UG366: *Bluetooth*® Mesh Node Configuration User's Guide

This users guide describes the DCD (Device Composition Data) Configurator and the Memory Configurator, both available in the Graphical User Interface of the Bluetooth mesh SDK.

KEY POINTS

- Modify the Device Composition Data, including device information, elements, and models.
- Set memory configuration options to optimize the RAM and persistent storage usage.

1 Introduction

Silicon Laboratories Bluetooth mesh SDK supports configuring the Device Composition Data (DCD) and the memory use of a Bluetooth mesh project.

The Device Composition Data (DCD) contains information about a Bluetooth mesh node, the elements it includes, and the supported models. DCD exposes the node information to a configuration client so that it knows the potential functionalities the node supports and based on that can configure the node.

Several memory configuration options for a Mesh project are available. Some only affect the RAM consumption, and others affect both RAM and the persistent storage. Because the space available in RAM and persistent storage is limited, developers should set the configuration option values in a suitable manner.

A graphic user interface is available in the .isc file, which is created automatically when you start a new project. The interface supports configuring the GATT database, the Device Composition Data, and the memory allocations for the project. *UG365: Bluetooth GATT Configurator User's Guide* describes how to configure the GATT database for a Bluetooth project. This document focuses on the other two parts – Device Composition Data and memory allocation. Tabs in the Bluetooth mesh Configurator allow you to change between the two, as shown in the following figure.

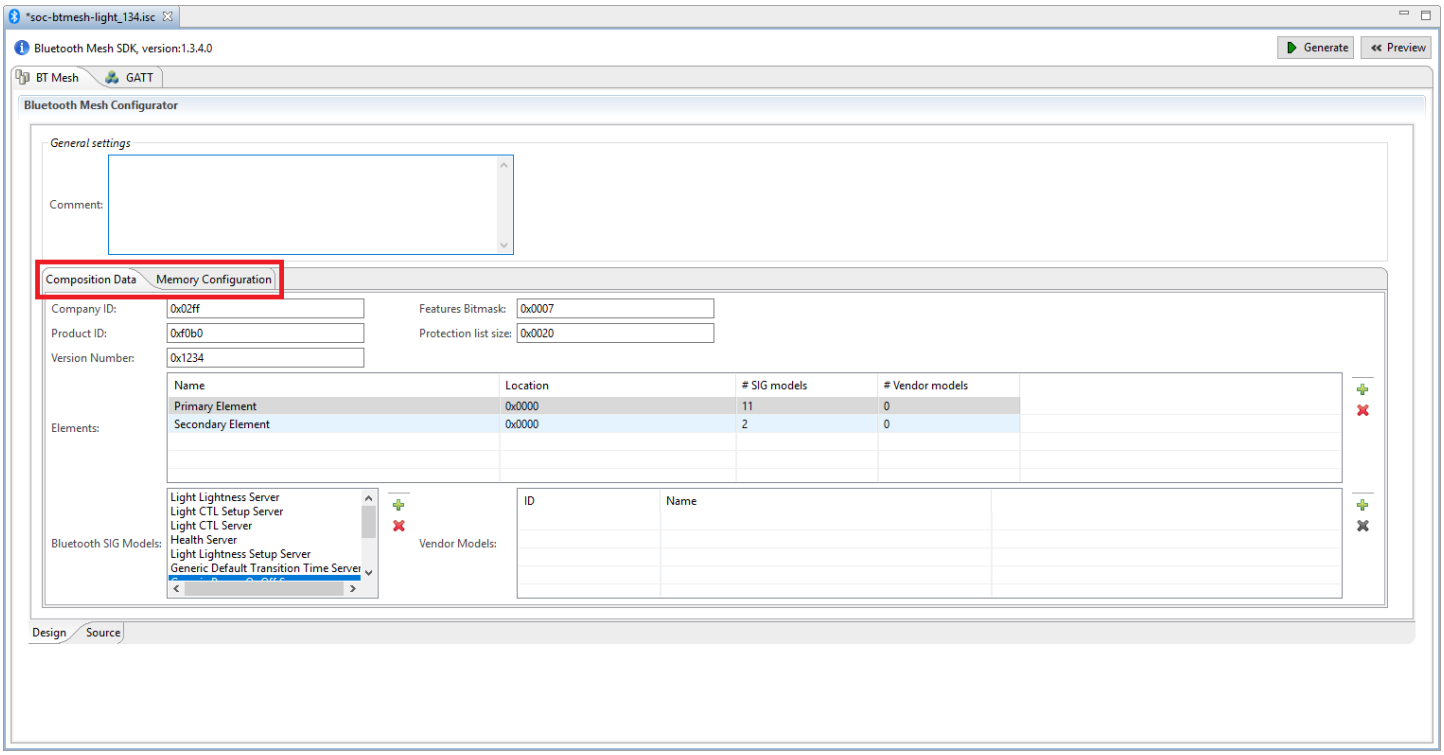


Figure 1-1. Graphic DCD and Memory Configurators

2 Device Composition Data Configurator

When you click the **Composition Data** tab, all the items shown under it compose the Device Composition Data. They are presented in three areas: device information, elements, and models.

