



Bluetooth + X

Version: 20240613

Contents

1 Overview	2
2 Data structure	3
2.1 tuya_ble_ext_module_active_data_t	3
3 API description	4
3.1 Macro configuration	4
3.2 Report extension module information	4
3.3 Callback for querying extension module information	5
3.4 Callback for receiving activation information	8
4 How to use	11
4.1 Sequence diagram	11
4.2 Callback event processing	11
4.3 Code files	12
5 Things to note	13
6 Support and help	14

This topic describes the Bluetooth + X capability.

1 Overview

A typical Tuya-enabled Bluetooth device communicates with the cloud solely through the Bluetooth channel.

With the Bluetooth + X capability enabled, where X is an extension module like Wi-Fi, Cat.1, or NB-IoT, the device continues to use Bluetooth for pairing and control when not connected to an X module. When connected to an X module, the device can communicate with the cloud either over Bluetooth or through the X channel once it activates the X module via Bluetooth.

2 Data structure

2.1 tuya_ble_ext_module_active_data_t

```
1 typedef struct{
2     uint8_t *p_data;    /**< ext module active data. */
3     uint16_t data_len;  /**< active data length. */
4 } tuya_ble_ext_module_active_data_t;
```

- `p_data`: The pointer to the information used to activate the extension module.
- `data_len`: The length of the information used to activate the extension module.

3 API description

3.1 Macro configuration

Macro definition	TUYA_BLE_FEATURE_EXT_MODULE_ENABLE
Dependency	None
Description	Enable Bluetooth + X:
	<pre>#define TUYA_BLE_FEATURE_EXT_MODULE_ENABLE (1)</pre>
	Disable Bluetooth + X:
	<pre>#define TUYA_BLE_FEATURE_EXT_MODULE_ENABLE (0)</pre>
Notes	

3.2 Report extension module information

Function prototype	<pre>tuya_ble_status_t tuya_ble_ext_module_info_asynchronous_response (uint8_t *p_data, uint32_t len);</pre>
Description	Respond with the extension module information asynchronously or report it proactively.
Parameter description	<code>p_data[in]</code>: The pointer to the extension module information. <code>len[in]</code>: The length of the extension module information.
Return value	TUYA_BLE_SUCCESS: Success. TUYA_BLE_ERR_INVALID_PARAM: Invalid parameter. TUYA_BLE_ERR_NO_MEM: Failed to request memory allocation. TUYA_BLE_ERR_NO_EVENT: Other errors.

Function prototype	<code>tuya_ble_status_t</code>
	<code>tuya_ble_ext_module_info_asynchronous_response</code> (<code>uint8_t *p_data</code> , <code>uint32_t len</code>);
Notes	When the mobile app requests extension module information, the Bluetooth device can respond with the corresponding information. Or, the Bluetooth device proactively reports extension module information when changes occur. See the data format description below.

3.3 Callback for querying extension module information

Callback event	<code>TUYA_BLE_CB_EVT_QUERY_EXT_MODULE_DEV_INFO</code>
Data structure	None.
Description	The mobile app requests extension module information.
Notes	The Bluetooth device assembles data in the specified format and sends it to the mobile app by reporting extension module information .

Format for extension module information:

<code>inc_em_info</code>	<code>emt_type</code>	<code>N_TLD</code>	<code>TLD1</code>	...	<code>TLDn</code>
1 byte	1 byte	1 byte	1 + 2 + n bytes	...	1 + 2 + n bytes

- `inc_em_info`: Specify whether extension module information is included. `0x01`: Included. `0x00`: Not included.
- `emt_type`: The type of the extension module.

- 0x01: NB-IoT module
- 0x02: Wi-Fi module
- 0x03: Cat.1 module
- 0x04: Zigbee module
- N_TLD: The number of TLD data units.
- TLD: The data unit.
 - Type (1 byte): The type.
 - Length (2 bytes): The length of the data.
 - Data: The payload. See the following table.
- TLD 0x01 for extension module information

Type	Length	Data
0x01	...	When emt_type is NB-IoT module, the device information in JSON format is <pre>{ "pv" : "3.0" , "bv" : "3.0" , "sv" : "1.0.0" , "i" : "866242050005256" , "opt" :1, "h" :28800 }</pre> pv: The protocol version, in string. bv: The baseline version, in string. sv: The firmware version, in string. i: The IMEI of the device, in string. h: The heartbeat time, in integer. Unit: seconds. opt: The APN, in integer. opt is used to map the carrier. 0: China Telecom. 0: China Unicom. 1: China Mobile. 2: Directly connected to the Tuya Cloud.

TypeLengthData

When emt_type is Wi-Fi module, there is no need to include device information

because the module will report it upon activation.

When emt_type is Cat.1 module, there is no need to include device information

because the module will report it upon activation.

- TLD 0x02 for the presence status of the extension module

Type	Length	Data
------	--------	------

0x02	0x01	0x00: The extension module is ejected .
------	------	--

		0x01: The extension module is installed .
--	--	--

- TLD 0x03 for channel priority

TypeLengthData

0x03	...	A priority is represented by one byte, with the order indicating priority from high to low. For example, if the priority is defined as Bluetooth Low Energy > MQTT, the data will be 0x03, 0x01.
------	-----	--

0x00: LAN

0x01: MQTT

0x02: HTTP

0x03: Bluetooth Low Energy

0x04: Bluetooth mesh

0x05: Tuya proprietary mesh

0x06: Beacon

- TLD 0x04 for OTA priority (**not used**)
- TLD 0x05 for the binding status of the extension module

Type	Length	Data
0x05	0x01	0x00: The extension module is not bound . 0x01: The extension module is bound .

