This application note demonstrates the use of APIs to create a provisioning application using BLE as the mode of transferring provisioning data. A reusable module is provided for provisioning the device in the following location: *freertos\_sdk\_x.y\components\prov*. The code in this sample application can be used in any user application to use the provisioning feature.

For more information regarding the module and the APIs, refer: *prov\_apiref.pdf*.

# Topology

Talaria TWO

BLE Mobile Application

Access Point

BLE

Wi-Fi

Provisioning

Figure : Topology

# List of APIs

1. prov\_start(): Initializes and starts the Provisioning service.
2. prov\_conf\_param\_str\_get(): Gets the configuration parameters of type string.
3. prov\_conf\_param\_int\_get(): Gets the configuration parameter of type integer.
4. prov\_is\_provisioned(): Checks if the device is provisioned at least once.
5. wcm\_create(): Creates a Wi-Fi network interface.
6. wcm\_add\_network\_profile(): Adds a network profile to WCM.
7. wcm\_auto\_connect(): Starts or stops auto connect.
8. wcm\_notify\_enable(): Enables callbacks of link and IP address changes.

# Functionality overview

Provisioning module exposes states and data through Provisioning service. Each state and data are provided through respective characteristics.

Following are a list of states and data exposed using the Provisioning Service:

1. Connection state: Wi-fi connection state
2. Connect: Connect to the configured Wi-Fi N/w
3. Scan state: Wi-Fi Scanning state
4. Scan: Trigger scan
5. Scan results: Scan results
6. Conf data: Configuration data. This data can be read from and writen to configuration file (currently it is: /root/part.json file)
7. Reset: Device reset
8. SSID: Configure SSID
9. Passphrase: Configure Passphrase

These states and data can be used by the Provisioning application (application running on a smart device) in various combination.

Following are the list of activities that can be executed by Provisioning application using these states and data:

1. Trigger scan and get scan results:

Wi-Fi scan can be triggered by setting the “Scan” state. This will trigger the Wi-Fi scan and scan results are stored in the internal buffer in Talaria TWO. After setting the “Scan”, the application needs to poll the “Scan state”. When the Scan state becomes 2, scan results are ready to be read. At this point read “scan results”.

1. Get configuration data:

Configuration data can be acquired by reading the “Conf data”. This is the entire content of the configuration file.

1. Write Configuration Data:

After changing the required configuration parameters, configuration data can be written back(stored) to Talaria TWO by writing “Conf data”. The updated data is stored back in the configuration file in root fs.

1. Check connectivity:

Once the configuration data is written(Provisioning is done), Wi-Fi connectivity can be checked for. Setting “Connect” will trigger the same.

1. SSID and Passphrase can be configured without getting the full configuration data. These parameters can be written using “ssid” and “passphrase”.

One way to provision is to change the SSID and Passphrase by writing to “ssid” and “passphrase” characteristics. If any more parameters needs to be changed, the following procedure needs to be followed:

Another way to provision the device would be to get the config data file content, change the required parameters and write it back. Additionally, before changing the config parameter like SSID, user can select the network from the SSID list in the scan results. After provisioning, connectivity check can also be done.

Either one of the mentioned methods/procedures need to be followed to provision the device.

# Data Exchange Formats

## Scan Data

This data is sent from Talaria TWO to the Provisioning application when “scan results” is read.

Format:

|  |
| --- |
| <Length of the scan data – 4 bytes>< Scan Data …> |

Scan data format:

Scan data is in the json format with one or more elements. Each element of the array gives SSID, BSSID and the channel for a discovered network.

|  |
| --- |
| { "result": [ {"ssid":<ssid string>", "bssid":"<bssid string>", "channel":<channel num str>}…….]} |

## Configuration Data Format

Configuration data is the content of part.json file. It is sent from Talaria TWO to the Provisioning application and vice versa in the following format:

|  |
| --- |
| <Length of the config data – 4 bytes>< Config Data …> |

Configuration data is currently stored in part.json file. Hence, when the read operation is executed on this characteristic, whole of the part.json file content is sent as is. The configuration parameters are stored in this file between bootargs\_start and bootargs\_end. The Provisioning application changes the required parameters and sends the entire file content back. At Talaria TWO, the file is overwritten with new data.

Ensure that the “provisioned” is set to 1 in the updated configuration data.

Following is the basic content of part.json file:

|  |
| --- |
| {  "image" : [  {  "name" : "fota",  "version" : "1.0",  "start\_sector" : 32,  "bootargs\_start": 1,  "ssid" : "inno\_test",  "passphrase" : "1234567890",  "bootargs\_end" : 1  },  {  "name" : "test\_app",  "version" : "1.0",  "start\_sector" : 154,  "bootargs\_start": 1,  "ssid" : "inno\_test",  "passphrase" : "1234567890",  "bootargs\_end" : 1  }  ],  "provisioned" : 0,  "baudrate" : 2560000,  "timeout" : 0,  "verbose" : 1  } |

# Provisioning Profile

Provisioning profile has one service and multiple characteristics. The service name is configurable. The following sections provide information about the provisioning service and the characteristics.

## Service

Service UUID:

|  |
| --- |
| 11111111-1111-1111-1111111111111111 |

Service name:

Configurable. Default is "Inno\_Provisioning".

## Characteristics

1. Connection state:
   1. UUID: 0xAB30
   2. Properties: Read
   3. Description: Gives the current state of Wi-Fi connection of Talaria TWO
   4. Potential Values:

|  |  |
| --- | --- |
| **Value** | **Description** |
| Failure | Unable to connect to the provisioned Wi-Fi network |
| Success | Connected |
| Waiting | Connection in Progress |

Table 5: Potential values of the connection state

1. Connect:
   1. UUID: 0xAB31
   2. Properties: Write
   3. Description: When written, any 1-byte value (just typing 1), will trigger the Wi-Fi connect using the configured SSID and passphrase.
2. Scan State
   1. UUID: 0xAB32
   2. Properties: Read
   3. Description: Gives the current state of the Wi-Fi scanning activity
   4. Potential Values:

|  |  |
| --- | --- |
| **Value** | **Description** |
| 0 | Not Scanning |
| 1 | Scanning Complete |
| 2 | Scanning in Progress |

Table 6: Potential values of the scan state

1. Scan:
   1. UUID: 0xAB33
   2. Properties: Write
   3. Description: When any 1-byte value is written (just typing 1), will trigger Wi-Fi scan. The scan results are stored internally but are not sent.
2. Scan Results:
   1. UUID: 0xAB34
   2. Properties: Read
   3. Description: When read, the scan results are sent from Talaria TWO to the Provisioning application. The format of the scan results are as provided in section: *Scan Data*. Provisioning application may have to read multiple times to get the full data.
3. Configuration data:
   1. UUID: 0xAB35
   2. Properties: Read/Write
   3. Description: When read, configuration data is sent from Talaria TWO to the Provisioning application. The format of the scan results is as provided in section: *Scan Data*. Provisioning application may have to read multiple times to get the full data.
4. Apply and Reset
   1. UUID: 0xAB36
   2. Properties: Write
   3. Description: When 1 is written, the configuration will be applied (stored into the configuration file) and the device is reset.
5. Apply
   1. UUID: 0xAB37
   2. Properties: Write
   3. Description: When written with any 1-byte value, the configuration will be applied (stored into the configuration file).
6. ssid
   1. UUID: 0xAB38
   2. Properties: Write
   3. Description: Write a maximum of 31 bytes of SSID.
7. passphrase
   1. UUID: 0xAB39
   2. Properties: Write
   3. Description: Write a maximum of 31 bytes of passphrase.

# Code Walkthrough

## Application Flow

In this application, provisioning is done over BLE, and the configuration data is stored in the file system.

Following are the steps:

1. Initiate BLE provisioning by creating a GATT server.
2. Provision the SSID and passphrase by selecting the network from the SSID list created from scan results using the BLE Provisioning mobile app.
3. Store the configuration data.
4. Check Wi-Fi connectivity using the configured SSID and passphrase.

## Sample Code Walkthrough

A user-defined data struct is created to store provisioning data.

|  |
| --- |
| struct param\_t {  const char \*ssid;  const char \*passphrase;  const char \*dummy;  }; |

prov\_start\_prams\_t and prov\_cfg is the pointer to the data structure of type prov\_start\_prams\_t.

