AN1405: Bluetooth Mesh on Advertising Extensions

This document describes the Bluetooth Mesh Advertising Extensions feature. The non-standard Bluetooth Mesh modification achieves better performance by increasing the network packet size. It does so by utilizing the Bluetooth 5 Advertising Extensions, which allows sending much larger advertisement packets.

KEY POINTS

- Faster Device Firmware Update over Advertising Extensions
- Not dependent on specific vendor models
- Not a standard Bluetooth Mesh feature
- If used, the code is not in compliance with Bluetooth Mesh 1.1

1 Introduction

1.1 Mesh Bearers

The Mesh Profile specification defines two mesh bearers over which mesh messages may be transported:

- An advertising bearer
- A GATT bearer

When using the advertising bearer, a mesh packet is sent in the advertising data of a Bluetooth Low Energy advertising protocol data unit (PDU) that uses the Mesh Message AD Type identified by «Mesh Message». Any advertisement that uses the Mesh Message AD Type is sent using non-connectable and non-scannable undirected advertising events. If a node receives a Mesh Message AD Type in a connectable advertisement or scannable advertising event, the message is ignored.

The GATT bearer is provided to enable devices that are not capable of supporting the advertising bearer to participate in a mesh network. The GATT bearer is used to transmit and receive PDUs between two devices over a GATT connection.

1.2 LE Advertising Extensions

The LE Advertising Extensions feature in the Bluetooth Core Specification introduces new PDUs, including the ADV_EXT_IND, AUX_ADV_IND, AUX_SYNC_IND, and AUX_CHAIN_IND PDUs. These PDUs share the same Advertising Physical Channel PDU payload format referred to as the “Common Extended Advertising Payload Format”.

1.3 Extended Advertising

The LE Advertising Extensions feature allows a device to transmit up to 1,650 octets of Host advertising data over secondary advertising channels, offloading data that would otherwise be transmitted on the primary advertising channels by splitting this data into several legacy advertising events. The advertising packet on the primary advertising channel (that is, the ADV_EXT_IND) contains LE PHY, Channel Index and the offset to the start time of the auxiliary packet.

The secondary advertising physical channel can use any LE PHY. All advertising packets on the secondary advertising physical channel in the same extended advertising event use the same PHY, which is specified in the advertising packet on the primary advertising physical channel.

1.4 Current Limitation

Currently, a mesh message cannot exceed 236 octets. While the models defined in the Mesh Model specification are designed to be as efficient as possible and to provide a large amount of functionality with very small payloads, use cases exist (such as device firmware update, sensor calibration data transfer, and diagnostic data collection) that require much larger payloads to be transmitted over the mesh network.

The current packet size of 236 bytes was chosen because of the limits in BGStack. Additional tests are still needed with longer packets though, to remove any barriers that may remain in the mesh stack. The packet size can be adjusted between 29 and 236 bytes, 29 being the minimal/legacy packet size. In case of 29 bytes, the Stack ignores the AE messages, but the Link Layer still processes them.

In crowded environments (radio-wise), activating the Advertising Extensions could slow down the scanning, while the radio is following the AE packet pointers.

