UG103.12: Silicon Labs Connect Fundamentals

This version of UG103.12 has been deprecated.

For the latest version, see docs.silabs.com.

****

This document describes the features and functions of Silicon Labs Connect, including its device types, modes of operation, stack structure, and the IEEE 802.15.4 support for RAIL vs. Connect.

Silicon Labs' *Fundamentals* series covers topics that project managers, application designers, and developers should understand before beginning to work on an embedded networking solution using Silicon Labs chips, networking stacks such as EmberZNet PRO or Silicon Labs *Bluetooth*®, and associated development tools. The documents can be used as a starting place for anyone needing an introduction to developing wireless networking applications, or who is new to the Silicon Labs development environment.

The *Connect User's Guide* provides in-depth information for developers who are using the Silicon Labs Connect Stack for their application development. Two series are published, one for Connect SDK v2.x and one for Connect SDK v3.x. Refer to *UG235.01: Developing Code with Silicon Labs Connect v2.x* and *UG435.01: Developing Code with Silicon Labs Connect v3.x* for an overview of the chapters in their respective *Connect User's Guide*.

Proprietary is supported on all EFR32FG devices. For others, check the device's data sheet under Ordering Information > Protocol Stack to see if Proprietary is supported. In Proprietary SDK version 2.7.n, Connect is not supported on EFR32xG22.

KEY POINTS

- Devices
- Modes of operation
- Stack structure
- IEEE 802.15.4 support: RAIL vs. Connect

---

1. Introduction

Silicon Labs is developing products designed to meet customer demands as we move to an ever-connected world of devices in the home, often referred to as the IoT (Internet of Things). At a high level, the IoT goals for Silicon Labs are to:

- Connect all devices in the home with best-in-class mesh networking, whether with Ember ZigBee PRO or other emerging standards.
- Leverage the company's expertise in low-power, constrained devices.
- Enhance established low-power, mixed-signal chips.
- Provide low-cost bridging to existing Ethernet and Wi-Fi devices.
- Enable cloud services and connectivity to smartphones and tablets that promote ease of use and a common user experience for customers.

Achieving these goals will increase adoption rates and user acceptance for IoT devices in the Connected Home.

A common challenge in the IoT is managing devices requiring low power consumption, such as battery-powered devices where long battery life is essential. To meet this challenge, Silicon Labs has developed the Silicon Labs Connect stack. Connect provides a fully-featured, easily customizable wireless networking solution optimized for devices that require low power consumption and are used in a simple network topology. Connect is configurable to be compliant with regional communications standards worldwide. Each RF configuration is designed for maximum performance under each regional standard.

The Silicon Labs Connect stack supports many combinations of radio modulation, frequency, and data rates. The stack provides support for end nodes, coordinators, and range extenders. It includes all wireless MAC (Medium Access Control) layer functions such as scanning and joining, setting up a point-to-point or star network, and managing device types such as sleepy end devices, routers, and coordinators. With all this functionality already implemented in the stack, users can focus on their end application development and not worry about the lower-level radio and network details.

The Connect stack should be used in applications with simple network topologies, such as a set of data readers feeding information directly to a single central collection point (star or extended star topology), or a set of nodes in the same range exchanging data in a single-hop fashion (direct devices or MAC devices). It does not provide a full mesh networking solution such as that provided by the EmberZNet PRO or Silicon Labs Thread stacks.

The Connect stack is part of the Silicon Labs Flex SDK (Software Development Kit), installed through Simplicity Studio. Connect runs on top of RAIL (Radio Abstraction Interface Layer), also included with the Flex SDK. RAIL provides an intuitive, easily-customizable radio interface layer that is designed to support proprietary or standards-based wireless protocols. For more information, see *UG103.13: RAIL Fundamentals*.

The Connect stack supports efficient application development through its "building block" plug-in design. When used with the Simplicity Studio IDE (Integrated Development Environment), developers can easily select the functions that should be included in the application. The resulting applications are completely portable, in that they can be recompiled for different regions and EFR32 devices.

2. Connect Modes of Operation

Connect supports three distinct modes of operation. Only one mode is allowed in any single network and there is no simple way to upgrade from one mode to the other. As a result, you should select the mode carefully early in the design process.