Provisioning method, device name, appearance, manufacturer name, user callback and context pointer are passed to structure members of prov\_start\_prams\_t.

|  |
| --- |
| if(0 == prov\_is\_provisioned()){  /\*Provisioning\*/  prov\_start\_prams\_t prov\_cfg = {0};  prov\_cfg.method = PROV\_METHOD\_BLE;  prov\_cfg.cb = app\_prov\_cb;  /\*"custparam" is configurable parameter present in part.json file  provisioning module does not add this parametr bu default to the prov  service. Need to add it exlicietly\*/  prov\_custom\_param\_add(0xAA44, "custparam");  prov\_start(&prov\_cfg);  os\_sem\_wait(&app\_lock); } |

prov\_start API starts the provisioning service and prov\_is\_provisioned checks if provisioning is complete. Here, the prov\_conf\_param\_str\_get API is used to get the provisioning parameters such as SSID and passphrase. These parameters are stored in user defined structure param\_t.

|  |
| --- |
| /\*if provisioning is done. use provisioned params\*/  if(prov\_is\_provisioned()){  param.ssid = prov\_conf\_param\_str\_get("ssid");  param.passphrase = prov\_conf\_param\_str\_get("passphrase");  os\_printf("\n[APP] %s %s", param.ssid, param.passphrase); } |

wcm\_create()API creates a Wi-Fi network interface and connects to network using provisioned parameters.

|  |
| --- |
| wcm\_handle = wcm\_create(NULL);  app\_wcm\_connect(wcm\_handle, (char \*)param.ssid, (char \*)param.passphrase);  if(!wcm\_connect\_success) {  os\_printf("\n[APP]Error: Failed to connect to WiFi N/w");  goto exit; }  os\_printf("\n[APP]Success: Connected to %s N/w", (char \*)param.ssid); |

# Running the Application

## Write the Provisioning File into Talaria TWO Filesystem

Using the Download Tool’s Write Files feature, write the file (part.json) present in *freertos\_sdk\_x.y\examples\prov\data* into Talaria TWO filesystem.

This can be accomplished by using the option: Filesystem->Write Files from a directory->(File Open Icon). Choose the directory *sdk\_x.y\examples\prov\data* and click on Write Files option.

**Note**: To re-provision the application, write the files using the procedure described above.

## Programming Talaria TWO board with ELF

Program prov\_app.elf (*examples\prov\bin\prov\_app.elf*) using the Download tool.

1. Launch the Download tool provided with InnoPhase Talaria TWO SDK.
2. In the GUI window:
   1. Boot Target: Select the appropriate EVK from the drop-down.
   2. ELF Input: Load the prov\_app.elf by clicking on Select ELF File.
   3. Programming: Click on Prog Flash.

Once done, follow the instructions mentioned in section 7.3 to run the mobile

**Note**:

1. There is no requirement of boot arguments. User can run the application directly. On running the application, it creates a provisioning GATT server.

## Running the Application using Android or iOS App

1. Download and install the Android or iOS mobile application from the play/app store using the following links:
   1. **Android**:<https://play.google.com/store/apps/details?id=com.innophase.provisioning&hl=en&gl=US>
   2. **iOS**: <https://apps.apple.com/in/app/ble-provisioning/id1627682345>

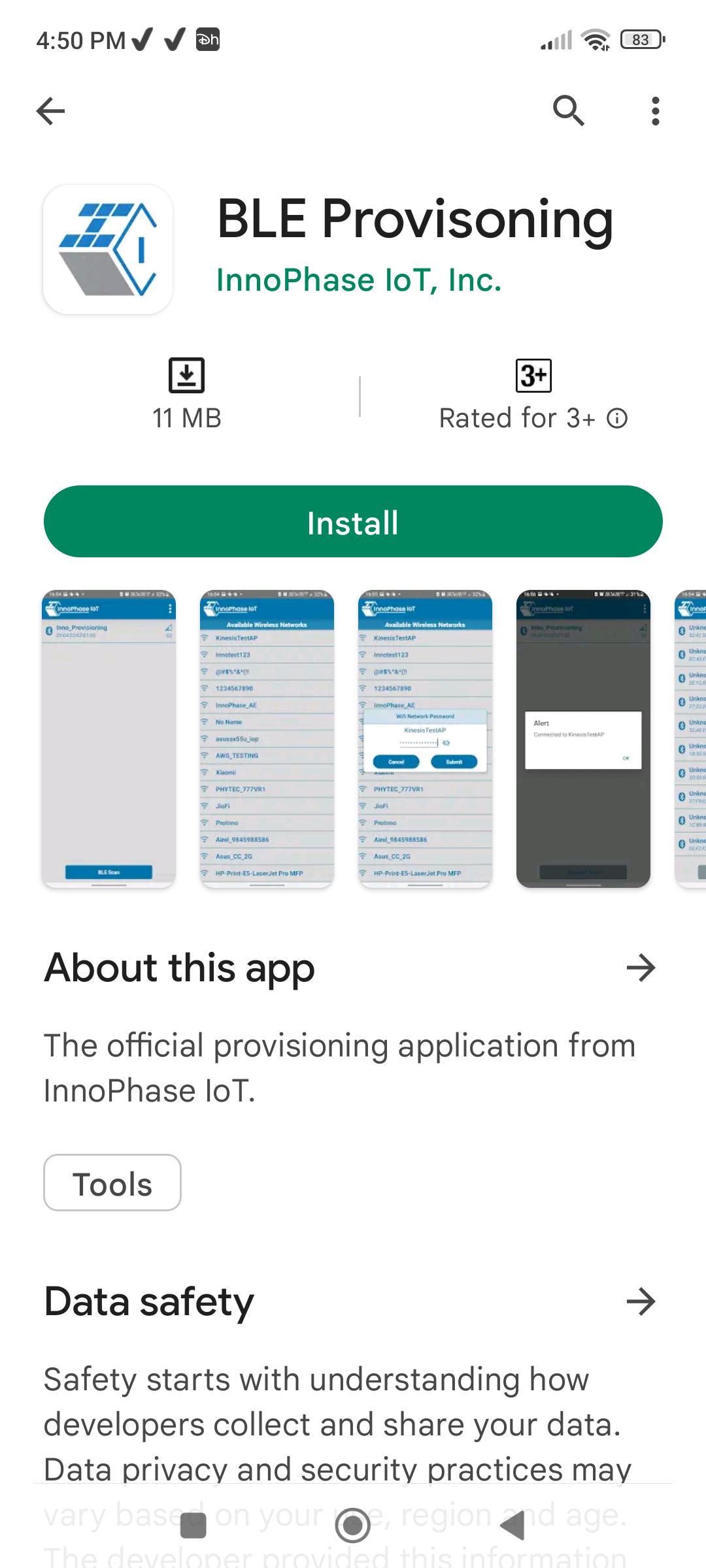


Figure : Android mobile app in play store

Graphical user interface, application

Description automatically generated

Figure : iOS mobile app in app store

1. BLE Provisioning can be enabled via non-secure and secure BLE connections. Following are the three supported methods:
   1. Non-secure provisioning
   2. Secure Provisioning – Just Work Method
   3. Secure Provisioning – Passkey Method
2. Non-secure Provisioning

On flashing prov\_app.elf onto Talaria TWO using the Download Tool, the following console logs are observed.

Talaria TWO console output:

|  |
| --- |
| Y-BOOT 208ef13 2019-07-22 12:26:54 -0500 790da1-b-7  ROM yoda-h0-rom-16-0-gd5a8e586  FLASH:PWWWWWWAE  Build $Id: git-2eebd42 $  Flash detected. flash.hw.uuid: 39483937-3207-0080-0055-ffffffffffff  Bootargs: prov\_every\_time=1  Application Information:  ------------------------  Name : Prov App  Version : 1.0  Build Date : Sep 8 2023  Build Time : 13:58:53  Heap Available: 297 KB (304152 Bytes)  [APP]Bootparams :  --------------------  ssid= <null>  passphrase= <null>  dummy= <null>  secured\_prov =0  pass\_key=0  prov\_every\_time=1  [APP]Bootparams end here....  [PROV]prov\_is\_provisioned : val = 1  [PROV] bt\_gap\_init  [PROV] common\_server\_create: Inno\_Provisioning InnophaseInc 0  [PROV] bt\_gatt\_create\_service\_128 ret = 0x000bf278  [PROV] Adding Characteristics  [PROV] bt\_gatt\_add\_service  [PROV] bt\_gap\_discoverable\_mode, ret = 0 |

1. Secure Provisioning

In case of secure BLE provisioning, flash the prov\_app.elf onto Talaria TWO with the following boot arguments as applicable:

* 1. Just Work Method: secured\_prov=1,pass\_key=0, or
  2. Passkey Method: secured\_prov=1,pass\_key=1

Console output:

**secured\_prov=1, pass\_key=0**

|  |
| --- |
| Y-BOOT 208ef13 2019-07-22 12:26:54 -0500 790da1-b-7  ROM yoda-h0-rom-16-0-gd5a8e586  FLASH:PWWWWWWAE  Build $Id: git-2eebd42 $  Flash detected. flash.hw.uuid: 39483937-3207-0080-0055-ffffffffffff  Bootargs: prov\_every\_time=1 secured\_prov=1  Application Information:  ------------------------  Name : Prov App  Version : 1.0  Build Date : Sep 8 2023  Build Time : 13:58:53  Heap Available: 297 KB (304152 Bytes)  [APP]Bootparams :  --------------------  ssid= <null>  passphrase= <null>  dummy= <null>  secured\_prov =1  pass\_key=0  prov\_every\_time=1  [APP]Bootparams end here....  [PROV]prov\_is\_provisioned : val = 1  [PROV] bt\_gap\_init  [PROV] common\_server\_create: Inno\_Provisioning InnophaseInc 1  [PROV] bt\_gatt\_create\_service\_128 ret = 0x000bf278  [PROV] Adding Characteristics  [PROV] bt\_gatt\_add\_service  [PROV] bt\_gap\_discoverable\_mode, ret = 0 |

