BC660K-GL&BC950K-GL CoAP Application Note

NB-IoT Module Series

Version: 1.0

Date: 2023-07-07

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.

About the Document

Revision History

Version	Date	Author	Description
-	2023-04-13	Randy LI/ Caden ZHANG	Creation of the document
1.0	2023-07-07	Yance YANG/ Randy LI/ Caden ZHANG	First official release

Contents

Table Index

1 Introduction

This document explains how to use the CoAP feature on Quectel BC660K-GL and BC950K-GL modules through AT commands.

2 Description of CoAP 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 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 1: Types of AT Commands

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.

2.2. Declaration of AT Command Examples

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

2.3. AT Command Details

2.3.1. AT+QCOAPCFG Configure Optional Parameters of CoAP Client

This command configures optional parameters of a CoAP client.

AT+QCOAPCFG Configure Optional Parameters of CoAP Client	
Write Command Query/Set the DTLS mode for a specified CoAP client. **AT+QCOAPCFG="dtls",<clientID>[,<DTLS_enable>]**	Response If the optional parameter is omitted, query the current setting: **+QCOAPCFG: "dtls",<DTLS_enable>** **OK** If the optional parameter is specified, set the DTLS mode for the specified CoAP client: **OK** If there is any error: **ERROR** Or **+CME ERROR: <err>**
Write Command Query/Set the PSK used in handshake for a specified CoAP client. **AT+QCOAPCFG="psk",<clientID>[,<identity>,<key>]**	Response If the optional parameters are omitted, query the current setting: **+QCOAPCFG: "psk",<identity>,<key>** **OK** If the optional parameters are specified, set the PSK used in handshake for a specified CoAP client: **OK**

	If there is any error: **ERROR** Or **+CME ERROR: <err>**
Maximum Response Time	300 ms
Characteristics	This command takes effect immediately; The configurations are not saved.

Parameter

<clientID> Integer type. CoAP client identifier. Range: 0–4.

<DTLS_enable> Integer type. Whether to enable DTLS mode for CoAP client.
0 Use normal UDP connection for CoAP client
1 Use DTLS connection for CoAP client

<identity> String type. PSK identity. Length: 0–152. Unit: byte.

<key> Hex string type. PSK key. Length: 0–256. Unit: byte.

<err> Integer type. Error code. See ***Chapter 6*** for more details.

Example

```
AT+QCOAPCFG="dtls",0,1                   //Use DTLS connection for CoAP client.
OK
AT+QCOAPCFG="dtls",0                     //Query the current DTLS configuration.
+QCOAPCFG: "dtls",1

OK                                       //Set the handshake PSK for the specified CoAP client.
AT+QCOAPCFG="psk",0,"869154040004132","46694a6871617a3062706d68454c6e4c"
OK
AT+QCOAPCFG="psk",0                      //Query the handshake PSK for the specified CoAP client.
+QCOAPCFG: "psk","869154040004132","46694a6871617a3062706d68454c6e4c"

OK
AT+QCOAPOPEN=0,"220.180.239.212",8032        //Create a CoAP context.
OK

+QCOAPOPEN: 0,0
```

2.3.2. AT+QCOAPOPEN Create a CoAP Context

This command creates a CoAP context.

AT+QCOAPOPEN Create a CoAP Context	
Test Command **AT+QCOAPOPEN=?**	Response **+QCOAPOPEN: (**range of supported **<clientID>**s)**,<CoAP_server>,(**range of supported **<port>**s**)** **OK**
Read Command **AT+QCOAPOPEN?**	Response **[+QCOAPOPEN: <clientID>,<CoAP_server>,<port>,<status>]** **OK**
Write Command **AT+QCOAPOPEN=<clientID>,<CoAP_server>,<port>**	Response **OK** **+QCOAPOPEN: <clientID>,<result>** If there is any error: **ERROR** Or **+CME ERROR: <err>**
Maximum Response Time	300 ms
Characteristics	This command takes effect immediately; The configurations are not saved.

