate sent by the server for the specified SSL context: **AT+QSSLCFG="ignorecertitem",<SSL_ctxID>[,<ignore_check_item>]**	Response If the optional parameter is omitted, query whether the client ignores one or more checks specified in the certificate sent by the server for the specified SSL context: **+QSSLCFG: "ignorecertitem",<SSL_ctxID>,<ignore_check_item>** **OK** If the optional parameter is specified, configure whether the client ignores one or more checks specified in the certificate sent by the server for the specified SSL context: **OK** Or **ERROR**
Write Command Configure whether to ignore multiple level certificate chain verification for the specified SSL context: **AT+QSSLCFG="ignoremulticertchainverify",<SSL_ctxID>[,<ignore_multic**	Response If the optional parameter is omitted, query whether the multiple level certificate chain verification is ignored for the specified SSL context: **+QSSLCFG: "ignoremulticertchainverify",<SSL_ctxID>,<ignore_multicertchain_verify>**

ertchain_verify>]	**OK** If the optional parameter is specified, set whether or not to ignore multiple level certificate chain verification for the specified SSL context: **OK** Or **ERROR**
Write Command Enable or disable the specified SSL context session caching feature: **AT+QSSLCFG="session_cache",<SSL_ctxID>[,<session_cache_enable>]**	Response If the optional parameter is omitted, query whether the specified SSL context enables SSL session caching: **+QSSLCFG: "session_cache",<SSL_ctxID>,<session_cache_enable>** **OK** If the optional parameter is specified, enable or disable the specified SSL context session caching feature: **OK** Or **ERROR**
Write Command Configure the DTLS protocol version that specifies the SSL context: **AT+QSSLCFG="dtlsversion",<SSL_ctxID>[,<DTLS_version>]**	Response If the optional parameter is omitted, query the DTLS protocol version that specifies the SSL context: **+QSSLCFG: "dtlsversion",<SSL_ctxID>,<DTLS_version>** **OK** If the optional parameter is specified, configure the DTLS protocol version that specifies the SSL context: **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	The command takes effect immediately. The configurations are not saved.

Parameter

<SSL_ctxID>	Integer type. SSL context ID. Range: 0–5.
<SSL_version>	Integer type. SSL version. 0 SSL 3.0

	1 TLS 1.0 2 TLS 1.1 3 TLS 1.2 4 All
<DTLS_enable>	Integer type. Enable/disable the DTLS function. 0 Disable 1 Enable
<cipher_suites>	Numeric type in HEX format. SSL cipher suites. 0X0035 TLS_RSA_WITH_AES_256_CBC_SHA 0X002F TLS_RSA_WITH_AES_128_CBC_SHA 0X0005 TLS_RSA_WITH_RC4_128_SHA 0X0004 TLS_RSA_WITH_RC4_128_MD5 0X000A TLS_RSA_WITH_3DES_EDE_CBC_SHA 0X003D TLS_RSA_WITH_AES_256_CBC_SHA256 0XC002 TLS_ECDH_ECDSA_WITH_RC4_128_SHA 0XC003 TLS_ECDH_ECDSA_WITH_3DES_EDE_CBC_SHA 0XC004 TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA 0XC005 TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA 0XC007 TLS_ECDHE_ECDSA_WITH_RC4_128_SHA 0XC008 TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA 0XC009 TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA 0XC00A TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA 0XC011 TLS_ECDHE_RSA_WITH_RC4_128_SHA 0XC012 TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA 0XC013 TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA 0XC014 TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA 0xC00C TLS_ECDH_RSA_WITH_RC4_128_SHA 0XC00D TLS_ECDH_RSA_WITH_3DES_EDE_CBC_SHA 0XC00E TLS_ECDH_RSA_WITH_AES_128_CBC_SHA 0XC00F TLS_ECDH_RSA_WITH_AES_256_CBC_SHA 0XC023 TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 0xC024 TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 0xC025 TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA256 0xC026 TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA384 0XC027 TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 0XC028 TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 0xC029 TLS_ECDH_RSA_WITH_AES_128_CBC_SHA256 0XC02A TLS_ECDH_RSA_WITH_AES_256_CBC_SHA384 0XC02F TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 0xC030 MBEDTLS_TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 0XFFFF Support all cipher suites
<seclevel>	Integer type. The authentication mode. 0 No authentication 1 Perform server authentication

