# Connection Manager UseCases

## Station Mode Wi-Fi Connection

### Description

This use case is to demonstrate the station mode configuration and connecting to the Wi-Fi Access Point.

### Prerequisites

Access Point configured with WPA/WPA2 personal security.

### Command Description

|  |  |
| --- | --- |
| **Command** | **Description** |
| *./conmgr scan* | Scan |
| *./conmgr connect <SSID><AP PWD>* | Connect |
| *./conmgr ip* | Get IP address |
| *./conmgr status* | Get WLAN status |
| *./conmgr disconnect* | Disconnect |

Table 47: Station Mode Wi-Fi Connection - commands

### Procedure

Execute the following operations on the Talaria TWO:

**Step 1**: Scan the network.

**Step 2**: Connect to the desired network by providing SSID and passphrase.

**Step 3**: Get the IP address of the Talaria TWO module.

**Step 4**: Get the WLAN status of the Talaria TWO module.

**Step 5**: Disconnect from the connected network.

### Console Logs

1. ./conmgr scan

Text

Description automatically generated

Figure 44: ./conmgr scan – output

1. ./conmgr connect innotest 123456789

A computer screen with white text

Description automatically generated

Figure 45: ./conmgr connect - output

1. ./conmgr ip

A screenshot of a computer

Description automatically generated with medium confidence

Figure 46: ./conmgr ip – output

1. ./conmgr status

Text

Description automatically generated

Figure 47: ./conmgr status – output

1. ./conmgr disconnect

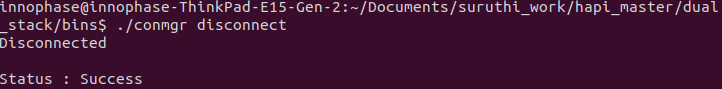


Figure 48: ./conmgr disconnect – output

## Keepalive and Host Wake Up Mechanism

### Description

This use case demonstrates keep alive and host wake up mechanism on Dual-Stack solution.

### Prerequisites

1. Access Point configured with WPA/WPA2 personal security.
2. Netcat server on Linux host machine.

### Command Description

|  |  |
| --- | --- |
| **Command** | **Description** |
| *./conmgr connect <SSID><AP PWD>* | Connect |
| *./conmgr kaconfig <ip address of server> <srv\_port> <interval> <heartbeat msg> <wakeup word> <timeout>* | Keepalive configuration |
| *./conmgr kaconfigget* | Get keepalive configuration |
| *./conmgr kastart* | Start sending keepalive to server |
| *./conmgr <wakeupconfig> <wakeup pin> <0/1> <0/1>* | Configure wake up pin |
| *./conmgr wakeupreasoncls* | Clear wakeup reason |
| *./conmgr wakeupcfgget* | Get wakeup configuration status |
| *./conmgr shutdown* | Power off the host |

Table 48: Keepalive and host wake up mechanism – Commands

### Procedure

Execute the following operations:

**Step 1**: Start the netcat server on the Linux host machine. This host should be connected to same network to which Talaria TWO will be connected .