**secured\_prov=1,pass\_key=1**

|  |
| --- |
| Y-BOOT 208ef13 2019-07-22 12:26:54 -0500 790da1-b-7  ROM yoda-h0-rom-16-0-gd5a8e586  FLASH:PWWWWWWAE  Build $Id: git-2eebd42 $  Flash detected. flash.hw.uuid: 39483937-3207-0080-0055-ffffffffffff  Bootargs: prov\_every\_time=1 secured\_prov=1 pass\_key=1  Application Information:  ------------------------  Name : Prov App  Version : 1.0  Build Date : Sep 8 2023  Build Time : 13:58:53  Heap Available: 297 KB (304152 Bytes)  [APP]Bootparams :  --------------------  ssid= <null>  passphrase= <null>  dummy= <null>  secured\_prov =1  pass\_key=1  prov\_every\_time=1  [APP]Bootparams end here....  [PROV]prov\_is\_provisioned : val = 1  [PROV] bt\_gap\_init  [PROV] common\_server\_create: Inno\_Provisioning InnophaseInc 1  [PROV] bt\_gatt\_create\_service\_128 ret = 0x000bf278  [PROV] Adding Characteristics  [PROV] bt\_gatt\_add\_service  [PROV] bt\_gap\_discoverable\_mode, ret = 0 |

1. Launch the downloaded BLE Provisioning mobile app. App icon and UI are as shown in Figure 4.

A screenshot of a phone

Description automatically generated with medium confidence A picture containing text, screenshot, software, web page

Description automatically generated

Figure : BLE Provisioning home screen

1. To check the version of the Talaria TWO Provisioning App and other related details, user can click and open the About Us option.

Graphical user interface, text, application

Description automatically generated

Figure : About Us

Talaria TWO Provisioning Mobile App version is displayed as shown in Figure 6. Version varies for each release.

User has the following two options via which they can contact the InnoPhase support team.

1. Contact the sales team via email: [sales@innophaseiot.com](mailto:sales@innophaseiot.com)
2. LinkedIn, which redirects them to the InnoPhase website.



Figure : About Us - App version and other related details

1. Click on BLE Scan if there are no services listed.

Shape

Description automatically generated with low confidence

Figure : BLE Scan

1. If the user needs to scan for unknown devices, then the highlighted toggle switch should be enabled.

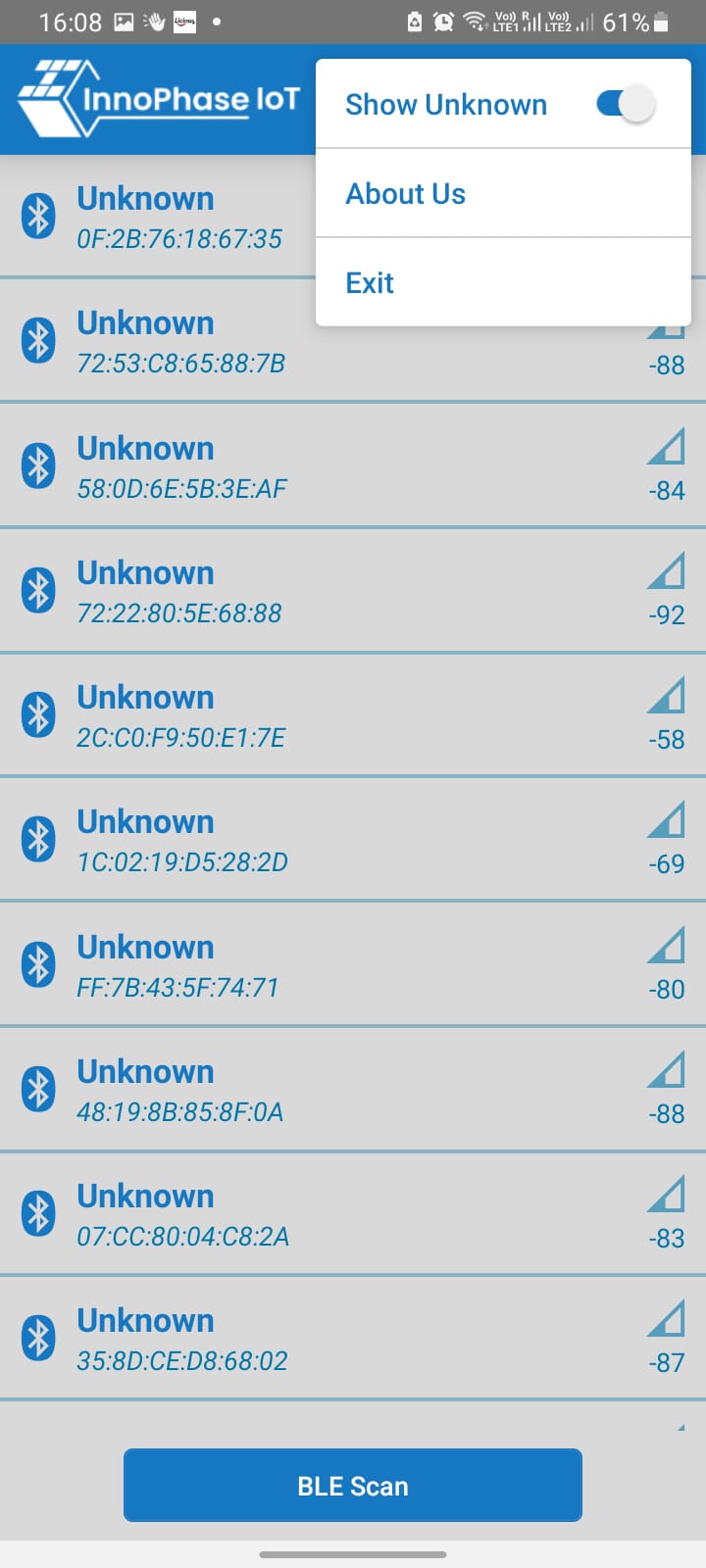


Figure : Show/List ‘Unknown’ devices

1. Click on the Inno\_Provisioning service.

Shape

Description automatically generated with low confidence

Figure : Starting Inno\_Provisioning

Console output: **non-secured**

|  |
| --- |
| [121.460,022] BT connect[0]: ia:68:be:27:2e:5f:f9 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  Creating wcm interfaceaddr e0:69:3a:00:2c:3c  [PROV]140 : len = 11  [PROV] WiFi scan: Found 7 networks  [PROV]168 : wc.scan\_rslts\_len = 75  [PROV]168 : wc.scan\_rslts\_len = 128  [PROV]168 : wc.scan\_rslts\_len = 194  [PROV]168 : wc.scan\_rslts\_len = 257  [PROV]168 : wc.scan\_rslts\_len = 322  [PROV]168 : wc.scan\_rslts\_len = 382  [PROV]168 : wc.scan\_rslts\_len = 446  [PROV]174 : len = 448  [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=448  [PROV]prov\_wifi\_scan\_rslts\_send: offset = 0 remaining\_bytes = 448, copy\_len = 18  [PROV] prov\_wifi\_scan\_rslts\_send : wc.index = 18  [PROV]prov\_wifi\_scan\_rslts\_send: index = 18, scan\_rslts\_len = 448  Sending  {"result":[{"ssid"  [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=448  [PROV]prov\_wifi\_scan\_rslts\_send: offset = 22 remaining\_bytes = 430, copy\_len = 22  [PROV] prov\_wifi\_scan\_rslts\_send : wc.index = 40  [PROV]prov\_wifi\_scan\_rslts\_send: index = 40, scan\_rslts\_len = 448  Sending  :"InnoPhase\_AE","bssid  [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=448  [PROV]prov\_wifi\_scan\_rslts\_send: offset = 44 remaining\_bytes = 408, copy\_len = 22  [PROV] prov\_wifi\_scan\_rslts\_send : wc.index = 62  [PROV]prov\_wifi\_scan\_rslts\_send: index = 62, scan\_rslts\_len = 448  Sending  ":"98:da:c4:73:b7:76",  [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=448  [PROV]prov\_wifi\_scan\_rslts\_send: offset = 66 remaining\_bytes = 386, copy\_len = 22  [PROV] prov\_wifi\_scan\_rslts\_send : wc.index = 84  [PROV]prov\_wifi\_scan\_rslts\_send: index = 84, scan\_rslts\_len = 448  Sending  "channel":11},{"ssid":  [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=448  [PROV]prov\_wifi\_scan\_rslts\_send: offset = 88 remaining\_bytes = 364, copy\_len = 22  [PROV] prov\_wifi\_scan\_rslts\_send : wc.index = 106  [PROV]prov\_wifi\_scan\_rslts\_send: index = 106, scan\_rslts\_len = 448  Sending  "","bssid":"aa:da:c4:7  [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=448  [PROV]prov\_wifi\_scan\_rslts\_send: offset = 110 remaining\_bytes = 342, copy\_len = 22  [PROV] prov\_wifi\_scan\_rslts\_send : wc.index = 128  [PROV]prov\_wifi\_scan\_rslts\_send: index = 128, scan\_rslts\_len = 448 |

1. For secured BLE provisioning, pairing with the device is a must. On clicking Inno\_Provisioning on the mobile application, a pop-up appears with a request to pair:

**Note**: If the mobile application is already paired with an existing provisioning advertisement, the user will have to explicitly remove the same from paired devices before proceeding.