Parameter

<clientID>	Integer type. CoAP client identifier. Range: 0–4.
<CoAP_server>	String type. CoAP server address. It can only be an IP address in IPv4 or IPv6. Maximum length: 100 bytes.
<port>	Integer type. CoAP server port. Range: 1–65535.
<status>	Integer type. Current status of the specified CoAP client. 0 Idle or disconnected. 1 CoAP client is opening. 2 CoAP client is connecting to the CoAP server. 3 CoAP client is connected to the CoAP server. 4 CoAP client is disconnecting from the CoAP server.
<result>	Integer type. Command execution result. See ***Chapter 5*** for details.
<err>	Integer type. Error codes. See ***Chapter 6*** for details.

Example

```
AT+QCOAPOPEN=?                                //Query the supported parameter value.
+QCOAPOPEN: (0-4),<CoAP_server>,(1-65535)

OK
AT+QCOAPOPEN=0,"220.180.239.212",8032         //Create a CoAP context.
OK

+QCOAPOPEN: 0,0
AT+QCOAPOPEN?                                 //Query the current status of the CoAP client.
+QCOAPOPEN: 0,"220.180.239.212",8032,3

OK
```

2.3.3. AT+QCOAPCLOSE Delete a CoAP Context

This command deletes a CoAP context.

AT+QCOAPCLOSE Delete a CoAP Context	
Test Command **AT+QCOAPCLOSE=?**	Response **+QCOAPCLOSE: (**range of supported **<clientID>**s**)** **OK**
Write Command **AT+QCOAPCLOSE=<clientID>**	Response **OK** **+QCOAPCLOSE: <clientID>,<result>** If there is any error: **ERROR** Or **+CME ERROR: <err>**
Maximum Response Time	300 ms
Characteristics	/

Parameter

<clientID>	Integer type. CoAP client identifier. Range: 0–4.
<result>	Integer type. Command execution result. See ***Chapter 5*** for details.
<err>	Integer type. Error codes. See ***Chapter 6*** for details.

Example

```
AT+QCOAPCLOSE=?                              //Query the supported parameter value.
+QCOAPCLOSE: (0-4)

OK
AT+QCOAPOPEN=0,"220.180.239.212",8032        //Create a CoAP context.
OK

+QCOAPOPEN: 0,0
AT+QCOAPCLOSE=0                              //Delete a CoAP context.
OK

+QCOAPCLOSE: 0,0
```

2.3.4. AT+QCOAPOPTION Configure CoAP Message Options

This command configures the options of a CoAP message.

AT+QCOAPOPTION Configure CoAP Message Options	
Test Command **AT+QCOAPOPTION=?**	Response **+QCOAPOPTION: (**range of supported **<clientID>**s**),(**list of supported **<opt_index>**s**),<opt_name>,(**range of supported **<opt_length>**s**),<opt_value>,(**list of supported **<opt_flag>**s**)** **OK**
Write Command If **<opt_name>**is not 11 **AT+QCOAPOPTION=<clientID>,<opt_index>,<opt_name>,<opt_length>,<opt_value>**	Response **OK** If there is any error: **ERROR** Or **+CME ERROR: <err>**
Write Command If **<opt_name>**is 11 **AT+QCOAPOPTION=<clientID>,<opt_index>,<opt_name>,<opt_length>,<opt_value>[,<opt_flag>]**	Response If the optional parameter is omitted, use "/" to divide Uri-Path into multiple CoAP options by default: **OK** If the optional parameter is specified, configure Uri-Path message options: **OK** If there is any error:

	ERROR Or **+CME ERROR: <err>**
Maximum Response Time	300 ms
Characteristics	This command takes effect immediately; The configurations are not saved.