|  |
| --- |
| nc -l -p 8888 |

**Step 2**: On Talaria TWO, connect to the desired network by providing SSID and passphrase.

**Step 3**: Configure Keepalive.

**Step 4**: Read the Keepalive configuration.

**Step 5**: start sending Keepalive message keepalivemsg to the server.

**Step 6**: Configure GPIO20 for host wake up.

**Step 7**: Clear wakeup reason.

**Step 8**: Get wakeup configuration status.

**Step 9**: Power OFF the Host.

**Step 10**: From netcat server, send the wakeup word hello. Talaria TWO receives and wakes up the host.

**Step 11**: Check the wakeup reason.

**Step 12**: Stop Keepalive.

### Console Logs

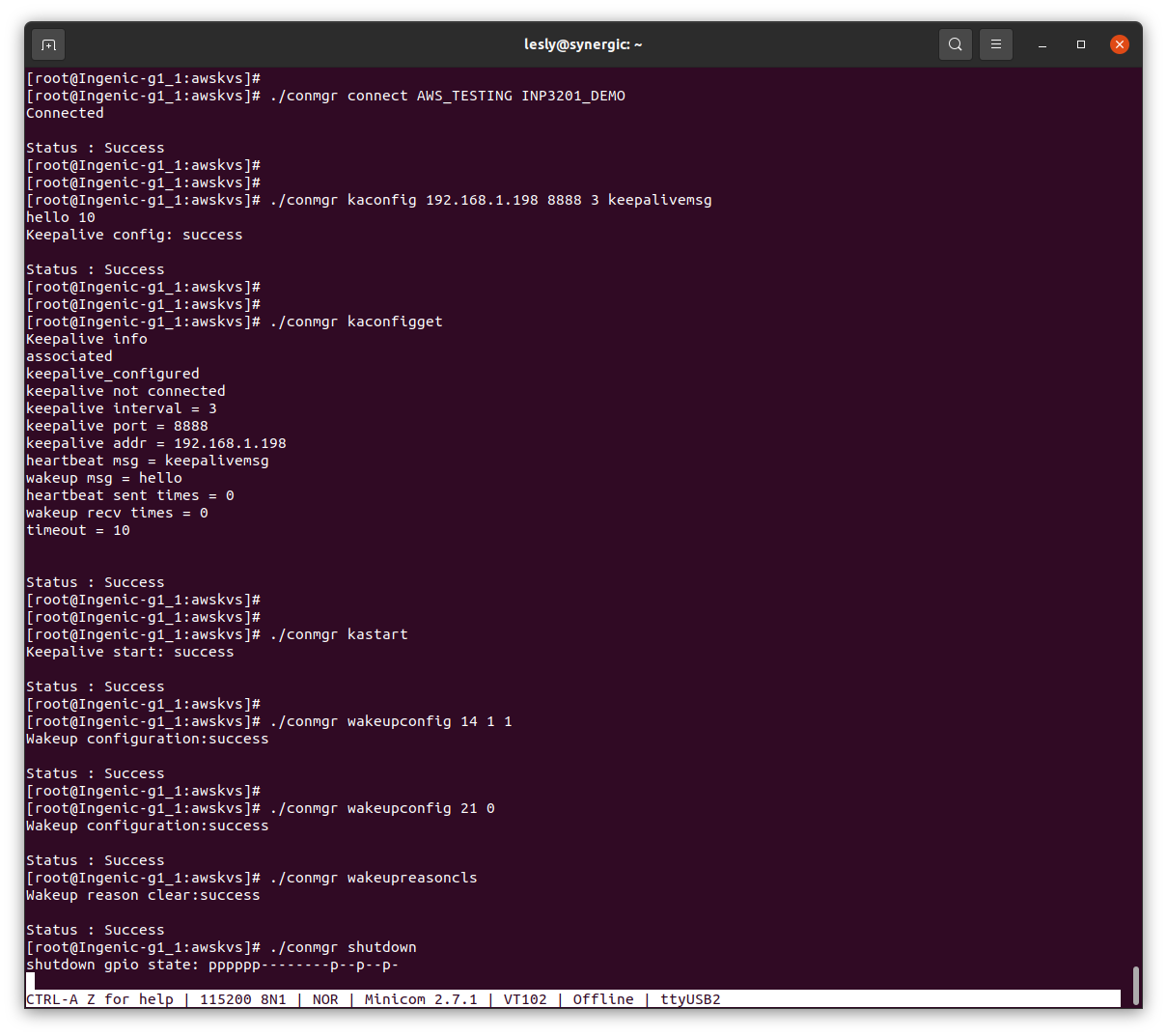


Figure 49: Keepalive config and host shutdown with MCU– output

A screenshot of a computer

Description automatically generated

Figure 50: Keepalive config and host shutdown without MCU– output

A screenshot of a computer

Description automatically generated

Figure 51: Host wakeup– output

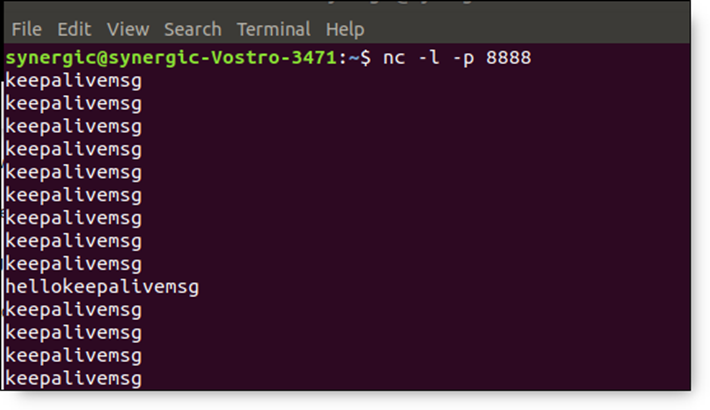
 

Figure 52: netcat server – output

## Autoscan Operation

### Description

This use case demonstrates enabling/disabling auto scan operation for finding the specific SSID and SSID that are not in the vicinity of Talaria TWO.

### Prerequisites

Access Point configured with WPA/WPA2 personal security.

### Command Description

|  |  |
| --- | --- |
| **Command** | **Description** |
| *./conmgr autoscanenable <interval> <ssid>* | Enable auto scan |
| *./conmgr autoscancfgget* | Get autoscan configuration status |
| *./conmgr autoscandisable* | Disable auto scan |

Table 49: autoscanenable and autoscandisable – commands

### Procedure

Execute the following operations on the Talaria TWO:

**Step 1**: Enable auto scan with SSID information of an AP.

**Step 2**: Enable auto scan with SSID information of an AP which is not in the vicinity.

**Step 3**: Get autoscan configuration status with the appropriate interval and SSID.

**Step 4**: Disable the autoscan.

### Console Logs

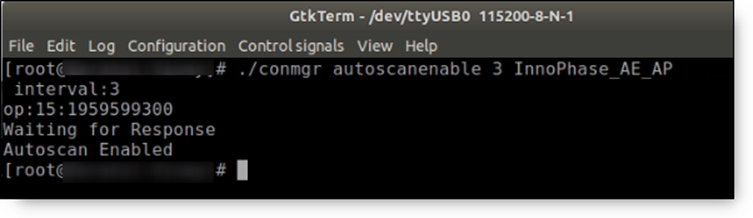


Figure 53: autoscanenable for specific SSID - host console output

Talaria TWO console log:

|  |
| --- |
| event:9in dualstack\_autoscan\_process: entry:121864:0:  scan start  pssid:InnoPhase\_AE\_AP:InnoPhase\_AE\_APFound 1 nets:  ssid:InnoPhase\_AE\_AP |