	2 Perform server and client authentication if requested by the remote server
<cacertpath>	String type. The path of the trusted CA certificate.
<client_cert_path>	String type. The path of the client certificate.
<client_key_path>	String type. The path of the client private key.
<SNI>	Integer type. Disable/enable Server Name Indication feature. 0 Disable 1 Enable
<ignore_ltime>	Integer type. Whether or not to ignore validity period check of certificate. 0 Not to ignore 1 Ignore
<negotiate_time>	Integer type. Maximum timeout used in SSL negotiation stage. Range: 10–300. Default value: 300. Unit: second.
<ignore_invalid_certsign>	Integer type. Indicates whether or not to ignore the invalid certificate signature. 0 Not to ignore 1 Ignore
<ignore_check_item>	Integer type. Whether the client ignores one or more checks specified in the certificate sent by the server. The parameter applies an accumulative value if the client ignores more checks. 0 Not to ignore any check item 1 Ignore that the certificate validity has expired 4 Ignore the certificate common name does not match the expected common name 8 Ignore that the certificate is not correctly signed by the trusted CA 256 Ignore other reasons (The reason used to verify the callback) 2048 Ignore that usage does not match the keyUsage extension 4096 Ignore that usage does not match the extendedKeyUsage extension 8192 Ignore that usage does not match the nsCertType extension 32768 Ignore that the certificate signed with unacceptable public key algorithm (such as RSA, ECDSA) 65536 Ignore that the certificate signed with an unacceptable key 1048575 Ignore all check items, that is, not to check the certificate
<ignore_multicertchain_verify>	Integer type. Indicates whether or not to ignore the multiple level certificate chains verification. 0 Not to ignore 1 Ignore
<session_cache_enable>	Integer type. Enable or disable the SSL context session caching feature. 0 Disable 1 Enable
<DTLS_version>	Integer type. DTLS version. 0 DTLS 1.0 1 DTLS 1.2

2.2.2. AT+QSSLOPEN Open an SSL Socket to Connect a Remote Server

The command sets up an SSL connection. During the negotiation between the module and the Internet, parameters configured by **AT+QSSLCFG** will be used in the handshake procedure. After shaking hands with the Internet successfully, the module can send or receive data via this SSL connection. Also, the module can set up several SSL connections based on one SSL context.

According to steps mentioned in ***Chapter 1.3***, execute **AT+QIACT** first to activate the PDP context and then execute **AT+QSSLOPEN**. It is suggested to wait for a specific period of time (refer to the Maximum Response Time below) for **+QSSLOPEN: <clientID>,<err>** URC to be outputted. If the URC response cannot be received during the time, **AT+QSSLCLOSE** can be used to close the SSL connection.

AT+QSSLOPEN Open an SSL Socket to Connect a Remote Server	
Test Command **AT+QSSLOPEN=?**	Response **+QSSLOPEN: (**range of supported **<contextID>**s**),(**range of supported **<SSL_ctxID>**s**),(**range of supported **<clientID>**s**), <serveraddr>,<server_port>[,(**range of supported **<access_mode>**s**)]** **OK**
Write Command **AT+QSSLOPEN=<contextID>,<SSL_ctxID>,<clientID>,<serveraddr>,<server_port>[,<access_mode>]**	Response If the **<access_mode>**=2 and the SSL connection is successfully set up: **CONNECT** If there is any error: **ERROR** If the **<access_mode>**=0/1: **OK** **+QSSLOPEN: <clientID>,<err>** **<err>** is 0 when SSL socket is opened successfully, and **<err>** is not 0 when opening SSL socket fails. If there is any error: **ERROR**
Maximum Response Time	Maximum network response time of 150 s, plus configured time of **<negotiate_time>**.
Characteristics	The command takes effect immediately. The configurations are not saved.

