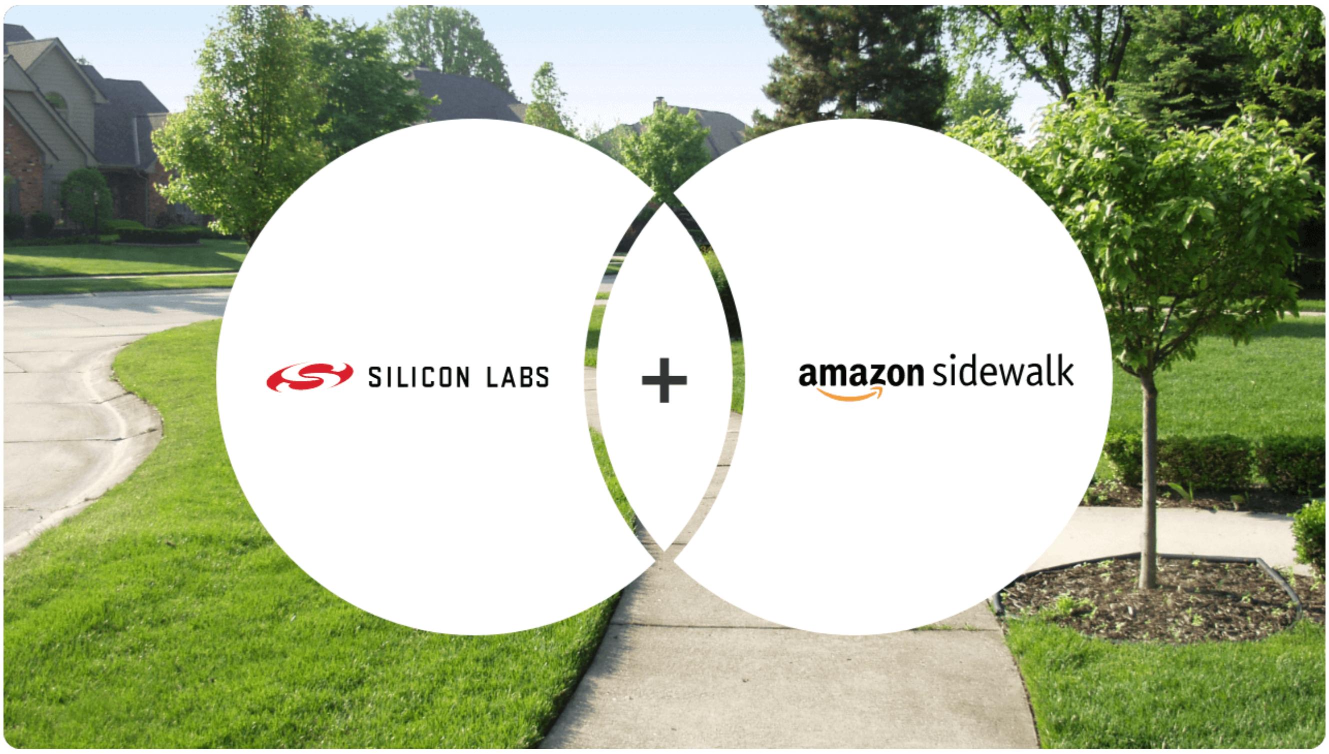
Amazon Sidewalk

Developing with Amazon Sidewalk

Developing with Amazon Sidewalk

Amazon Sidewalk is a shared wireless network that uses Amazon Sidewalk Bridges, such as compatible Amazon Echo and Ring devices, to enable communication among devices communicating on the network. Amazon Sidewalk enables reliable, low-bandwidth, and long-range connectivity at home and beyond. It connects IoT devices and applications such as outdoor lights, motion sensors, and location-based devices. It uses Bluetooth Low Energy for short-distance communication and CSS and FSK radio protocols at 900 MHz frequencies to cover longer distances.



These pages are intended for those who are actively exploring or already developing an application using the Silicon Labs integrated solution for Amazon Sidewalk. Contents focus on application development. For a high-level overview of the entire process, from buying parts to manufacturing and deployment, see the Amazon Sidewalk Developer Journey with Silicon Labs.

For our recommended Amazon Sidewalk development platform: See the Silicon Labs Pro Kit for Amazon Sidewalk.

To see the Silicon Labs Amazon Sidewalk integrated solution in action: Follow the Pro Kit out-of-the-box demo instructions.

For background about the protocol: The Amazon Sidewalk Protocol Overview discusses important elements of the specification.

To start your Amazon Sidewalk development using Simplicity Studio example applications: See the Getting Started Guide.

If you are already in development: See the Developer's Guide for resources to support crucial aspects of your development effort. You can also refer to the Amazon Sidewalk Sid API Developer Guide.

Out-of-the-Box Demo

Running the "Out-of-the-Box" Demo with Silicon Labs Pro Kit for Amazon Sidewalk

Congratulations on your purchase of the Silicon Labs Pro Kit for Amazon Sidewalk! This guide will help you get the most from the "out-of-the-box" demo application that is pre-installed on the radio boards in your new kit.

- Prepare the kit hardware: Describes your options with the Pro Kit hardware, and explains how to prepare your kit to run the demo.
- Ensure Access to the Amazon Sidewalk Network: Provides details on connecting the endpoint created by the demo application with the Amazon Sidewalk network through an in-range Sidewalk gateway.
- Run the Demo: Describes how, with the endpoint ready and a Sidewalk gateway nearby, you can establish your Sidewalk endpoint and access the associated AWS application from your mobile device.
- Going Further: Includes "next steps" to explore once you've witnessed the ease with which the Sidewalk network allows you to interact between an endpoint and the cloud, along with helpful tips if your demo did not operate smoothly.

Kit Preparation

Prepare the Kit Hardware

The Silicon Labs Pro Kit for Amazon Sidewalk includes nearly everything you need to run the "out-of-the-box" demo application.

> ⓘ INFO ⓘ: A USB-C cable, required to power the device, is *not* included in the kit - you must use a cable that is appropriate for your power source.

Main Boards and Radio Boards

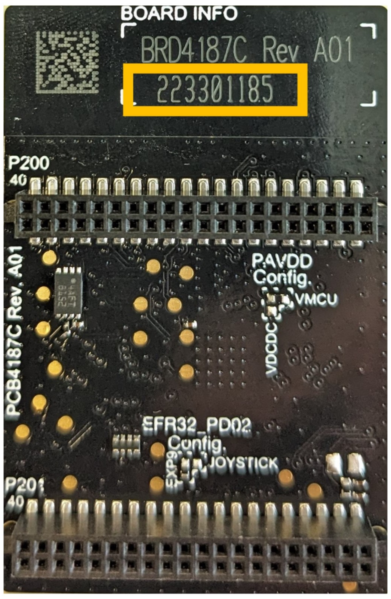
Required kit items for the "out-of-the-box" demo are:

- Wireless Pro Kit Main Board (BRD4002A), which provides power and peripherals to the radio boards
- Either of the two radio boards:
  - EFR32xG24 2.4 GHz 20 dBm Radio Board (BRD4187C)
  - KG100S Sidewalk Module Radio Board (BRD4332A) + 915Mhz antenna

Demo firmware is pre-flashed on both the EFR32xG24 and the KG100S Module radio boards. Silicon Labs recommends using the EFR32xG24 when you first run the demo, as this version presents a more complete experience. The differences are:

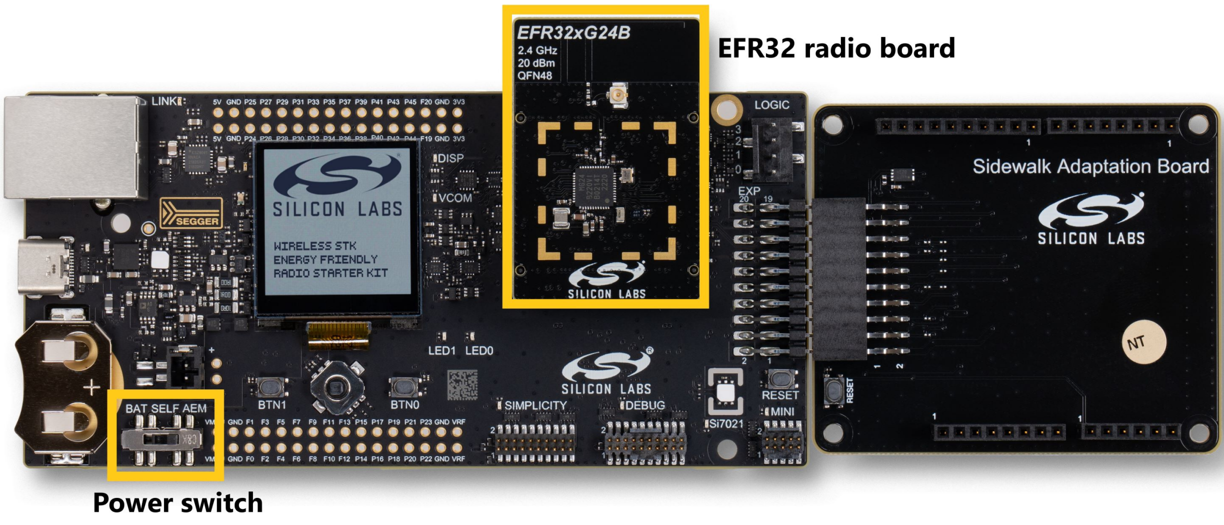
1. EFR32xG24 Radio Board - this demo app drives the LCD on the main board to display a QR code for your mobile device to scan. This application uses only Bluetooth Low Energy (BLE) to communicate with the gateway.
2. KG100S Radio Board - the QR code is printed on a sticker on the KG100S radio board. Make sure to power up the board first before scanning the QR code. This application communicates with the gateway over both BLE and the sub-GHz FSK radios.

As an alternative to scanning the QR code, the radio board serial number can be entered in Silicon Labs' "out-of-the-box" demo Web interface at silabssidewalkdemo.com. The radio board serial number is located behind the board as shown in the following picture.



Assembling the Demo Hardware

Next, install the radio board on the main board. Take care to align the contact points between the radio and main boards. For reference, the **EFR32xG24** is shown mounted on a main board.



Ensure that the power switch is in the **AEM** position, which supports supplying power to the main board as described in the next section. The additional Sidewalk Adaptation Board shown in the image above is not needed for this demo.

If you are using the KG100S Module, you must also connect the 915 MHz antenna onto that radio board.

Powering the Demo Hardware

With the power switch in the **AEM** position, the kit hardware is powered through the main board's USB-C connector. Connecting the main board to a USB power source starts the demo, so first check for network availability as described in the next step.

For the purposes of the "out-of-the-box" demo, only bus power is required - enumeration with a USB host is not necessary.

Next Step: Sidewalk Network Access

With your kit assembled and a USB-C power source available, next confirm Amazon Sidewalk Network Availability at your location.

