Below is a complete list of BleuIO AT commands, along with descriptions and usage examples.

⬇️ Download AT Commands in PDF

AT

- *Basic AT-Command.*

Command	Syntax
AT	AT

Example:

```
AT
OK
```

```
AT
{"C":1,"cmd":"AT"}
{"A":1,"err":0,"errMsg":"ok"}
{"E":1,"nol":3}
```

ATA

- *Shows/hides ASCII values from notification/indication/read responses. ATA0 hides the ASCII values, ATA1 shows the ASCII values. ATA reads current ATA settings.*

Command	Syntax
ATA	ATA
	ATA0
	ATA1

Example of a notification response with and without ASCII value shown With ASCII on:

```
handle_evt_gattc_notification: conn_idx=0000 handle=0039 length=5


Value received: #Eg‰
Hex: 0x0123456789
Size: 5
```

```
{777:"0000","001f":{"ascii":"{638:
{T:"00DC",H:"00EA",CO2:"01AE",IAQ:"0075",PPM:"00C8"}}","len":59}}
{777:"0000","001f":
{"hex":"0x7B3633383A7B543A2230304443222C483A2230304541222C434F323A22303141452
22C4941513A2230303735222C50504D3A2230304338227D7D00","len":59}}
```

With ASCII off:

```
handle_evt_gattc_notification: conn_idx=0000 handle=0039 length=5


Hex: 0x0123456789
Size: 5
```

```
{"777": "0000","001f": {"hex":
"0x7B3633383A7B543A2230304443222C483A2230304541222C434F323A2230314145222C4941
513A2230303735222C50504D3A2230304338227D7D00","len": 59}}
```

Example:

```
ATA
Show Ascii On!
ATA0
Show Ascii Off!
ATA1
Show Ascii On!
```

```
ATA
{"C":0,"cmd":"ATA"}
{"A":0,"err":0,"errMsg":"ok"}
{"R":0,"showAscii":true}
{"E":0,"nol":4}
ATA0
{"C":1,"cmd":"ATA0"}
{"A":1,"err":0,"errMsg":"ok"}
{"R":1,"showAscii":false}
{"E":1,"nol":4}
ATA1
{"C":2,"cmd":"ATA1"}
{"A":2,"err":0,"errMsg":"ok"}
{"R":2,"showAscii":true}
{"E":2,"nol":4}
```

ATAR

- *Enables/disables auto reconnect (on by default). ATAR0 turns auto reconnect off, ATAR1 turns auto reconnect on, ATAR reads current ATAR settings. Reconnect will be attempted only if disconnect reason is 3E (Connection failed to be established). Max 6 retries.*

Command	Syntax
ATAR	ATAR
	ATAR0
	ATAR1

Example:

```
ATAR
AUTO RECONNECT ON
ATAR0
AUTO RECONNECT OFF
ATAR1
AUTO RECONNECT ON
```

```
ATAR
{"C":89,"cmd":"ATAR"}
{"A":89,"err":0,"errMsg":"ok"}
{"R":89,"auto_reconnect":true}
{"E":89,"nol":4}
ATAR0
{"C":90,"cmd":"ATAR0"}
{"A":90,"err":0,"errMsg":"ok"}
{"R":90,"auto_reconnect":false}
{"E":90,"nol":4}
ATAR1
{"C":91,"cmd":"ATAR1"}
{"A":91,"err":0,"errMsg":"ok"}
{"R":91,"auto_reconnect":true}
{"E":91,"nol":4}
```

ATB Pro

- *Put dongle in bootloader mode.*

Command	Syntax
ATB	

> Type 'QUIT' to manually exit bootloader mode.

Example:

```
ATB
OK
Bootloader on.
>
```

```
ATB
{"C":0,"cmd":"ATB"}
{"A":0,"err":0,"errMsg":"ok"}
{"R":0,"action":"bootloader on"}
{"E":0,"nol":4}
>
```

ATASPS

- *Toggle between ASCII and hex responses received from SPS. ATASPS reads current ATASPS settings, ATASPS0 shows hex values, ATASPS1 shows ASCII. ASCII is on by default.*

Command	Syntax
ATASPS	ATASPS
	ATASPS0
	ATASPS1

Example of a SPS response with HEX shown and with ASCII shown

With ASCII shown:

```
# On Server:
[Received]: HELLO

# On Client:
handle_evt_gattc_notification: conn_idx=0000 handle=000d length=6
HELLO
```

```
# Client
{777:"0000","SPS":{"ascii":"TEST","len":5}}
# Server
{279:"0000","SPS":{"ascii":"TEST","len":5}}
```

With Hex shown:

```
# On Server:
(conn_idx=0000 len=6)
```

```
48454C4C4F00


# On Client:
(conn_idx=0000 length=6)48454C4C4F00
```

```
# Client
{777:"0000","SPS":{"hex":"0x5445535400","len":5}}
# Server
{279:"0000","SPS":{"hex":"0x5445535400","len":5}}
```

Example:

```
ATASPS
Show SPS Ascii On!
ATASPS0
Show SPS Ascii Off!
ATASPS1
Show SPS Ascii On!
```

```
ATASPS
{"C":92,"cmd":"ATASPS"}
{"A":92,"err":0,"errMsg":"ok"}
{"R":92,"showSPSAscii":true}
{"E":92,"nol":4}
ATASPS0
{"C":93,"cmd":"ATASPS0"}
{"A":93,"err":0,"errMsg":"ok"}
{"R":93,"showSPSAscii":false}
{"E":93,"nol":4}
ATASPS1
{"C":94,"cmd":"ATASPS1"}
{"A":94,"err":0,"errMsg":"ok"}
{"R":94,"showSPSAscii":true}
{"E":94,"nol":4}
```

ATASSN

- *Turns on/off showing device names, if present, in scan results from AT+FINDSCANDATA and AT+SCANTARGET scans. (Off per default).*

Command	Syntax
ATASSN	ATASSN
ATASSN	ATASSN0

Command	Syntax
	ATASSN1

AT+FINDSCANDATA results with ATASSN on

```
[C1:12:32:45:65:54] (HibouAIR) Device Data [RESP]: 09094869626F75414952
```

```
{"SF":3,"addr":"C1:12:32:45:65:54","type":4,"data":"09094869626F75414952","na
me":"HibouAIR"}
```

AT+SCANTARGET results with ATASSN on

```
[C1:12:32:45:65:54] (HibouAIR) Device Data [RESP]: 09094869626F75414952
```

```
{"ST":4,"addr":"C1:12:32:45:65:54","type":4,"data":"09094869626F75414952","na
me":"HibouAIR"}
```

Example:

```
ATASSN
Always Show Scan Name Off!
ATASSN1
Always Show Scan Name On!
ATASSN0
Always Show Scan Name Off!
```

```
ATASSN
{"C":96,"cmd":"ATASSN"}
{"A":96,"err":0,"errMsg":"ok"}
{"R":96,"assn":false}
{"E":96,"nol":4}
ATASSN1
{"C":97,"cmd":"ATASSN1"}
{"A":97,"err":0,"errMsg":"ok"}
{"R":97,"assn":true}
{"E":97,"nol":4}
ATASSN0
{"C":98,"cmd":"ATASSN0"}
{"A":98,"err":0,"errMsg":"ok"}
{"R":98,"assn":false}
{"E":98,"nol":4}
```

ATASSM

- *Turns on/off showing Manufacturing Specific ID (Company ID), if present, in scan results from AT+GAPSCAN, AT+FINDSCANDATA and AT+SCANTARGET scans. (Off per default).*

Command	Syntax
ATASSM	ATASSM
	ATASSM0
	ATASSM1

AT+GAPSCAN results with ATASSM on

```
[36] Device: [1]1B:A3:EE:2F:8F:18  RSSI: -67 [MFSID: 004C]
```

```
{"S":4,"rssi":-67,"mfsid":"004C","addr":"[1]1B:A3:EE:2F:8F:18"}
```

AT+FINDSCANDATA results with ATASSM on

```
[68:0E:8E:2E:F2:8A] [MFSID: 004C] Device Data [ADV]:
02011A020A0C0CFF4C0010011233445566778 8
```

```
{"SF":5,"addr":"68:0E:8E:2E:F2:8A","type":0,"mfsid":"004C","data":"02011A020A
0C0CFF4C00100112334455667788"}
```

AT+SCANTARGET results with ATASSM on

```
[68:0E:8E:2E:F2:8A] [MFSID: 004C] Device Data [ADV]:
02011A020A0C0CFF4C00100112334455667788
```

```
{"ST":6,"addr":"68:0E:8E:2E:F2:8A","type":0,"mfsid":"004C","data":"02011A020A
0C0CFF4C00100112334455667788"}
```

Example:

```
ATASSM
Always Show Scan MFSID Off!
ATASSM1
Always Show Scan MFSID On!
ATASSM0
Always Show Scan MFSID Off!
```

```
ATASSM
{"C":99,"cmd":"ATASSM"}
{"A":99,"err":0,"errMsg":"ok"}
{"R":99,"assm":false}
{"E":99,"nol":4}
ATASSM1
{"C":100,"cmd":"ATASSM1"}
{"A":100,"err":0,"errMsg":"ok"}
{"R":100,"assm":true}
{"E":100,"nol":4}
ATASSM0
{"C":101,"cmd":"ATASSM0"}
{"A":101,"err":0,"errMsg":"ok"}
{"R":101,"assm":false}
{"E":101,"nol":4}
```

ATDS

- *Turns auto discovery of services on/off when connecting. ATDS0 off, ATDS1 on, ATDS reads back current ATDS settings. On by default. This command can be used in both central and peripheral role.*

Command	Syntax
ATDS	ATDS
	ATDS0
	ATDS1

Example:

```
ATDS
Auto discover Services On!
ATDS0
Auto discover Services Off!
ATDS1
Auto discover Services On!
```

```
ATDS
{"C":105,"cmd":"ATDS"}
{"A":105,"err":0,"errMsg":"ok"}
```

```
{"R":105,"autoDiscoverSvc":true}
{"E":105,"nol":4}
ATDS0
{"C":106,"cmd":"ATDS0"}
{"A":106,"err":0,"errMsg":"ok"}
{"R":106,"autoDiscoverSvc":false}
{"E":106,"nol":4}
ATDS1
{"C":107,"cmd":"ATDS1"}
{"A":107,"err":0,"errMsg":"ok"}
{"R":107,"autoDiscoverSvc":true}
{"E":107,"nol":4}
```

ATE

- *Turns echo on/off or reads back current echo settings. (On per default).*

Command	Syntax
ATE	ATE
	ATE0
	ATE1

Example:

```
ATE0
ECHO OFF
```

```
ATE
{"C":108,"cmd":"ATE"}
{"A":108,"err":0,"errMsg":"ok"}
{"R":108,"showEcho":true}
{"E":108,"nol":4}
ATE0
{"C":109,"cmd":"ATE0"}
{"A":109,"err":0,"errMsg":"ok"}
{"R":109,"showEcho":false}
{"E":109,"nol":4}
{"C":110,"cmd":"ATE1"}
{"A":110,"err":0,"errMsg":"ok"}
{"R":110,"showEcho":true}
{"E":110,"nol":4}
```

ATES Pro

- *Toggles showing extended scan results on/off. Off by default.*

Command	Syntax
ATES	ATES=0
	ATES=1

Example:

```
ATES=0
OK
ATES=1
OK
```

```
ATES=0
{"C":12,"cmd":"ATES=0"}
{"A":12,"err":0,"errMsg":"ok"}
{"E":12,"nol":3}
ATES=1
{"C":13,"cmd":"ATES=1"}
{"A":13,"err":0,"errMsg":"ok"}
{"E":13,"nol":3}
```

ATEW

- *Turn WRITTEN DATA echo on/off after GATTCWRITE commands. (On per default). ATEW reads current ATEW settings.*

Command	Syntax
ATEW	ATEW
	ATEW0
	ATEW1

Commands affected
AT+GATTCWRITE
AT+GATTCWRITEB
AT+GATTCWRITEWR
AT+GATTCWRITEWRB

Instead of response for AT+GATTCWRITE & AT+GATTCWRITEWR:

```
DATA WRITTEN: <data>
```

or for AT+GATTCWRITEB & AT+GATTCWRITEWRB:

```
DATA WRITTEN: <data>
Size: <size>
```

It will instead acknowledge that the command has been accepted with *"DATA WRITTEN!"*.
An example using AT+GATTCWRITEWRB with GATTC WRITE ECHO OFF (ATEW0):

```
AT+GATTCWRITEWRB=001B 0101
DATA WRITTEN!
```

Example:

```
ATEW
GATTC WRITE ECHO ON
ATEW0
GATTC WRITE ECHO OFF
ATEW1
GATTC WRITE ECHO ON
```

```
ATEW
{"C":111,"cmd":"ATEW"}
{"A":111,"err":0,"errMsg":"ok"}
{"R":111,"showGattcWriteEcho":true}
{"E":111,"nol":4}
ATEW0
{"C":112,"cmd":"ATEW0"}
{"A":112,"err":0,"errMsg":"ok"}
{"R":112,"showGattcWriteEcho":false}
{"E":112,"nol":4}
ATEW1
{"C":113,"cmd":"ATEW1"}
{"A":113,"err":0,"errMsg":"ok"}
{"R":113,"showGattcWriteEcho":true}
{"E":113,"nol":4}
```

ATI

- *Device information query. Returns firmware version, hardware type and unique organization identifier.*

Command	Syntax

Command	Syntax
ATI	ATI

Example:

```
ATI
Smart Sensor Devices
DA14683
Smart USB Dongle
Firmware Version: 1.1.0

Peripheral role

Not Connected

Not Advertising
```

```
ATI
{"C":12,"cmd":"ATI"}
{"A":12,"err":0,"errMsg":"ok"}
{"R":12,"dev":"Smart Sensor Devices","hw":"DA14683","name":"BleuIO"}
{"R":12,"fwVer":"2.1.5b","gap_role:"peripheral"}
{"R":12,"connected":false,"advertising":false}
{"E":12,"nol":6}
```

ATR

- *Trigger platform reset.*

Command	Syntax
ATR	ATR

Example:

```
ATR
OK
```

```
ATR
{"C":14,"cmd":"ATR"}
{"A":14,"err":0,"errMsg":"ok"}
{"R":14,"action":"restart"}
{"E":14,"nol":4}
```

ATV

- *Turn **VERBOSE** mode on/off. (Off per default).*

Command	Syntax
ATV	ATV0
	ATV1

Changes the format for all the BleuIO's output. See the **VERBOSE** Mode information page for more details.
Example:

```
ATV1
VERBOSE ON
```

```
ATV0
{"C":13,"cmd":"ATV0"}
{"A":13,"err":0,"errMsg":"ok"}
{"R":13,"verboseOn":false}
{"E":13,"nol":4}
```

```
ATV1
VERBOSE ON
```

AT+ADVEXTPARAM Pro

- *Sets advertising parameters for extended advertising.*

Command	Syntax
AT+ADVEXTPARAM	AT+ADVEXTPARAM=handle=parameters
	AT+ADVEXTPARAM

Order	Parameters	Input Type	Span
1	Advertising handle	Unsigned Integer	0 - 1
2	GAP discoverability mode	Unsigned Integer	0 - 3
3	Advertising Properties	Unsigned Integer or String	1-127 or 'C' - 'CSDHLAT'
4	Min Adv Interval	Unsigned Integer	32 - 65535
5	Max Adv Interval	Unsigned Integer	32 - 65535
6	Adv Channel Map	Unsigned Integer	1 - 7
7	Local Addr Type	Unsigned Integer	0 - 1

Order	Parameters	Input Type	Span
8	Filt Policy	Unsigned Integer	0 - 1
9	TX Power	Signed Integer	-52 - 20
10	Primary PHY	Unsigned Integer	0 - 1
11	Secondary Max Event Skip	Unsigned Integer	0 - 255
12	Secondary PHY	Unsigned Integer	0 - 2
13	SID	Unsigned Integer	0 - 15
14	Enable Scan Req Noti	Unsigned Integer	0 - 1
15	(Optional)Peer Addr Type	String	'public' or 'private'
16	(Optional)Peer Address	Address String	'00:00:00:00:00:00' - 'FF:FF:FF:FF:FF:FF'

Handle

Parameters Value	Meaning
0 - 1	Which Advertisings set is used

GAP discoverability mode

Parameters Value	Meaning
0	Non-Discoverable mode
1	General-Discoverable mode
2	Limited-Discoverable mode
3	Broadcaster mode

Advertising Properties

Parameters Value	Meaning
(1 << 0)	(C) Advertising is connectable
(1 << 1)	(S) Advertising is scannable
(1 << 2)	(D) Advertising is directed
(1 << 3)	(H) Advertising is high duty connectable (<= 3.75 ms advertising interval)
(1 << 4)	(L) Advertising is legacy
(1 << 5)	(A) Advertising is anonymous (Non-Connectable, Non-Scannable, EXT ADV only)

Parameters Value	Meaning
(1 << 6)	(T) Include TxPower in the extended header of at least one advertising PDU (EXT ADV only)

Several properties can be combined.
For example: (1 << 0) | (1 << 1) | (1 << 2) = 7

