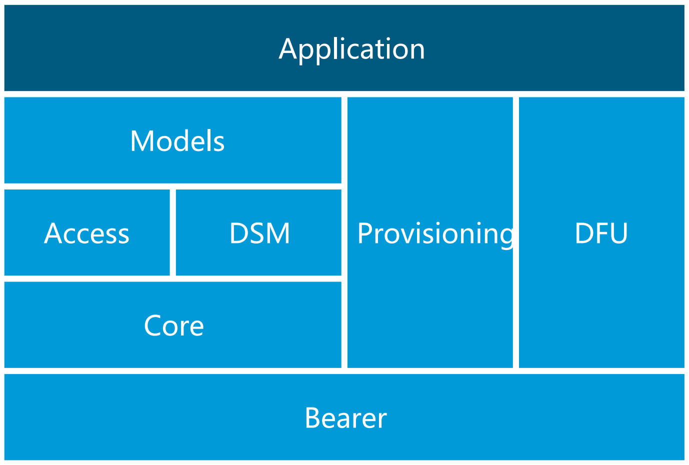
nRF5 SDK for Mesh architecture

The mesh stack consists of a number of subsystems that are interfaced through a set of API modules. The API modules hide the complexity of their subsystems. The functionality provided in the API is sufficient to make a functioning mesh device, so that there is no need to bypass the API.



Basic architecture of the mesh stack in the nRF5 SDK for Mesh

The mesh stack's structure corresponds to the structure of the Bluetooth Mesh Specification and follows the same naming conventions:

- Models: The Bluetooth Mesh models present and implement device behavior.
- Access: The Bluetooth Mesh access layer organizes models and communication.
- DSM: The Device State Manager stores addresses and encryption keys for usage in the models.
- Core: The Core Bluetooth Mesh layer takes care of encryption and message relaying.
- Provisioning: The Bluetooth Mesh provisioning protocol is used for adding devices to the network.
- Bearer: The Bearer layer takes care of low-level radio operation.
- DFU: The Device Firmware Upgrade module cooperates with a bootloader to enable firmware upgrades through the mesh.
- Mesh Stack (not pictured): Top level functionality for initializing and starting the stack.
- Serial (not pictured): Application-level serialization of the Mesh API allows the mesh to be controlled by a separate host device.

See Basic Bluetooth Mesh concepts for an introduction to the Bluetooth Mesh.

Models

API: Mesh Models

The models define the behavior and communication formats of all data that is transmitted across the mesh. Equivalent to Bluetooth Low Energy's GATT services, the Mesh Models are independent, immutable implementations of specific behaviors or services. All mesh communication happens through models, and any application that exposes its behavior through the mesh must channel the communication through one or more models.

The Bluetooth Mesh Specification defines a set of immutable models for typical usage scenarios, but vendors are also free to implement their own models.

You can read more about how to implement your own models in Creating new models.

Access

API: Access

The access layer controls the device's model composition. It holds references to the models that are present on the device, the messages these models accept, and the configuration of these models. As the device receives mesh messages, the access layer finds which models the messages are for and forwards them to the model implementations.

Device State Manager

API: Device State Manager

The Device State Manager stores the encryption keys and addresses used by the mesh stack. When models get assigned application keys and publish addresses through configuration, the Device State Manager stores the raw values and provides handles for the models to use when referencing these values.

The Device State Manager stores its data in persistent storage, which it can recover on bootup.

Mesh Core

API: Core

Consisting of a network and a transport layer, the Mesh Core module provides the mesh-specific transport for the messages.

The transport layer provides in-network security by encrypting mesh packets with *application keys* and splitting them into smaller segments that can go on air. The transport layer re-assembles incoming packet segments and presents the full mesh message to the access layer.

The network layer encrypts each transport layer packet segment with a *network key* and populates the source and destination address fields. When receiving a mesh packet, the network layer decrypts the message, inspects the source and destination addresses, and decides whether the packet is intended for this device and whether the network layer should relay it.

The Mesh Core provides protection against malicious behavior and attacks against the mesh network through two-layer encryption, replay protection, and packet header obfuscation.

Provisioning

API: Provisioning

Provisioning is the act of adding a device to a mesh network. The Provisioning module takes care of both sides of this process, by implementing a provisioner role (the network owner) and a provisionee role (the device to add).

To participate in mesh communication, each device must be provisioned. Through the provisioning process, the new device receives a range of addresses, a network key, and a device key. For a detailed guide on how to use provisioning, see mesh provisioning.

The mesh stack provides two ways to provision a device: directly through the PB-ADV provisioning bearer, or through remote provisioning. The PB-ADV provisioning can only happen between a provisioner and a provisionee that are within radio range of each other, while remote provisioning implements two mesh models that together create a tunnel through the mesh, allowing the provisioner to add devices from a distance, with the help of a PB-ADV proxy device.

Note
The remote provisioning is a Nordic proprietary feature that cannot be used with devices from other vendors.

The Remote Provisioning example demonstrates remote provisioning. The light switch example shows the provisioner and provisionee side of PB-ADV as a first step to establishing the network.

Bearer

API: Bearer

The Bearer is the low-level radio controller and provides an asynchronous interface to the radio packet sending and receiving for the upper layers. It enforces Bluetooth low energy compliance for packet formats and timing and operates directly on radio hardware through the SoftDevice Timeslot API.

The Bearer is an internal module that normally does not need to be accessed by the application.

DFU

API: DFU

The Device Firmware Upgrade module provides firmware update capabilities over the mesh by cooperating with a bootloader. It is capable of concurrent, authenticated firmware transfers to all devices in a network, without halting the application.

Note that the DFU procedure is not compatible with the Bluetooth low energy secure DFU procedure used in the nRF5 SDK.

Note
The mesh DFU is a Nordic proprietary feature that cannot be used with devices from other vendors.

For more information about DFU, see the DFU protocol section, including information about how to run Mesh DFU.

Mesh Stack

API: Mesh stack

The Mesh Stack module is a thin wrapper around the top-level mesh modules that makes it easy to get started using the mesh. It takes care of mesh initialization and enabling. It also contains functions for storing and erasing provisioning and state related data.

Serial

API: Serial

The Serial module provides full serialization of the mesh API, allowing other devices to control the nRF5 mesh device through a UART interface. Intended for network gateways and similar complex applications, the serial interface provides a way to access the mesh through a Nordic device, without making it the unit's main controller.

The mesh serial interface is based on the nRF8001 <https://www.nordicsemi.com/Products/Low-power-short-range-wireless/nRF8000-series> ACI serial interface and optionally supports SLIP <https://en.wikipedia.org/wiki/Serial_Line_Internet_Protocol>-encoded operation. The serial protocol can be run as a stand-alone application (see the serial example) or alongside a normal mesh application.

An overview of the serial packet format, commands, and events can be found in the serial documentation.