Sidewalk Network Access

Ensure Access to the Amazon Sidewalk Network

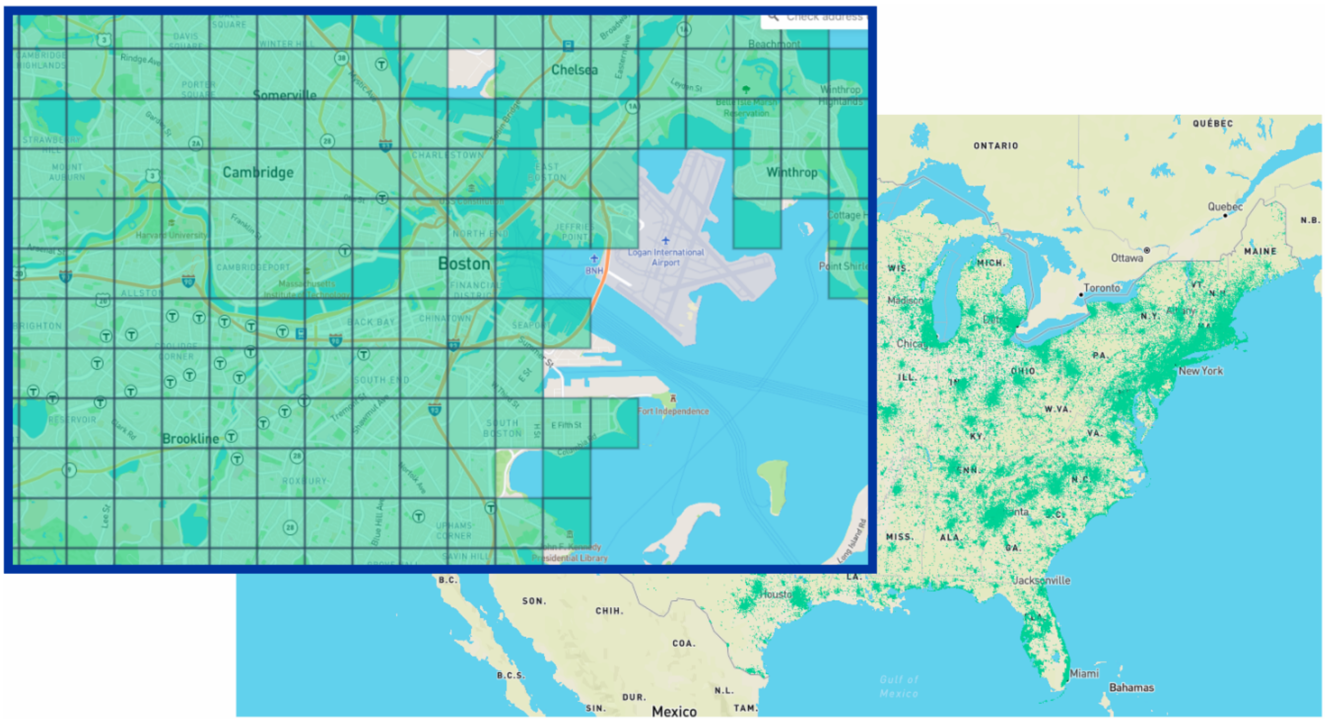
When it executes, the "out-of-the-box" demo firmware on the Silicon Labs Pro Kit radio board forms a Sidewalk endpoint. In order to reach the cloud, this device must be in range of a suitable Amazon Sidewalk gateway.

Sidewalk Network Coverage

Amazon currently supports the Amazon Sidewalk network only on gateways that communicate with AWS from US-based IP addresses.

> ⚠ WARNING ⚠: Regional regulations may also restrict unauthorized usage of the Sidewalk-supported sub-GHz bands outside of the United States. Consult local laws and ensure compliance (radio isolation by RF chamber, etc.) as needed. See Using Sidewalk Gateways Outside the USA for more information.

You may already have access to the Amazon Sidewalk network at your location. Amazon provides a Sidewalk Network Coverage map you can use to see if this is likely: https://coverage.sidewalk.amazon



Set up a Gateway

If you aren't already in range of a Sidewalk gateway, you can easily create your own.

The list of Amazon Sidewalk gateways and their supported capabilities is available at the following link:
https://docs.sidewalk.amazon/introduction/sidewalk-gateways.html

The Amazon Echo 4th generation is the recommended Amazon Sidewalk gateway for development purposes. Silicon Labs validates the Amazon Sidewalk software development kit against this product.

Follow the standard Echo product configuration procedures, and make sure to enable Sidewalk support on your gateway so it will access the Sidewalk network.

> ⓘ INFO ⓘ: To enable Sidewalk support visit Enable or Disable Amazon Sidewalk for Your Account for more help.

For more information on this topic, see the Getting Started Prerequisites page.

Next Step: Run the Demo

After you have access to the Amazon Sidewalk network, you are ready to Run the Demo.

Run the Demo

Running the Out-of-the-Box Demo

With your chosen radio board mounted and an Amazon Sidewalk gateway in BLE range of your device, you can run the demo by simply connecting USB-C power to the main board.

At power on, the pre-installed demo application forms a Sidewalk endpoint that automatically registers with the network through the nearest gateway. Amber LED(s) on the main board then blink periodically to indicate that the endpoint is awaiting time sync with the Sidewalk network. Once properly synchronized, the LED(s) stop blinking and the endpoint associates with a web application in the cloud, allowing you to interact with the endpoint over the Sidewalk network.

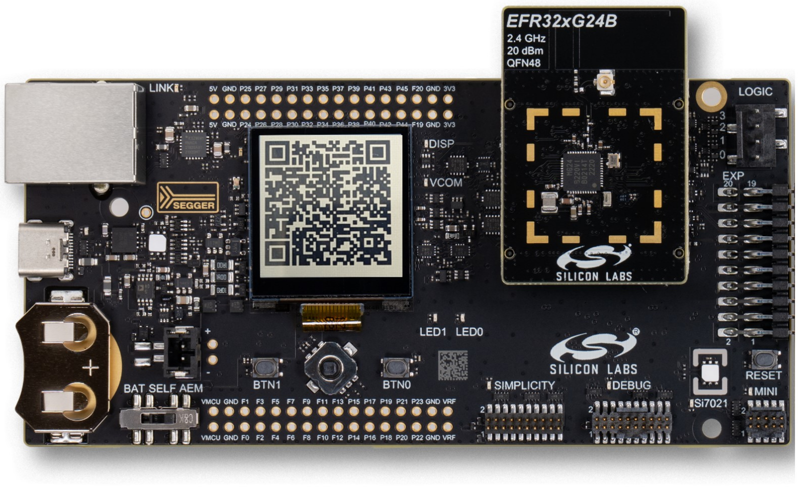
This entire process occurs within about a minute (or less) of powering on your device. If you are using the EFR32xG24 radio board, you will see a QR code displayed on the LCD. If you do not see a QR code (or if, when using the KG100S, LED1 continues to blink) after a minute or two, see Troubleshooting the OOB Demo.

Accessing the Demo Application in the Cloud

How you access the cloud application differs between the two radio boards in the kit. Each approach is described in the following sections.

When Using the EFR32xG24

In this version of the demo, the EFR32xG24 drives the LCD on the main board. Once the web application is prepared, a QR code is displayed on the LCD.



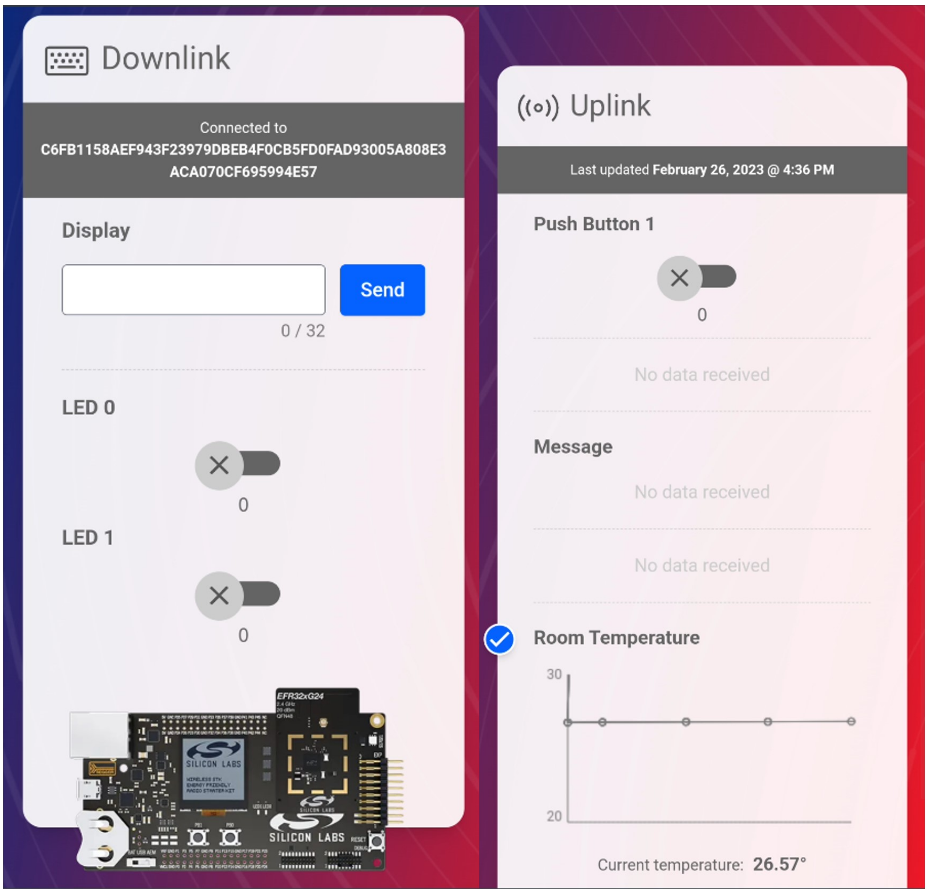
Use your mobile device or webcam to read this QR code and access the embedded URL in a browser. This page loads the demo web application already associated with your Sidewalk end device.

When Using the KG100S Sidewalk Module

A QR code sticker was applied to the KG100S device during manufacturing that already encodes the web application URL. After powering on the device, once LED1 stabilizes you can read the embedded URL and access the web application in your browser.

How to use the Cloud Demo Application

The cloud application is divided into two main sections, each providing a different way to interact over the Sidewalk network. These features are described below.



Downlink Section

These elements send data from the cloud down to the end device. For the xG24 radio board, you have access to the following capabilities:

- **Display** - If your demo is using the onboard LCD, type a message in this field and click **Send**. Your message is sent to the endpoint and appears on the LCD.
- **LED 0** - Tap to flip this toggle on or off. For the EFR32xG24, LED0 on the main board updates to your desired state. The KG100S radio board does not use this LED.
- **LED 1** - Tap to flip this toggle on or off. LED1 on the main board updates to your desired state.

Uplink Section

These elements display data sent from the end device up to the cloud. For the xG24 radio board, you have access to the following capabilities:

- **Push Button 1** - Press the associated button on your main board. The graphic toggles between 0 and 1 with each button press.
- **Room Temperature** - Ambient temperature is sampled by the endpoint, sent through the Sidewalk network to the cloud, and plotted over time on this Room Temperature chart. The temperature sensor is the Si7021 mounted on the main board.

For the KG100S radio board, dummy temperature data is sampled by the endpoint.

Next Step: Going Further

When you're ready to move beyond the out-of-the-box demo and more fully explore the Amazon Sidewalk developer experience provided by Silicon Labs, visit the Going Further page. You'll find resources to help you move beyond the demo, along with guidance to troubleshoot any issues you may have.

Going Further

Going Beyond the Demo

The Silicon Labs Pro Kit for Amazon Sidewalk demo was designed to provide users with effortless access to the kind of streamlined experience that Amazon Sidewalk can provide for your customers.

Silicon Labs supports Amazon Sidewalk with a set of tools, capability, and documentation that similarly lowers the barriers of entry to Amazon Sidewalk application development.

The following resources are great places to start if you're ready to learn more.

Developer Resources

- Getting Started Guide - Step-by-step journey through AWS (deploying application, profile and device creation) and building an example Amazon Sidewalk application in Simplicity Studio.
- Developer's Guide - Goes beyond the "Getting Started" introduction to deep-dive into the most important aspects of embedded development for Amazon Sidewalk with Silicon Labs.
- Amazon Sidewalk Protocol Overview - Review of many relevant technical elements of the Amazon Sidewalk network.

A Note on Preserving the Demo Functionality

The out-of-the-box demo relies on device-specific credentials pre-flashed to the USERDATA page on your device. This page is not affected by mass-erase operations, but *can* be explicitly erased by a targeted page erase.

Typical erase/flash cycles as you repurpose the kit radio boards during development are of no concern. In fact, Credentials Backup/Restore Feature describes a process by which you can restore a working out-of-the-box demo application after a mass-erase. However, care should be taken to NOT perform a page erase of USERDATA, or the out-of-the-box demo cannot be restored.

Troubleshooting the Out-of-the-Box Demo

If you encountered problems running the demo, the following guidance may help get things back on track:

- Kit Setup
  - Ensure the main board power switch is in the **AEM** position.
  - Disconnect USB-C power, then re-seat the radio board.
- Board-Specific
  - If not already, try using the EFR32xG24 radio board. This demo version relies only on BLE, and provides feedback at various stages on the LCD.
  - If using the KG100S radio board, install the 915 MHz antenna found in your kit.
- Network-Specific
  - If relying on ambient Sidewalk network coverage, obtain your own gateway to ensure network access.
  - If using your own gateway, confirm that Sidewalk is **enabled**, and verify the gateway has access to the internet (try asking *"Alexa, are you online?"*) from a US-based IP address.