Parameter

<clientID>	Integer type. CoAP client identifier. Range: 0–4.
<opt_index>	Integer type. Index of the CoAP option to be added/deleted. Only 0 is supported currently, indicating not to configure any option index.
<opt_name>	Integer type. CoAP option name. See *RFC 7252* and *RFC7959* for details. 1 If-Match 3 Uri-Host 4 ETag 5 If-None-Match 6 Observe 8 Location-Path 11 Uri-Path 12 Content-Format 14 Max-Age 15 Uri-Query 17 Accept 20 Location-Query 23 Block2 27 Block1 28 Size2 35 Proxy-Uri
<opt_length>	Integer type. Length of **<opt_value>**. Range: 0–255. Unit: byte.
<opt_value>	String type. CoAP option values. Maximum length: 255 bytes. See *RFC 7252* and *RFC7959* for details. If **<opt_name>** is 23 or 27, use the formula below, which requires inputting the expected values of block_num, block_more and block_size, to calculate the value of **<opt_value>**: **<opt_value> = (block_num << 4) | (block_more&0x8) | log2(block_size/16)** If **<opt_name>** is 12 or 17, the value of **<opt_value>** is as follows: “0” Text-plain “40” Application/link-format “41” Application/xml “42” Application/octet-stream “47” Application/exi “50” Application/json

<opt_flag>	Integer type. Indicates whether the CoAP option value is segmented, and only need to be configured when **<opt_name>**=11 (the option is Uri-path). 0 Use "/" to divide Uri-Path into multiple CoAP options 1 Uri-Path as a whole
<err>	Integer type. Error codes. See ***Chapter 6*** for details.

Example

```
AT+QCOAPOPTION=?
+QCOAPOPTION: (0-4),(0),<opt_name>,(0-255),<opt_value>,(0,1)

OK
//Set the CoAP option to 11 (Uri-path), and specify the CoAP option value as "rd". The length of the value is 2 bytes, and Uri-path is divided using "/" as a separator.
AT+QCOAPOPTION=0,0,11,2,"rd"
OK
//Set the CoAP option to 15 (Uri-Query), and specify the CoAP option value as "ep=86370303". The length of the value is 11 bytes.
AT+QCOAPOPTION=0,0,15,11,"ep=86370303"
OK
```

NOTE

Please note the option definitions in the table below:

<opt_name>	<opt_value>	<opt_length>
1 (If-Match)	String type	Range: 0–8
3 (Uri-Host)	String type	Range: 1–255
4 (ETag)	String type	Range: 1–8
5 (If-None-Match)	NULL	supports 0 only
6 (Observe)	Integer type	Range: 0–255
8 (Location-Path)	String type	Range: 0–255
11 (Uri-Path)	String type	Range: 0–255
12 (Content-Format)	String type	Range: 1–255
14 (Max-Age)	Integer type	Range: 0–255
15 (Uri-Query)	String type	Range: 0–255

17 (Accept)	String type	Range: 1–255
20 (Location-Query)	String type	Range: 0–255
23 (Block2)	Integer type	Range: 0–255
27 (Block1)	Integer type	Range: 0–255
28 (Size2)	Integer type	Range: 0–255
35 (Proxy-Uri)	String type	Range: 1–255

2.3.5. AT+QCOAPHEAD Configure CoAP Message ID and Token

This command sets the CoAP message ID and token.

AT+QCOAPHEAD Configure CoAP Message ID and Token	
Test Command **AT+QCOAPHEAD=?**	Response **+QCOAPHEAD: (**range of supported **<clientID>**s**),(**range of supported **<mode>**s）**,(**range of supported **<msgID>**）**,**（range of supported **<token_length>**s）**,<token>** **OK** If there is any error: **ERROR** Or **+CME ERROR:<err>**
Write Command **AT+QCOAPHEAD=<clientID>,<mode>[,[<msgID>][,<token_length>,<token>]]**	Response **OK** If there is any error: **ERROR** Or **+CME ERROR:<err>**
Maximum Response Time	300 ms
Characteristics	This command takes effect immediately; The configurations are not saved.

Parameter

<clientID>	Integer type. CoAP client identifier. Range: 0–4.