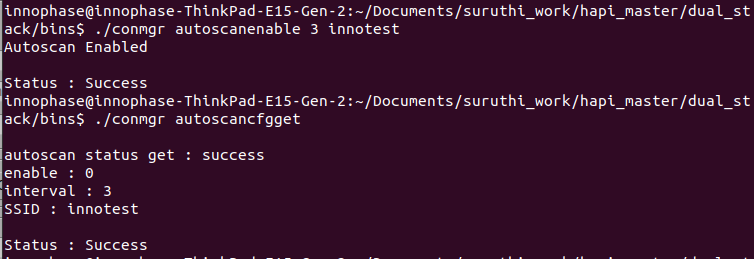


Figure 54: autoscanenable and get autoscan status – output

Text

Description automatically generated

Figure 55: autoscanenable for SSIDs not in the vicinity of station – host console output

Talaria TWO console log:

|  |
| --- |
| scan start  pssid:InnoPhase\_AE:InnoPhase\_AEFound 0 nets:  in dualstack\_autoscan\_process: exit  Dtx rate:{837(561),0(0),0(0),0(0),0(0),0(0),0(0),0(0) 59(26),0(0),0(0),0(0),4(2),1(0),17(7),0(0) 0(0),0(0),7(5),7(1),119(55),0(0),0(0),0(0)}  ampdu:{0,0,0,0,0,0,0,0} sf:0.0 ba:0%(0%) to:8  Drx rate:{286,0,0,0,0,0,0,0 0,0,0,0,0,0,0,0 0,0,0,0,4,7,23,13}  mcs:{0(0,0),0(0,0),0(0,0),0(0,0),0(0,0),0(0,0),0(0,0),0(0,0)} no\_mem:0/0 rx\_idle:86755  Drts: 81(54)  Dlwip: 660(0/0/0)  Dmem: heap:122336 page\_faults:3825  event:9in dualstack\_autoscan\_process: entry:122296:0:  scan start  pssid:InnoPhase\_AE:InnoPhase\_AEFound 0 nets:  in dualstack\_autoscan\_process: exit |

## Enable Connect/Disconnect Callback

### Description

This use case demonstrates invocation of callback function during connection and disconnection process with an Access Point.

### Prerequisites

Access Point configured with WPA/WPA2 personal security.

### Command Description

|  |  |
| --- | --- |
| **Command** | **Description** |
| *./conmgr connect <SSID><AP PWD>* | Connect |
| *./conmgr cbenable* | Callback enable |
| *./conmgr disconnect* | Disconnect |
| *./conmgr connect <SSID><AP PWD>* | Connect |

Table 50: Connect/disconnect callback – commands

### Procedure

Execute the following operations on the Talaria TWO:

**Step 1**: Connect to the desired network by providing SSID and passphrase.

**Step 2:** Enable callback.

**Step 3**: Disconnect from the network.

**Step 4**: Connect to the network.

### Console Logs

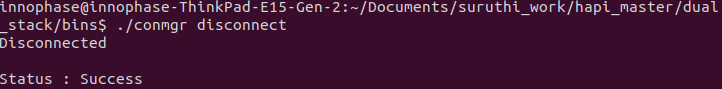


Figure 56: Disconnect - output

A computer screen with white text

Description automatically generated

Figure 57: Connect – output

Talaria TWO console log:

|  |
| --- |
| [1162.370,516]  dualstack\_wcm\_notify:201:1[1162.371,762] DISCONNECTED  event:24[1162.388,153]  dualstack\_wcm\_notify:203:1  event:22  Disconnection callback  [1174.265,374] CONNECT:98:da:c4:73:b7:76 Channel:2 rssi:-40 dBm  [1174.340,947] MYIP 192.168.0.228  [1174.341,114] IPv6 [fe80::e069:3aff:fe00:46e]-link  event:16  owcb:1:InnoPhase\_AE\_AP::1  linkup:1  Connection callback |

## Firmware Upgrade Over Serial

### Description

This use case demonstrates firmware upgrade process from host to Talaria TWO via SPI/SDIO interface.

### Prerequisites

TFTP server on Linux machine.

### Command Description with Procedure

1. On Linux console machine, open a terminal and issue the following command to put Talaria TWO into bootloader mode.

|  |
| --- |
| ./script/boot.py --device /dev/ttyUSB2 --reset=evk42\_bl |

1. Start openocd from the SDK root folder *(sdk\_x.y\)*. This enables in-system programming on Talaria TWO.

Command:

|  |
| --- |
| openocd -s conf/ -f ftdi\_swd.cfg -f t2\_swd.cfg |

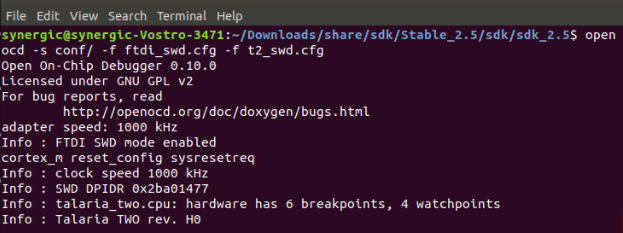
 

Figure 58: In-system programming on Talaria TWO

1. In a separate terminal, start arden.py script from SDK root folder *(sdkx.y\)* on a new terminal:

Command:

|  |
| --- |
| ./script/arden.py ./apps/gordon-jtag/bin/gordon-jtag.elf |



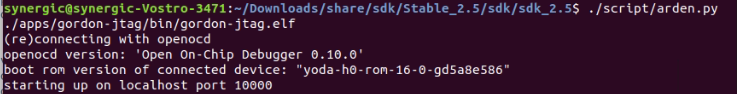


Figure 59: Start arden.py from SDK root folder

1. Run the script gdbrun.py from SDK root folder *(sdk\_x.y\)* on a new terminal. Given a host with running OpenOCD and UART relay, gdbrun.py connects to host and loads/boots an ELF image.