2.1 Extended Star Mode

In this mode, Connect supports star and extended star topology networks per the following figure.
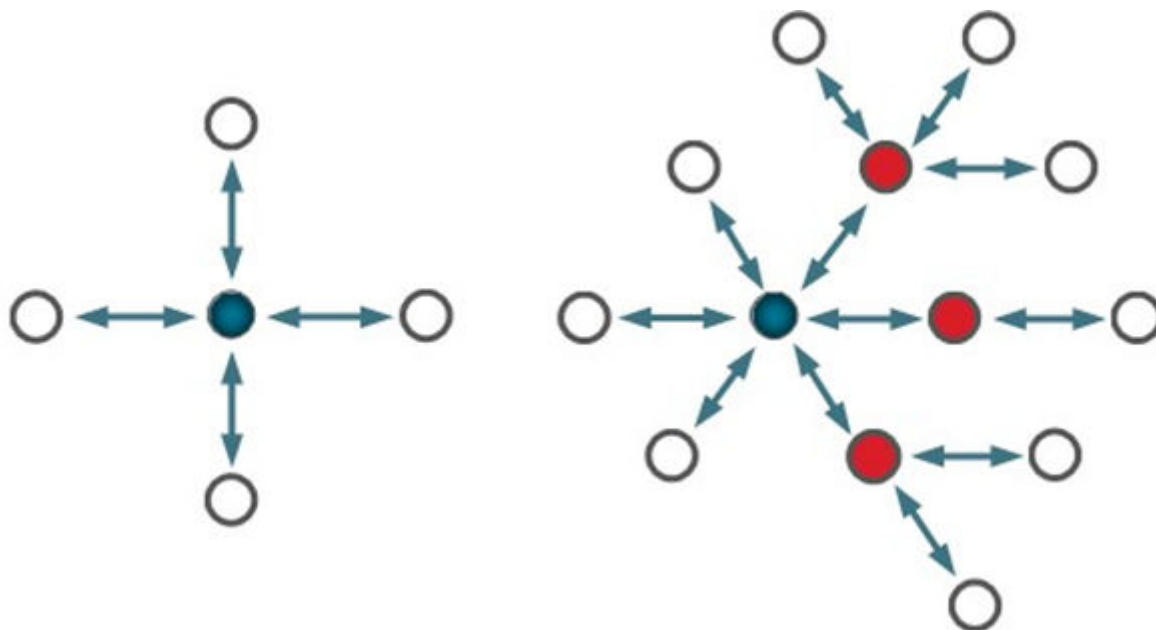

Figure 2.1. Star and Extended Star Topology

Extended Start Mode supports the following device types.

Star coordinator	Blue circle	The star coordinator forms and manages the star or extended star network. The star coordinator also communicates with other range extenders and end nodes. Each Connect star network has a single coordinator.
Star range extender	Red circle	A device between the star coordinator and one or more star end nodes that can be used to extend the range of the star end nodes.
Star node	White circle	Joins to a star coordinator or a star range extender.

Data message routing between any two devices is supported by the network layer in this mode. Extended star topology is a centralized network. Hence, joining to the network requires acceptance by the Personal Area Network (PAN) coordinator. Short address allocation and assignment can be handled by the PAN coordinator. This mode is not fully IEEE 802.15.4 compliant.

2.2 Direct Mode

In this mode, Connect only provides connections between devices that are in range of each other (see the figure below). This is not a centralized topology.
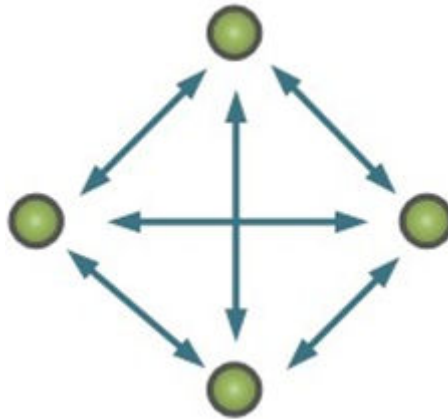

Figure 2.2. Direct Topology

Direct mode supports the **Direct device** type, which send (and receive) messages to (and from) other (Direct) devices in range on the same PAN, with no star topology restrictions. Message forwarding can be implemented in an upper layer.

The Connect network layer is still enabled in this mode, but it does not provide routing, only endpoints. Routing protocols can be implemented in the application layer. Any device can join the PAN by setting the right PAN parameters. Short address allocation is not provided by the Connect stack and address duplication must be avoided by the application. This mode is not fully IEEE 802.15.4 compliant.

2.3 MAC Mode

MAC mode supports the **MAC device** type, which send and receive standard 802.15.4 messages from other 802.15.4 devices in range. Message forwarding can be implemented in an upper layer.

MAC mode is a fully IEEE 802.15.4 compliant setup of the Connect MAC layer; the network layer is not enabled. The API is more complex compared to Direct mode and requires some knowledge of the IEEE 802.15.4 standard. To make it fully IEEE 802.15.4 compliant, make sure to set up an IEEE802.15.4 compliant radio configuration.