<mode>	Integer type. Message ID and token mode selection. Range: 0–5. 0 Generate message ID and token values randomly. 1 Generate message ID randomly, and set the token value. 2 Generate message ID randomly, token value is omitted. 3 Set message ID only; token value is omitted. 4 Set message ID; generate token values randomly. 5 Configure message ID and token values.
<msgID>	Integer type. CoAP message ID, which only needs to be configured when the **<mode>** is 3, 4 or 5. Range: 0–65535.
<token_length>	Integer type. Length of token values, which only needs to be configured when the **<mode>** is 1 or 5. Range: 1–8. Unit: byte.
<token>	String in Hex. Token values, which can be configured when the **<mode>** is 1 or 5.
<err>	Integer type. Error codes. See ***Chapter 6*** for details.

Example

```
AT+QCOAPHEAD=?                          //Query the supported parameter range.
+QCOAPHEAD: (0-4),(0-5),(0-65535),(1-8),<token>

OK
AT+QCOAPHEAD=0,0                        //Generate message ID and token values randomly.
OK
AT+QCOAPHEAD=0,1,4,”02040608”    //Generate message ID randomly, and set the token value to
                                           "02040608".
OK
AT+QCOAPHEAD=0,2                        //Generate message ID randomly, token value is omitted.
OK
AT+QCOAPHEAD=0,3,13940                  //Set the Message ID to 13940; token value is omitted.
OK
AT+QCOAPHEAD=0,4,13940                  //Set the Message ID to 13940, and randomly generate
                                          the token value.
OK
AT+QCOAPHEAD=0,5,13940,4,"02040608"          //Set the Message ID to 13940 and the token
                                             value to "02040608".
OK
```

NOTE

If this command is not used, the module will randomly generate CoAP Message ID and the token value by default.

2.3.6. AT+QCOAPSEND Send CoAP Message

This command sends data to the CoAP server. After the CON data is sent, the result will be automatically provided to the terminal. The CON data state can also be queried by the **AT+QCOAPSEND=<clientID>** through the terminal.

AT+QCOAPSEND Send CoAP Message	
Test Command **AT+QCOAPSEND=?**	Response **+QCOAPSEND: (**range of supported **<clientID>**s**),(**range of supported **<type>**s**),<method/rspcode>,(**list of supported **<opt_bitmask>**s**),(**range of supported **<payload_length>**s**),<data>** **OK**
Write Command **AT+QCOAPSEND=<clientID>**	Response **+QCOAPSEND: <clientID>,<status>** **OK** If there is any error: **ERROR** Or **+CME ERROR: <err>**
Write Command **AT+QCOAPSEND=<clientID>,<type>,<method/rspcode>,<opt_bitmask>** After **>** is returned, input **<data>** to be sent. Tap "**Ctrl**" + "**Z**" to send the data, or tap "**Esc**" to cancel the operation.	Response **>** input the data to be sent **OK** If there is any error: **ERROR** Or **+CME ERROR: <err>**
Write Command **AT+QCOAPSEND=<clientID>,<type>,<method/rspcode>,<opt_bitmask>,<payload_length>** After **>** is returned, input **<data>** to be sent. When the **<payload_length>** is reached, the data will send automatically.	Response If **<payload_length>** is 0: **OK** If **<payload_length>** is not 0: **>** input the data to be sent **OK** If there is any error: **ERROR** Or

	+CME ERROR: <err>
Write Command **AT+QCOAPSEND=<clientID>,<type>,<method/rspcode>,<opt_bitmask>,<payload_length>,<data>**	Response **OK** If there is any error: **ERROR** Or **+CME ERROR: <err>**
Maximum Response Time	300 ms
Characteristics	This command take effect immediately; The configuration is not saved.

