BG95&BG77&BG600L Series
SSL Application Note

LPWA Module Series

Version: 1.1

Date: 2021-10-22

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.

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.

About the Document

Revision History

Version	Date	Author	Description
1.0	2019-10-12	Terrence YANG	Initial
1.1	2021-10-22	Terrence YANG	1. Added an applicable module BG600L-M3. 2. Updated the supported SSL cipher suites (Table 3 and Chapter 2.3.1). 3. Added AT+QSSLCFG="renegotiation" (Chapter 2.3.1). 4. Added AT+QSSLRECV=<clientID>,0 (Chapter 2.3.4). 5. Added AT+QSSLCRYPT (Chapter 2.3.7). 6. Added an example about setting up a DTLS connection based on PSK encryption (Chapter 3.6).

Contents

About the Document **3**
Contents **4**
Table Index **6**

1 Introduction **7**
1.1. Applicable Modules 7
1.2. SSL Versions and Cipher Suites 8
1.3. Using SSL Function 10
1.4. Description of Data Access Modes 10
1.5. Certificate Validity Check 11
1.6. Server Name Indication 12

2 Description of SSL AT Commands **13**
2.1. AT Command Introduction 13
2.1.1. Definitions 13
2.1.2. AT Command Syntax 13
2.2. Declaration of AT Command Examples 14
2.3. Description of SSL AT Commands 14
2.3.1. AT+QSSLCFG Configure Parameters of an SSL Context 14
2.3.2. AT+QSSLOPEN Open an SSL Socket to Connect a Remote Server 21
2.3.3. AT+QSSLSEND Send Data via SSL Connection 22
2.3.4. AT+QSSLRECV Retrieve Data Through SSL Connection 24
2.3.5. AT+QSSLCLOSE Close an SSL Connection 25
2.3.6. AT+QSSLSTATE Query the State of SSL Connections 25
2.3.7. AT+QSSLCRYPT Encrypt/Decrypt Data with a Specified Algorithm 27
2.4. Description of URCs 28
2.4.1. +QSSLURC: "recv" Notify Received Data 28
2.4.2. +QSSLURC: "closed" Notify Abnormal Disconnection 28

3 Examples **29**
3.1. Configure and Activate a PDP Context 29
3.1.1. Configure a PDP Context 29
3.1.2. Activate a PDP Context 29
3.1.3. Deactivate a PDP Context 29
3.2. Configure an SSL Context 29
3.3. SSL Client in Buffer Access Mode 30
3.3.1. Set up an SSL Connection and Enter Buffer Access Mode 30
3.3.2. Send Data in Buffer Access Mode 30
3.3.3. Retrieve Data in Buffer Access Mode 30
3.3.4. Close an SSL Connection 31
3.4. SSL Client in Direct Push Mode 31
3.4.1. Set up an SSL Connection and Enter Direct Push Mode 31
3.4.2. Send Data in Direct Push Mode 31

3.4.3. Retrieve Data in Direct Push Mode 31
3.4.4. Close an SSL Connection 32
3.5. SSL Client in Transparent Access Mode 32
3.5.1. Set up an SSL Connection and Send Data in Transparent Access Mode 32
3.5.2. Set up an SSL Connection and Retrieve Data in Transparent Access Mode 32
3.5.3. Close an SSL Connection 32
3.6. DTLS Connection Based on PSK Encryption 33
3.6.1. Set up a DTLS Connection 33
3.6.2. Close an SSL Connection 33

4 Check for Failure in SSL Connection 34

5 Summary of Error Codes 35

6 Appendix References 37

Table Index

Table 1: Applicable Modules 7
Table 2: Supported SSL Versions 8
Table 3: Supported SSL Cipher Suites 8
Table 4: Types of AT Commands 13
Table 5: Summary of Error Codes 35
Table 6: Related Documents 37
Table 7: Terms and Abbreviations 37

1 Introduction

Quectel BG95 series, BG77 and BG600L-M3 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 to prevent tampering, eavesdropping attacks, or data forging.

