UG235.01: Developing Code with Silicon Labs Connect v2.x

The *Connect v2.x User's Guide* provides in-depth information for developers who are using the Silicon Labs Connect stack as a foundation for their application development on the Wireless Gecko (EFR32™) portfolio or custom hardware. The Connect stack is delivered as part of the Silicon Labs Proprietary Flex SDK. The *Connect v2.x User's Guide* assumes that you have already installed the Simplicity Studio development environment and the Flex SDK and that you are familiar with the basics of configuring, compiling, and flashing Connect-based applications.

If you are new to Connect v2.x and the Proprietary Flex SDK, see *QSG138: Proprietary Flex SDK v2.x Quick Start 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

- Introduces the *Connect User's Guide* and its contents.
- Describes available software and hardware tools.
- Provides an overview for Silicon Labs Connect application development.
- Describes configuration options available through AppBuilder in Simplicity Studio.
- Describes the Silicon Labs Connect Application Framework.

1. Introducing the Silicon Labs *Connect User's Guide*

Silicon Labs recommends that you review *QSG138: Getting Started with the Silicon Labs Flex Software Development Kit for the Wireless Gecko (EFR32™) Portfolio* before proceeding with the more focused discussion presented in the *Connect User's Guide* series. For an overview of Connect and its features, see *UG103.12: Application Development Fundamentals: Silicon Labs Connect*. For the Connect API documentation, go to https://docs.silabs.com/connect-stack/latest.

The *Connect User's Guide* series provides in-depth guidance in the following chapters:

- *UG235.01: Developing Code with Silicon Labs Connect*—Introduces the *Connect User's Guide* and provides a general overview of the code development process for Connect-based applications, and the hardware and software tools that support it.
- *UG235.02: Using Silicon Labs Connect with IEEE 802.15.4*—Connect is based on the IEEE 802.15.4-2011 standard which defines various Physical (PHY) and Media Access Control (MAC) layer modes designed for short-range communication in Personal Area Networks (PANs). UG235.02 gives a short introduction to the IEEE 802.15.4 features that are used in Connect, suitable to understand the Silicon Labs Connect MAC layer without reading the full specification.
- *UG235.03: Architecture of the Silicon Labs Connect Stack*—Presents a complete review of the Connect stack architecture and capabilities: available modes of operation, supported network topologies, stack layers (PHY, MAC, Network), and features applicable to each configuration.
- *UG235.04: Customizing Applications with Silicon Labs Connect*—Silicon Labs Application Builder (AppBuilder) selectively implements plugins, callbacks, and events on top of the Application Framework to provide a developer-configurable sophisticated state machine for complex features and application behavior. UG235.04 describes how to apply this Connect Application Framework to develop firmware optimized for your target application.
- *UG235.05: Using Micrium OS (RTOS) with Silicon Labs Connect*—Describes the process to implement a Connect-based application on top of the Micrium OS RTOS (real-time operating system).
- *UG235.06: Bootloading and OTA with Silicon Labs Connect*—Explains the bootloader options (standalone, application, and Over the Air (OTA)) available for use within Connect-based applications.
- *UG235.07: Energy Saving with Silicon Labs Connect*—Discusses techniques to reduce power consumption of network applications based on the Silicon Labs Connect Stack.
- *UG235.08: Using the Command Line Interface with Silicon Labs Connect*—Describes how to use the powerful Command Line Interface (CLI) with Silicon Labs Connect and how to add a CLI command.
- *UG235.09: Using NCP with Silicon Labs Connect*—Discusses the implementation of a Network Coprocessor (NCP) application based on the Connect stack.

2. Connect Overview

The Silicon Labs Connect stack 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.

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 common wireless MAC 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 Connect stack, developers 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). It does not provide a full mesh networking solution such as that provided by the EmberZNet PRO stack.

The Connect stack supports efficient application development through its building block plugin design. When used with AppBuilder, developers can easily select which functions should be included in their applications. The resulting applications are completely portable—they can be recompiled for different regions, different MCUs, and different radios.