For unpairing a device, go to:

* + 1. Settings → Connections → Bluetooth
    2. Click on the Settings icon on the paired device → Unpair.

1. secured\_prov=1,pass\_key=0

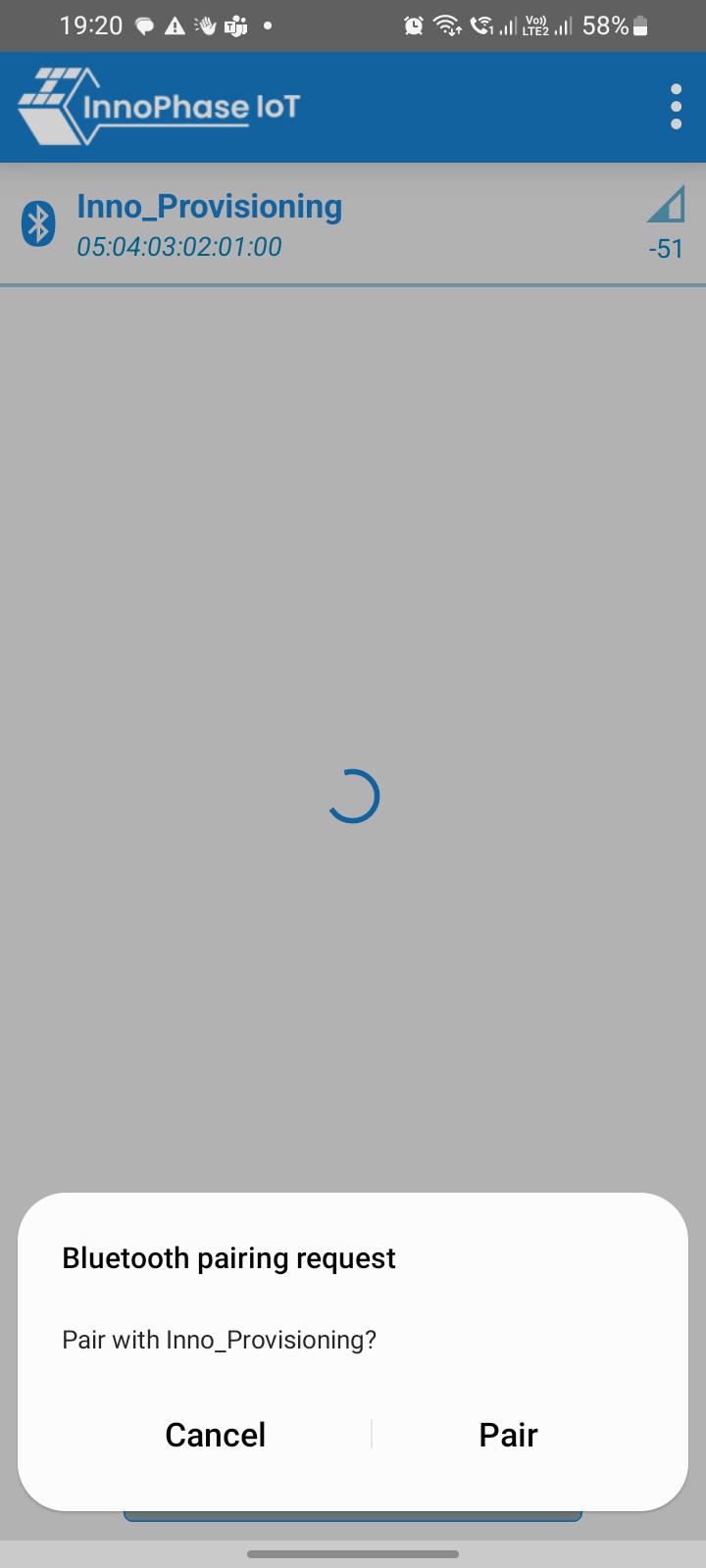


Figure : Pairing - open networks

1. secured\_prov=1,pass\_key=1

Enter the appropriate pass\_key (PIN) from the Talaria TWO console log to complete pairing.