If these all fail, review the Getting Started Guide. This will incrementally build what should become a known-good baseline (verifying kit hardware integrity and Sidewalk network access along the way).

For Silicon Labs Technical Support, contact us on our Support Platform.

Getting Started

Getting Started with Amazon Sidewalk Development in Simplicity Studio

- **Prerequisites**: Describes the hardware and software prerequisites for starting your development.
- **Create and Compile your Sidewalk Application**: Once Sidewalk is added to your GSDK, you can choose a sample Sidewalk application to compile and flash on your device.
- **Provision your Amazon Sidewalk Device**: Now your application is running on your endpoint, and you can create the application server on AWS and add the corresponding certificates to your device.
- **Interacting with the Cloud**: As the last step, you can send and receive messages between your endpoint and the server application in AWS.

For Silicon Labs Technical Support, contact us on our Support Platform.

Prerequisites

Prerequisites

To develop for Amazon Sidewalk on Silicon Labs SoCs and modules, you need the hardware and software resources detailed on this page.

Hardware

Silicon Labs Wireless Development Hardware

Silicon Labs produces many different kits to support development for a broad range of wireless technologies. New users can jump-start their journey with a Silicon Labs Pro Kit for Amazon Sidewalk. While experienced users may have much of the required hardware on-hand already, Silicon Labs recommends they too start with Silicon Labs Pro Kit for Amazon Sidewalk.

For Amazon Sidewalk solutions, the typical Silicon Labs development hardware is a wireless main board paired with a radio board that supports Sidewalk. The Sidewalk-specific wireless technologies supported in this configuration (Bluetooth Low Energy (BLE), sub-GHz FSK, and sub-GHz CSS) depend on the radio board, as shown in the following table. Note that for sub-GHz support, some devices will require both an EFR32 radio board and an additional Semtech LoRa shield with the appropriate adapter board. Advanced users who have already bought a Pro Kit or have a WSTK/WPK can augment their existing hardware with additional radio boards as needed.

The following list and table summarizes these requirements:

- A Sidewalk-supported Radio board:
  - EFR32xG21B Radio Board (BRD4181C)
  - EFR32xG23B Radio Board (BRD4204D, BRD4210A, BRD4264C, BRD4263C, BRD4204C)
  - EFR32FG23 868-915 MHz +14 dBm Dev Kit (BRD2600A)
  - EFR32MG24B Radio Board (BRD4186B, BRD4186C, BRD4187B, or BRD4187C - included in the Silicon Labs Pro Kit for Amazon Sidewalk)
  - EFR32xG28B Radio Board (BRD2705A, BRD4400A, BRD4400B, BRD4400C, BRD4401A, BRD4401B, BRD4401C)
  - KG100S Module Radio Board (BRD4332A, included in the Silicon Labs Pro Kit for Amazon Sidewalk)
- A Wireless Starter Kit Main board (BRD4001x), or a Wireless Pro Kit Main Board (BRD4002x) included in the Silicon Labs Pro Kit for Amazon Sidewalk. *(Note: a main board is NOT required when using a Dev Kit from the list above)*
- For sub-GHz applications: A 915MHz antenna (included in most sub-GHz kits) should be installed on the appropriate connector. If using EFR32xG21 or EFR32xG24, a Semtech SX1262MB2CAS LoRa shield and Sidewalk Adaptation Board (BRD8042A, included in the Silicon Labs Pro Kit for Amazon Sidewalk) are required.

Radio Board Model	BLE	FSK	CSS
EFR32xG21	x		
EFR32xG21 + Semtech shield	x	x	x
EFR32xG23		x	
EFR32xG24	x		
EFR32xG24 + Semtech shield	x	x	x
EFR32xG28	x	x	
KG100S	x	x	x

ⓘ INFO ⓘ: Amazon Sidewalk support requires that target hardware have Secure Element (SE) firmware versions of at least v1.2.9 for xG21 SoCs, v2.1.7 for xG24 SoCs, and v1.2.9 for KG100S Modules. Follow

> instructions in Section 4.4 of AN1222: Production Programming of Series 2 Devices to update the SE firmware.

> ⚠ WARNING ⚠: EFR32xG21B and EFR32xG28 radio boards are available with devices of various flash sizes. Minimum flash size to run the Hello Neighbor example applications is 768 kB. This restriction does not apply to EFR32xG23 devices, which do not support BLE.

For simplicity, this *Getting Started Guide* focuses on the **EFR32MG24** and demonstrates how to use the **Amazon Sidewalk - SoC Hello Neighbor** example application provided in the SDK for Amazon Sidewalk.

Amazon Sidewalk Gateway

The Sidewalk protocol requires a gateway to provide endpoints access to the AWS cloud. Several Amazon products can act as a gateway. These products have different functions and varying support for Amazon Sidewalk network features. The list of Amazon Sidewalk gateways and their supported radio capabilities is available at the following link: https://docs.sidewalk.amazon/introduction/sidewalk-gateways.html

The Amazon Echo 4th generation is the recommended Amazon Sidewalk gateway for development purposes. Silicon Labs validates the Amazon Sidewalk software development kit against this product.

Developers are advised to use their own gateway, as this affords the greatest control over providing consistent network access during the development phase. However, you may already have access to the Amazon Sidewalk network at your location. Amazon provides a Sidewalk Network Coverage map you can use to see if this is likely: https://coverage.sidewalk.amazon

Sidewalk Gateway Setup

Follow the standard product installation procedures to set up a new gateway. Configuring your gateway with Sidewalk support depends on a few additional requirements:

- The gateway must be set up using a US-based Amazon account (see Change your Amazon Account Country to adjust the Country setting for an existing account)
- Amazon Sidewalk must be enabled on the Amazon account (see Enable or Disable Amazon Sidewalk for Your Account for more help)
- The gateway must have a US-localized IP address, and be configured with a US-based location (see Change Your Alexa Device Location if needed)

Using Sidewalk Gateways Outside the USA

> ⚠ WARNING ⚠: Amazon Sidewalk is available only in the United States of America. To the extent that any Sidewalk gateway functionality might be used outside of the U.S., it should be used ONLY for Amazon Sidewalk endpoint development purposes. In addition, Silicon Labs recommends that you consult with your local regulatory bodies and check if the gateway is allowed to operate its radio in your locale, as U.S. license-free band devices, only for development. Developers are solely responsible for ensuring compliance with local regulations. Additionally, Silicon Labs recommends a shielded chamber or similar aparatus to capture and contain wireless signals within your development environment.

To enable operation outside of the US for your development, you need to use a VPN router that supports OpenVPN Client functionality in conjunction with a cloud VPN service provider. Refer to the following steps for more details. You are also solely responsible for compliance with any local regulations regarding VPN use.

Steps:

1. Buy a Wi-Fi router that supports OpenVPN. Not all routers support OpenVPN. Other VPN protocols such as L2TP, PPTP are not supported.
2. Sign up for a VPN service supporting OpenVPN.
3. Configure the Wi-Fi router as a VPN client using your VPN service account credentials and enable the VPN feature on the router.
4. Set up your Echo device, using the Alexa app for devices without screens and using the screen for Echo Show devices. See here for details.
5. Connect your Echo device via Wi-Fi to the SSID broadcast by the router running the VPN client.
6. Set up your Echo device up with a US account location. See here for details.

> ⓘ INFO ⓘ: VPN clients running on a laptop or mobile device or using any other VPN protocol (such as L2TP or PPTP) are not supported.

Amazon Frustration-Free Setup (FFS)

A fully operational gateway is linked to an Amazon account - usually during initial setup using the Alexa app. However, this device-account linking can instead be initiated at time of purchase if you buy the gateway from Amazon and check the box labeled "*Link device to your Amazon account to simplify setup*." This account linking has implications for Sidewalk feature support and gateway setup:

- Currently, FSK support is enabled on only one (by default, the first) compatible gateway linked to an account.
- If you purchase a gateway device using an Amazon account other than the one you plan to use for testing and in the Alexa app to configure the gateway, do NOT check the "*Link device to your Amazon account...*" box. Doing so may prevent the successful setup of your gateway until you request Amazon Customer Support to remove the original device-account link. More info: Amazon Frustration-Free Setup Frequently Asked Questions.
- The "*Link device...*" checkbox also enables Amazon Wi-Fi Simple Setup, which can simplify getting your new gateway connected when powered on. However, if you plan to connect the gateway to a different SSID than you already use with other devices known to your Amazon account, the FFS-driven automatic selection of Wi-Fi networks can impede your efforts to connect the gateway to your preferred SSID.

Software

To get started with your Amazon Sidewalk development, you need:

- Simplicity Studio 5, with the SDK extension for Amazon Sidewalk installed (see version and installation guidance below)
- J-Link RTT Viewer (see version guidance below)

> ⓘ INFO ⓘ: Amazon Sidewalk support requires the Amazon Sidewalk SDK extension. The extension version 1.0.0 requires at least Simplicity Studio v5, the Simplicity Commander version included in that release, GSDK 4.2.2, and JLink version 7.84.

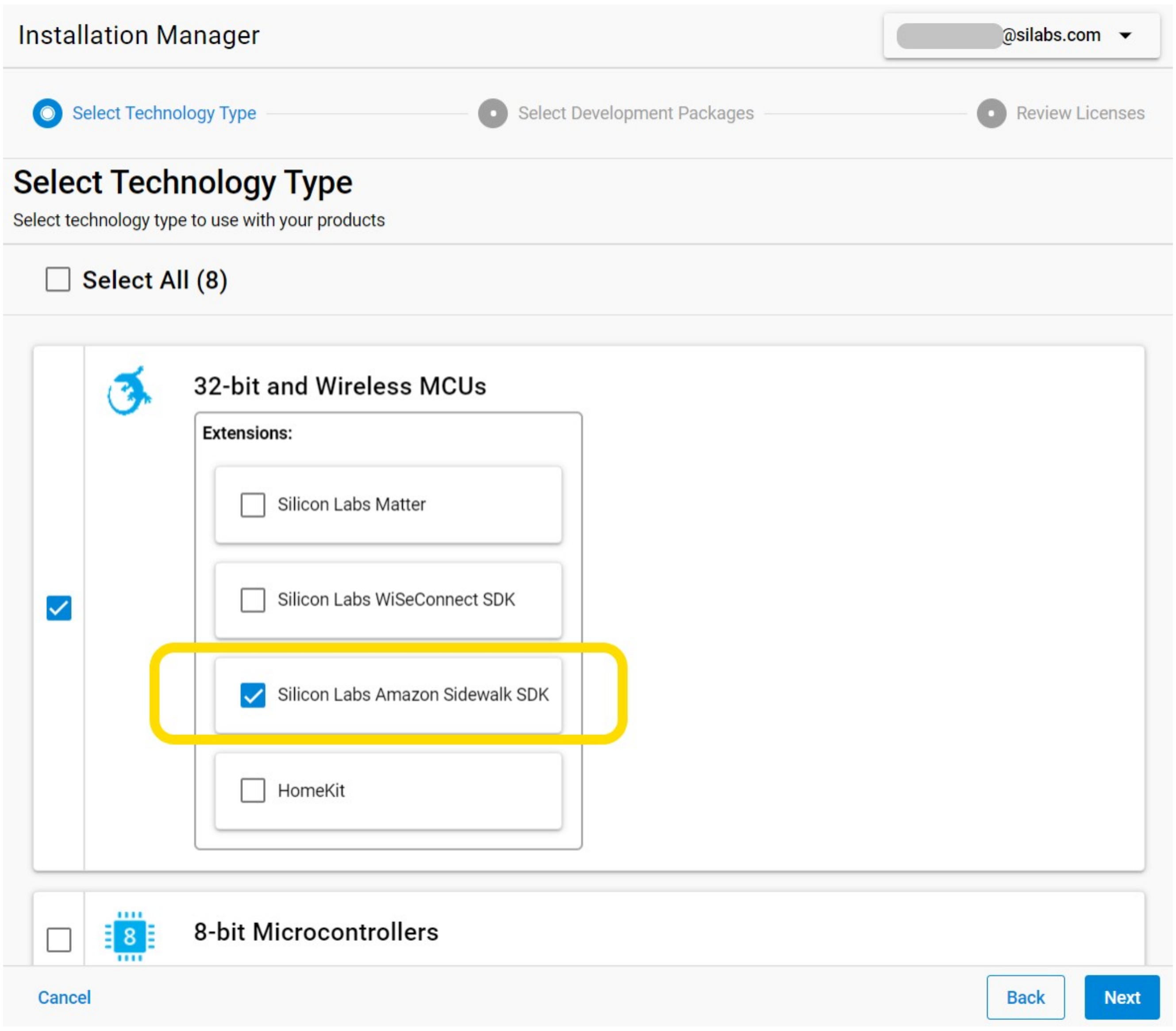
Silicon Labs SDK Extension for Amazon Sidewalk

During the initial install of Simplicity Studio, support for Amazon Sidewalk can be added by checking both the *32-bit and Wireless MCUs* and the *Amazon Sidewalk SDK* extension boxes in the Installation Manager as shown in the "step 1" figure below. Click **Next** to install the latest 32-bit and Wireless MCU GSDK and the Sidewalk SDK extension. If Simplicity Studio is already installed, follow step 1 below to add GSDK support for the first time. Alternatively, for installations that have already added at least one GSDK version, jump to step 2 to ensure you install Sidewalk support with the latest GSDK.