2.1 Connect Developer Resources

Silicon Labs equips developers with an array of tools to accelerate the code development process for applications based on the Connect stack, reducing time-to-market, and facilitating the creation of robust wireless solutions. These include:

- **Simplicity Studio**, featuring:
  - **Eclipse-based Integrated Development Environment (IDE):** Enables code editing, downloading, and debugging for all Silicon Labs EFR32 MCUs.
  - **Flex SDK (contains the Connect Stack):** Supports both GCC (included) and IAR toolchains. Provides pre-compiled stack libraries, fully documented stack API, and full source code for comprehensive and configurable application framework.
  - **Application Builder (AppBuilder, which includes the Radio Configurator):** Pivotal utility that facilitates development of Connect-based applications with a graphical interface for Radio/PHY configuration and a modular application code constructor.
  - **Network Analyzer:** Enables debugging of complex wireless systems by capturing a trace of wireless network activity (such as timestamps, Link Quality Indicator (LQI), Relative Received Signal Strength Indicator (or RSSI), CRC pass/fail results, etc.) that can be examined in detail live or later–all without any software overhead.
  - **Multi-Node Energy Profiler:** Performs Advanced Energy Monitoring (AEM) which enables network power profiling of code in real time by measuring the per-node power consumption. Code correlation links the power consumed with the associated line of code.
  - **Device Console:** Provides a CLI with a powerful way to interact with Connect sample applications and custom applications that implement the serial interface.
- Hardware tools:
  - **Wireless Starter Kit (WSTK):** Provides a common motherboard to support development and debug features for all EFR32 variants. The on-board SEGGER J-Link debugger can target an installed radio board during proof-of-concept phase or an external customer board to support in-system bring-up.
  - **EFR32 Radio Boards:** Tuned for various operational frequencies, transmit powers, etc. The boards contain both a PCB antenna and a U.FL cable connector to support immediate range testing and RF lab measurements.
  - **Easy copy-and-paste reference designs:** Design files for the WSTK and radio boards are accessible from within Simplicity Studio.

For general guidance on how to install, access, and utilize these products, see *QSG138: Getting Started with the Silicon Labs Flex Software Development Kit for the Wireless Gecko (EFR32™) Portfolio*.

3. Code Development Process for Connect-based Applications

The Connect stack has been structured to support optional functionality blocks. These software components are called **plugins**. An Application Framework is layered on top of the Connect stack, consumes the stack handler interfaces, and exposes its own more highly abstracted and application-specific interface to developers. The Connect Application Framework is also structured in plugins.

Provided as either a standalone pre-compiled library or a set of source code, each plugin adds a specific feature set to the project. Unnecessary features can be disabled by removing the associated plugin(s), which removes the code from the build and reduces the project's resource footprint. A plugin can have plugin options, depend on other plugins, can provide callbacks, subscribe to, and implement callbacks.

Connect stack APIs can be invoked directly by the application. Stack callbacks are implemented by the associated plugin and dispatched to the application and subscribing plugins using the Application Framework bookkeeping functionality. The procedure to construct an application based on the Connect stack generally involves the following steps:

1. Using AppBuilder in Simplicity Studio: configure a new project to support your desired wireless protocol (that is, configure the radio) and selectively enable any features/plugins/callbacks your application will require. Click **[Generate]** to populate your project workspace with the files and linked resources necessary to implement your application.
2. Within the Simplicity Studio IDE: write the specifics of your application into the callback functions that were generated along with your project files. Use the Connect Application Framework API and Hardware Abstraction Layer (HAL) support to interact with the network and perform the tasks required of your application.
3. Build and flash the project to your target EFR32 device(s).
4. Leverage the powerful toolset (Network Analyzer, Multi-node Energy Profiler, CLI, etc.) in Simplicity Studio to verify or debug your application.

