GigaDevice Semiconductor Inc.

GD32VW553 Basic Commands User Guide

Application Note

AN153

Revision 1.2

(Jul.2024)

Table of Contents

List of Figures

List of Tables

1. Basic user commands

Connect the test machine to the development board through a USB cable. Open the UART tool, and connect it to the correct COM port. After powering on and starting the development board correctly, issue a command through the UART tool, and the development board can complete the corresponding operation according to the command content.

In this manual, < > after the command indicates that the option is required, and [ ] indicates that the option is optional. Note that the commands are strictly case-sensitive.

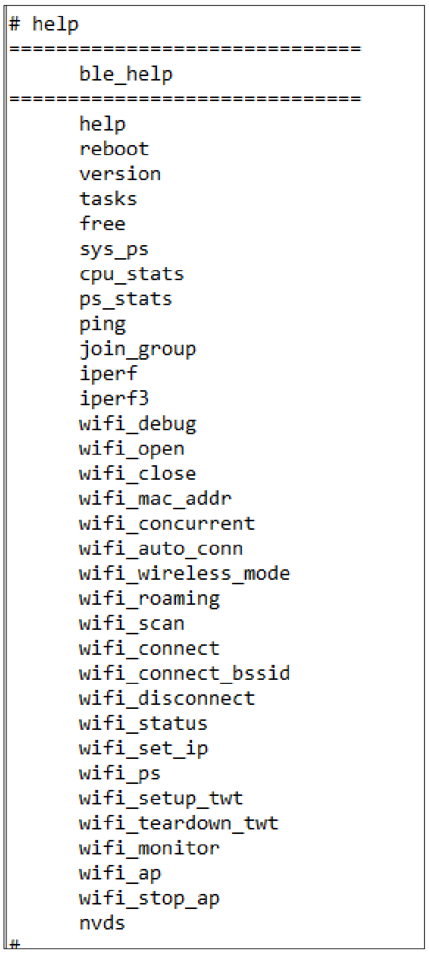
1.1. help

The command has no option.

As shown in ***Figure 1-1. help command***, the help command lists all commands supported by the development board.

Note: View BLE related commands through the ble_help command.

Figure 1-1. help command



```
# help
==============================
      ble_help
==============================
      help
      reboot
      version
      tasks
      free
      sys_ps
      cpu_stats
      ps_stats
      ping
      join_group
      iperf
      iperf3
      wifi_debug
      wifi_open
      wifi_close
      wifi_mac_addr
      wifi_concurrent
      wifi_auto_conn
      wifi_wireless_mode
      wifi_roaming
      wifi_scan
      wifi_connect
      wifi_connect_bssid
      wifi_disconnect
      wifi_status
      wifi_set_ip
      wifi_ps
      wifi_setup_twt
      wifi_teardown_twt
      wifi_monitor
      wifi_ap
      wifi_stop_ap
      nvds
#
```

1.2. reboot

The command has no option.

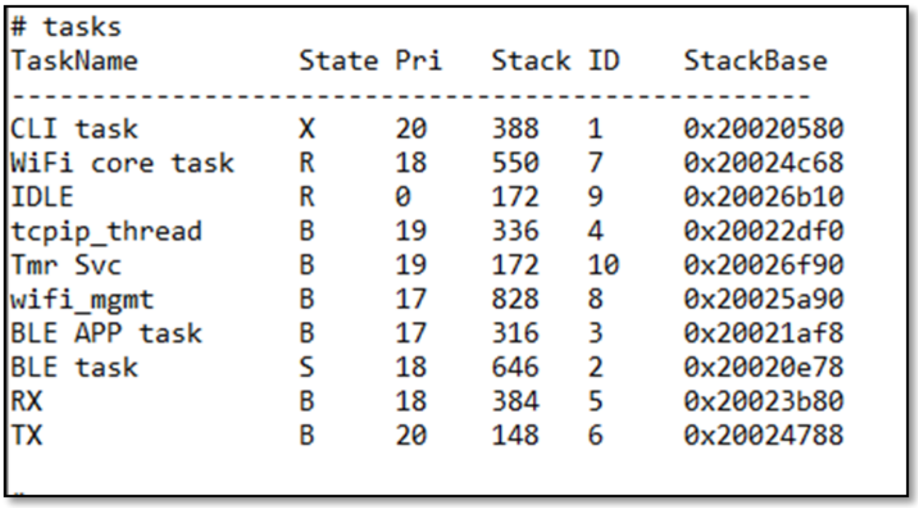
After the command is executed, the development board will restart and the serial port will print the start information. The command has a similar function as that of the reset button.

1.3. tasks

The command has no option.

After the command is executed, task-related information, including the status, priority, remaining minimum stack space for the task since task creation, task No., and the base address of the stack used by the task, will be printed, as shown in ***Figure 1-2. tasks command***.

Figure 1-2. tasks command



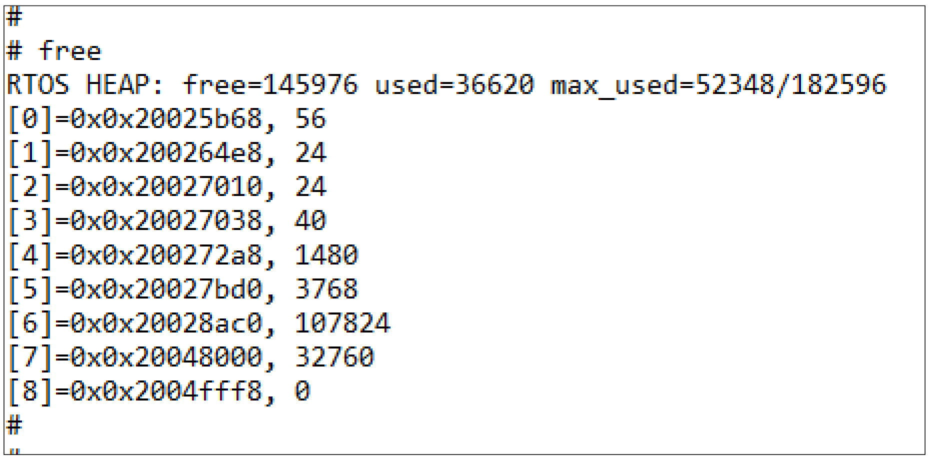
```
# tasks
TaskName          State Pri   Stack ID    StackBase
--------------------------------------------------
CLI task          X     20    388   1     0x20020580
WiFi core task    R     18    550   7     0x20024c68
IDLE              R     0     172   9     0x20026b10
tcpip_thread      B     19    336   4     0x20022df0
Tmr Svc           B     19    172   10    0x20026f90
wifi_mgmt         B     17    828   8     0x20025a90
BLE APP task      B     17    316   3     0x20021af8
BLE task          S     18    646   2     0x20020e78
RX                B     18    384   5     0x20023b80
TX                B     20    148   6     0x20024788
```

1.4. free

The command has no option.

After the command is executed, heap related information, including remaining heap, used heap, maximum used heap, maximum available heap, and the address and size of each available mem block, will be printed, as shown in ***Figure 1-3. free command***.

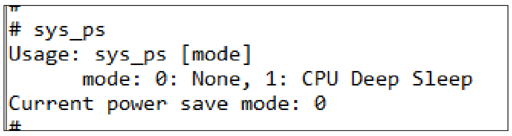
Figure 1-3. free command



```
#
# free
RTOS HEAP: free=145976 used=36620 max_used=52348/182596
[0]=0x0x20025b68, 56
[1]=0x0x200264e8, 24
[2]=0x0x20027010, 24
[3]=0x0x20027038, 40
[4]=0x0x200272a8, 1480
[5]=0x0x20027bd0, 3768
[6]=0x0x20028ac0, 107824
[7]=0x0x20048000, 32760
[8]=0x0x2004fff8, 0
#
```

1.5. sys_ps

Figure 1-4. sys_ps command



```
# sys_ps
Usage: sys_ps [mode]
       mode: 0: None, 1: CPU Deep Sleep
Current power save mode: 0
#
```

Figure 1-4. sys_ps command shows how to use the command. There are three modes:

Leave blank: No settings. Only print the current CPU power save mode;

0: Disable CPU power save.

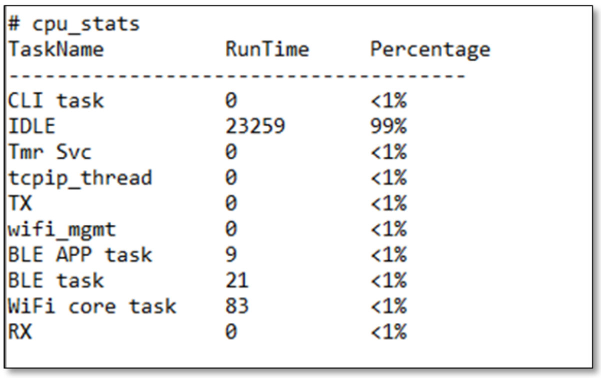
1: Enable CPU power save. The mode is deep sleep. When the CPU is idle and WiFi is in a connected state, it automatically enters the deep sleep mode, and then it can be automatically woken up by WiFi/ble or actively woken up by a uart rx event.

1.6. cpu_stats

The command has no option.

After this command is executed, the CPU usage of each task will be printed, including running time. as shown in *Figure 1-5. cpu_stats command*.

Figure 1-5. cpu_stats command



```
# cpu_stats
TaskName          RunTime     Percentage
--------------------------------------
CLI task          0           <1%
IDLE              23259       99%
Tmr Svc           0           <1%
tcpip_thread      0           <1%
TX                0           <1%
wifi_mgmt         0           <1%
BLE APP task      9           <1%
BLE task          21          <1%
WiFi core task    83          <1%
RX                0           <1%
```

1.7. rmem

This command is used to get the value of memory.

- Usage: rmem <addr> [count] [width]

<addr>: memory address.

[count]: the number of value.

[width]: the width of value. The unit is byte with the range of 1, 2, 4.

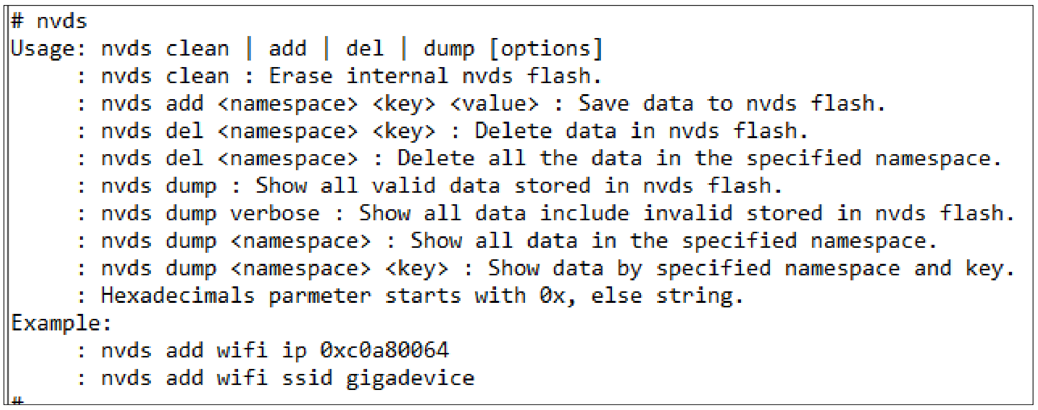
1.8. version

The command has no option.

After this command is executed, the SDK version, the generation time of SDK and the firmware version will be printed.

1.9. nvds

Figure 1-6. nvds command



```
# nvds
Usage: nvds clean | add | del | dump [options]
      : nvds clean : Erase internal nvds flash.
      : nvds add <namespace> <key> <value> : Save data to nvds flash.
      : nvds del <namespace> <key> : Delete data in nvds flash.
      : nvds del <namespace> : Delete all the data in the specified namespace.
      : nvds dump : Show all valid data stored in nvds flash.
      : nvds dump verbose : Show all data include invalid stored in nvds flash.
      : nvds dump <namespace> : Show all data in the specified namespace.
      : nvds dump <namespace> <key> : Show data by specified namespace and key.
      : Hexadecimals parmeter starts with 0x, else string.
Example:
      : nvds add wifi ip 0xc0a80064
      : nvds add wifi ssid gigadevice
#
```

Figure 1-6. nvds command shows how to use the command.

- nvds clean

This command is used to erase internal nvds flash.

- nvds add <namespace> <key> <value>

This command is used to save data to nvds flash.

- nvds del <namespace> <key>

This command is used to delete data in nvds flash.

- nvds del <namespace>

This command is used to delete all the data in the specified namespace.

- nvds dump

This command is used to show all valid data stored in nvds flash.

- nvds dump verbose

This command is used to show all data include invalid stored in nvds flash.

- nvds dump <namespace>

This command is used to show all data in the specified namespace.

- nvds dump <namespace> <key>

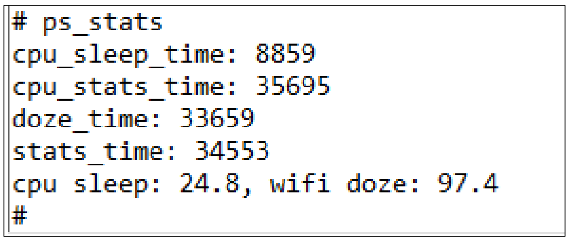
This command is used to show data by specified namespace and key.

1.10. ps_stats

The command has no option. As shown in *Figure 1-7. ps_stats command*,

After this command is executed, the information of system power save will be printed, . including CPU sleep time, CPU statistics time, WiFi doze time, WiFi statistics time, the percentage of CPU sleep and the percentage of WiFi doze. The unit of time is ms.

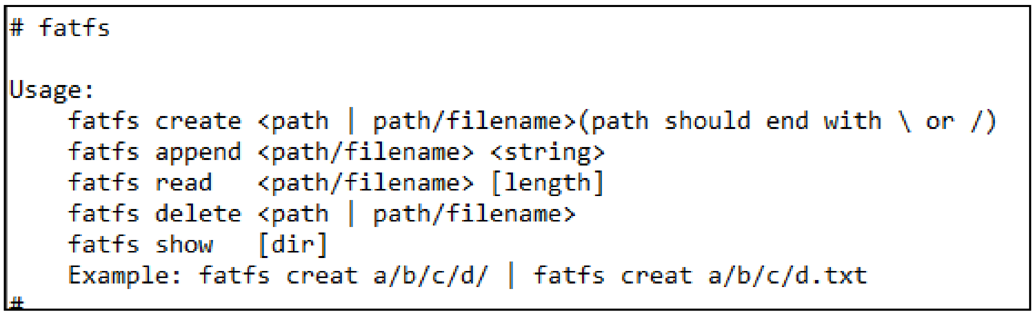
Figure 1-7. ps_stats command



```
# ps_stats
cpu_sleep_time: 8859
cpu_stats_time: 35695
doze_time: 33659
stats_time: 34553
cpu sleep: 24.8, wifi doze: 97.4
#
```

1.11. fatfs

Figure 1-8. fatfs command



```
# fatfs

Usage:
    fatfs create <path | path/filename>(path should end with \ or /)
    fatfs append <path/filename> <string>
    fatfs read   <path/filename> [length]
    fatfs delete <path | path/filename>
    fatfs show   [dir]
    Example: fatfs creat a/b/c/d/ | fatfs creat a/b/c/d.txt
#
```

Figure 1-8. fatfs command shows how to use the command.

- fatfs create <path | path/filename>

Create a folder with path <path> or a file with path <path/filename> on the root directory.

- fatfs append <path/filename> <string>

Write the contents of <string> at the end of the file with path <path/filename> in the form of append.

- fatfs read <path/filename> [length]

Read [length] bytes of data from the beginning of a file with path <path/filename>. If the length of the file is less than [length], the entire file is read. Default is to read the whole file.

- fatfs delete <path | path/filename>

Delete the folder with path <path> and all files in the folder, or delete the file with path <path/filename>.

- fatfs show [dir]

Print the file names and file lengths of the files in the folder with path [dir]. The default is the root directory.

1.12. Wi-Fi

This section introduces Wi-Fi related commands.

1.12.1. wifi_open

The command has no option.

This command is used to enable WiFi functions. Other WiFi-related commands can be executed provided that WiFi is enabled. After the development board is started correctly, WiFi is enabled by default, so this command is unnecessarily executed to repeatedly enable WiFi.

This command generally works with wifi_close by enabling WiFi after it is turned off by wifi_close. If WiFi is enabled, the serial port will give a prompt.

1.12.2. wifi_close

The command has no option.

wifi_close can turn off WiFi, but in this case, some commands, such as wifi_scan and wifi_connect, can not be executed.

The command has different execution results for different conditions of the development board:

- If the development board is connected to AP, the command will disconnect them and then turn off WiFi.
- If the development board is not connected to AP, the command will directly turn off WiFi.
- If the development board in SoftAP mode is connected to sta, the command will disconnect them and then turn off WiFi.
- If the development board in SoftAP mode is not connected to sta, the command will directly turn off WiFi.
- If WiFi is turned off, the serial port will give a prompt.

1.12.3. wifi_debug

- Usage: wifi_debug <0 or 1>

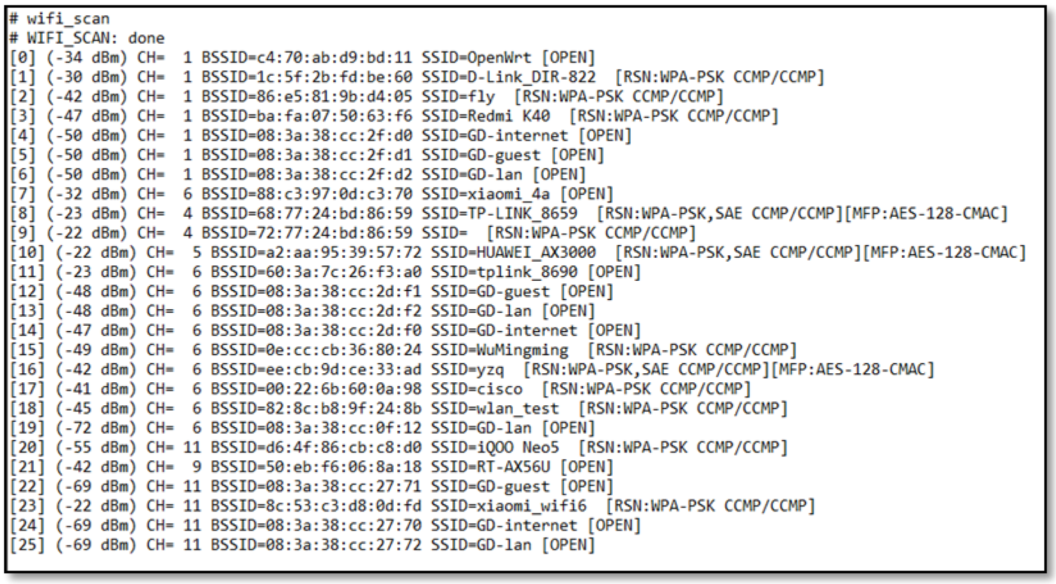
This command is used to control the printing of WiFi debug logs. 0 means printing is disabled, while 1 means printing is enabled.

1.12.4. wifi_scan

The command has no option.

After the command is executed, AP information scanned by the development board, including RSSI, channel, BSSID, SSID, and encryption method, will be printed, as shown in ***Figure 1-9. wifi_scan command***.