1.1. Applicable Modules

Table 1: Applicable Modules

Module Series	Model	Description
BG95	BG95-M1	Cat M1 only
	BG95-M2	Cat M1/Cat NB2
	BG95-M3	Cat M1/Cat NB2/EGPRS
	BG95-M4	Cat M1/Cat NB2, 450 MHz Supported
	BG95-M5	Cat M1/Cat NB2/EGPRS, Power Class 3
	BG95-M6	Cat M1/Cat NB2, Power Class 3
	BG95-MF	Cat M1/Cat NB2, Wi-Fi Positioning
BG77	BG77	Cat M1/Cat NB2
BG600L	BG600L-M3	Cat M1/Cat NB2/EGPRS

1.2. SSL Versions and Cipher Suites

The following SSL versions are supported by BG95 series, BG77 and BG600L-M3 modules.

Table 2: Supported SSL Versions

SSL Version
SSL3.0
TLS1.0
TLS1.1
TLS1.2

SSL cipher suites supported by the modules are presented in the table below. For detailed description of cipher suites, refer to *RFC 2246: The TLS Protocol Version 1.0*.

Table 3: Supported SSL Cipher Suites

Cipher Suite Cide	Cipher Suite Name
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
0XC02B	TLS-ECDHE-ECDSA-WITH-AES-128-GCM-SHA256
0XC02F	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
0XC0A8	TLS_PSK_WITH_AES_128_CCM_8
0X00AE	TLS_PSK_WITH_AES_128_CBC_SHA256
0XC0AE	TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8

0XFFFF	Support all cipher suites above

1.3. Using SSL Function

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

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

Step 3: Configure the SSL version, cipher suite, trusted CA certificate path and the security level for a specified SSL context with **AT+QSSLCFG**.

Step 4: Open an SSL socket to connect a remote server with **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 detailed information about how to send and receive data in each data access mode, see ***Chapter 1.4***

Step 6: Close SSL connection with **AT+QSSLCLOSE**.

Step 7: Deactivate the PDP context with **AT+QIDEACT**. See ***document [1]*** for detailed information.

1.4. Description of Data Access Modes

The SSL connection supports three data access modes:

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

When opening an SSL connection via **AT+QSSLOPEN**, you can specify the data access mode with **<access_mode>**. After the SSL connection is established, you can switch the data access mode with **AT+QISWTMD** (see ***document [1]*** for details).

1. In buffer access mode, the module buffers data upon receiving them and reports a URC in the format of **+QSSLURC: "recv",<clientID>** to notify the host of the incoming data. In such a case, the host can retrieve the buffered data with **AT+QSSLRECV**.

2. In direct push mode, the module outputs the received data directly through UART1, USB modem or USB AT port as URC of **+QSSLURC: "recv",<clientID>,<current_recvlength><CR><LF><data>**.

3. In transparent transmission mode, the corresponding COM port (UART port, USB modem port, etc.) enters exclusive mode, in which the data received from the port are directly sent to the Internet, and data received from the Internet are directly outputted via the port.

- **Exit from transparent transmission mode**

The module can exit from transparent transmission mode through either of the following ways.

1) Execute **+++**. Follow the requirements below to prevent the **+++** from being misinterpreted as data:

   a) Do not input any character within 1 second before and after inputting **+++**.
   b) Input **+++** within 1 second, and wait until **OK** is returned. When **OK** is returned, the module is switched to buffer access mode.

2) Change MAIN_DTR from LOW to HIGH to make the module enter command mode. In this case, set **AT&D1** (see ***document [2]***) before the module enters transparent transmission mode.

- **Return to transparent transmission mode**

1) By **AT+QISWTMD**. Specify **<access_mode>** as 2 when executing this command. When transparent transmission mode is entered successfully, **CONNECT** is returned.