Min & Max Adv Interval

Parameters Value	Meaning
32 - 65535	Advertising interval in miliseconds

Adv Channel Map

Parameters Value	Meaning
(1 << 0)	Advertising channel 37 is used
(1 << 1)	Advertising channel 38 is used
(1 << 2)	Advertising channel 39 is used

Several Channel Maps can be combined.
For example: (1 << 0) | (1 << 1) | (1 << 2) = 7

Local Addr Type

Parameters Value	Meaning
0	Public Address Type
1	Private Address Type

Filt Policy

Parameters Value	Meaning
0	Allow all scan and connect requests
1	Allow all connect requests. Scan requests are accepted only from whitelist devices.
2	Allow all scan requests. Connect requests are accepted only from whitelist devices.
3	Allow scan and connect requests only from whitelist devices

TX Power

Parameters Value	Meaning
-52 - 20	TX Power in dBm

Parameters Value	Meaning
127	No preference

Primary PHY & Secondary PHY

Parameters Value	Meaning
0	No PHY
1	Bit rate of 1 megabit per second (Mb/s)
2	Allow all scan requests. Connect requests are accepted only from whitelist devices.
3	Bit rate of 2 megabit per second (Mb/s) (Only for Secondary PHY)

Secondary Max Event Skip

Parameters Value	Meaning
0	No events shall be skipped (EXT ADV only)
1 - 255	Maximum advertising events the Controller can skip (EXT ADV only)

SID

Parameters Value	Meaning
0 - 15	Advertising SID (EXT ADV only)

Enable Scan Req Noti [Not Implemented]

Parameters Value	Meaning
0	Disable scan request notifications on scannable advertising
1	Enable scan request notifications on scannable advertising

Example:

```
AT+ADVEXTPARAM=0=1=19=160=320=7=0=0=0=1=0=1=0=0
OK

AT+ADVEXTPARAM
OK
HANDLE: 1
mode: 1
properties: 19 (CSL)
primary_intv_min: 160
primary_intv_max: 320
primary_channel_map: 7
local_addr_type: 0
filt_policy: 0
```

```
tx_power: 0
primary_phy: 1
secondary_max_event_skip: 0
secondary_phy: 1
sid: 0
scan_req_notif_enable: 0
peer_address: public, 00:00:00:00:00:00
```

```
AT+ADVEXTPARAM=0=1=19=160=320=7=0=0=0=1=0=1=0=0
{"C":1,"cmd":"AT+ADVEXTPARAM=0=1=19=160=320=7=0=0=0=1=0=1=0=0"}
{"A":1,"err":0,"errMsg":"ok"}
{"E":1,"nol":3}
AT+ADVEXTPARAM
{"C":4,"cmd":"AT+ADVEXTPARAM"}
{"A":4,"err":0,"errMsg":"ok"}
{"R":4,"info":"ext adv params","handle":01}
{"R":4,"mode":1,"prop":"19
(CSL)","p_intv_min":160,"p_intv_max":320,"p_chnl_map":7,"l_addr_t":0,"filt_po
l":0}
{"R":4,"tx_power":0,"p_phy":1,"s_max_event_skip":0,"s_phy":1,"sid":0,"scan_re
q_notif_enable":0,"peer_address":"public, 00:00:00:00:00:00"}
{"E":4,"nol":6}
```

AT+ADVEXTSTART Pro

- *Sets extended advertising data and/or scan response data and starts extended advertising. AT+ADVEXTPARAM need to be run first to create an advertising set.*

Command	Syntax
AT+ADVEXTSTART	AT+ADVEXTSTART=***handle***=***Adv Data***=***Scan Resp Data***

Handle

Parameters Value	Meaning
0 - 1	Which Advertisings set is used

Adv & Scan Response Data

Parameters Value	Meaning
Hex string format: xxxxxxxxxx	Data

Max Adv Data Length

Length (B)	Advertising Properties Conditions
1650	Not Legacy, Not Connectable and Not Scannable

Length (B)	Advertising Properties Conditions
242	Not Legacy, Not Scannable and Not Directed
236	Not Legacy and Not Scannable
28	Legacy, Connectable, Scannable and Not Directed
31	Legacy, Not Connectable and Not Directed

> In all other cases, ***Adv Data*** should be set to -

Max Scan Resp Data Length

Length (B)	Advertising Properties Conditions
1650	Scannable, Not Legacy, Not Connectable
28	Scannable, Legacy, Connectable and Not Directed
31	Scannable, Legacy, Not Connectable and Not Directed

> In all other cases, ***Scan Resp Data*** should be set to -

Example:

```
AT+ADVEXTSTART=0=16094469616c6f672041445620657874656e73696f6e7316ff4469616c6f
672041445620657874656e73696f6e73=-
Advertising Handle: 0 Adv Data Len: 46 Scan Resp Len: 0 Duration: 0 Max Adv
Evts: 0 Data Fragment Pref: 0

ADVERTISING...
```

```
AT+ADVEXTSTART=0=16094469616c6f672041445620657874656e73696f6e7316ff4469616c6f
672041445620657874656e73696f6e73=-
{"C":6,"cmd":"AT+ADVEXTSTART=0=16094469616c6f672041445620657874656e73696f6e73
16ff4469616c6f672041445620657874656e73696f6e73=-"}
{"A":6,"err":0,"errMsg":"ok"}
{"R":6,"evt":{"action":"advertising"}}
{"R":6,"handle":0,"advDataLen":46,"rspDataLen":0}
{"R":6,"duration":0,"maxAdvEvents":0,"dataFragmentPref":0}
{"E":6,"nol":6}
```

AT+ADVEXTUPD Pro

- *Sets extended advertising data and/or scan response data when advertising. AT+ADVEXTPARAM need to be run first to create an advertising set.*

Command	Syntax
AT+ADVEXTUPD	AT+ADVEXTUPD=***handle***=***Adv Data***=***Scan Resp Data***

Handle

Parameters Value	Meaning
0 - 1	Which Advertisings set is used

Adv & Scan Response Data

Parameters Value	Meaning
Hex string format: xxxxxxxxxx	Data

Max Adv Data Length

Length (B)	Advertising Properties Conditions
1650	Not Legacy, Not Connectable and Not Scannable
242	Not Legacy, Not Scannable and Not Directed
236	Not Legacy and Not Scannable
28	Legacy, Connectable, Scannable and Not Directed
31	Legacy, Not Connectable and Not Directed

> In all other cases, ***Adv Data*** should be set to -

Max Scan Resp Data Length

Length (B)	Advertising Properties Conditions
1650	Scannable, Not Legacy, Not Connectable
28	Scannable, Legacy, Connectable and Not Directed
31	Scannable, Legacy, Not Connectable and Not Directed

> In all other cases, ***Scan Resp Data*** should be set to -

Example:

```
AT+ADVEXTUPD=0=16094469616c6f67204144562065787465 6e73696f6e7316ff4469616c6f67
2041445620657874656e73696f8383=-
OK
```

```
AT+ADVEXTUPD=0=16094469616c6f6720414456206578746 56e73696f6e7316ff4469616c6f67
2041445620657874656e73696f8383=-
{"C":8,"cmd":"AT+ADVEXTUPD=0=16094469616c6f67204144562065787465 6e73696f6e7316
ff4469616c6f672041445620657874656e73696f8383=-"}
{"A":8,"err":0,"errMsg":"ok"}
{"E":8,"nol":3}
```

AT+ADVDATA

- _Sets or queries the advertising data. Data must be provided as a hex string. The content takes effect only while advertising is active (after AT+ADVSTART).
  If you want to send several data types separate them with a space. _

Command	Syntax
AT+ADVDATA	AT+ADVDATA
	AT+ADVDATA=***xx:xx:xx:xx:xx:xx***
	AT+ADVDATA=***(xx:xx:xx:xx) (xx:xx:xx:xx:xx)***

Max length of legacy advertising data is 31 bytes.
The first 3 bytes are reserved for the 'connectable' flag (0x02 0x01 0x06) when using connectable advertising modes (all modes except 'Non-connectable mode'). This will be inserted automatically at the beginning of the advertising data when starting advertising with AT+ADVSTART unless choosing 'Non-connectable mode'.

Example:

```
AT+ADVDATA=04:09:43:41:54
OK

ADVERTISING DATA: 0409434154
```

```
AT+ADVDATA
{"C":0,"cmd":"AT+ADVDATA"}
{"A":0,"err":0,"errMsg":"ok"}
{"R":0,"adv":"03FF5B071107B75C49D204A34071A0B535853EB08307000000000000000000"
}
{"E":0,"nol":4}
```

```
AT+ADVDATA=04:09:43:41:54
{"C":1,"cmd":"AT+ADVDATA=04:09:43:41:54"}
{"A":1,"err":0,"errMsg":"ok"}
{"R":1,"adv":"04094341540000000000000000000000000000000000000000000000000000"
}
{"E":1,"nol":4}
```

AT+ADVDATAI

- *Sets advertising data in a way that lets it be used as an iBeacon.*

Command	Syntax

Command	Syntax
AT+ADVDATAI	AT+ADVDATAI=***(UUID)(MAJOR)(MINOR)(TX)***

Example:

```
AT+ADVDATAI=5f2dd896-b886-4549-ae01-e41acd7a354a0203010400
OK
iBeacon created with uuid: 5f2dd896-b886-4549-ae01-e41acd7a354a
```

```
AT+ADVDATAI=5f2dd896-b886-4549-ae01-e41acd7a354a0203010400
{"C":2,"cmd":"AT+ADVDATAI=5f2dd896-b886-4549-ae01-e41acd7a354a0203010400"}
{"A":2,"err":0,"errMsg":"ok"}
{"R":2,"uuid":"5f2dd896-b886-4549-ae01-e41acd7a354a"}
{"E":2,"nol":4}
```

AT+ADVSTART

- *Starts advertising. Advertising interval can optionally be specified in milliseconds (100 to 3000ms). Time_ms controlls for how long the dongle will advertise. Set to 0 for unlimited. Returns ERROR if advertising is already active or if the device is in central role.*

Command	Syntax
AT+ADVSTART	AT+ADVSTART
	AT+ADVSTART=***(mode);(intv_min);(intv_max);(time_ms);***

Supported modes:

Code	Mode
0:	Non-connectable mode
1:	Undirected mode
2:	Directed mode
3:	Directed Low Duty Cycle mode

Default settings are:

- Mode: GAP_CONN_MODE_UNDIRECTED
- intv_min: 1100
- intv_max: 1100
- time_ms: 0

The 'connectable' flag (0x02 0x01 0x06) will be inserted automatically at the beginning of the advertising data when starting advertising with AT+ADVSTART unless choosing 'Non-connectable mode'.

Example:

```
AT+ADVSTART=1;200;300;20;
Advertising type: GAP_CONN_MODE_UNDIRECTED Advertising interval minimum: 200
maximum: 300

ADVERTISING...
```

```
{"C":3,"cmd":"AT+ADVSTART"}
{"A":3,"err":0,"errMsg":"ok"}
{"R":3,"action":"advertising"}
{"R":3,"advType":"GAP_CONN_MODE_UNDIRECTED","intvMin":"1100","intvMax":"1100"
}
{"E":3,"nol":5}
```

```
AT+ADVSTART=1;200;300;20;
{"C":16,"cmd":"AT+ADVSTART=1;200;300;20;"}
{"A":16,"err":0,"errMsg":"ok"}
{"R":16,"action":"advertising"}
{"R":16,"advType":"GAP_CONN_MODE_UNDIRECTED","intvMin":"200","intvMax":"300"}
{"E":16,"nol":5}
```

AT+ADVSTOP

- *Stops advertising. Returns ERROR if not already advertising.*

Command	Syntax
AT+ADVSTOP	AT+ADVSTOP

Example:

```
AT+ADVSTOP
STOPPING ADVERTISING...

ADVERTISING STOPPED.
```

```
{"C":4,"cmd":"AT+ADVSTOP"}
{"A":4,"err":0,"errMsg":"ok"}
{"R":4,"info":"stopping advertising"}
{"E":4,"nol":4}
{260:"FFFF","action":"advertising stopped"} # <--- Event
```

AT+ADVRESP

- *Sets or queries scan response data. Data must be provided as hex string. The content takes effect only while advertising is active (after AT+ADVSTART).*

Command	Syntax
AT+ADVRESP	
	AT+ADVRESP=***(xx:xx:xx:xx:xx:xx:xx)***

Example:

```
AT+ADVRESP=04:09:43:41:54
OK

RESPONSE DATA: 0409434154
```

```
{"C":5,"cmd":"AT+ADVRESP"}
{"A":5,"err":0,"errMsg":"ok"}
{"R":5,"rsp":"1109426C6575494F000000000000000000000000000000000000000000000000"
}
{"E":5,"nol":4}
```

```
AT+ADVRESP=04:09:43:41:54
{"C":6,"cmd":"AT+ADVRESP=04:09:43:41:54"}
{"A":6,"err":0,"errMsg":"ok"}
{"R":6,"adv":"0409434154"}
{"E":6,"nol":4}
```

AT+AUTOEXEC

- *Sets or displays up to 20 commands that will be run upon the BleuIO starting up.*

Command	Syntax
AT+AUTOEXEC	AT+AUTOEXEC
	or
	AT+AUTOEXEC=**<cmd>**

Max command lenght is currently set at 255 characters.

If the command added in the auto execute (AUTOEXEC) list caused an error that resulted in an unexpected reboot (or if adding the reset command ATR) all commands in the list will be removed to prevent an never ending loop of resets. The first time querying listed commands after such an error will display that all commands have been cleared due to an error:

```
AT+AUTOEXEC
OK

AUTOEXEC FAILED ON STARTUP. ALL COMMANDS REMOVED.
```

```
{"C":0,"cmd":"AT+AUTOEXEC"}
{"A":0,"err":0,"errMsg":"ok"}
{"R":0,"info":"Autoexec failed on startup. All cmds removed."}
{"E":0,"nol":4}
```

Example:

```
# Adding command in list
AT+AUTOEXEC=AT+ADVSTART
OK
```

```
# If no commands in list
AT+AUTOEXEC
OK

AUTOEXEC LIST EMPTY!

# If commands in list exist
AT+AUTOEXEC
OK
CMD1: AT+ADVSTART
```

```
# Adding command in list
AT+AUTOEXEC=AT+ADVSTART
{"C":2,"cmd":"AT+AUTOEXEC=AT+ADVSTART"}
{"A":2,"err":0,"errMsg":"ok"}
{"E":2,"nol":3}
```

```
# If no commands in list
AT+AUTOEXEC
{"C":1,"cmd":"AT+AUTOEXEC"}
{"A":1,"err":0,"errMsg":"ok"}
{"R":1,"info":"Autoexec list empty"}
{"E":1,"nol":4}

# If commands in list exist
{"C":3,"cmd":"AT+AUTOEXEC"}
{"A":3,"err":0,"errMsg":"ok"}
```

```
{"R":3,"cmd1":"AT+ADVSTART"}}
{"E":3,"nol":4}
```

AT+AUTOEXECLOOP

- *Sets start/end of loop in auto execute list(AUTOEXEC). All auto execute commands between loop start and loop end will loop forever or till stopped by clearing the auto execute list using AT+CLRAUTOEXEC*

Command	Syntax
AT+AUTOEXECLOOP	AT+AUTOEXECLOOP=**START** or **END**

Any commands added to the auto execute list before loop start will be run only once.
Any commands added after the loop end will be ignored.
If only loop start is set, but not loop end, all the commands will run only once.

Details on **how to use Auto Execution loop** and examples are available here. **Example usage**

Example:

```
AT+AUTOEXECLOOP=START
OK
# Add commands using AT+AUTOEXEC that will be in the loop
AT+AUTOEXEC=ATI
OK
AT+AUTOEXEC=AT+WAIT=10
OK
AT+AUTOEXECLOOP=END
OK
AT+AUTOEXEC
OK
CMD1: LOOP_START
CMD2: ATI
CMD3: AT+WAIT=10
CMD4: LOOP_END
```

```
AT+AUTOEXECLOOP=START
{"C":513,"cmd":"AT+AUTOEXECLOOP=START"}
{"A":513,"err":0,"errMsg":"ok"}
{"E":513,"nol":3}
# Add commands using AT+AUTOEXEC that will be in the loop
AT+AUTOEXEC=ATI
{"C":514,"cmd":"AT+AUTOEXEC=ATI"}
{"A":514,"err":0,"errMsg":"ok"}
{"E":514,"nol":3}
AT+AUTOEXEC=AT+WAIT=10
{"C":515,"cmd":"AT+AUTOEXEC=AT+WAIT=10"}
{"A":515,"err":0,"errMsg":"ok"}
{"E":515,"nol":3}
```

```
AT+AUTOEXECLOOP=END
{"C":516,"cmd":"AT+AUTOEXECLOOP=END"}
{"A":516,"err":0,"errMsg":"ok"}
{"E":516,"nol":3}
AT+AUTOEXEC
{"C":517,"cmd":"AT+AUTOEXEC"}
{"A":517,"err":0,"errMsg":"ok"}
{"R":517,"cmd1":"LOOP_START"}
{"R":517,"cmd2":"ATI"}
{"R":517,"cmd3":"AT+WAIT=10"}
{"R":517,"cmd4":"LOOP_END"}
{"E":517,"nol":7}
```

AT+CANCELCONNECT

- *While in Central Mode, cancels any ongoing connection attempts.*

  Useful for when the peripheral has stopped advertising before a successful connection, or if an incorrect MAC address has been entered.