2.4 Examples of Connect Modes and Network Topologies

An example of a Connect network is a network of temperature and humidity sensor end nodes installed throughout a home. Each end node periodically takes a reading and transmits that data either directly to a coordinator (sink) or, for those sensors placed farther away from the coordinator, to a range extender. The range extenders take data from the sensors and forward them to the coordinator. The coordinator not only forms and manages the network, but also sends the compiled data to an environmental management system that is part of another network.

Another example is a topology of two minimally-featured nodes that exchange data in both directions. This topology can be used as a generic wire replacement.

A third example is a topology of N direct devices or MAC devices all in range that exchange data between any pair of two nodes in both directions in a single-hop fashion.

3. Stack Structure

The Connect stack provides code organized into three functional layers, as shown in the following figure:

- PHY (physical)
- MAC (Media Access Control)
- Network

The PHY and MAC layers are based on the IEEE 802.15.4-2011 standard (abbreviated to IEEE 802.15.4). The Network layer is based on a proprietary protocol.
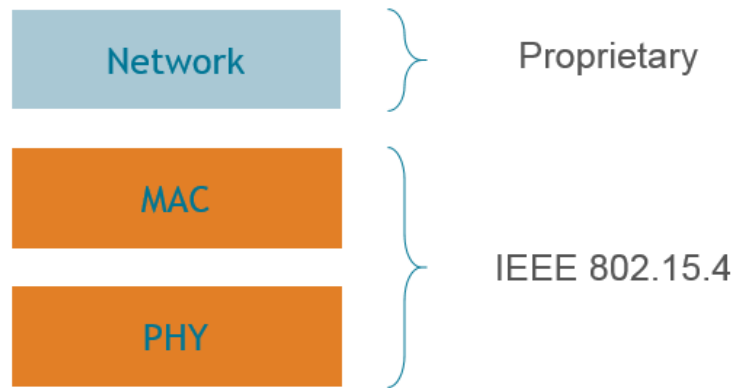

Figure 3.1. Connect Stack Layers

Additionally, the Connect Application Framework provides a complete tool and API infrastructure over the underlying stack layers. Functionality within the Application Framework and the Connect stack layers is provided in the form of individual building blocks called plugins. Details of the plugins for each layer are provided in the *Silicon Labs Connect Application Framework API Reference* included in the Connect stack documentation.

3.1 PHY and MAC Layers

The IEEE 802.15.4 specification is a standard for wireless communication that defines the MAC and PHY operating at subGHz frequencies and in the 2.4GHz band. IEEE 802.15.4 was designed with low power in mind. The Connect stack is based on the IEEE 802.15.4 standard, however it does not fully comply with the standard. On the other hand, some of the difference is beneficial, as the Connect PHY is highly customizable (whereas the IEEE 802.15.4 specification is rigidly restrictive).

The 802.15.4 MAC layer is used for basic message handling and congestion control. The Connect network layer builds on these underlying mechanisms to provide end-to-end communications in the network. The MAC layer includes a CSMA/CA (Carrier Sense Multiple Access / Collision Avoidance) mechanism for devices to listen on and determine whether the active channel is clear, as well as handle retries and acknowledgement of messages for reliable communications between adjacent devices. The MAC layer also provides security functionality (authentication, encryption, and replay attack protection). The MAC auxiliary header indicates which security scheme is used for a given packet. Optional security schemes can be implemented through the AES (Advanced Encryption Standard) plugin. The destination node looks at the auxiliary header and uses the correct security scheme (if it supports it) to decrypt and authenticate the incoming packet. The Connect stack supports both short (2-byte) and long (8-byte) identifiers. A network is identified by a 2-byte PAN ID.

One of the characteristics derived from the need for low power and limiting the BER (Bit Error Rate) is enforcing smaller sized packets to be sent over the air. These can be up to a maximum of 127 bytes at the PHY layer payload. The MAC layer payload can vary depending on the security options and addressing type as illustrated in the figure below.
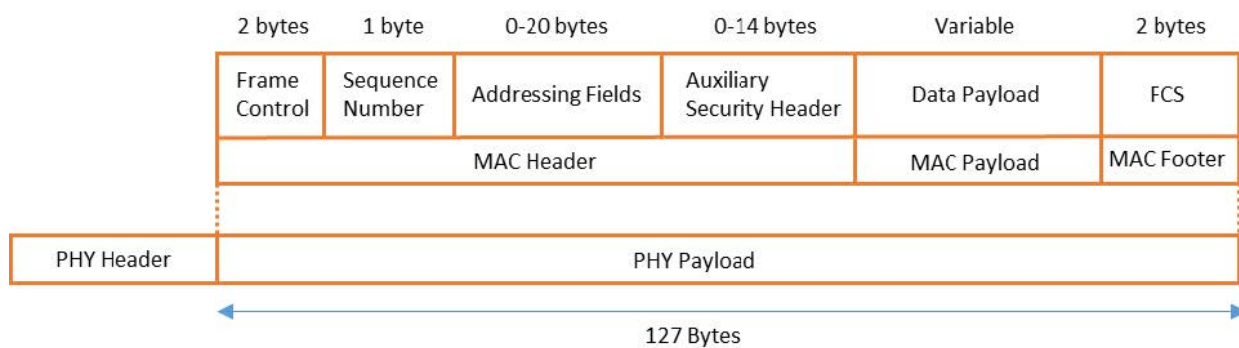