Parameter

<contextID>	Integer type. PDP context ID. Range: 1–7.
<SSL_ctxID>	Integer type. SSL context ID. Range: 0–5.
<clientID>	Integer type. Socket index. Range: 0–11.
<serveraddr>	String type. The address of remote server.
<server_port>	Integer type. The listening port of remote server.
<access_mode>	Integer type. The data access mode of SSL connection. 0 Buffer access mode 1 Direct push mode 2 Transparent access mode
<err>	Integer type. The error code of the operation. See ***Chapter 5***.
<negotiate_time>	Integer type. Maximum timeout in SSL negotiation stage. Range: 10–300. Default: 300. Unit: second.

2.2.3. AT+QSSLSEND Send Data via SSL Connection

After the connection is established, the module can send data through the SSL connection.

AT+QSSLSEND Send Data via SSL Connection	
Test Command **AT+QSSLSEND=?**	Response **+QSSLSEND: (**range of supported **<clientID>**s**)[,(**range of supported **<sendlen>**s**)]** **OK**
Write Command Send variable-length data **AT+QSSLSEND=<clientID>**	Response **>** After the above response, input the data to be sent. Tap **CTRL+Z** to send, and tap **ESC** to cancel the operation. If the connection has been established and the data have been sent successfully: **SEND OK** If connection has been established but buffer is full: **SEND FAIL** If connection cannot be established, abnormally closed, or the parameter is incorrect: **ERROR**
Write Command Send fixed-length data **AT+QSSLSEND=<clientID>,<sendlen>**	Response **>** After the above response, input the data until the data length

	equals **<sendlen>**. If connection has been established and the data have been sent successfully: **SEND OK** If connection has been established but buffer is full: **SEND FAIL** If connection cannot be established, abnormally closed, or the parameter is incorrect: **ERROR**
Maximum Response Time	300 ms
Characteristics	The command takes effect immediately. The configurations are not saved.

Parameter

<clientID>	Integer type. Socket index. Range: 0–11.
<sendlen>	Integer type. The length of data to be sent. Range: 1–1460. Unit: byte.

NOTE

The data to be sent includes fixed-length data and variable-length data, and their maximum length is 1460 bytes.

2.2.4. AT+QSSLRECV Receive Data via SSL Connection

When the data access mode of an SSL connection is buffer access mode, the module will report URC as **+QSSLURC: "recv",<clientID>** when it receives data from the Internet. You can read the data from buffer by **AT+QSSLRECV**.

AT+QSSLRECV Receive Data via SSL Connection	
Test Command **AT+QSSLRECV=?**	Response **+QSSLRECV: (**range of supported **<clientID>**s**),(**range of supported **<readlen>**s**)** **OK**
Write Command **AT+QSSLRECV=<clientID>,<readlen>**	Response If the specified connection has received data: **+QSSLRECV: <have_readlen><CR><LF><data>**

	OK If the buffer is empty: **+QSSLRECV: 0** **OK** If connection cannot be established, abnormally closed, or the parameter is incorrect: **ERROR**
Write Command When **<read_length>** is 0, query the length of the read data: **AT+QSSLRECV=<clientID>,0**	Response If the specified connection exists: **+QSSLRECV: <total_receive_length>,<have_read_length>,<unread_length>** **OK** If there is any error: **ERROR**
Maximum Response Time	300 ms
Characteristics	The command takes effect immediately. The configurations are not saved.

Parameter

<clientID>	Integer type. Socket index. Range: 0–11.
<readlen>	Integer type. The length of data to be read. Range: 1–1500. Unit: byte.
<have_readlen>	Integer type. The actual length of data read by **AT+QSSLRECV**. Unit: byte.
<data>	The actual data read. Unit: byte.
<total_receive_length>	Integer type. The total length of the received data. Unit: byte.
<have_read_length>	Integer type. The length of the read data. Unit: byte.
<unread_length>	Integer type. The length of the unread data. Unit: byte.

2.2.5. AT+QSSLCLOSE Close an SSL Connection

The command closes an SSL connection. If all the SSL connections based on the same SSL context are closed, the module will release the SSL context.