Command:

|  |
| --- |
| ./script/gdbrun.py ./apps/gordon-jtag/bin/gordon-jtag.elf --noconsole --nowait |

1. Flash SSBL partition table and dual\_stack\_sdio.elf/dual\_stack.elf to Talaria TWO using the following commands:
   1. Invalidate the boot image.

|  |
| --- |
| ./script/flash.py --device localhost:10000 write 0x1000 ./solutions/dual\_stack/firmware\_upgrade\_images/empty.img |





Figure 60: Invalidate boot image – Terminal Output

* 1. Flash SSBL partition.

|  |
| --- |
| ./script/flash.py –device localhost:10000 from\_json ./tools/partition\_files/ssbl\_part\_table.json |



Figure 61: Flash SSBL Partition - Terminal Output

* 1. Flash root filesystem.

|  |
| --- |
| ./script/flash.py --device localhost:10000 write 0x180000 ./solutions/dual\_stack/firmware\_upgrade\_images/root\_sdio.img |



Figure 62: Flash root filesystem – Termina output

* 1. Flash SSBL.

|  |
| --- |
| ./script/flash.py –device localhost:10000 write 0x1000 ./apps/ssbl/fast\_ssbl.img |

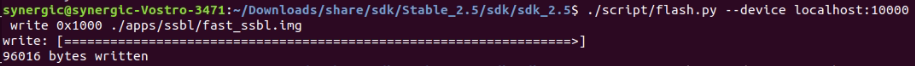


Figure 63: Flash SSBL – Terminal Output

* 1. Flash Dual-Stack ELF.

|  |
| --- |
| ./script/flash.py --device localhost:10000 write 0x20000 ./solutions/dual\_stack/bin/dual\_stack\_sdio.elf |

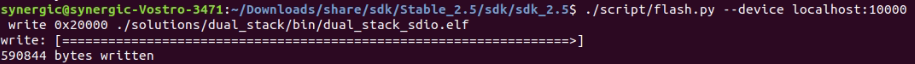


Figure 64: Flash Dual-Stack ELF - Terminal Output

1. Using TFTP or SD card, copy the dual\_stack\_sdio.elf/dual\_stack.elf from \solutions\dual\_stack\bin\ folder to the lib\modules\ folder on host using TFTP or SD card.