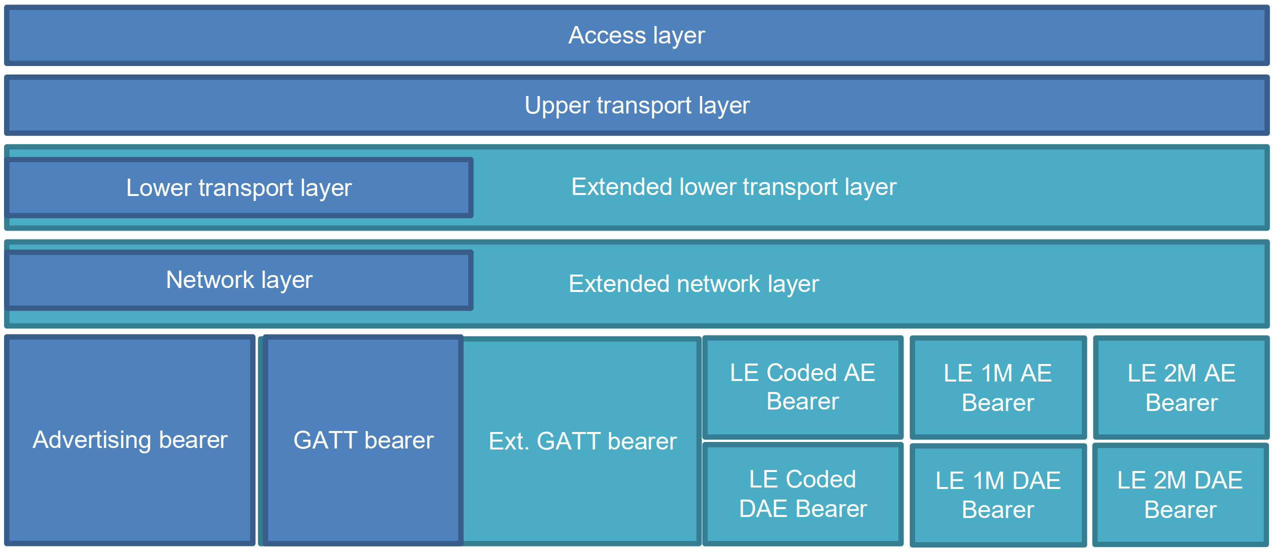
1.5 Usage

Utilizing the Advertising Extensions in a Bluetooth mesh network would be the wisest for BLOB data types, which are about 90% of Device Firmware Update packets, so this feature is specifically directed to OTA.

2 Architecture

This specification maintains the same layered architecture defined in the Mesh Profile specification and introduces the following extensions:

- New bearers and network interface behaviors
- New GATT behavior
- Extended Network PDU format and behavior
- Extended lower transport segmentation



The new extensions introduced by this specification are marked as green.

3 Bluetooth Mesh Device Firmware Update over AE

The idea is to distribute the firmware update image more quickly by utilizing the Advertising Extension of Bluetooth LE. To demonstrate this feature, Silicon Labs provides a Python application which, in collaboration with an NCP flashed WSTK, demonstrates most of the capabilities of the Silicon Labs Bluetooth mesh SDK.

3.1 btmesh_host_dfu

This software implements a wide variety of functions, such as:

- Connection to and configuring the NCP node as a Provisioner
- Scanning and listing nearby nodes
- Provisioning nodes, creating Groups, renaming
- Factory resetting the NCP node and the script database
- Configuring AE and DFU parameters
- Assigning a Distributor, uploading, deleting and querying the firmware list and capabilities
- Spreading the new Firmware in the network from the Distributor
- Argument based or interactive command execution, detailed and interactive help

You may find the script in your GSDK folder: <SDK_DIR>/app/btmesh/example_host/btmesh_host_dfu/btmesh_host_dfu.py

To launch the script and connect to the NCP node, you must provide one of two connection arguments, either `'--ip'` or `'--usb'`. The SDK API XML can be referenced with `'--xapi'`, and the configuration file with `'--cfg'` (the script stores a template of the default configuration values, so, if the file does not exist, it will be created with those values inserted in it). To start the script in interactive mode, use the argument `'--interactive'`.

Command format:

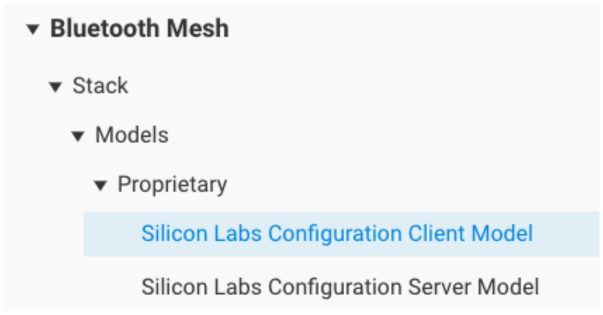
```
dist upload -d *Dist* -f 0x02ff:s:ultimatefw -m s:1234 app.gbl
```

Upload the `'app.gbl'` file to the Node with the name `'Dist'` in it, and with the firmware identifier assembled from Silicon Labs' vendor_id and the text `'ultimatefw'` as a string (metadata is also provided as a string but is invalid in this case).

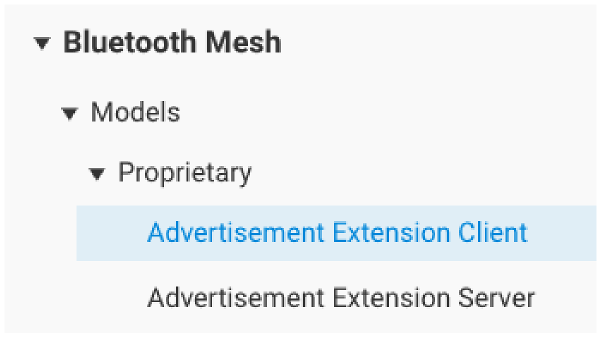
4 Silicon Labs' Solution

Our solution provides multiple new components for different layers and different requirements. These components must be installed from the Simplicity Studio Project Configurator in order to utilize the feature.