Figure 1-9. wifi_scan command



```
# wifi_scan
# WIFI_SCAN: done
[0] (-34 dBm) CH=  1 BSSID=c4:70:ab:d9:bd:11 SSID=OpenWrt [OPEN]
[1] (-30 dBm) CH=  1 BSSID=1c:5f:2b:fd:be:60 SSID=D-Link_DIR-822  [RSN:WPA-PSK CCMP/CCMP]
[2] (-42 dBm) CH=  1 BSSID=86:e5:81:9b:d4:05 SSID=fly  [RSN:WPA-PSK CCMP/CCMP]
[3] (-47 dBm) CH=  1 BSSID=ba:fa:07:50:63:f6 SSID=Redmi K40  [RSN:WPA-PSK CCMP/CCMP]
[4] (-50 dBm) CH=  1 BSSID=08:3a:38:cc:2f:d0 SSID=GD-internet [OPEN]
[5] (-50 dBm) CH=  1 BSSID=08:3a:38:cc:2f:d1 SSID=GD-guest [OPEN]
[6] (-50 dBm) CH=  1 BSSID=08:3a:38:cc:2f:d2 SSID=GD-lan [OPEN]
[7] (-32 dBm) CH=  6 BSSID=88:c3:97:0d:c3:70 SSID=xiaomi_4a [OPEN]
[8] (-23 dBm) CH=  4 BSSID=68:77:24:bd:86:59 SSID=TP-LINK_8659  [RSN:WPA-PSK,SAE CCMP/CCMP][MFP:AES-128-CMAC]
[9] (-22 dBm) CH=  4 BSSID=72:77:24:bd:86:59 SSID=  [RSN:WPA-PSK CCMP/CCMP]
[10] (-22 dBm) CH=  5 BSSID=a2:aa:95:39:57:72 SSID=HUAWEI_AX3000  [RSN:WPA-PSK,SAE CCMP/CCMP][MFP:AES-128-CMAC]
[11] (-23 dBm) CH=  6 BSSID=60:3a:7c:26:f3:a0 SSID=tplink_8690 [OPEN]
[12] (-48 dBm) CH=  6 BSSID=08:3a:38:cc:2d:f1 SSID=GD-guest [OPEN]
[13] (-48 dBm) CH=  6 BSSID=08:3a:38:cc:2d:f2 SSID=GD-lan [OPEN]
[14] (-47 dBm) CH=  6 BSSID=08:3a:38:cc:2d:f0 SSID=GD-internet [OPEN]
[15] (-49 dBm) CH=  6 BSSID=0e:cc:cb:36:80:24 SSID=WuMingming  [RSN:WPA-PSK CCMP/CCMP]
[16] (-42 dBm) CH=  6 BSSID=ee:cb:9d:ce:33:ad SSID=yzq  [RSN:WPA-PSK,SAE CCMP/CCMP][MFP:AES-128-CMAC]
[17] (-41 dBm) CH=  6 BSSID=00:22:6b:60:0a:98 SSID=cisco  [RSN:WPA-PSK CCMP/CCMP]
[18] (-45 dBm) CH=  6 BSSID=82:8c:b8:9f:24:8b SSID=wlan_test  [RSN:WPA-PSK CCMP/CCMP]
[19] (-72 dBm) CH=  6 BSSID=08:3a:38:cc:0f:12 SSID=GD-lan [OPEN]
[20] (-55 dBm) CH= 11 BSSID=d6:4f:86:cb:c8:d0 SSID=iQOO Neo5  [RSN:WPA-PSK CCMP/CCMP]
[21] (-42 dBm) CH=  9 BSSID=50:eb:f6:06:8a:18 SSID=RT-AX56U [OPEN]
[22] (-69 dBm) CH= 11 BSSID=08:3a:38:cc:27:71 SSID=GD-guest [OPEN]
[23] (-22 dBm) CH= 11 BSSID=8c:53:c3:d8:0d:fd SSID=xiaomi_wifi6  [RSN:WPA-PSK CCMP/CCMP]
[24] (-69 dBm) CH= 11 BSSID=08:3a:38:cc:27:70 SSID=GD-internet [OPEN]
[25] (-69 dBm) CH= 11 BSSID=08:3a:38:cc:27:72 SSID=GD-lan [OPEN]
```

1.12.5. wifi_concurrent

- Usage: wifi_concurrent [0 or 1]

This command is used to control enabling of WiFi concurrent mode. 0 means the mode is disabled, while 1 means the mode is enabled. When this option is not set, the current enabled state is printed only.

This command can not be executed until the macro CFG_WIFI_CONCURRENT in MSDK\macsw\export\ wlan_config.h file is opened.

1.12.6. wifi_connect

- Usage: wifi_connect <SSID> [PASSWORD]

The command is used to connect to an AP. The development board cannot be in SoftAP mode when the command is executed.

- wifi_connect <SSID>

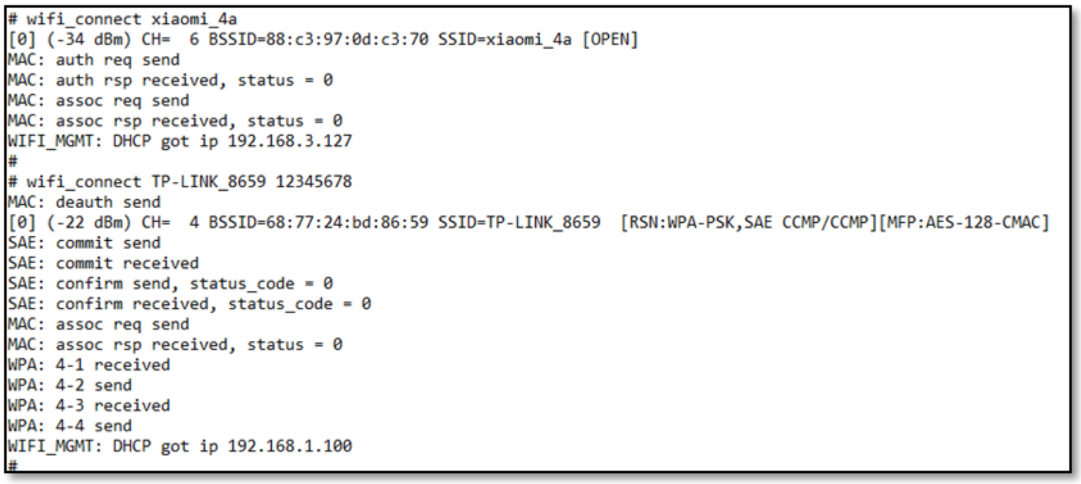
It is used to connect to an unencrypted AP.

- wifi_connect <SSID> <PASSWORD>

It is used to connect to an encrypted AP.

The connection process is as shown in ***Figure 1-10. Wifi_connect command***, and the serial port prints out the connection process information. If the wifi_connect command is executed after the AP is connected, the development board will first disconnect from the original AP and then connect to the new AP.

Figure 1-10. Wifi_connect command



```
# wifi_connect xiaomi_4a
[0] (-34 dBm) CH=  6 BSSID=88:c3:97:0d:c3:70 SSID=xiaomi_4a [OPEN]
MAC: auth req send
MAC: auth rsp received, status = 0
MAC: assoc req send
MAC: assoc rsp received, status = 0
WIFI_MGMT: DHCP got ip 192.168.3.127
#
# wifi_connect TP-LINK_8659 12345678
MAC: deauth send
[0] (-22 dBm) CH=  4 BSSID=68:77:24:bd:86:59 SSID=TP-LINK_8659  [RSN:WPA-PSK,SAE CCMP/CCMP][MFP:AES-128-CMAC]
SAE: commit send
SAE: commit received
SAE: confirm send, status_code = 0
SAE: confirm received, status_code = 0
MAC: assoc req send
MAC: assoc rsp received, status = 0
WPA: 4-1 received
WPA: 4-2 send
WPA: 4-3 received
WPA: 4-4 send
WIFI_MGMT: DHCP got ip 192.168.1.100
#
```

1.12.7. wifi_connect_bssid

- Usage: wifi_connect_bssid <BSSID> [PASSWORD]

This command is similar to the wifi_connect command, executed by using the same method. The only difference is that SSID in the option is modified to BSSID.

1.12.8. wifi_connect_eap_tls

- Usage: wifi_connect_eap_tls <SSID>

This command uses EAP-TLS authentication to connect to enterprise APs.

This command has only one parameter <SSID>. Other conditions required for the connection, such as root certificate, client certificate, etc. are already included in the SDK code.

1.12.9. wifi_disconnect

The command has no option.

After the command is executed, the development board will disconnect from the AP. If the execution is successful, the serial port will print the following information:

MAC: deauth send

MGMT: disconnect complete

1.12.10. wifi_auto_conn

- Usage: wifi_auto_conn [0 or 1]

This command is used to set automatic connection to AP upon startup or not. 0 means no automatic connection, while 1 means automatic connection. When this option is not set, the current settings are printed only.

If automatic connection is set, when AP is successfully connected again, AP information will be saved in flash; however, if AP is repeatedly connected, the AP successfully connected last will be recoded as valid AP. Rebooted development board will be automatically connected to AP according to AP information in flash. If AP is not connected after automatic connection is set, rebooted development board will not be automatically connected to AP.

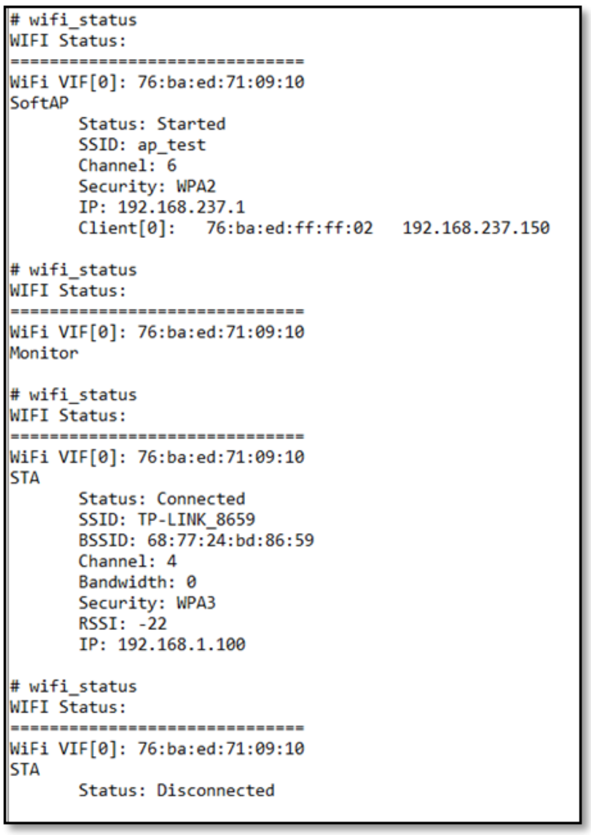
1.12.11. wifi_status

The command has no option.

After the command is executed, the serial port will print the Wi-Fi status of the current development board.

Wi-Fi currently has three modes: SoftAP, monitor, and station. The information printed by the command in different modes is different, as shown in ***Figure 1-11. wifi_status command***.

Figure 1-11. wifi_status command



```
# wifi_status
WIFI Status:
==============================
WiFi VIF[0]: 76:ba:ed:71:09:10
SoftAP
       Status: Started
       SSID: ap_test
       Channel: 6
       Security: WPA2
       IP: 192.168.237.1
       Client[0]:   76:ba:ed:ff:ff:02   192.168.237.150

# wifi_status
WIFI Status:
==============================
WiFi VIF[0]: 76:ba:ed:71:09:10
Monitor

# wifi_status
WIFI Status:
==============================
WiFi VIF[0]: 76:ba:ed:71:09:10
STA
       Status: Connected
       SSID: TP-LINK_8659
       BSSID: 68:77:24:bd:86:59
       Channel: 4
       Bandwidth: 0
       Security: WPA3
       RSSI: -22
       IP: 192.168.1.100

# wifi_status
WIFI Status:
==============================
WiFi VIF[0]: 76:ba:ed:71:09:10
STA
       Status: Disconnected
```

This first line shows MAC address of the current Wi-Fi device; the second line shows the mode of the current Wi-Fi device, one of the above three modes.

In AP mode, the status, SSID, channel, encryption method, and IP address will be displayed. If any devices are connected to this AP, the information of these devices, including the MAC address and IP address, will also be displayed, and multiple devices will be sorted in sequence.

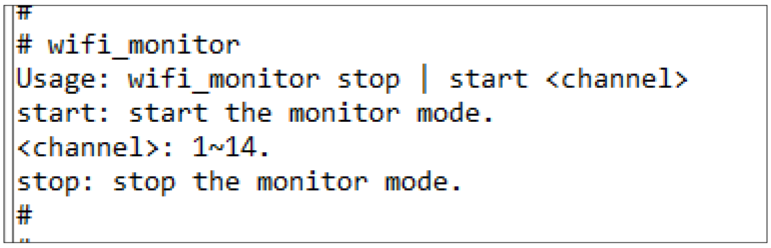
In station mode, WIFI Status indicates whether the current Wi-Fi device is connected to the AP; Connected means connected, and Disconnected means disconnected. If it is connected, the SSID, BSSID, and channel of the AP will be displayed.

1.12.12. wifi_monitor

- Usage: wifi_monitor stop | start <channel>

Figure 1-12. wifi_monitor command shows how to use the command. The wifi_monitor start <channel> command is used to start the monitor mode, and the monitoring channel needs to be specified; the wifi_monitor stop command is used to close the monitor mode and switch to the station mode.

Figure 1-12. wifi_monitor command

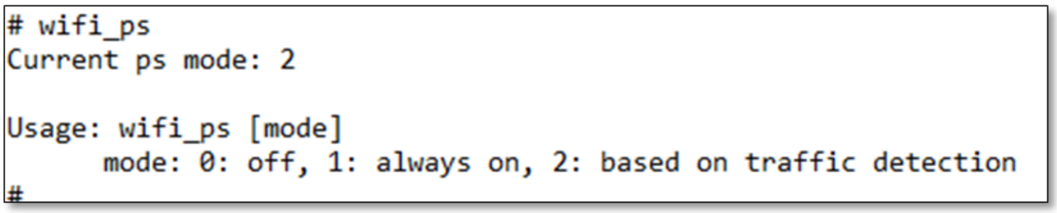


```
#
# wifi_monitor
Usage: wifi_monitor stop | start <channel>
start: start the monitor mode.
<channel>: 1~14.
stop: stop the monitor mode.
#
```

1.12.13. wifi_ps

- Usage: wifi_ps [mode]

Figure 1-13. wifi_ps command



```
# wifi_ps
Current ps mode: 2

Usage: wifi_ps [mode]
      mode: 0: off, 1: always on, 2: based on traffic detection
#
```

Figure 1-13. wifi_ps command shows how to use the command. There are three modes:

0: Disable power save;

1: Enable power save. The mode is Normal mode, and the Wi-Fi module will always be in power save mode;

2: Enable power save. The mode is Dynamic mode, and the Wi-Fi module will decide whether to enter or exit the power save mode based on the WiFi TX/RX traffic;

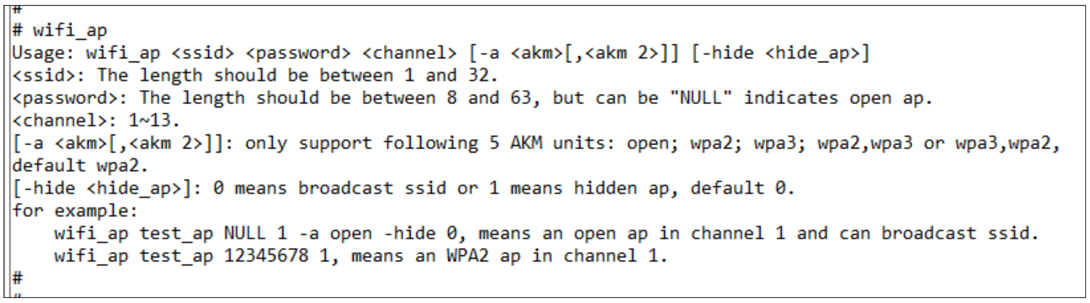
When this option is not set, the current wifi ps mode is printed only.

1.12.14. wifi_ap

■ Usage: wifi_ap <ssid> <password> <channel> [-a <akm>[,<akm 2>]] [-hide <hide_ap>]

The command is used to enable or disable the SoftAP mode. ***Figure 1-14. wifi_ap command*** shows how to use the command.

Figure 1-14. wifi_ap command



```
#
# wifi_ap
Usage: wifi_ap <ssid> <password> <channel> [-a <akm>[,<akm 2>]] [-hide <hide_ap>]
<ssid>: The length should be between 1 and 32.
<password>: The length should be between 8 and 63, but can be "NULL" indicates open ap.
<channel>: 1~13.
[-a <akm>[,<akm 2>]]: only support following 5 AKM units: open; wpa2; wpa3; wpa2,wpa3 or wpa3,wpa2,
default wpa2.
[-hide <hide_ap>]: 0 means broadcast ssid or 1 means hidden ap, default 0.
for example:
    wifi_ap test_ap NULL 1 -a open -hide 0, means an open ap in channel 1 and can broadcast ssid.
    wifi_ap test_ap 12345678 1, means an WPA2 ap in channel 1.
#
```

ssid does not support Chinese characters. When "NULL" is filled in the password, it means that an open AP is enabled, and the -a configuration will be ignored. In addition, if an encrypted AP is enabled and the -a option is not configured to specify the encryption method, the default is wpa2 encryption.

1.12.15. wifi_stop_ap

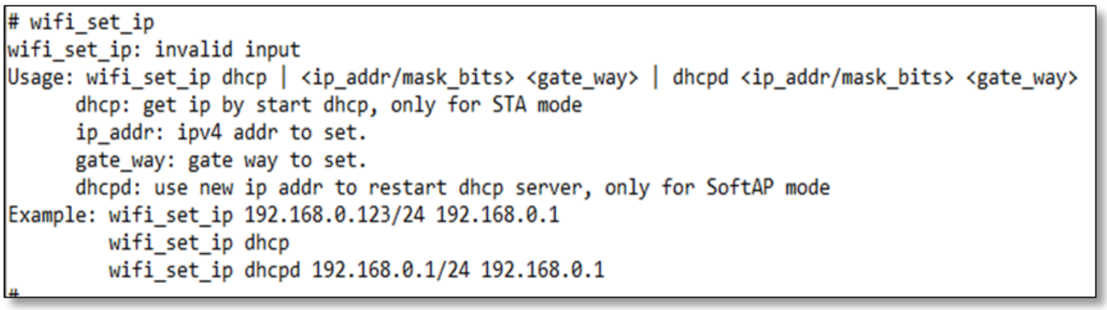
The command has no option. After the command is executed, the SoftAP mode will stop and switch to the station mode.

1.12.16. wifi_set_ip

■ Usage: wifi_set_ip dhcp | <ip_addr/mask_bits> <gate_way> | dhcpd <ip_addr/mask_bits> <gate_way>

The command is used to manually set a static IP address or automatically get an IP address through DHCP in station mode. It can also set the IP address and gateway in SoftAP mode. ***Figure 1-15. wifi_set_ip command*** shows how to use the command.

Figure 1-15. wifi_set_ip command



```
# wifi_set_ip
wifi_set_ip: invalid input
Usage: wifi_set_ip dhcp | <ip_addr/mask_bits> <gate_way> | dhcpd <ip_addr/mask_bits> <gate_way>
      dhcp: get ip by start dhcp, only for STA mode
      ip_addr: ipv4 addr to set.
      gate_way: gate way to set.
      dhcpd: use new ip addr to restart dhcp server, only for SoftAP mode
Example: wifi_set_ip 192.168.0.123/24 192.168.0.1
         wifi_set_ip dhcp
         wifi_set_ip dhcpd 192.168.0.1/24 192.168.0.1
#
```

1.12.17. wifi_mac_addr

- Usage: wifi_mac_addr [xx:xx:xx:xx:xx:xx]

This command is used to set temporary MAC address of WiFi, which will become valid after wifi_close and wifi_open command is executed and become invalid after reboot or power-down reboot.

When this option is not set, the current MAC address is printed only.

1.12.18. wifi_wireless_mode

- Usage: wifi_wireless_mode [bg or bgn or bgnax]

This command is used to set temporary WiFi wireless mode. There are three modes: bg, bgn and bgnax, which will become valid after wifi_close and wifi_open command is executed and become invalid after reboot or power-down reboot.

When this option is not set, the current WiFi wireless mode is printed only.

1.12.19. wifi_roaming

- Usage: wifi_roaming [enable] [rssi_threshold]

This command is used to set the function of regularly checking RSSI and roaming based on the results in station mode connected state.

- wifi_roaming

This command is used to print current configuration.

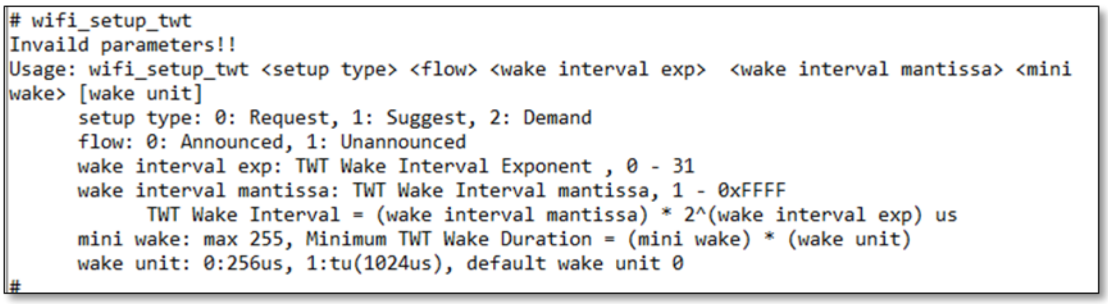
- wifi_roaming [enable] [rssi_threshold]

Turn off the RSSI roaming function when ‘enable’ is 0, and enable it when ‘enable’ is 1.