2.1 Device Information

The device information contains five fields, shown in the following figure.

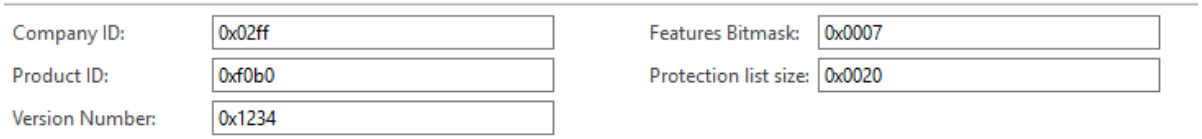


Figure 2-1. Device Information Section

The meaning of each field is shown in the following table.

Table 2-1. Device Information Fields

Field Name	Notes
Company ID	16-bit company identifier assigned by the Bluetooth SIG (available here)
Product ID	16-bit vendor-assigned product identifier, vendor-specific
Version Number	16-bit vendor-assigned product version identifier, vendor-specific
Features Bitmask	Bit field indicating the device features, as defined in the following table
Protection list size	16-bit value representing the minimum number of replay protection list entries in a device, refer to section 3.5 Rpl Size for more information

The Features Bitmask field indicates the capabilities of the node. The Feature Bitmask field format is shown in the following table.

Table 2-2. Features Bitmask Field Format

Bit	Feature	Notes
0	Relay	Relay feature support: 0 = False, 1 = True
1	Proxy	Proxy feature support: 0 = False, 1 = True
2	Friend	Friend feature support: 0 = False, 1 = True
3	Low Power	Low Power feature support: 0 = False, 1 = True
4-15	RFU	Reserved for Future Use

2.2 Elements

An element is an addressable entity within a node. Each node can have one or more elements, the first called the primary element and the others called secondary elements. Each element is assigned a unicast address during provisioning so that it can be used to identify which node is transmitting or receiving a message. The primary element is addressed using the first unicast address assigned to the node, and the secondary elements are addressed using the subsequent addresses. An elements editor, shown in the following figure, allows you to configure your project's elements. Click the green plus symbol (+) to add an element or select an element and click the red X symbol (x) to remove it.

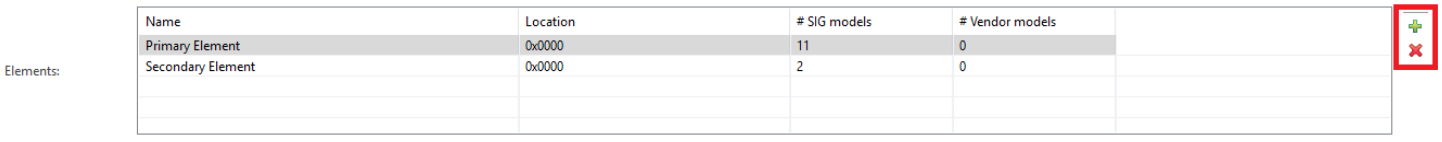


Figure 2-2. Elements Editor

2.3 Models

A model defines the basic functionality of a node, and a node may include multiple models. A model defines the required states, the messages that act upon those states, and any associated behaviors.

Models may be defined and adopted by the Bluetooth SIG and may also be defined by vendors. Models defined by the Bluetooth SIG are known as SIG-adopted models, and models defined by vendors are known as vendor models. Each model is identified by a 32-bit unique identifier, including a 16-bit vendor identifier and 16-bit model identifier. For the SIG-adopted models, the 16-bit vendor identifier uses 0xFFFF.

Model specifications are designed to be very small and self-contained. At specification definition time, a model can require other models that must also be instantiated within the same node. This is called extending, which means a model can extend other models.

Models that do not extend other models are referred to as root models. Model specifications are immutable. In other words, it is not possible to remove or add behavior to a model, whether the desired behavior is optional behavior or mandatory. Models are not versioned and have no feature bits. If additional behavior is required in a model, then a new extended model is defined that exposes the required behavior and can be implemented alongside the original model.

Therefore, knowledge of the models supported by an element determines the exact behavior exposed by that element.

The DCD configurator supports configuring both SIG-adopted models and vendor models through individual editors.

2.3.1 SIG-Adopted Model Editor

The SIG-adopted model editor is on the bottom left of the configurator. Once you click any element, the models included in the element are listed in the editor as shown in the following figure.

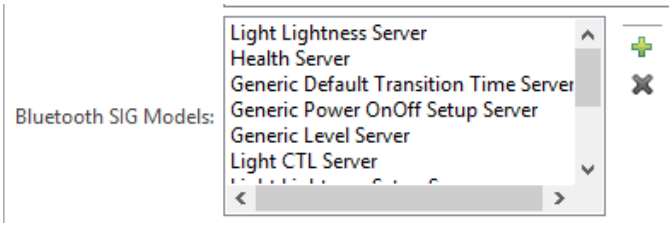


Figure 2-3. SIG adopted Model Editor

To delete a model, select it and click the red X symbol (x). To add a SIG-adopted model, click the green plus symbol (+). A list of all the SIG-adopted models is displayed, and you can choose the one you want to as shown in the following figure. Note that, although all the SIG-adopted models are listed, not all of them are currently supported by the Bluetooth mesh SDK. For the information on the supported models, please check the SDK release notes.