2) By **ATO** (see ***document [2]***). After a connection exits from transparent transmission mode, executing **ATO** will switch the data access mode back to transparent transmission mode. When transparent transmission mode is entered successfully, **CONNECT** is returned. If no connection has entered transparent transmission mode before, **ATO** returns **NO CARRIER**.

1.5. Certificate Validity Check

To check certificate validity, compare your local time against certificate validity range ("Not before" and "Not after"). If the local time is earlier than the "Not before" time or later than the "Not after" time the certificate has expired.

When certificate validity check is required (set **<ignore_localetime>** as 0 when executing **AT+QSSLCFG**), to avoid validity period check failure, execute **AT+CCLK** to configure the module time within the certificate validity period. See ***document [2]*** for detailed information about **AT+CCLK**.

1.6. Server Name Indication

SNI (Server Name Indication) allows the server to safely host multiple TLS Certificates since it provides Server Host Name information as an extension in the client hello message. It thus enhances connection security with multiple virtual servers based on a single IP address. This feature is only applicable to 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 Commands

Command Type	Syntax	Description
Test Command	**AT+<cmd>=?**	Test the existence of corresponding Write Command and return information about the type, value, or range of its parameter.
Read Command	**AT+<cmd>?**	Check the current parameter value of a corresponding Write 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.

2.2. Declaration of AT Command Examples

The AT command examples in this document are provided to help you learn about how to use the AT commands introduced herein. 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.

2.3. Description of SSL AT Commands

2.3.1. AT+QSSLCFG Configure Parameters of an SSL Context

The command configures the SSL version, cipher suite, security level, CA certificate, client certificate and client key. 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. Several SSL connections can be established based on one SSL context. 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: "ciphersuite",(**range of supported **<SSL_ctxID>**s**),(**list of supported **<cipher_suites>**s**)** **+QSSLCFG: "cacert",(**range of supported **<SSL_ctxID>**s**),<cacertpath>** **+QSSLCFG: "clientcert",(**range of supported **<SSL_ctxID>**s**),<clientcertpath>** **+QSSLCFG: "clientkey",(**range of supported **<SSL_ctxID>**s**),<clientkeypath>** **+QSSLCFG: "seclevel",(**range of supported **<SSL_ctxID>**s**),(**range of supported **<seclevel>**s**)** **+QSSLCFG: "session",(**range of supported **<SSL_ctxID>**s**),(**list of supported **<session>**s**)** **+QSSLCFG: "sni",(**range of supported **<SSL_ctxID>**s**),(**list of supported **<SNI>**s**)** **+QSSLCFG: "checkhost",(**range of supported **<SSL_ctxID>**s**),(**list of supported **<check_host>**s**)**