The rssi_threshold is the RSSI threshold when enable RSSI roaming function and it’s value must be less than 0.

1.12.20. wifi_setup_twt

Figure 1-16. wifi_setup_twt command



```
# wifi_setup_twt
Invaild parameters!!
Usage: wifi_setup_twt <setup type> <flow> <wake interval exp>  <wake interval mantissa> <mini
wake> [wake unit]
      setup type: 0: Request, 1: Suggest, 2: Demand
      flow: 0: Announced, 1: Unannounced
      wake interval exp: TWT Wake Interval Exponent , 0 - 31
      wake interval mantissa: TWT Wake Interval mantissa, 1 - 0xFFFF
            TWT Wake Interval = (wake interval mantissa) * 2^(wake interval exp) us
      mini wake: max 255, Minimum TWT Wake Duration = (mini wake) * (wake unit)
      wake unit: 0:256us, 1:tu(1024us), default wake unit 0
#
```

Figure 1-16. wifi_setup_twt command shows how to use the command.

- setup type, 'request' means that the TWT parameters are expected to be determined by AP; 'suggest' means that the TWT parameters are determined through consultation between both; 'demand' means that the TWT parameters are determined by STA and cannot be modified.
- flow, 'announced' means that after waking up, STA needs to send PS-poll or QOS-NULL HE-TB PPDU frame to inform AP that it has wokenup; 'unannounced' means that there is no need to inform AP after STA wakes up.
- wake interval exp, the exponential part of the calculation formula for TWT Wake interval.
- wake interval mantissa, the mantissa part of the calculation formula for TWT Wake interval. The specific calculation formula is shown in the above figure.
- mini wake, the maximum time from TWT SP to be in awake state and the unit is determined by 'wake unit'.
- wake unit, the unit of 'mini wake'. 0 means 256us, 1 means 1024us. The default value is 0.

1.12.21. wifi_teardown_twt

- Usage: wifi_teardown_twt <flow id> [negotiation type]

The command is used to terminate a TWT flow.

- flow id, the id of TWT flow which will be terminated.
- negotiation type, the value of negotiation type field in TWT Teardown Frame, the default is 0.

1.12.22. wifi_listen_interval

- Usage: wifi_listen_interval [interval]

- interval: 0: listen beacon by dtim, 1 - 10 , the interval of listen beacon.

The command is used to set the interval at which the hardware listens for beacon frames in low-power mode.

Use this command with caution! Modifying this interval may result in serious frame loss!

1.12.23. wps

- Usage: wps pbc | pin <pin code>

This command is used to connect to AP by WPS method.

- wps pbc

Use WPS PBC mode.

- wps pin <pin code>

Use WPS PIN mode.

1.13. Wi-Fi APP

1.13.1. ping

- Usage: ping <target_ip | stop> [-n count] [-l size] [-i interval] [-t total time]

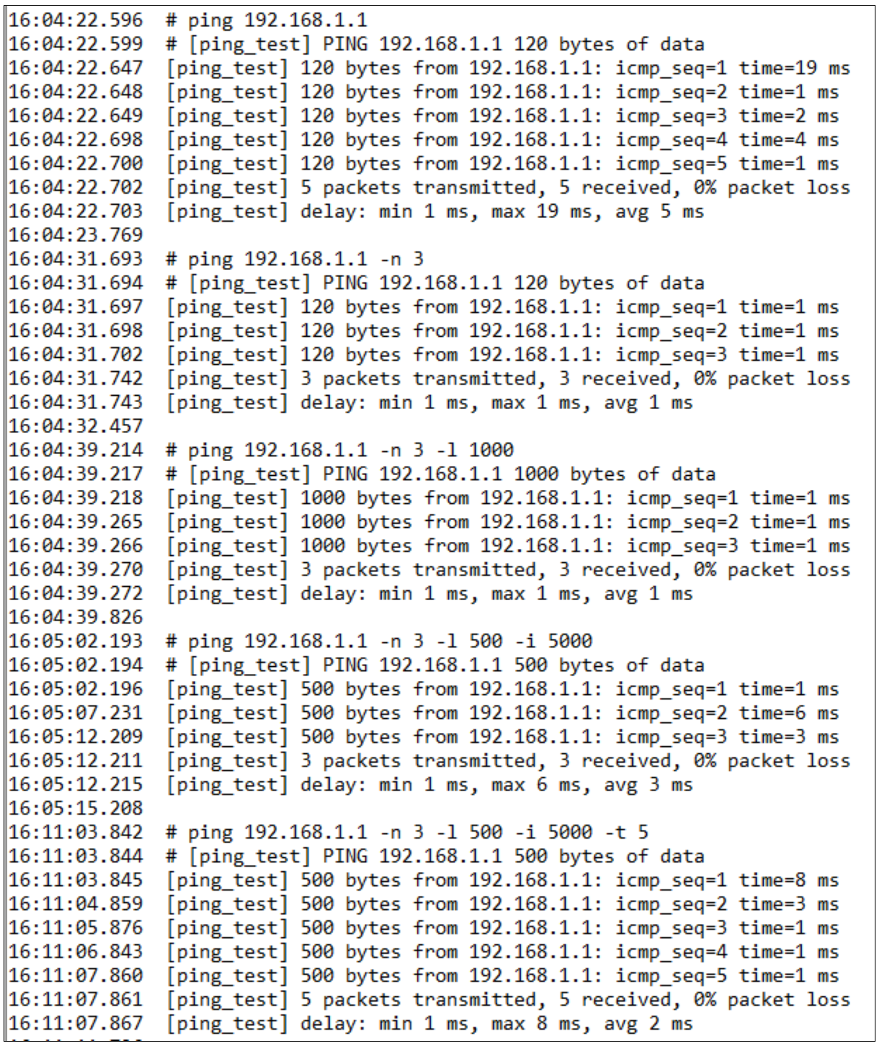
The command is used to perform the ping test.

The target_ip is a peer address. IPv4 is in a format of <ipv4_addr>, while IPv6 is in a format of <-6 ipv6_addr> (provided that Ipv6 is enabled).

In the parameters of the command, count indicates the number of ping packets; size indicates the packet length, in bytes; interval indicates the packet sending interval, in milliseconds; total time indicates the total running time, in seconds. By default, count is 5, size is 120, interval is 10, and total time is not used; if the total time option is used, the count and interval options do not work; interval defaults to 1000 ms, and count will be equal to the total time value.

Figure 1-17. ping command shows how to use the ping command.

Figure 1-17. ping command

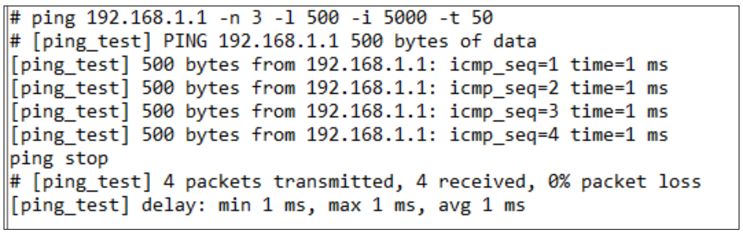


```
16:04:22.596  # ping 192.168.1.1
16:04:22.599  # [ping_test] PING 192.168.1.1 120 bytes of data
16:04:22.647  [ping_test] 120 bytes from 192.168.1.1: icmp_seq=1 time=19 ms
16:04:22.648  [ping_test] 120 bytes from 192.168.1.1: icmp_seq=2 time=1 ms
16:04:22.649  [ping_test] 120 bytes from 192.168.1.1: icmp_seq=3 time=2 ms
16:04:22.698  [ping_test] 120 bytes from 192.168.1.1: icmp_seq=4 time=4 ms
16:04:22.700  [ping_test] 120 bytes from 192.168.1.1: icmp_seq=5 time=1 ms
16:04:22.702  [ping_test] 5 packets transmitted, 5 received, 0% packet loss
16:04:22.703  [ping_test] delay: min 1 ms, max 19 ms, avg 5 ms
16:04:23.769
16:04:31.693  # ping 192.168.1.1 -n 3
16:04:31.694  # [ping_test] PING 192.168.1.1 120 bytes of data
16:04:31.697  [ping_test] 120 bytes from 192.168.1.1: icmp_seq=1 time=1 ms
16:04:31.698  [ping_test] 120 bytes from 192.168.1.1: icmp_seq=2 time=1 ms
16:04:31.702  [ping_test] 120 bytes from 192.168.1.1: icmp_seq=3 time=1 ms
16:04:31.742  [ping_test] 3 packets transmitted, 3 received, 0% packet loss
16:04:31.743  [ping_test] delay: min 1 ms, max 1 ms, avg 1 ms
16:04:32.457
16:04:39.214  # ping 192.168.1.1 -n 3 -l 1000
16:04:39.217  # [ping_test] PING 192.168.1.1 1000 bytes of data
16:04:39.218  [ping_test] 1000 bytes from 192.168.1.1: icmp_seq=1 time=1 ms
16:04:39.265  [ping_test] 1000 bytes from 192.168.1.1: icmp_seq=2 time=1 ms
16:04:39.266  [ping_test] 1000 bytes from 192.168.1.1: icmp_seq=3 time=1 ms
16:04:39.270  [ping_test] 3 packets transmitted, 3 received, 0% packet loss
16:04:39.272  [ping_test] delay: min 1 ms, max 1 ms, avg 1 ms
16:04:39.826
16:05:02.193  # ping 192.168.1.1 -n 3 -l 500 -i 5000
16:05:02.194  # [ping_test] PING 192.168.1.1 500 bytes of data
16:05:02.196  [ping_test] 500 bytes from 192.168.1.1: icmp_seq=1 time=1 ms
16:05:07.231  [ping_test] 500 bytes from 192.168.1.1: icmp_seq=2 time=6 ms
16:05:12.209  [ping_test] 500 bytes from 192.168.1.1: icmp_seq=3 time=3 ms
16:05:12.211  [ping_test] 3 packets transmitted, 3 received, 0% packet loss
16:05:12.215  [ping_test] delay: min 1 ms, max 6 ms, avg 3 ms
16:05:15.208
16:11:03.842  # ping 192.168.1.1 -n 3 -l 500 -i 5000 -t 5
16:11:03.844  # [ping_test] PING 192.168.1.1 500 bytes of data
16:11:03.845  [ping_test] 500 bytes from 192.168.1.1: icmp_seq=1 time=8 ms
16:11:04.859  [ping_test] 500 bytes from 192.168.1.1: icmp_seq=2 time=3 ms
16:11:05.876  [ping_test] 500 bytes from 192.168.1.1: icmp_seq=3 time=1 ms
16:11:06.843  [ping_test] 500 bytes from 192.168.1.1: icmp_seq=4 time=1 ms
16:11:07.860  [ping_test] 500 bytes from 192.168.1.1: icmp_seq=5 time=1 ms
16:11:07.861  [ping_test] 5 packets transmitted, 5 received, 0% packet loss
16:11:07.867  [ping_test] delay: min 1 ms, max 8 ms, avg 2 ms
```

- ping stop

This command is used to stop the ping test, as shown in ***Figure 1-18. ping stop command***,

Figure 1-18. ping stop command



```
# ping 192.168.1.1 -n 3 -l 500 -i 5000 -t 50
# [ping_test] PING 192.168.1.1 500 bytes of data
[ping_test] 500 bytes from 192.168.1.1: icmp_seq=1 time=1 ms
[ping_test] 500 bytes from 192.168.1.1: icmp_seq=2 time=1 ms
[ping_test] 500 bytes from 192.168.1.1: icmp_seq=3 time=1 ms
[ping_test] 500 bytes from 192.168.1.1: icmp_seq=4 time=1 ms
ping stop
# [ping_test] 4 packets transmitted, 4 received, 0% packet loss
[ping_test] delay: min 1 ms, max 1 ms, avg 1 ms
```

1.13.2. join_group

- Usage: join_group <group ip eg:224.0.0.5>

The development board must be connected to the AP before the command is executed. After the command is executed, the development board will join a multicast group, such as:

- join_group 224.0.0.5

During this period, sniffer can be used to capture the IGMP protocol packet sent by the development board after the command is executed.

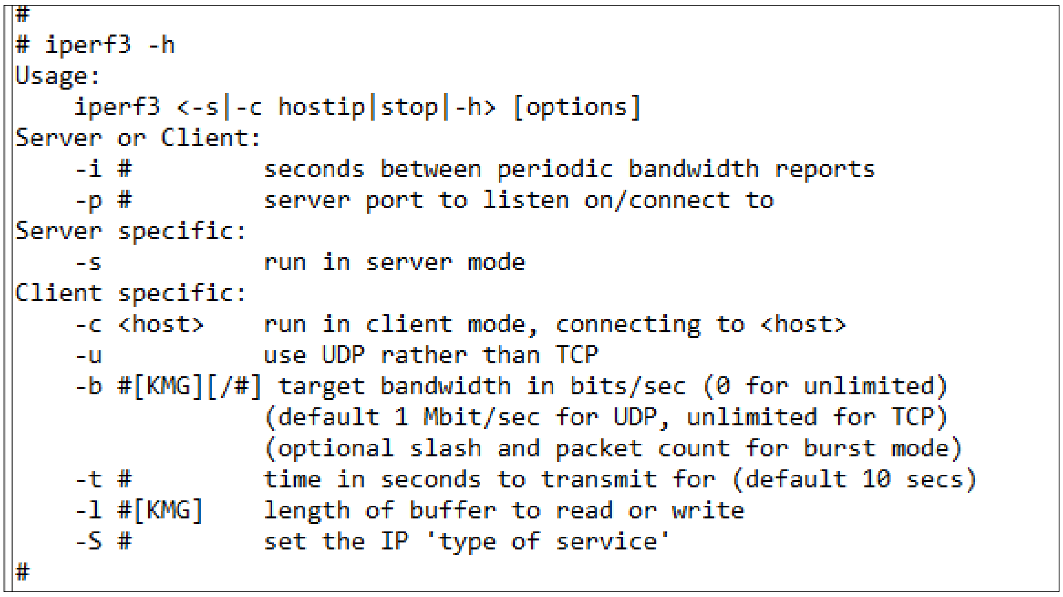
1.13.3. iperf3

The iperf3 command uses iperf3 for network speed test.

iperf3 -h

As shown in ***Figure 1-19. iperf3 -h command***, the serial port will print out the options related to the iperf3 command.

Figure 1-19. iperf3 -h command



```
#
# iperf3 -h
Usage:
    iperf3 <-s|-c hostip|stop|-h> [options]
Server or Client:
    -i #          seconds between periodic bandwidth reports
    -p #          server port to listen on/connect to
Server specific:
    -s            run in server mode
Client specific:
    -c <host>     run in client mode, connecting to <host>
    -u            use UDP rather than TCP
    -b #[KMG][/#] target bandwidth in bits/sec (0 for unlimited)
                  (default 1 Mbit/sec for UDP, unlimited for TCP)
                  (optional slash and packet count for burst mode)
    -t #          time in seconds to transmit for (default 10 secs)
    -l #[KMG]     length of buffer to read or write
    -S #          set the IP 'type of service'
#
```

iperf3 -s [options]

- iperf3 -s

Enable an iperf3 server to listen to TCP/UDP data on port 5201 by default. Other options are default values.

- -p <port>

Set the port that the server listens on. The port range is 0-65535, and the default is 5201.

For example: iperf3 -s -p 5003

The server listens on port 5003.

- -i <interval>

Set the period of the test results printed by the serial port (Interval column), in seconds, with a range of 0.1-60 and 0. When it is set to 0, it means that no periodic report is printed and only the final test results are output. The default is 4.

For example: iperf3 -s -i 0.5,

The period of the test results printed by the serial port is 0.5 s.

iperf3 -c <hostip> [options]

- iperf3 -c <hostip>

Enable an iperf3 client, and make a TCP connection with the server whose IP address is <host> on the default port 5201. All other options are default values.

- -u

Enable an iperf client, and make a UDP connection with the server whose IP address is <host> on the default port 5201. The -u option is usually used with the -b option to specify the data bandwidth to be sent.

- -p <port>

Set the client connection port, which must be the same as the port that the server listens on.

- -i <interval>

The -i option settings are the same as those on the server.

- -b <bandwidth/number>

The unit of bandwidth is bits/sec and the format is data[KMG]. For example, 50K, 50k or 50000 means that the bandwidth is set to 50 Kbits/sec; when bandwidth is 0, it means that there is no limit. The default in UDP mode is 1 Mbit/sec, and there is no limit for tcp connection.

When "/number" is not added after bandwidth, iperf3 will calculate the number of data packets that need to be sent per second to reach the specified bandwidth based on the length of each data packet, and then send each data packet at an average interval.

For example: iperf3 -c 192.168.3.132 -u -b 200k

When "/number" is added after bandwidth, the system enters the burst mode, and iperf3 will continuously send the specified number of data packets at one time without interval, but there is an even interval between the batches.

For example: iperf3 -c 192.168.3.132 -u -b 200k/60

- -t <time>

Set the data transmission time, in seconds. The default value is 10.

- -l <length>

Set the length of the read and write buffer, in bytes; the format is: data[KMG], the same as the -n option. It is recommended to set this value to 1472 in UDP mode and 1460 in TCP mode.

- -S <QOS value>

Set the QOS service type of the outstack packet. The number range is 0-255, and hexadecimal (0x prefix), octal (0 prefix), and decimal can be used, such as 0x16 == 026 == 22.

iperf3 stop

The command is used to stop the iperf3 test.

iperf3 test example

- Connect the development board and test machine to the same AP, and then view the IP address of the development board.
  - Use the wifi_connect command to connect the development board to the AP, and use the wifi_status command to view the IP address.
- The test machine opens the iperf3 command window and starts the test.
  - The server first executes the command: iperf3 -s -p <port> -i <interval>
  - The client then executes the command: iperf3 -c <host> -l <length> -p <port> -i <interval> -u -b <bandwidth/number> -t <time>
  - The -l, -p, -i, -u, -b, and -t options are optional. The -p option must be used by the server and client at the same time and have the same value; the -i option does not need to be used by the server and client at the same time and can have different values;
  - For example:
  - iperf3 -s -p 5004 -i 1
  - iperf3 -c 192.168.1.104 -l 1460 -p 5004 -i 2 -t 20   //TCP
  - iperf3 -c 192.168.1.104 -l 1472 -p 5004 -i 4 -t 30 -u -b 50M   //UDP
- After the server executes the command, the print information is displayed in the window, telling us that the server is enabled and listening on the corresponding port. After the client executes the command, the test machine and development board will print the test information at the same time.

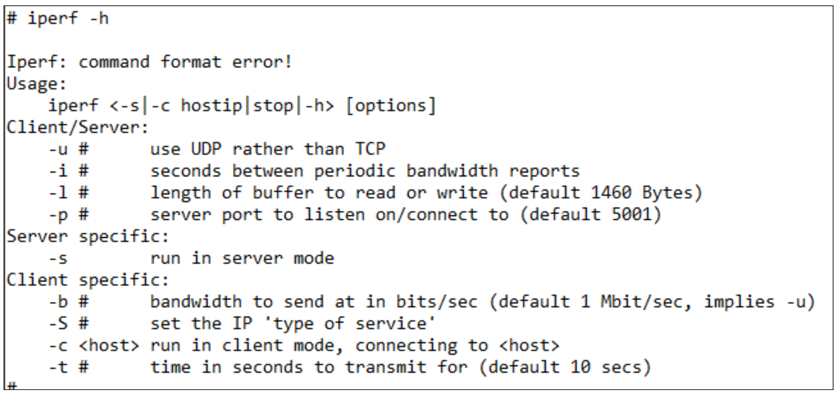
1.13.4. iperf

The iperf command calls iperf2 to perform network speed test. iperf runs in TCP mode by default, and the UDP mode must be specified through the -u option. The relevant options of the command (case-sensitive) are as follows.

iperf -h

As shown in ***Figure 1-20. iperf -h command***, the serial port will print out the options related to the iperf command.