To utilize the functionality on the Stack Level, install the following component for a Client or Server implementation, respectively.

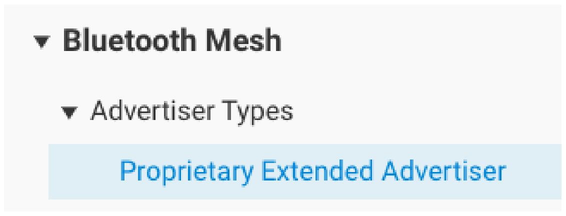

Bluetooth Mesh
Stack
Models
Proprietary
Silicon Labs Configuration Client Model
Silicon Labs Configuration Server Model

To utilize the functionality on the SDK Level, install the following components for a Client or Server implementation, respectively.


Bluetooth Mesh
Models
Proprietary
Advertisement Extension Client
Advertisement Extension Server

The following is the Silicon Labs Proprietary Extended Advertiser implementation, required by the SDK level components to utilize the AE feature. This component is not a dependency of the Configuration Server. In order to be able to use the Mesh over AE feature, it must be installed manually.

Note: This component deviates from the Bluetooth Mesh 1.1 specification, and its usage will result in non-compliant code.



5 Server API – Stack Level

5.1 Initialize/Deinitialize the Server Side

This is a simple API for turning on and off AE Mesh packet reception.

```
/***************************************************************************//**
 *
 * Initialize Silabs Configuration Server model
 *
 *
 * @return SL_STATUS_OK if successful. Error code otherwise.
 *
 ******************************************************************************/
sl_status_t sl_btmesh_silabs_config_server_init();

/***************************************************************************//**
 *
 * Deinitialize Silabs Configuration Server model. After this call, the model
 * cannot be used until it is initialized again. See @ref
 * sl_btmesh_silabs_config_server_init.
 *
 *
 * @return SL_STATUS_OK if successful. Error code otherwise.
 *
 ******************************************************************************/
sl_status_t sl_btmesh_silabs_config_server_deinit();
```

5.2 Set/Get the Physical Layer Used

```
/***************************************************************************//**
 *
 * Set TX PHY for the node.
 *
 * @param[in] phy TX PHY for long packets (packets that would be segmented).
 *     - 1: LE 1M PHY
 *     - 2: LE 2M PHY
 * @param[in] options TX options. NOTE: Currently not used: reserved for future
 *   use.
 *
 * @return SL_STATUS_OK if successful. Error code otherwise.
 *
 ******************************************************************************/
sl_status_t sl_btmesh_silabs_config_server_set_tx(uint8_t phy,
                                                  uint32_t options);

/***************************************************************************//**
 *
 * Get current TX PHY.
 *
 * @param[out] phy TX PHY for long packets (packets that would be segmented).
 *     - 1: LE 1M PHY
 *     - 2: LE 2M PHY
 * @param[out] options TX options. NOTE: Currently not used: reserved for future
 *   use.
 *
 * @return SL_STATUS_OK if successful. Error code otherwise.
 *
 ******************************************************************************/
sl_status_t sl_btmesh_silabs_config_server_get_tx(uint8_t *phy,
                                                  uint32_t *options);
```

5.3 Enable/Disable the Function for a Specific Vendor Model

Enables/disables the function for the Bluetooth SIG model if vendor_id equals 0xffff.

```
/***************************************************************************//**
 *
 * Enable/Disable usage of extended packets for a model.
 *
 * @param[in] elem_index Element index of model to configure.
 * @param[in] vendor_id Vendor ID of model to configure.
 * @param[in] model_id Model ID of model to configure.
 * @param[in] value Extended packet size to set, currently 1 to use extended 0
 *   to not
 *
 * @return SL_STATUS_OK if successful. Error code otherwise.
 *
 ******************************************************************************/
sl_status_t sl_btmesh_silabs_config_server_set_model_enable(uint8_t elem_index,
                                                            uint16_t vendor_id,
                                                            uint16_t model_id,
                                                            uint16_t value);

/***************************************************************************//**
 *
 * Get model settings.
 *
 * @param[in] elem_index Element index of model to get.
 * @param[in] vendor_id Vendor ID of model to get.
 * @param[in] model_id Model ID of model to get.
 * @param[out] value Extended packet size is used, currently 1 is in use, 0 is
 *   not
 *
 * @return SL_STATUS_OK if successful. Error code otherwise.
 *
 ******************************************************************************/
sl_status_t sl_btmesh_silabs_config_server_get_model_enable(uint8_t elem_index,
                                                            uint16_t vendor_id,
                                                            uint16_t model_id,
                                                            uint16_t *value);
```