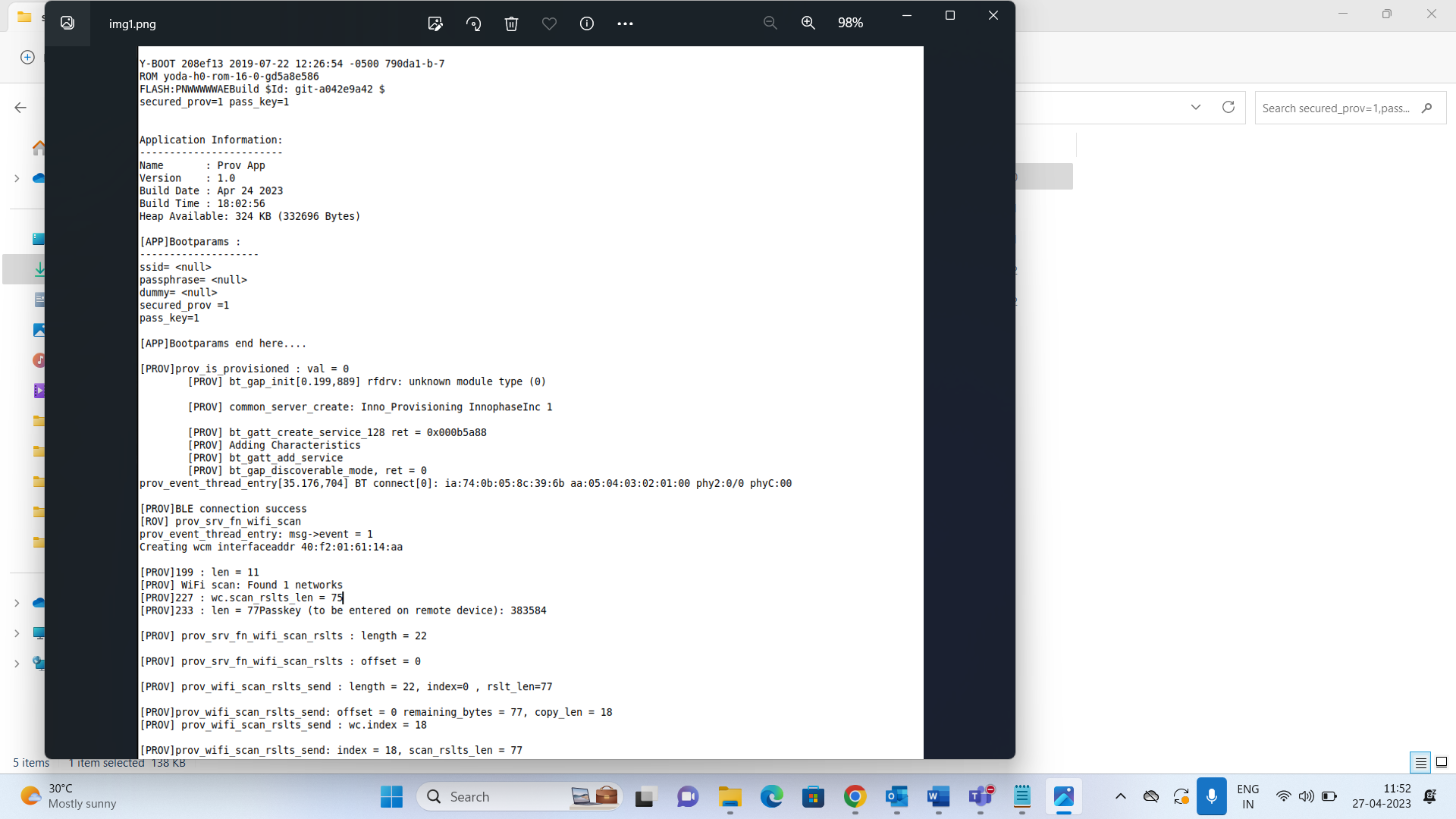


Figure : passkey (PIN) from the console log

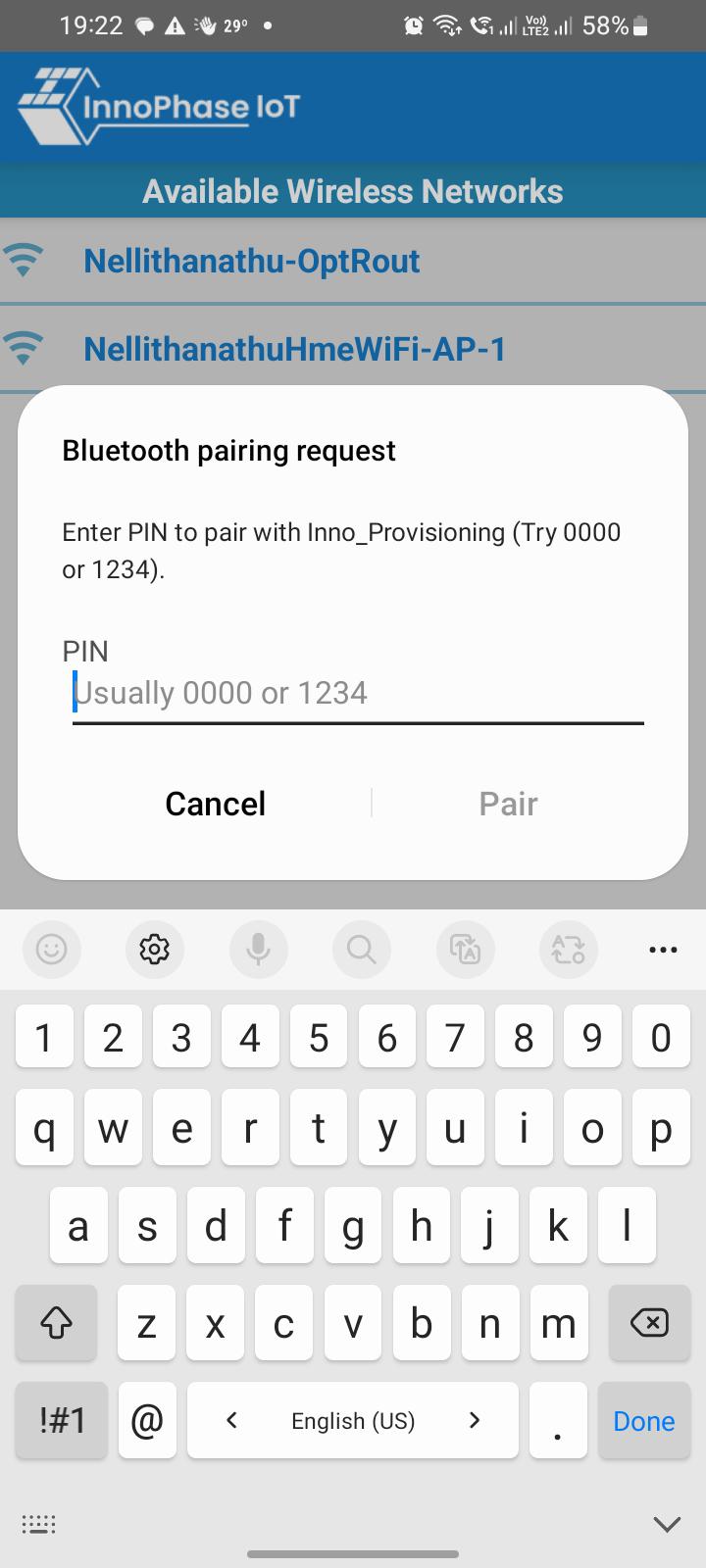


Figure : Pairing - with pass\_key

Once the pairing is complete, the screen with a list of available APs in the vicinity opens.

1. Available Wireless Network screen opens the AP listing available in the vicinity. Here, select the appropriate AP from the list.

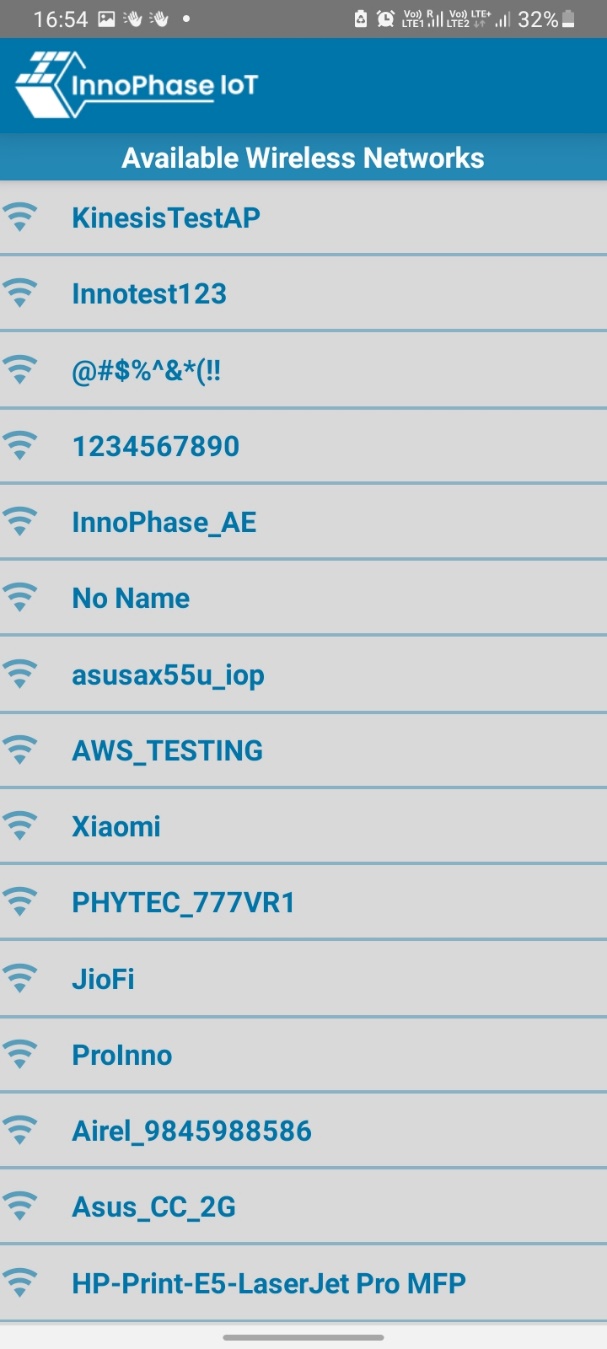


Figure : Selecting the appropriate AP from the list

1. Provide the passphrase for the secured AP and click on Submit. User can use the hide or show option to check the passphrase entered.

**Note**: If the user is connecting to an open network, they can keep the password space empty.

Graphical user interface, text, application

Description automatically generated

Figure : Connecting to InnoPhase\_AE network by providing passphrase

1. On successfully connecting to the AP, the following output is observed:

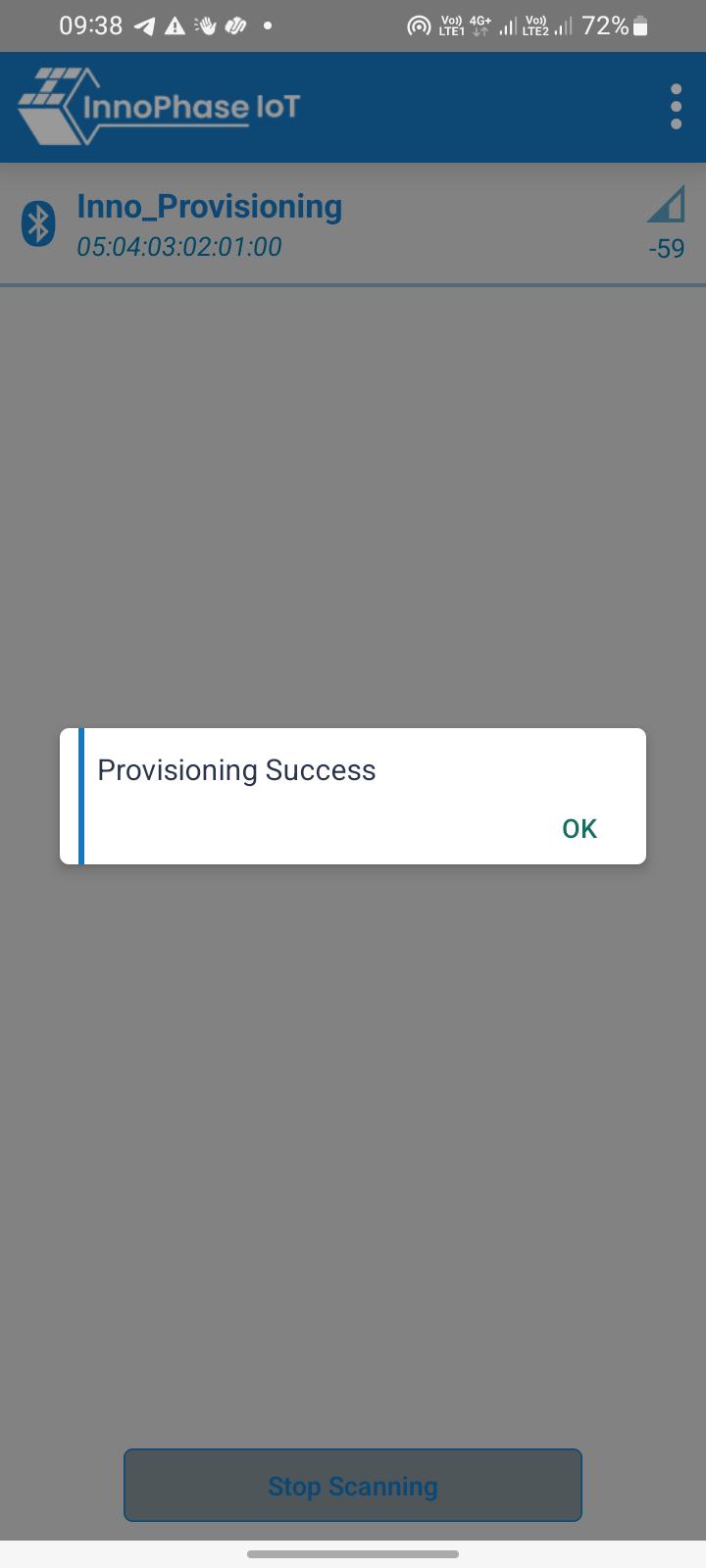


Figure : Successfully connected to InnoPhase network

Console output:

To connect to the configured network, “Apply and reset” is used to save the provisioned data and reset the application.