- TLD 0x06 for OTA channel and version (**not used**)
- TLD 0x07 for the flag of passive/proactive data reporting

Type	Length	Data
0x0	0x01	0x00: The device reports data passively in response to a status query. 0x01: The device proactively reports data. The mobile app does not acknowledge the receipt of proactive reporting.

When the extension module is Wi-Fi or Cat.1, its information will be reported upon activation. The Bluetooth device synchronizes with the mobile app regarding the presence of the extension module, binding status, and channel priority.

See [tal_feature_ext_module](#) and [tal_ble_x_demo.c](#) for examples of the extension module information returned.

3.4 Callback for receiving activation information

TUYA_BLE_CB_EVT_EXT_MODULE_ACTIVE_INFO_RECEIVED

Callback event

Data structure

tuya_ble_ext_module_active_data_t

Description

The device receives the activation information for the extension module from the mobile app.

The data format for the activation information:

emt_type	TLD
1 byte	1 + 2 + n bytes

- **emt_type**: The type of the extension module.
 - 0x01: NB-IoT module
 - 0x02: Wi-Fi module
 - 0x03: Cat.1 module
 - 0x04: Zigbee module
- TLD: The data unit.
 - Type (1 byte): The type.
 - Length (2 bytes): The length of the data.
 - Data: The payload. See the following table.
- When **emt_type** is **NB-IoT module**, the TLD definition is:

Type	Length	Data
0x01	...	A JSON string, including devID and secKey : <pre>{ "s" : "nYqC0z6yuqJzrggALzatsB42K0PCz****" }</pre>

- When **emt_type** is **Wi-Fi or Cat.1 module**, the TLD definition is:

TypeLengthData

0x01	...	A JSON string, including devID , secKey , and localKey , which matches the one stored in the cloud and is passed in raw format through the mobile app and Bluetooth: <pre>{ "devId" : "6cba0dda87e9ae35f8c****" , "region" : "AY" , </pre>
------	-----	--

Type	Length	Data
		<pre> “env” : “pr_0” , “localKey” : “a427f948b8451***” } </pre>

- When `emt_type` is **Zigbee module**, the TLD definition is:

Type	Length	Data
0x01	...	<code>PAN</code>: 2 bytes. <code>exPAN</code>: 8 bytes. <code>Channel</code>: 1 byte. <code>net key</code>: 16 bytes.

4 How to use

4.1 Sequence diagram

```
1  %%{init: { "sequence": { "wrap": true} } }%%
2  sequenceDiagram
3      participant A as Application
4      participant X as X Module
5      participant T as Tuya Bluetooth LE SDK
6      participant M as Mobile App
7      participant C as Cloud
8      Note over A, C: Connected to Tuya Bluetooth LE
9      M ->> T: Request device information.
10     T ->> A: TUYA_BLE_CB_EVT_QUERY_EXT_MODULE_DEV_INFO
11     Note over A, X: Get the device information <br>from the X module.
12     A ->> T: tuyabluetooth_ext_module_info_asynchronous_response()
13     T ->> M: Return device information.
14     opt X module exists.
15         M ->> C: Report device information.
16         C ->> M: Return activation information.
17         M ->> T: Send activation information.
18         T ->> A: TUYA_BLE_CB_EVT_EXT_MODULE<br>_ACTIVE_INFO_RECEIVED
19         T ->> M: Respond to activation information.
20         Note over A, X: Send activation information <br>to X module.
21         X ->> M: Activation succeeds.
22     end
```

4.2 Callback event processing

```

1  STATIC VOID_T tuya_ble_protocol_callback(tuya_ble_cb_evt_param_t* ev
2  ent)
3  {
4      switch(event->evt)
5      {
6          //...
7  #if defined(TUYA_BLE_FEATURE_EXT_MODULE_ENABLE) && (TUYA_BLE_FEATURE
8  _EXT_MODULE_ENABLE != 0)
9      case TUYA_BLE_CB_EVT_QUERY_EXT_MODULE_DEV_INFO: {
10         //...
11     } break;
12     case TUYA_BLE_CB_EVT_EXT_MODULE_ACTIVE_INFO_RECEIVED: {
13         //...
14     } break;
15 #endif
16     //...
17 }
18 }

```

4.3 Code files

```

1  tal_feature_ext_module
2      include
3      tal_ble_x_demo.h
4      tal_feature_ext_module.h
5      src
6      tal_ble_x_demo.c
7      tal_feature_ext_module.c

```

5 Things to note

- To support the full Bluetooth + X feature, you need to implement the required capabilities on both the Bluetooth device and the X module. Before using the full Bluetooth + X feature, verify with the product manager or developer if the X module is compatible.
- The Bluetooth + X capability allows for activating the X module via Bluetooth. This covers neither the activation process between the Bluetooth and X modules nor data exchange for DP or OTA through the X channel. To implement these features, you need to add the code for Bluetooth and X module interaction in your project.

6 Support and help

If you have any problems with TuyaOS development, you can post your questions in the [Tuya Developer Forum](#).