Command	Syntax
AT+CANCELCONNECT	AT+CANCELCONNECT

Example:

```
AT+CANCELCONNECT
OK
```

```
AT+CANCELCONNECT
{"C":11,"cmd":"AT+CANCELCONNECT"}
{"A":11,"err":0,"errMsg":"ok"}
{"E":11,"nol":3}
```

AT+CENTRAL

- *Sets the device Bluetooth role to central role. Advertising must be stopped and any connection must be terminated before the role change is accepted.*

Command	Syntax
AT+CENTRAL	AT+CENTRAL

Example:

```
AT+CENTRAL
OK
```

```
AT+CENTRAL
{"C":12,"cmd":"AT+CENTRAL"}
{"A":12,"err":0,"errMsg":"ok"}
{"E":12,"nol":3}
```

AT+CLEARINDI

- *Disables indication for selected characteristic.*

Command	Syntax
AT+CLEARINDI	AT+CLEARINDI=***(indication handle)***

Which is the indication handle?

```
0011 char 6e400001-b5a3-f393-e0a9-e50e24dcca9e prop=20 (-----I--) # <--------
The I means that the Characteristic Properties has Indication
0012 ---- 6e400003-b5a3-f393-e0a9-e50e24dcca9e  # <--------- This (0012) is
the handle you want to send in.
0013 desc 0x2902                # You can use AT+GATTCREAD to read the
description handle (here 0013) to see if indication is set. (0x0100 = set,
0x0000 = not set)
```

- *If the status=0 in handle_evt_gattc_write_completed string, that means you have successfully written to it.*

Example:

```
AT+CLEARINDI=0012
handle_evt_gattc_write_completed: conn_idx=0000 handle=0013 status=0
```

```
{"C":19,"cmd":"AT+CLEARINDI=0014"}
{"A":19,"err":0,"errMsg":"ok"}
{"E":19,"nol":3}
{776:"0000","0015":{"writeStatus":0}} # <--- Event
```

AT+CLEARNOTI

- *Disables notification for selected characteristic.*

Command	Syntax
AT+CLEARNOTI	AT+CLEARNOTI=***(notification handle)***

Which is the notification handle?

```
0011 char 6e400001-b5a3-f393-e0a9-e50e24dcca9e prop=10 (----N---) # <--------
The N means that the Characteristic Properties has Notify
0012 ---- 6e400003-b5a3-f393-e0a9-e50e24dcca9e  # <--------- This (0012) is
the handle you want to send in.
0013 desc 0x2902                # You can use AT+GATTCREAD to read the
description handle (here 0013) to see if notification is set. (0x0100 = set,
0x0000 = not set)
```

- *If the status=0 in handle_evt_gattc_write_completed string, that means you have successfully written to it.*

Example:

```
AT+CLEARNOTI=0012
handle_evt_gattc_write_completed: conn_idx=0000 handle=0013 status=0
```

```
AT+CLEARNOTI=000D
{"C":21,"cmd":"AT+CLEARNOTI=000D"}
{"A":21,"err":0,"errMsg":"ok"}
{"E":21,"nol":3}
{776:"0000","000e":{"writeStatus":0}} # <--- Event
```

AT+CLIENT

- *Only usable in Dual role. Sets the dongle role towards the targeted connection to client.*

Command	Syntax
AT+CLIENT	AT+CLIENT

Example:

```
AT+CLIENT
OK
```

```
AT+CLIENT
{"C":29,"cmd":"AT+CLIENT"}
{"A":29,"err":0,"errMsg":"ok"}
{"E":29,"nol":3}
```

AT+CLRAUTOEXEC

- *Clear any commands in the auto execute (AUTOEXEC) list.*

Command	Syntax
AT+CLRAUTOEXEC	AT+CLRAUTOEXEC

Example:

```
AT+CLRAUTOEXEC
OK
```

```
{"C":3,"cmd":"AT+CLRAUTOEXEC"}
{"A":3,"err":0,"errMsg":"ok"}
{"E":3,"nol":3}
```

AT+CLRAUTOEXECPWD

- *Used to clear/remove existing password (requires entering password first). BleuIO will go back to initial state were no password is set.*

Command	Syntax
AT+CLRAUTOEXECPWD	
	AT+CLRAUTOEXECPWD

Example:

```
AT+CLRAUTOEXECPWD
NOT ALLOWED!
ENTER AUTOEXEC PASSWORD...
AT+ENTERAUTOEXECPWD=my_password
PASSWORD ACCEPTED
AUTOEXEC PWD CLEARED
OK
```

```
AT+CLRAUTOEXECPWD
{"C":134,"cmd":"AT+CLRAUTOEXECPWD"}
{"A":134,"err":5,"errMsg":"not allowed, enter autoexec password..."}
{"E":134,"nol":3}
AT+ENTERAUTOEXECPWD=my_password
{"C":135,"cmd":"AT+ENTERAUTOEXECPWD=my_password"}
{"A":135,"err":0,"errMsg":"ok"}
{"R":135,"status":"password accepted","action":"autoexec password cleared"}
{"E":135,"nol":4}
```

AT+CLRUOI

- *Clear any set Unique Organization ID.*

Command	Syntax
AT+CLRUOI	AT+CLRUOI

Example:

```
ATI
Smart Sensor Devices
DA14683 (P25Q80LE)
BleuIO
Firmware Version: 2.7.6


Peripheral role


Not Connected


Advertising


Your Unique Organization ID
AT+CLRUOI
OK
ATI
Smart Sensor Devices
DA14683 (P25Q80LE)
BleuIO
Firmware Version: 2.7.6


Peripheral role


Not Connected


Advertising
```

```
ATI
{"C":1,"cmd":"ATI"}
{"A":1,"err":0,"errMsg":"ok"}
{"R":1,"dev":"Smart Sensor Devices","hw":"DA14683
(P25Q80LE)","name":"BleuIO"}
{"R":1,"fwVer":"2.7.6","gap_role":"peripheral"}
{"R":1,"connected":false,"advertising":true}
{"R":1,"uoi":"Your Unique Organization ID"}
{"E":1,"nol":7}
AT+CLRUOI
```

```
{"C":2,"cmd":"AT+CLRUOI"}
{"A":2,"err":0,"errMsg":"ok"}
{"E":2,"nol":3}
ATI
{"C":3,"cmd":"ATI"}
{"A":3,"err":0,"errMsg":"ok"}
{"R":3,"dev":"Smart Sensor Devices","hw":"DA14683
(P25Q80LE)","name":"BleuIO"}
{"R":3,"fwVer":"2.7.6","gap_role":"peripheral"}
{"R":3,"connected":false,"advertising":true}
{"E":3,"nol":6}
```

AT+CONNECTBOND

- *Scan for and initiates a connection with a selected bonded device. Works even if the peer bonded device is advertising with a Private Random Resolvable Address. Upon successfully connecting it will attempts to display the services of the peer device (if auto-discover services (ATDS) is on). The dongle must be in central or dual role.*

Command	**Syntax**
AT+CONNECTBOND	AT+CONNECTBOND=***bonded address***

This command will do a scan for up to 12 seconds to see if the bonded address is advertising. It will also resolve any addresses it can with it's IRK (Identity Resolving Key) and check if any resolved address matches the bonded address. If it finds the bonded address, either advertising with a static address or Private Random Resolvable Address, it will stop the scan and connect to it.

The bonded address must be in the bonded list (AT+GETBOND).

Example:

```
AT+CONNECTBOND=40:48:FD:EA:E8:38
Trying to connect...

Identity Address: 40:48:FD:EA:E8:38

SCAN COMPLETE

CONNECTED.

Target conn indx changed to=0000

handle_evt_gap_connected: conn_idx=0000 address=40:48:fd:ea:e8:38 CI max is
24.
```

```
AT+CONNECTBOND=40:48:FD:EA:E8:38
{"C":5,"cmd":"AT+CONNECTBOND=40:48:FD:EA:E8:38"}
```

```
{"A":5,"err":0,"errMsg":"ok"}
{"R":5,"evt":{"action":"connecting","addr":"40:48:FD:EA:E8:38"}}
{"E":5,"nol":4}
{"S":0,"idAddr":"40:48:FD:EA:E8:38"}
{"SE":0,"evt":{"action":"scan completed"}}
{256:"0000","evt":{"action":"target conn index changed","target":"0000"}}
{256:"0000","evt":
{"action":"connected","addr":"40:48:fd:ea:e8:38","CImin":24,"CImax":24}}
```

AT+CONNPARAM

- *Sets or displays preferred connection parameters. When run while connected will update connection parameters on the current target connection.*

Command	Syntax
AT+CONNPARAM	AT+CONNPARAM
	or
	AT+CONNPARAM=**intv_min=intv_max=slave_latency=supervision_timeout**

Setting connection parameters when not connected will result in using set connection parameters as the default when using the AT+GAPCONNECT command without arguments.

Querying connection parameters using the command without parameters will display the current set preferred connection parameters. When used while connected will instead show current active connection parameters.

Connection Parameter	Range
intv_min (Connection Interval Min)	7.5ms-4000ms
intv_max (Connection Interval Max)	7.5ms-4000ms
slave_latency (Slave Latency)	0-499*
supervision_timeout (Supervision Time-out)	100ms-32000ms

(*) The slave latency can also not exceed ((supervision_timeout / (intv_max * 2)) -1).

The CI min and CI max as shown upon connection event are calculated as such: (interval in ms) * 100 / 125. So if you for example set the intv_min to 30 it will show as CI min of 24.

Example:

```
AT+CONNPARAM
Connection Interval min = 24
Connection Interval max = 24
Slave Latency = 1
Supervision Timeout = 1000ms.
```

```
AT+CONNPARAM
{"C":0,"cmd":"AT+CONNPARAM"}
{"A":0,"err":0,"errMsg":"ok"}
{"R":0,"evt":{"action":"getting conn param"}}
{"R":0,"CImin":24,"CImax":24,"sl":1,"sup_to(ms)":1000}
{"E":0,"nol":5}
```

When used while not connected:

```
AT+CONNPARAM=30=60=1=1000
OK
DEFAULT CONN PARAM UPDATED!
```

```
AT+CONNPARAM=30=60=1=1000
{"C":0,"cmd":"AT+CONNPARAM=30=60=1=1000"}
{"A":0,"err":0,"errMsg":"ok"}
{"R":0,"evt":{"action":"default conn param updated"}}
{"E":0,"nol":4}
```

Used when connected:

```
AT+CONNPARAM=30=30=1=1000
OK
ACTIVE CONN PARAM UPDATED!

Peripheral updated CI min is 24, CI max is 24.
```

```
AT+CONNPARAM=30=30=1=1000
{"C":5,"cmd":"AT+CONNPARAM=30=30=1=1000"}
{"A":5,"err":0,"errMsg":"ok"}
{"R":5,"evt":{"action":"active conn param updated"}}
{"E":5,"nol":4}
{263:"0000","evt":{"action":"conn param updated","CImin":24,"CImax":24}}
{272:"0000","evt":{"action":"conn param updated completed","status":0}}
```

AT+CONNSCANPARAM

- *Set or queries the connection scan window and interval used.*

Command	Syntax
AT+CONNSCANPARAM	AT+CONNSCANPARAM=***scan intv ms***=***scan win ms***
	AT+CONNSCANPARAM

Scan Interval is the duration of time between scans.
Scan Window determines how long to scan for at each interval.
The Scan Window parameter must be equal or smaller than the Scan Interval.

The scan interval and window can be set in steps of 0.625ms. Allowed values for interval span in the range of 2.5ms to 10240ms. This command set the scan parameters used for initiated connections.

Return scan window and scan interval values are in units of 0.625ms and can be converted to ms like this:
return value * 0.625
So for example 160 * 0.625 = 100ms
and 320 * 0.625 = 200ms

Example:

```
AT+CONNSCANPARAM=200=100
OK
AT+CONNSCANPARAM
OK
Connection Scan Interval = 320
Connection Scan Window = 160
```

```
AT+CONNSCANPARAM=200=100
{"C":0,"cmd":"AT+CONNSCANPARAM=200=100"}
{"A":0,"err":0,"errMsg":"ok"}
{"E":0,"nol":3}
AT+CONNSCANPARAM
{"C":1,"cmd":"AT+CONNSCANPARAM"}
{"A":1,"err":0,"errMsg":"ok"}
{"R":1,"evt":{"action":"getting conn scan param"}}
{"R":1,"conn_scan_intv":320,"conn_scan_win":160}
{"E":1,"nol":5}
```

AT+CUSTOMSERVICE

- Sets or queries Custom Service. Max 5 Characteristics can be added.
- ***Several values cannot be changed while connected/connecting or advertising.***

Command	Syntax
AT+CUSTOMSERVICE	
	AT+CUSTOMSERVICE
	AT+CUSTOMSERVICE=*Idx*=UUID=***XXXX*** or AT+CUSTOMSERVICE=*Idx*=UUID=***xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx***
	AT+CUSTOMSERVICE=*Idx*=PROP=***C...***
	AT+CUSTOMSERVICE=*Idx*=PERM=***C...***
	AT+CUSTOMSERVICE=*Idx*=LEN=***l***

Command	Syntax
	AT+CUSTOMSERVICE=*Idx*=VALUE=***data***
	AT+CUSTOMSERVICE=*Idx*=VALUEB=***data***
	AT+CUSTOMSERVICE=*Idx*=DPERM=***C...***
	AT+CUSTOMSERVICE=*Idx*=DLEN=***l***
	AT+CUSTOMSERVICE=*Idx*=DVALUE=***data***
	AT+CUSTOMSERVICE=*Idx*=DVALUEB=***XXX***

The first parameter when using the AT+CUSTOMSERVICE= to setup a Service is the index to allow you to target which Characteristic you want to setup or set the UUID of the Service.

Index	Type
0	Service
1	Characteristic1
2	Characteristic2
3	Characteristic3
4	Characteristic4
5	Characteristic5

AT+CUSTOMSERVICE can take the following parameters:

Parameter	Description	Input Value	Example	Can be changed after starting Service?
UUID	Set service or characteristic UUID.	16-bit UUID or 128-bit UUID (Hex Big Endian)	*For setting 16-bit UUID of Service:*	NO
			AT+CUSTOMSERVICE=0=UUID=1801	
			For setting 16-bit UUID of Characteristic1	
			AT+CUSTOMSERVICE=1=UUID=1802	
PROP	Set the characteristic properties.	Characteristic Properties	AT+CUSTOMSERVICE=1=PROP=WNR	NO

Parameter	Description	Input Value	Example	Can be changed after starting Service?
	See the properties table below for supported properties.			
PERM	Set the characteristic permission.	Characteristic Permissions	AT+CUSTOMSERVICE=5=PERM=RW	NO
	See the permission table below for supported permission.			
LEN	Max size in bytes of the value the Characteristic can hold.	Number between 1-250	AT+CUSTOMSERVICE=3=LEN=112	NO
VALUE	Set value of characteristic in ASCII.	Characteristic value	AT+CUSTOMSERVICE=1=VALUE=CharDATA1	YES
	Will overwrite Byte Value if any.			
VALUEB	Set value of characteristic in Bytes.	Characteristic hex value	AT+CUSTOMSERVICE=1=VALUEB=0102030405	YES
	Will overwrite ASCII Value if any.			
DPERM	Set the characteristic properties.	Descriptor Permissions	AT+CUSTOMSERVICE=5=DPERM=R	NO

Parameter	Description	Input Value	Example	Can be changed after starting Service?
	See the permission table below for supported permission.			
DLEN	Max size in bytes of the value the Descriptor can hold.	Number between 1-250	AT+CUSTOMSERVICE=3=LEN=112	NO
DVALUE	Set value of descriptor in ASCII.	Descriptor value	AT+CUSTOMSERVICE=2=DVALUE=DescDATA1	YES
	Will overwrite Byte Value if any.			
DVALUEB	Set value of descriptor in Bytes.	Descriptor hex value	AT+CUSTOMSERVICE=1=DVALUEB=FF02020202	YES
	Will overwrite ASCII Value if any.			

> **NOTE:** Setting at least a valid Service UUID is required to start the service.
> Characteristics without an UUID will not be added, nor will their descriptor.
> Descriptors will only be added if the have a value when starting the service.
> Characteristic & Descriptor Max Size default value is 0. Max Size need to be increased before setting Characteristic or Descriptor Value.
> Characteristic & Descriptor Max Size cannot be set lower than the Size of the current Value, if any.
> Characteristic & Descriptor Value cannot be larger than the currently set Max Size for that Characteristic/Descriptor.

Valid input values:

Input Value	Format	Example:	Case-sensitive?