**Note**: If there is a need to re-provision the application, follow the procedure described in section 7.1.

**Non-secure provisioning**:

|  |
| --- |
| [PROV]prov\_srv\_fn\_cfg\_ssid:9 InnoPhase\_AE  [PROV]prov\_conf\_update: key = ssid, val = InnoPhase\_AE  [PROV]prov\_conf\_update: prov\_json\_object\_update ret = 0  prov\_srv\_fn\_cfg\_ssid : 1 0  [PROV]prov\_conf\_update: key = passphrase, val = Inno@1234  [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 = InnoPhase\_AE  passphrase = Inno@1234  [PROV]Checking connectivity to...  ssid = InnoPhase\_AE passphrase = Inno@1234  network profile created for ssid: InnoPhase  [PROV] prov\_srv\_fn\_wifi\_conn\_state: data = waiting[54.777,862] CONNECT:00:5f:67:cd:c5:a6 Channel:11 rssi:-54 dBm  [PROV] prov\_srv\_fn\_wifi\_conn\_state: data = waiting[55.512,208] MYIP 192.168.0.106  [55.512,488] IPv6 [fe80::e269:3aff:fe00:1390]-link  [PROV] prov\_srv\_fn\_wifi\_conn\_state: data = success  [PROV] prov\_srv\_fn\_wifi\_conn\_state: data = success  [ROV] prov\_srv\_fn\_cfg\_apply\_and\_reset  [PROV]prov\_conf\_set\_provisioned 168  [PROV]prov\_conf\_set\_provisioned 192  [PROV]prov\_conf\_data\_apply:342  [PROV]prov\_conf\_data\_apply:345  {  "image": [  {  "name": "prov\_app",  "version": "1.0",  "start\_sector": 32,  "bootargs\_start": 1,  "ssid": "InnoPhase\_AE",  "passphrase": "Inno@1234",  "custparam": "inno",  "bootargs\_end": 1  }  ],  "provisioned": 1,  "baudrate": 2560000,  "timeout": 0,  "verbose": 1  }  [PROV]prov\_conf\_data\_apply:350  app\_prov\_cb : status = 1  Y-BOOT 208ef13 2019-07-22 12:26:54 -0500 790da1-b-7  ROM yoda-h0-rom-16-0-gd5a8e586  FLASH:P  UART:SNWWWWAE  Build $Id: git-34e3eddb8 $  hio.baudrate=921600  flash: Gordon ready!  UART:SNWWWWWAEBuild $Id: git-34e3eddb8 $  Application Information:  ------------------------  Name : Prov App  Version : 1.0  Build Date : Aug 3 2022  Build Time : 03:56:17  Heap Available: 326 KB (333976 Bytes)  [APP]Bootparams :  --------------------  ssid= <null>  passphrase= <null>  dummy= <null>  [APP]Bootparams end here....  [PROV]prov\_is\_provisioned : val = 1  [PROV]prov\_is\_provisioned : val = 1  ssid = InnoPhase\_AE  passphrase = Inno@1234  [APP] InnoPhase\_AE Inno@1234addr e0:69:3a:00:13:90  app\_wcm\_connect:144 0x000bfa20 0x000bfa78 network profile created for ssid: InnoPhase\_AE  [2.340,175] CONNECT:98:da:c4:73:b7:76 Channel:11 rssi:-33 dBm  [2.387,308] MYIP 192.168.0.164  [2.387,586] IPv6 [fe80::e269:3aff:fe00:1390]-link  [APP]Success: Connected to InnoPhase\_AE N/w  [APP]------ Program Exit------------- |

**secured\_prov=1, pass\_key=0**

|  |
| --- |
| prov\_event\_thread\_entry[5.094,119] BT connect[0]: ia:7c:90:13:12:df:43 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  Creating wcm interfaceaddr e0:69:3a:00:16:cc  [PROV]199 : len = 11  [PROV] WiFi scan: Found 4 networks  [PROV]227 : wc.scan\_rslts\_len = 68  [PROV]227 : wc.scan\_rslts\_len = 132  [PROV]227 : wc.scan\_rslts\_len = 199  [PROV]227 : wc.scan\_rslts\_len = 258  [PROV]233 : len = 260Authentication succeeded.  [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=260  [PROV]prov\_wifi\_scan\_rslts\_send: offset = 0 remaining\_bytes = 260, copy\_len = 18  [PROV] prov\_wifi\_scan\_rslts\_send : wc.index = 18  [PROV]prov\_wifi\_scan\_rslts\_send: index = 18, scan\_rslts\_len = 260  [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=260  [PROV]prov\_wifi\_scan\_rslts\_send: offset = 44 remaining\_bytes = 220, copy\_len = 22  [PROV] prov\_wifi\_scan\_rslts\_send : wc.index = 62  [PROV]prov\_wifi\_scan\_rslts\_send: index = 62, scan\_rslts\_len = 260  Sending  7:2c:7c:ee:f4","channe  [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=260  [PROV]prov\_wifi\_scan\_rslts\_send: offset = 66 remaining\_bytes = 198, copy\_len = 22  [PROV] prov\_wifi\_scan\_rslts\_send : wc.index = 84  [PROV]prov\_wifi\_scan\_rslts\_send: index = 84, scan\_rslts\_len = 260  Sending  l":10},{"ssid":"dlink8  [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=260  [PROV]prov\_wifi\_scan\_rslts\_send: offset = 88 remaining\_bytes = 176, copy\_len = 22  [PROV] prov\_wifi\_scan\_rslts\_send : wc.index = 106  [PROV]prov\_wifi\_scan\_rslts\_send: index = 106, scan\_rslts\_len = 260  Sending  25\_iop","bssid":"34:0a  [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=260  [PROV]prov\_wifi\_scan\_rslts\_send: offset = 110 remaining\_bytes = 154, copy\_len = 22  [PROV] prov\_wifi\_scan\_rslts\_send : wc.index = 128  [PROV]prov\_wifi\_scan\_rslts\_send: index = 128, scan\_rslts\_len = 260  Sending  :33:91:bd:30","channel  [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=260  [PROV]prov\_wifi\_scan\_rslts\_send: offset = 132 remaining\_bytes = 132, copy\_len = 22  [PROV] prov\_wifi\_scan\_rslts\_send : wc.index = 150  [PROV]prov\_wifi\_scan\_rslts\_send: index = 150, scan\_rslts\_len = 260  Sending  ":1},{"ssid":"Asus\_86U  [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=260  [PROV]prov\_wifi\_scan\_rslts\_send: offset = 154 remaining\_bytes = 110, copy\_len = 22  [PROV] prov\_wifi\_scan\_rslts\_send : wc.index = 172  [PROV]prov\_wifi\_scan\_rslts\_send: index = 172, scan\_rslts\_len = 260  Sending  \_2G\_iop","bssid":"24:4  [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=260  [PROV]prov\_wifi\_scan\_rslts\_send: offset = 176 remaining\_bytes = 88, copy\_len = 22  [PROV] prov\_wifi\_scan\_rslts\_send : wc.index = 194  [PROV]prov\_wifi\_scan\_rslts\_send: index = 194, scan\_rslts\_len = 260  Sending  b:fe:5e:fd:d8","channe  [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=260  [PROV]prov\_wifi\_scan\_rslts\_send: offset = 198 remaining\_bytes = 66, copy\_len = 22  [PROV] prov\_wifi\_scan\_rslts\_send : wc.index = 216  [PROV]prov\_wifi\_scan\_rslts\_send: index = 216, scan\_rslts\_len = 260  Sending  l":1},{"ssid":"Asus777  [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=260  [PROV]prov\_wifi\_scan\_rslts\_send: offset = 220 remaining\_bytes = 44, copy\_len = 22  [PROV] prov\_wifi\_scan\_rslts\_send : wc.index = 238  [PROV]prov\_wifi\_scan\_rslts\_send: index = 238, scan\_rslts\_len = 260  Sending  ","bssid":"58:11:22:71  [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=260  [PROV]prov\_wifi\_scan\_rslts\_send: offset = 242 remaining\_bytes = 22, copy\_len = 22  [PROV] prov\_wifi\_scan\_rslts\_send : wc.index = 260  [PROV]prov\_wifi\_scan\_rslts\_send: index = 260, scan\_rslts\_len = 260  Sending  :f2:a8","channel":6}]}  [PROV]prov\_srv\_fn\_cfg\_ssid:12 dlink825\_iop  [PROV]prov\_conf\_update: key = ssid, val = dlink825\_iop  [PROV]prov\_conf\_update: prov\_json\_object\_update ret = 0  prov\_srv\_fn\_cfg\_ssid : 1 0  [PROV]prov\_conf\_update: key = passphrase, val = InnoQA2023$  [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 = dlink825\_iop  passphrase = InnoQA2023$  [PROV]Checking connectivity to...  ssid = dlink825\_iop passphrase = InnoQA2023$  network profile created for ssid: dlink825\_iop  [30.915,617] CONNECT:34:0a:33:91:bd:30 Channel:1 rssi:-32 dBm  [PROV] prov\_srv\_fn\_wifi\_conn\_state: data = waiting[35.658,449] MYIP 192.168.1.214  [35.658,613] IPv6 [fe80::e269:3aff:fe00:16cc]-link  [35.661,211] 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:347  [PROV]prov\_conf\_data\_apply:350  {  "image": [  {  "name": "prov\_app",  "version": "1.0",  "start\_sector": 32,  "bootargs\_start": 1,  "ssid": "dlink825\_iop",  "passphrase": "InnoQA2023$",  "custparam": "inno",  "new\_boot\_param\_key": "new\_boot\_param\_val",  "bootargs\_end": 1  },  {  "name": "prov\_app",  "version": "1.0",  "start\_sector": 208,  "bootargs\_start": 1,  "ssid": "Innophasetest",  "passphrase": "123456789",  "custparam": "inno",  "new\_boot\_param\_key": "new\_boot\_param\_val",  "bootargs\_end": 1  }  ],  "provisioned": 1,  "baudrate": 2560000,  "timeout": 0,  "verbose": 1  }  [PROV]prov\_conf\_data\_apply:355  app\_prov\_cb : status = 1  Y-BOOT 208ef13 2019-07-22 12:26:54 -0500 790da1-b-7  ROM yoda-h0-rom-16-0-gd5a8e586  FLASH:PNWWWWWWAEBuild $Id: git-a042e9a42 $  secured\_prov=1 pass\_key=0  Application Information:  ------------------------  Name : Prov App  Version : 1.0  Build Date : Apr 24 2023  Build Time : 20:57:23  Heap Available: 323 KB (331672 Bytes)  [APP]Bootparams :  --------------------  ssid= <null>  passphrase= <null>  secured\_prov =1  pass\_key=0  [APP]Bootparams end here....  [PROV]prov\_is\_provisioned : val = 1  [PROV]prov\_is\_provisioned : val = 1  ssid = dlink825\_iop  passphrase = InnoQA2023$  [APP] dlink825\_iop InnoQA2023$addr e0:69:3a:00:16:cc  network profile created for ssid: dlink825\_iop  [1.356,659] Trying to connect in 1 seconds  [2.637,148] Trying to connect in 2 seconds  [4.940,709] Trying to connect in 4 seconds  [10.444,878] CONNECT:34:0a:33:91:bd:30 Channel:1 rssi:-32 dBm  [11.800,951] MYIP 192.168.1.214  [11.801,114] IPv6 [fe80::e269:3aff:fe00:16cc]-link  [APP]Success: Connected to dlink825\_iop N/w  [APP]------ Program Exit------------- |