3.1 Application Builder (AppBuilder)

Connect-based application development begins with and depends heavily upon AppBuilder within Simplicity Studio, a graphical user interface that works in harmony with the Flex SDK (including the Connect stack) and the Gecko Bootloader (see Figure 3.1 Silicon Labs Application Builder (AppBuilder) on page 5). AppBuilder allows developers to start a new project based on an existing framework of best practice application state machine code developed and tested by Silicon Labs. This framework sits on top of the Connect stack to interface with the HAL and provide application-layer functionality, including the following:

- Start-up routines
- Mechanisms for finding, joining, or forming networks
- High-level APIs for creating, parsing, and handling message payloads
- Configuration of the networking stack
- CLIs for control of the program
- Human-readable debug output tailored to the needs of the developer
- Incorporation of customer-provided libraries and code modules

As depicted in the following figure, AppBuilder has the following seven tabbed sections:

- **General:** Defines the output directory where AppBuilder will generate the project file structure. Specifies the target architecture for the project (toolchain and target hardware). For Connect sample applications, describes how to utilize and interact with the application.
- **HAL:** Specifies whether to include bootloader capability (and what kind). Provides for integration with the Hardware Configurator (to define utilization and routing for peripherals, timers, GPIO, etc.). See *AN1115: Configuring Peripherals for 32-Bit Devices in Simplicity Studio* for details.
- **Radio Configuration:** Allows full configuration of the PHY. Select from the pre-built PHYs optimized for specific regional standards or fully-customize to accommodate your desired radio parameters. See *AN971: EFR32 Radio Configurator Guide* for details.
- **Printing:** Configure debug printing and CLI feature support. See *UG235.08: Using the Command Line Interface with Silicon Labs Connect* for details.
- **Plugins:** Select from a wide variety of plugins that each implement one or more callbacks on top of the Application Framework to provide a sophisticated state machine for complex features like application security, OTA bootloader support, etc. Source code for most plugins is available so the code can be used as a point of reference for customized implementations. See *UG235.04: Customizing Applications with Silicon Labs Connect* for details.
- **Callbacks:** Callbacks allow user applications to be notified about important events or make decisions based on runtime state. Software designers can choose which callbacks are needed for their application and can leave the rest disabled so that only the chosen callbacks appear in the user's Callbacks C file. Callbacks are the places where custom application code can be added on top of the Silicon Labs existing framework to give that application unique behaviors and decide how it will react. See *UG235.04: Customizing Applications with Silicon Labs Connect* for details.
- **Other:** Additional macros and filesystem resources can be defined for the project. Additionally, events can be defined, which allow the Connect stack to support scheduling of timing-specific actions. See *UG235.04: Customizing Applications with Silicon Labs Connect* for details.