Parameter

<clientID>	Integer type. CoAP client identifier. Range: 0–4.	
<type>	Integer type. CoAP message type. For details, refer to *RFC 7252*.	
	0	Confirmable (CON)
	1	Non-confirmable (NON)
	2	Acknowledgement (ACK)
	3	Reset (RST)
<method>	Integer type. CoAP request method. Refer to the *RFC 7252*.	
	1	GET
	2	POST
	3	PUT
	4	DELETE
<rspcode>	Integer type. CoAP response code. Refer to the *RFC 7252*.	
	0	Empty Message
	201	2.01, Created
	202	2.02, Deleted
	203	2.03, Valid
	204	2.04, Changed
	205	2.05, Content
	400	4.00, Bad Request
	401	4.01, Unauthorized
	402	4.02, Bad Option
	403	4.03, Forbidden
	404	4.04, Not Found
	405	4.05, Method Not Allowed
	406	4.06, Not Acceptable
	412	4.12, Precondition Failed
	413	4.13, Request Entity Too Large
	415	4.15, Unsupported Content-Format

	500 5.00, Internal Server Error 501 5.01, Not Implemented 502 5.02, Bad Gateway 503 5.03, Service Unavailable 504 5.04, Gateway Timeout 505 5.05, Proxying Not Supported
<opt_bitmask>	Integer type. CoAP data packet adding options. Currently only 0 is supported, indicating no option is added to the CoAP packet.
<payload_length>	Integer type. Length of data to be sent. Range: 0–1024. Unit: byte.
<data>	String type. Data to be sent. If **>** is returned, the input data is a text string, otherwise, it is in hexadecimal format.
<status>	Integer type. Status of sent CON data. 0 Not sent 1 Sent, waiting for response from the CoAP server 2 Failed sending 3 Timeout 4 Sent successfully 5 Get reset message
<err>	Integer type. Error codes. See ***Chapter 6*** for details.

Example

```
AT+QCOAPSEND=?
+QCOAPSEND: (0-4),(0-3),<method/rspcode>,(0),(0-1024),<data>

OK
AT+QCOAPSEND=0
+QCOAPSEND: 0,0

OK
//Send a CON type GET request to the server.
AT+QCOAPSEND=0,0,1,0,127,"6838363737323530333035363936383902020000000000000010060B8
07B7C89F893C3AE82E2E9527598E16DAB2CDC5D92B65EE24665F21DA270272DF7DC0BEC89D7
DCDFDF33EC4F22C83D927DF1DD6A84D42EE14510951C31967B1EA7CC6E83C00B9CD410E53D
A35F8A76D53DD6CBC9CF0D1246F3F481D7FEE6D0B0000E516"
OK

+QCOAPURC: 0,2,205,26924,133,6,"514CE1000000",12,"0",14,"196607",133,"This is a test server
made with libcoap (see https://libcoap.net)Copyright (C) 2010--2021 Olaf Bergmann
<bergmann@tzi.org> and others"            //Received a response from the server.
AT+QCOAPSEND=0
+QCOAPSEND: 0,4                           //Data was sent successfully.

OK
```

NOTE

1. It is recommended to retrieve the CON data status through **AT+QCOAPSEND=<clientID>** before sending the next CON or NON data.
2. **AT+QCOAPSEND=<clientID>** is only used to query the status of previously sent CON data.
3. URC **+QCOAPURC: <clientID>,<type>,<method/rspcode>,<msgID>,<existence>[,<token_length>,<token>][,<opt_name>,<opt_value>[,...]][,<data_len>,<data>]** is reported upon successful transmission and receiving a response from the server. See ***Chapter 3*** for details.

3 Description of CoAP URC

This provides an overview and description of the CoAP-related URC and its functionalities.

+QCOAPURC Notify TE to Respond to Requests from CoAP Server	
+QCOAPURC: <clientID>,<type>,<method/rspcode>,<msgID>,<existence>[,<token_length>,<token>][,<opt_name>,<opt_value>[,...]][,<data_len>,<data>]	Notify TE to respond to requests from CoAP server or to handle the response from the CoAP server