#1. Add GSDK and SDK extension for Amazon Sidewalk to an existing Simplicity Studio installation.

- Open Simplicity Studio 5.
- Click the **Install** icon on the toolbar and select **Install by technology type**.

- A **Select Technology Type** dialog opens. Select **32-bit and Wireless MCUs**, **Amazon Sidewalk SDK**, and click **Next** (see figure below).
- In the **Package Installation Options** dialog, select **Auto** and click **Next**.
- Wait for the installation to complete, click **Finish**, and go to step 2 to verify the installed GSDK and Sidewalk support.



Select Technology Type

Select technology type to use with your products

☐ Select All (8)

☑ 32-bit and Wireless MCUs

Extensions:

☐ Silicon Labs Matter

☐ Silicon Labs WiSeConnect SDK

☑ Silicon Labs Amazon Sidewalk SDK

☐ HomeKit

☐ 8-bit Microcontrollers

#2. Add or confirm the SDK extension for Amazon Sidewalk when GSDK is already installed.

- Open Simplicity Studio 5.
- Click the **Install** icon on the toolbar and select **Manage installed packages**.
- In the **Installation Manager**, go to the **SDKs** tab.
- On the **Gecko SDK - 32-bit and Wireless MCUs** card, verify that the latest version is installed.
- If a different version is installed, click **Add...**.
- On the **Versions** drop-down, select the latest version.
- Note the **location** of the new GSDK folder, you may want to refer to this in a later step.
- If not already selected, check the box next to **Amazon Sidewalk SDK**.
- Click **Finish**.

If needed, Simplicity Studio 5 now downloads and installs the GSDK and Sidewalk SDK extension.

After you have finished installing the GSDK, install the adapter pack for the Sidewalk Assistant:

- Click the **Preferences** icon on the toolbar.
- In the **Simplicity Studio** > **Adapter Packs** menu, click **Add...**.
- Browse to your freshly installed GSDK folder, then select the folder `extension/sidewalk/tools/sidewalk_assistant`.
- Click **Select Folder**.
- **Sidewalk Assistant** should appear in your adapter packs list.
- For macOS and Linux platforms you also need to give execution permissions to the binaries in `<your_gsdk_installation>/extension/sidewalk/tools/sidewalk_assistant/deploy/`.

AWS Command Line Interface (CLI)

To perform the operations that create and manage the cloud-based elements of your Amazon Sidewalk applications, your scripts and the Sidewalk Assistant use AWS CLI. With the AWS CLI, these tools can run commands that implement functionality equivalent to that provided by the browser-based AWS Management Console. Use the resources below to set up AWS CLI for these purposes.

> ⓘ INFO ⓘ: Note that Amazon has currently activated Sidewalk only for the North Virginia region ("**us-east-1**"). To support Sidewalk development (and operation), your AWS CLI and AWS web interface should be localized on this **us-east-1** region.

- Prerequisites to use the AWS CLI. To prepare your system to support AWS CLI, you must create users in your AWS CLI account. For convenience, Silicon Labs recommends using *long-term IAM credentials* (via IAM) to access AWS during your evaluation:
  1. Follow the guidance in Getting set up with IAM to create an AWS account if you do not already have one.
  2. Follow steps 1 thru 3 in Authenticate using long-term credentials to create an administrator IAM account (select the *AdministratorAccess* policy), get your access keys, and update the shared credentials file. (For additional assistance on these topics, see Getting started with IAM and Managing access keys for IAM users)
- Install AWS CLI. If AWS CLI is not already installed, install it as described here.
- Configure AWS CLI. Configure AWS CLI to leverage your IAM administrator account. (Sidewalk Assistant supports the recommended *credentials file* approach described in prior bullets and *environment variables*.)

Getting Support

Silicon Labs supports developers in many ways, including with documentation and active community forums where other users and Silicon Labs employees offer guidance to those in need.

- Amazon Sidewalk at Silicon Labs is the home page of the site that contains this getting started guide and other helpful resources
- Sidewalk Community Forum for all things Sidewalk-related
- Simplicity Studio 5 User's Guide documents the features and usage guidance for Simplicity Studio
- Simplicity Studio Forum for assistance with the development environment and resource management provided through Simplicity Studio
- What Do the Lights on Your Echo Device Mean? can help you understand the visual feedback from your Amazon Echo device
- Amazon Sidewalk Documentation provides additional details on Amazon Sidewalk

Taking the Next Step

Once you have completed the required hardware and software setup tasks, you can begin creating and compiling an Amazon Sidewalk Application.

Create and Compile an Application

Create and Compile your Sidewalk Application

Amazon Sidewalk Sample Applications

Silicon Labs SDK for Amazon Sidewalk includes multiple example applications, including:

- Amazon Sidewalk - SoC Hello Neighbor
- Amazon Sidewalk - SoC CLI
- Amazon Sidewalk - SoC OOB Demo (delivered in binary only)

When a Sidewalk-supported target device is selected, additional examples are available in the Simplicity Studio **Launcher** perspective, **EXAMPLE PROJECTS & DEMOS** tab. Type "sidewalk" into the "**Filter on Keywords**" field at top left, and press enter to view only Amazon Sidewalk examples.

For simplicity, this guide walks through **Amazon Sidewalk - SoC Hello Neighbor** example on a **EFR32xG24** radio board. You can then repeat this procedure with other examples in the SDK.

Create an Amazon Sidewalk Project

With the Sidewalk resources added to your Gecko SDK, reopen Simplicity Studio 5. You can now use the graphical interface to create a Sidewalk project.

- Mount the EFR32xG24 radio board onto the main board, then connect the assembly to your computer through the USB connector on the main board.
- A new entry should appear in the **Debug Adapters** view.
- Select the board and make sure your Gecko SDK with Amazon Sidewalk SDK extension installed is selected in the "Preferred SDK" section of the **General Information** card. If the "Secure FW" version depicted on the card is below the minimum SE FW for your target device specified by the Prerequisites page, click the nearby link to upgrade the SE FW before proceeding.
- Go to the **EXAMPLE PROJECTS & DEMOS** tab.
- Filter the example list by typing ***sidewalk*** in the "**Filter on Keywords**" field and press enter.
- Click **Create** next to the **Amazon Sidewalk - SoC Hello Neighbor** example.
- In the **New Project Wizard**, select the **Copy contents** radio button, then click **Finish**.

> ⓘ **INFO** ⓘ: Simplicity Studio and the GSDK support multiple toolchains in addition to the default GCC. However, for projects using the Amazon Sidewalk SDK extension, GCC is required.

Compile and Flash the Project

Simplicity Studio adds the project to the workspace folder. You can now compile and flash the project on the EFR32xG24 radio board.

- In the Simplicity IDE perspective, select the Sidewalk project (.slcp file).
- On the top menu click **Run**, then select **Debug**.
- Wait for the build and flash operations to succeed.
- In the **Debug** perspective, click **Run**.
- Select **Resume**.

ⓘ INFO ⓘ: It is a "best practice" to erase the main flash before writing new application firmware to your device. Simplicity Studio provides many ways to do so, including Simplicity Commander and the Flash Programmer.

⚠ WARNING ⚠: If using a radio board from the Silicon Labs Pro Kit for Amazon Sidewalk, the out-of-the-box (OOB) demo relies on device-specific credentials pre-flashed to the USERDATA page on your device. This page is not affected by mass-erase operations, but can be explicitly erased by a targeted page erase. Credentials Backup/Restore Feature describes a process by which you can restore a working out-of-the-box demo application after a mass-erase. However, care should be taken to NOT perform a page erase of USERDATA, or the OOB demo cannot be restored.

View the Application Logs through J-Link RTT

The UART interface is not always available to report traces from Sidewalk example applications. Instead, the applications leverage the J-Link RTT interface.

⚠ WARNING ⚠: J-Link RTT and Simplicity Studio use the same channel to communicate with the board. If you do not see logs in J-Link RTT, try closing and re-opening Simplicity Studio to reset the connection. RTT Viewer needs to be disconnected from a device to allow flashing again with commander or Simplicity Studio.

To set up the communication between your PC and the EFR32, follow these instructions:

- Install the J-Link RTT Viewer.
- Open the J-Link RTT Viewer.
- In the **Configuration** panel, **Connection to J-Link** section, select **USB**.
- In the **Specify Target Device** list, select the connected part. For the EFR32xG24 radio board BRD4187C, select EFR32MG24AxxxF1536.
- In the **Target Interface & Speed panel**, select **SWD** and **4000 kHz**.
- In the **RTT Control Block** panel, select **Auto Detection**.
- Click **OK**.

ⓘ INFO ⓘ: For the Quectel KG100S module, select EFR32BG21BxxxF1024 as the target device.

A terminal opens and the Sidewalk application traces are output as shown:

```
00> [00000000] <info> App - sid_ble application started
00> [00000050] <info> MFG NVM3 start info:
00>  OK
00>
00> [00000100] <info> KV NVM3 start info:
00>  OK
00>
00> [00000101] <info> MFG Store opened with 0 objects
00>
00> > [00000102] <error> Sall mgm core create failed: -8
00> [00000103] <error> App - failed to initialize sidewalk: -8
00> [00000103] <error> App - fatal error
```

The above console log indicates that the example application is running, but found no objects in the device's NVM3 flash area. For any Sidewalk application (including this one) to run properly, Sidewalk resources in the cloud must be prepared. Move on to the next section, Provision your Amazon Sidewalk Device, to provision your device and prepare your Amazon Sidewalk cloud resources to interact with it.

If you encounter any issues during this Getting Started procedure, see our troubleshooting page with solutions to common problems.

Provision your Device

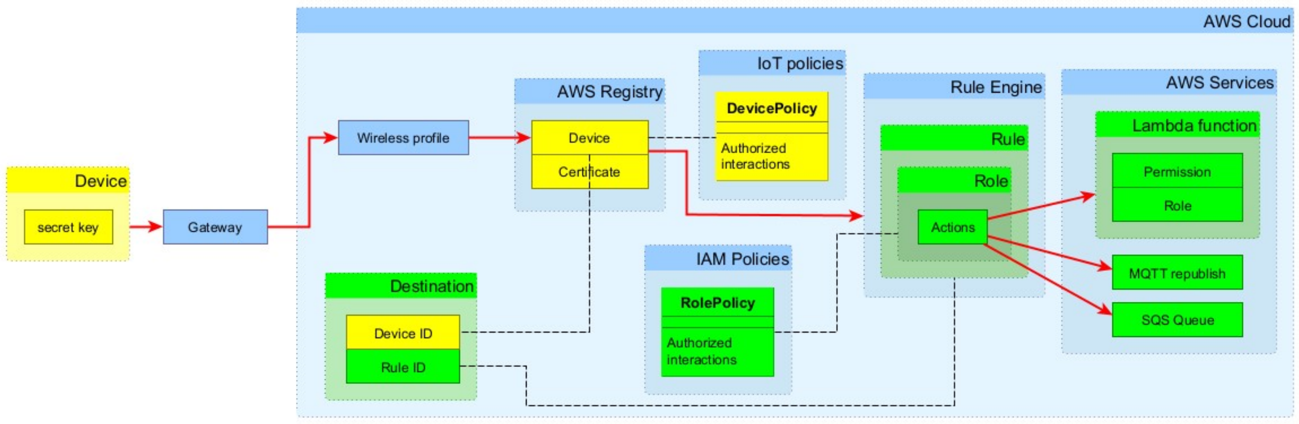
Provision your Amazon Sidewalk Device

Firmware running on an embedded device forms only part of an Amazon Sidewalk solution. Cloud resources must be created to recognize and process data moving through the network between AWS Services and your endpoint. Credentials must be generated and written to the device so that it is accepted and granted access to the Sidewalk network by in-range gateways.

