# Light switch example

Note
This example is not supported by the nRF52810 Series.

This example demonstrates the mesh ecosystem that contains devices acting in two roles: Provisioner role and Node role (also referred to as provisionee role). It also demonstrates how to use Mesh models by using the Generic OnOff model in an application.

Table of contents

The example is composed of three minor examples:

- Light switch server: A minimalistic server that implements a Generic OnOff server model, which is used to receive the state data and control the state of LED 1 on the board.
- Light switch client: A minimalistic client that implements four instances of a Generic OnOff client model. When a user presses any of the buttons, an OnOff Set message is sent out to the configured destination address.
- Mesh Provisioner: A simple static provisioner implementation that sets up the demonstration network. This provisioner provisions all the nodes in one mesh network. Additionally, the provisioner also configures key bindings and publication and subscription settings of mesh model instances on these nodes to enable them to talk to each other.

Note
For provisioning purposes, you can either use the static provisioner example or use the nRF Mesh mobile app <https://www.nordicsemi.com/Software-and-Tools/Development-Tools/nRF-Mesh >.

The Generic OnOff Client/Server is used for manipulating the on/off state. Note that when the server has a publish address set (as in this example), the server will publish any operation of its state change to its publish address. More information about the Generic OnOff model can be found in the Generic OnOff model documentation and Generic OnOff server behavior documentation.

For a more detailed overview of the example structure and an introduction to various SDK APIs, see the following pages:

- Light switch server details and Mesh APIs
- Light switch client details and Mesh APIs
- Mesh provisioner details and Mesh APIs

The following figure gives the overall view of the mesh network that will be set up by the static provisioner. Numbers in parentheses indicate the addresses that are assigned to these nodes by the provisioner.

mesh-nw-demo_r02.svg
Mesh network demonstration

Both the light switch server and light switch client examples have provisionee role. They support provisioning over Advertising bearer (PB-ADV) and GATT bearer (PB-GATT) and also support Mesh Proxy Service (Server). Read more about the Proxy feature in GATT provisioning and Proxy (experimental).

Note
The *Proxy Client* role is not supported.

## Hardware requirements

You need at least two supported boards for this example:

- One nRF5 development board for the client.
- One or more nRF5 development boards for the servers.

Additionally, you need one of the following:

- One nRF5 development board for the provisioner if you decide to use the static provisioner example.
- nRF Mesh mobile app <https://www.nordicsemi.com/Software-and-Tools/Development-Tools/nRF-Mesh > (iOS <https://itunes.apple.com/us/app/nrf-mesh/id1380726771?mt=8 > or Android <https://play.google.com/store/apps/details?id=no.nordicsemi.android.nrfmeshprovisioner >) if you decide to provision using the application.

See Compatibility for the supported boards.

## Setup

You can find the source code of this example and its minor examples in the following folder: `<InstallFolder>/examples/light_switch`

### LED and button assignments

The buttons (1 to 4) are used to initiate certain actions, and the LEDs (1 to 4) are used to reflect the status of actions as follows:

- Server:
  - During provisioning process:
    - LED 3 and 4 blinking: Device identification active.
    - LED 1 to 4: Blink four times to indicate provisioning process is completed.
  - After provisioning and configuration is over:
    - LED 1: Reflects the value of OnOff state on the server.
      - LED ON: Value of the OnOff state is 1 (`true`).
      - LED OFF: Value of the OnOff state is 0 (`false`).
- Client:
  - During provisioning process:
    - LED 3 and 4 blinking: Device identification active.
    - LED 1 to 4: Blink four times to indicate provisioning process is completed.

  - After provisioning and configuration is over, buttons on the client are used to send OnOff Set messages to the servers:
    - Button 1: Send a message to the odd group (address: 0xC003) to turn on LED 1.
    - Button 2: Send a message to the odd group (address: 0xC003) to turn off LED 1.
    - Button 3: Send a message to the even group (address: 0xC002) to turn on LED 1.
    - Button 4: Send a message to the even group (address: 0xC002) to turn off LED 1.
- Provisioner:
  - Button 1: Start provisioning.
  - LED 1: Reflects the state of the provisioning.
    - LED ON: Provisioning of the node is in progress.
    - LED OFF: No ongoing provisioning process.
  - LED 2: Reflects the state of the configuration.
    - LED ON: Configuration of the node is in progress.
    - LED OFF: No ongoing configuration process.

---

## Testing the example

To test the light switch example, build the examples by following the instructions in Building the mesh stack.