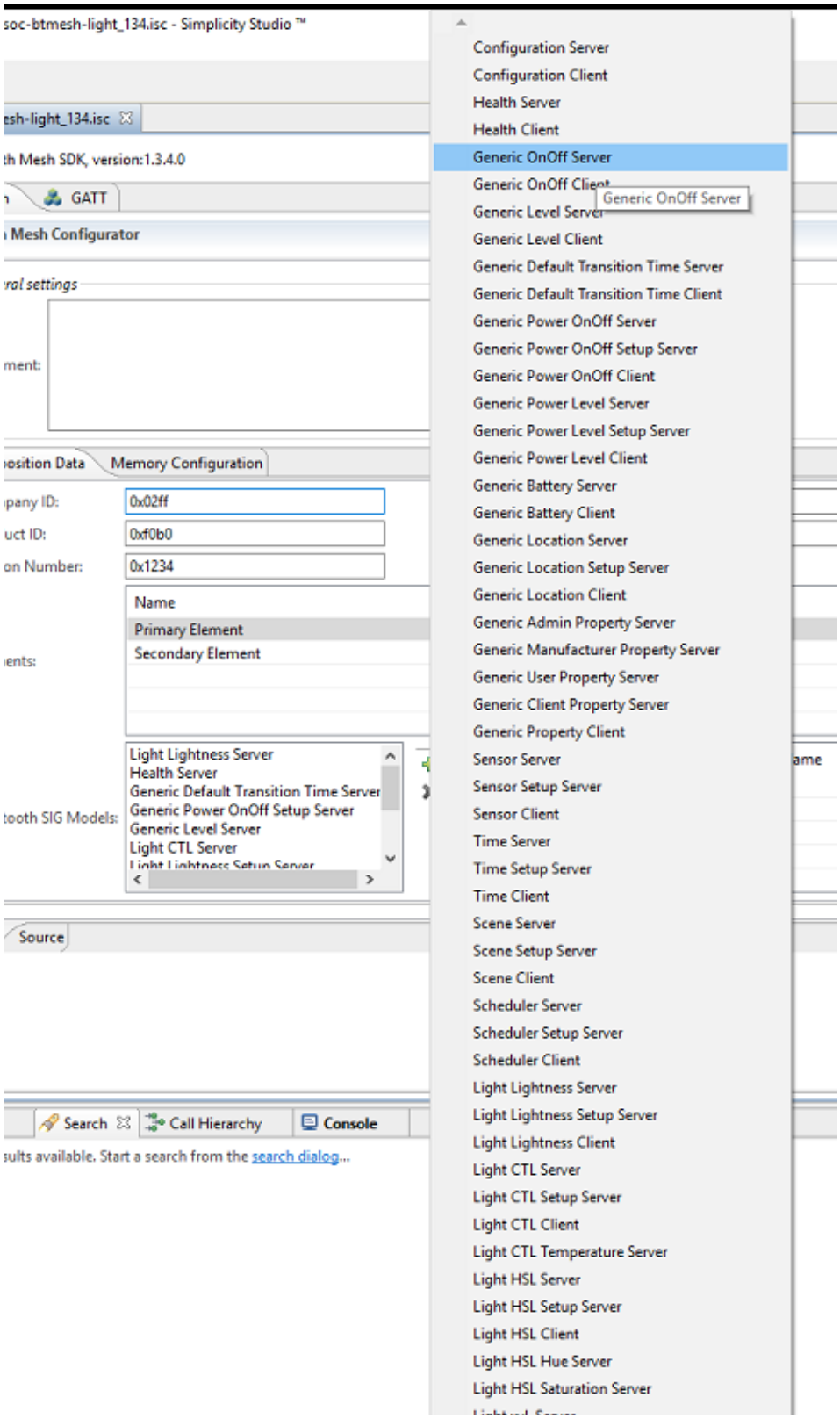


Figure 2-4. All SIG-Adopted Models

Due to the extension mechanism of models mentioned above, attention is needed when adding models to your project:

1. When adding a model that is not a root model, in other words an extended model, all the models it extends from should be also added. The Bluetooth Mesh Model Specification has the detailed definition of the models' relationship. For example, to add a Light Lightness Server model, you need to make sure that the Generic Power OnOff Server model and the Generic Level Server model are both also present in the settings, because the Light Lightness Server model is extended from them.
2. One element can only have one instance of any model. For example, if you want to add model A and Model B which are both extended from Model C to your project, you cannot add all of them to a single element because it requires two Model C instances. The appropriate way to achieve this is to have two elements, put model A and model C in one element and model B and model C in the other one. For example, the light example in the Bluetooth mesh SDK has two elements in order to have two Generic Level Server model instances.