AWS CloudFormation

AWS CloudFormation is a service that helps you model and set up your AWS resources so that you can spend less time managing those resources and more time focusing on your applications that run in AWS. You create a template that describes all the AWS resources that you want (like Amazon EC2 instances or Amazon RDS DB instances), and CloudFormation takes care of provisioning and configuring those resources for you. You do not need to individually create and configure AWS resources and figure out dependencies, because CloudFormation handles that on your behalf.

The Hello Neighbor example depends on AWS resources created using AWS CloudFormation. In the graphic below, green indicates the resources created by the CloudFormation script and yellow indicates the resources strictly associated with a device. The red arrows show the flow of information before an action can be performed on the cloud side.



Even with helpful task aggregator tools like CloudFormation, many steps are involved with the process to create and interlink these resources. Silicon Labs provides Sidewalk developers with a tool to rapidly accomplish these tasks for our example applications.

Sidewalk Assistant

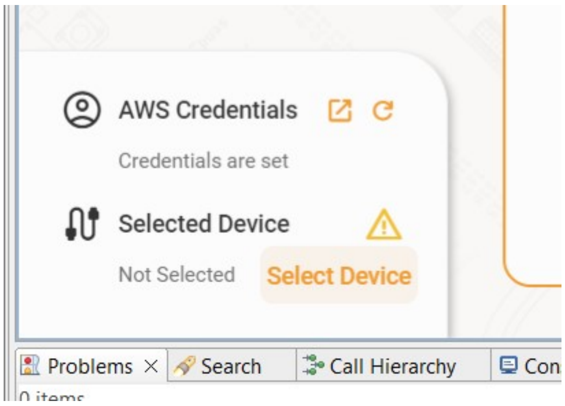
After creating your example project, when your new .slcp project file opens an additional tab **sidewalk.asconf** containing the **Sidewalk Assistant** tool also appears. If you had closed your project tabs you can reopen them by going to the root of the project folder in Project Explorer view, and double-clicking the .slcp and .asconf files.

The Sidewalk Assistant tool abstracts multiple underlying scripted interactions via AWS CLI (or manual interactions with the AWS Management Console) into an intuitive graphical user interface that vastly simplifies these operations. Using this tool allows you to focus on the high-level results of your development, rather than the details of each step. You can delve deeper into these details later, when you are ready to move beyond the simplistic AWS constructs of our example applications.

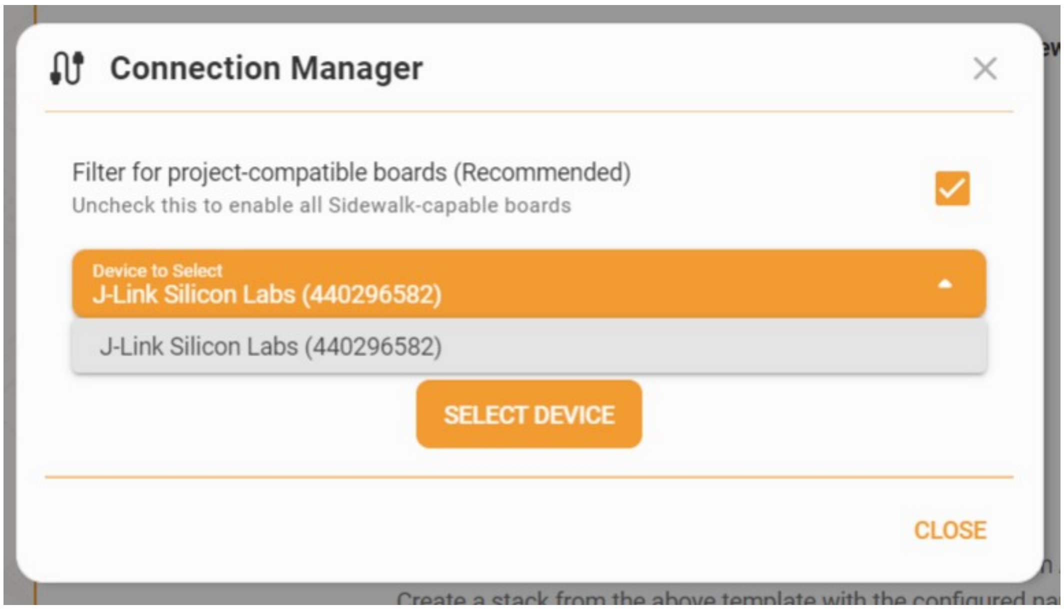
Sidewalk Assistant: AWS Credentials & Selected Device

The **sidewalk.asconf** file (in /config/sidewalk/ within the project directory) opens in the Sidewalk Assistant on project creation, or you can access the tool by navigating to and opening the .asconf file.

The tool automatically scans your system for appropriate AWS credentials, established when AWS CLI was installed and AWS accounts created previously, and indicates that no target device is yet selected:



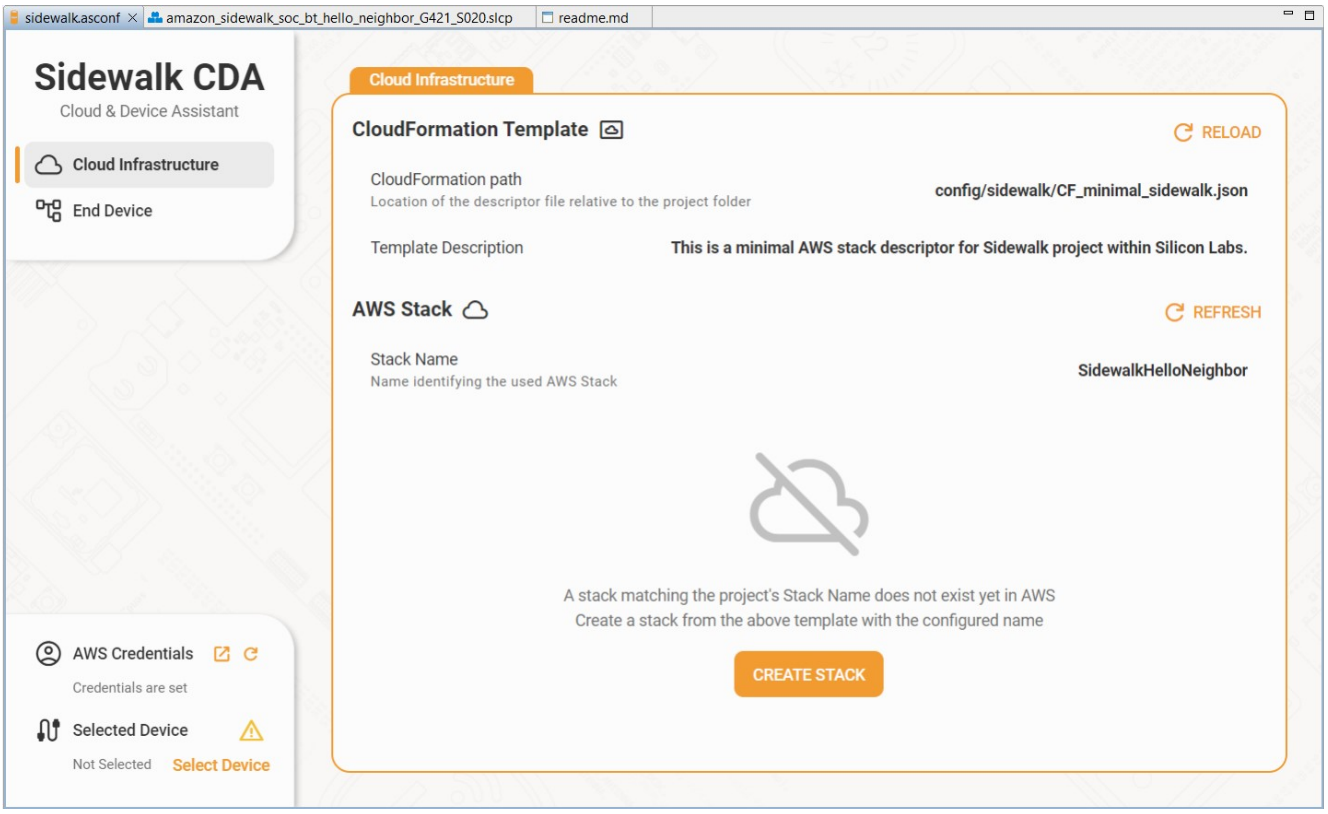
Click **Select Device** to open the Connection Manager dialog and choose from any of the attached devices (the checkbox at top right filters the list to those devices with Sidewalk support).


Connection Manager
Filter for project-compatible boards (Recommended)
Uncheck this to enable all Sidewalk-capable boards
Device to Select
J-Link Silicon Labs (440296582)
J-Link Silicon Labs (440296582)
SELECT DEVICE
CLOSE
Create a stack from the above template with the configured na

The Sidewalk Assistant contains two primary pages, the Cloud Infrastructure page and the End Device page.

Sidewalk Assistant: Cloud Infrastructure

Initially, for a new project the Sidewalk Assistant Cloud Infrastructure page may look like this:


sidewalk.asconf | amazon_sidewalk_soc_bt_hello_neighbor_G421_S020.slcp | readme.md

Sidewalk CDA
Cloud & Device Assistant

Cloud Infrastructure

CloudFormation Template

CloudFormation path
Location of the descriptor file relative to the project folder
config/sidewalk/CF_minimal_sidewalk.json

Template Description
This is a minimal AWS stack descriptor for Sidewalk project within Silicon Labs.