5.4 Set/Get the AE Mesh Packet Maximum Size

```
/***************************************************************************//**
 *
 * Set max network pdu for the node.
 *
 * @param[in] max_size Max size of packet to be used, in range 29 - 236 bytes.
 *
 * @return SL_STATUS_OK if successful. Error code otherwise.
 *
 ******************************************************************************/
sl_status_t sl_btmesh_silabs_config_server_set_network_pdu(uint16_t max_size);

/***************************************************************************//**
 *
 * Get max network pdu in use.
 *
 * @param[out] max_size Max size of packet, in range 29 - 236 bytes.
 *
 * @return SL_STATUS_OK if successful. Error code otherwise.
 *
 ******************************************************************************/
sl_status_t sl_btmesh_silabs_config_server_get_network_pdu(uint16_t *max_size);
```

6 Client API – Stack Level

A parallel set of APIs are provided on the Client side.

6.1 Initialize/Deinitialize the Client Side

```
/***************************************************************************//**
 *
 * Initialize Silabs Configuration client model.
 *
 *
 * @return SL_STATUS_OK if successful. Error code otherwise.
 *
 ******************************************************************************/
sl_status_t sl_btmesh_silabs_config_client_init();

/***************************************************************************//**
 *
 * Deinitialize Silabs Configuration Client model. After this call, the model
 * cannot be used until it is initialized again. See @ref
 * sl_btmesh_silabs_config_client_init.
 *
 *
 * @return SL_STATUS_OK if successful. Error code otherwise.
 *
 ******************************************************************************/
sl_status_t sl_btmesh_silabs_config_client_deinit();
```

6.2 Set/Get the Physical Layer Used

```
/***************************************************************************//**
 *
 * Set node specific TX configuration.
 *
 * @param[in] server_address Destination address of the message. It can be a
 *   unicast address, a group address, or a virtual address.
 * @param[in] appkey_index The application key index used
 * @param[in] tx_phy TX PHY for long packets (packets that would be segmented).
 *     - 0: Legacy (segmented)
 *     - 1: LE 1M PHY
 *     - 2: LE 2M PHY
 * @param[in] tx_options_bitmap TX options. NOTE: Currently not used: reserved
 *   for future use, set as 0.
 *
 * @return SL_STATUS_OK if successful. Error code otherwise.
 *
 * @b Events
 *   - @ref sl_btmesh_evt_silabs_config_client_tx_status
 *
 ******************************************************************************/
sl_status_t sl_btmesh_silabs_config_client_set_tx(uint16_t server_address,
                                                  uint16_t appkey_index,
                                                  uint8_t tx_phy,
                                                  uint32_t tx_options_bitmap);

/***************************************************************************//**
 *
 * Get node specific TX configuration.
 *
 * @param[in] server_address Destination address of the message. It can be a
 *   unicast address, a group address, or a virtual address.
 * @param[in] appkey_index The application key index used
 *
 * @return SL_STATUS_OK if successful. Error code otherwise.
 *
 * @b Events
 *   - @ref sl_btmesh_evt_silabs_config_client_tx_status
 *
 ******************************************************************************/
sl_status_t sl_btmesh_silabs_config_client_get_tx(uint16_t server_address,
                                                  uint16_t appkey_index);
```