Input Value	Format	Example:	Case-sensitive?
128-bit UUID	xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx	fa284580-9d88-4b70-9775-ce4786a6bb96	NO
16-bit UUID*	XXXX	1801	NO
Characteristic & Descriptor value	String	Hello	YES
Characteristic & Descriptor hex value	XXXXXXXX...	0102030405060A0BDDFFEE	YES
Number	String	12	NO
Characteristic Properties	String	RWXI	YES
Characteristic & Descriptor Permissions	String	RW	YES

Properties Table:

Value	Meaning	Description
R	Read	Client can read the Characteristic value.
		Read permission need to also be set.
W	Write	Client can write to the Characteristic value and expect an acknowledgment (response) from server.
		Write permission need to also be set.
X	Write without Response	Client can write to the Characteristic value without the server acknowledging (responding) to client.
		Write permission need to also be set.
N	Notify	Client can subscribe to characteristic value allowing server to initiate data transfer when changing characteristic value.
		Client does not acknowledge (respond) data.
I	Indicate	Same as Notify but client has to acknowledge data on arrival.

> **NOTE:** Changing Characteristic Value with either Notify or Indicate property set and connected to peers will notify/indicate all peers that has subscribed to the characteristic.

Permissions Table:

Value	Meaning	Description
R	Read	No encryption required (Security Level 1)
		Required for client to be able to read the characterisitc/descriptor.
E	Read (Encryption)	Connection with encryption required (Security Level 2+)
		Required for client to be able to read the characterisitc/descriptor.
A	Read (Authentication)	Authenticated connection Required (Security Level 3+)
		Required for client to be able to read the characterisitc/descriptor.
W	Write	No encryption required (Security Level 1)
		Required for client to be able to write to the characterisitc/descriptor.
D	Write (Encryption)	Connection with encryption required (Security Level 2+)
		Required for client to be able to write to the characterisitc/descriptor.
B	Write (Authentication)	Authenticated connection Required (Security Level 3+)
		Required for client to be able to write to the characterisitc/descriptor.

> **NOTE:** Setting permissions requiring Encryption/Authentication requires that the BleuIO is setup to achive the required Security Mode otherwise the client will be unable to interact with the characterisitc/descriptor.

Properties and Permissions can be in any order. For example setting Read, Write and Notify Properties can be set like this:
AT+CUSTOMSERVICE=1=PROP=RWN or **AT+CUSTOMSERVICE=1=PROP=NRW**

Properties can also be added in any combination. With the exception of **Notify** and **Indicate**, which are exclusive. Meaning you can only choose one or the other.

Read Permissions and Write Permissions are also exclusive. You can only have one type of Read Permission and/or only one type of Write Permission but in any combination.
For example *Read (Encryption)* and *Write (Authentication)*:
AT+CUSTOMSERVICE=1=PERM=EB
or *Read* and *Write (Encryption)*:
AT+CUSTOMSERVICE=1=PERM=RD

Output when Client is writing to a custom characteristic/descriptor

```
cust_char1 RX: conn_idx=0000 handle=001a length=5
```

```
Value received: hello
Hex: 0x68656C6C6F
```

```
{281:"0000","evt":
{"handle":"001e","handleName":"cust_char2","ascii":"hello","len":5}}
{281:"0000","evt":
{"handle":"001e","handleName":"cust_char2","hex":"0x68656C6C6F","len":5}}
```

Output when Client is setting notification/indication

```
Indication turned on by connIdx=0000 for cust_char1 (handle=001B)

Notification/Indication turned off by connIdx=0000 for cust_char2
(handle=001F)

Notification turned on by connIdx=0000 for cust_char2 (handle=001F)
```

```
## notification:
{282:"0000","evt":{"action":"noti/indi
change","handle":"001f","handleName":"cust_char2","value":0001}}
## indication:
{282:"0000","evt":{"action":"noti/indi
change","handle":"001b","handleName":"cust_char1","value":0002}}
## turning off notification/indication:
{282:"0000","evt":{"action":"noti/indi
change","handle":"001b","handleName":"cust_char1","value":0000}}
```

Example:

```
AT+CUSTOMSERVICE
CUSTOMSERVICE SERVICE UUID = Not set!

AT+CUSTOMSERVICE=0=UUID=ee6ec068-7447-4045-9fd0-593f3ba3c2ee
OK
AT+CUSTOMSERVICE=1=UUID=018f55d9-d747-4c4e-a87b-e9b074ffd2b6
OK
AT+CUSTOMSERVICE=1=PROP=RWI
OK
AT+CUSTOMSERVICE=1=PERM=RW
OK
AT+CUSTOMSERVICE=1=LEN=100
OK
AT+CUSTOMSERVICE=1=VALUE=CharDATA1
OK
```

```
AT+CUSTOMSERVICE=1=DLEN=100
OK
AT+CUSTOMSERVICE=1=DVALUE=DescriptorText1
OK
AT+CUSTOMSERVICE=1=DPERM=RW
OK

AT+CUSTOMSERVICE
CUSTOMSERVICE SERVICE UUID= ee6ec068-7447-4045-9fd0-593f3ba3c2ee

CUSTOMSERVICE CHAR1 UUID=018f55d9-d747-4c4e-a87b-e9b074ffd2b6
PROP=RWI
PERM=RW
SIZE=100
VALUE(ASCII)=CharDATA1

CUSTOMSERVICE CHAR1 DESC
PERM=RW
SIZE=100
VALUE(ASCII)=DescriptorText1
```

```
AT+CUSTOMSERVICE
{"C":5,"cmd":"AT+CUSTOMSERVICE"}
{"A":5,"err":0,"errMsg":"ok"}
{"R":5,"customServiceUUID":"","set":"false"}
{"E":5,"nol":4}
AT+CUSTOMSERVICE=0=UUID=ee6ec068-7447-4045-9fd0-593f3ba3c2ee
{"C":0,"cmd":"AT+CUSTOMSERVICE=0=UUID=ee6ec068-7447-4045-9fd0-593f3ba3c2ee"}
{"A":0,"err":0,"errMsg":"ok"}
{"E":0,"nol":3}
AT+CUSTOMSERVICE=1=UUID=018f55d9-d747-4c4e-a87b-e9b074ffd2b6
{"C":1,"cmd":"AT+CUSTOMSERVICE=1=UUID=018f55d9-d747-4c4e-a87b-e9b074ffd2b6"}
{"A":1,"err":0,"errMsg":"ok"}
{"E":1,"nol":3}
AT+CUSTOMSERVICE=1=PROP=RWI
{"C":2,"cmd":"AT+CUSTOMSERVICE=1=PROP=RWI"}
{"A":2,"err":0,"errMsg":"ok"}
{"E":2,"nol":3}
AT+CUSTOMSERVICE=1=PERM=RW
{"C":3,"cmd":"AT+CUSTOMSERVICE=1=PERM=RW"}
{"A":3,"err":0,"errMsg":"ok"}
{"E":3,"nol":3}
AT+CUSTOMSERVICE=1=LEN=100
{"C":4,"cmd":"AT+CUSTOMSERVICE=1=LEN=100"}
{"A":4,"err":0,"errMsg":"ok"}
{"E":4,"nol":3}
AT+CUSTOMSERVICE=1=VALUE=CharDATA1
{"C":5,"cmd":"AT+CUSTOMSERVICE=1=VALUE=CharDATA1"}
{"A":5,"err":0,"errMsg":"ok"}
{"E":5,"nol":3}
AT+CUSTOMSERVICE=1=DLEN=100
{"C":6,"cmd":"AT+CUSTOMSERVICE=1=DLEN=100"}
{"A":6,"err":0,"errMsg":"ok"}
```

```
{"E":6,"nol":3}
AT+CUSTOMSERVICE=1=DVALUE=DescriptorText1
{"C":7,"cmd":"AT+CUSTOMSERVICE=1=DVALUE=DescriptorText1"}
{"A":7,"err":0,"errMsg":"ok"}
{"E":7,"nol":3}
AT+CUSTOMSERVICE=1=DPERM=RW
{"C":8,"cmd":"AT+CUSTOMSERVICE=1=DPERM=RW"}
{"A":8,"err":0,"errMsg":"ok"}
{"E":8,"nol":3}

AT+CUSTOMSERVICE
{"C":5,"cmd":"AT+CUSTOMSERVICE"}
{"A":5,"err":0,"errMsg":"ok"}
{"R":5,"customServiceUUID":"ee6ec068-7447-4045-9fd0-
593f3ba3c2ee","set":"true"}
{"R":5,"customChar1":{"UUID":"018f55d9-d747-4c4e-a87b-
e9b074ffd2b6","prop":"RWI","perm":"RW","Size":100,"value(ascii)":"CharDATA1"}
}
{"R":5,"customDesc1":
{"perm":"RW","Size":100,"value(ascii)":"DescriptorText1"}}
{"E":5,"nol":6}
```

AT+CUSTOMSERVICESTART

- *Starts the Custom Service based on the settings set by AT+CUSTOMSERVICE= Command.*
  Cannot be started while connected/connecting or advertising.

Command	Syntax
AT+CUSTOMSERVICESTART	AT+CUSTOMSERVICESTART

Example:

```
AT+CUSTOMSERVICESTART
OK
```

```
AT+CUSTOMSERVICESTART
{"C":27,"cmd":"AT+CUSTOMSERVICESTART"}
{"A":27,"err":0,"errMsg":"ok"}
{"E":27,"nol":3}
```

AT+CUSTOMSERVICESTOP

- *Stops the Custom Service.*
  Cannot be changed while connected/connecting or advertising.

Command	Syntax
AT+CUSTOMSERVICESTOP	AT+CUSTOMSERVICESTOP

Example:

```
AT+CUSTOMSERVICESTOP
OK
```

```
AT+CUSTOMSERVICESTOP
{"C":27,"cmd":"AT+CUSTOMSERVICESTOP"}
{"A":27,"err":0,"errMsg":"ok"}
{"E":27,"nol":3}
```

AT+CUSTOMSERVICERESET

- *Stops the Custom Service and resets the Custom Service settings set by the AT+CUSTOMSERVICE= command to it's default values.*
  Cannot be changed while connected/connecting or advertising.

Command	Syntax
AT+CUSTOMSERVICERESET	AT+CUSTOMSERVICERESET

Example:

```
AT+CUSTOMSERVICERESET
OK
```

```
AT+CUSTOMSERVICERESET
{"C":27,"cmd":"AT+CUSTOMSERVICERESET"}
{"A":27,"err":0,"errMsg":"ok"}
{"E":27,"nol":3}
```

AT+DEVICENAME

- *Get or sets the device name used for GAP service. Cannot be set while connected or advertising.*

Command	Syntax
AT+DEVICENAME	AT+DEVICENAME=**new_device_name**
	or
	AT+DEVICENAME

Example:

```
AT+DEVICENAME=NEW_NAME
SET DEVICE NAME=NEW_NAME

OK
```

```
AT+DEVICENAME
{"C":30,"cmd":"AT+DEVICENAME"}
{"A":30,"err":0,"errMsg":"ok"}
{"R":30,"deviceName":"BleuIO"}
{"E":30,"nol":4}
```

```
AT+DEVICENAME=TESTNAME
{"C":0,"cmd":"AT+DEVICENAME=TESTNAME"}
{"A":0,"err":0,"errMsg":"ok"}
{"R":0,"deviceName":"TESTNAME"}
{"E":0,"nol":4}
```

AT+DIS

- *Shows the Device Information Service information to be used.*

Command	Syntax
AT+DIS	AT+DIS

Shows the Device Information Service information to be used. Will show the default BleuIO information if no custom information has been set.

Example:

```
AT+DIS
DIS Service is set but not locked in.
dis_info_start:
manufacturer=MAN_NAME
model_number=MOD_NUM
serial_number=SERIAL
hw_revision=HW_REV
fw_revision=FW_REV
sw_revision=SW_REV
dis_info_end
```

```
AT+DIS
{"C":32,"cmd":"AT+DIS"}
{"A":32,"err":0,"errMsg":"ok"}
{"R":32,"info":"DIS service info is locked in"}
```

```
{"R":32,"manuf":"Smart Sensor Devices","mdlNr":"SSD005","sNr":"000000000000"}
{"R":32,"hwRev":"Rev.A","fwRev":"2.1.5","swRev":"bleuio.2.1.5_Release"}
{"E":32,"nol":6}
```

AT+DUAL

- *Sets the device Bluetooth role to dual role. This means it has the capabilities of both Central and Peripheral role. Advertising must be stopped and, any connection must be terminated before the role change is accepted.*

Command	Syntax
AT+DUAL	AT+DUAL

Example:

```
AT+DUAL
OK
```

```
AT+DUAL
{"C":27,"cmd":"AT+DUAL"}
{"A":27,"err":0,"errMsg":"ok"}
{"E":27,"nol":3}
```

AT+ENTERAUTOEXECPWD

- *Used to enter autoexec password when prompted.*

Command	Syntax
AT+ENTERAUTOEXECPWD	
	AT+ENTERAUTOEXECPWD=**xxxxxxx..**

Example:

```
AT+AUTOEXEC
NOT ALLOWED!
ENTER AUTOEXEC PASSWORD...
AT+ENTERAUTOEXECPWD=my_password
PASSWORD ACCEPTED
OK

AUTOEXEC LIST EMPTY!
```

```
AT+AUTOEXEC
{"C":132,"cmd":"AT+AUTOEXEC"}
{"A":132,"err":5,"errMsg":"not allowed, enter autoexec password..."}
{"E":132,"nol":3}
AT+ENTERAUTOEXECPWD=my_password
{"C":133,"cmd":"AT+ENTERAUTOEXECPWD=my_password"}
{"A":133,"err":0,"errMsg":"ok"}
{"R":133,"status":"password accepted","info":"Autoexec list empty"}
{"E":133,"nol":4}
```

AT+ENTERPASSKEY

- *When faced with this message: **BLE_EVT_GAP_PASSKEY_REQUEST** use the **AT+ENTERPASSKEY** command to enter the 6-digit passkey to continue the pairing request.*

Command	Syntax
AT+ENTERPASSKEY	
	AT+ENTERPASSKEY=***xxxxxx***

Example:

```
AT+ENTERPASSKEY=123456
OK
```

```
AT+ENTERPASSKEY=123456
{"C":6,"cmd":"AT+ENTERPASSKEY=123456"}
{"A":6,"err":0,"errMsg":"ok"}
{"E":6,"nol":3}
```

AT+FINDSCANDATA

- *Scans for all advertising/response data which contains the search params for infinite time unless an optional scan timer parameter is set.*
  Scan can be stopped at any time by sending CTRL+C (0x03).

Command	Syntax
AT+FINDSCANDATA	AT+FINDSCANDATA=***search params***
	AT+FINDSCANDATA=***search params***=***seconds***

Example:

```
AT+FINDSCANDATA=FF5
SCANNING...
```

```
[00:D4:2E:CD:72:23] Device Data [ADV]:
02010618FF5B00110101010101010101011405326845343864353466757675

[D7:D3:AA:00:B5:24] Device Data [ADV]:
02010618FF540930430924302903049320943094F90890548359049E495432

[F9:0D:00:E7:72:21] Device Data [ADV]:
02010618FF540930430924302903049320943094F90890548359049E495432

[F3:00:ED:AD:8A:22] Device Data [ADV]:
02010618FF5A40930430924302903049320943094F90890548359049E49543

[F3:DE:00:D5:96:33] Device Data [ADV]:
0201061AFF5C49320943094F90890548359049E49543249320943094F90890

[C9:70:27:AF:01:54] Device Data [ADV]:
02010618FF5E924302903049320943094F9089054835992430290304932094

[D7:00:AA:07:00:EE] Device Data [ADV]:
02010618FF530101010101010101011405326845343864353466701010101010
```

```
AT+FINDSCANDATA=FF5B07
{"C":36,"cmd":"AT+FINDSCANDATA=FF5B07"}
{"A":36,"err":0,"errMsg":"ok"}
{"R":36,"action":"scanning"}
{"E":36,"nol":4}
{"SF":36,"addr":"CC:EA:2B:D7:88:48","type":0,"data":"0201061AFF5B0705040578EB
DD008427E800C00052010000000000000020600"}
{"SF":36,"addr":"D0:97:8B:FE:18:6D","type":0,"data":"0201061BFF5B070504220080
0E008127E900B000CF000000000000000020703"}
# Until stopped with Ctrl+C
```

```
AT+FINDSCANDATA=FF5B07=2
{"C":38,"cmd":"AT+FINDSCANDATA=FF5B07=2"}
{"A":38,"err":0,"errMsg":"ok"}
{"R":38,"action":"scanning"}
{"E":38,"nol":4}
{"SF":38,"addr":"F5:50:35:CF:B1:ED","type":0,"data":"0201061BFF5B07050422013F
BD007D27E000BB00F41900000000000000020A02"}
{"SF":38,"addr":"D2:B1:28:3F:42:D4","type":0,"data":"0201061BFF5B070504220049
880B7F27EE00AB000A01000000000000000024503"}
{"SE":38,"action":"scan completed"}
```

AT+FRSSI

- *Sets/reads RSSI filter for scan results. Afterwards scans will only show results with the chosen max value or below.*

Command	Syntax
AT+FRSSI	AT+FRSSI
	AT+FRSSI=**max_rssi**
	AT+FRSSI=OFF

Acceptable parameter range: -1 to -99

Example:

```
AT+FRSSI
OK
RSSI FILTER OFF
AT+FRSSI=-76
OK
RSSI FILTER ON
FILTER_VALUE: -76
AT+FRSSI
OK
RSSI FILTER ON
FILTER_VALUE: -76
AT+FRSSI=OFF
OK
RSSI FILTER OFF
```

```
AT+FRSSI
{"C":129,"cmd":"AT+FRSSI"}
{"A":129,"err":0,"errMsg":"ok"}
{"R":129,"filter_rssi":false}
{"E":129,"nol":4}
AT+FRSSI=-76
{"C":131,"cmd":"AT+FRSSI=-76"}
{"A":131,"err":0,"errMsg":"ok"}
{"R":131,"filter_rssi":true,"rssi_filer_value":"-76"}
{"E":131,"nol":4}
AT+FRSSI
{"C":132,"cmd":"AT+FRSSI"}
{"A":132,"err":0,"errMsg":"ok"}
{"R":132,"filter_rssi":true,"rssi_filer_value":"-76"}
{"E":132,"nol":4}
AT+FRSSI=OFF
{"C":134,"cmd":"AT+FRSSI=OFF"}
{"A":134,"err":0,"errMsg":"ok"}
{"R":134,"filter_rssi":false}
{"E":134,"nol":4}
```

AT+GAPADDRTYPE

- *Sets or queries what address type the dongle will use. Changing address type cannot be done while advertising or while connected to other devices.*

Command	Syntax
AT+GAPADDRTYPE	AT+GAPADDRTYPE=**address_type_code**
	or
	AT+GAPADDRTYPE

Table of supported address types:

Address Type	Description	Code
PUBLIC_STATIC_ADDRESS	Public Static Address	1
PRIVATE_STATIC_ADDRESS	Private Static Address	2
PRIVATE_RANDOM_RESOLVABLE_ADDRESS	Private Random Resolvable Address	3
PRIVATE_RANDOM_NONRESOLVABLE_ADDRESS	Private Random Non-resolvable Address	4
PRIVATE_CNTL	Private Random Resolvable address using Bluetooth LE Privacy v1.2	5

Non-static addresses will renew themselves every 150 seconds.
When 'Private Non-resolvable Address' is set BleuIO will not be able to connect to other devices or advertise in any other mode than in Non-connectable mode: GAP_CONN_MODE_NON_CONN. This will be chosen by default if starting advertising with the AT+ADVSTART command (without parameters) if Private Random Non-resolvable Address is set.