Parameter

<clientID>	Integer type. CoAP client identifier.
<type>	Integer type. CoAP message type, refer to the *RFC 7252*. 0 Confirmable (CON) 1 Non-confirmable (NON) 2 Acknowledgement (ACK) 3 Reset (RST)
<method>	Integer type. CoAP request method, refer to the *RFC 7252*. 1 GET 2 POST 3 PUT 4 DELETE
<rspcode>	Integer type. CoAP response code, refer to the *RFC 7252*. 0 Empty Message 201 2.01, Created 202 2.02, Deleted 203 2.03, Valid 204 2.04, Changed 205 2.05, Content 400 4.00, Bad Request 401 4.01, Unauthorized 402 4.02, Bad Option 403 4.03, Forbidden 404 4.04, Not Found 405 4.05, Method Not Allowed 406 4.06, Not Acceptable

412	4.12, Precondition Failed
413	4.13, Request Entity Too Large
415	4.15, Unsupported Content-Format
500	5.00, Internal Server Error
501	5.01, Not Implemented
502	5.02, Bad Gateway
503	5.03, Service Unavailable
504	5.04, Gateway Timeout
505	5.05 Proxying Not Supported

<msgID> Integer type. CoAP message ID. Range: 0–65535.

<existence> Integer type. Indicates the presence of a token, data and option counts.

Bit 0	Indicates the presence of data. 1 means there is data; 0 means there is no data.
Bit 1-6	Represent the option counts when converted to decimal.
Bit 7	Indicates the presence of a token. 1 means there is a token; 0 means there is no token.

Example: If **<existence>** is 131, it is converted to binary as 1000 0011, which means that both token and data are present, and the option counts is 1.

<token_length> Integer type. Token length of the CoAP server. Range: 1–8. Unit: byte.

<token> String in Hex. Token values.

<opt_name> Integer type. CoAP option name. See *RFC 7252* and *RFC7959* for more details.

1	If-Match
3	Uri-Host
4	ETag
5	If-None-Match
6	Observe
8	Location-Path
11	Uri-Path
12	Content-Format
14	Max-Age
15	Uri-Query
17	Accept
20	Location-Query
23	Block2
27	Block1
28	Size2
35	Proxy-Uri

<opt_value> String type. CoAP option values. Maximum length: 255 bytes.

If **<opt_name>** is 23 or 27, calculate the value of **<opt_value>** according to the following formula:

<opt_value> = (block_num << 4) | (block_more&0x8) | log2(block_size/16)

If **<opt_name>**=12 or 17, the value of **<opt_value>** is as follows:

"0"	Text-plain
"40"	Application/link-format

	"41" Application/xml "42" Application/octet-stream "47" Application/exi "50" Application/json
<data_len>	Integer type. Data length. Unit: byte.
<data>	String type. Received data. If **<opt_name>**=12 and **<opt_value>**=0/41/50, the data is in string format, otherwise, the data is in hexadecimal format.