Figure 1-20. iperf -h command



```
# iperf -h

Iperf: command format error!
Usage:
    iperf <-s|-c hostip|stop|-h> [options]
Client/Server:
    -u #      use UDP rather than TCP
    -i #      seconds between periodic bandwidth reports
    -l #      length of buffer to read or write (default 1460 Bytes)
    -p #      server port to listen on/connect to (default 5001)
Server specific:
    -s        run in server mode
Client specific:
    -b #      bandwidth to send at in bits/sec (default 1 Mbit/sec, implies -u)
    -S #      set the IP 'type of service'
    -c <host> run in client mode, connecting to <host>
    -t #      time in seconds to transmit for (default 10 secs)
#
```

iperf -s [options]

- iperf -s

Enable an iperf2 server in TCP mode, which listens on port 5001 by default, and other options are default values.

- iperf -s -u

Enable an iperf2 server in UDP mode, which listens on port 5001 by default, and other options are default values.

- -i <interval>

Set the period of the test results printed by the serial port (Interval column), in seconds, with a range of 1-3600 (which must be integer; non-integer should be rounded down). The default is 1.

- -l <length>

Set the length of the read and write buffer, in bytes. The default is 1460 bytes, the maximum value in UDP mode is 2380, and the maximum value in TCP mode is 4380. The recommended value is 1472 in UDP mode and 1460 in TCP mode.

- -p <port>

Set the port that the server listens on. The port range is 0-65535, and the default is 5001.

iperf -c <hostip> [options]

- iperf -c <hostip>

Enable an iperf2 client, and make a TCP connection with the server whose IP address is <host> on the default port 5001. All other options are default values.

- iperf -c <hostip> -u

Enable an iperf3 client, and make a UDP connection with the server whose IP address is <host> on the default port 5001. All other options are default values.

- -i <interval>

- -l <length>

The -l and -i option settings are the same as those on the server.

- -p <port>

Set the client connection port, which must be the same as the port that the server listens on.

- -b <bandwidth>

The unit of bandwidth is bits/sec and the format is data[KMG]. For example, 50K, 50k or 50000 means that the bandwidth is 50 Kbits/sec; when bandwidth is 0, it means that there is no limit. The default is 1 Mbit/sec. Only used in UDP mode.

- -t <time>

Set the total transmission time. The default is 10 seconds.

- -S <QOS value>

Set the QOS service type of the IP packet. The number range is 0-255, and hexadecimal (0x prefix) or decimal can be used, such as 0x16 = 22.

iperf stop

The command is used to stop the iperf2 test.

iperf2 test example

- Connect the development board and test machine to the same AP, and then view the IP address of the development board.
  - Use the wifi_connect command to connect the development board to the AP, and use the wifi_status command to view the IP address.
  - The test machine opens the iperf2 command window and starts the test.
  - The server first executes the command:
  - iperf -s -p <port> -i <interval> -l <length>   //TCP
  - iperf -s -p <port> -i <interval> -l <length> -u   //UDP
  - The client then executes the command:
  - iperf -c <host> -l <length> -p <port> -i <interval> -b <bandwidth/number> -t <time> -S <number>//TCP
  - iperf -c <host> -l <length> -p <port> -i <interval> -u -b <bandwidth/number> -t <time> -S <number>//UDP
  - The -l, -p, -i, -u, -b, -t, and -S options are optional.
  - ! ! Note: The -p option must be used by the server and client at the same time and have the same value; the -i option does not need to be used by the server and client at the same time and can have different values; the -u option must be used by the server and client at the same time.
  - For example:
  - iperf -s -p 5004 -i 1   //TCP

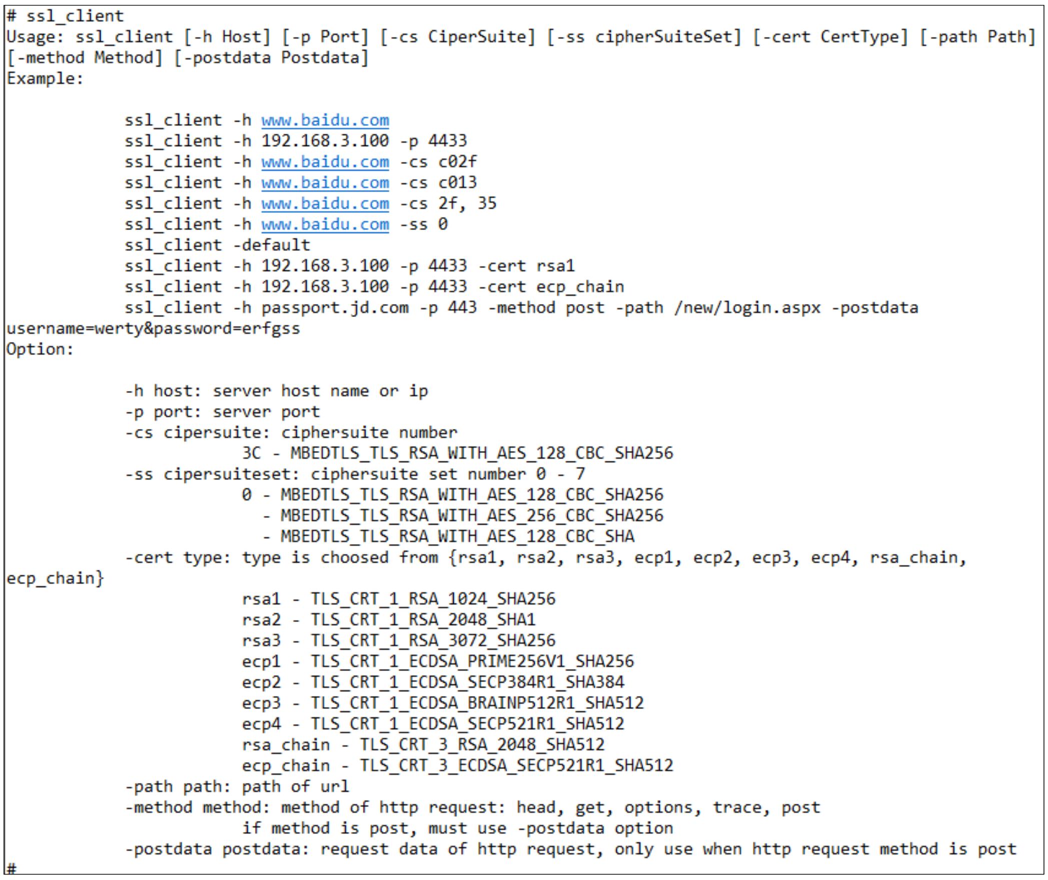
- iperf -s -p 5004 -i 1 –u //UDP
- iperf -c 192.168.1.104 -l 1460 -p 5004 -i 2 -t 20 -S 0xe0 //TCP
- iperf -c 192.168.1.104 -l 1472 -p 5004 -i 4 -t 30 -S 0xe0 -u -b 50M //UDP

■ After the server executes the command, the print information is displayed in the window, telling us that the server is enabled and listening on the corresponding port. After the client executes the command, the test machine and development board will print the test information at the same time.

1.13.5. ssl_client

This command uses the MbedTLS component to implement an HTTPS client that can access and interact with the HTTPS server.

Figure 1-21. ssl_client command



```
# ssl_client
Usage: ssl_client [-h Host] [-p Port] [-cs CiperSuite] [-ss cipherSuiteSet] [-cert CertType] [-path Path]
[-method Method] [-postdata Postdata]
Example:

            ssl_client -h www.baidu.com
            ssl_client -h 192.168.3.100 -p 4433
            ssl_client -h www.baidu.com -cs c02f
            ssl_client -h www.baidu.com -cs c013
            ssl_client -h www.baidu.com -cs 2f, 35
            ssl_client -h www.baidu.com -ss 0
            ssl_client -default
            ssl_client -h 192.168.3.100 -p 4433 -cert rsa1
            ssl_client -h 192.168.3.100 -p 4433 -cert ecp_chain
            ssl_client -h passport.jd.com -p 443 -method post -path /new/login.aspx -postdata
username=werty&password=erfgss
Option:

            -h host: server host name or ip
            -p port: server port
            -cs cipersuite: ciphersuite number
                        3C - MBEDTLS_TLS_RSA_WITH_AES_128_CBC_SHA256
            -ss cipersuiteset: ciphersuite set number 0 - 7
                        0 - MBEDTLS_TLS_RSA_WITH_AES_128_CBC_SHA256
                          - MBEDTLS_TLS_RSA_WITH_AES_256_CBC_SHA256
                          - MBEDTLS_TLS_RSA_WITH_AES_128_CBC_SHA
            -cert type: type is choosed from {rsa1, rsa2, rsa3, ecp1, ecp2, ecp3, ecp4, rsa_chain,
ecp_chain}
                        rsa1 - TLS_CRT_1_RSA_1024_SHA256
                        rsa2 - TLS_CRT_1_RSA_2048_SHA1
                        rsa3 - TLS_CRT_1_RSA_3072_SHA256
                        ecp1 - TLS_CRT_1_ECDSA_PRIME256V1_SHA256
                        ecp2 - TLS_CRT_1_ECDSA_SECP384R1_SHA384
                        ecp3 - TLS_CRT_1_ECDSA_BRAINP512R1_SHA512
                        ecp4 - TLS_CRT_1_ECDSA_SECP521R1_SHA512
                        rsa_chain - TLS_CRT_3_RSA_2048_SHA512
                        ecp_chain - TLS_CRT_3_ECDSA_SECP521R1_SHA512
            -path path: path of url
            -method method: method of http request: head, get, options, trace, post
                        if method is post, must use -postdata option
            -postdata postdata: request data of http request, only use when http request method is post
#
```

As shown in ***Figure 1-21. ssl_client command***,

■ ssl_client -default

Use the default configuration for HTTPS client and then the client can access the server--www.baidu.com.

■ -h host

The domain name or IP address of the server.

- -p Port

The port of the server.

- -cs CiperSuite

The key suite used to access the server.

- -ss cipherSuiteSet

The key suite set used to access the server.

Only one of options -cs and -ss should be used. When both are used at the same time, the option entered later will overwrite the previous option.

- -cert CertType

Use the certificate to access the server. The name of CertType is custom set in the code.

- -path Path

The path is part of a URL address and is used in combination with a domain name.

- -method Method

HTTP request methods, including GET, HEAD, TRACE, POST, etc. But not all of them are supported by the server.

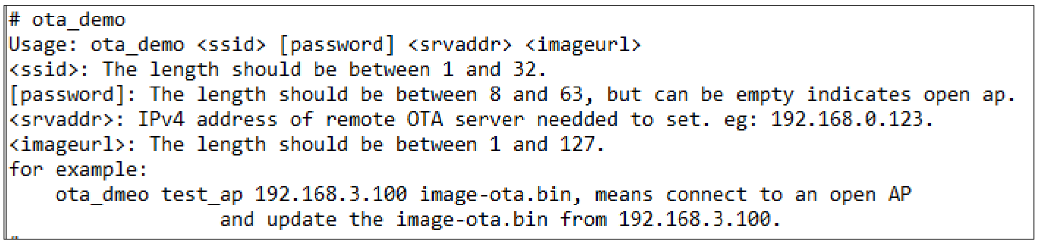
- -postdata Postdata

Input content when HTTP request method POST is used.

1.13.6. ota_demo

This command is an OTA demo. It can get a new firmware from a remote server and then perform a firmware update.

Figure 1-22. ota_demo command



```
# ota_demo
Usage: ota_demo <ssid> [password] <srvaddr> <imageurl>
<ssid>: The length should be between 1 and 32.
[password]: The length should be between 8 and 63, but can be empty indicates open ap.
<srvaddr>: IPv4 address of remote OTA server needded to set. eg: 192.168.0.123.
<imageurl>: The length should be between 1 and 127.
for example:
    ota_dmeo test_ap 192.168.3.100 image-ota.bin, means connect to an open AP
                  and update the image-ota.bin from 192.168.3.100.
```

- ssid

The ssid of an AP. The remote server can be accessed after connecting to this AP.

- password

The password of the AP. No input is required if this AP is an open AP.

- srvaddr

The IPv4 address of the remote server.

- imageurl

The URL address of the new firmware.

1.13.7. ali_cloud

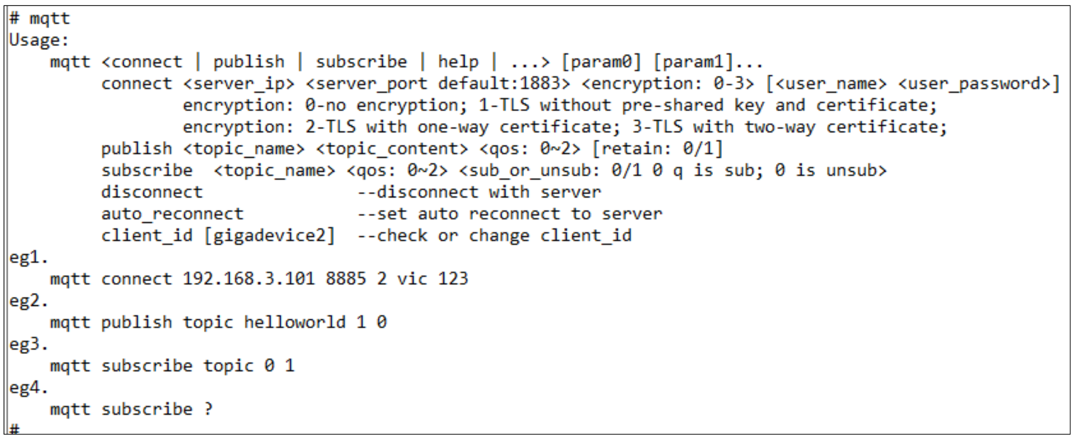
This command is used for device to access Alibaba Cloud. Two methods are supported now, one is smart config and the other is SoftAP.

- Usage: ali_cloud <mode>
- <mode>: 1 - smart config, 2 - softap config, 0 - stop alicloud

1.13.8. mqtt

This command implements a MQTT client that can access MQTT server and subscribe /unsubscribe/publish message.

Figure 1-23. mqtt command



```
# mqtt
Usage:
    mqtt <connect | publish | subscribe | help | ...> [param0] [param1]...
         connect <server_ip> <server_port default:1883> <encryption: 0-3> [<user_name> <user_password>]
                 encryption: 0-no encryption; 1-TLS without pre-shared key and certificate;
                 encryption: 2-TLS with one-way certificate; 3-TLS with two-way certificate;
         publish <topic_name> <topic_content> <qos: 0~2> [retain: 0/1]
         subscribe  <topic_name> <qos: 0~2> <sub_or_unsub: 0/1 0 q is sub; 0 is unsub>
         disconnect                --disconnect with server
         auto_reconnect            --set auto reconnect to server
         client_id [gigadevice2]   --check or change client_id
eg1.
    mqtt connect 192.168.3.101 8885 2 vic 123
eg2.
    mqtt publish topic helloworld 1 0
eg3.
    mqtt subscribe topic 0 1
eg4.
    mqtt subscribe ?
#
```

mqtt help

Print mqtt command introduction.

mqtt connect <server_ip> <server_port default:1883> <encryption: 0-3> [<user_name> <user_password>]

mqtt client connect to the mqtt server.

- server_ip

The IPv4 address or domain name of the server. IPv6 address is not supported.

- server_port

The port of the server.

- encryption

0: no encryption; 1: TLS without pre-shared key and certificate; 2: TLS with one-way certificate; 3: TLS with two-way certificate.

- user_name user_password

The user name and user password provided by the server are not mandatory.

mqtt publish <topic_name> <topic_content> <qos: 0~2> [retain: 0/1]

mqtt client publish message.

- topic_name

The name of the topic to which the message belongs..

- topic_content

The content of the message.

- qos

0: The receiver receives the massage at most once; 1: The receiver receives the massage at least once; 2: The receiver receives the massage just once.

- retain

0: The server dose not save the message as a retain message; 1: The server saves the message as a retain message.

mqtt subscribe <topic_name> <qos: 0~2> <sub_or_unsub: 0/1>

mqtt client subscribe/unsubscribe message.

- topic_name

The name of the topic to subscribe/unsubscribe to.

- qos

The same as above.

- sub_or_unsub

0: unsubscribe; 1: subscribe.

mqtt disconnect

mqtt client disconnect from the mqtt server.

mqtt auto_reconnect [0: disable; 1: enable]

mqtt client auto reconnect setting. 0: disable auto reconnect; 1: enable auto reconnect.

mqtt client_id [new client id]

mqtt client change client id. The current client id is printed when no parameters are entered.

1.13.9. azure

This command has no option.

This command is used for device to access Azure Cloud.

1.13.10. coap_client

This command implements an coap client that can access and modify the resources on coap server corresponding to the URI.

- Usage: coap_client [-m get|put] [-v log_level] [-N] <URI> [data]
- -m get|put

get: The client uses the GET method to access the URI resource.

put: The client uses the PUT method to update the URI resource.

- -v log_level

Specify the log level of the current client. The range of log level is 0-8, which corresponds to EMERG/ALERT/CRIT/ERR/WARN/NOTICE/INFO/DEBUG/OSCORE respectively. The default log level is 6-INFO.

- -N

If the command carries the -N option, it indicates that the message type sent by current client is Non-Confirmable Message, otherwise CON-Confirmable Message.

- URI

The address and resource label of the current server. For example, coap://192.168.1.1/example, indicates that the server is located in 192.168.1.1 and need to access the resource with URI example on the server.

- data

When client uses the PUT method, [data] is the specfic content of the corresponding URI resource within the server that client updates.

1.13.11. coap_server

This command implements an coap server.

- coap_server

Start an coap server.

- coap_server stop

Stop an coap server.

1.13.12. socket_client

This command uses the LwIP Sockets API to implement an TCP/UDP client that can connect and communicate with the server.

- socket_client <0:TCP or 1:UDP> <remote ip> <remote port>
- remote ip: the IPv4 address of the server. remote port: the port of the server.

1.13.13. socket_server

This command uses the LwIP Sockets API to implement an TCP/UDP server and client can connect and communicate with it.

- socket_server <0:TCP or 1:UDP> <server port>
- server port: the port of the server.

1.13.14. socket_close

This command is used to close TCP/UDP client/server.

- socket_close <fd>
- fd: the socket descriptor of TCP/UDP client/server.

1.13.15. socket_get_status

This command is used to get the status of TCP/UDP client/server which is implemented using LwIP Sockets API.

This command has no option.

1.14. BLE

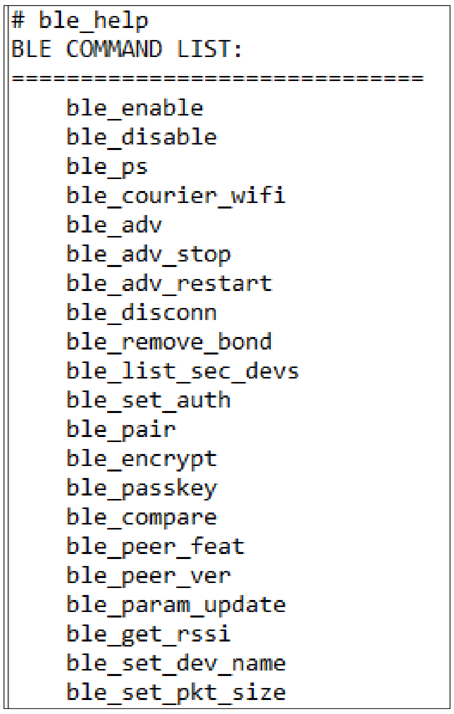
This section introduces BLE related commands.

1.14.1. ble_help

This command has no option.

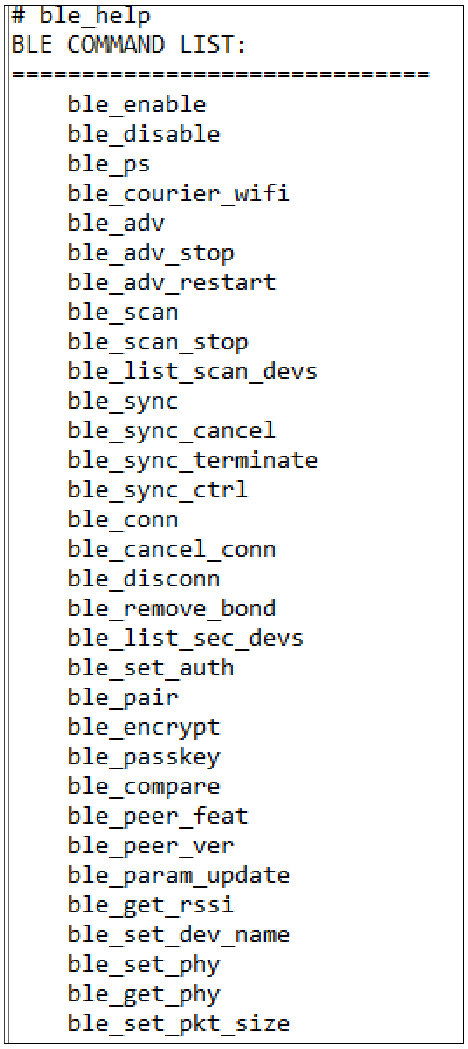
As shown in *Figure 1-24. ble_help command (msdk configuration)* and *Figure 1-25. ble_help command (msdk_ffd configuration)*, the ble_help command will list all commands of BLE. Different BLE commands can be executed in different configurations, so the ble_help command will list different BLE commands.