AT+QSSLCLOSE Close an SSL Connection	
Test Command **AT+QSSLCLOSE=?**	Response **+QSSLCLOSE: (**range of supported **<clientID>**s**),(**range of supported **<close_timeout>**s**)**

	OK
Write Command **AT+QSSLCLOSE=<clientID>[,<close_timeout>]**	Response If the SSL connection is successfully closed: **OK** If there is any error: **ERROR**
Maximum Response Time	Determined by **<close_timeout>**
Characteristics	The command takes effect immediately. The configurations are not saved.

Parameter

<clientID>	Integer type. Socket index. Range: 0–11.
<close_timeout>	Integer type. The timeout of executing **AT+QSSLCLOSE**. Range: 0–65535. Default: 10. Unit: second. 0 means immediate execution of the command.

2.2.6. AT+QSSLSTATE Query the State of SSL Connection

The command queries the socket connection status, and can only query the SSL connection status.

AT+QSSLSTATE Query the State of SSL Connection	
Test Command **AT+QSSLSTATE=?**	Response **OK**
Write Command **AT+QSSLSTATE=<clientID>**	Response **+QSSLSTATE: <clientID>,"SSLClient",<IP_address>,<remote_port>,<local_port>,<socket_state>,<PDP_ctxID>,<serverID>,<access_mode>,<AT_port>,<SSL_ctxID>** **OK**
Execution Command **AT+QSSLSTATE**	Response List of **(+QSSLSTATE: <clientID>,"SSLClient",<IP_address>,<remote_port>,<local_port>,<socket_state>,<PDP_ctxID>,<serverID>,<access_mode>,<AT_port>,<SSL_ctxID>)** **OK**
Maximum Response Time	300 ms
Characteristics	/

Parameter

<contextID>	Integer type. PDP context ID. Range: 1–7.
<clientID>	Integer type. Socket index. Range: 0–11.
<IP_address>	String type. The address of remote server.
<remote_port>	Integer type. The port of remote server. Range: 0–65535.
<local_port>	Integer type. The local port. Range: 0–65535.
<socket_state>	Integer type. The state of SSL connection. 0 "Initial" Connection has not been established 1 "Opening" Client is connecting 2 "Connected" Client connection has been established 4 "Closing" Connection is closing
<serverID>	Integer type. Reserved.
<access_mode>	Integer type. The access mode of SSL connection. 0 Buffer access mode 1 Direct push mode 2 Transparent access mode
<AT_port>	String type. COM port.
<SSL_ctxID>	Integer type. SSL context ID. Range: 0–5.

2.3. Description of URCs

2.3.1. +QSSLURC: "recv" URC Indicating Incoming Data

The URC notifies the host of received data which comes from the server.

+QSSLURC: "recv" URC Indicating Incoming Data	
+QSSLURC: "recv",<clientID>	The URC of SSL data incoming in buffer access mode. The data can be received by **AT+QSSLRECV**.
+QSSLURC: "recv",<clientID>,<current_recvlength><CR><LF><data>	The URC of SSL data incoming in direct push mode.

Parameter

<clientID>	Integer type. Socket index. Range: 0–11.
<current_recvlength>	Integer type. The length of actual received data. Unit: byte.
<data>	The actual data read. Unit: byte.

2.3.2. +QSSLURC: "closed" Notify Abnormal Disconnection

The URC notifies that the connection has been disconnected. Disconnection can be caused by many reasons. For example, the Internet closes the connection or the state of GPRS PDP is deactivated, and the SSL connection state based on the specified socket may be "closing". In such case, **AT+QSSLCLOSE=<clientID>** must be executed to change the SSL connection state to "initial".

+QSSLURC: "closed" Notify Abnormal Disconnection	
+QSSLURC: "closed",<clientID>	The SSL connection based on the specified socket is closed.

Parameter

<clientID>	Integer type. Socket index. Range: 0–11.