	+QSSLCFG: "ignorelocaltime",(range of supported **<SSL_ctxID>**s**),(**list of supported **<ignore_localtime>**s**)** **+QSSLCFG: "negotiatetime",(**range of supported **<SSL_ctxID>**s**),(**range of supported **<negotiate_time>**s**)** **+QSSLCFG: "renegotiation",(**range of supported **<SSL_ctxID>**s**),(**list of supported **<renegotiation>**s**)** **+QSSLCFG: "dtls",(**range of supported **<SSL_ctxID>**s**),(**list of supported **<DTLS_enable>**s**)** **+QSSLCFG: "dtlsversion",(**range of supported **<SSL_ctxID>**s**),(**range of supported **<DTLS_version>**s**)** **OK**
Write Command **AT+QSSLCFG="sslversion",<SSL_ctxID>[,<SSL_version>]**	Response If the optional parameter is omitted, query the current SSL version of 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 **AT+QSSLCFG="ciphersuite",<SSL_ctxID>[,<cipher_suites>]**	Response If the optional parameter is omitted, query the current SSL cipher suite of 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 **AT+QSSLCFG="cacert",<SSL_ctxID>[,<cacertpath >]**	Response If the optional parameter is omitted, query the current path of 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 **AT+QSSLCFG="clientcert",<SSL_ctxID>[,<clientcertpath>]**	Response If the optional parameter is omitted, query the current path of client certificate for the specified SSL context: **+QSSLCFG: "clientcert",<SSL_ctxID>,<clientcertpath>** **OK** If the optional parameter is specified, set the path of client certificate for the specified SSL context: **OK** Or **ERROR**
Write Command **AT+QSSLCFG="clientkey",<SSL_ctxID>[,<clientkeypath>]**	Response If the optional parameter is omitted, query the current path of client private key for the specified SSL context: **+QSSLCFG: "clientkey",<SSL_ctxID>,<clientkeypath>** **OK** If the optional parameter is specified, set the path of the client private key for the specified SSL context: **OK** Or **ERROR**
Write Command **AT+QSSLCFG="seclevel",<SSL_ctxID>[,<seclevel>]**	Response If the optional parameter is omitted, query the current authentication mode of 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 **AT+QSSLCFG="session",<SSL_ctxID>[,<session>]**	Response If the optional parameter is omitted, query whether SSL Resumption feature is enabled for the specified SSL context: **+QSSLCFG: "session",<SSL_ctxID>,<session>**

	OK If the optional parameter is specified, enable/disable SSL Resumption feature for the specified SSL context: **OK** Or **ERROR**
Write Command **AT+QSSLCFG="sni",<SSL_ctxID>[,<SNI>]**	Response If the optional parameter is omitted, query whether server name indication feature is enabled for the specified SSL context: **+QSSLCFG: "sni",<SSL_ctxID>,<SNI>** **OK** If the optional parameter is specified, enable/disable server name indication feature for the specified SSL context: **OK** Or **ERROR**
Write Command **AT+QSSLCFG="checkhost",<SSL_ctxID>[,<check_host>]**	Response If the optional parameter is omitted, query whether the hostname validation feature is enabled for the specified SSL context: **+QSSLCFG: "checkhost",<SSL_ctxID>,<check_host>** **OK** If the optional parameter is specified, enable/disable hostname validation feature for the specified SSL context: **OK** Or **ERROR**
Write Command **AT+QSSLCFG="ignorelocaltime",<SSL_ctxID>[,<ignore_localtime>]**	Response If the optional parameter is omitted, query whether the certificate validity check is ignored for the specified SSL context: **+QSSLCFG: "ignorelocaltime",<SSL_ctxID>,<ignore_localtime>** **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 **AT+QSSLCFG="negotiatetime",<SSL_ctxID>[,<negotiate_time>]**	Response If the optional parameter is omitted, query the maximum timeout of SSL negotiation for the specified SSL context: **+QSSLCFG: "negotiatetime",<SSL_ctxID>,<negotiate_time>** **OK** If the optional parameter is specified, set the maximum timeout of SSL negotiation for the specified SSL context: **OK** Or **ERROR**
Write Command **AT+QSSLCFG="renegotiation",<SSL_ctxID>[,<renegotiation>]**	Response If the optional parameter is omitted, query whether support for TLS renegotiation is enabled for the specified SSL context: **+QSSLCFG: "renegotiation",<SSL_ctxID>,<renegotiation>** **OK** If the optional parameter is specified, set whether to enable support for TLS renegotiation for the specified SSL context: **OK** Or **ERROR**
Write Command **AT+QSSLCFG="dtls",<SSL_ctxID>[,<DTLS_enable>]**	Response If the optional parameter is omitted, query whether DTLS feature is enabled for the specified SSL context: **+QSSLCFG: "dtls",<SSL_ctxID>,<DTLS_enable>** **OK** If the optional parameter is specified, set whether to enable DTLS feature for the specified SSL context: **OK** Or **ERROR**

Write Command **AT+QSSLCFG="dtlsversion",<SSL_ctxID>[,<DTLS_version>]**	Response If the optional parameter is omitted, query the current DTLS version of the specified SSL context: **+QSSLCFG: "dtlsversion",<SSL_ctxID>,<DTLS_version>** **OK** If the optional parameter is specified, set the DTLS version of the specified SSL context: **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	The command takes effect immediately. The configurations will not be saved.