Figure 3.2. 802.15.4 MAC Payload

3.2 Network Layer

In extended star mode, the proprietary network layer provides network formation and full routing support, meaning every node in the network can communicate with any other node in the network in both directions. Routing is transparent to the application layer. Routing is not performed for direct devices.

The network formation functionality offers an association mechanism that, while similar to that in the 802.15.4 protocol, has been improved and made more secure by providing a special encrypted association request command not present in the 802.15.4 protocol. Network formation also includes centralized address allocation at the star coordinator. The network layer additionally provides APIs for commissioning node network parameters like node ID, PAN ID, channel, and transmission power. Direct devices can join to a network using the commissioning APIs. MAC devices can join to a network using either the standard 802.15.4 joining procedure or the commissioning APIs.

3.3 Application Framework

The Connect Application Framework leverages the Ember Application Framework v6 and its bookkeeping functionality implemented in callbacks such as `init()`, `tick()`, and `stackStatus()`. Application Framework plugins can provide callbacks and can implement the callbacks of other plugins.

3.4 Functionality Blocks

Every Connect application includes the following functionality blocks. The HAL (hardware abstraction layer) and Simulated EEPROM functionality blocks reside below the Connect stack. The PHY, event system, and message builder/parser functionality blocks are part of the stack itself.

- **HAL:** Drivers for devices and their peripherals such as SPI (Serial Peripheral Interface), UART (Universal Asynchronous Receiver/Transmitter), timers, and so on.
- **Simulated EEPROM:** Wear-leveled persistent storage of network and application data.
- **Event system:** System that allows the stack and the application to schedule code to run after some specified time interval. Events are also useful when an ISR (Interrupt Service Routine) needs to initiate an action that should run outside the ISR context.
- **PHY:** Software module that interfaces with the transceiver and provides basic radio TX, RX, and radio sleep functionality.
- **Message builder/parser:** Provides a 15.4-like PHY/MAC packet format builder/parser and a proprietary network layer format builder/parser.
- **Dynamic memory allocation:** A generic lightweight module that provides dynamic memory allocation and garbage collection.

4. IEEE 802.15.4 Support: RAIL vs. Connect

The Connect stack relies on hardware-accelerated IEEE 802.15.4 functionality implemented by the RAIL library, and this interface is also available to application code. However, there are some differences in the developer experience (and application solution space) between Connect-based and "bare metal" RAIL-based 802.15.4 applications.

In most cases, using Connect for building an IEEE 802.15.4 application is easier than implementing the same feature set based on the RAIL library directly. Connect already provides the following features:

- Association procedure to connect nodes to a network
- CSMA/CA to minimize collision and enable higher throughput / less lost packets
- Security (authentication, encryption, and replay attack protection)
- Frame assembly (the customer only needs to provide the message payload)

As these features are not implemented in RAIL, it is the customer's responsibility to implement them (if desired) when not using Connect.

Cases when the RAIL library may be preferred over the Connect stack for IEEE 802.15.4-based communication are those where the device lacks sufficient code memory for the Connect stack, or where a non-standard implementation is necessary.

The RAIL library IEEE 802.15.4 API is available at https://docs.silabs.com/rail/latest/group-i-e-e-e802-15-4 for reference.

5. Next Steps

See *QSG138: Getting Started with the Silicon Labs Flex Software Development Kit for the Wireless Gecko (EFR32™) Portfolio* for instructions on using Simplicity Studio and the WSTK to develop a Connect-based application using the Flex SDK.

See *UG235: Silicon Labs Connect User's Guide* and https://docs.silabs.com/connect-stack/latest/ for detailed Connect stack documentation.

See *AN902: Building Low Power Networks with the Silicon Labs Connect Stack* for instructions specific to low-power implementations.

Smart. Connected.
Energy-Friendly.

IoT Portfolio
www.silabs.com/products

Quality
www.silabs.com/quality

Support & Community
www.silabs.com/community

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

SILICON LABS

www.silabs.com