AWS Stack

Stack Name
Name identifying the used AWS Stack
SidewalkHelloNeighbor

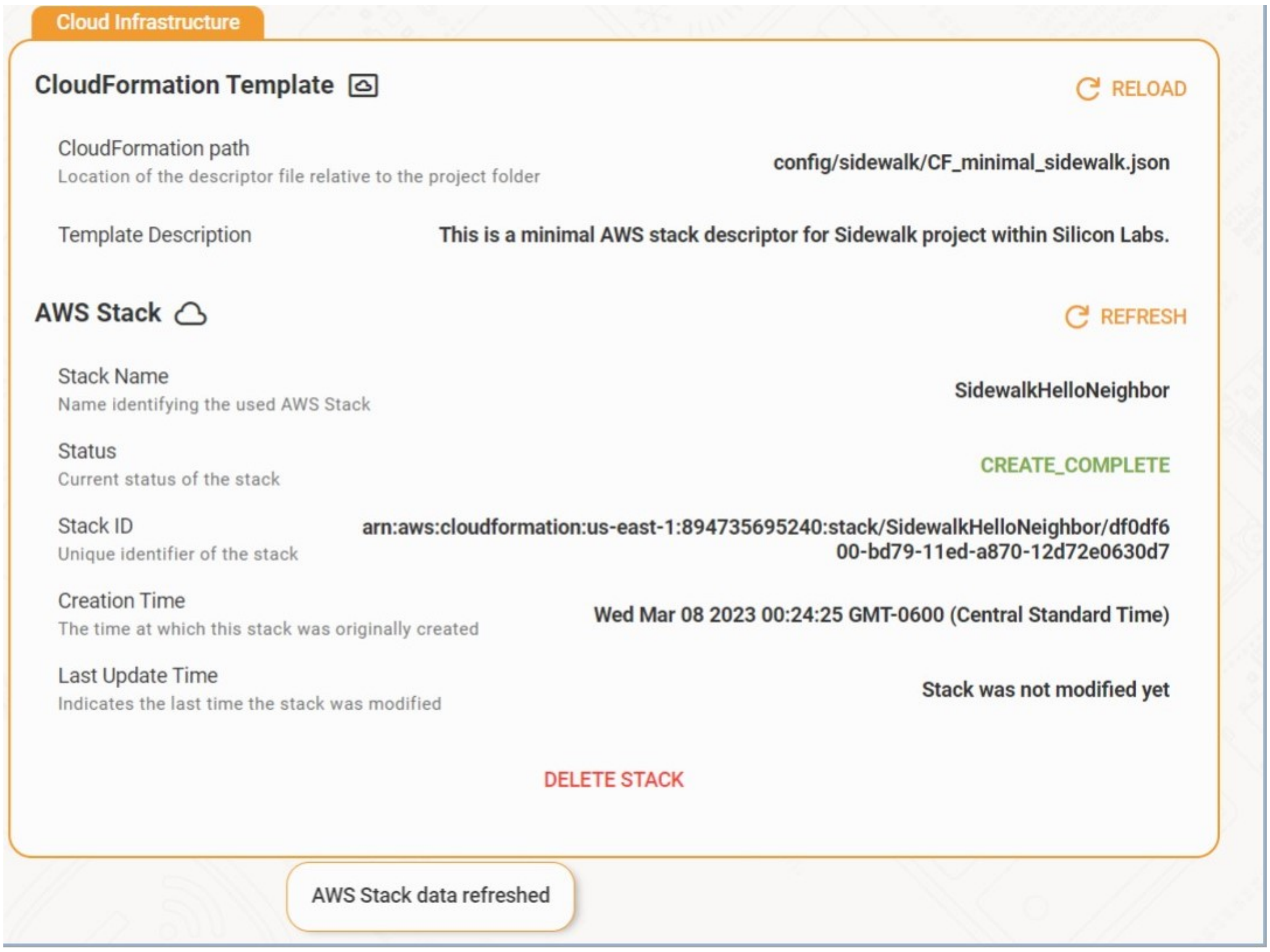
A stack matching the project's Stack Name does not exist yet in AWS
Create a stack from the above template with the configured name

AWS Credentials
Credentials are set

Selected Device
Not Selected

The CloudFormation Template used by this example is described in the top section. The AWS Stack name defined in the project is also visible.

Clicking **Create Stack** triggers the creation of the stack using AWS CloudFormation. The interface displays 'Creation in Progress' during the process. You may have to click the **Refresh Data** control multiple times, until the results are ready. When stack creation is successfully completed, the AWS Stack section now displays details about the new stack and a **DELETE STACK** control.



CloudFormation Template

CloudFormation path Location of the descriptor file relative to the project folder	config/sidewalk/CF_minimal_sidewalk.json
Template Description	This is a minimal AWS stack descriptor for Sidewalk project within Silicon Labs.

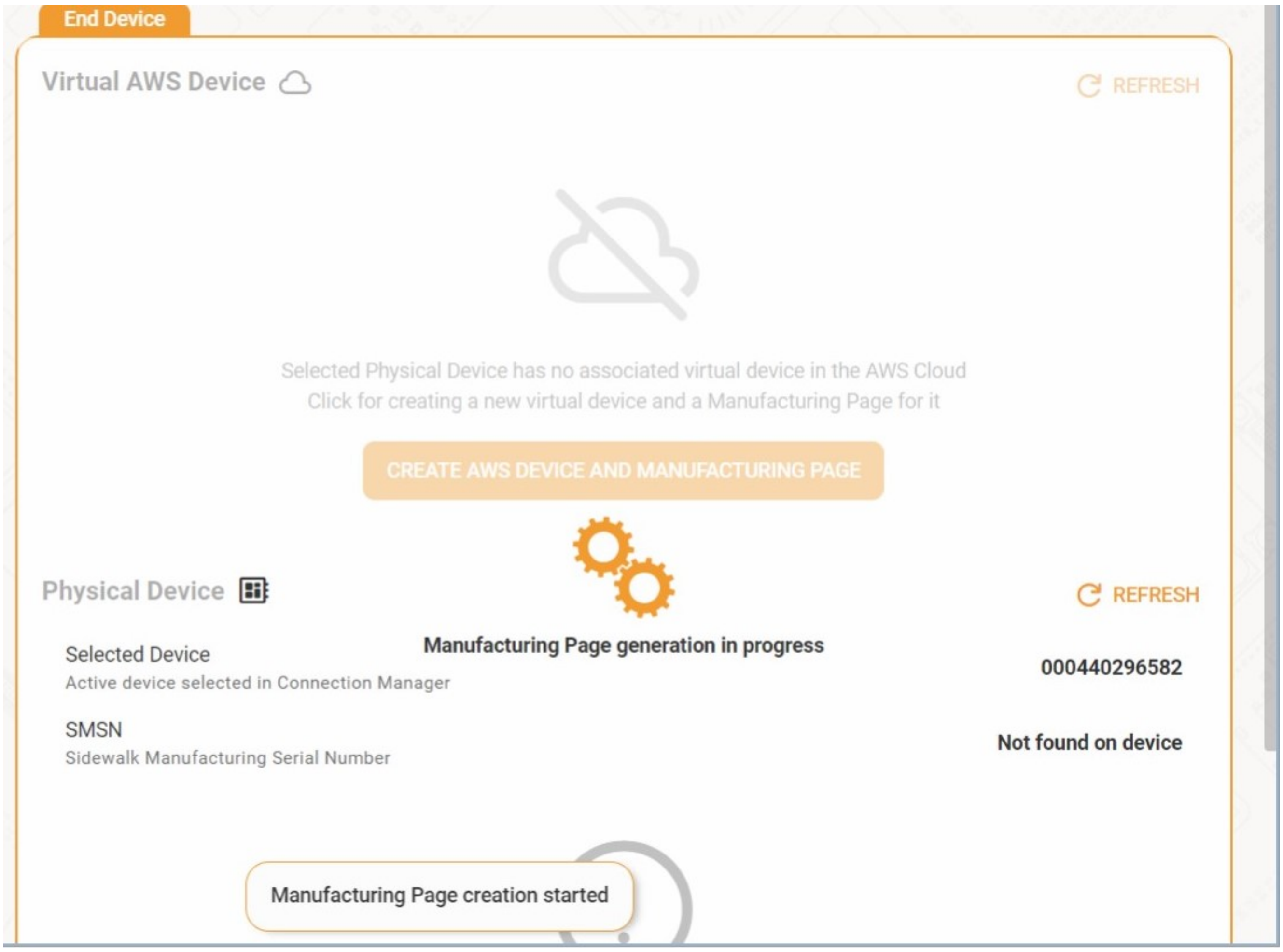
AWS Stack

Stack Name Name identifying the used AWS Stack	SidewalkHelloNeighbor
Status Current status of the stack	CREATE_COMPLETE
Stack ID Unique identifier of the stack	arn:aws:cloudformation:us-east-1:894735695240:stack/SidewalkHelloNeighbor/df0df600-bd79-11ed-a870-12d72e0630d7
Creation Time The time at which this stack was originally created	Wed Mar 08 2023 00:24:25 GMT-0600 (Central Standard Time)
Last Update Time Indicates the last time the stack was modified	Stack was not modified yet

Sidewalk Assistant: End Device

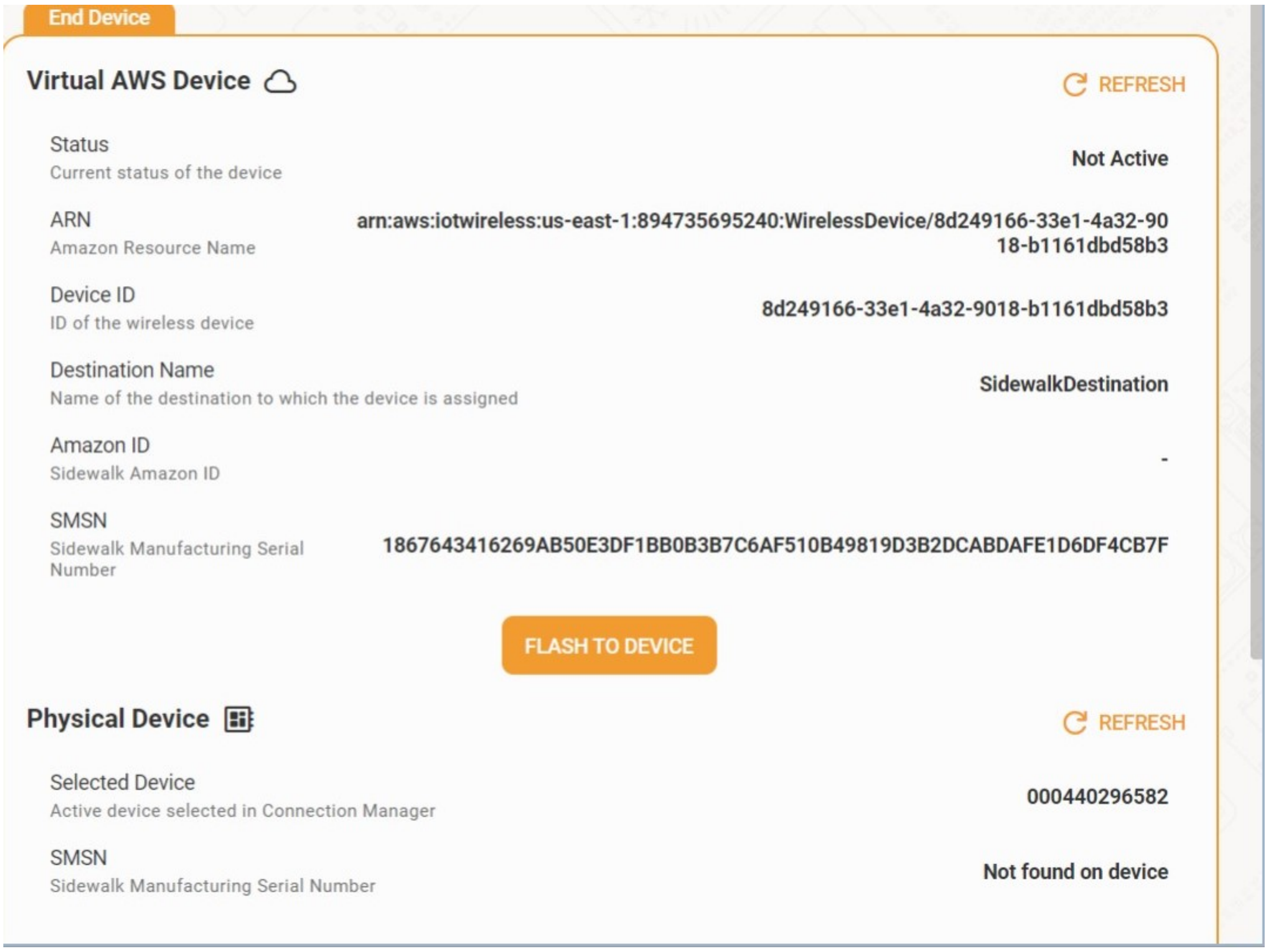
The cloud must be provided with a virtual representation of your device, and a "manufacturing page" must be created from the AWS resources that will equip the endpoint with necessary cryptographic resources to successfully authenticate and encrypt/decrypt traffic to/from your AWS resources in the Sidewalk network.

The manufacturing page is a `.s37` binary file containing the keys and certificates for your device that will be flashed to a specific address. (`.s37` files contain the program bytes and the target address.) To let the Sidewalk Assistant design your virtual device in the cloud and generate a manufacturing page for your physical endpoint, click the **CREATE AWS DEVICE AND MANUFACTURING PAGE** button.


End Device
Virtual AWS Device
REFRESH
Selected Physical Device has no associated virtual device in the AWS Cloud
Click for creating a new virtual device and a Manufacturing Page for it
CREATE AWS DEVICE AND MANUFACTURING PAGE
Physical Device
REFRESH
Manufacturing Page generation in progress
Selected Device
Active device selected in Connection Manager
000440296582
SMSN
Sidewalk Manufacturing Serial Number
Not found on device
Manufacturing Page creation started

When this first step is complete, the Virtual AWS Device section displays details of the newly-created virtual device.