Figure 1-24. ble_help command (msdk configuration)



```
# ble_help
BLE COMMAND LIST:
==============================
    ble_enable
    ble_disable
    ble_ps
    ble_courier_wifi
    ble_adv
    ble_adv_stop
    ble_adv_restart
    ble_disconn
    ble_remove_bond
    ble_list_sec_devs
    ble_set_auth
    ble_pair
    ble_encrypt
    ble_passkey
    ble_compare
    ble_peer_feat
    ble_peer_ver
    ble_param_update
    ble_get_rssi
    ble_set_dev_name
    ble_set_pkt_size
```

Figure 1-25. ble_help command (msdk_ffd configuration)


```
# ble_help
BLE COMMAND LIST:
==============================
    ble_enable
    ble_disable
    ble_ps
    ble_courier_wifi
    ble_adv
    ble_adv_stop
    ble_adv_restart
    ble_scan
    ble_scan_stop
    ble_list_scan_devs
    ble_sync
    ble_sync_cancel
    ble_sync_terminate
    ble_sync_ctrl
    ble_conn
    ble_cancel_conn
    ble_disconn
    ble_remove_bond
    ble_list_sec_devs
    ble_set_auth
    ble_pair
    ble_encrypt
    ble_passkey
    ble_compare
    ble_peer_feat
    ble_peer_ver
    ble_param_update
    ble_get_rssi
    ble_set_dev_name
    ble_set_phy
    ble_get_phy
    ble_set_pkt_size
```

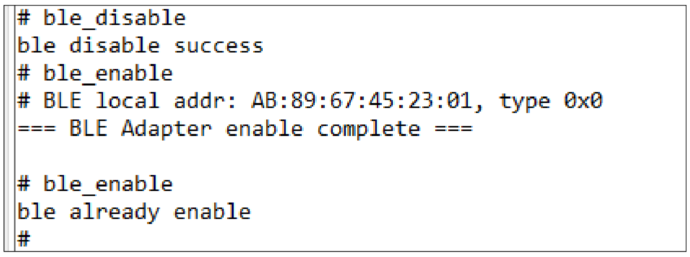
1.14.2. ble_enable

This command has no option.

ble_enable is used to enable BLE. The execution of other BLE related commands takes effect only when BLE is enabled. After the development board is started correctly, BLE is enabled by default, so this command does not need to be executed. This command is usually used together with ble_disable. If the ble_enable command is used after BLE is disabled, BLE will enter the initial state and will not resume to the state before the ble_disable operation.

As shown in ***Figure 1-26. ble_enable command***, after BLE is disabled, executing ble_enable will enable BLE, and the serial port will display the reset log; if BLE is already enabled, the serial port will prompt that BLE is enabled.

Figure 1-26. ble_enable command



```
# ble_disable
ble disable success
# ble_enable
# BLE local addr: AB:89:67:45:23:01, type 0x0
=== BLE Adapter enable complete ===

# ble_enable
ble already enable
#
```

1.14.3. ble_disable

This command has no option.

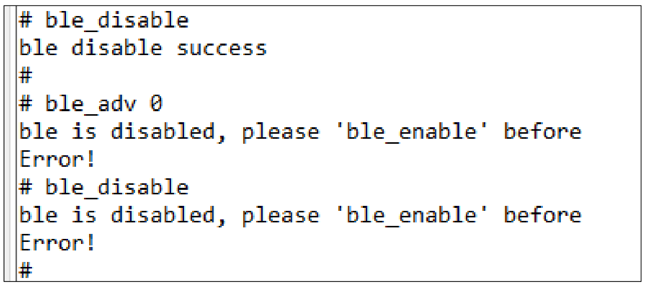
ble_disable can be used to disable BLE. After BLE is disabled, some commands such as ble_adv, ble_scan, and ble_conn cannot be executed.

The BLE software and hardware will be reset after this command is executed, and then BLE will be disabled, so the command has slightly different execution results for different conditions of the development board, for example:

- If the development board does not enable any function of BLE, BLE will be disabled directly;
- If the development board has created a connection, the connection will be disconnected and then BLE will be disabled;
- If the development board has enabled advertising, it will stop advertising and then BLE will be disabled;
- If the development board has enabled scanning, it will stop scanning and then BLE will be disabled;
- If BLE is disabled, the serial port will prompt that BLE is disabled.

As shown in ***Figure 1-27. ble_disable command***, a prompt will be printed after ble_disable is executed.

Figure 1-27. ble_disable command



```
# ble_disable
ble disable success
#
# ble_adv 0
ble is disabled, please 'ble_enable' before
Error!
# ble_disable
ble is disabled, please 'ble_enable' before
Error!
#
```

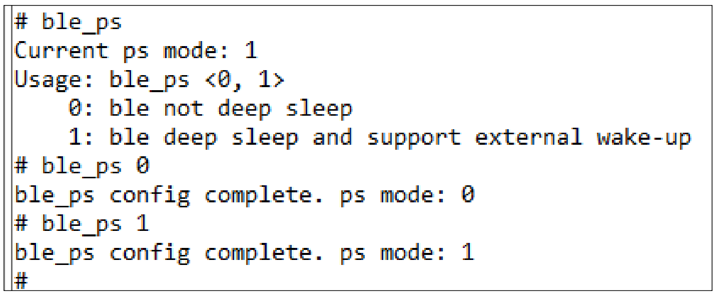
1.14.4. ble_ps

- Usage: ble_ps <0 or 1>

This command is used to configure the power save function of BLE which is enabled by default. When ps mode is 1, the power save mode is enabled. When nothing of BLE is being processed or adv/scan/connection interval is greater than 5 ms, BLE core will be able to enter the sleep mode to save power. When ps mode is 0, the power save mode is disabled, and BLE core will not enter the sleep mode.

As shown in ***Figure 1-28. ble_ps command***, a prompt will be printed after ble_ps is executed.

Figure 1-28. ble_ps command



```
# ble_ps
Current ps mode: 1
Usage: ble_ps <0, 1>
    0: ble not deep sleep
    1: ble deep sleep and support external wake-up
# ble_ps 0
ble_ps config complete. ps mode: 0
# ble_ps 1
ble_ps config complete. ps mode: 1
#
```

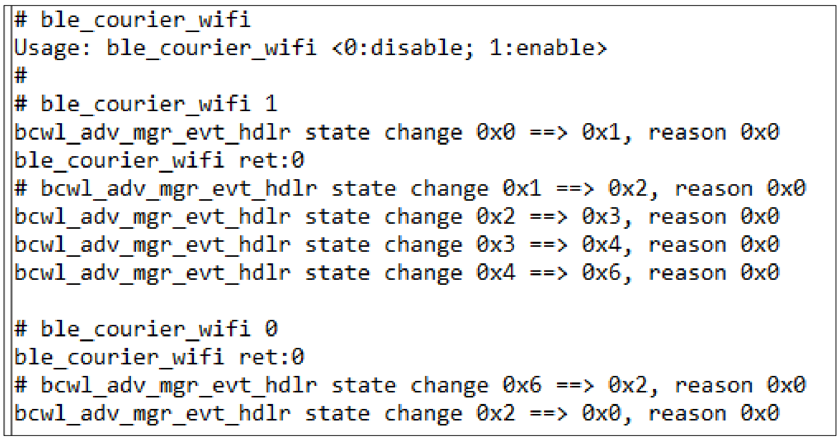
1.14.5. ble_courier_wifi

- Usage: ble_courier_wifi <0:disable or 1:enable>

This command is used to enable and disable the Bluetooth distribution network (Wi-Fi network configuration) function, which is disabled by default. After the function is enabled, the device will send advertising packets for the mobile phone to discover. The WeChat applet "GD Bluetooth distribution network" can be used to connect with the development board and continue the distribution operations. When the function is disabled, advertising will also be disabled.

As shown in ***Figure 1-29. ble_courier_wifi command***, a prompt will be printed after ble_courier_wifi is executed.

Figure 1-29. ble_courier_wifi command



```
# ble_courier_wifi
Usage: ble_courier_wifi <0:disable; 1:enable>
#
# ble_courier_wifi 1
bcwl_adv_mgr_evt_hdlr state change 0x0 ==> 0x1, reason 0x0
ble_courier_wifi ret:0
# bcwl_adv_mgr_evt_hdlr state change 0x1 ==> 0x2, reason 0x0
bcwl_adv_mgr_evt_hdlr state change 0x2 ==> 0x3, reason 0x0
bcwl_adv_mgr_evt_hdlr state change 0x3 ==> 0x4, reason 0x0
bcwl_adv_mgr_evt_hdlr state change 0x4 ==> 0x6, reason 0x0

# ble_courier_wifi 0
ble_courier_wifi ret:0
# bcwl_adv_mgr_evt_hdlr state change 0x6 ==> 0x2, reason 0x0
bcwl_adv_mgr_evt_hdlr state change 0x2 ==> 0x0, reason 0x0
```

1.14.6. ble_adv

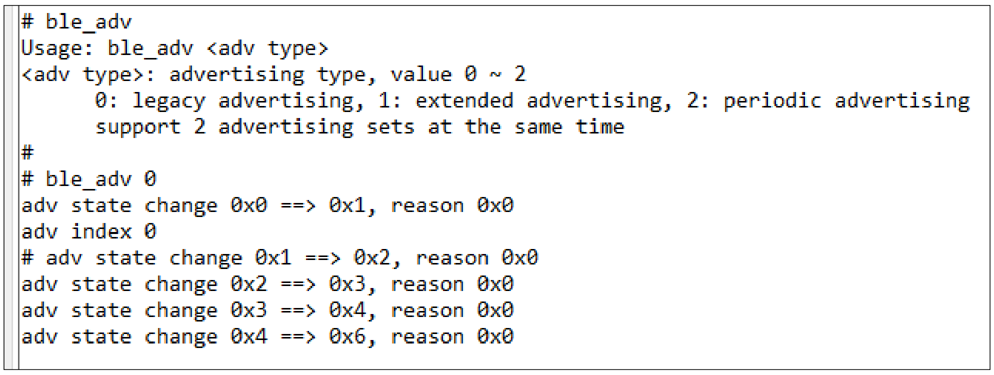
- Usage: ble_adv <adv type>

This command is used to enable advertising so that local device can be found and connected by other BLE devices. The advertising type can be set to legacy advertising (scannable connectable undirected), extended advertising (connectable undirected), or periodic advertising (undirected periodic) through adv type parameter. Only 1 advertising set is supported in msdk configuration while 2 advertising sets are supported in msdk_ffd configuration.

If the device is connected by other devices, advertising will be stopped but will not be removed.

As shown in ***Figure 1-30. ble_adv command***, a prompt will be printed after ble_adv is executed. When adv state is 0x6, it means success; otherwise, the execution fails. Adv index will also be prompted and can be used for the ble_adv_stop or ble_adv_restart command. For example, adv idx in the figure below is 0.

Figure 1-30. ble_adv command


```
# ble_adv
Usage: ble_adv <adv type>
<adv type>: advertising type, value 0 ~ 2
      0: legacy advertising, 1: extended advertising, 2: periodic advertising
      support 2 advertising sets at the same time
#
# ble_adv 0
adv state change 0x0 ==> 0x1, reason 0x0
adv index 0
# adv state change 0x1 ==> 0x2, reason 0x0
adv state change 0x2 ==> 0x3, reason 0x0
adv state change 0x3 ==> 0x4, reason 0x0
adv state change 0x4 ==> 0x6, reason 0x0
```

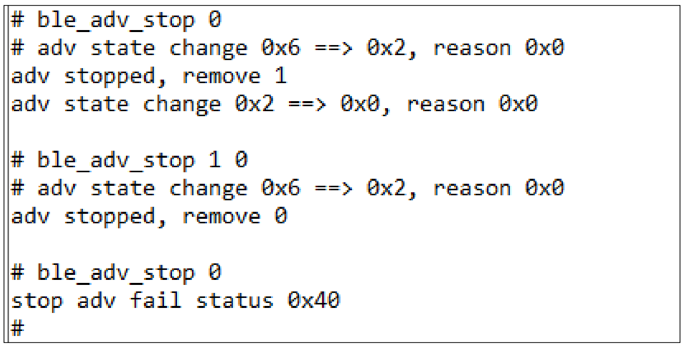
1.14.7. ble_adv_stop

- Usage: ble_adv_stop <adv idx> [remove]
- adv idx: advertising index, which can be obtained from the log of executing the ble_adv command
- remove: whether advertising needs to be removed after it is stopped. The default value is 1, and in this case, advertising will be removed after it is stopped. If the value is set to 0, advertising will not be removed, and can be enabled again through ble_adv_restart. This operation skips the creation procedure compared with the operation of enabling advertising through ble_adv.

This command is used to disable advertising.

As shown in ***Figure 1-31. ble_adv_stop command***, a prompt will be printed after ble_adv_stop is executed. When an invalid adv idx is used, a fail prompt and the non-zero status will be displayed.

Figure 1-31. ble_adv_stop command



```
# ble_adv_stop 0
# adv state change 0x6 ==> 0x2, reason 0x0
adv stopped, remove 1
adv state change 0x2 ==> 0x0, reason 0x0

# ble_adv_stop 1 0
# adv state change 0x6 ==> 0x2, reason 0x0
adv stopped, remove 0

# ble_adv_stop 0
stop adv fail status 0x40
#
```

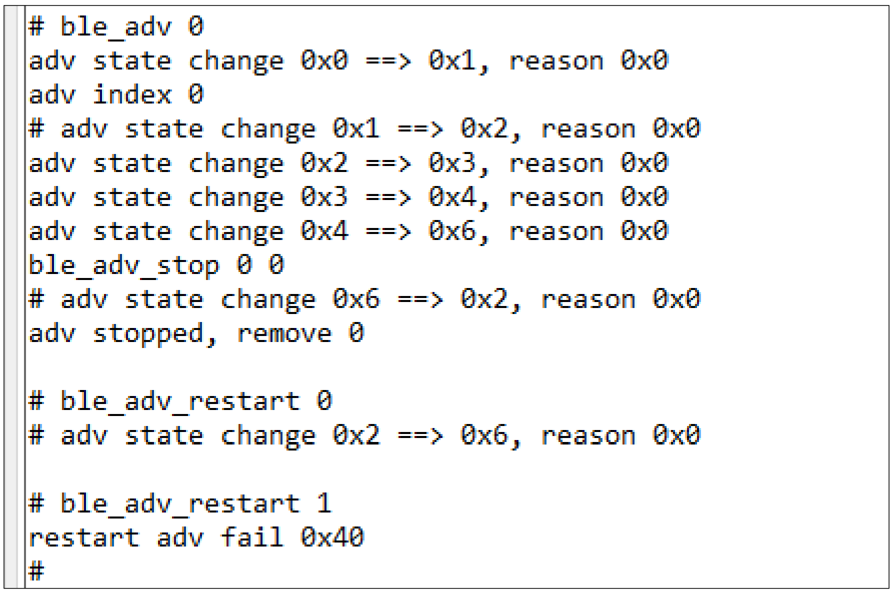
1.14.8. ble_adv_restart

- Usage: ble_adv_restart <adv idx>
- adv idx: advertising index, which can be obtained from the log of executing the ble_adv command

This command is used to restart advertising which is in stopped state. In the following two scenarios, the advertising can be restarted through ble_adv_restart : 1. after ble_adv enables advertising and a connection is established, advertising will be stopped; 2. after 'ble_adv_stop idx 0' is executed, advertising will be stopped but not removed.

As shown in ***Figure 1-32. ble_adv_restart command***, a prompt will be printed after ble_adv_restart is executed. When adv state is 0x6, it means restart success; otherwise, it means failure; if adv idx is an illegal index, a failure log will be printed.

Figure 1-32. ble_adv_restart command



```
# ble_adv 0
adv state change 0x0 ==> 0x1, reason 0x0
adv index 0
# adv state change 0x1 ==> 0x2, reason 0x0
adv state change 0x2 ==> 0x3, reason 0x0
adv state change 0x3 ==> 0x4, reason 0x0
adv state change 0x4 ==> 0x6, reason 0x0
ble_adv_stop 0 0
# adv state change 0x6 ==> 0x2, reason 0x0
adv stopped, remove 0

# ble_adv_restart 0
# adv state change 0x2 ==> 0x6, reason 0x0

# ble_adv_restart 1
restart adv fail 0x40
#
```

1.14.9. ble_scan

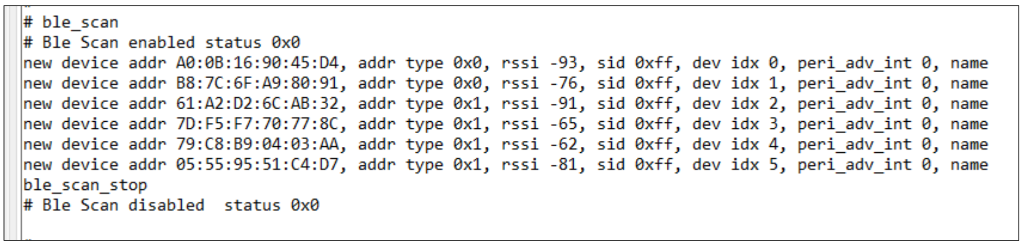
This command has no option.

This command can only be used in msdk_ffd configuration.

This command is used to enable BLE scan procedure, which will remove the information scanned last time and print out information of the devices scanned this time, including the device address, device address type, rssi, name, dev idx, etc. Of which, dev idx can be used to connect or sync. ble_scan_stop can be used to stop the scan procedure.

As shown in ***Figure 1-33. ble scan command***, a prompt will be printed after ble_scan is executed.

Figure 1-33. ble_scan command



```
# ble_scan
# Ble Scan enabled status 0x0
new device addr A0:0B:16:90:45:D4, addr type 0x0, rssi -93, sid 0xff, dev idx 0, peri_adv_int 0, name
new device addr B8:7C:6F:A9:80:91, addr type 0x0, rssi -76, sid 0xff, dev idx 1, peri_adv_int 0, name
new device addr 61:A2:D2:6C:AB:32, addr type 0x1, rssi -91, sid 0xff, dev idx 2, peri_adv_int 0, name
new device addr 7D:F5:F7:70:77:8C, addr type 0x1, rssi -65, sid 0xff, dev idx 3, peri_adv_int 0, name
new device addr 79:C8:B9:04:03:AA, addr type 0x1, rssi -62, sid 0xff, dev idx 4, peri_adv_int 0, name
new device addr 05:55:95:51:C4:D7, addr type 0x1, rssi -81, sid 0xff, dev idx 5, peri_adv_int 0, name
ble_scan_stop
# Ble Scan disabled  status 0x0
```

1.14.10. ble_scan_stop

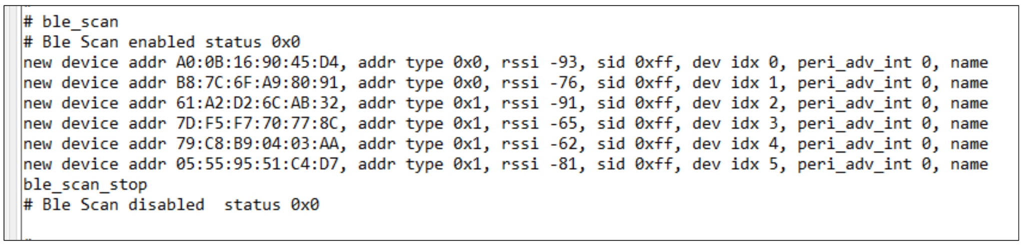
This command has no option.

This command can only be used in msdk_ffd configuration.

This command is used to disable the scan procedure. The status is 0 after success; otherwise, it fails.

As shown in ***Figure 1-34. ble scan stop command***, a prompt will be printed after ble_scan_stop is executed.