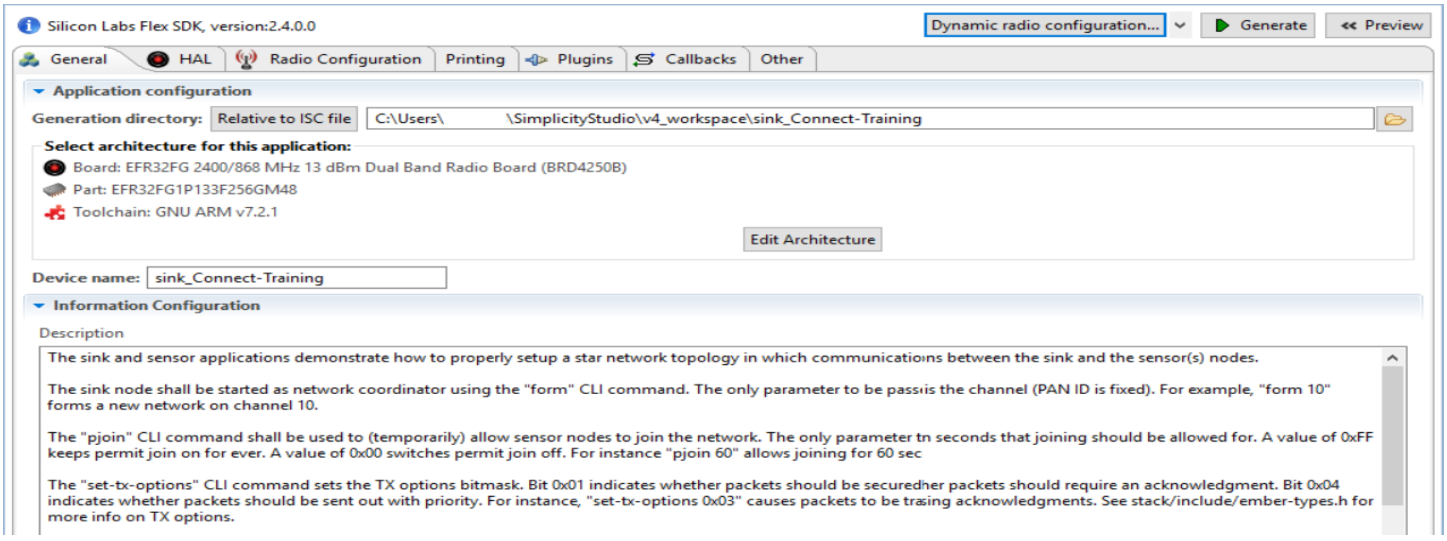


Figure 3.1. Silicon Labs Application Builder (AppBuilder)

Once a developer has completely configured application details within AppBuilder, clicking the **[Generate]** button (see Section 3.2 Application Generation) will generate a software project with customized header files and array definitions in C code to represent the desired application behavior. The project can then be built from within the Simplicity Studio IDE to compile a binary that can be loaded

onto the target device. For more information on this process (compiling, flashing, and debugging), see *AN0822: Simplicity Studio™ User's Guide*.

3.2 Application Generation

The project can be generated by clicking the **[Generate]** button in the top right corner of the AppBuilder frame. This will pull in all the required files for the project and generate the radio and other config files. Before overwriting anything, it will open a Generate validation window, as depicted in the following figure.