The GUI also now provides a **FLASH TO DEVICE** control - click this to flash the manufacturing page to your endpoint. This complements the application image already flashed to the device in a previous step.



End Device

Virtual AWS Device

REFRESH

Status Current status of the device	Not Active
ARN Amazon Resource Name	arn:aws:iotwireless:us-east-1:894735695240:WirelessDevice/8d249166-33e1-4a32-9018-b1161dbd58b3
Device ID ID of the wireless device	8d249166-33e1-4a32-9018-b1161dbd58b3
Destination Name Name of the destination to which the device is assigned	SidewalkDestination
Amazon ID Sidewalk Amazon ID	-
SMSN Sidewalk Manufacturing Serial Number	1867643416269AB50E3DF1BB0B3B7C6AF510B49819D3B2DCABDAFE1D6DF4CB7F

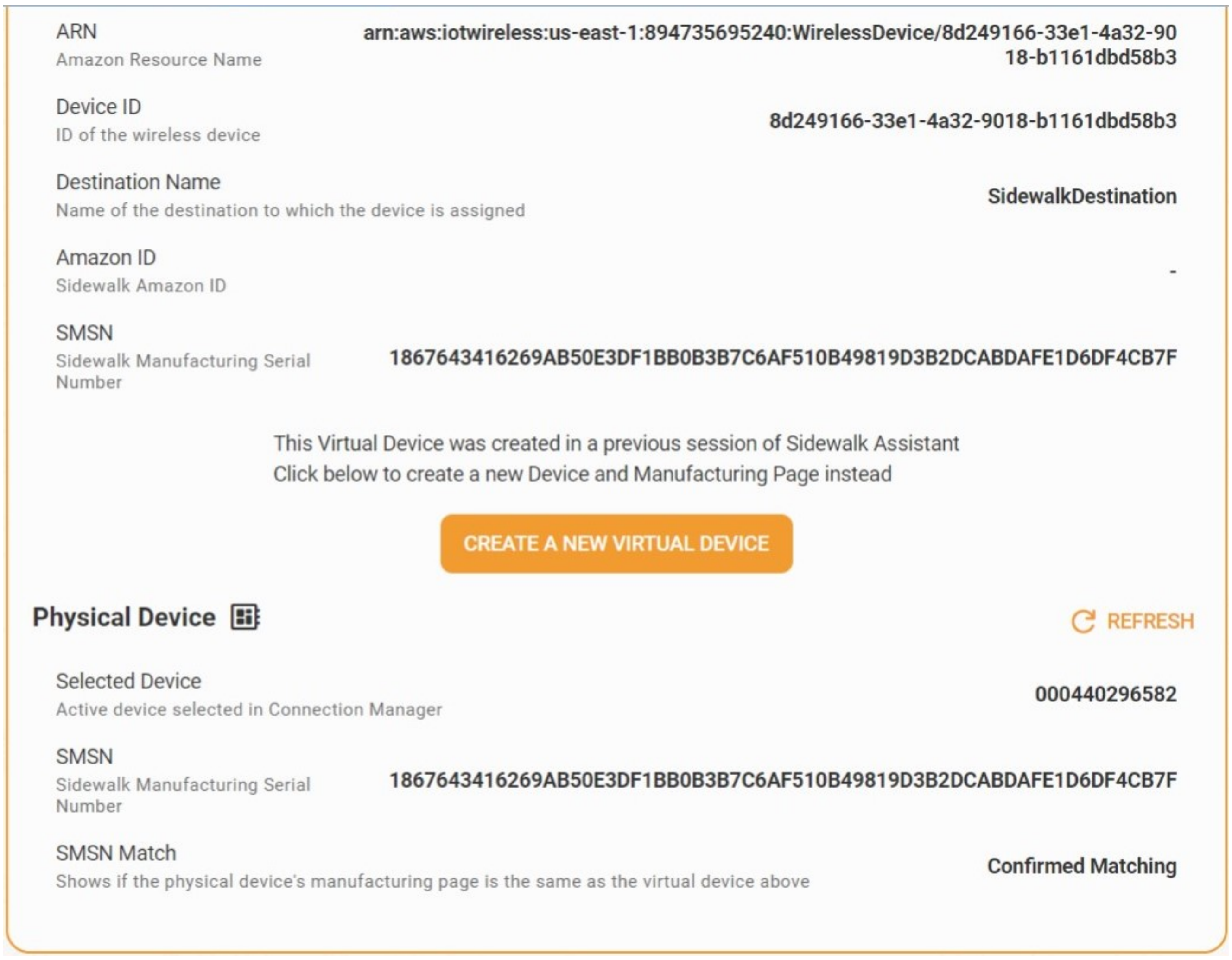
FLASH TO DEVICE

Physical Device

REFRESH

Selected Device Active device selected in Connection Manager	000440296582
SMSN Sidewalk Manufacturing Serial Number	Not found on device

If successful, the End Device page should now display details for both the virtual and physical devices.


ARN
Amazon Resource Name
arn:aws:iotwireless:us-east-1:894735695240:WirelessDevice/8d249166-33e1-4a32-9018-b1161dbd58b3

Device ID
ID of the wireless device
8d249166-33e1-4a32-9018-b1161dbd58b3

Destination Name
Name of the destination to which the device is assigned
SidewalkDestination

Amazon ID
Sidewalk Amazon ID
-

SMSN
Sidewalk Manufacturing Serial Number
1867643416269AB50E3DF1BB0B3B7C6AF510B49819D3B2DCABDAFE1D6DF4CB7F

This Virtual Device was created in a previous session of Sidewalk Assistant
Click below to create a new Device and Manufacturing Page instead

Physical Device

Selected Device
Active device selected in Connection Manager
000440296582

SMSN
Sidewalk Manufacturing Serial Number
1867643416269AB50E3DF1BB0B3B7C6AF510B49819D3B2DCABDAFE1D6DF4CB7F

SMSN Match
Shows if the physical device's manufacturing page is the same as the virtual device above
Confirmed Matching

> ⚠ WARNING ⚠: The FLASH TO DEVICE control used above currently also performs an initial mass-erase operation before writing the manufacturing image to the device. This means the application image flashed in the previous step is NOT retained. To restore the primary application image, repeat the Compile and Flash the Project step in the previous section before continuing with these instructions.

Check the Device Registration and Time Sync

Recall that in Create and Compile your Sidewalk Application you saw the following log from your device:

```
00> [00000000] <info> App - sid_ble application started
00> [00000050] <info> MFG NVM3 start info:
00>  OK
00>
00> [00000100] <info> KV NVM3 start info:
00>  OK
00>
00> [00000101] <info> MFG Store opened with 0 objects
00>
00> > [00000102] <error> Sall mgm core create failed: -8
00> [00000103] <error> App - failed to initialize sidewalk: -8
00> [00000103] <error> App - fatal error
...
```

Now that you've used the Sidewalk Assistant to flash the manufacturing page, if you return to the RTT Viewer (you may need to reset the device and/or reconnect the RTT Viewer), you will see the following log from your device:

```
[00000000] <info> App - sid_ble application started
[00000011] <info> MFG NVM3 start info:
 OK

[00000022] <info> KV NVM3 start info:
 OK

[00000023] <info> MFG Store opened with 36 objects

> [00000116] <info> DR state [1]
[00000138] <info> App - sidewalk status changed: 1
[00000139] <info> App - registration status: 1, time sync status: 1, link status: 0
...
```

The following log snippet contains three statuses:

```
[00000139] <info> App - registration status: 1, time sync status: 1, link status: 0
```

- **registration status**
  Registration status indicates if the device is registered on Amazon's backend or not.
  - 0: the device is registered
  - 1: the device is not registered
- **time sync status**
  Time sync status indicates if the device has synchronized its current time with the gateway or not. You need to be time-synchronized in order to send and receive messages.
  - 0: time is synchronized
  - 1: time is not synchronized
- **link status**
  Link status displays the currently active communication channel.
  - 0: No connection
  - 1: BLE
  - 2: FSK
  - 4: CSS

For your device to communicate properly, registration status and time sync must be 0. If the endpoint is in-range of a Sidewalk gateway, this process (registration, if a new device, followed by successful time synchronization) should occur within a few minutes. Continue to Interacting with the Cloud to see how to fully exercise this example application in both directions across the Sidewalk network.

Interacting with the Cloud

Interacting with the Cloud

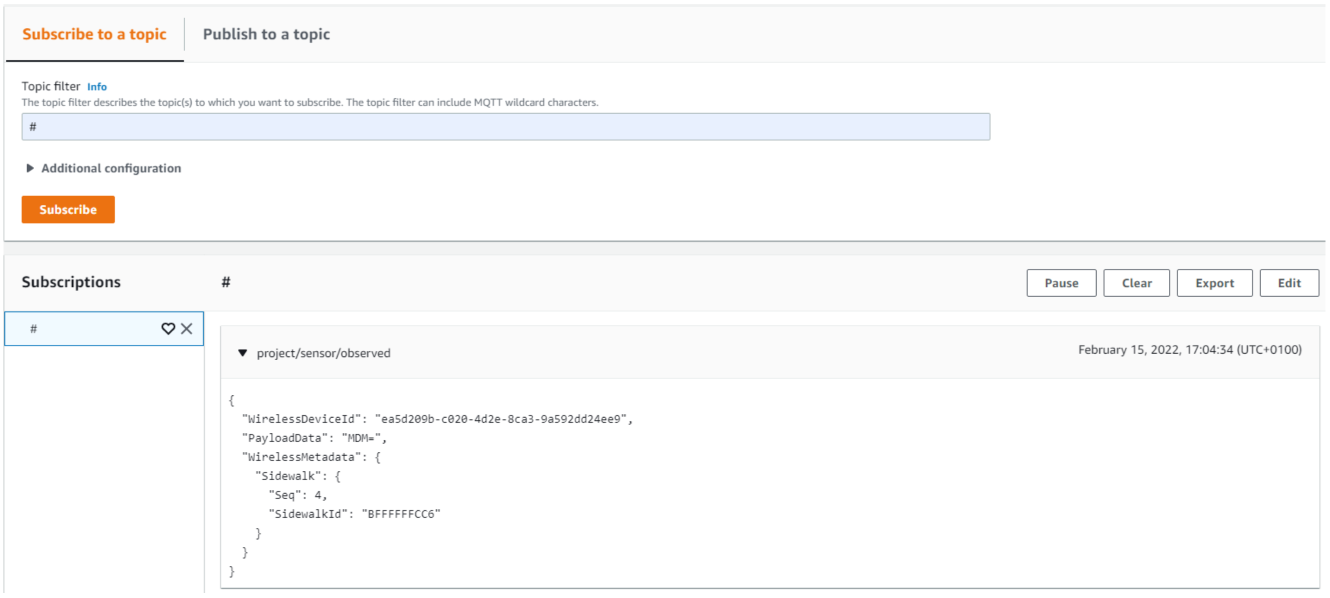
The Sidewalk solution allows an endpoint to natively exchange data with the AWS cloud. Before trying to send and receive message you should verify that your endpoint achieved time sync. You should see the following line in your device logs.

```
[00063831] <info> App - registration status: 0, time sync status: 0, link status: 0
```

Send Data

First, the endpoint sends data to the AWS cloud platform. An AWS Rule reroutes the data to an MQTT topic.

- Go to the MQTT test client in AWS.
- Type “#" in the **Topic filter** field.
- Click **Subscribe**.
- Press the main board's button (PB1 for KG100S and PB0 for all other boards). It triggers the Sidewalk endpoint to send a counter value to the AWS IoT Core. For more information on button and CLI action, see the readme of the **Amazon Sidewalk - SoC Hello Neighbor** sample application.


Subscribe to a topic
Publish to a topic
Topic filter Info
The topic filter describes the topic(s) to which you want to subscribe. The topic filter can include MQTT wildcard characters.
#
Additional configuration
Subscribe
Subscriptions
#
Pause
Clear
Export
Edit
#
project/sensor/observed
February 15, 2022, 17:04:34 (UTC+0100)

```
{
  "WirelessDeviceId": "ea5d209b-c020-4d2e-8ca3-9a592dd24ee9",
  "PayloadData": "MDM=",
  "WirelessMetadata": {
    "Sidewalk": {
      "Seq": 4,
      "SidewalkId": "BFFFFFFCC6"
    }
  }
}
```

An MQTT message appears in the AWS MQTT console.