Note
: If you have more than 30 boards for the servers and decided to use the static provisioner example, set `SERVER_NODE_COUNT` in `light_switch_example_common.h` to the number of boards available and rebuild the provisioner example.

After building is complete, use one of the following methods, depending on the preferred provisioning approach:

- Evaluating using the static provisioner
- Evaluating using the nRF Mesh mobile app

Once the provisioning is complete, you can start interacting with the boards.

### Evaluating using the static provisioner

1. Flash the examples by following the instructions in Running examples, including:
   a. Erase the flash of your development boards and program the SoftDevice.
   b. Flash the provisioner and the client firmware on individual boards and the server firmware on other boards.
2. After the reset at the end of the flashing process, press Button 1 on the provisioner board to start the provisioning process:
   - The provisioner first provisions and configures the client and assigns the address 0x100 to the client node.
   - The two instances of the OnOff client models are instantiated on separate secondary elements. For this reason, they get consecutive addresses starting with 0x101.
   - Finally, the provisioner picks up the available devices at random, assigns them consecutive addresses, and adds them to odd and even groups.

   Note
   : You can use RTT viewer to view the RTT output generated by the provisioner. The provisioner prints details about the provisioning and the configuration process in the RTT log.

3. Observe that the LED 1 on the provisioner board is turned ON when provisioner is scanning and provisioning a device.
4. Observe that the LED 2 on the provisioner board is turned ON when configuration procedure is in progress.
5. Wait until LED 1 on the provisioner board remains ON steadily for a few seconds, which indicates that all available boards have been provisioned and configured.

If the provisioner encounters an error during the provisioning or configuration process for a certain node, you can reset the provisioner to restart this process for that node.

### Evaluating using the nRF Mesh mobile app

1. Flash the examples by following the instructions in Running examples, including:
   a. Erase the flash of your development boards and program the SoftDevice.
   b. Flash the client firmware on individual boards and the server firmware on other board or boards.
2. Open the nRF Mesh mobile app.
3. Provision the nodes. The client board is `nRF5x Mesh Switch`, the server board is `nRF5x Mesh Light`.
4. Bind the Generic OnOff client and server model instances on the nodes with the same app key:
   a. Select the Network tab.
   b. On the server board tile, tap the **Configure** button to open Node Configuration.
   c. Expand the Elements section and tap **Generic OnOff Server**.
   d. In the Bound App Keys section, tap the **Bind Key** button and select the app key.
   e. On the client board tile, tap the **Configure** button to open Node Configuration.
   f. Expand the Elements section and tap **Generic OnOff Client**.
   g. In the Bound App Keys section, tap the **Bind Key** button and select the app key.
5. In the client Node Configuration, expand the Elements section.
6. Select the first Generic OnOff Client model instance.
   在 “ 发布 ” 部分中，将 “ 发布地址 ” 设置为服务器节点的以下地址之一：
7. In the Publish section, set the Publish Address to one of the following addresses of the server nodes:
   任何服务器节点的单播地址。 这将配置客户端示例，如下所示：
   - the unicast address of any server node. This configures the client example as follows:
     客户端板上的按钮1打开相应服务器主板上的LED 1
     - The Button 1 on the client board turns ON LED 1 on the corresponding server board.
     客户端板上的按钮2关闭相应服务器主板上的LED
     - The Button 2 on the client board turns OFF LED 1 on the corresponding server board.
   - group addresses – if you choose this option, remember to subscribe the server nodes to these group addresses. 组地址 - 如果选择此选项，请记住将服务器节点订阅到这些组地址。

> Note
>
> You can also configure the publish address of the second Generic OnOff client model instance. Use one of the options mentioned in step 5 above. If you set the address the unicast address of any server node, the client example will be configured as follows:
>
> - The Button 3 on the client board turns ON LED 1 on the corresponding server board.
> - The Button 4 on the client board turns OFF LED 1 on the corresponding server board.

### Interacting with the boards

Once the provisioning and the configuration of the client node and at least one of the server nodes are complete, you can press buttons on the client to see the LEDs getting toggled on the associated servers. See LED and button assignments section.

If an RTT terminal is available and connected to the client, sending the ASCII numbers `0`–`3` will have the same effect as pressing the buttons.

If you are using RTT log, you can also press Button 1 on the servers to locally toggle the state of their LED 1, and the status reflecting this state will be sent to the client board. You can see the status printed in the RTT log of the client board.

If any of the devices is powered off and back on, it will remember its flash configuration and rejoin the network. For more information about the flash manager, see Flash manager.