|  |
| --- |
| tftp -g -r <filename> <tftp server IP> |

1. Start the tunadapter in the background.

|  |
| --- |
| ./tunadapter & |

1. Start the firmware upgrade from host to Talaria TWO.

|  |
| --- |
| ./conmgr <fos> <elf\_path> <hash> <auto\_reset> |

**Note:** Turn off Talaria TWO powersave before upgrading the firmware by using the command ./conmgr powersave 0.

### Console Logs

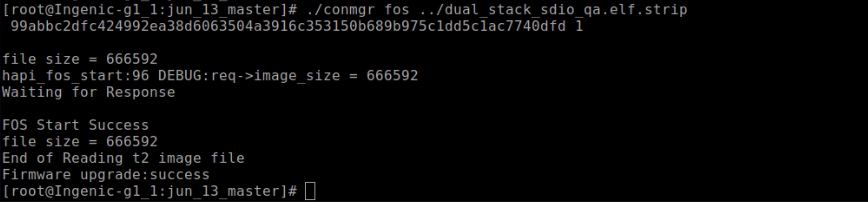


Figure 65: Firmware upgrade – output

|  |
| --- |
| [root@:Aug30\_Master]# ./tunadapter &  [root@: Aug30\_Master]# ./tunadapter: platform=<host>\_SDIO  Opening Serial device /dev/wlanSDIO0  Speed=10000000  [ 49.161440] sw\_open  hapi\_recv\_thread:943 DEBUG:recv thread entry  wakeup gpio(gpio\_63) conf success  in hapi\_wakeup\_config. dev=/sys/class/gpio/gpio63/value  wakeup\_fd = 4  hapi\_config:1526 DEBUG:hapi\_config. 14 1 0 0  hapi\_config:1586 DEBUG:before calling hio\_query  hapi\_hio\_query:352 DEBUG:Connected to T2, max packet size is 4092  hapi\_hio\_query:353 DEBUG:Firmware version: , patch version  hapi\_hio\_query:354 DEBUG:Hapi version: 1  hapi\_config:1588 DEBUG:after calling hio\_query  hapi init:start  Registering Wake up Indication Handlers  Registering Shutdown Indication Handlers  hapi init:done  dual\_stack\_status:688896:1:0  dual\_stack\_ver:e4f2ebcd:SDK\_2.6:1.0.05  port:70-40000  dual\_stack\_sockid:0:2:1  Creating tun interface  Initialising tun interface  IP: 192.168.1.131  Successfully connected to interface  net.ipv4.ip\_local\_port\_range = 70 40000  cmd:ifconfig tun0 192.168.1.131 netmask 255.255.255.0  ip set done  Adding routing table entry:echo nameserver 192.168.1.1 > /etc/resolv\_ds.conf  readlink() returned /etc/resolv\_ds.conf  Adding default route  UDP socket create success  ICMP Raw socket create success  Regsitering WCM Indication Handlers  waiting for data from tun....  ptsname:/dev/pts/0  [root@:Aug30\_Master]# ./conmgr fos ../dual\_stack\_sdio\_qa.elf.strip 99abbc2dfc424992ea38d6063504a3916c353150b689b975c1dd5c1ac7740dfd 1  file size = 666592  hapi\_fos\_start:96 DEBUG:req->image\_size = 666592  Waiting for Response  FOS Start Success  file size = 666592  End of Reading t2 image file  Firmware upgrade:success |

## Wi-Fi Provisioning over BLE

### Description

Using the device provisioning feature, SSID and passphrase can be configured onto the device using the mobile application.

For information on downloading and using the mobile application for device provisioning, refer section: *7.3 Running the Application using Android or iOS App*, of the document: Example\_using\_Provisioning.pdf (*sdk\_x.y/examples\prov\doc*).

Once the provisioning is complete, the configured SSID and passphrase is provided to the Host, which stores these parameters for further use.

### Prerequisites

Mobile application (BLE Provisioning) to provision the device.

### Command Description

|  |  |
| --- | --- |
| **Command** | **Description** |
| *./conmgr provstart* | Start provisioning |
| *./conmgr waitforprovinfo* | Wait for the provisioned info |
| *./conmgr provstop* | Stop provisioning |
| *./conmgr provstart* | Start provisioning |

Table 51: Wi-Fi provisioning over BLE – commands

### Procedure

**Step 1**: Initiate provisioning using ./conmgr provstart command

**Note**: ./conmgr waitforprovinfo should be passed immediately after ./conmgr provstart command

**Step 2**: Wait for the user to provision the device. This can be achieved using the ./conmgr waitforprovinfo command

**Step 3**: Launch the InnophaseIoT mobile application on Android/iOS mobile and connect to the device

**Step 4**: Provide the appropriate SSID and passphrase on the mobile application

**Step 5**: If the right SSID and passphrase is provided, the same will be provided to the Host and ./conmgr waitforprovinfo command is executed