About Bluetooth LE Privacy:

> *"When Bluetooth LE Privacy is in use and advertising packets contain randomly generated MAC addresses disguising your device's identity, the real MAC address remains hidden away. But what use is this if the outside world sees your device as having a different address?*
>
> *The answer lies in the Bluetooth pairing process—Bluetooth® users are familiar with this process. Pairing indicates you trust the other device and want to interact with it. For example, if you pair your activity tracker with your phone, from that point on, the phone will have a special, trusted relationship with the tracker. What happens is much more involved but after pairing, the two devices will possess various encryption keys, one of which is concerned with privacy. This key is called the Identity Resolution Key (IRK). IRK allows the first device to translate those special, random MAC addresses which appear in the advertising packets from the second device, to the real MAC address in the second device. This capability is only in devices you have explicitly trusted."*
> - From "Bluetooth Technology Protecting Your Privacy"

Example:

```
AT+GAPADDRTYPE=3
OK
```

```
AT+GAPADDRTYPE
{"C":41,"cmd":"AT+GAPADDRTYPE"}
{"A":41,"err":0,"errMsg":"ok"}
{"R":41,"addrType":"PUBLIC_STATIC_ADDRESS"}
{"E":41,"nol":4}
```

```
AT+GAPADDRTYPE=2
{"C":8,"cmd":"AT+GAPADDRTYPE=2"}
{"A":8,"err":0,"errMsg":"ok"}
{"E":8,"nol":3}
```

AT+GAPCONNECT

- *Initiates a connection with a specific slave device. Upon successfully connecting it will attempts to display the services of the peer device (if auto-discover services (ATDS) is on). The dongle must be in central or dual role.*

Command	Syntax
AT+GAPCONNECT	AT+GAPCONNECT=***[addr_type]slave address***
	AT+GAPCONNECT=***[addr_type]slave_address=intv min:intv max:slave latency:supervision timeout:***
	AT+GAPCONNECT=*deviceIndex!*
	AT+GAPCONNECT=*deviceIndex!=intv min:intv max:slave latency:supervision timeout:*

Address Type	
[0]	PUBLIC_ADDRESS
[1]	PRIVATE_ADDRESS

If no connection parameters are entered then the parameters set with the AT+CONNPARAM command will be used, otherwise the default values will be used.
Default values:
intv min=30
intv max=30
slave latency=0
supervision timeout=1000

Connection Parameter	Range
intv min (Connection Interval Min)	7.5ms-4000ms
intv max (Connection Interval Max)	7.5ms-4000ms
slave latency (Slave Latency)	0-499*

Connection Parameter	Range
supervision timeout (Supervision Time-out)	100ms-32000ms

(*) The **slave latency** can also not exceed ((supervision timeout / (intv max * 2)) -1).

The CI min and CI max as shown upon connection event are calculated as such: (interval in ms) * 100 / 125. So if you for example set the intv_min to 30 it will result in a CI min of 24.

If a connection fails to be established and causes a disconnection event right after a connection event, the dongle will now automatically try to reconnect instead of showing the disconnecting event. The response shown will be: "**Connection failed to be established. Reconnecting...**" It will try up to 6 times. If it fails after that this response will be shown: "**DISCONNECTED. FAILED TO ESTABLISH CONNECTION!**" along with a disconnection event.

This command has a two-part response.
First the _"Trying to connect..."_ response is the acknowledgement that the command has been accepted and will follow immediately after the command has been run.

The *"handle_evt_gap_connected:..."* response will trigger afterwards when the operation is completed. If auto-discover services (ATDS) is not turned off then all services will be discovered as well and the event *"handle_evt_gattc_browse_completed:..."* will trigger when it's done.

Be aware that between the first and second response other events can trigger, for example, notifications or write operations.

Instead of using MAC Address (type and address) as a parameter it is possible to use the index shown in the scan results after running AT+GAPSCAN. (For verbose mode use the command ATSIV to show index in the scan result). The indexes will reset when doing a new scan.

Scan result Index Example:

```
AT+GAPSCAN=2
SCANNING...

[01] Device: [0]12:34:56:78:9A:BC  RSSI: -65

SCAN COMPLETE

AT+GAPCONNECT=1!
```

```
AT+GAPSCAN=2
{"C":9,"cmd":"AT+GAPSCAN=2"}
{"A":9,"err":0,"errMsg":"ok"}
{"R":9,"evt":{"action":"scanning"}}
{"E":9,"nol":4}
{"S":9,"indx":1,"rssi":-58,"addr":"[0]12:34:56:78:9A:BC"}
{"SE":9,"evt":{"action":"scan completed"}}
AT+GAPCONNECT=1!
```

Example:

```
AT+GAPCONNECT=[0]01:01:01:01:01:01=8:8:0:100:
Trying to connect...

CONNECTED.

Target conn indx changed to=0000 # (only shown on first connection)

handle_evt_gap_connected: conn_idx=0000 address=01:01:01:01:01:01 CI max is
24.

(discover and shows services, shows mtu and connection parameter changes...)
# (only discovers/shows services if auto discover services (ATDS) is on)
```

```
AT+GAPCONNECT=[0]00:00:00:00:00:00=8:8:0:100:
{"C":28,"cmd":"AT+GAPCONNECT=[0]00:00:00:00:00:00=8:8:0:100:"}
{"A":28,"err":0,"errMsg":"ok"}
{"R":28,"action":"connecting","addr":"00:00:00:00:00:00"}
{"E":28,"nol":4}
{256:"0000",{"action":"target conn index changed","target":"0000"}} # <---
Event (only shown on first connection)
{256:"0000",
{"action":"connected","addr":"00:00:00:00:00:00","CImin":24,"CImax":24}} # <-
-- Event
(discover and shows services, shows mtu and connection parameter changes...)
# (only discovers/shows services if auto discover services (ATDS) is on)
```

```
AT+GAPCONNECT=2!=8:8:0:100:
Trying to connect...

CONNECTED.

Target conn indx changed to=0000 # (only shown on first connection)

handle_evt_gap_connected: conn_idx=0000 address=01:01:01:01:01:01 CI max is
24.

(discover and shows services, shows mtu and connection parameter changes...)
# (only discovers/shows services if auto discover services (ATDS) is on)
```

```
AT+GAPCONNECT=2!=8:8:0:100:
{"C":28,"cmd":"AT+GAPCONNECT=2!=8:8:0:100:"}
{"A":28,"err":0,"errMsg":"ok"}
{"R":28,"action":"connecting","addr":"00:00:00:00:00:00"}
{"E":28,"nol":4}
```

```
{256:"0000",{"action":"target conn index changed","target":"0000"}} # <---
Event (only shown on first connection)
{256:"0000",
{"action":"connected","addr":"00:00:00:00:00:00","CImin":24,"CImax":24}} # <-
-- Event
(discover and shows services, shows mtu and connection parameter changes...)
# (only discovers/shows services if auto discover services (ATDS) is on)
```

AT+GAPDISCONNECT

- *Disconnects from a peer Bluetooth device. This command can be used in both central and peripheral role.*

Command	Syntax
AT+GAPDISCONNECT	AT+GAPDISCONNECT

Example:

```
AT+GAPDISCONNECT
handle_evt_gap_disconnected: conn_idx=0000 address=01:01:01:01:01:01.

DISCONNECTED.

Target conn indx changed to=0001 # (only shown if connected to multiple
devices)
```

```
AT+GAPDISCONNECT
{"C":26,"cmd":"AT+GAPDISCONNECT"}
{"A":26,"err":0,"errMsg":"ok"}
{"R":26,"disconnect_target":"0000"}
{"E":26,"nol":4}
{258:"0000",
{"action":"disconnected","addr":"00:00:00:00:00:00","reason":"16"}} # <---
Event
```

AT+GAPDISCONNECTALL

- *Disconnects from all connected peer Bluetooth devices. This command can be used in both central and peripheral role.*

Command	Syntax
AT+GAPDISCONNECTALL	AT+GAPDISCONNECTALL

This command has several responses.
First the *"Disconnecting all connected devices..."* response is the acknowledgement that the command has been accepted and will follow immediately after the command has been run.

The *"handle_evt_gap_disconnected:..."* response will trigger afterwards when a disconnection action is completed.

Lastly the *"All connections terminated."* response will trigger when all devices have been disconnected.

Be aware that between the different responses other events can trigger, for example, notifications or write operations.

Example:

```
AT+GAPDISCONNECTALL
Disconnecting all connected devices...
handle_evt_gap_disconnected: conn_idx=0000 address=01:01:01:01:01:01.

DISCONNECTED.

Target conn indx changed to=0001

handle_evt_gap_disconnected: conn_idx=0001 address=02:02:02:02:02:02.

DISCONNECTED.

Target conn indx changed to=0002

handle_evt_gap_disconnected: conn_idx=0002 address=03:03:03:03:03:03.

DISCONNECTED.

All connections terminated.
```

```
AT+GAPDISCONNECTALL
{"C":3,"cmd":"AT+GAPDISCONNECTALL"}
{"A":3,"err":0,"errMsg":"ok"}
{"R":3,"disconnect_target":"all"}
{"E":3,"nol":4}
{258:"0000",
{"action":"disconnected","addr":"00:00:00:00:00:00","reason":"16"}} # <---
Event
{258:"0000",{"action":"target conn index changed","target":"0001"}} # <---
Event
{258:"0001",
{"action":"disconnected","addr":"01:01:01:01:01:01","reason":"16"}} # <---
Event
{258:"0001",{"info":"all connections terminated"}} # <--- Event
```

AT+GAPIOCAP

- *Sets or queries what input and output capabilities the device has. Parameter is number between 0 to 4. Description of supported modes bellow.*

Command	Syntax
AT+GAPIOCAP	
	AT+GAPIOCAP=**x**
	or
	AT+GAPIOCAP

Supported modes:

Code	Mode
0:	(GAP_IO_CAP_DISP_ONLY) Display only
1:	(GAP_IO_CAP_DISP_YES_NO) Display + yes & no
2:	(GAP_IO_CAP_KEYBOARD_ONLY) Keyboard only
3:	(GAP_IO_CAP_NO_INPUT_OUTPUT) No input no output
4:	(GAP_IO_CAP_KEYBOARD_DISP) Keyboard + display

iocapabilities

Example:

```
AT+GAPIOCAP=4
OK

I/O CAP = GAP_IO_CAP_KEYBOARD_DISP
```

```
AT+GAPIOCAP
{"C":4,"cmd":"AT+GAPIOCAP"}
{"A":4,"err":0,"errMsg":"ok"}
{"R":4,"ioCap":"GAP_IO_CAP_NO_INPUT_OUTPUT"}
{"E":4,"nol":4}
```

```
AT+GAPIOCAP=4
{"C":10,"cmd":"AT+GAPIOCAP=4"}
{"A":10,"err":0,"errMsg":"ok"}
{"R":10,"ioCap":"GAP_IO_CAP_KEYBOARD_DISP"}
{"E":10,"nol":4}
```

AT+GAPPAIR

- *Starts a pairing (AT+GAPPAIR) or bonding procedure (AT+GAPPAIR=BOND). Depending on whether the device is master or slave on the connection, it will send a pairing or a security request respectively. Only usable when connected to a device.*

Command	Syntax
AT+GAPPAIR	
	AT+GAPPAIR=BOND
	or
	AT+GAPPAIR

> **Important!**
> **Firmware Ver <= 2.7.4:**
> BleuIO can save the bonding information of up to 8 devices. When 8 devices had been stored no new bonding can be done until the user manually removed one or all bonding information using the AT+GAPUNPAIR command.
> > **Firmware Ver >= 2.7.5:**
> BleuIO can save the bonding information of up to 16 devices. When 16 devices had been stored the user can still bond. The BleuIO will remove the bonding information of the first bonded device that was added to the list that isn't currently connected.
> Since BleuIO support up to 8 connections at the time there will always be at least 8 bonded devices that isn't currently connected. The first device that was bonded will be removed to make room for the new bonded device information.

Example:

```
AT+GAPPAIR
PAIRING...

handle_evt_gap_pair_completed: conn_idx=0000 status=0 bond=0 mitm=0 # <---
Event

PAIRING SUCCESS
```

```
AT+GAPPAIR=BOND
PAIRING...

handle_evt_gap_pair_completed: conn_idx=0000 status=0 bond=1 mitm=0 # <---
Event

PAIRING SUCCESS
BONDING SUCCESS

# Firmware Ver >= 2.7.5 if bonding list is full
AT+GAPPAIR=BOND
BOND LIST FULL. REMOVING FIRST BONDING INFO IN LIST TO MAKE ROOM.
40:48:FD:E8:92:FD UNPARIED.

PAIRING...

handle_evt_gap_pair_completed: conn_idx=0000 status=0 bond=1 mitm=0 # <---
Event
```

```
PAIRING SUCCESS
BONDING SUCCESS
```

```
AT+GAPPAIR
{"C":19,"cmd":"AT+GAPPAIR"}
{"A":19,"err":0,"errMsg":"ok"}
{"R":19,"action":"pairing"}
{"E":19,"nol":4}
{265:"0000","evt":{"action":"pair
completed","status":0,"bond":"false","mitm":"false"}} # <--- Event
```

```
AT+GAPPAIR=BOND
{"C":22,"cmd":"AT+GAPPAIR=BOND"}
{"A":22,"err":0,"errMsg":"ok"}
{"R":22,"action":"pairing"}
{"E":22,"nol":4}
{265:"0000","evt":{"action":"pair
completed","status":0,"bond":"true","mitm":"false"}} # <--- Event

# Firmware Ver >= 2.7.5 if bonding list is full
AT+GAPPAIR=BOND
{"C":15,"cmd":"AT+GAPPAIR=BOND"}
{"A":15,"err":0,"errMsg":"ok"}
{"R":15,"evt":{"action":"bond list full, removing first addr in
list","addr":"40:48:FD:E8:95:BC"}}
{"R":15,"evt":{"action":"pairing"}}
{"E":15,"nol":5}
{265:"0000","evt":{"action":"pair
completed","status":0,"bond":"true","mitm":"false"}} # <--- Event
```

AT+GAPUNPAIR

- *Unpair paired devices. This will also remove the device bond data from BLE storage. Usable both when device is connected and when not.*
  Either unpair all paired devices (AT+GAPUNPAIR) or selected paired device (AT+GAPUNPAIR=device_mac_address). Public= [0] or private= [1] address type prefix required before mac address.