6.3 Enable/Disable the Function for a Specific Vendor Model

Enables/disables the function for the Bluetooth SIG model if vendor_id equals 0xffff.

```
/***************************************************************************//**
 *
 * Set TX over AE on per-model basis enable/disable state.
 *
 * @param[in] server_address Destination address of the message. It can be a
 *   unicast address, a group address, or a virtual address.
 * @param[in] appkey_index The application key index used
 * @param[in] elem_index The index of the server target element, 0 is the
 *   primary element.
 * @param[in] vendor_id Vendor ID of the model to configure.
 * @param[in] model_id Model ID of the server target.
 * @param[in] enable Non zero - enable, otherwise disable
 *
 * @return SL_STATUS_OK if successful. Error code otherwise.
 *
 * @b Events
 *   - @ref sl_btmesh_evt_silabs_config_client_model_status
 *
 ******************************************************************************/
sl_status_t sl_btmesh_silabs_config_client_set_model_enable(uint16_t server_address,
                                                            uint16_t appkey_index,
                                                            uint16_t elem_index,
                                                            uint16_t vendor_id,
                                                            uint16_t model_id,
                                                            uint8_t enable);

/***************************************************************************//**
 *
 * Get TX over AE on per-model basis enable/disable state.
 *
 * @param[in] server_address Destination address of the message. It can be a
 *   unicast address, a group address, or a virtual address.
 * @param[in] appkey_index The application key index used
 * @param[in] elem_index The index of the server target element, 0 is the
 *   primary element.
 * @param[in] vendor_id Vendor ID of the model to configure.
 * @param[in] model_id Model ID of the server target.
 *
 * @return SL_STATUS_OK if successful. Error code otherwise.
 *
 * @b Events
 *   - @ref sl_btmesh_evt_silabs_config_client_model_status
 *
 ******************************************************************************/
sl_status_t sl_btmesh_silabs_config_client_get_model_enable(uint16_t server_address,
                                                            uint16_t appkey_index,
                                                            uint16_t elem_index,
                                                            uint16_t vendor_id,
                                                            uint16_t model_id);
```

6.4 Set/Get the AE Mesh Packet Maximum Size

```
/***************************************************************************//**
 *
 * Set network PDU state.
 *
 * @param[in] server_address Destination address of the message. It can be a
 *   unicast address, a group address, or a virtual address.
 * @param[in] appkey_index The application key index used
 * @param[in] pdu_max_size Network PDU max size. Valid range: 29 - 398.
 *
 * @return SL_STATUS_OK if successful. Error code otherwise.
 *
 * @b Events
 *   - @ref sl_btmesh_evt_silabs_config_client_network_pdu_status
 *
 ******************************************************************************/
sl_status_t sl_btmesh_silabs_config_client_set_network_pdu(uint16_t server_address,
                                                           uint16_t appkey_index,
                                                           uint16_t pdu_max_size);

/***************************************************************************//**
 *
 * Get network PDU state.
 *
 * @param[in] server_address Destination address of the message. It can be a
 *   unicast address, a group address, or a virtual address.
 * @param[in] appkey_index The application key index used
 *
 * @return SL_STATUS_OK if successful. Error code otherwise.
 *
 * @b Events
 *   - @ref sl_btmesh_evt_silabs_config_client_network_pdu_status
 *
 ******************************************************************************/
sl_status_t sl_btmesh_silabs_config_client_get_network_pdu(uint16_t server_address,
                                                           uint16_t appkey_index);
```

7 Client API – SDK Level

The Client side provides a function for handling AE events.

```
/***************************************************************************//**
 * Handle Advertisement Extension Client events.
 *
 * This function is called automatically by Universal Configurator after
 * enabling the component.
 *
 * @param[in] evt  Pointer to incoming event.
 *
 ******************************************************************************/
void sl_btmesh_ae_client_on_event(const sl_btmesh_msg_t *const evt);
```