Parameter

<SSL_ctxID>	Integer type. SSL context ID. Range: 0–5.	
<SSL_version>	Integer type. SSL Version.	
	0	SSL3.0
	1	TLS1.0
	2	TLS1.1
	3	TLS1.2
	4	All
<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
	0XC02B	TLS-ECDHE-ECDSA-WITH-AES-128-GCM-SHA256
	0XC02F	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
	0XC0A8	TLS_PSK_WITH_AES_128_CCM_8
	0X00AE	TLS_PSK_WITH_AES_128_CBC_SHA256
	0XC0AE	TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8
	0XFFFF	Support all
<cacertpath>	String type. The path of the trusted CA certificate.	
<clientcertpath>	String type. The path of the client certificate.	
<clientkeypath>	String type. The path of the client private key.	
<seclevel>	Integer type. Authentication mode.	
	0	No authentication
	1	Perform server authentication
	2	Perform server and client authentication if requested by the remote server
<session>	Integer type. Enable or disable SSL Resumption feature.	
	0	Disable
	1	Enable
<SNI>	Integer type. Enable or disable Server Name Indication feature. DNS hostnames are currently the only supported server names.	
	0	Disable
	1	Enable
<check_host>	Integer type. Enable or disable hostname validation feature (Subject Common Name (CN) matches the specified host name).	
	0	Disable
	1	Enable
<ignore_localtime>	Integer type. Whether or not to ignore certificate validity check.	
	0	Not to ignore
	1	Ignore
<negotiate_time>	Integer type. The maximum timeout of SSL negotiation. Range: 10–300. Default value: 300. Unit: second.	

<renegotiation>	Integer type. Enable or disable support for TLS renegotiation. 0 Disable 1 Enable
<DTLS_enable>	Integer type. Enable or disable DTLS feature. 0 Disable 1 Enable
<DTLS_version>	Integer type. DTLS version. 0 DTLS1.0 1 DTLS1.2 2 Both

2.3.2. 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 with **AT+QSSLCFG** will be used in the handshake procedure. After successful handshake with the Internet, the module can send or receive data via this SSL connection. In addition, 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 cannot be received during that time, use **AT+QSSLCLOSE** 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 **<PDP_ctxID>**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=<PDP_ctxID>,<SSL_ctxID>,<clientID>,<serveraddr>,<server_port>[,<access_mode>]**	Response If the **<access_mode>**=2 (transparent access mode) and the SSL connection is successfully set up: **CONNECT** If the **<access_mode>**=0/1 (buffer access mode or direct push mode): **OK**

	+QSSLOPEN: <clientID>,<err> 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 will not be saved.

Parameter

<PDP_ctxID>	Integer type. PDP context ID. Range: 1–16.
<SSL_ctxID>	Integer type. SSL context ID. Range: 0–5.
<clientID>	Integer type. Socket index. Range: 0–11.
<serveraddr>	String type. Remote server address.
<server_port>	Integer type. The listening port of remote server. Range: 0–65535.
<access_mode>	Integer type. The data access mode of SSL connection. 0 Buffer access mode 1 Direct push mode 2 Transparent transmission mode
<err>	The error code of the operation. See ***Chapter 5***.
<negotiate_time>	Integer type. The maximum timeout of SSL negotiation. Range: 10–300. Default value: 300. Unit: second.

NOTE

Error description can be got via **AT+QIGETERROR**. For more details, refer to ***document [1]***.