Command	Syntax
AT+GAPUNPAIR	
	AT+GAPUNPAIR=***[x]xx:xx:xx:xx:xx:xx***
	or
	AT+GAPUNPAIR

Example:

```
AT+GAPUNPAIR
UNPARIED.
```

```
AT+GAPUNPAIR
{"C":20,"cmd":"AT+GAPUNPAIR"}
{"A":20,"err":0,"errMsg":"ok"}
{"R":20,"action":"unpaired"}
{"E":20,"nol":4}
{258:"0000",
{"action":"disconnected","addr":"00:00:00:00:00:00","reason":"16"}} # <---
Event
```

AT+GAPSCAN

- *Starts a Bluetooth device scan with or without timer set in seconds. Only accepted when device is in central role and not connected. The scan will continue indefinitely if no parameter is set or until the amount of time set is reached. Scanning aborts if user presses CTRL+C.*

Command	Syntax
AT+GAPSCAN	AT+GAPSCAN
	AT+GAPSCAN=***seconds***

Example:

```
AT+GAPSCAN
SCANNING...

[01] Device: [1]30:63:C5:D0:B1:DE RSSI: -38

[02] Device: [0]D0:76:50:80:0A:98 RSSI: -75(closebeacon.com)

[03] Device: [1]27:5D:B8:2E:96:B0 RSSI: -51

[04] Device: [1]5E:CE:CF:C5:20:BB RSSI: -84

SCAN COMPLETE
```

```
AT+GAPSCAN
{"C":23,"cmd":"AT+GAPSCAN"}
{"A":23,"err":0,"errMsg":"ok"}
{"R":23,"action":"scanning"}
{"E":23,"nol":4}
{"S":23,"rssi":-86,"addr":"[1]CC:EA:2B:D7:88:48"}
```

```
{"S":23,"rssi":-63,"addr":"[1]D9:CA:5D:68:D0:79"}
# Until stopped with Ctrl+C
```

```
AT+GAPSCAN=2
{"C":5,"cmd":"AT+GAPSCAN=2"}
{"A":5,"err":0,"errMsg":"ok"}
{"R":5,"action":"scanning"}
{"E":5,"nol":4}
{"S":5,"rssi":-75,"addr":"[1]F6:9B:76:F9:A7:AB"}
{"S":5,"rssi":-60,"addr":"[1]C6:33:E2:A7:03:A4"}
{"SE":5,"action":"scan completed"}
```

AT+GAPSTATUS

- *Reports the Bluetooth role.*

Command	Syntax
AT+GAPSTATUS	AT+GAPSTATUS

Example:

```
AT+GAPSTATUS
Central role

Not Connected

Not Advertising
```

```
AT+GAPSTATUS
{"C":6,"cmd":"AT+GAPSTATUS"}
{"A":6,"err":0,"errMsg":"ok"}
{"R":6,"gap_role:"dual"}
{"R":6,"connected":true,"noConns":1,"advertising":false}
{"E":6,"nol":5}
```

AT+GATTCREAD

- *Read attribute of remote GATT server. Can only be used in the Central role and when connected to a peripheral.*

Command	Syntax
AT+GATTCREAD	AT+GATTCREAD=***handle param***

This command has a two-part response.
First *"DATA WRITTEN..."* response is the acknowledgement that the command has been accepted and will

follow immediately after the command has been run.

The *"handle_evt_gattc_read_completed:..."* response will trigger afterwards when the operation is completed and will include a status code to signify if the operation was successful. (status=0 means OK, anything else means ERROR).

Be aware that between the first and second response other events can trigger, for example, notifications or connections.

Example:

```
AT+GATTCREAD=001B
OK

handle_evt_gattc_read_completed: conn_idx=0000 handle=001b status=0

Value read: HELLO
Hex: 0x48454C4C4F
Size: 5
```

```
AT+GATTCREAD=0003
{"C":2,"cmd":"AT+GATTCREAD=0003"}
{"A":2,"err":0,"errMsg":"ok"}
{"E":2,"nol":3}
{775:"0000","0003":{"readStatus":0,"ascii":"BleuIO","len":6}} # <--- Event
{775:"0000","0003":{"readStatus":0,"hex":"0x426C6575494F","len":6}} # <---
Event
```

AT+GATTCWRITE

- *Write attribute to remote GATT server in ASCII. Can only be used in the Central role and when connected to a peripheral.*

Command	Syntax
AT+GATTCWRITE	AT+GATTCWRITE=***(handle param) (msg)***

This command has a two-part response.
First *"DATA WRITTEN..."* response is the acknowledgement that the command has been accepted and will follow immediately after the command has been run.

The *"handle_evt_gattc_write_completed:..."* response will trigger afterwards when the operation is completed and will include a status code to signify if the operation was successful. (status=0 means OK, anything else means ERROR).

Be aware that between the first and second response other events can trigger, for example, notifications or connections.

Example:

```
AT+GATTCWRITE=001B HELLO
DATA WRITTEN: HELLO

handle_evt_gattc_write_completed: conn_idx=0000 handle=001b status=0
```

```
AT+GATTCWRITE=0014 TEST
{"C":4,"cmd":"AT+GATTCWRITE=0014 TEST"}
{"A":4,"err":0,"errMsg":"ok"}
{"R":4,"data":"TEST"}
{"E":4,"nol":4}
{776:"0000","0014":{"writeStatus":0}} # <--- Event
```

AT+GATTCWRITEB

- *Write attribute to remote GATT server in Hex. Can only be used in the Central role and when connected to a peripheral.*

Command	Syntax
AT+GATTCWRITEB	AT+GATTCWRITEB=***(handle param) (msg)***

This command has a two-part response.
First *"DATA WRITTEN..."* response is the acknowledgement that the command has been accepted and will follow immediately after the command has been run.

The *"handle_evt_gattc_write_completed:..."* response will trigger afterwards when the operation is completed and will include a status code to signify if the operation was successful. (status=0 means OK, anything else means ERROR).

Be aware that between the first and second response other events can trigger, for example, notifications or connections.

Example:

```
AT+GATTCWRITEB=001B 0101
DATA WRITTEN: 0101
Size: 2

handle_evt_gattc_write_completed: conn_idx=0000 handle=001b status=0
```

```
AT+GATTCWRITEB=0014 0101
{"C":5,"cmd":"AT+GATTCWRITEB=0014 0101"}
{"A":5,"err":0,"errMsg":"ok"}
{"R":5,"data":"0101"}
{"E":5,"nol":4}
{776:"0000","0014":{"writeStatus":0}}  # <--- Event
```

AT+GATTCWRITEWR

- *Write (without response) attribute to remote GATT server in ASCII. Can only be used in the Central role and when connected to a peripheral.*

Command	Syntax
AT+GATTCWRITEWR	AT+GATTCWRITEWR=***(handle param) (msg)***

This command has a two part response.
First *"DATA WRITTEN..."* response is the acknowledgement that the command has been accepted and will follow immediately after the command has been run.

The *"handle_evt_gattc_write_completed:..."* response will trigger afterwards when the operation is completed and will include a status code to signify if the operation was successful. (status=0 means OK, anything else means ERROR).

Be aware that between the first and second response other events can trigger, for example, notifications or connections.

Example:

```
AT+GATTCWRITEWR=001B HELLO
DATA WRITTEN: HELLO

handle_evt_gattc_write_completed: conn_idx=0000 handle=001b status=0
```

```
AT+GATTCWRITEWR=0014 TEST
{"C":9,"cmd":"AT+GATTCWRITEWR=0014 TEST"}
{"A":9,"err":0,"errMsg":"ok"}
{"R":9,"data":"TEST"}
{"E":9,"nol":4}
{776:"0000","0014":{"writeStatus":0}} # <--- Event
```

AT+GATTCWRITEWRB

- *Write (without response) attribute to remote GATT server in Hex. Can only be used in the Central role and when connected to a peripheral.*

Command	Syntax
AT+GATTCWRITEWRB	AT+GATTCWRITEWRB=***(handle param) (msg)***

This command has a two part response.
First *"DATA WRITTEN..."* response is the acknowledgement that the command has been accepted and will follow immediately after the command has been run.

The *"handle_evt_gattc_write_completed:..."* response will trigger afterwards when the operation is completed and will include a status code to signify if the operation was successful. (status=0 means OK, anything else means ERROR).

Be aware that between the first and second response other events can trigger, for example, notifications or connections.

Example:

```
AT+GATTCWRITEWRB=001B 0101
DATA WRITTEN: 0101
Size: 2

handle_evt_gattc_write_completed: conn_idx=0000 handle=001b status=0
```

```
AT+GATTCWRITEWRB=0014 0101
{"C":8,"cmd":"AT+GATTCWRITEWRB=0014 0101"}
{"A":8,"err":0,"errMsg":"ok"}
{"R":8,"data":"0101"}
{"E":8,"nol":4}
{776:"0000","0014":{"writeStatus":0}} # <--- Event
```

AT+GETBOND

- *Displays all MAC address of bonded devices.*

Command	Syntax
AT+GETBOND	AT+GETBOND

Example:

```
# If no bonded devices
AT+GETBOND
No Bonded Devices found.

# If bonded devices exist
AT+GETBOND
D0:76:50:80:0A:98
```

```
# If no bonded devices
AT+GETBOND
{"C":1,"cmd":"AT+GETBOND"}
{"A":1,"err":0,"errMsg":"ok"}
{"R":1,"bondedAddr":"none"}
{"E":1,"nol":4}

# If bonded devices exist
AT+GETBOND
```

```
{"C":2,"cmd":"AT+GETBOND"}
{"A":2,"err":0,"errMsg":"ok"}
{"R":2,"bondedAddr":"D0:76:50:80:0A:98"}
{"E":2,"nol":4}
```

AT+GETCONN

- *Gets a list of currently connected devices along with their mac addresses, connection index, our role towards this connection and if it's bonded/paired.*

Command	Syntax
AT+GETCONN	AT+GETCONN

Example:

```
AT+GETCONN
[0]00:08:0D:05:20:B0, conn_idx=0000, Our Role: Client, bonded=false,
paired=false
[0]00:01:0D:04:65:30, conn_idx=0001, Our Role: Server, bonded=true,
paired=true
```

```
AT+GETCONN
{"C":10,"cmd":"AT+GETCONN"}
{"A":10,"err":0,"errMsg":"ok"}
{"R":10,"0000":{"addr":"
[0]00:00:00:00:00:00","ourRole":"Client","bonded":false,"paired":false}}
{"E":10,"nol":4}
```

AT+GETMAC

Returns MAC address of the BleuIO device.

Command	Syntax
AT+GETMAC	AT+GETMAC

Example:

```
AT+GETMAC
OK
OWN MAC ADDRESS:00:00:00:00:00:28
```

```
AT+GETMAC
{"C":11,"cmd":"AT+GETMAC"}
{"A":11,"err":0,"errMsg":"ok"}
```

```
{"R":11,"own_mac_addr":"00:00:00:00:00:28"}
{"E":11,"nol":4}
```

AT+GETSERVICEDETAILS

- *Discovers all characteristics and descriptors of a selected service. Must run AT+GETSERVICESONLY command first to get the service handle.*

Command	Syntax
AT+GETSERVICEDETAILS	AT+GETSERVICEDETAILS=***(service handle param)***

This command has a two part response.
First the *"OK"* response is the acknowledgement that the command has been accepted and will follow immediately after the command has been run.

The *"handle_evt_gattc_browse_completed:..."* response will trigger afterwards when the operation is completed and will include a status code to signify if the operation was successful. (status=0 means OK, anything else means ERROR).

Be aware that between the first and second response other events can trigger, for example, notifications or connections.

Example:

```
AT+GETSERVICEDETAILS=0017
OK
        0017 serv 0783b03e-8535-b5a0-7140-a304d2495ce1
        0018 char 0783b03e-8535-b5a0-7140-a304d2495ce1 prop=10 (----N---)
        0019 ---- 0783b03e-8535-b5a0-7140-a304d2495ce2
        001a desc 0x2902
        001b desc 0x2901
        001c char 0783b03e-8535-b5a0-7140-a304d2495ce1 prop=04 (--X-----)
        001d ---- 0783b03e-8535-b5a0-7140-a304d2495ce3
        001e desc 0x2902
        001f desc 0x2901
        0020 char 0783b03e-8535-b5a0-7140-a304d2495ce1 prop=14 (--X-N---)
        0021 ---- 0783b03e-8535-b5a0-7140-a304d2495ce4
        0022 desc 0x2902
        0023 desc 0x2901

handle_evt_gattc_browse_completed: conn_idx=0000 status=0
```

```
AT+GETSERVICEDETAILS=0017
{"C":16,"cmd":"AT+GETSERVICEDETAILS=0017"}
{"A":16,"err":0,"errMsg":"ok"}
{"E":16,"nol":3}
```

AT+GETSERVICES

- *Rediscovers a peripheral's services and characteristics.*

Command	Syntax
AT+GETSERVICES	AT+GETSERVICES

This command has a two part response.
First, the *"OK"* response is the acknowledgement that the command has been accepted and will follow immediately after the command has been run.

The *"handle_evt_gattc_browse_completed:..."* response will trigger afterwards when the operation is completed and will include a status code to signify if the operation was successful. (status=0 means OK, anything else means ERROR).

Be aware that between the first and second response other events can trigger, for example, notifications or connections.

Example:

```
AT+GETSERVICES
OK
        0001 serv 0x1800
        0002 char 0x1800 prop=0a (-R-W----)
        0003 ---- 0x2a00
        0004 char 0x1800 prop=02 (-R------)
        0005 ---- 0x2a01
        0006 char 0x1800 prop=02 (-R------)
        0007 ---- 0x2a04
        0008 char 0x1800 prop=02 (-R------)
        0009 ---- 0x2aa6
        000a serv 0x1801
        000b char 0x1801 prop=20 (-----I--)
        000c ---- 0x2a05
        000d desc 0x2902
        000e serv 6e400001-b5a3-f393-e0a9-e50e24dcca9e
        000f char 6e400001-b5a3-f393-e0a9-e50e24dcca9e prop=0c (--XW----)
        0010 ---- 6e400002-b5a3-f393-e0a9-e50e24dcca9e
        0011 char 6e400001-b5a3-f393-e0a9-e50e24dcca9e prop=10 (----N---)
        0012 ---- 6e400003-b5a3-f393-e0a9-e50e24dcca9e
        0013 desc 0x2902

        handle_evt_gattc_browse_completed: conn_idx=0000 status=0
```

```
AT+GETSERVICES
{"C":13,"cmd":"AT+GETSERVICES"}
{"A":13,"err":0,"errMsg":"ok"}
{"E":13,"nol":3}
```

AT+GETSERVICESONLY

- *Discovers a peripherals services.*

Command	Syntax
AT+GETSERVICESONLY	AT+GETSERVICESONLY

This command has a two part response.
First, the *"OK"* response is the acknowledgement that the command has been accepted and will follow immediately after the command has been run.

The *"handle_evt_gattc_discover_completed:..."* response will trigger afterwards when the operation is completed and will include a status code to signify if the operation was successful. (status=0 means OK, anything else means ERROR).

Be aware that between the first and second response other events can trigger, for example, notifications or connections.

Example:

```
AT+GETSERVICESONLY
OK
        0001 serv 0x1800
        000a serv 0x1801
        000b serv 0783b03e-8535-b5a0-7140-a304d2495cb7
        0017 serv 0783b03e-8535-b5a0-7140-a304d2495ce1
        0024 serv 0x180a

handle_evt_gattc_discover_completed: conn_idx=0000 type=SVC status=0
```

```
AT+GETSERVICESONLY
{"C":15,"cmd":"AT+GETSERVICESONLY"}
{"A":15,"err":0,"errMsg":"ok"}
{"E":15,"nol":3}
```

AT+INDI

- *Shows list of set indication handles.*

Command	Syntax
AT+INDI	AT+INDI

Shows list of set indication handles along with the connection index so you can see what indication you have enabled on which connected device.

Example:

```
AT+INDI
(conn_idx=0000 indi_handle=0012)
```

```
(conn_idx=0001 indi_handle=001F)
```

```
AT+INDI
{"C":2,"cmd":"AT+INDI"}
{"A":2,"err":0,"errMsg":"ok"}
{"R":2,"resp":"No indications set"}
{"E":2,"nol":4}
```

AT+LED Pro

- *Controls the LED.*

Command	Syntax
AT+LED	AT+LED=**on/off control parameters**
	AT+LED=**T**
	AT+LED=T=**On intv**=**Off intv**

Control Parameters

Parameters	Meaning
on/off control parameters	0 = off, 1 = on
T	Toggle. Without other parameters turns on default toggle, toggling on/off every 500ms
On intv	Decideds how long in ms the LED should be turned on.
Off intv	Decideds how long in ms the LED should be turned off.