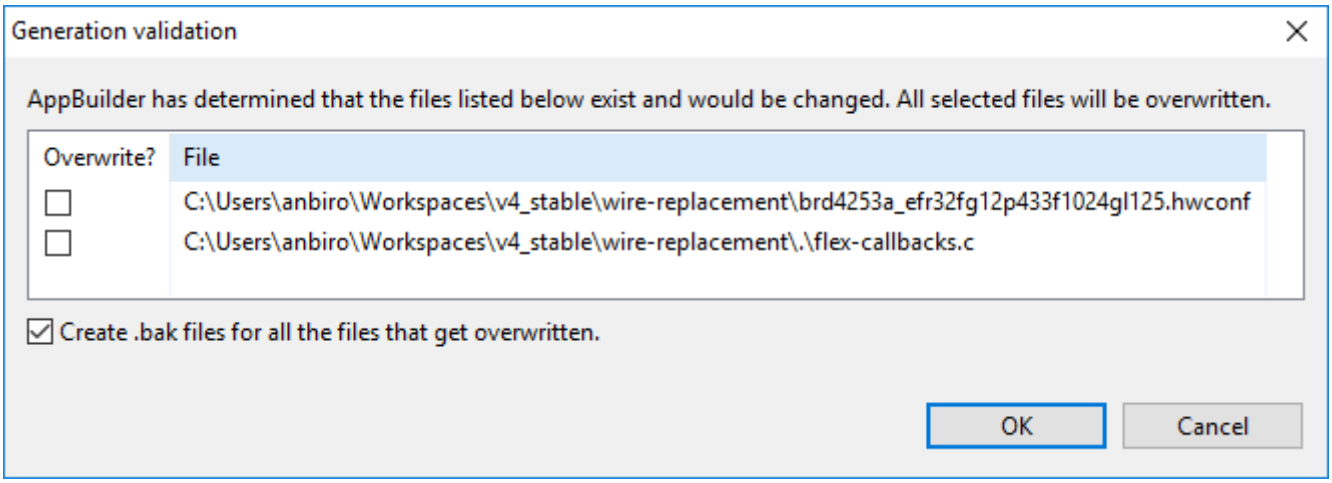
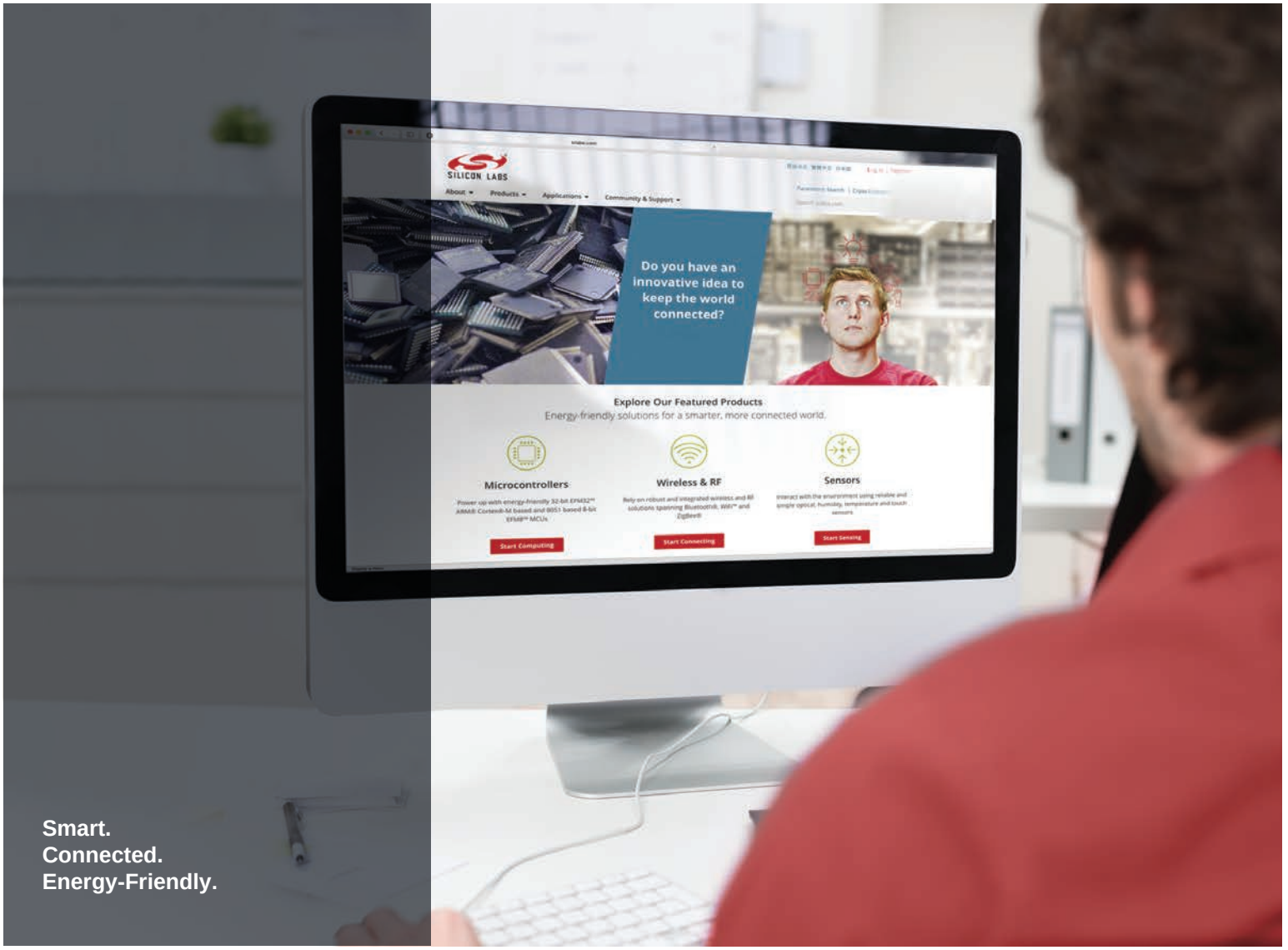


Figure 3.2. Generation Validation Window

This will let you decide which files should be overwritten. Some files, like `flex-callbacks.c` or the `.hwconf` files are not overwritten by default: these files should only be overwritten when you create an application from scratch. If you are instead modifying an existing application (or generating an example), these files likely include customized content, so overwriting them would reset/revert them to their default versions.

4. Next Steps

This document provides a general overview of the workflow and resources available to support developers of Connect-based applications. Consult the remaining chapters of the *Connect User's Guide* series for in-depth guidance on the multiple components to consider when developing Connect-based applications.



Products
www.silabs.com/products

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