**Step 6**: Use ./conmgr provstop to stop provisioning as per requirement

### Console Logs

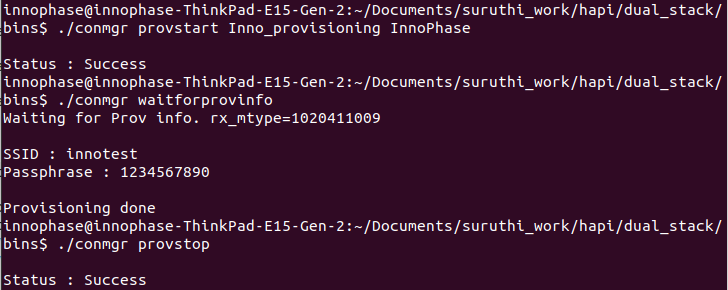


Figure 66: Device provisioning

|  |
| --- |
| Y-BOOT 208ef13 2019-07-22 12:26:54 -0500 790da1-b-7  ROM yoda-h0-rom-16-0-gd5a8e586  FLASH:PNWWWWWWAE4 DWT comparators, range 0x8000  Build  hio.transport=0 ds.pf\_method=2  $App:git-0ec71a86  SDK Ver: SDK\_2.7\_master  DualStack App  Serial-to-Wireless: Ready  Registering dual stack apis  Standard HIOs Enabled: fota fos prov  Custom HIOs Enabled:  gpio\_ctrl custom\_echo pir Registering pir apis  chip\_mon custom\_mqtt [0.207,059] mqtt\_ping\_interval=60  custom\_n/w custom\_wifi\_connect rssi\_monitor  DualStack: Ready...  dual\_stack\_param\_getaddr e0:69:3a:00:0a:5a  dual\_stack\_param\_update  store:::1  linkup:0  dual\_stack\_btinitBT HCI: Ready  dual\_stack\_host\_ready  dual\_stack\_btinit  [PROV] bt\_gap\_init  [PROV] common\_server\_create: Inno\_provisioning InnoPhase 0  [PROV] bt\_gatt\_create\_service\_128 ret = 0x0008f650  [PROV] Adding Characteristics  [PROV] bt\_gatt\_add\_service  [PROV] bt\_gap\_discoverable\_mode, ret = 0  prov\_event\_thread\_entry[13.386,045] BT connect[0]: ia:54:c5:dd:79:4d:ad aa:05:04:03:02:01:00 phy2:0/0 phyC:00  [PROV]BLE connection success  [ROV] prov\_srv\_fn\_wifi\_scan  prov\_event\_thread\_entry: msg->event = 1  [PROV]203 : len = 11  [PROV] WiFi scan: Found 6 networks  [PROV]231 : wc.scan\_rslts\_len = 70  [PROV]231 : wc.scan\_rslts\_len = 136  [PROV]231 : wc.scan\_rslts\_len = 197  [PROV]231 : wc.scan\_rslts\_len = 250  [PROV]231 : wc.scan\_rslts\_len = 313  [PROV]231 : wc.scan\_rslts\_len = 373  [PROV]237 : len = 375  [PROV] prov\_srv\_fn\_wifi\_scan\_rslts : length = 22  [PROV] prov\_srv\_fn\_wifi\_scan\_rslts : offset = 0  [PROV] prov\_wifi\_scan\_rslts\_send : length = 22, index=0 , rslt\_len=375  [PROV]prov\_wifi\_scan\_rslts\_send: offset = 0 remaining\_bytes = 375, copy\_len = 18  [PROV] prov\_wifi\_scan\_rslts\_send : wc.index = 18  [PROV]prov\_wifi\_scan\_rslts\_send: index = 18, scan\_rslts\_len = 375  Sending  [PROV] prov\_srv\_fn\_wifi\_scan\_rslts : length = 22  [PROV] prov\_srv\_fn\_wifi\_scan\_rslts : offset = 22  [PROV] prov\_wifi\_scan\_rslts\_send : length = 22, index=18 , rslt\_len=375  [PROV]prov\_wifi\_scan\_rslts\_send: offset = 22 remaining\_bytes = 357, copy\_len = 22  [PROV] prov\_wifi\_scan\_rslts\_send : wc.index = 40  [PROV]prov\_wifi\_scan\_rslts\_send: index = 40, scan\_rslts\_len = 375  Sending  :"innotest","bssid":"c  [PROV] prov\_srv\_fn\_wifi\_scan\_rslts : length = 22  [PROV] prov\_srv\_fn\_wifi\_scan\_rslts : offset = 44  [PROV] prov\_wifi\_scan\_rslts\_send : length = 22, index=40 , rslt\_len=375  [PROV]prov\_wifi\_scan\_rslts\_send: offset = 44 remaining\_bytes = 335, copy\_len = 22  [PROV] prov\_wifi\_scan\_rslts\_send : wc.index = 62  [PROV]prov\_wifi\_scan\_rslts\_send: index = 62, scan\_rslts\_len = 375  Sending  e:66:97:a2:74:ec","cha  [PROV] prov\_srv\_fn\_wifi\_scan\_rslts : length = 22  [PROV] prov\_srv\_fn\_wifi\_scan\_rslts : offset = 66  [PROV] prov\_wifi\_scan\_rslts\_send : length = 22, index=62 , rslt\_len=375  [PROV]prov\_wifi\_scan\_rslts\_send: offset = 66 remaining\_bytes = 313, copy\_len = 22  [PROV] prov\_wifi\_scan\_rslts\_send : wc.index = 84  [PROV]prov\_wifi\_scan\_rslts\_send: index = 84, scan\_rslts\_len = 375  Sending  nnel":6},{"ssid":"DEN  [PROV] prov\_srv\_fn\_wifi\_scan\_rslts : length = 22  [PROV] prov\_srv\_fn\_wifi\_scan\_rslts : offset = 88  [PROV] prov\_wifi\_scan\_rslts\_send : length = 22, index=84 , rslt\_len=375  [PROV]prov\_wifi\_scan\_rslts\_send: offset = 88 remaining\_bytes = 291, copy\_len = 22  [PROV] prov\_wifi\_scan\_rslts\_send : wc.index = 106  [PROV]prov\_wifi\_scan\_rslts\_send: index = 106, scan\_rslts\_len = 375  Sending  speaker.o,","bssid":"f  [PROV] prov\_srv\_fn\_wifi\_scan\_rslts : length = 22  [PROV] prov\_srv\_fn\_wifi\_scan\_rslts : offset = 110  [PROV] prov\_wifi\_scan\_rslts\_send : length = 22, index=106 , rslt\_len=375  [PROV]prov\_wifi\_scan\_rslts\_send: offset = 110 remaining\_bytes = 269, copy\_len = 22  [PROV] prov\_wifi\_scan\_rslts\_send : wc.index = 128  [PROV]prov\_wifi\_scan\_rslts\_send: index = 128, scan\_rslts\_len = 375  Sending  a:8f:ca:85:b6:69","cha  [PROV] prov\_srv\_fn\_wifi\_scan\_rslts : length = 22  [PROV] prov\_srv\_fn\_wifi\_scan\_rslts : offset = 132  [PROV] prov\_wifi\_scan\_rslts\_send : length = 22, index=128 , rslt\_len=375  [PROV]prov\_wifi\_scan\_rslts\_send: offset = 132 remaining\_bytes = 247, copy\_len = 22  [PROV] prov\_wifi\_scan\_rslts\_send : wc.index = 150  [PROV]prov\_wifi\_scan\_rslts\_send: index = 150, scan\_rslts\_len = 375  Sending  nnel":6},{"ssid":"Mura  [PROV] prov\_srv\_fn\_wifi\_scan\_rslts : length = 22  [PROV] prov\_srv\_fn\_wifi\_scan\_rslts : offset = 154  [PROV] prov\_wifi\_scan\_rslts\_send : length = 22, index=150 , rslt\_len=375  [PROV]prov\_wifi\_scan\_rslts\_send: offset = 154 remaining\_bytes = 225, copy\_len = 22  [PROV] prov\_wifi\_scan\_rslts\_send : wc.index = 172  [PROV]prov\_wifi\_scan\_rslts\_send: index = 172, scan\_rslts\_len = 375  Sending  li3G","bssid":"8c:a3:9  [PROV] prov\_srv\_fn\_wifi\_scan\_rslts : length = 22  [PROV] prov\_srv\_fn\_wifi\_scan\_rslts : offset = 176  [PROV] prov\_wifi\_scan\_rslts\_send : length = 22, index=172 , rslt\_len=375  [PROV]prov\_wifi\_scan\_rslts\_send: offset = 176 remaining\_bytes = 203, copy\_len = 22  [PROV] prov\_wifi\_scan\_rslts\_send : wc.index = 194  [PROV]prov\_wifi\_scan\_rslts\_send: index = 194, scan\_rslts\_len = 375  Sending  9:57:ef:b4","channel":  [PROV] prov\_srv\_fn\_wifi\_scan\_rslts : length = 22  [PROV] prov\_srv\_fn\_wifi\_scan\_rslts : offset = 198  [PROV] prov\_wifi\_scan\_rslts\_send : length = 22, index=194 , rslt\_len=375  [PROV]prov\_wifi\_scan\_rslts\_send: offset = 198 remaining\_bytes = 181, copy\_len = 22  [PROV] prov\_wifi\_scan\_rslts\_send : wc.index = 216  [PROV]prov\_wifi\_scan\_rslts\_send: index = 216, scan\_rslts\_len = 375  Sending  11},{"ssid":"","bssid"  [PROV] prov\_srv\_fn\_wifi\_scan\_rslts : length = 22  [PROV] prov\_srv\_fn\_wifi\_scan\_rslts : offset = 220  [PROV] prov\_wifi\_scan\_rslts\_send : length = 22, index=216 , rslt\_len=375  [PROV]prov\_wifi\_scan\_rslts\_send: offset = 220 remaining\_bytes = 159, copy\_len = 22  [PROV] prov\_wifi\_scan\_rslts\_send : wc.index = 238  [PROV]prov\_wifi\_scan\_rslts\_send: index = 238, scan\_rslts\_len = 375  Sending  :"8e:a3:99:57:ef:b4","  [PROV] prov\_srv\_fn\_wifi\_scan\_rslts : length = 22  [PROV] prov\_srv\_fn\_wifi\_scan\_rslts : offset = 242  [PROV] prov\_wifi\_scan\_rslts\_send : length = 22, index=238 , rslt\_len=375  [PROV]prov\_wifi\_scan\_rslts\_send: offset = 242 remaining\_bytes = 137, copy\_len = 22  [PROV] prov\_wifi\_scan\_rslts\_send : wc.index = 260  [PROV]prov\_wifi\_scan\_rslts\_send: index = 260, scan\_rslts\_len = 375  Sending  channel":11},{"ssid":"  [PROV] prov\_srv\_fn\_wifi\_scan\_rslts : length = 22  [PROV] prov\_srv\_fn\_wifi\_scan\_rslts : offset = 264  [PROV] prov\_wifi\_scan\_rslts\_send : length = 22, index=260 , rslt\_len=375  [PROV]prov\_wifi\_scan\_rslts\_send: offset = 264 remaining\_bytes = 115, copy\_len = 22  [PROV] prov\_wifi\_scan\_rslts\_send : wc.index = 282  [PROV]prov\_wifi\_scan\_rslts\_send: index = 282, scan\_rslts\_len = 375  Sending  Mithrandir","bssid":"0  [PROV] prov\_srv\_fn\_wifi\_scan\_rslts : length = 22  [PROV] prov\_srv\_fn\_wifi\_scan\_rslts : offset = 286  [PROV] prov\_wifi\_scan\_rslts\_send : length = 22, index=282 , rslt\_len=375  [PROV]prov\_wifi\_scan\_rslts\_send: offset = 286 remaining\_bytes = 93, copy\_len = 22  [PROV] prov\_wifi\_scan\_rslts\_send : wc.index = 304  [PROV]prov\_wifi\_scan\_rslts\_send: index = 304, scan\_rslts\_len = 375  Sending  0:5f:67:f9:12:e4","cha  [PROV] prov\_srv\_fn\_wifi\_scan\_rslts : length = 22  [PROV] prov\_srv\_fn\_wifi\_scan\_rslts : offset = 308  [PROV] prov\_wifi\_scan\_rslts\_send : length = 22, index=304 , rslt\_len=375  [PROV]prov\_wifi\_scan\_rslts\_send: offset = 308 