2.3.3. 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	Response **>**

AT+QSSLSEND=<clientID>	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 sending is successful: **SEND OK** If the connection has been established but sending buffer is full: **SEND FAIL** If the connection has not been established, abnormally closed, or any 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 the connection has been established and sending is successful: **SEND OK** If the connection has been established butsending buffer is full: **SEND FAIL** If the connection has not been established, abnormally closed, or the parameter is incorrect: **ERROR**
Maximum Response Time	300 ms
Characteristics	/

Parameter

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

2.3.4. AT+QSSLRECV Retrieve Data Through SSL Connection

When an SSL connection is opened with **<access_mode>** specified as 0 (buffer access mode), the module reports URC **+QSSLURC: "recv",<clientID>** when it receives data from the Internet. You can read the data from buffer with **AT+QSSLRECV**.

AT+QSSLRECV Retrieve Data Through SSL Connection	
Test Command **AT+QSSLRECV=?**	Response **+QSSLRECV: (**range of supported **<clientID>**s**),(**range of supported **<readlen>**s**)** **OK**
Write Command When **<readlen>** is not 0, retrieve the data in specified length **AT+QSSLRECV=<clientID>[,<readlen>]**	Response If the specified connection has received data: **+QSSLRECV: <read_actual_length><CR><LF><data>** **OK** When the buffer is empty: **+QSSLRECV: 0** **OK** If any parameter is incorrect or the connection cannot be established: **ERROR**
Write Command When **<readlen>** is 0, query the retrieved data length **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	/

Parameter

<clientID>	Integer type. Socket index. Range: 0–11.
<readlen>	Integer type. The length of data to be retrieved. Range: 0–1500. Default

	value: 1500. Unit: byte.
<read_actual_length>	Integer type. Length of the data that have been actually retrieved. Unit: byte.
<data>	String type. Retrieved data.
<total_receive_length>	Integer type. Total length of received data. Unit: byte.
<have_read_length>	Integer type. Length of the data that have been retrieved. Unit: byte.
<unread_length>	Integer type. Length of the data that have not been retrieved. Unit: byte.

2.3.5. 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 releases 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 **OK** Or **ERROR**
Maximum Response Time	10 s by default, determined by parameter **<close_timeout>**
Characteristics	The command takes effect immediately. The configuration will not be saved.

Parameter

<clientID>	Integer type. Socket index. Range: 0–11.
<close_timeout>	Integer type. The timeout value of **AT+QSSLCLOSE**. Range: 0–65535. Default value: 10. Unit: second. 0 means closing immediately.

2.3.6. AT+QSSLSTATE Query the State of SSL Connections

The command queries the state of SSL connections.

AT+QSSLSTATE Query the State of SSL Connections	
Test Command **AT+QSSLSTATE=?**	Response **OK**
Write Command **AT+QSSLSTATE=<clientID>**	Response **+QSSLSTATE: <clientID>,"SSLClient",<IP_addr**

	ess>,<remote_port>,<local_port>,<socket_state>,<PDP_ctxID>,<serverID>,<access_mode>,<AT_port>,<SSL_ctxID> **OK**
Execution Command **AT+QSSLSTATE**	Response **+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

<clientID>	Integer type. Socket index. Range: 0–11.
<IP_address>	String type. Remote server address.
<remote_port>	Integer type. Remote server port. Range: 0–65535.
<local_port>	Integer type. 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" Connection has been established. 4 "Closing" Connection is closing
<PDP_ctxID>	Integer type. PDP context ID. Range: 1–16.
<serverID>	Integer type. Reserved. The value is usually the same as **<clientID>**.
<access_mode>	Integer type. The data access mode of SSL connection. 0 Buffer access mode 1 Direct push mode 2 Transparent transmission mode
<AT_port>	String type. COM port.
<SSL_ctxID>	Integer type. SSL context ID. Range: 0–5.

2.3.7. AT+QSSLCRYPT Encrypt/Decrypt Data with a Specified Algorithm

This command encrypts/decrypts data with a specified algorithm. It supports AES-128-CBC encryption/decryption and MD5 encryption.