4 Example

4.1. Register to IoT Platform Without DTLS

```
AT+CGATT?                                  // Query the service status of the current PS domain.
+CGATT: 1                                  // PS domain is attached.

OK
AT+QCOAPCFG="dtls",0                       // Query DTLS Mode with client ID 0.
+QCOAPCFG: "dtls",0                        // DTLS Mode is off with client ID 0.

OK
AT+QCOAPOPEN=0,"220.180.239.212",7002      //Create a CoAP context with client ID 0.
OK

+QCOAPOPEN: 0,0
//Configure the CoAP message ID as 12345, the token value as "01020304", and the length as 4 bytes.
AT+QCOAPHEAD=0,5,12345,4,"01020304"
OK
//Configure the CoAP option as 11 (Uri-Path), with the option value “rd” and the length of 2 bytes.
AT+QCOAPOPTION=0,0,11,2,"rd"
OK
//Configure the CoAP option as 12(Content-Format), with the option value “40”(Application/link-format),
and the length of 2 bytes.
AT+QCOAPOPTION=0,0,12,2,"40"
OK
//Configure the CoAP option as 15 (Uri-Query), with the option value “lwm2m=1.0&ep=867725030012276
&b=U&lt=900”, and the length of 39 bytes. Use "&" symbol to connect multiple options.
AT+QCOAPOPTION=0,0,15,39,"lwm2m=1.0&ep=867725030012276&b=U&lt=900"
OK
AT+QCOAPSEND=0                             //Query the sending status of CoAP CON data.
+QCOAPSEND: 0,0                            //Data is not sent.

OK
AT+QCOAPSEND=0,0,2,0                       //Send a CON type POST request to the COAP server.
>
</>;rt="oma.lwm2m";ct=11543,</1/0>,</3/0>,</4/0>,</5/0>,</19/0>,</19/1>
```

```
OK
//URC below is returned after the request is received from the server, with <rspcode> as 201, <msgID>
as 12345, <existence> as 132, <token_length> as 4 bytes, <token> as "01020304", <opt_name> as 8
and 8 corresponding to " rd " and "867725030012276" of <opt_value> respectively.
+QCOAPURC: 0,2,201,12345,132,4,"01020304",8,"rd",8," 867725030012276"

//After receiving the request from the server, the following URC is returned with <Type> as 0,<method>
as 1,  <msgID> as 21152, <existence> as 134, <token_length> as 8 bytes, <token> as
"E8532E3363BE54B9", <opt_name> as 6, <opt_value> as 0, <opt_name> as 11, <opt_value> as 3,
<opt_name> as 11, <opt_value> as 0.
+QCOAPURC: 0,0,1,21152,134,8,"E8532E3363BE54B9",6,"0",11,"3",11,"0"

AT+QCOAPSEND=0
+QCOAPSEND: 0,4                                  //CON data was sent successfully.

OK
//According to the last requested URC content, configure CoAP message ID as 21152, token length as 8
bytes, and token value as "E8532E3363BE54B9".
AT+QCOAPHEAD=0,5,21152,8,"E8532E3363BE54B9"
OK

// Send a 205 response code and "AAAA" data to the IoT platform using the ACK type.
AT+QCOAPSEND=0,2,205,0
>
AAAA
OK

AT+QCOAPCLOSE=0                                  //Delete the CoAP context.
OK

+QCOAPCLOSE: 0,0                                 //The CoAP context is deleted successfully.
```

4.2. Register to IoT Platform with DTLS

```
AT+CGATT?                                        //Query the service status of the current PS domain.
+CGATT: 1                                        // PS domain is attached.

OK
AT+QCOAPCFG="dtls",0,1                           //Enable DTLS mode for CoAP client with ID 0.
OK
//Configure PSK, with <identity> "869154040004132", <key> "67514b6c5a45334e31576a6d5733387
a".
```

```
AT+QCOAPCFG="psk",0,"869154040004132","67514b6c5a45334e31576a6d5733387a"
OK
AT+QCOAPOPEN=0,"220.180.239.212",7002    //Create a CoAP context with a client ID of 0.
OK

+QCOAPOPEN: 0,0
//Configure the CoAP message ID as 12345, the token value as "01020304", and the length as 4 bytes.
AT+QCOAPHEAD=0,5,12345,4,"01020304"
OK
//Configure the CoAP option as 11 (Uri-Path), with the option value “rd”, and the length of 2 bytes.
AT+QCOAPOPTION=0,0,11,2,"rd"
OK
//Configure the CoAP option as 12(Content-Format), with the option value “40”(Application/link-format),
and the length of 2 bytes.
AT+QCOAPOPTION=0,0,12,2,"40"
OK
//Configure the CoAP option as 15 (Uri-Query), with the option value
“lwm2m=1.0&ep=869619050023990&b=U&lt=1600”, and the length of 40 bytes. Use "&" symbol to
connect multiple options.
AT+QCOAPOPTION=0,0,15,40,"lwm2m=1.0&ep=869154040004132&b=U&lt=1600"
OK
AT+QCOAPSEND=0                           //Query the sending status of CoAP CON data
+QCOAPSEND: 0,0                          //Data is not sent.

OK
AT+QCOAPSEND=0,0,2,0                     //Send a CON type POST request to the COAP server.
>
</>;rt="oma.lwm2m";ct=11543,</1/0>,</3/0>,</4/0>,</5/0>,</19/0>,</19/1>
OK
//URC below is returned after the request is received from the server, with <rspcode> as 201, <msgID>
as 12345, <existence> as 132, <token_length> as 4 bytes, <token> as "01020304", <opt_name> as 8
and 8 corresponding to " rd " and "869154040004132" of <opt_value> respectively.
+QCOAPURC: 0,2,201,12345,132,4,"01020304",8,"rd",8,"869154040004132"

// After receiving the request from the server, the following URC is received with <Type> as 0,<method>
as 1, <msgID> as 41128, <existence> as 134, <token_length> as 8 bytes, <token> as
“E8532D1363BE54B9”, <opt_name> as 6, <opt_value> as 0, <opt_name> as 11, <opt_value> as 3,
<opt_name> as 11, <opt_value> as 0.
+QCOAPURC: 0,0,1,41128,134,8,"E8532D1363BE54B9",6,"0",11,"3",11,"0"

AT+QCOAPSEND=0
+QCOAPSEND: 0,4                          //CON data was sent successfully.

OK
```

```
//According to the last requested URC content, configure CoAP message ID as 41128, token length as 8 bytes, and token value as “E8532D1363BE54B9".
AT+QCOAPHEAD=0,5,41128,8,"E8532D1363BE54B9"
OK

// Send a 205 response code and "AAAA" data to the IoT platform using the ACK type.
AT+QCOAPSEND=0,2,205,0
>
AAAA
OK

OK
AT+QCOAPCLOSE=0                                    //Delete the CoAP context.
OK

+QCOAPCLOSE: 0,0                                   //The CoAP context is deleted successfully.
```