Example:

```
AT+LED=0
OK
AT+LED=T
OK
AT+LED=T=1000=2000
OK
```

```
AT+LED=0
{"C":16,"cmd":"AT+LED=0"}
{"A":16,"err":0,"errMsg":"ok"}
{"E":16,"nol":3}
AT+LED=T
{"C":17,"cmd":"AT+LED=T"}
```

```
{"A":17,"err":0,"errMsg":"ok"}
{"E":17,"nol":3}
AT+LED=T=1000=2000
{"C":18,"cmd":"AT+LED=T=1000=2000"}
{"A":18,"err":0,"errMsg":"ok"}
{"E":18,"nol":3}
```

AT+MTU

- *Sets or queries the max allowed MTU size. Minimum allowed value: 67, Maximum allowed value: 512 Cannot be changed while connected/connecting or advertising.*

Command	Syntax
AT+MTU	
	AT+MTU
	or
	AT+MTU=***max mtu size***

Example:

```
AT+MTU=300
OK


AT+MTU
OK
Max MTU Size: 300
```

```
AT+MTU=300
{"C":0,"cmd":"AT+MTU=300"}
{"A":0,"err":0,"errMsg":"ok"}
{"E":0,"nol":3}
```

```
AT+MTU
{"C":1,"cmd":"AT+MTU"}
{"A":1,"err":0,"errMsg":"ok"}
{"R":1,"maxMTUsize":"300"}
{"E":1,"nol":4}
```

AT+NOTI

- *Shows list of set notification handles.*

Command	Syntax
AT+NOTI	AT+NOTI

Shows list of set notification handles along with the connection index so you can see what notification you have enabled on which connected device.

Example:

```
AT+NOTI
(conn_idx=0000 noti_handle=000D)
(conn_idx=0000 noti_handle=0014)
(conn_idx=0000 noti_handle=0019)
(conn_idx=0000 noti_handle=0021)
```

```
AT+NOTI
{"C":17,"cmd":"AT+NOTI"}
{"A":17,"err":0,"errMsg":"ok"}
{"R":17,"0000":{"notiHandle":"000D"}}
{"R":17,"0000":{"notiHandle":"0014"}}
{"R":17,"0000":{"notiHandle":"0019"}}
{"R":17,"0000":{"notiHandle":"0021"}}
{"E":17,"nol":7}
```

AT+NUMCOMPA

- *Used for accepting a numeric comparison authentication request or enabling/disabling auto-accepting numeric comparisons. AT+NUMCOMPA=0 for disabled and AT+NUMCOMPA=1 for enable. Enabled by default.*

Command	Syntax
AT+NUMCOMPA	
	AT+NUMCOMPA=<0 or 1>
	or
	AT+NUMCOMPA

Example:

```
AT+NUMCOMPA=0
NUMERIC COMPARISON AUTO-ACCEPT OFF
```

```
AT+NUMCOMPA=0
{"C":17,"cmd":"AT+NUMCOMPA=0"}
```

```
{"A":17,"err":0,"errMsg":"ok"}
{"R":17,"autoNumComp":false}
{"E":17,"nol":4}
```

```
AT+NUMCOMPA=1
{"C":18,"cmd":"AT+NUMCOMPA=1"}
{"A":18,"err":0,"errMsg":"ok"}
{"R":18,"autoNumComp":true}
{"E":18,"nol":4}
```

```
{276:"0000","action":"numeric request","numericKey":"972226"} # <--- Event
{276:"0000","info":"run AT+NUMCOMPA to accept"} # <--- Event
AT+NUMCOMPA
{"C":14,"cmd":"AT+NUMCOMPA"}
{"A":14,"err":0,"errMsg":"ok"}
{"E":14,"nol":3}
```

AT+PERIPHERAL

- *Sets the device Bluetooth role to peripheral. Any connection must be terminated before the role change is accepted.*

Command	Syntax
AT+PERIPHERAL	AT+PERIPHERAL

Example:

```
AT+PERIPHERAL
OK
```

```
AT+PERIPHERAL
{"C":21,"cmd":"AT+PERIPHERAL"}
{"A":21,"err":0,"errMsg":"ok"}
{"E":21,"nol":3}
```

AT+SCANFILTER

- *Sets or queries the scanfilter. There are three types of scanfilter, filter by name, filter by uuid or by manufacturer specific ID.*

Command	Syntax
AT+SCANFILTER	

Command	Syntax
	AT+SCANFILTER
	AT+SCANFILTER=NAME=***name***
	AT+SCANFILTER=UUID=***XXXX*** or AT+SCANFILTER=UUID=***xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx***
	AT+SCANFILTER=MFSID=***XXXX***
	AT+SCANFILTER=CLEAR

AT+SCANFILTER can take these parameters:

Parameter	Description	Input Value	AD Flags checked for data
NAME	Filter data that includes set local name.	Device Name (ASCII)	«Shortened Local Name» (0x08)
			«Complete Local Name» (0x09)
UUID	Filter data that includes set service UUID.	16-bit UUID or 128-bit UUID (Hex Big Endian)	«Incomplete List of 16-bit Service UUIDs» (0x02)
			«Complete List of 16-bit Service UUIDs» (0x03)
			«Incomplete List of 128-bit Service UUIDs» (0x06)
			«Complete List of 128-bit Service UUIDs» (0x07)
			«Service Data - 16 bit UUID» (0x16)
			«Service Data - 128 bit UUID» (0x21)
MFSID	Filter data that includes set Manufacturer Specific ID.	16-bit Company Identifier Code (Hex Big Endian)	«Manufacturer Specific Data» (0xFF)
CLEAR	Clears all set filters.		

Input Value	Format	Example:	Case-sensitive?
Device Name	String	BleuIO	YES
128-bit UUID	xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx	fa284580-9d88-4b70-9775-ce4786a6bb96	NO
16-bit UUID*	XXXX	1801	NO

Input Value	Format	Example:	Case-sensitive?
16-bit Company Identifier Code*	XXXX	075B	NO

* *Value FFFF is invalid for 16-bit UUID and/or 16-bit Company Identifier Code*

All the filters work for AT+FINDSCANDATA command results. All filters can be active at once, or two in any combination, or just a single one. For AT+GAPSCAN command results only the NAME filter works. No filters will be active on AT+TARGETSCAN= command results.

> **NOTE:** For AT+FINDSCANDATA command results only the Advertising packet or the Scan response packet that includes the data set by the filter will be shown. Example: If AT+SCANFILTER=NAME=BlueIO is set and the local name BleuIO is only found in the scan response packet only the scan response packet will be shown.

Example:

```
AT+SCANFILTER=NAME=BleuIO
OK

AT+SCANFILTER=UUID=0783b03e-8535-b5a0-7140-a304d2495cb7
OK

AT+SCANFILTER=MFSID=075B
OK

AT+SCANFILTER
SCANFILTER=NAME= BleuIO

SCANFILTER=MFSID= 075B

SCANFILTER=UUID= 0783b03e-8535-b5a0-7140-a304d2495cb7

AT+SCANFILTER=CLEAR
OK

AT+SCANFILTER
SCANFILTER=NAME Not set!

SCANFILTER=MFSID Not set!

SCANFILTER=UUID Not set!
```

```
AT+SCANFILTER=NAME=BleuIO
{"C":0,"cmd":"AT+SCANFILTER=NAME=BleuIO"}
{"A":0,"err":0,"errMsg":"ok"}
{"E":0,"nol":3}
```

```
AT+SCANFILTER=UUID=0783b03e-8535-b5a0-7140-a304d2495cb7
{"C":1,"cmd":"AT+SCANFILTER=UUID=0783b03e-8535-b5a0-7140-a304d2495cb7"}
{"A":1,"err":0,"errMsg":"ok"}
{"E":1,"nol":3}
```

```
AT+SCANFILTER=MFSID=075B
{"C":2,"cmd":"AT+SCANFILTER=MFSID=075B"}
{"A":2,"err":0,"errMsg":"ok"}
{"E":2,"nol":3}
```

```
AT+SCANFILTER
{"C":3,"cmd":"AT+SCANFILTER"}
{"A":3,"err":0,"errMsg":"ok"}
{"R":3,"scanfilter":"NAME","set":"true","value":"BleuIO"}
{"R":3,"scanfilter":"MFSID","set":"true","value":"075B"}
{"R":3,"scanfilter":"UUID","set":"true","value":"0783b03e-8535-b5a0-7140-
a304d2495cb7"}
{"E":3,"nol":6}
```

```
AT+SCANFILTER=CLEAR
{"C":4,"cmd":"AT+SCANFILTER=CLEAR"}
{"A":4,"err":0,"errMsg":"ok"}
{"E":4,"nol":3}
AT+SCANFILTER
{"C":5,"cmd":"AT+SCANFILTER"}
{"A":5,"err":0,"errMsg":"ok"}
{"R":5,"scanfilter":"NAME","set":"false"}
{"R":5,"scanfilter":"MFSID","set":"false"}
{"R":5,"scanfilter":"UUID","set":"false"}
{"E":5,"nol":6}
```

AT+SCANPARAM

- *Set or queries the scan parameters used.*

Command	Syntax
AT+SCANPARAM	AT+SCANPARAM=***scan mode***=***scan type***=***scan intv ms***=***scan win ms***=***filt dupl***
	AT+SCANPARAM

Scan Mode

Acceptable Value	Meaning

Acceptable Value	Meaning
0	General-Discoverable mode
2	Observer mode (default)

In General-discoverable mode the scan will stop after 15 seconds while in Observer mode it continues until stopped.

Scan Type

Acceptable Value	Meaning
0	Active Scan type (default)
1	Passive Scan type

Passive scan only scans Advertising Data.
Active scans Advertising Data and requests Scan Response Data.

Scan interval and window

Scan Interval is the duration of time between scans.
Scan Window determines how long to scan for at each interval.
The Scan Window parameter must be equal or smaller than the Scan Interval.

The scan interval and window can be set in steps of 0.625ms. Allowed values for interval span in the range of 2.5ms to 10240ms. This command set the scan parameters used for scans used by the commands **AT+GAPSCAN**, **AT+FINDSCANDATA** and **AT+SCANTARGET**.

Return scan window and scan interval values are in units of 0.625ms and can be converted to ms like this:
return value * 0.625
So for example 160 * 0.625 = 100ms
and 320 * 0.625 = 200ms

Filter Duplicate

Acceptable Value	Meaning
0	Off
1	On (Displays one advertising data and/or scan response packet per MAC address. Up to 255 unique addresses.)

Example:

```
AT+SCANPARAM=2=0=200=100=0
OK
AT+SCANPARAM
OK
Scan Mode = 2
Scan Type = 0
Scan Interval = 320
Scan Window = 160
Scan Filter Duplicate = false
```

```
AT+SCANPARAM=2=0=200=100=0
{"C":1,"cmd":"AT+SCANPARAM=2=0=200=100=0"}
{"A":1,"err":0,"errMsg":"ok"}
{"E":1,"nol":3}
AT+SCANPARAM
{"C":2,"cmd":"AT+SCANPARAM"}
{"A":2,"err":0,"errMsg":"ok"}
{"R":2,"evt":{"action":"getting scan param"}}
{"R":2,"scan_mode":2,"scan_type":0,"scan_intv":320,"scan_win":160,"filt_dupl"
:"false"}
{"E":2,"nol":5}
```

AT+SCANTARGET

- *Scan a target device for infinite time unless an optional scan timer parameter is set. Displaying the advertising and response data as it updates. Can scan up to 3 devices at once. Separate addresses with a semicolon ';'.*
  Scan can be stopped at any time by sending CTRL+C (0x03).

Command	Syntax
AT+SCANTARGET	AT+SCANTARGET=***[addr_type]slave_address***
AT+SCANTARGET	AT+SCANTARGET=***[addr_type]slave_address***=***seconds***

Example:

```
AT+SCANTARGET=[0]00:00:00:00:00:01
SCANNING...

[00:00:00:00:00:01] Device Data [ADV]: DataXYZ

[00:00:00:00:00:01] Device Data [RESP]: DataXYZ

[00:00:00:00:00:01] Device Data [ADV]: DataXYZ

[00:00:00:00:00:01] Device Data [RESP]: DataXYZ

[00:00:00:00:00:01] Device Data [ADV]: DataXYZ
```

```
AT+SCANTARGET=[0]00:00:00:00:00:00
{"C":23,"cmd":"AT+SCANTARGET=[0]00:00:00:00:00:00"}
{"A":23,"err":0,"errMsg":"ok"}
```

```
{"R":23,"action":"scanning"}
{"E":23,"nol":4}
{"ST":23,"addr":"00:00:00:00:00:00","type":0,"data":"02010603FF5B071107B75C49
D204A34071A0B535853EB08307000000000000"}
{"ST":23,"addr":"00:00:00:00:00:00","type":4,"data":"1109426C6575494F00000000
000000000000000000000000000000000000"}
# Until stopped with Ctrl+C
```

```
AT+SCANTARGET=[0]00:00:00:00:00:00=2
{"C":24,"cmd":"AT+SCANTARGET=[0]00:00:00:00:00:00=2"}
{"A":24,"err":0,"errMsg":"ok"}
{"R":24,"action":"scanning"}
{"E":24,"nol":4}
{"ST":24,"addr":"00:00:00:00:00:00","type":0,"data":"02010603FF5B071107B75C49
D204A34071A0B535853EB08307000000000000"}
{"ST":24,"addr":"00:00:00:00:00:00","type":4,"data":"1109426C6575494F00000000
000000000000000000000000000000000000"}
{"SE":24,"action":"scan completed"}
```

AT+SECLVL

- *Sets or queries what minimum security level will be used when connected to other devices.*
  When used while connected will trigger a pairing request. When used while not connected, sets what minimum security level the dongle will demand when connecting to another device. If the other device does not meet the security requirement, the dongle will disconnect.
  Querying security level, while connected, will show actual security level. While not connected, querying will show what minimum security level has been set that the dongle will require when connecting to another device. Note that what security level is supported is dependent on what Input/Output capability is set. See the table for more information.

Input/Output Cababilities and supported security levels:

IO CAP	DISPLAY ONLY	DISPLAY, YESNO	KEYBOARD ONLY	NO INPUT/OUTPUT	KEYBOARD, DISPLAY
DISPLAY ONLY	(1) (2)	(1) (2)	(1) (3) (4)	(1) (2)	(1) (4)
DISPLAY, YESNO	(1) (2)	(1) (2) (4)	(1) (3) (4)	(1) (2)	(1) (4)
KEYBOARD ONLY	(1) (3) (4)	(1) (3) (4)	(1) (4)	(1) (2)	(1) (4)
NO INPUT/OUTPUT	(1) (2)	(1) (2)	(1) (2)	(1) (2)	(1) (2)
KEYBOARD, DISPLAY	(1) (4)	(1) (4)	(1) (4)	(1) (2)	(1) (4)

BLE pairing summary

source : Bluetooth® Low Energy Security Modes and Procedures"

Supported modes:

Code	Mode
1:	(Security Level 1) No security (No authentication and no encryption).
2:	(Security Level 2) Unauthenticated pairing with encryption.
3:	(Security Level 3) Authenticated pairing with encryption.
4:	(Security Level 4) Authenticated LE Secure Connections pairing with encryption.

Setting the security level to 2 or higher will initiate a pairing automatically if connected or when connecting to another device.

Command	Syntax
AT+SECLVL	
	AT+SECLVL=**x**
	or
	AT+SECLVL

Example:

```
AT+SECLVL=2
OK
```

```
AT+SECLVL
{"C":25,"cmd":"AT+SECLVL"}
{"A":25,"err":0,"errMsg":"ok"}
{"R":25,"setSecLvl":1}
{"E":25,"nol":4}
```

```
AT+SECLVL=2
{"C":26,"cmd":"AT+SECLVL=2"}
{"A":26,"err":0,"errMsg":"ok"}
{"E":26,"nol":3}
```

ATSIV

- *Turns showing verbose scan result index on/off. (Off per default). ATSIV reads current ATSIV settings.*

Command	Syntax

Command	Syntax
ATSIV	ATSIV
	ATSIV0
	ATSIV1

Example:

```
ATSIV
Show Verbose Scan Index Off!
ATSIV1
Show Verbose Scan Index On!
ATSIV0
Show Verbose Scan Index Off!
```

```
ATSIV
{"C":119,"cmd":"ATSIV"}
{"A":119,"err":0,"errMsg":"ok"}
{"R":119,"showScanIndex":false}
{"E":119,"nol":4}
ATSIV1
{"C":120,"cmd":"ATSIV1"}
{"A":120,"err":0,"errMsg":"ok"}
{"R":120,"showScanIndex":true}
{"E":120,"nol":4}
ATSIV0
{"C":121,"cmd":"ATSIV0"}
{"A":121,"err":0,"errMsg":"ok"}
{"R":121,"showScanIndex":false}
{"E":121,"nol":4}
```

ATSRA

- *Turns showing resolved addr in scan results on/off. (Off per default). ATSRA reads current ATSRA settings.*

Command	Syntax
ATSRA	ATSRA
	ATSRA0
	ATSRA1

Example:

```
ATSRA
Show Resolved Addresses Off!
```

```
ATSRA1
Show Resolved Addresses On!
ATSRA0
Show Resolved Addresses Off!
```

```
ATSRA
{"C":122,"cmd":"ATSRA"}
{"A":122,"err":0,"errMsg":"ok"}
{"R":122,"showResolvedAddresses":false}
{"E":122,"nol":4}
ATSRA1
{"C":123,"cmd":"ATSRA1"}
{"A":123,"err":0,"errMsg":"ok"}
{"R":123,"showResolvedAddresses":true}
{"E":123,"nol":4}
ATSRA0
{"C":124,"cmd":"ATSRA0"}
{"A":124,"err":0,"errMsg":"ok"}
{"R":124,"showResolvedAddresses":false}
{"E":124,"nol":4}
```

ATSAT

- *Turns on/off showing address types in scan results from AT+FINDSCANDATA and AT+SCANTARGET scans. (Off per default). ATSAT reads current ATSAT settings.*

Command	Syntax
ATSAT	ATSAT
	ATSAT0
	ATSAT1

AT+FINDSCANDATA results with ATSAT on

```
[[1]DD:9E:C5:BF:F6:8E] Device Data [ADV]:
0201061BFF5B0705040578E03D001527B800D301B9010000000000000001B602
```

```
{"SF":6,"addr":"
[1]DD:9E:C5:BF:F6:8E","type":0,"data":"0201061BFF5B0705040578E045001427B700D3
01A70000000000000001B503"}
```

AT+SCANTARGET results with ATSAT on

```
[[1]DD:9E:C5:BF:F6:8E] Device Data [ADV]:
0201061BFF5B0705040578E04A001427B700D30197010000000000000001B801
```

```
{"ST":7,"addr":"
[1]DD:9E:C5:BF:F6:8E","type":0,"data":"0201061BFF5B0705040578E04F001427B700D3
0197010000000000000001B601"}
```

Example:

```
ATSAT
SHOW ADDRESS TYPE OFF
ATSAT1
SHOW ADDRESS TYPE ON
ATSAT0
SHOW ADDRESS TYPE OFF
```

```
ATSAT
{"C":126,"cmd":"ATSAT"}
{"A":126,"err":0,"errMsg":"ok"}
{"R":126,"showAddrType":false}
{"E":126,"nol":4}
ATSAT1
{"C":127,"cmd":"ATSAT1"}
{"A":127,"err":0,"errMsg":"ok"}
{"R":127,"showAddrType":true}
{"E":127,"nol":4}
ATSAT0
{"C":128,"cmd":"ATSAT0"}
{"A":128,"err":0,"errMsg":"ok"}
{"R":128,"showAddrType":false}
{"E":128,"nol":4}
```

AT+SUOTASTART

- *Enables the SUOTA Service and start the SUOTA Advertising.*
  Cannot be started while connected/connecting or advertising.