3 Examples

3.1. Configure and Activate a PDP Context

3.1.1. Configure a PDP Context

```
AT+QICSGP=1,1,"UNINET","","",1          //Configure PDP context as 1. APN is "UNINET" for China Unicom.

OK
```

3.1.2. Activate a PDP Context

```
AT+QIACT=1                              //Activate PDP context as 1.
OK                                      //Activate successfully.
AT+QIACT?                               //Query the state of PDP context.
+QIACT: 1,1,1,"10.7.157.1"

OK
```

3.1.3. Deactivate a PDP Context

```
AT+QIDEACT=1                            //Deactivate PDP context 1.
OK                                      //Deactivate successfully.
```

3.2. Configure an SSL Context

```
AT+QSSLCFG="sslversion",1,1             //Set SSL context ID and SSL version as 1.
OK
AT+QSSLCFG="ciphersuite",1,0X0035       //Set SSL context ID as 1 and SSL cipher suites as TLS_RSA_WITH_AES_256_CBC_SHA.

OK
AT+QSSLCFG="seclevel",1,1               //Set SSL context ID as 1 and authentication mode as server authentication.
```

```
OK
AT+QSSLCFG="cacert",1,"UFS:cacert.pem" //Set SSL context ID as 1 and the path of the trusted CA
                                       certificate as UFS:cacert.pem.
OK
```

3.3. SSL Client in Buffer Access Mode

3.3.1. Set up an SSL Connection and Enter Buffer Access Mode

```
AT+QSSLOPEN=1,1,4,"220.180.239.212",8010,0
OK

+QSSLOPEN: 4,0                    //Set up an SSL connection successfully.
AT+QSSLSTATE                      //Query the state of all SSL connections.
+QSSLSTATE: 4,"SSLClient","220.180.239.212",8010,65344,2,1,4,0,"usbmodem",1

OK
```

3.3.2. Send Data in Buffer Access Mode

3.3.2.1. Send Variable-length Data

```
AT+QSSLSEND=4                     //Send variable-length data.
>
Test data from SSL
<CTRL+Z>
SEND OK
```

3.3.2.2. Send Fixed-length Data

```
AT+QSSLSEND=4,18                  //Send fixed-length data with the length of 18 bytes.
>
Test data from SSL
SEND OK
```

3.3.3. Receive Data in Buffer Access Mode

```
+QSSLURC: "recv",4                  //The socket 4 (<clientID> = 4) has received data.

AT+QSSLRECV=4,1500                  //Read data. The length of data to be read is 1500 bytes.
+QSSLRECV: 18                       //The length of actual retrieved data is 18 bytes.
Test data from SSL

OK
AT+QSSLRECV=4,1500
+QSSLRECV: 0                        //No data in buffer.

OK
```

3.3.4. Close an SSL Connection

```
AT+QSSLCLOSE=4                      //Close an SSL connection (<clientID> = 4). Depending on the
                                      network, the maximum response time is 10 s.
OK
```

3.4. SSL Client in Direct Push Mode

3.4.1. Set up an SSL Connection and Enter Direct Push Mode

```
AT+QSSLOPEN=1,1,4,"220.180.239.212",8011,1
OK

+QSSLOPEN: 4,0                      //Set up SSL connection successfully.
AT+QSSLSTATE                        //Query the state of all SSL connections.
+QSSLSTATE: 4,"SSLClient","220.180.239.212",8011,65047,2,1,4,1,"usbmodem",1

OK
```

3.4.2. Send Data in Direct Push Mode

```
AT+QSSLSEND=4                       //Send variable-length data.
>
Test data from SSL
<CTRL+Z>
SEND OK
```

```
AT+QSSLSEND=4,18                    //Send fixed-length data and the data length is 18 bytes.
>
Test data from SSL
SEND OK
```

3.4.3. Receive Data in Direct Push Mode

```
+QSSLURC: "recv",4,18
Test data from SSL
```

3.4.4. Close an SSL Connection

```
AT+QSSLCLOSE=4                      //Close an SSL connection (<clientID> = 4). Depending on the
                                      network, the maximum response time is 10 s.
OK
```