AT+ QSSLCRYPT Encrypt/Decrypt Data with a Specified Algorithm	
Test Command **AT+QSSLCRYPT=?**	Response **+QSSLCRYPT: ("AES","MD5"),<input_data>,(**list of supported **<format>**s**),(**list of supported **<mode>**s**),<key>** **OK**
Write Command **AT+QSSLCRYPT="AES",<input_data>,<format><mode>,<key>**	Response **+QSSLCRYPT: "AES",<output_data>** **OK** If there is any error: **ERROR**
Write Command **AT+QSSLCRYPT="MD5",<input_data>**	Response **+QSSLCRYPT: "MD5",<output_data>** **OK** If there is any error: **ERROR**
Maximum Response Time	300 ms
Characteristic	/

Parameter

<input_data>	In AES-128-CBC mode: The data to be encrypted or decrypted. The data format is specified by **<format>**. The maximum length of **<input_data>** is 256 bytes. The length of data to be encrypted/decrypted with AES-128-CBC should be a multiple of 16 bytes. In MD5 mode: String type. The data to be encrypted. The maximum length of **<input_data>** is 256 bytes.
<format>	Integer type. The format of input data. 0 String format 1 Hex format
<mode>	Integer type. AES-128-CBC mode. 0 AES-128-CBC decryption

	1 AES-128-CBC encryption
<key>	String type. The encryption/decryption key of AES-128-CBC. The length must be 16 bytes.
<output_data>	Hex String type. The encrypted data or decrypted data.

2.4. Description of URCs

2.4.1. +QSSLURC: "recv" Notify Received Data

The URC notifies the data received from peer in buffer access mode and direct push mode.

+QSSLURC: "recv" Notify Received Data	
+QSSLURC: "recv",<clientID>	The URC of SSL data incoming in buffer access mode. SSL data can be retrieved with **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 received data.

2.4.2. +QSSLURC: "closed" Notify Abnormal Disconnection

The URC notifies that the SSL 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=<connectID>** 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,"CMNBIOT","","",1     //Configure PDP context as 1. APN is "CMNBIOT" for China Mobile NB-IoT network
OK
```

3.1.2. Activate a PDP Context

```
AT+QIACT=1                          //Activate PDP context 1.
OK                                  //Activated the context successfully.
AT+QIACT?                           //Query the context state, protocol type and IP address of PDP context 1.
+QIACT: 1,1,1,"100.142.162.0"

OK
```

3.1.3. Deactivate a PDP Context

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

3.2. Configure an SSL Context

```
AT+QSSLCFG="sslversion",1,1         //Set SSL context ID as 1 and SSL version as TLS1.0.
OK
AT+QSSLCFG="ciphersuite",1,0X0035   //Set SSL context ID as 1 and cipher suite 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,"cacert.pem"    //Set path of the trusted CA certificate of SSL context ID 1
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    //Set up an SSL connection.
OK

+QSSLOPEN: 4,0                                //Set up the SSL connection successfully.
AT+QSSLSTATE                                  //Query the state of 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

```
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
```

3.3.3. Retrieve Data in Buffer Access Mode

```
+QSSLURC: "recv",4             //The Socket 4 (<clientID>=4) has received data.
AT+QSSLRECV=4,1500             //Retrieve the data. The length of data to be retrieved is 1500 bytes.
+QSSLRECV: 18                  //The retrieved data length is 18 bytes.
Test data from SSL

OK
AT+QSSLRECV=4,1500
+QSSLRECV: 0                   //The buffer is empty.
```

```
OK
```