Command	Syntax
AT+SUOTASTART	AT+SUOTASTART

Enables the SOUTA service and starts the SOUTA Advertising so that the user can update the BleuIO firmware over BLE (using App or another BleuIO Dongle with script).
Adverting and Scan Response Data can not be changed while SUOTA is enabled. Advertising cannot be stopped using AT+ADVSTOP.

To disable the SUOTA service and stop the SUOTA Advertising use the command AT+SUOTASTOP.

After a successful update the BleuIO Dongle will reset and the updateded firmware will start.

Example:

```
AT+SUOTASTART
OK
SUOTA STARTED...
```

```
AT+SUOTASTART
{"C":0,"cmd":"AT+SUOTASTART"}
{"A":0,"err":0,"errMsg":"ok"}
{"R":0,"info":"suota started"}
{"E":0,"nol":4}
```

AT+SUOTASTOP

- *Disables the SUOTA Service and stops the SUOTA Advertising.*
  Cannot be used while connected/connecting.

Command	Syntax
AT+SUOTASTOP	AT+SUOTASTOP

Example:

```
AT+SUOTASTOP
OK
SUOTA STOPPED.
```

```
AT+SUOTASTOP
{"C":1,"cmd":"AT+SUOTASTOP"}
{"A":1,"err":0,"errMsg":"ok"}
{"R":1,"info":"suota stopped"}
{"E":1,"nol":4}
```

AT+SERVER

- *Only usable in Dual role. Sets the dongle role towards the targeted connection to server.*

Command	Syntax
AT+SERVER	AT+SERVER

Example:

```
AT+SERVER
OK
```

```
AT+SERVER
{"C":29,"cmd":"AT+SERVER"}
{"A":29,"err":0,"errMsg":"ok"}
{"E":29,"nol":3}
```

AT+SETAUTOEXECPWD

- *Create/set a password to protect access to AUTOEXEC list. Password persists through power cycles. Updating firmware will clear password. Can also be used to change existing password (change will only pass if old password is entered). (Max password length 255 ASCII characters)*

Command	Syntax
AT+SETAUTOEXECPWD	
	AT+SETAUTOEXECPWD=**xxxxxxx..**

The intended way to use this feature would be to add commands to the AUTOEXEC list as normal then lock it by setting a password using the AT+SETAUTOEXECPWD command. If AUTOEXEC list needs to be modified afterwards use AT+CLRAUTOEXECPWD to clear the password, modify the AUTOEXEC list then lock it again by using the AT+SETAUTOEXECPWD command.

When a password is set the following commands will require a password to be entered before they are executed: AT+CLRAUTOEXECPWD AT+SETAUTOEXECPWD AT+AUTOEXEC AT+AUTOEXECLOOP AT+CLRAUTOEXEC

Example:

```
AT+SETAUTOEXECPWD=my_password
OK
AUTOEXEC PASSWORD SET!
```

```
AT+SETAUTOEXECPWD=my_password
{"C":131,"cmd":"AT+SETAUTOEXECPWD=my_password"}
{"A":131,"err":0,"errMsg":"ok"}
{"R":131,"evt":{"action":"autoexec password set"}}
{"E":131,"nol":4}
```

AT+SETDIS

- *Sets the Device Information Service information.*

Command	Syntax
AT+SETDIS	AT+SETDIS=<man_name>=<model_num>=<serial_num>=<hw_rev>=<fw_rev>=<sw_rev>

Sets information used in the Device Information Servie (DIS). Information can only be set before starting advertising. If no custom information is set, the default BleuIO device information will be used. Once advertising is started the information set to be used will be locked in and cannot be changed during runtime.

To change the device information again the dongle will need to be restarted, either by unplugging it and plugging it back in, or by using the ATR command.

Max length is 100 characters per parameter.

Example:

```
AT+SETDIS=MAN_NAME=MOD_NUM=SERIAL=HW_REV=FW_REV=SW_REV
DIS Service info is set!
```

```
AT+SETDIS=MAN_NAME=MOD_NUM=SERIAL=HW_REV=FW_REV=SW_REV
{"C":0,"cmd":"AT+SETDIS=MAN_NAME=MOD_NUM=SERIAL=HW_REV=FW_REV=SW_REV"}
{"A":0,"err":0,"errMsg":"ok"}
{"R":0,"action":"DIS service info is set"}
{"E":0,"nol":4}
```

AT+SETINDI

- *Enable indication for selected characteristic.*

Command	Syntax
AT+SETINDI	AT+SETINDI=***(indication handle)***

Which is the indication handle?

```
0011 char 6e400001-b5a3-f393-e0a9-e50e24dcca9e prop=20 (-----I--) # <---------
The I means that the Characteristic Properties has Indication
0012 ---- 6e400003-b5a3-f393-e0a9-e50e24dcca9e  # <--------- This (0012) is
the handle you want to send in.
0013 desc 0x2902
```

- *If the status=0 in handle_evt_gattc_write_completed string, that means you have successfully written to it.*

Example of a indication response

```
handle_evt_gattc_indication: conn_idx=0000 handle=0039 length=5

Value received: Hello # <---- ascii value (can be shown/hidden with the ATA
command)
Hex: 0x48656C6C6F # <---- hex value
Size: 5
```

```
{778:"0000","0044":{"ascii":"TEST","len":4}} # <---- ascii value (can be
shown/hidden with the ATA command)
{778:"0000","0044":{"hex":"0x54455354","len":4}}
```

Example:

```
AT+SETINDI=0012
handle_evt_gattc_write_completed: conn_idx=0000 handle=0013 status=0
```

```
AT+SETINDI=0021
{"C":4,"cmd":"AT+SETINDI=0021"}
{"A":4,"err":0,"errMsg":"ok"}
{"E":4,"nol":3}
{776:"0000","0022":{"writeStatus":0}} # <--- Event
```

AT+SETNOTI

- *Enable notification for selected characteristic.*

Command	Syntax
AT+SETNOTI	AT+SETNOTI=***(notification handle)***

Which is the notification handle?

```
0011 char 6e400001-b5a3-f393-e0a9-e50e24dcca9e prop=10 (----N---) # <--------
The N means that the Characteristic Properties has Notify
0012 ---- 6e400003-b5a3-f393-e0a9-e50e24dcca9e  # <--------- This (0012) is
the handle you want to send in.
0013 desc 0x2902
```

- *If the status=0 in handle_evt_gattc_write_completed string, that means you have successfully written to it.*

Example of a notification response

```
handle_evt_gattc_notification: conn_idx=0000 handle=0039 length=5

Value received: Hello # <---- ascii value (can be shown/hidden with the ATA command)
Hex: 0x48656C6C6F # <---- hex value
Size: 5
```

```
{777:"0000","0047":{"ascii":"TEST","len":4}} # <---- ascii value (can be shown/hidden with the ATA command)
{777:"0000","0047":{"hex":"0x54455354","len":4}}
```

Example:

```
AT+SETNOTI=0012
handle_evt_gattc_write_completed: conn_idx=0000 handle=0013 status=0
```

```
AT+SETNOTI=0021
{"C":8,"cmd":"AT+SETNOTI=0021"}
{"A":8,"err":0,"errMsg":"ok"}
{"E":8,"nol":3}
{776:"0000","0022":{"writeStatus":0}} # <--- Event
```

AT+SETPASSKEY

- *Setting or quering set passkey for passkey authentication.*

Command	Syntax
AT+SETPASSKEY	
	AT+SETPASSKEY=***xxxxxx***
	or
	AT+SETPASSKEY

Example:

```
AT+SETPASSKEY=123456
PASSKEY: 123456

OK
```

```
AT+SETPASSKEY=222222
{"C":1,"cmd":"AT+SETPASSKEY=222222"}
{"A":1,"err":0,"errMsg":"ok"}
{"R":1,"passkey":"222222"}
{"E":1,"nol":4}
```

```
AT+SETPASSKEY
{"C":5,"cmd":"AT+SETPASSKEY"}
{"A":5,"err":0,"errMsg":"ok"}
{"R":5,"passkey":"222222"}
{"E":5,"nol":4}
```

AT+SETUOI

- *Set Unique Organization ID. It will be stored in flash memory, and will persist through power cycles. If set, the Unique Organization ID string will be displayed in the ATI command's response. Will clear any previous set Unique Organization ID when set. Max length: 100 characters.*

Command	Syntax
AT+SETUOI	
	AT+SETUOI=**id_string...**

Example:

```
AT+SETUOI=Your Unique Organization ID
OK
ATI
Smart Sensor Devices
DA14683 (P25Q80LE)
BleuIO
Firmware Version: 2.7.6


Peripheral role


Not Connected


Advertising


Your Unique Organization ID
```

```
AT+SETUOI=Your Unique Organization ID
{"C":0,"cmd":"AT+SETUOI=Your Unique Organization ID"}
{"A":0,"err":0,"errMsg":"ok"}
{"E":0,"nol":3}
ATI
{"C":1,"cmd":"ATI"}
{"A":1,"err":0,"errMsg":"ok"}
{"R":1,"dev":"Smart Sensor Devices","hw":"DA14683
(P25Q80LE)","name":"BleuIO"}
{"R":1,"fwVer":"2.7.6","gap_role":"peripheral"}
{"R":1,"connected":false,"advertising":true}
{"R":1,"uoi":"Your Unique Organization ID"}
{"E":1,"nol":7}
```

AT+SHOWRSSI

- *Shows(1)/hides(0) RSSI in AT+FINDSCANDATA and AT+SCANTARGET scans. Hides RSSI by default. AT+SHOWRSSI reads current AT+SHOWRSSI settings.*

Command	Syntax
AT+SHOWRSSI	AT+SHOWRSSI
	AT+SHOWRSSI=**0 or 1**

Example:

```
AT+SHOWRSSI
OK
SHOW RSSI OFF
AT+SHOWRSSI=1
OK
SHOW RSSI ON
AT+SHOWRSSI=0
OK
SHOW RSSI OFF
```

```
AT+SHOWRSSI
{"C":0,"cmd":"AT+SHOWRSSI"}
{"A":0,"err":0,"errMsg":"ok"}
{"R":0,"showRssi":false}
{"E":0,"nol":4}
AT+SHOWRSSI=1
{"C":1,"cmd":"AT+SHOWRSSI=1"}
{"A":1,"err":0,"errMsg":"ok"}
{"R":1,"showRssi":true}
{"E":1,"nol":4}
AT+SHOWRSSI=0
{"C":2,"cmd":"AT+SHOWRSSI=0"}
```

```
{"A":2,"err":0,"errMsg":"ok"}
{"R":2,"showRssi":false}
{"E":2,"nol":4}
```

AT+SPSSEND

- *Send a message or data via the SPS profile. Without parameters it opens a stream for continiously sending data.*

Command	Syntax
AT+SPSSEND	AT+SPSSEND=***(message)***
	AT+SPSSEND

Example:

```
AT+SPSSEND=HELLO
[SENT]
```

```
AT+SPSSEND=TEST
{"C":11,"cmd":"AT+SPSSEND=TEST"}
{"A":11,"err":0,"errMsg":"ok"}
{"R":11,"target":"0000"}
{"E":11,"nol":4}
{776:"0000","SPS":{"writeStatus":0}} # <--- Event
```

```
AT+SPSSEND
{"C":12,"cmd":"AT+SPSSEND"}
{"A":12,"err":0,"errMsg":"ok"}
{"R":12,"action":"streaming","info":"to abort press ESC-key"}
{"E":12,"nol":4}

{"R":12,"action":"not streaming"}
```

```
# Recieved as Client
{777:"0000","SPS":{"ascii":"TEST","len":5}}
```

```
# Recieved as Server
{279:"0000","SPS":{"ascii":"TEST","len":5}}
```

AT+TXPWR Pro

- *Sets the TX output effect for advertsing, scan and/or initiate air operation .*

Command	Syntax
AT+TXPWR	AT+TXPWR=**air operation**=**tx output parameters**
	AT+TXPWR

Air Operation Parameters

Parameters	Meaning
1	Advertise air operation
2	Scan air operation
4	Initiate air operation

TX Power Parameters

Parameters	Meaning
17	6 dBm
16	5 dBm
15	4.5 dBm
14	4 dBm
13	3 dBm
12	2 dBm
11	1.5 dBm
10	0 dBm
9	-1 dBm
8	-2 dBm
7	-3 dBm
6	-6 dBm
5	-8 dBm
4	-12 dBm
3	-18 dBm
2	-22 dBm
1	-26 dBm
0	-50 dBm

Example:

```
AT+TXPWR=1=17
OK
AT+TXPWR
OK
ADV: 17 (6dBm)
SCAN: 10 (0dBm)
INIT: 10 (0dBm)
```

```
AT+TXPWR=1=17
{"C":21,"cmd":"AT+TXPWR=1=17"}
{"A":21,"err":0,"errMsg":"ok"}
{"E":21,"nol":3}
AT+TXPWR
{"C":22,"cmd":"AT+TXPWR"}
{"A":22,"err":0,"errMsg":"ok"}
{"R":22,"info":"tx pwr"}
{"R":22,"adv":"17 (6dBm)","scan":"10 (0dBm)","init":"10 (0dBm)"}
{"E":22,"nol":5}
```

AT+TARGETCONN

- *Setting or querying the connection index to use as the targeted connection.*
  When connected to several devices, the target connection decides which device you target when using commands such as AT+GATTCREAD, AT+GATTCWRITE, AT+GAPDISCONNECT, AT+GETSERVICES, AT+GAPPAIR or AT+SPSSEND etc.

Command	Syntax
AT+TARGETCONN	
	AT+TARGETCONN=***conn_idx***
	or
	AT+TARGETCONN

Example:

```
AT+TARGETCONN=0000
OK
```

```
AT+TARGETCONN
{"C":14,"cmd":"AT+TARGETCONN"}
{"A":14,"err":0,"errMsg":"ok"}
{"R":14,"target":"0000"}
{"E":14,"nol":4}
```

AT+WAIT

- *Sets a timer to timeout after x seconds. No auto execute commands will be run while timer is running. Intended to be used in auto execute list.*

Command	Syntax
AT+WAIT	
	AT+WAIT=**seconds**

Max value of timer is 3600 seconds (1 hour).

Example:

```
AT+WAIT=10
OK
USER WAIT TIMEOUT!
```

```
AT+WAIT=10
{"C":19,"cmd":"AT+WAIT=10"}
{"A":19,"err":0,"errMsg":"ok"}
{"E":19,"nol":3}
{283:"FFFF","evt":{"action":"user wait timeout"}}
```

--H

- *Shows all AT-Commands.*

Command	Syntax
--H	--H

Example:

```
--H
{List of all AT-Commands...}
...
```

```
AT
{"C":15,"cmd":"--H"}
{"A":15,"err":0,"errMsg":"OK"}
{"R":15,"cmd":"AT","help":"[A] Basic AT-Command."}
{"R":15,"cmd":"ATA","help":"[A] Shows/hides ascii values from
notification/indication/read responses. ATA0 off, ATA1 on."}
# Displaying all help info
...
```

```
...
...
```