In addition to the above points, make sure to follow the points below when editing the model setting of a node.

1. The configuration server model shall be supported by a primary element and shall not be supported by any secondary elements.
2. To develop a provisioner, you need to add at least the configuration client model in your project, and it should be in the primary element.
3. The health server model shall be supported by a primary element and may be supported by any secondary elements.
4. If the health client model is supported, it shall be supported by a primary element and may be supported by any secondary elements.

2.3.2 Vendor Models Editor

Vendor models give you more flexibility when developing products not covered by the SIG-adopted models. Vendors can define their own specification in these vendor models – including states, messages, and the associated behaviors. Sixty-four opcodes are available for each company identifier. The vendor model editor is shown in the following figure. The ID field contains the 32-bit vendor identifier and model identifier. The two least significant bytes of the ID are the vendor ID and the two most significant bytes are the model ID. In the following figure, 0x02FF is the vendor ID for Silicon Labs, and 0x0001 is the model ID.

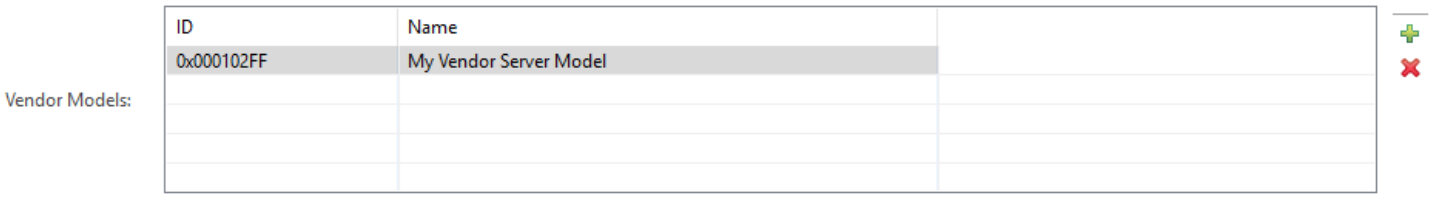


Figure 2-5. Vendor Model Editor

Click the green plus symbol (+) to add a vendor model, or select a model and click the red X symbol (x) to remove it.

3 Memory Configurator

The Bluetooth mesh SDK provides several configuration options for a mesh project to optimize the RAM consumption and the persistent storage (PS Store) usage. The main reason for the memory configuration is because there is a limited amount of RAM and persistent storage in an embedded system. Putting the appropriate value in the fields avoids extra RAM and flash usage in persistent storage so that there is more space for use by the application. The following figure contains all the fields in the memory configuration label.

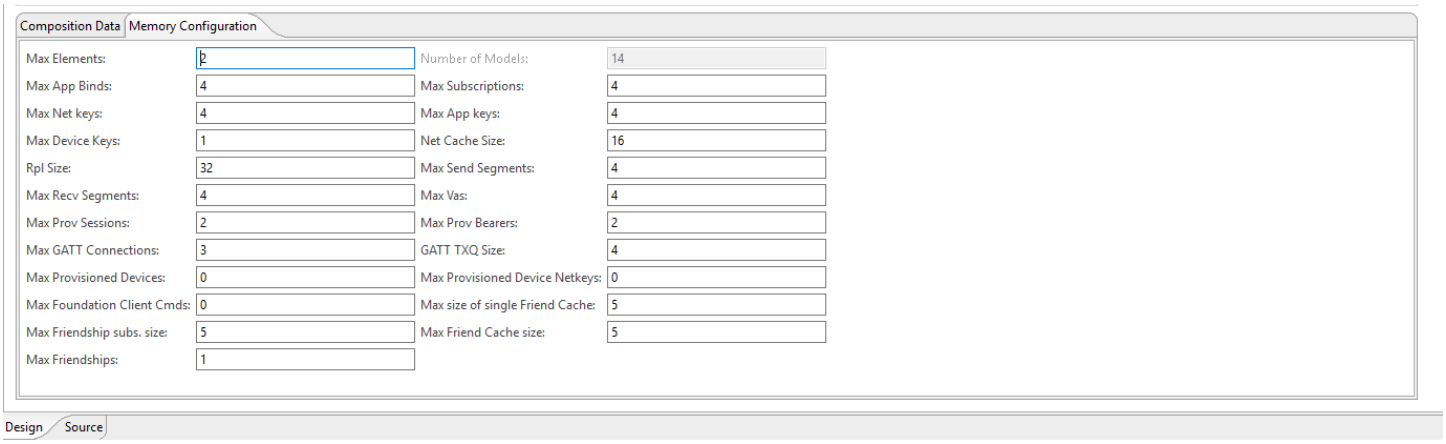


Figure 3-1. Memory Configurator

Some of the configuration options only affect the RAM usage during run time. Resources are allocated when the stack starts up and recycled as needed during operation. Others affect both RAM and persistent storage usage. The Bluetooth mesh stack allocates space in persistent storage to store the configuration data when it first starts up. The following table summarizes the configuration options. Each option is then discussed in more detail.