8 Server API – SDK Level

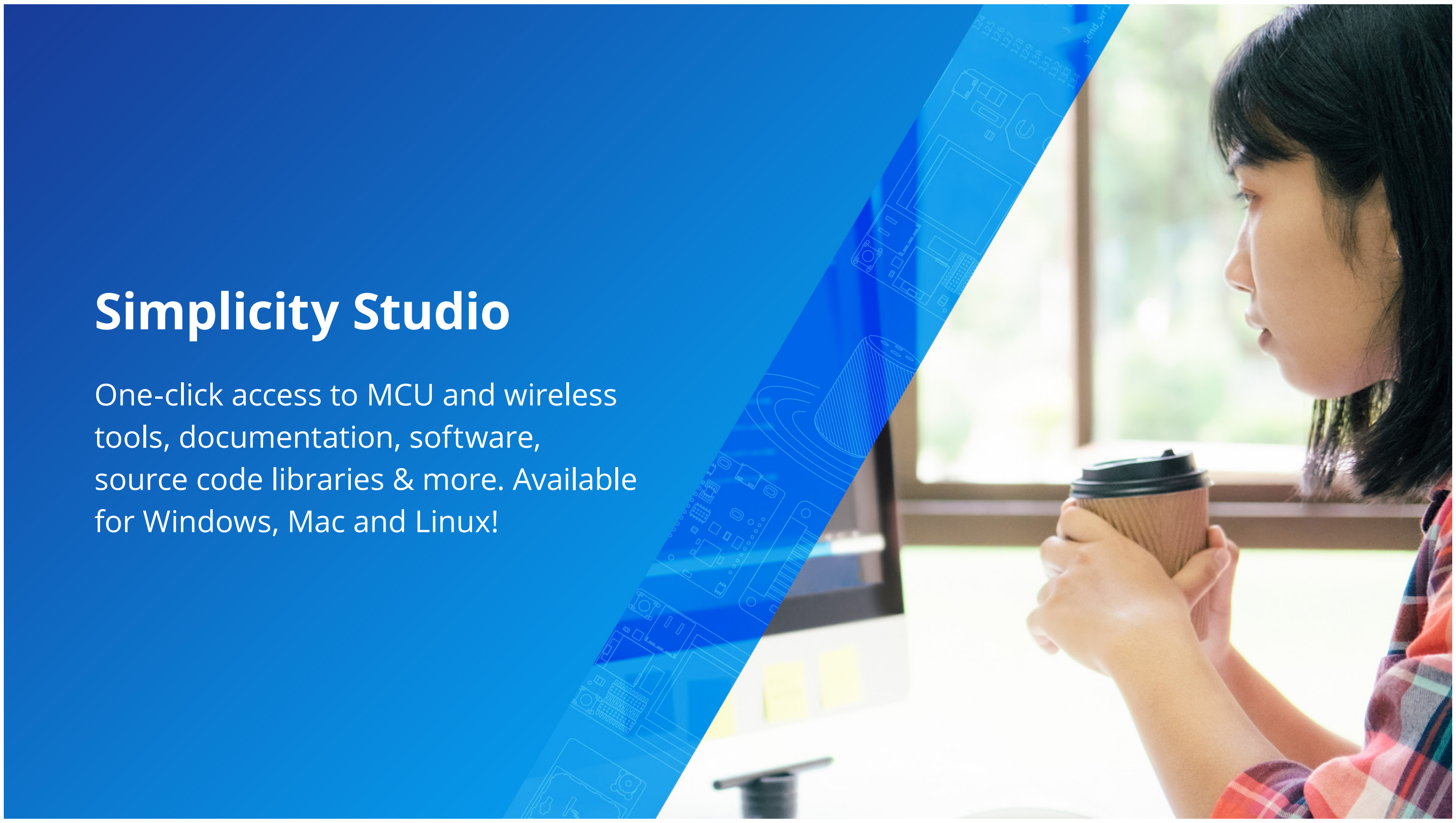
The Server side also provides a function for handling AE events.

```
/***************************************************************************//**
 * Handle Advertisement Extension Server events.
 *
 * This function is called automatically by Universal Configurator after
 * enabling the component.
 *
 * @param[in] evt  Pointer to incoming event.
 *
 ******************************************************************************/
void sl_btmesh_ae_server_on_event(const sl_btmesh_msg_t *const evt);
```

9 Mobile API

Silicon Labs provides an ADK for both iOS and Android systems. The Advertisement Extension can be enabled on all remote nodes that contain **Silabs Configuration Server** (Vendor Id: **0x02ff**, Model Id: **0xfffd**) on their primary elements. The **Client** counterpart (Vendor Id: **0x02ff**, Model Id: **0xfffc**) is required to configure a remote node. Silicon Labs BT Mesh ADK implements this client vendor model out of the box. All related configuration is done in **SBMAdvertisementExtension** class on iOS and **AdvertisementExtension** class on Android.

For further details and the API description, please see the latest version of *AN1200.1: iOS and Android ADK for Bluetooth® Mesh SDK 2.x and Higher*.


Simplicity Studio
One-click access to MCU and wireless tools, documentation, software, source code libraries & more. Available for Windows, Mac and Linux!

IoT Portfolio
www.silabs.com/IoT

SW/HW
www.silabs.com/simplicity

Quality
www.silabs.com/quality

Support & Community
www.silabs.com/community

SILICON LABS

Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
USA

www.silabs.com