Figure 1-34. ble_scan_stop command



```
# ble_scan
# Ble Scan enabled status 0x0
new device addr A0:0B:16:90:45:D4, addr type 0x0, rssi -93, sid 0xff, dev idx 0, peri_adv_int 0, name
new device addr B8:7C:6F:A9:80:91, addr type 0x0, rssi -76, sid 0xff, dev idx 1, peri_adv_int 0, name
new device addr 61:A2:D2:6C:AB:32, addr type 0x1, rssi -91, sid 0xff, dev idx 2, peri_adv_int 0, name
new device addr 7D:F5:F7:70:77:8C, addr type 0x1, rssi -65, sid 0xff, dev idx 3, peri_adv_int 0, name
new device addr 79:C8:B9:04:03:AA, addr type 0x1, rssi -62, sid 0xff, dev idx 4, peri_adv_int 0, name
new device addr 05:55:95:51:C4:D7, addr type 0x1, rssi -81, sid 0xff, dev idx 5, peri_adv_int 0, name
ble_scan_stop
# Ble Scan disabled  status 0x0
```

1.14.11. ble_list_scan_devs

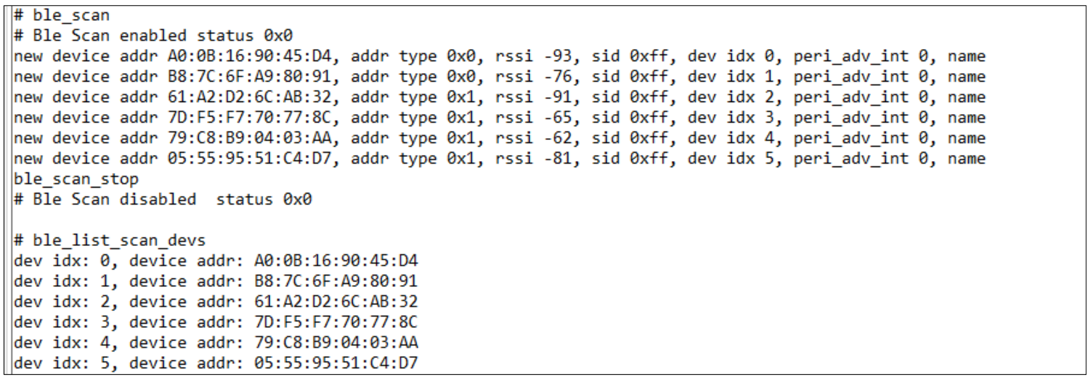
This command has no option.

This command can only be used in msdk_ffd configuration.

This command is used to query the latest scanned devices and device index and device address will be displayed.

As shown in *Figure 1-35. ble_list_scan_devs command*, a prompt will be printed after ble_list_scan_devs is executed.

Figure 1-35. ble_list_scan_devs command



```
# ble_scan
# Ble Scan enabled status 0x0
new device addr A0:0B:16:90:45:D4, addr type 0x0, rssi -93, sid 0xff, dev idx 0, peri_adv_int 0, name
new device addr B8:7C:6F:A9:80:91, addr type 0x0, rssi -76, sid 0xff, dev idx 1, peri_adv_int 0, name
new device addr 61:A2:D2:6C:AB:32, addr type 0x1, rssi -91, sid 0xff, dev idx 2, peri_adv_int 0, name
new device addr 7D:F5:F7:70:77:8C, addr type 0x1, rssi -65, sid 0xff, dev idx 3, peri_adv_int 0, name
new device addr 79:C8:B9:04:03:AA, addr type 0x1, rssi -62, sid 0xff, dev idx 4, peri_adv_int 0, name
new device addr 05:55:95:51:C4:D7, addr type 0x1, rssi -81, sid 0xff, dev idx 5, peri_adv_int 0, name
ble_scan_stop
# Ble Scan disabled  status 0x0

# ble_list_scan_devs
dev idx: 0, device addr: A0:0B:16:90:45:D4
dev idx: 1, device addr: B8:7C:6F:A9:80:91
dev idx: 2, device addr: 61:A2:D2:6C:AB:32
dev idx: 3, device addr: 7D:F5:F7:70:77:8C
dev idx: 4, device addr: 79:C8:B9:04:03:AA
dev idx: 5, device addr: 05:55:95:51:C4:D7
```

1.14.12. ble_sync

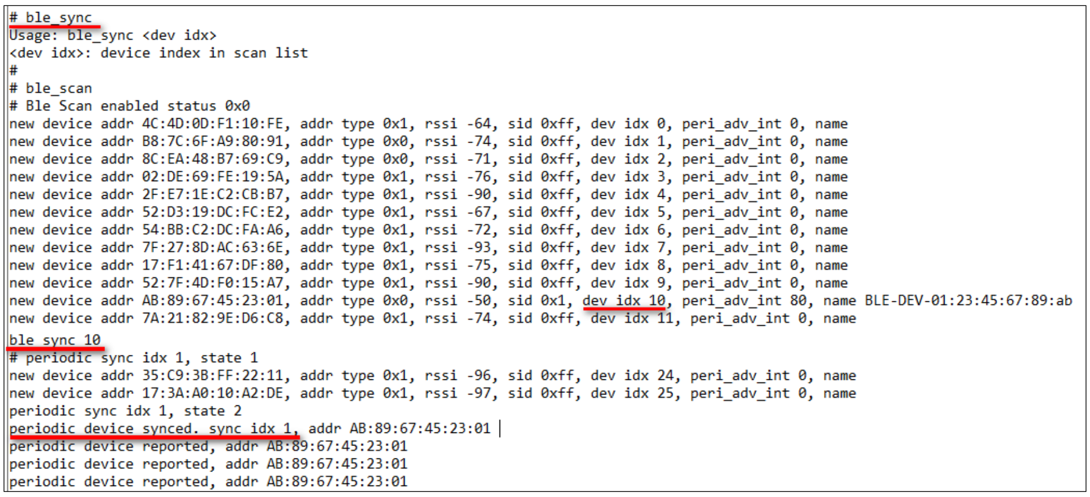
- Usage: ble_sync <dev idx>
- dev idx needs to be obtained from the scan list.

This command can only be used in msdk_ffd configuration.

This command is used to synchronize with a periodic advertising. BLE scan must be kept enabled until synchronization is established successfully. If sync is successful, the sync idx will be printed for the ble_sync_terminate or ble_syc_ctrl command to use. The periodic advertising report function is enabled by default, so the application will periodically print periodic advertising report logs. The ble_sync_ctrl command can be used to disable the report function.

As shown in *Figure 1-36. ble_sync command*, a prompt will be printed after ble_sync is executed.

Figure 1-36. ble_sync command



```
# ble_sync
Usage: ble_sync <dev idx>
<dev idx>: device index in scan list
#
# ble_scan
# Ble Scan enabled status 0x0
new device addr 4C:4D:0D:F1:10:FE, addr type 0x1, rssi -64, sid 0xff, dev idx 0, peri_adv_int 0, name
new device addr B8:7C:6F:A9:80:91, addr type 0x0, rssi -74, sid 0xff, dev idx 1, peri_adv_int 0, name
new device addr 8C:EA:48:B7:69:C9, addr type 0x0, rssi -71, sid 0xff, dev idx 2, peri_adv_int 0, name
new device addr 02:DE:69:FE:19:5A, addr type 0x1, rssi -76, sid 0xff, dev idx 3, peri_adv_int 0, name
new device addr 2F:E7:1E:C2:CB:B7, addr type 0x1, rssi -90, sid 0xff, dev idx 4, peri_adv_int 0, name
new device addr 52:D3:19:DC:FC:E2, addr type 0x1, rssi -67, sid 0xff, dev idx 5, peri_adv_int 0, name
new device addr 54:BB:C2:DC:FA:A6, addr type 0x1, rssi -72, sid 0xff, dev idx 6, peri_adv_int 0, name
new device addr 7F:27:8D:AC:63:6E, addr type 0x1, rssi -93, sid 0xff, dev idx 7, peri_adv_int 0, name
new device addr 17:F1:41:67:DF:80, addr type 0x1, rssi -75, sid 0xff, dev idx 8, peri_adv_int 0, name
new device addr 52:7F:4D:F0:15:A7, addr type 0x1, rssi -90, sid 0xff, dev idx 9, peri_adv_int 0, name
new device addr AB:89:67:45:23:01, addr type 0x0, rssi -50, sid 0x1, dev idx 10, peri_adv_int 80, name BLE-DEV-01:23:45:67:89:ab
new device addr 7A:21:82:9E:D6:C8, addr type 0x1, rssi -74, sid 0xff, dev idx 11, peri_adv_int 0, name
ble_sync 10
# periodic sync idx 1, state 1
new device addr 35:C9:3B:FF:22:11, addr type 0x1, rssi -96, sid 0xff, dev idx 24, peri_adv_int 0, name
new device addr 17:3A:A0:10:A2:DE, addr type 0x1, rssi -97, sid 0xff, dev idx 25, peri_adv_int 0, name
periodic sync idx 1, state 2
periodic device synced. sync idx 1, addr AB:89:67:45:23:01
periodic device reported, addr AB:89:67:45:23:01
periodic device reported, addr AB:89:67:45:23:01
periodic device reported, addr AB:89:67:45:23:01
```

1.14.13. ble_sync_cancel

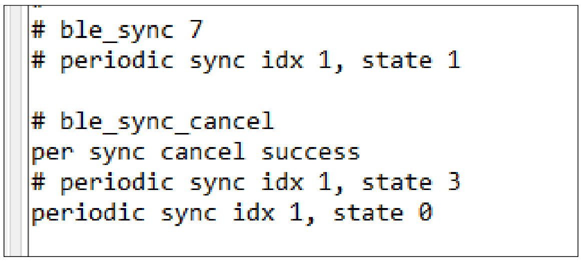
This command has no option.

This command can only be used in msdk_ffd configuration.

When sync is created but unestablished, this command can be used to cancel the operation.

As shown in ***Figure 1-37. ble sync cancel command***, a prompt will be printed after ble_sync_cancel is executed.

Figure 1-37. ble_sync_cancel command



```
# ble_sync 7
# periodic sync idx 1, state 1

# ble_sync_cancel
per sync cancel success
# periodic sync idx 1, state 3
periodic sync idx 1, state 0
```

1.14.14. ble_sync_terminate

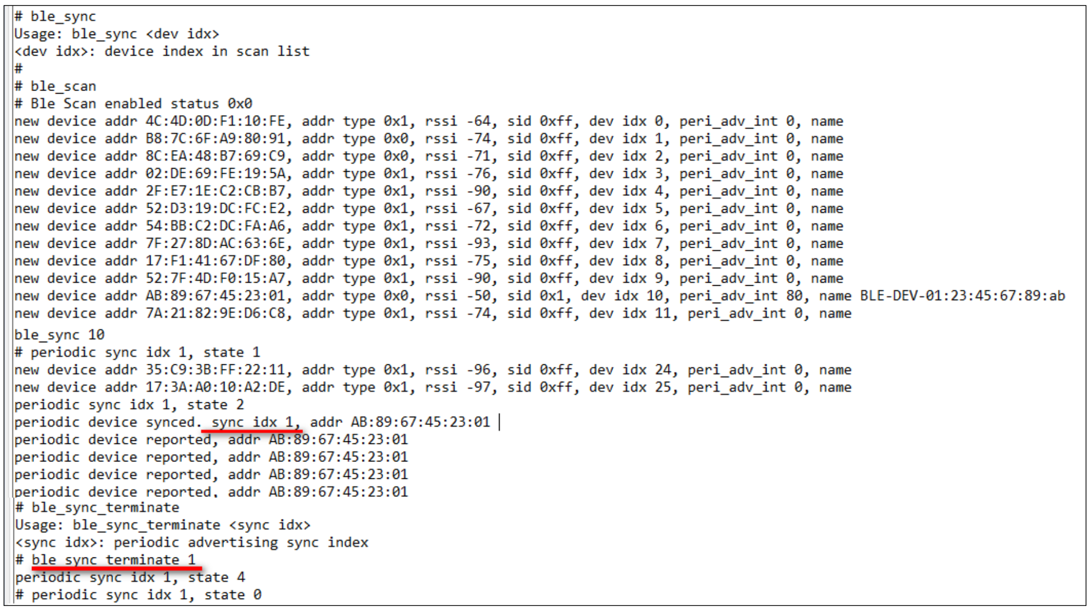
- Usage: ble_sync_terminate <sync idx>
- sync idx needs to be obtained from the log of successful establishment of sync through the ble_sync command.

This command is used to terminate the specified sync link.

This command can only be used in msdk_ffd configuration.

As shown in ***Figure 1-38. ble sync terminate command***, a prompt will be printed after ble_sync_terminate is executed.

Figure 1-38. ble_sync_terminate command



```
# ble_sync
Usage: ble_sync <dev idx>
<dev idx>: device index in scan list
#
# ble_scan
# Ble Scan enabled status 0x0
new device addr 4C:4D:0D:F1:10:FE, addr type 0x1, rssi -64, sid 0xff, dev idx 0, peri_adv_int 0, name
new device addr B8:7C:6F:A9:80:91, addr type 0x0, rssi -74, sid 0xff, dev idx 1, peri_adv_int 0, name
new device addr 8C:EA:48:B7:69:C9, addr type 0x0, rssi -71, sid 0xff, dev idx 2, peri_adv_int 0, name
new device addr 02:DE:69:FE:19:5A, addr type 0x1, rssi -76, sid 0xff, dev idx 3, peri_adv_int 0, name
new device addr 2F:E7:1E:C2:CB:B7, addr type 0x1, rssi -90, sid 0xff, dev idx 4, peri_adv_int 0, name
new device addr 52:D3:19:DC:FC:E2, addr type 0x1, rssi -67, sid 0xff, dev idx 5, peri_adv_int 0, name
new device addr 54:BB:C2:DC:FA:A6, addr type 0x1, rssi -72, sid 0xff, dev idx 6, peri_adv_int 0, name
new device addr 7F:27:8D:AC:63:6E, addr type 0x1, rssi -93, sid 0xff, dev idx 7, peri_adv_int 0, name
new device addr 17:F1:41:67:DF:80, addr type 0x1, rssi -75, sid 0xff, dev idx 8, peri_adv_int 0, name
new device addr 52:7F:4D:F0:15:A7, addr type 0x1, rssi -90, sid 0xff, dev idx 9, peri_adv_int 0, name
new device addr AB:89:67:45:23:01, addr type 0x0, rssi -50, sid 0x1, dev idx 10, peri_adv_int 80, name BLE-DEV-01:23:45:67:89:ab
new device addr 7A:21:82:9E:D6:C8, addr type 0x1, rssi -74, sid 0xff, dev idx 11, peri_adv_int 0, name
ble_sync 10
# periodic sync idx 1, state 1
new device addr 35:C9:3B:FF:22:11, addr type 0x1, rssi -96, sid 0xff, dev idx 24, peri_adv_int 0, name
new device addr 17:3A:A0:10:A2:DE, addr type 0x1, rssi -97, sid 0xff, dev idx 25, peri_adv_int 0, name
periodic sync idx 1, state 2
periodic device synced. sync idx 1, addr AB:89:67:45:23:01 |
periodic device reported, addr AB:89:67:45:23:01
periodic device reported, addr AB:89:67:45:23:01
periodic device reported, addr AB:89:67:45:23:01
periodic device reported, addr AB:89:67:45:23:01
# ble_sync_terminate
Usage: ble_sync_terminate <sync idx>
<sync idx>: periodic advertising sync index
# ble_sync_terminate 1
periodic sync idx 1, state 4
# periodic sync idx 1, state 0
```

1.14.15. ble_sync_ctrl

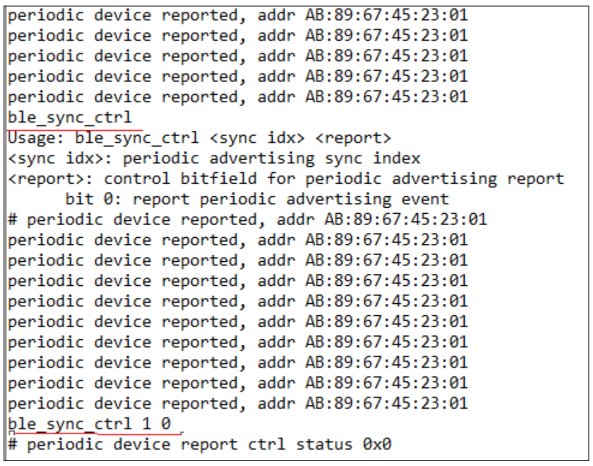
- Usage: ble_sync_ctrl <sync idx> <report>
- sync idx needs to be obtained from the log of successful establishment of sync through the ble_sync command.

This command can only be used in msdk_ffd configuration.

This command is used to enable or disable the periodic advertising report function, which is enabled by default. When enabled, every time a synchronized message is received, it will be reported to APP.

As shown in ***Figure 1-39. ble sync ctrl command***, a prompt will be printed after ble_sync_ctrl is executed.

Figure 1-39. ble_sync_ctrl command



```
periodic device reported, addr AB:89:67:45:23:01
periodic device reported, addr AB:89:67:45:23:01
periodic device reported, addr AB:89:67:45:23:01
periodic device reported, addr AB:89:67:45:23:01
periodic device reported, addr AB:89:67:45:23:01
ble_sync_ctrl
Usage: ble_sync_ctrl <sync idx> <report>
<sync idx>: periodic advertising sync index
<report>: control bitfield for periodic advertising report
      bit 0: report periodic advertising event
# periodic device reported, addr AB:89:67:45:23:01
periodic device reported, addr AB:89:67:45:23:01
periodic device reported, addr AB:89:67:45:23:01
periodic device reported, addr AB:89:67:45:23:01
periodic device reported, addr AB:89:67:45:23:01
periodic device reported, addr AB:89:67:45:23:01
periodic device reported, addr AB:89:67:45:23:01
periodic device reported, addr AB:89:67:45:23:01
periodic device reported, addr AB:89:67:45:23:01
periodic device reported, addr AB:89:67:45:23:01
ble_sync_ctrl 1 0
# periodic device report ctrl status 0x0
```

1.14.16. ble_conn

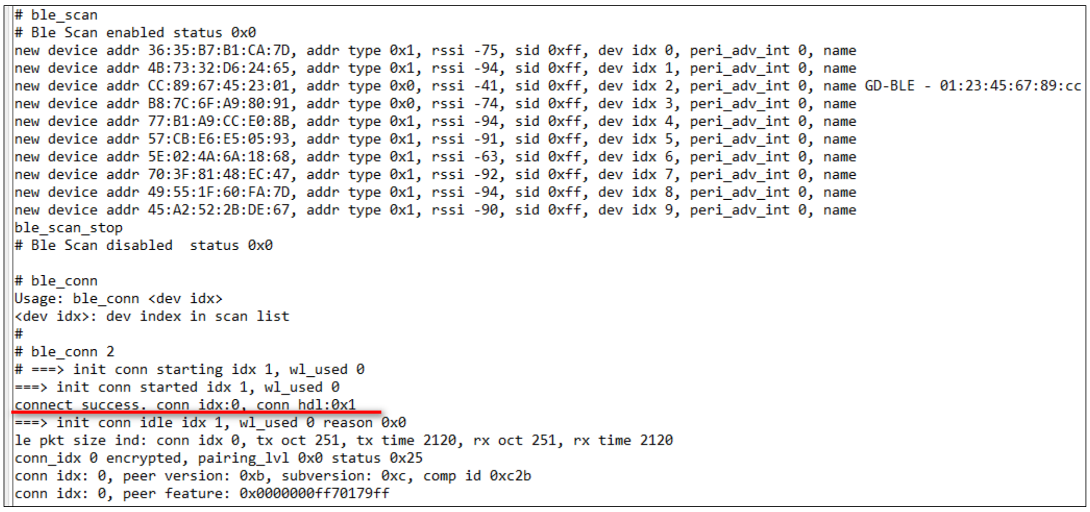
- Usage: ble_conn <dev idx>
- dev idx needs to be obtained from the scan list.

This command can only be used in msdk_ffd configuration.

This command is used to initiate a connection. Before executing this command, ble_scan should be used to get dev idx in the scanned list. If the target device is not scanned, the connection cannot be established.

As shown in ***Figure 1-40. ble_conn command***, a prompt will be printed after ble_conn is executed. If the connection is successfully established, the log with a red underline in ***Figure 1-40. ble_conn command*** will be printed in which conn idx can be used in commands such as ble_disconn, ble_pair, and ble_encrypt.