Table 3-1. Summary of Configuration Options

Configuration Option	Effects on RAM / Persistent Storage	Notes
Max Net keys	Both	No larger than 4
Max App Keys	Both	No larger than 4
Max App Binds	Both	No larger than 4
Max Subscriptions	Both	No larger than 4
Rpl size	Both	Dividable by 16
Number of Models	Both	No need to set manually
Max Vas	Both	Set to 0 if virtual address not used
Max Recv Segments	RAM	Set to a low number if not much segmentation is used
Max Prov Sessions	RAM	1 for node, maybe over 1 for provisioner
Max GATT Connections	RAM	Can be 0 if PB-GATT and GATT bearer are not supported
Max Foundation Client Cmds	RAM	Only applicable for provisioner
Max Friendships	RAM	Only applicable for friend node
Net Cached Size	RAM	Network density dependent
Max Prov Bearers	RAM	Number of supported bearers, GATT, ADV, or both
GATT TXQ Size	RAM	Connection interval dependent
Max size of single Friend Cache	RAM	Only applicable for friend node
Max Friend Cache size	RAM	Only applicable for friend node
Max Friendship subs.size	RAM	Only applicable for friend node

3.1 Max Net Keys

This option affects both RAM and persistent storage usage. Each network key needs 32 bytes of persistent storage for key data (16 bytes for the current network key value and 16 bytes reserved for the old network key value during key refresh) as well as space for key index and storage overhead. Two keys are stored for each PS Store entry to cut the PS Store per-entry overhead by half. Each two-key entry takes 88 bytes with overhead included.

As well as the entries where the keys are stored, the number of network keys that can be stored affects the storage requirements of key refresh state and configuration server model state.

Network key number should be set close to the expected number of keys that will be used in an installation, rounded to the closest even number.

The number of network keys cannot in practice be larger than 4.

3.2 Max App Keys

This option affects both RAM and persistent storage usage. Like network keys, two key values (16 bytes for the current application key value and 16 bytes reserved for the old application key value during key refresh) need to be stored for each application key, in addition to the key index, the bound network key index, and storage overhead.

The application key number should be set close to the expected number of keys that will be used in a network, rounded to the closest even number.

The number of application keys cannot in practice be larger than 4.

3.3 Max App Binds

This option affects both RAM and persistent storage usage. Every model on a node requires some space to store the indices of the application keys bound to the model. While each binding only takes 2 bytes, it should be noted that the total amount of space used is multiplied by the number of models on the node.

The number of bindings should not be set larger than the number of application keys that can be stored on the device. The number of bindings can be set to a smaller number if it is expected that each model will be bound to only one or a few keys.

The number of bindings cannot in practice be larger than 4.

3.4 Max Subscriptions

This option affects both RAM and persistent storage usage. Every model on a node requires some space to store the subscription addresses a model subscribes to. While each subscription entry takes 3 bytes (the 2-byte address plus an additional 1-byte data reference for virtual addresses), it should be noted that the total amount of space used is multiplied by the number of models on the node.

The number of subscriptions should be set to the expected count of subscriptions to be made or slightly larger.

The number of subscriptions cannot in practice be larger than 4.

3.5 Rpl Size

This option affects both RAM and persistent storage usage. The replay protection list stores the sequence numbers received from each peer the node communicates with. To cut down on PS Store overhead, 16 RPL entries are stored per PS Store entry. Each PS Store entry uses about 150 bytes.

The number of replay protection list entries should be set to the number of peers a node is expected to communicate with, rounded up to the nearest number divisible by 16.

3.6 Number of Models

This option affects both RAM and persistent storage usage. This setting is greyed out because the tool will automatically read out the number of models in the DCD configurator. The configuration state for each model takes up roughly 30 bytes, depending on the number

of subscriptions and bindings, plus overhead. As the total size of allocation grows with the number of models, there should not be any unneeded models on a node.

3.7 Max Vas

This option affects both RAM and persistent storage usage. The full 16-byte virtual addresses must be stored persistently if virtual addresses are used in the system, as the full data is needed for decrypting messages destined to virtual addresses.

The number of virtual addresses should be set to as small a number as possible, or zero if it is expected that virtual addresses are not used.

3.8 Max Provisioned Devices

This option affects both RAM and persistent storage usage. This number specifies the max number of devices that can be provisioned by this device. The option is only applicable if the device is a provisioner, can be set to 0 if the device is a regular node in the network.

Due to the limited space of PS Store, the current maximum value of Max Provisioned Devices cannot be larger than 13.

3.9 Max Recv Segments

This option only affects RAM usage. When receiving a message that has been segmented on transport layer, a state machine must be allocated to keep track of received data and trigger timeouts when needed. This option specifies the number of these state machines and thus limits how many ongoing segmented message receptions can be handled concurrently.

A device with standard models rarely receives segmented messages (encryption key deployment being one example) so a low number can be used if none of the vendor models need segmentation.

3.10 Max Prov Sessions

This option only affects RAM usage. The value decides how many ongoing provisioning sessions can be handled concurrently. This does not need to be over 1 for a regular node and may be over 1 for a provisioner that provisions devices concurrently.