remaining\_bytes = 71, copy\_len = 22  [PROV] prov\_wifi\_scan\_rslts\_send : wc.index = 326  [PROV]prov\_wifi\_scan\_rslts\_send: index = 326, scan\_rslts\_len = 375  Sending  nnel":13},{"ssid":"TP-  [PROV] prov\_srv\_fn\_wifi\_scan\_rslts : length = 22  [PROV] prov\_srv\_fn\_wifi\_scan\_rslts : offset = 330  [PROV] prov\_wifi\_scan\_rslts\_send : length = 22, index=326 , rslt\_len=375  [PROV]prov\_wifi\_scan\_rslts\_send: offset = 330 remaining\_bytes = 49, copy\_len = 22  [PROV] prov\_wifi\_scan\_rslts\_send : wc.index = 348  [PROV]prov\_wifi\_scan\_rslts\_send: index = 348, scan\_rslts\_len = 375  Sending  LINK","bssid":"b0:be:7  [PROV] prov\_srv\_fn\_wifi\_scan\_rslts : length = 22  [PROV] prov\_srv\_fn\_wifi\_scan\_rslts : offset = 352  [PROV] prov\_wifi\_scan\_rslts\_send : length = 22, index=348 , rslt\_len=375  [PROV]prov\_wifi\_scan\_rslts\_send: offset = 352 remaining\_bytes = 27, copy\_len = 22  [PROV] prov\_wifi\_scan\_rslts\_send : wc.index = 370  [PROV]prov\_wifi\_scan\_rslts\_send: index = 370, scan\_rslts\_len = 375  Sending  6:67:c7:a5","channel":  [PROV] prov\_srv\_fn\_wifi\_scan\_rslts : length = 22  [PROV] prov\_srv\_fn\_wifi\_scan\_rslts : offset = 374  [PROV] prov\_wifi\_scan\_rslts\_send : length = 22, index=370 , rslt\_len=375  [PROV]prov\_wifi\_scan\_rslts\_send: offset = 374 remaining\_bytes = 5, copy\_len = 5  [PROV] prov\_wifi\_scan\_rslts\_send : wc.index = 375  [PROV]prov\_wifi\_scan\_rslts\_send: index = 375, scan\_rslts\_len = 375  Sending  11}]}  scan\_len is greaterthan index  [PROV] prov\_srv\_fn\_wifi\_scan\_rslts : length = 22  [PROV] prov\_srv\_fn\_wifi\_scan\_rslts : offset = 0  [PROV] prov\_wifi\_scan\_rslts\_send : length = 22, index=0 , rslt\_len=0  [PROV] prov\_wifi\_scan\_rslts\_send : line = 257  [PROV]prov\_srv\_fn\_cfg\_ssid:8 innotest  [PROV]prov\_conf\_update: key = ssid, val = innotest  [PROV]prov\_conf\_update: prov\_json\_object\_update ret = 0  [PROV]prov\_conf\_update: prov\_json\_object\_update ret = 0  prov\_srv\_fn\_cfg\_ssid : 1 0  [PROV]prov\_conf\_update: key = passphrase, val = 1234567890  [PROV]prov\_conf\_update: prov\_json\_object\_update ret = 0  [PROV]prov\_conf\_update: prov\_json\_object\_update ret = 0  prov\_srv\_fn\_cfg\_passphrase : 1 1  prov\_event\_thread\_entry: msg->event = 2  ssid = innotest  passphrase = 1234567890  [PROV]Checking connectivity to...  ssid = innotest passphrase = 1234567890  network profile created for ssid: innotest  [27.398,799] CONNECT:ce:66:97:a2:74:ec Channel:6 rssi:-53 dBm  [PROV] prov\_srv\_fn\_wifi\_conn\_state: data = waiting[33.014,922] MYIP 172.20.10.2  [33.015,085] IPv6 [fe80::e269:3aff:fe00:a5a]-link  [33.015,149] IPv6 [2401:4900:4e5a:edaa:e269:3aff:fe00:a5a]  prov\_hio\_data\_cb  Sending Packet, size=: 16    prov\_hio\_data\_cb  Sending Packet, size=: 18  [33.220,501] DISCONNECTED  [PROV] prov\_srv\_fn\_wifi\_conn\_state: data = success  [PROV] prov\_srv\_fn\_wifi\_conn\_state: data = success  [ROV] prov\_srv\_fn\_cfg\_apply  [PROV]prov\_conf\_set\_provisioned 173  [PROV]prov\_conf\_set\_provisioned 197  [PROV]prov\_conf\_data\_apply:354  [PROV]prov\_conf\_data\_apply:357  {  "image": [  {  "name": "dual\_stack\_app",  "version": "1.0",  "start\_sector": 32,  "bootargs\_start": 1,  "hio.transport": "0",  "hio.maxsize": "4096",  "hio.sdio\_mhz": "10",  "krn.coredump": "1",  "krn.trace": "R:0x3ff",  "krn.cop\_error\_fatal": "1",  "ssid": "innotest",  "passphrase": "1234567890",  "bootargs\_end": 1  },  {  "name": "dual\_stack\_app",  "version": "0.0",  "start\_sector": 208,  "bootargs\_start": 1,  "hio.transport": "0",  "hio.maxsize": "4096",  "hio.sdio\_mhz": "10",  "krn.coredump": "1",  "krn.trace": "R:0x3ff",  "krn.cop\_error\_fatal": "1",  "ssid": "innotest",  "passphrase": "1234567890",  "bootargs\_end": 1  }  ],  "baudrate": 2560000,  "timeout": 0,  "verbose": 1  }  [PROV]prov\_conf\_data\_apply:362  prov\_hio\_cb : status = 1  prov\_hio\_data\_cb  Sending Packet, size=: 8    prov\_stop: Interface(Bit map) = 2  dual\_stack\_btinitBT HCI: Ready |