3.5. SSL Client in Transparent Access Mode

3.5.1. Set up an SSL Connection and Send Data in Transparent Access Mode

```
AT+QSSLOPEN= 1,1,4,"220.180.239.212",8011,2    //Set up an SSL connection.
CONNECT                                        //Enter transparent access mode.
                                    //Client is sending data from COM port to the Internet directly (The data
                                      is not visible in the example).
OK                                  //Use +++ or DTR (executing AT&D1 first) to exit transparent access
                                      mode. The NO CARRIER result code indicates that the server has
                                      stopped the SSL connection.
```

3.5.2. Set up an SSL Connection and Receive Data in Transparent Access Mode

```
AT+QSSLOPEN= 1,1,4,"220.180.239.212",8011,2    //Set up an SSL connection.
CONNECT                                        //Enter transparent access mode.
<Received data>                     //Client is reading the data.
OK                                  //Use +++ or DTR (executing AT&D1 first) to exit transparent access
                                      mode. The NO CARRIER result code indicates that the server has
                                      stopped the SSL connection.
```

3.5.3. Close an SSL Connection

```
AT+QSSLCLOSE=4          //Close an SSL connection (<clientID> = 4). Depending on the network, the maximum response time is 10 s.

OK
```

4 Check for Failure in SSL Connection

Please find out reasons for the failure in opening an SSL connection as follows:

1. Query the status of the specified PDP context by **AT+QIACT?** to check whether the specified PDP context has been activated.

2. Since an invalid DNS server address cannot convert domain name to IP address, if the address of remote server is a domain name, please check whether the address of DNS server is valid by **AT+QIDNSCFG=<contextID>**. See ***document [1]*** for details of **AT+QIDNSCFG**.

3. Check the SSL configuration by **AT+QSSLCFG**, especially the SSL version and cipher suite, to make ensure that they are supported on server side. If **<seclevel>** has been configured as 1 or 2, then the trusted CA certificate has to be uploaded to the module with **AT+QFUPL**. If the server side has configured “SSLVerifyClient required”, then the client certificate and client private key have to be uploaded to the module with **AT+QFUPL**. For details about validity period check of certificate, please see ***Chapter 1.5***. See ***document [2]*** for details of **AT+QFUPL**.

5 Error Codes

If an **ERROR** or URC error code is returned after executing SSL AT commands, the details of error can be queried by **AT+QIGETERROR**. Please note that **AT+QIGETERROR** just returns error code of the latest SSL AT command. See ***document [1]*** for details of **AT+QIGETERROR**.

Table 5: Error Codes

<err>	Description
0	Operation successful
550	Unknown error
551	Operation blocked
552	Invalid parameter
553	Memory not enough
554	Create socket failed
555	Operation not supported
556	Socket bind failed
557	Socket listen failed
558	Socket write failed
559	Socket read failed
560	Socket accept failed
561	Open PDP context failed
562	Close PDP context failed
563	Socket identity has been used
564	DNS busy

565	DNS parse failed
566	Socket connection failed
567	Socket has been closed
568	Operation busy
569	Operation timeout
570	PDP context break down
571	Cancel send
572	Operation not allowed
573	APN not configured
574	Port busy
579	SSL handshake fail

6 Appendix References

Table 6: Related Documents

Document Name
[1] Quectel_EC200U&EG91xU_Series_TCP (IP)_Application_Note
[2] Quectel_EC200U&EG91xU_Series_FILE_Application_Note
[3] Quectel_EC200U&EG91xU_Series_AT_Commands_Manual

Table 7: Terms and Abbreviations

Abbreviation	Description
APN	Access Point Name
CA	Certificate Authority
CR	Carriage Return
CMUX	Connection Multiplexing
CN	Common Name
DNS	Domain Name Server
DTR	Data Terminal Ready
DTLS	Datagram Transport Layer Security
GPRS	General Packet Radio Service
LF	Line Feed
PDP	Packet Data Protocol
PK	Public Key
PSK	Pre-Shared Key

SNI	Server Name Indication
SSL	Security Socket Layer
TCP/IP	Transmission Control Protocol/Internet Protocol
TLS	Transport Layer Security
UART	Universal Asynchronous Receiver/Transmitter
UFS	Universal Flash Storage
URC	Unsolicited Result Code
USB	Universal Serial Bus