3.11 Max GATT Connections

This option only affects RAM usage. The value decides the number of GATT connections that can be handled concurrently. For a regular node that is provisioned over PB-GATT and configured by a GATT proxy bearer, this may need to be set to 2 if the provisioner does not cleanly close the provisioning connection by disabling notifications on the data characteristic before starting to configure the node. It can be 0 if PB-GATT and GATT proxy bearer are not supported.

3.12 Max Foundation Client Cmds

This option only affects RAM usage. The value decides the number of concurrent foundation (configuration and health) client requests that can be waiting for a response from the corresponding servers. Not applicable to a regular node, only to a provisioner.

3.13 Max Friendships

This option only affects RAM usage. The value decides the number of concurrent friendships a friend node can handle. Not applicable to a non-friend node.

3.14 Net Cached Size

This option only affects RAM usage. The network layer cache is used to stop processing copies of messages that have already been processed recently. Receiving copies may occur due to node configuration (network layer retransmissions) or message repetition by overlapping relays.

The suitable value is dependent on expected network density, node configuration, and traffic frequency. For example, if network layer repetition is configured on, with an interval that allows multiple messages to be injected to the network by other nodes during the interval, the cache should be large enough to handle this and not flush the previous Tx before the repetition.

3.15 Max Prov Bearers

This option only affects RAM usage. It may be set to 1 if only PB-ADV or PB-GATT is supported, otherwise set to 2.

3.16 GATT TXQ Size

This option only affects RAM usage. The value decides the number of PDUs that may be pending transmit on the GATT bearer. The value may be small for a node that is only configured over a GATT proxy bearer (default is 4). It may need to be larger for a node that acts as a GATT proxy between the network and a legacy device, and the connection interval for the GATT connection is long.

3.17 Max Size of Single Friend Cache

This option only affects RAM usage. The value decides the amount of memory allocated to one LPN (low power node) for friend message caching, represented as the number of messages that can be stored. Not applicable to a non-friend node. Must be equal to or smaller than the total amount as defined in **Max Friend Cache Size**.

3.18 Max Friend Cache Size

This option only affects RAM usage. The value decides the amount of memory allocated in total for friend message caching, represented as the number of messages that can be stored. Not applicable to a non-friend node.

3.19 Max Friendship Subs.size

This option only affects RAM usage. Messages are forwarded to LPNs if the destination of the message is either one of the LPN's unicast addresses or an address the LPN subscribes to. This configuration parameter specifies the maximum amount of subscription addresses that are tracked per each LPN that has formed a friendship with the friend node.

3.20 Max Elements

This value should be set to the number of elements in total in your project.

3.21 Others

TBD (Use default value).

4 After Configuration

After editing the configurations in the .isc file, click **Generate** in the up right corner of the interface to bring all the changes and configurations into effect, as shown in the following figure.