Figure 1-40. ble_conn command



```
# ble_scan
# Ble Scan enabled status 0x0
new device addr 36:35:B7:B1:CA:7D, addr type 0x1, rssi -75, sid 0xff, dev idx 0, peri_adv_int 0, name
new device addr 4B:73:32:D6:24:65, addr type 0x1, rssi -94, sid 0xff, dev idx 1, peri_adv_int 0, name
new device addr CC:89:67:45:23:01, addr type 0x0, rssi -41, sid 0xff, dev idx 2, peri_adv_int 0, name GD-BLE - 01:23:45:67:89:cc
new device addr B8:7C:6F:A9:80:91, addr type 0x0, rssi -74, sid 0xff, dev idx 3, peri_adv_int 0, name
new device addr 77:B1:A9:CC:E0:8B, addr type 0x1, rssi -94, sid 0xff, dev idx 4, peri_adv_int 0, name
new device addr 57:CB:E6:E5:05:93, addr type 0x1, rssi -91, sid 0xff, dev idx 5, peri_adv_int 0, name
new device addr 5E:02:4A:6A:18:68, addr type 0x1, rssi -63, sid 0xff, dev idx 6, peri_adv_int 0, name
new device addr 70:3F:81:48:EC:47, addr type 0x1, rssi -92, sid 0xff, dev idx 7, peri_adv_int 0, name
new device addr 49:55:1F:60:FA:7D, addr type 0x1, rssi -94, sid 0xff, dev idx 8, peri_adv_int 0, name
new device addr 45:A2:52:2B:DE:67, addr type 0x1, rssi -90, sid 0xff, dev idx 9, peri_adv_int 0, name
ble_scan_stop
# Ble Scan disabled  status 0x0

# ble_conn
Usage: ble_conn <dev idx>
<dev idx>: dev index in scan list
#
# ble_conn 2
# ===> init conn starting idx 1, wl_used 0
===> init conn started idx 1, wl_used 0
connect success. conn idx:0, conn hdl:0x1
===> init conn idle idx 1, wl_used 0 reason 0x0
le pkt size ind: conn idx 0, tx oct 251, tx time 2120, rx oct 251, rx time 2120
conn_idx 0 encrypted, pairing_lvl 0x0 status 0x25
conn idx: 0, peer version: 0xb, subversion: 0xc, comp id 0xc2b
conn idx: 0, peer feature: 0x0000000ff70179ff
```

1.14.17. ble_cancel_conn

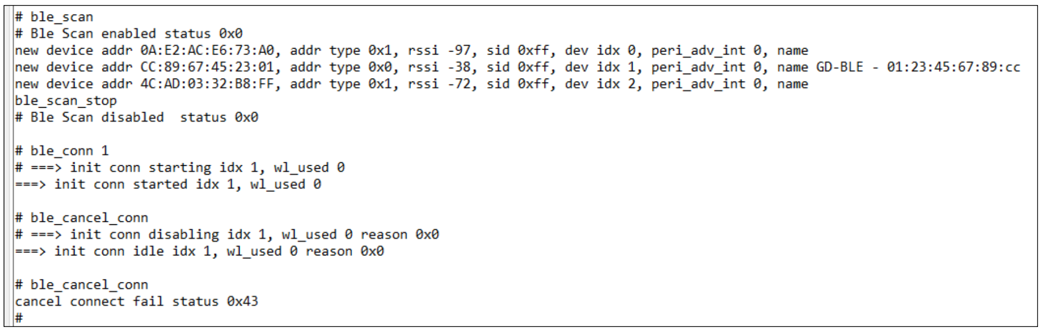
This command has no option.

This command can only be used in msdk_ffd configuration.

This command is used to cancel an unestablished connection after the ble_conn command is executed. If the connection is successfully established and needs to be disconnected, execute the ble_disconn command.

As shown in ***Figure 1-41. ble_cancel_conn command***, a prompt will be printed after ble_cancel_conn is executed. When init conn enters the idle state, it means that the execution is successful.

Figure 1-41. ble_cancel_conn command



```
# ble_scan
# Ble Scan enabled status 0x0
new device addr 0A:E2:AC:E6:73:A0, addr type 0x1, rssi -97, sid 0xff, dev idx 0, peri_adv_int 0, name
new device addr CC:89:67:45:23:01, addr type 0x0, rssi -38, sid 0xff, dev idx 1, peri_adv_int 0, name GD-BLE - 01:23:45:67:89:cc
new device addr 4C:AD:03:32:B8:FF, addr type 0x1, rssi -72, sid 0xff, dev idx 2, peri_adv_int 0, name
ble_scan_stop
# Ble Scan disabled  status 0x0

# ble_conn 1
# ===> init conn starting idx 1, wl_used 0
===> init conn started idx 1, wl_used 0

# ble_cancel_conn
# ===> init conn disabling idx 1, wl_used 0 reason 0x0
===> init conn idle idx 1, wl_used 0 reason 0x0

# ble_cancel_conn
cancel connect fail status 0x43
#
```

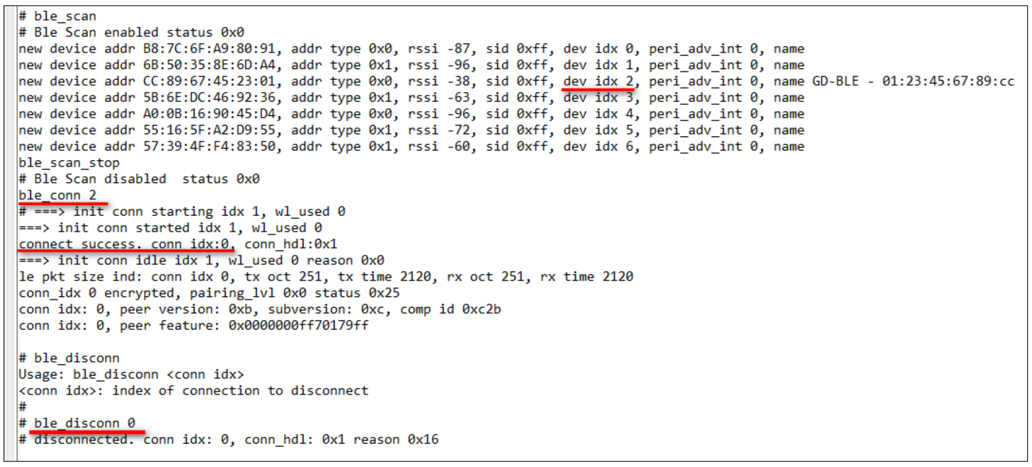
1.14.18. ble_disconn

- Usage: ble_disconn <conn idx>
- When the device establishes a connection successfully, conn idx will be printed. For example, it can be obtained from the ble_conn log.

This command is used by the device to disconnect the established connection.

As shown in ***Figure 1-42. ble_disconn command***, a prompt will be printed after ble_disconn is executed.

Figure 1-42. ble_disconn command



```
# ble_scan
# Ble Scan enabled status 0x0
new device addr B8:7C:6F:A9:80:91, addr type 0x0, rssi -87, sid 0xff, dev idx 0, peri_adv_int 0, name
new device addr 6B:50:35:8E:6D:A4, addr type 0x1, rssi -96, sid 0xff, dev idx 1, peri_adv_int 0, name
new device addr CC:89:67:45:23:01, addr type 0x0, rssi -38, sid 0xff, dev idx 2, peri_adv_int 0, name GD-BLE - 01:23:45:67:89:cc
new device addr 5B:6E:DC:46:92:36, addr type 0x1, rssi -63, sid 0xff, dev idx 3, peri_adv_int 0, name
new device addr A0:0B:16:90:45:D4, addr type 0x0, rssi -96, sid 0xff, dev idx 4, peri_adv_int 0, name
new device addr 55:16:5F:A2:D9:55, addr type 0x1, rssi -72, sid 0xff, dev idx 5, peri_adv_int 0, name
new device addr 57:39:4F:F4:83:50, addr type 0x1, rssi -60, sid 0xff, dev idx 6, peri_adv_int 0, name
ble_scan_stop
# Ble Scan disabled  status 0x0
ble_conn 2
# ===> init conn starting idx 1, wl_used 0
===> init conn started idx 1, wl_used 0
connect success. conn idx:0, conn_hdl:0x1
===> init conn idle idx 1, wl_used 0 reason 0x0
le pkt size ind: conn idx 0, tx oct 251, tx time 2120, rx oct 251, rx time 2120
conn_idx 0 encrypted, pairing_lvl 0x0 status 0x25
conn idx: 0, peer version: 0xb, subversion: 0xc, comp id 0xc2b
conn idx: 0, peer feature: 0x0000000ff70179ff

# ble_disconn
Usage: ble_disconn <conn idx>
<conn idx>: index of connection to disconnect
#
# ble_disconn 0
# disconnected. conn idx: 0, conn_hdl: 0x1 reason 0x16
```

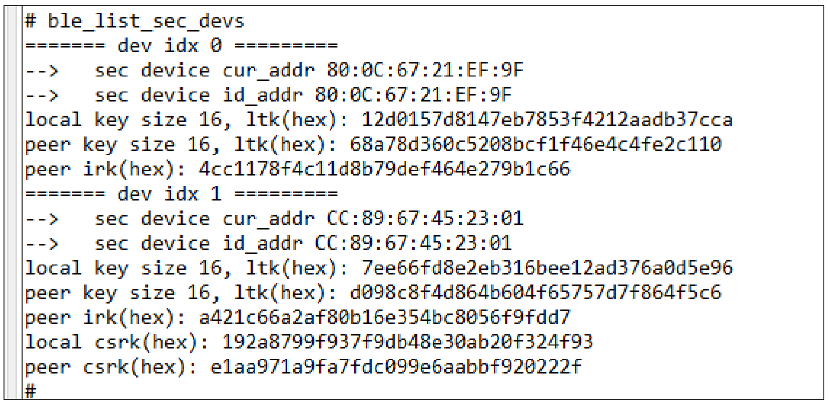
1.14.19. ble_list_sec_devs

This command has no option.

This command is used to query the bonded device information stored in flash and the peer device information currently connected, including dev idx, id_addr, LTK, IRK, etc.

As shown in ***Figure 1-43. ble_list_sec_devs command***, a prompt will be printed after ble_list_sec_devs is executed.

Figure 1-43. ble_list_sec_devs command



```
# ble_list_sec_devs
======= dev idx 0 =========
-->   sec device cur_addr 80:0C:67:21:EF:9F
-->   sec device id_addr 80:0C:67:21:EF:9F
local key size 16, ltk(hex): 12d0157d8147eb7853f4212aadb37cca
peer key size 16, ltk(hex): 68a78d360c5208bcf1f46e4c4fe2c110
peer irk(hex): 4cc1178f4c11d8b79def464e279b1c66
======= dev idx 1 =========
-->   sec device cur_addr CC:89:67:45:23:01
-->   sec device id_addr CC:89:67:45:23:01
local key size 16, ltk(hex): 7ee66fd8e2eb316bee12ad376a0d5e96
peer key size 16, ltk(hex): d098c8f4d864b604f65757d7f864f5c6
peer irk(hex): a421c66a2af80b16e354bc8056f9fdd7
local csrk(hex): 192a8799f937f9db48e30ab20f324f93
peer csrk(hex): e1aa971a9fa7fdc099e6aabbf920222f
#
```

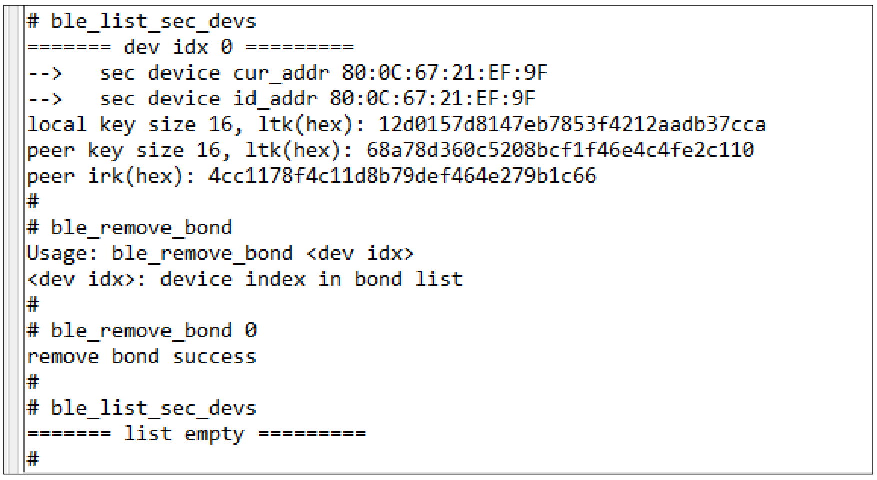
1.14.20. ble_remove_bond

- Usage: ble_remove_bond <dev idx>
- dev idx needs to be obtained from the ble_list_sec_devs command.

This command is used to remove the bond information of the device. If the device is in the connected state, the connection will be disconnected, and then the bond information will be removed. The corresponding content in flash will also be removed.

As shown in ***Figure 1-44. ble_remove_bond command***, a prompt will be printed after ble_remove_bond is executed.

Figure 1-44. ble_remove_bond command



```
# ble_list_sec_devs
======= dev idx 0 =========
-->   sec device cur_addr 80:0C:67:21:EF:9F
-->   sec device id_addr 80:0C:67:21:EF:9F
local key size 16, ltk(hex): 12d0157d8147eb7853f4212aadb37cca
peer key size 16, ltk(hex): 68a78d360c5208bcf1f46e4c4fe2c110
peer irk(hex): 4cc1178f4c11d8b79def464e279b1c66
#
# ble_remove_bond
Usage: ble_remove_bond <dev idx>
<dev idx>: device index in bond list
#
# ble_remove_bond 0
remove bond success
#
# ble_list_sec_devs
======= list empty =========
#
```

1.14.21. ble_set_auth

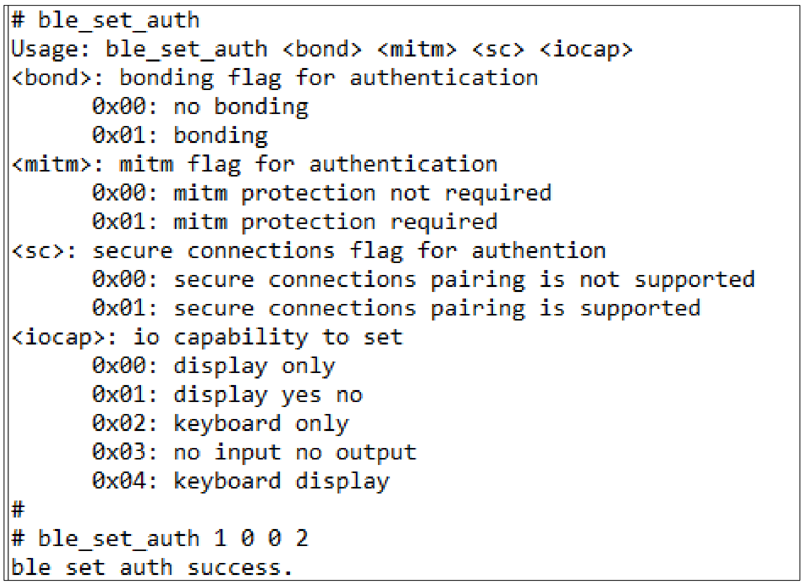
- Usage: ble_set_auth <bond> <mitm> <sc> <iocap>

This command is used to configure device security strategies: whether to save pairing information after pairing, whether to support man-in-the-middle attack protection, whether to support secure connection and IO capabilities, etc.

If bond flag is configured, the LTK, IRK, CSRK, and other information of the peer device will be saved to flash after the device is paired successfully. The configuration of mitm flag means that man-in-the-middle attack protection is supported. If the peer device also supports it, different pairing methods can be selected according to IO capabilities. The configuration of sc flag means that the device supports secure connection. If the peer device also supports it, a long-term key can be generated through the ECDH key exchange algorithm. The configuration of iocap allows the selection of IO capacities to be used during pairing, which support display only, display yes no, keyboard only, no input no output, keyboard display, etc.

As shown in ***Figure 1-45. ble set auth command***, a prompt will be printed after ble_set_auth is executed.

Figure 1-45. ble_set_auth command



```
# ble_set_auth
Usage: ble_set_auth <bond> <mitm> <sc> <iocap>
<bond>: bonding flag for authentication
      0x00: no bonding
      0x01: bonding
<mitm>: mitm flag for authentication
      0x00: mitm protection not required
      0x01: mitm protection required
<sc>: secure connections flag for authention
      0x00: secure connections pairing is not supported
      0x01: secure connections pairing is supported
<iocap>: io capability to set
      0x00: display only
      0x01: display yes no
      0x02: keyboard only
      0x03: no input no output
      0x04: keyboard display
#
# ble_set_auth 1 0 0 2
ble set auth success.
```

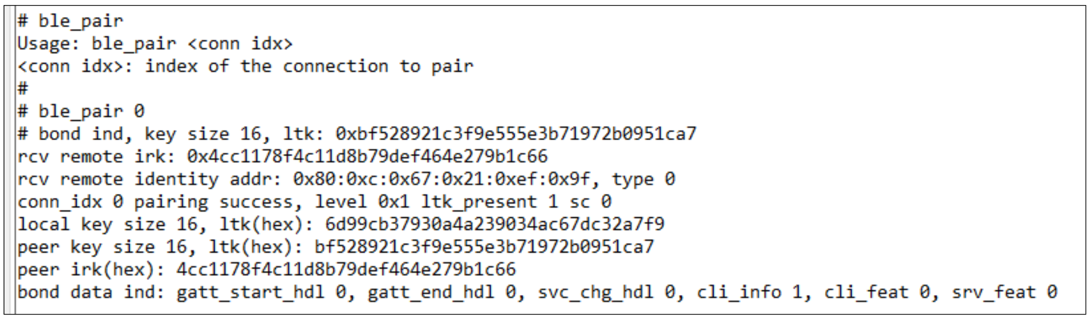
1.14.22. ble_pair

- Usage: ble_pair <conn idx>
- When the device establishes a connection successfully, conn idx will be printed. For example, it can be obtained from the ble_conn log.

This command is used to start pairing with the specified device connected. The pairing operation is used to create a key that can be used to encrypt the link.

As shown in ***Figure 1-46. ble pair command***, a prompt will be printed after ble_pair is executed.

Figure 1-46. ble_pair command



```
# ble_pair
Usage: ble_pair <conn idx>
<conn idx>: index of the connection to pair
#
# ble_pair 0
# bond ind, key size 16, ltk: 0xbf528921c3f9e555e3b71972b0951ca7
rcv remote irk: 0x4cc1178f4c11d8b79def464e279b1c66
rcv remote identity addr: 0x80:0xc:0x67:0x21:0xef:0x9f, type 0
conn_idx 0 pairing success, level 0x1 ltk_present 1 sc 0
local key size 16, ltk(hex): 6d99cb37930a4a239034ac67dc32a7f9
peer key size 16, ltk(hex): bf528921c3f9e555e3b71972b0951ca7
peer irk(hex): 4cc1178f4c11d8b79def464e279b1c66
bond data ind: gatt_start_hdl 0, gatt_end_hdl 0, svc_chg_hdl 0, cli_info 1, cli_feat 0, srv_feat 0
```

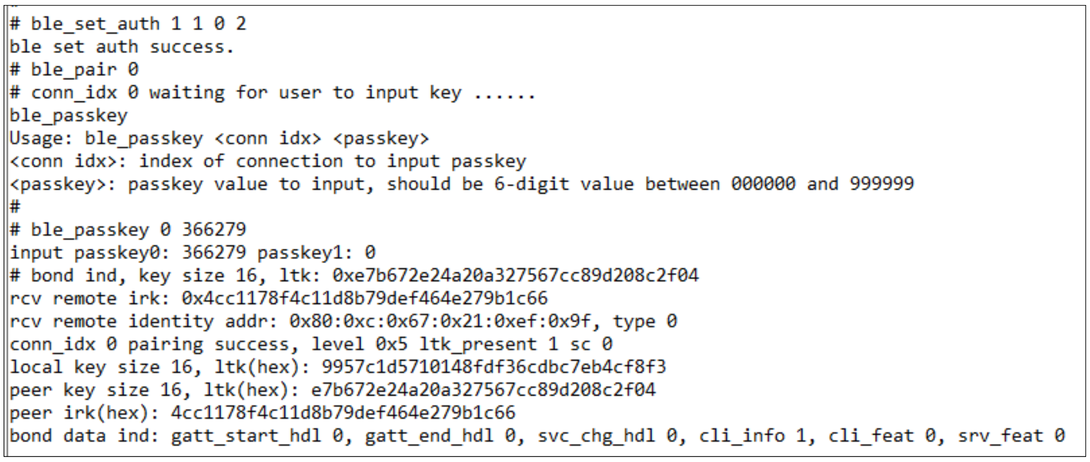
1.14.23. ble_passkey

- Usage: ble_passkey <conn idx> <passkey>
- When the device establishes a connection successfully, conn idx will be printed. For example, it can be obtained from the ble_conn log.