**secured\_prov=1,pass\_key=1**

|  |
| --- |
| prov\_event\_thread\_entry[130.106,485] BT connect[0]: ia:57:34:ac:96:d3:11 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  Creating wcm interfaceaddr e0:69:3a:00:16:cc  [PROV]199 : len = 11  [PROV] WiFi scan: Found 10 networks  [PROV]227 : wc.scan\_rslts\_len = 79  [PROV]227 : wc.scan\_rslts\_len = 137  [PROV]227 : wc.scan\_rslts\_len = 190  [PROV]227 : wc.scan\_rslts\_len = 255  [PROV]227 : wc.scan\_rslts\_len = 317  [PROV]227 : wc.scan\_rslts\_len = 378  [PROV]227 : wc.scan\_rslts\_len = 442  [PROV]227 : wc.scan\_rslts\_len = 506  [PROV]227 : wc.scan\_rslts\_len = 571  [PROV]227 : wc.scan\_rslts\_len = 630  [PROV]233 : len = 632Passkey (to be entered on remote device): 455291  [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=632  [PROV]prov\_wifi\_scan\_rslts\_send: offset = 0 remaining\_bytes = 632, copy\_len = 18  [PROV] prov\_wifi\_scan\_rslts\_send : wc.index = 18  [PROV]prov\_wifi\_scan\_rslts\_send: index = 18, scan\_rslts\_len = 632  Sending  id":"ct\_ds","bssid":"1  [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=632  [PROV]prov\_wifi\_scan\_rslts\_send: offset = 110 remaining\_bytes = 526, copy\_len = 22  [PROV] prov\_wifi\_scan\_rslts\_send : wc.index = 128  [PROV]prov\_wifi\_scan\_rslts\_send: index = 128, scan\_rslts\_len = 632  Sending  c:b7:2c:7c:ee:f4","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=632  [PROV]prov\_wifi\_scan\_rslts\_send: offset = 132 remaining\_bytes = 504, copy\_len = 22  [PROV] prov\_wifi\_scan\_rslts\_send : wc.index = 150  [PROV]prov\_wifi\_scan\_rslts\_send: index = 150, scan\_rslts\_len = 632  Sending  nnel":10},{"ssid":"","  [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=632  [PROV]prov\_wifi\_scan\_rslts\_send: offset = 154 remaining\_bytes = 482, copy\_len = 22  [PROV] prov\_wifi\_scan\_rslts\_send : wc.index = 172  [PROV]prov\_wifi\_scan\_rslts\_send: index = 172, scan\_rslts\_len = 632  Sending  bssid":"7c:10:c9:02:f2  [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=632  [PROV]prov\_wifi\_scan\_rslts\_send: offset = 176 remaining\_bytes = 460, copy\_len = 22  [PROV] prov\_wifi\_scan\_rslts\_send : wc.index = 194  [PROV]prov\_wifi\_scan\_rslts\_send: index = 194, scan\_rslts\_len = 632  Sending  id":"tplink\_A91","bssi  [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=632  [PROV]prov\_wifi\_scan\_rslts\_send: offset = 286 remaining\_bytes = 350, copy\_len = 22  [PROV] prov\_wifi\_scan\_rslts\_send : wc.index = 304  [PROV]prov\_wifi\_scan\_rslts\_send: index = 304, scan\_rslts\_len = 632  Sending  d":"60:32:b1:33:b5:7b"  [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=632  [PROV]prov\_wifi\_scan\_rslts\_send: offset = 308 remaining\_bytes = 328, copy\_len = 22  [PROV] prov\_wifi\_scan\_rslts\_send : wc.index = 326  [PROV]prov\_wifi\_scan\_rslts\_send: index = 326, scan\_rslts\_len = 632  Sending  ,"channel":6},{"ssid":  [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=632  [PROV]prov\_wifi\_scan\_rslts\_send: offset = 330 remaining\_bytes = 306, copy\_len = 22  [PROV] prov\_wifi\_scan\_rslts\_send : wc.index = 348  [PROV]prov\_wifi\_scan\_rslts\_send: index = 348, scan\_rslts\_len = 632  Sending  "low\_rssi","bssid":"74  [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=632  [PROV]prov\_wifi\_scan\_rslts\_send: offset = 352 remaining\_bytes = 284, copy\_len = 22  [PROV] prov\_wifi\_scan\_rslts\_send : wc.index = 370  [PROV]prov\_wifi\_scan\_rslts\_send: index = 370, scan\_rslts\_len = 632  Sending  :da:88:a6:9c:ea","chan  [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=632  [PROV]prov\_wifi\_scan\_rslts\_send: offset = 374 remaining\_bytes = 262, copy\_len = 22  [PROV] prov\_wifi\_scan\_rslts\_send : wc.index = 392  [PROV]prov\_wifi\_scan\_rslts\_send: index = 392, scan\_rslts\_len = 632  Sending  nel":11},{"ssid":"dlin  [PROV] prov\_srv\_fn\_wifi\_scan\_rslts : length = 22  [PROV] prov\_srv\_fn\_wifi\_scan\_rslts : offset = 396  [PROV] prov\_wifi\_scan\_rslts\_send : length = 22, index=392 , rslt\_len=632  [PROV]prov\_wifi\_scan\_rslts\_send: offset = 396 remaining\_bytes = 240, copy\_len = 22  [PROV] prov\_wifi\_scan\_rslts\_send : wc.index = 414  [PROV]prov\_wifi\_scan\_rslts\_send: index = 414, scan\_rslts\_len = 632  Sending  k650\_iop","bssid":"e0:  [PROV] prov\_srv\_fn\_wifi\_scan\_rslts : length = 22  [PROV] prov\_srv\_fn\_wifi\_scan\_rslts : offset = 418  [PROV] prov\_wifi\_scan\_rslts\_send : length = 22, index=414 , rslt\_len=632  [PROV]prov\_wifi\_scan\_rslts\_send: offset = 418 remaining\_bytes = 218, copy\_len = 22  [PROV] prov\_wifi\_scan\_rslts\_send : wc.index = 436  [PROV]prov\_wifi\_scan\_rslts\_send: index = 436, scan\_rslts\_len = 632  Sending  1c:fc:a3:5b:04","chann  [PROV] prov\_srv\_fn\_wifi\_scan\_rslts : length = 22  [PROV] prov\_srv\_fn\_wifi\_scan\_rslts : offset = 440  [PROV] prov\_wifi\_scan\_rslts\_send : length = 22, index=436 , rslt\_len=632  [PROV]prov\_wifi\_scan\_rslts\_send: offset = 440 remaining\_bytes = 196, copy\_len = 22  [PROV] prov\_wifi\_scan\_rslts\_send : wc.index = 458  [PROV]prov\_wifi\_scan\_rslts\_send: index = 458, scan\_rslts\_len = 632  Sending  el":1},{"ssid":"ASUS\_O  [PROV] prov\_srv\_fn\_wifi\_scan\_rslts : length = 22  [PROV] prov\_srv\_fn\_wifi\_scan\_rslts : offset = 462  [PROV] prov\_wifi\_scan\_rslts\_send : length = 22, index=458 , rslt\_len=632  [PROV]prov\_wifi\_scan\_rslts\_send: offset = 462 remaining\_bytes = 174, copy\_len = 22  [PROV] prov\_wifi\_scan\_rslts\_send : wc.index = 480  [PROV]prov\_wifi\_scan\_rslts\_send: index = 480, scan\_rslts\_len = 632  Sending  utside","bssid":"58:11  [PROV] prov\_srv\_fn\_wifi\_scan\_rslts : length = 22  [PROV] prov\_srv\_fn\_wifi\_scan\_rslts : offset = 484  [PROV] prov\_wifi\_scan\_rslts\_send : length = 22, index=480 , rslt\_len=632  [PROV]prov\_wifi\_scan\_rslts\_send: offset = 484 remaining\_bytes = 152, copy\_len = 22  [PROV] prov\_wifi\_scan\_rslts\_send : wc.index = 502  [PROV]prov\_wifi\_scan\_rslts\_send: index = 502, scan\_rslts\_len = 632  Sending  :22:71:ee:10","channel  [PROV] prov\_srv\_fn\_wifi\_scan\_rslts : length = 22  [PROV] prov\_srv\_fn\_wifi\_scan\_rslts : offset = 506  [PROV] prov\_wifi\_scan\_rslts\_send : length = 22, index=502 , rslt\_len=632  [PROV]prov\_wifi\_scan\_rslts\_send: offset = 506 remaining\_bytes = 130, copy\_len = 22  [PROV] prov\_wifi\_scan\_rslts\_send : wc.index = 524  [PROV]prov\_wifi\_scan\_rslts\_send: index = 524, scan\_rslts\_len = 632  Sending  ":1},{"ssid":"asusax55  [PROV] prov\_srv\_fn\_wifi\_scan\_rslts : length = 22  [PROV] prov\_srv\_fn\_wifi\_scan\_rslts : offset = 528  [PROV] prov\_wifi\_scan\_rslts\_send : length = 22, index=524 , rslt\_len=632  [PROV]prov\_wifi\_scan\_rslts\_send: offset = 528 remaining\_bytes = 108, copy\_len = 22  [PROV] prov\_wifi\_scan\_rslts\_send : wc.index = 546  [PROV]prov\_wifi\_scan\_rslts\_send: index = 546, scan\_rslts\_len = 632  Sending  u\_iop","bssid":"04:42:  [PROV] prov\_srv\_fn\_wifi\_scan\_rslts : length = 22  [PROV] prov\_srv\_fn\_wifi\_scan\_rslts : offset = 550  [PROV] prov\_wifi\_scan\_rslts\_send : length = 22, index=546 , rslt\_len=632  [PROV]prov\_wifi\_scan\_rslts\_send: offset = 550 remaining\_bytes = 86, copy\_len = 22  [PROV] prov\_wifi\_scan\_rslts\_send : wc.index = 568  [PROV]prov\_wifi\_scan\_rslts\_send: index = 568, scan\_rslts\_len = 632  Sending  1a:bd:6e:08","channel"  [PROV] prov\_srv\_fn\_wifi\_scan\_rslts : length = 22  [PROV] prov\_srv\_fn\_wifi\_scan\_rslts : offset = 572  [PROV] prov\_wifi\_scan\_rslts\_send : length = 22, index=568 , rslt\_len=632  [PROV]prov\_wifi\_scan\_rslts\_send: offset = 572 remaining\_bytes = 64, copy\_len = 22  [PROV] prov\_wifi\_scan\_rslts\_send : wc.index = 590  [PROV]prov\_wifi\_scan\_rslts\_send: index = 590, scan\_rslts\_len = 632  Sending  :1},{"ssid":"Asus777",  [PROV] prov\_srv\_fn\_wifi\_scan\_rslts : length = 22  [PROV] prov\_srv\_fn\_wifi\_scan\_rslts : offset = 594  [PROV] prov\_wifi\_scan\_rslts\_send : length = 22, index=590 , rslt\_len=632  [PROV]prov\_wifi\_scan\_rslts\_send: offset = 594 remaining\_bytes = 42, copy\_len = 6  [PROV] prov\_wifi\_scan\_rslts\_send : wc.index = 596  [PROV]prov\_wifi\_scan\_rslts\_send: index = 596, scan\_rslts\_len = 632  Sending  "channel":6}]}  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 = 253  Authentication succeeded.  [PROV]prov\_srv\_fn\_cfg\_ssid:10 tplink\_A91  [PROV]prov\_conf\_update: key = ssid, val = tplink\_A91  [PROV]prov\_conf\_update: prov\_json\_object\_update ret = 0  prov\_srv\_fn\_cfg\_ssid : 1 0  [PROV]prov\_conf\_update: key = passphrase, val = InnoQA2023$  [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 = tplink\_A91  passphrase = InnoQA2023$  [PROV]Checking connectivity to...  ssid = tplink\_A91 passphrase = InnoQA2023$  network profile created for ssid: tplink\_A91  [196.123,714] CONNECT:60:32:b1:33:b5:7b Channel:6 rssi:-35 dBm  [198.756,859] MYIP 192.168.1.15  [198.757,022] IPv6 [fe80::e269:3aff:fe00:16cc]-link  [198.759,544] 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:347  [PROV]prov\_conf\_data\_apply:350  {  "image": [  {  "name": "prov\_app",  "version": "1.0",  "start\_sector": 32,  "bootargs\_start": 1,  "ssid": "tplink\_A91",  "passphrase": "InnoQA2023$",  "custparam": "inno",  "new\_boot\_param\_key": "new\_boot\_param\_val",  "bootargs\_end": 1  },  {  "name": "prov\_app",  "version": "1.0",  "start\_sector": 208,  "bootargs\_start": 1,  "ssid": "Innophasetest",  "passphrase": "123456789",  "custparam": "inno",  "new\_boot\_param\_key": "new\_boot\_param\_val",  "bootargs\_end": 1  }  ],  "provisioned": 1,  "baudrate": 2560000,  "timeout": 0,  "verbose": 1  }  [PROV]prov\_conf\_data\_apply:355  app\_prov\_cb : status = 1  Y-BOOT 208ef13 2019-07-22 12:26:54 -0500 790da1-b-7  ROM yoda-h0-rom-16-0-gd5a8e586  FLASH:PNWWWWWWAEBuild $Id: git-a042e9a42 $  secured\_prov=1 pass\_key=1  Application Information:  ------------------------  Name : Prov App  Version : 1.0  Build Date : Apr 24 2023  Build Time : 20:57:23  Heap Available: 323 KB (331672 Bytes)  [APP]Bootparams :  --------------------  ssid= <null>  passphrase= <null>  secured\_prov =1  pass\_key=1  [APP]Bootparams end here....  [PROV]prov\_is\_provisioned : val = 1  [PROV]prov\_is\_provisioned : val = 1  ssid = tplink\_A91  passphrase = InnoQA2023$  [APP] tplink\_A91 InnoQA2023$addr e0:69:3a:00:16:cc  network profile created for ssid: tplink\_A91  [0.930,675] CONNECT:60:32:b1:33:b5:7b Channel:6 rssi:-30 dBm  [1.007,676] MYIP 192.168.1.15  [1.007,841] IPv6 [fe80::e269:3aff:fe00:16cc]-link  [APP]Success: Connected to tplink\_A91 N/w  [APP]------ Program Exit------------- |

**Errors**:

1. Password length is less than 8 characters:

As shown in Figure 16, if the password length is less than 8 characters, an error message pops-up with the following message: Passphrase should be >= 8 characters or EMPTY for open security.

Graphical user interface, text, application, email

Description automatically generated

Figure : Incorrect or minimum length or open security password

1. Wrong password:

If user entered passphrase for secure connection is incorrect, a failure message on the Talaria TWO console screen and an alert message on the mobile app indicating the same is displayed. At this stage the user is required to reset the board or re-flash the application.



Figure : Alert message in case of a wrong passphrase