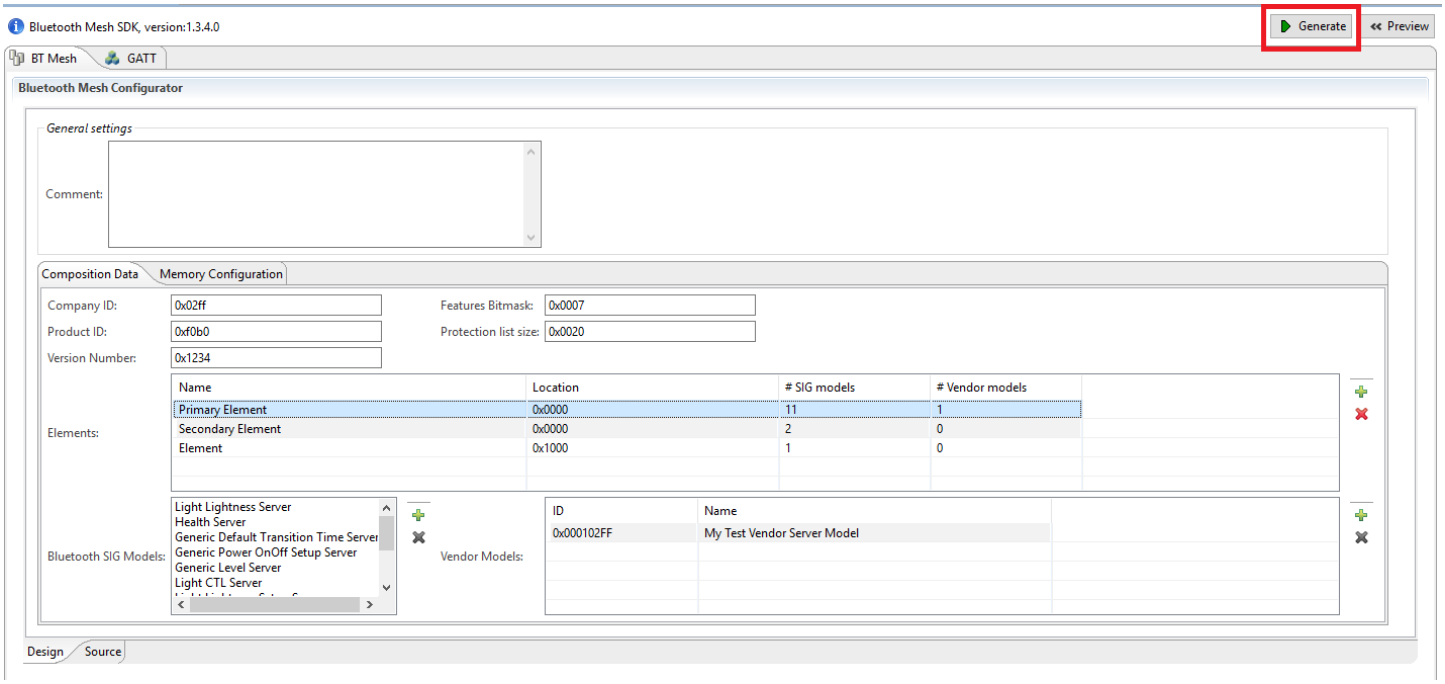


Figure 4-1. Generate Button

When you click **Generate**, Simplicity Studio will:

- Generate dcd.c and mesh_app_memory_config.h files, which are used to pass the configuration data to the Bluetooth mesh stack.
- Back up the current files, which will be overwritten by the new settings, if needed, as shown in the following figure.

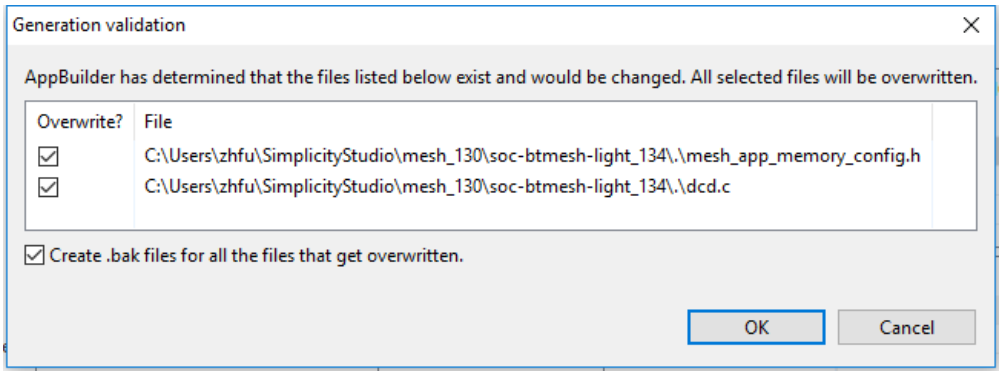


Figure 4-2. Generation Validation

When you click **OK**, you see the modification details, as shown in the following figure.