This command is used to input the passkey (6-digit number) during pairing procedure. The passkey must be the same with the one in the peer device to make sure pairing is successful.

As shown in ***Figure 1-47. ble_passkey command***, a prompt will be printed after ble_passkey is executed.

Figure 1-47. ble_passkey command



```
# ble_set_auth 1 1 0 2
ble set auth success.
# ble_pair 0
# conn_idx 0 waiting for user to input key ......
ble_passkey
Usage: ble_passkey <conn idx> <passkey>
<conn idx>: index of connection to input passkey
<passkey>: passkey value to input, should be 6-digit value between 000000 and 999999
#
# ble_passkey 0 366279
input passkey0: 366279 passkey1: 0
# bond ind, key size 16, ltk: 0xe7b672e24a20a327567cc89d208c2f04
rcv remote irk: 0x4cc1178f4c11d8b79def464e279b1c66
rcv remote identity addr: 0x80:0xc:0x67:0x21:0xef:0x9f, type 0
conn_idx 0 pairing success, level 0x5 ltk_present 1 sc 0
local key size 16, ltk(hex): 9957c1d5710148fdf36cdbc7eb4cf8f3
peer key size 16, ltk(hex): e7b672e24a20a327567cc89d208c2f04
peer irk(hex): 4cc1178f4c11d8b79def464e279b1c66
bond data ind: gatt_start_hdl 0, gatt_end_hdl 0, svc_chg_hdl 0, cli_info 1, cli_feat 0, srv_feat 0
```

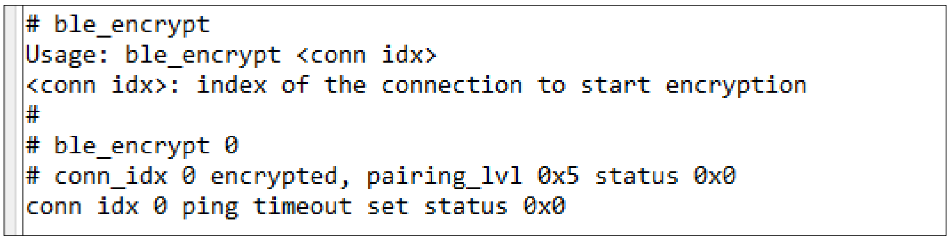
1.14.24. ble_encrypt

- Usage: ble_encrypt <conn idx>
- When the device establishes a connection successfully, conn idx will be printed. For example, it can be obtained from the ble_conn log.

The command is used to encrypt the specified connection. If the link is already encrypted, the encryption key will be refreshed.

As shown in ***Figure 1-48. ble_encrypt command***, a prompt will be printed after ble_encrypt is executed.

Figure 1-48. ble_encrypt command



```
# ble_encrypt
Usage: ble_encrypt <conn idx>
<conn idx>: index of the connection to start encryption
#
# ble_encrypt 0
# conn_idx 0 encrypted, pairing_lvl 0x5 status 0x0
conn idx 0 ping timeout set status 0x0
```

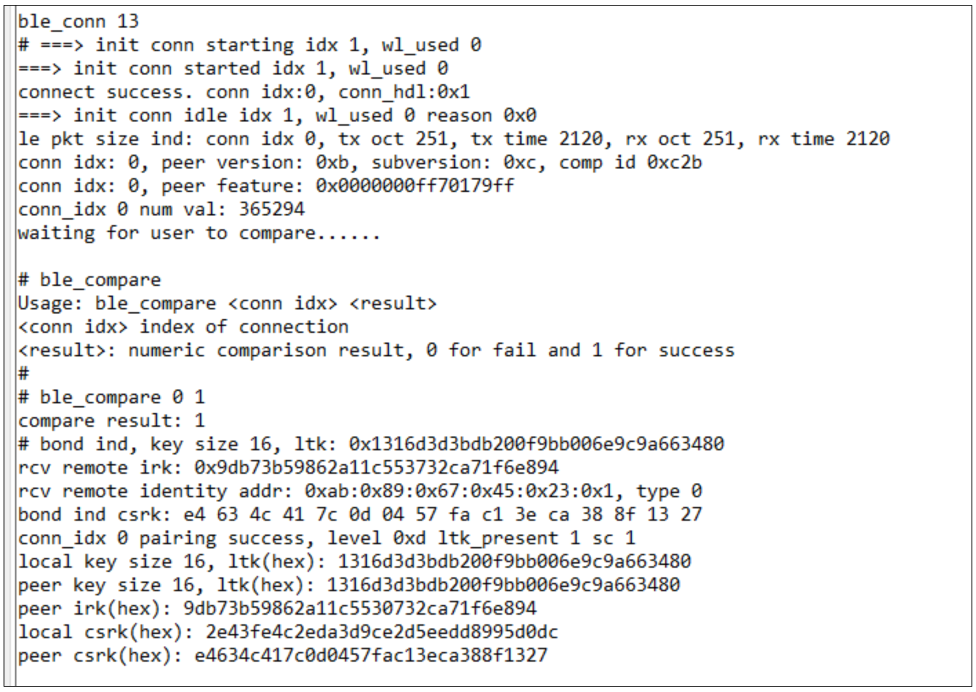
1.14.25. ble_compare

- Usage: ble_compare <conn idx> <result>
- When the device establishes a connection successfully, conn idx will be printed. For example, it can be obtained from the ble_conn log.

This command is used to determine whether the temporary keys generated at both sides are the same during the pairing with the specified device connected.

As shown in ***Figure 1-49. ble_compare command***, a prompt will be printed after ble_compare is executed.

Figure 1-49. ble_compare command



```
ble_conn 13
# ===> init conn starting idx 1, wl_used 0
===> init conn started idx 1, wl_used 0
connect success. conn idx:0, conn_hdl:0x1
===> init conn idle idx 1, wl_used 0 reason 0x0
le pkt size ind: conn idx 0, tx oct 251, tx time 2120, rx oct 251, rx time 2120
conn idx: 0, peer version: 0xb, subversion: 0xc, comp id 0xc2b
conn idx: 0, peer feature: 0x0000000ff70179ff
conn_idx 0 num val: 365294
waiting for user to compare......

# ble_compare
Usage: ble_compare <conn idx> <result>
<conn idx> index of connection
<result>: numeric comparison result, 0 for fail and 1 for success
#
# ble_compare 0 1
compare result: 1
# bond ind, key size 16, ltk: 0x1316d3d3bdb200f9bb006e9c9a663480
rcv remote irk: 0x9db73b59862a11c553732ca71f6e894
rcv remote identity addr: 0xab:0x89:0x67:0x45:0x23:0x1, type 0
bond ind csrk: e4 63 4c 41 7c 0d 04 57 fa c1 3e ca 38 8f 13 27
conn_idx 0 pairing success, level 0xd ltk_present 1 sc 1
local key size 16, ltk(hex): 1316d3d3bdb200f9bb006e9c9a663480
peer key size 16, ltk(hex): 1316d3d3bdb200f9bb006e9c9a663480
peer irk(hex): 9db73b59862a11c5530732ca71f6e894
local csrk(hex): 2e43fe4c2eda3d9ce2d5eedd8995d0dc
peer csrk(hex): e4634c417c0d0457fac13eca388f1327
```

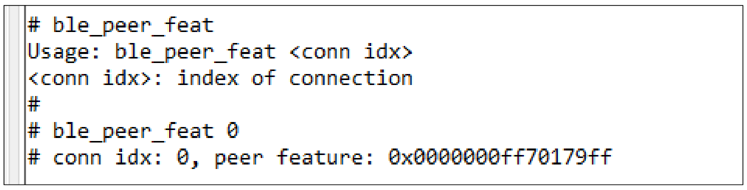
1.14.26. ble_peer_feat

- Usage: ble_peer_feat <conn idx>
- When the device establishes a connection successfully, conn idx will be printed. For example, it can be obtained from the ble_conn log.

This command is used to get the supported features of the specified device connected. For the meaning of each bit, refer to the FEATURE SUPPORT in the BLE Core Spec.

As shown in ***Figure 1-50. ble_peer_feat command***, a prompt will be printed after ble_peer_feat is executed.

Figure 1-50. ble_peer_feat command



```
# ble_peer_feat
Usage: ble_peer_feat <conn idx>
<conn idx>: index of connection
#
# ble_peer_feat 0
# conn idx: 0, peer feature: 0x0000000ff70179ff
```

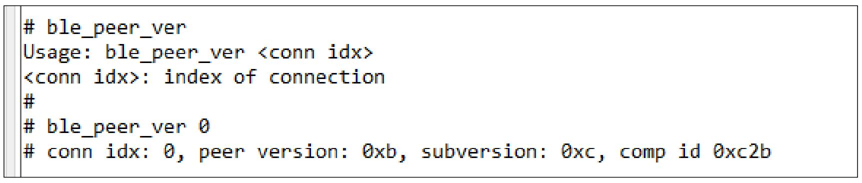
1.14.27. ble_peer_ver

- Usage: ble_peer_ver <conn idx>
- When the device establishes a connection successfully, conn idx will be printed. For example, it can be obtained from the ble_conn log.

This command is used to get the version information of the specified device connected, including Bluetooth version information (0xb: BT5.2), subversion information, and company identifier (GigaDevice: 0x0C2B).

As shown in ***Figure 1-51. ble_peer_ver command***, a prompt will be printed after ble_peer_ver is executed.

Figure 1-51. ble_peer_ver command



```
# ble_peer_ver
Usage: ble_peer_ver <conn idx>
<conn idx>: index of connection
#
# ble_peer_ver 0
# conn idx: 0, peer version: 0xb, subversion: 0xc, comp id 0xc2b
```

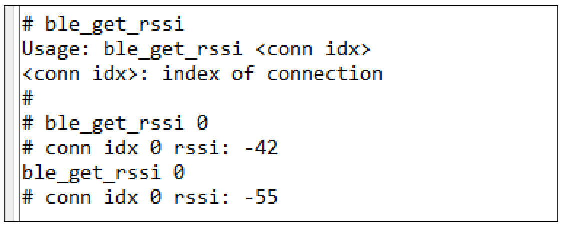
1.14.28. ble_get_rssi

- Usage: ble_get_rssi <conn idx>
- When the device establishes a connection successfully, conn idx will be printed. For example, it can be obtained from the ble_conn log.

This command is used to get RSSI information of the latest packet received from the specified device connected.

As shown in ***Figure 1-52. ble_get_rssi command***, a prompt will be printed after ble_get_rssi is executed.

Figure 1-52. ble_get_rssi command



```
# ble_get_rssi
Usage: ble_get_rssi <conn idx>
<conn idx>: index of connection
#
# ble_get_rssi 0
# conn idx 0 rssi: -42
ble_get_rssi 0
# conn idx 0 rssi: -55
```

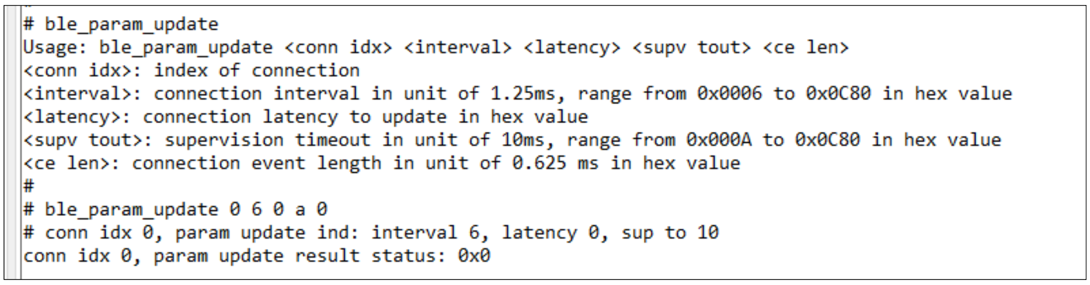
1.14.29. ble_param_update

- Usage: ble_param_update <conn idx> <interval> <latency> <supv tout> <ce len>
- When the device establishes a connection successfully, conn idx will be printed. For example, it can be obtained from the ble_conn log.

The command is used to update the parameters of the specified connection, such as connection interval, latency and supervision timeout.

As shown in ***Figure 1-53. ble_param_update command***, a prompt will be printed after ble_param_update is executed.

Figure 1-53. ble_param_update command



```
# ble_param_update
Usage: ble_param_update <conn idx> <interval> <latency> <supv tout> <ce len>
<conn idx>: index of connection
<interval>: connection interval in unit of 1.25ms, range from 0x0006 to 0x0C80 in hex value
<latency>: connection latency to update in hex value
<supv tout>: supervision timeout in unit of 10ms, range from 0x000A to 0x0C80 in hex value
<ce len>: connection event length in unit of 0.625 ms in hex value
#
# ble_param_update 0 6 0 a 0
# conn idx 0, param update ind: interval 6, latency 0, sup to 10
conn idx 0, param update result status: 0x0
```

1.14.30. ble_set_phy

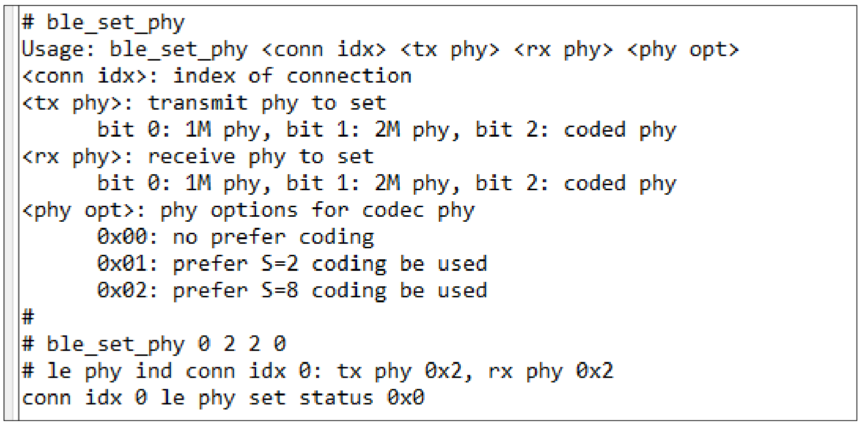
- Usage: ble_set_phy <conn idx> <tx phy> <rx phy> <phy opt>
- When the device establishes a connection successfully, conn idx will be printed. For example, it can be obtained from the ble_conn log.

This command can only be used in msdk_ffd configuration.

This command is used to set TX/RX PHY for the specified connection. If TX/RX PHY is set to 0, it means that all PHYs are supported; otherwise, the meaning of each bit is shown in the figure below.

As shown in ***Figure 1-54. ble_set_phy command***, a prompt will be printed after ble_set_phy is executed.

Figure 1-54. ble_set_phy command



```
# ble_set_phy
Usage: ble_set_phy <conn idx> <tx phy> <rx phy> <phy opt>
<conn idx>: index of connection
<tx phy>: transmit phy to set
      bit 0: 1M phy, bit 1: 2M phy, bit 2: coded phy
<rx phy>: receive phy to set
      bit 0: 1M phy, bit 1: 2M phy, bit 2: coded phy
<phy opt>: phy options for codec phy
      0x00: no prefer coding
      0x01: prefer S=2 coding be used
      0x02: prefer S=8 coding be used
#
# ble_set_phy 0 2 2 0
# le phy ind conn idx 0: tx phy 0x2, rx phy 0x2
conn idx 0 le phy set status 0x0
```

1.14.31. ble_get_phy

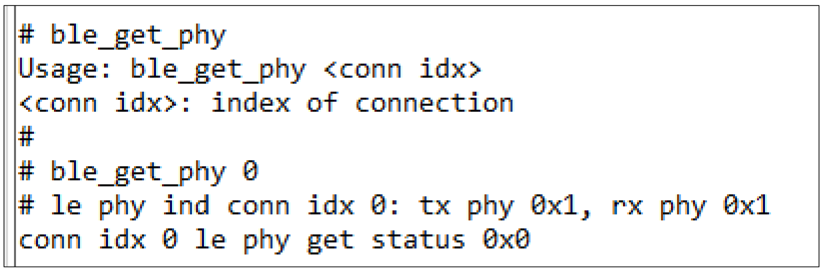
- Usage: ble_get_phy <conn idx>
- When the device establishes a connection successfully, conn idx will be printed. For example, it can be obtained from the ble_conn log.

This command can only be used in msdk_ffd configuration.

This command is used to get the current TX/RX PHY of the specified connection.

As shown in ***Figure 1-55. ble_get_phy command***, a prompt will be printed after ble_get_phy is executed, the result value meaning is 0x1: 1M; 0x2: 2M; 0x3: coded.

Figure 1-55. ble_get_phy command



```
# ble_get_phy
Usage: ble_get_phy <conn idx>
<conn idx>: index of connection
#
# ble_get_phy 0
# le phy ind conn idx 0: tx phy 0x1, rx phy 0x1
conn idx 0 le phy get status 0x0
```

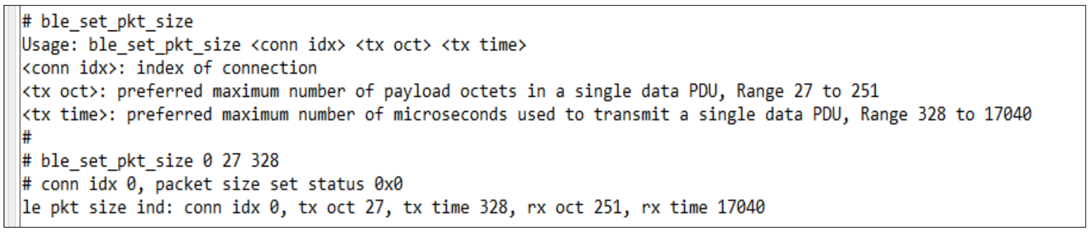
1.14.32. ble_set_pkt_size

- Usage: ble_set_pkt_size <conn idx> <tx oct> <tx time>
- When the device establishes a connection successfully, conn idx will be printed. For example, it can be obtained from the ble_conn log.

This command is used to set the maximum number of bytes and time for sending a PDU on the specified connection.

As shown in *Figure 1-56. ble set pkt size command*, a prompt will be printed after ble_set_pkt_size is executed.

Figure 1-56. ble_set_pkt_size command



```
# ble_set_pkt_size
Usage: ble_set_pkt_size <conn idx> <tx oct> <tx time>
<conn idx>: index of connection
<tx oct>: preferred maximum number of payload octets in a single data PDU, Range 27 to 251
<tx time>: preferred maximum number of microseconds used to transmit a single data PDU, Range 328 to 17040
#
# ble_set_pkt_size 0 27 328
# conn idx 0, packet size set status 0x0
le pkt size ind: conn idx 0, tx oct 27, tx time 328, rx oct 251, rx time 17040
```

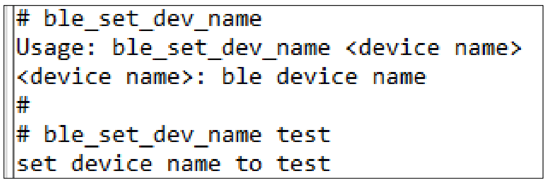
1.14.33. ble_set_dev_name

- Usage: ble_set_dev_name <device name>
- <device name>: ble device name

This command is used to set BLE local device name. If there are advertisings ongoing, advertising data will also be updated.

As show in *Figure 1-57. ble set dev name command*, a prompt will be printed after ble_set_dev_name is executed.

Figure 1-57. ble_set_dev_name command



```
# ble_set_dev_name
Usage: ble_set_dev_name <device name>
<device name>: ble device name
#
# ble_set_dev_name test
set device name to test
```

2. Revision history

Table 2-1. Revision history

Revision No.	Description	Date
1.0	Initial release	Nov.24.2023
1.1	add new command, nvds, ps_stats, wifi_setup_twt, wifi_teardown_twt, wifi_roaming, wifi_wireless_mode.	Feb.28.2024
1.2	add new command group-WiFi APP, including some WiFi demo command; add ble command: ble_set_dev_name.	Jul.12.2024

Important Notice