3.3.4. Close an SSL Connection

```
AT+QSSLCLOSE=4                  //Close the 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    //Set up an SSL connection.
OK

+QSSLOPEN: 4,0                                 //Set up the SSL connection successfully.
AT+QSSLSTATE                                   //Query the status of 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. Retrieve Data in Direct Push Mode

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

3.4.4. Close an SSL Connection

AT+QSSLCLOSE=4	//Close the 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 transmission mode. //The client is sending data from COM port to the Internet directly. (The data are not visible in the example.)
OK	//Use **+++** or MAIN_DTR (set **AT&D1** first) to exit transparent transmission mode. The **NO CARRIER** result code indicates that the server has stopped the SSL connection.

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

AT+QSSLOPEN= 1,1,4,"220.180.239.212",8011,2	//Set up an SSL connection.
CONNECT	
<Received data>	//The client is reading the data.
OK	//Use **+++** or MAIN_DTR (set **AT&D1** first) to exit transparent transmission 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 the connection (**<clientID>**=4). Depending on the network, the maximum response time is 10 s.
OK	

3.6. DTLS Connection Based on PSK Encryption

3.6.1. Set up a DTLS Connection

```
AT+QFUPL="0_server.psk"                    //Upload PSK file first. The PSK file should be named as SSL_ctxID_server.psk (0_server.psk for instance) and the content of the file should be in the format of <PSK_ID>&<PSK_key> (e.g., DTLS_Client&1a2b3c4d).
CONNECT
<Input file bin data>
+QFUPL: 24,553

OK
AT+QSSLCFG="dtls",0,1                      //Enable DTLS feature for SSL context 0.
OK
AT+QSSLCFG="dtlsversion",0,0               //Configure DTLS version to DTLS1.0 for SSL context 0.
OK
AT+QSSLCFG="ciphersuite",0,0X00AE          //Configure cipher suite for SSL context 0.
OK
AT+QSSLOPEN=1,0,0,"220.180.239.201",8010,0
OK

+QSSLOPEN: 0,0                             //Set up the SSL connection successfully.
AT+QSSLSTATE                               //Query the status of SSL connections.
+QSSLSTATE: 0,"SSLClient","220.180.239.201",8010,65344,2,1,4,0,"usbmodem",1

OK
```

3.6.2. Close an SSL Connection

```
AT+QSSLCLOSE=0                             //Close the connection (<clientID>=0). Depending on the network, the maximum response time is 10 s.

OK
```

4 Check for Failure in SSL Connection

Please find reasons for the failure to open an SSL connection as follows:

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

2. If the address of server is a domain name, please check whether the address of DNS server is valid with **AT+QIDNSCFG=<contextID>**. Because an invalid DNS server address cannot convert domain name to IP address.

3. Check the SSL configuration with **AT+QSSLCFG**, especially the SSL version and cipher suite, to make sure 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** (see ***document [3]***). If the server has configured "SSLVerifyClient required", the client certificate and client private key have to be uploaded to the module with **AT+QFUPL** (see ***document [3]***). For detailed information about certificate validity check, see ***Chapter 1.5***. And for more information about related FILE AT commands, see ***document [2]***.

5 Summary of Error Codes

If an **ERROR** 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 last SSL AT command.

Table 5: Summary of Error Codes

<err>	Meaning
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

6 Appendix References

Table 6: Related Documents

Document Name
[1] Quectel_BG95&BG77&BG600L_Series_TCP(IP)_Application_Note
[2] Quectel_BG95&BG77&BG600L_Series_AT_Commands_Manual_V2.0
[3] Quectel_BG95&BG77&BG600L_Series_FILE_Application_Note

Table 7: Terms and Abbreviations

Abbreviation	Description
APN	Access Point Name
CA	Certificate Authority
CR	Carriage Return
DNS	Domain Name Server
DTR	Data Terminal Ready
DTLS	Datagram Transport Layer Security
EGPRS	Enhanced General Packet Radio Service
GPRS	General Packet Radio Service
ID	Identifier
IP	Internet Protocol
LF	Line Feed
PDP	Packet Data Protocol
SNI	Server Name Indication

SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UART	Universal Asynchronous Receiver/Transmitter
URC	Unsolicited Result Code
USB	Universal Serial Bus