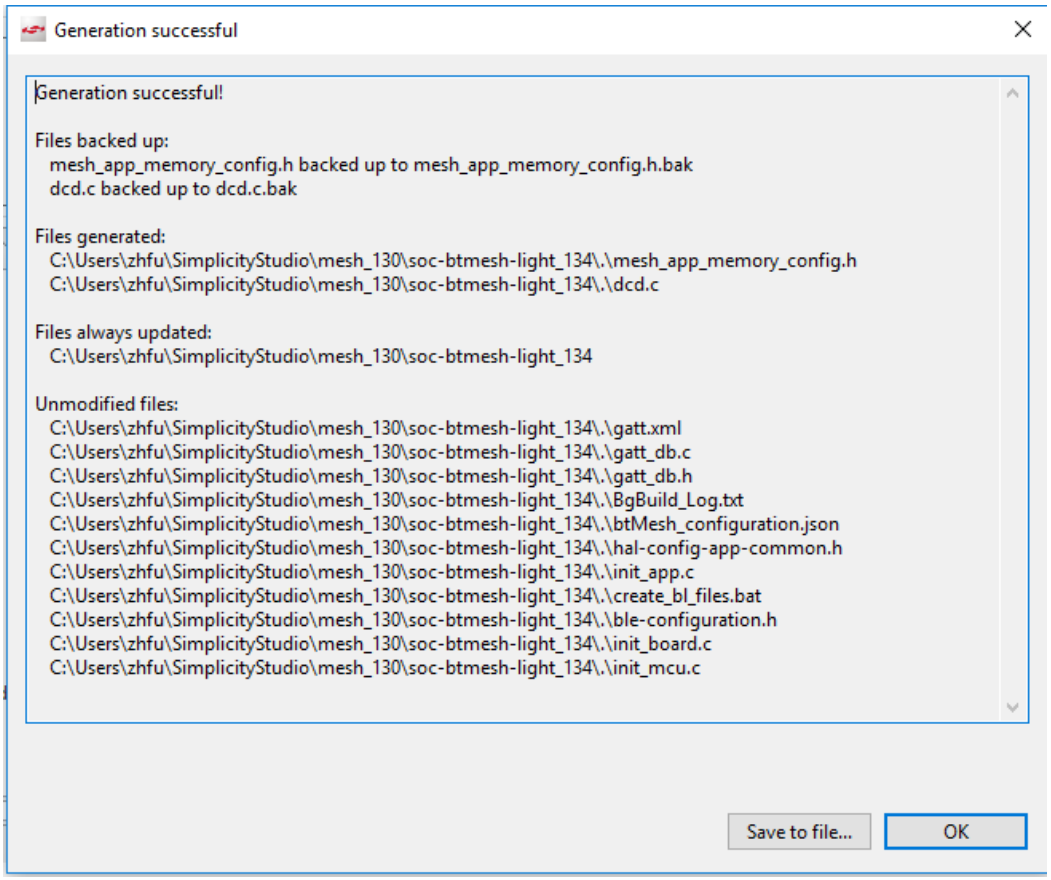
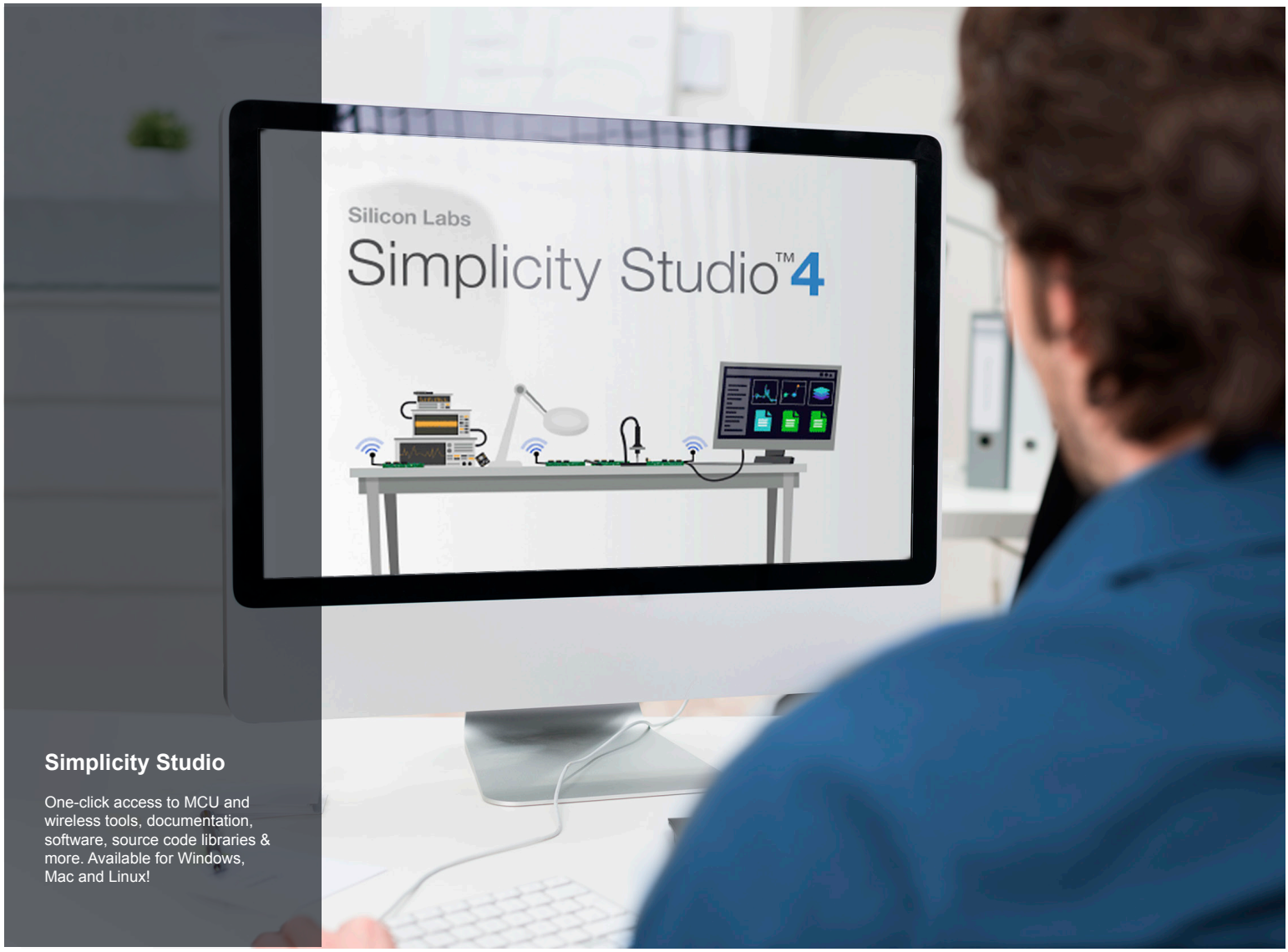


Figure 4-3. File Modification Details

Now when you compile the project, the new settings will take effect. However, if devices already have settings in the PS Store, the Bluetooth mesh stack will not overwrite those settings. You must completely erase the PS Store for the new configuration to take effect.



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 and Community
community.silabs.com

SILICON LABS

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

http://www.silabs.com