5 Summary of Result Codes

The following table lists some of the general result codes.

Table 2: Description of <result> Codes

Code of <result>	Meaning
0	Operation successful
-1	Invalid parameter
-2	Operation in processing
-3	Operation not allowed
-4	Network failure
-5	DNS error
-6	Data call activating
-7	Socket connection failure
-8	Out of memory error
-9	DTLS handshaking failure
-10	CoAP client identifier occupied
-11	Data sending failure

6 Summary of Error Codes

This chapter summarizes the error codes related to BC660K-GL and BC950K-GL modules.

The error codes listed in the following two tables are compliant with the 3GPP specifications. Please refer to *3GPP TS 27.007 V13.5.0*, sub-clause 9.2 for all possible **<err>** values.

Table 3: General Errors (27.007)

Code of <err>	Description
3	Operation not allowed
4	Operation not supported
23	Memory failure
30	No network service
50	Incorrect parameters
51	Command implemented but currently disabled
52	Command aborted by user
159	Uplink busy/flow control

NOTE

AT+CMEE=<n> command disables (**<n>**=0) or enables (**<n>**=1) the use of final result code "**+CME ERROR:<err>**". If **<n>**=1 or an error occurs, **+CME ERROR: <err>** will be returned. If **<n>**=0 or an error occurs, **ERROR** will be returned, see ***document [1]*** for details.

7 Appendix References

Table 4: Related Documents

Document Name
[1] Quectel_BC660K-GL&BC950K-GL_AT_Commands_Manual

Table 5: Terms and Abbreviations

Abbreviation	Description
3GPP	3rd Generation Partnership Project
ACK	Acknowledgement
CoAP	Constrained Application Protocol
DNS	Domain Name Server
DTLS	Datagram Transport Layer Security
ID	Identification
IoT	Internet of Things
IP	Internet Protocol
ME	Mobile Equipment
OPT	Option
PDU	Protocol Data Unit
PIN	Personal Identification Number
PS	Packet Switch
PUK	PIN Unlock Key
RST	Reset
SMS	Short Message Service

SMSC	Short Message Service Center
TA	Terminal Adapter
TE	Terminal Equipment
URC	Unsolicited Result Code
USIM	Universal Subscriber Identity Module