Receive Data

The Hello Neighbor applications don't support the MQTT test client publish feature to send message from the cloud to the endpoint. For this step, you will need AWS CLI configured and associated with your IAM user. Your user should at least have the following policies: *AdministratorAccess* and *AWSIoTWirelessDataAccess*. If your user is *Administrator*, you don't need to do anything. You can check your users in the IAM Identity Center.

You are now ready to receive data on your Sidewalk endpoint. Run the following command in a terminal:

```
aws iotwireless send-data-to-wireless-device --id=[Wireless-Device-ID] --transmit-mode 0 --payload-data="SGVsbG8gICBTaWRld2FsayE=" --
wireless-metadata "Sidewalk={Seq=1}"
```

Where:

- **Seq=x**, x should be unique any time you run the command.
- **--id** should be the “Device ID” of the device in the AWS portal IoT Core -> Wireless Connectivity -> Devices -> Sidewalk.
- **--payload-data** is the message to send in binary format (in this case, “Hello Sidewalk!”).

You should see “Hello Sidewalk!” on the EFR32 log console.

Protocol Overview

Amazon Sidewalk Protocol Overview

This section presents some of Amazon Sidewalk's main concepts, such as registration and sub-GHz configurations. Reference Amazon documentation is linked at the bottom of this page.

- **Frustration Free Networking**: Explanation of device registration to Amazon Sidewalk network.
- **FSK Configuration**: Short introduction of the protocol using FSK radio layer.
- **CSS Configuration**: Short introduction of the protocol using CSS radio layer.

Useful links to Amazon documentation:

- Introduction to Amazon Sidewalk
- Protocol Specification
- API Reference (PDF)

Frustration Free Networking

Frustration Free Networking

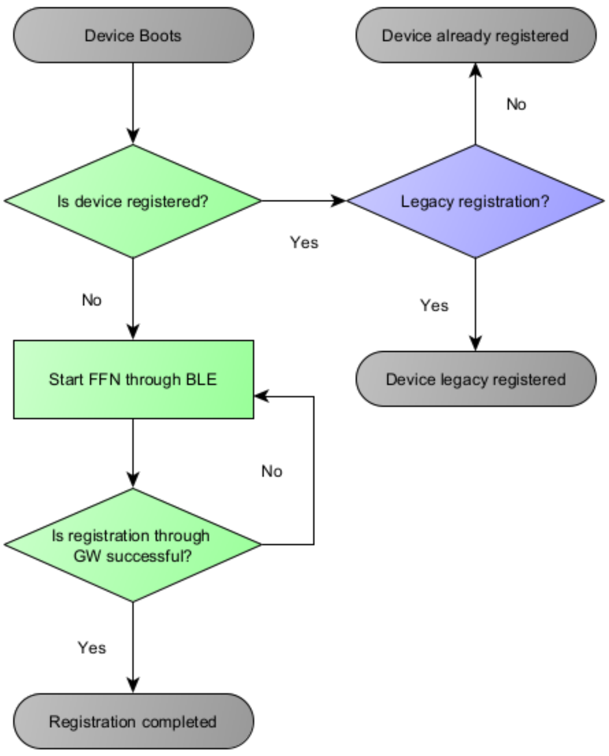
For Sidewalk, the Frustration Free Networking feature allows a device to register to the network without user action. This process is called Touchless Registration. It is an automatic process that occurs between the unregistered Sidewalk endpoint and the Sidewalk gateway over BLE or FSK. This method does not require associating the endpoint with the user's AWS account. To ensure your device registers using Touchless Registration, complete the following:

1. Ensure your gateway is Sidewalk-enabled and in range (the closer, the better).
2. Ensure your endpoint is up and running and open JLinkRTT Viewer for logs.
3. Registration should begin automatically. You should see some log output.
4. Successful registration will be indicated with the log below:

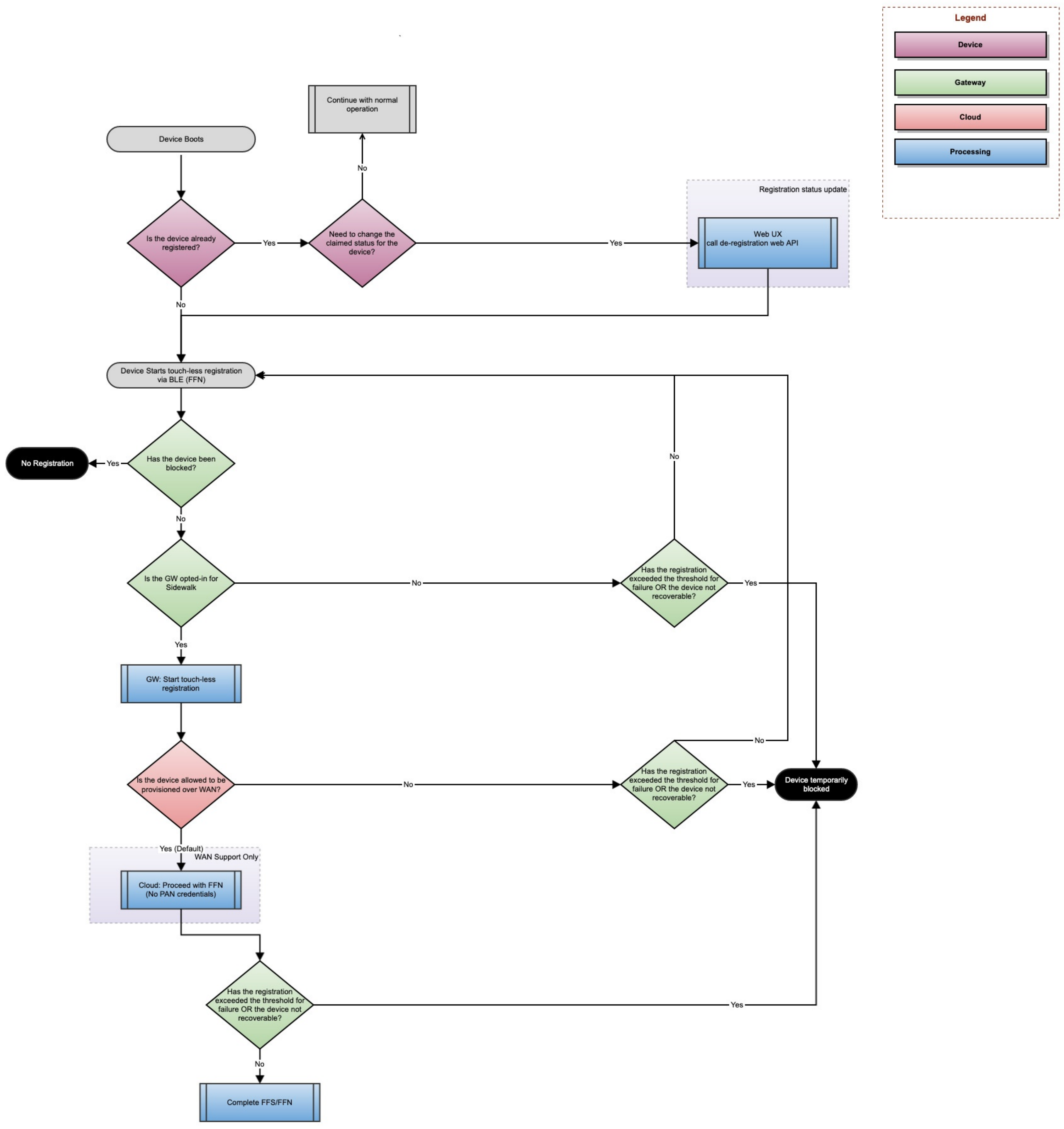
```
<info> app: Registration Status = 0, Time Sync Status = 0 and Link Status = 0
```

Diagram Flow

The following shows the Automatic Touchless Registration simplified flowchart:



If registration is attempted several times in a row and fails each time, the Gateway will block the device for a period of time. If that happens, rebooting the Gateway should reset the list of blocked devices. The following shows a full flowchart including device blocking.


Legend
Device
Gateway
Cloud
Processing
Continue with normal operation
Device Boots
Is the device already registered?
Yes
Need to change the claimed status for the device?
No
Yes
Registration status update
Web UX call de-registration web API
No
Device Starts touch-less registration via BLE (FFN)
No Registration
Yes
Has the device been blocked?
No
No
Is the GW opted-in for Sidewalk
No
Has the registration exceeded the threshold for failure OR the device not recoverable?
Yes
Yes
GW: Start touch-less registration
No
Is the device allowed to be provisioned over WAN?
No
Has the registration exceeded the threshold for failure OR the device not recoverable?
Yes
Device temporarily blocked
Yes (Default)
WAN Support Only
Cloud: Proceed with FFN (No PAN credentials)
Has the registration exceeded the threshold for failure OR the device not recoverable?
Yes
No
Complete FFS/FFN

Manual Registration Flow

Manual registration over BLE is an alternative registration mechanism to Automatic Touchless registration. Amazon Sidewalk also supports registration using the Amazon Sidewalk mobile SDK.

Note that this step requires a Mac, Windows or an Ubuntu PC with Bluetooth capability in order to execute the Sidewalk endpoint registration. As an alternative, a Bluetooth USB dongle can be used. Also note that it is good practice to disconnect all your other BLE devices connected to your computer (wireless headset, mouse, keyboard…) during registration.

Create a New Security Profile

- Go to the Login with Amazon page.
- Click **Create a New Security Profile**.

- Complete the required information: **Security Profile Name**, **Security Profile Description**, and **Consent Privacy Notice URL**. These values can be anything, for example:
  - Security Profile Name: SidewalkSecurityProfile
  - Security Profile Description: SidewalkSecurityProfile
  - Consent Privacy Notice URL: https://www.silabs.com/
- Click **Save**. You should see your new Security profile.
- Click **Show Client ID and Client Secret**, and note down the **Client ID** and **Client Secret** (or save them as environment variables).
- Click the gear to the right of your security profile's row.
- Click **Web Settings**.
- Click **Edit**, add http://localhost:8000/ to **Allowed Origins**, and click **Save**.

Obtain an LWA Token

An LWA (Login with Amazon) token is one of the ways to register your Sidewalk endpoint.

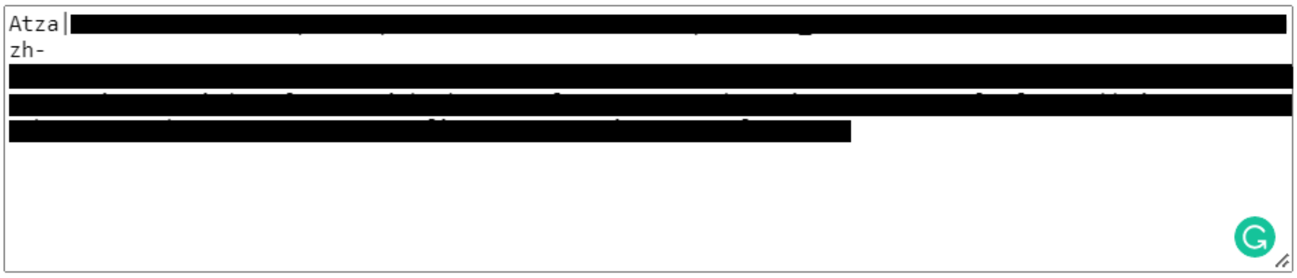
In your Amazon Sidewalk repository clone, go to */tools/scripts/public/sid_pc_link/apps/device_registration*, and install the module requirements:

```
pip3 install --user -r requirements.txt
```

Execute the following commands in a terminal:

```
python3 main.py -l --client-id <Client ID>
```

A browser window appears and opens a pop-up if allowed. Allow pop-ups on the page if not. The pop-up asks you for Amazon developer credentials. When logged in, the initial window displays an LWA token. The Python script automatically edits the *app_config.json* file with the LWA token. **The standard LWA token is only valid for an hour**.



```
Atza|[redacted]
zh-
[redacted]
```

Sidewalk Endpoint Registration

Open the *app_config.json* file in your Amazon Sidewalk repository folder */tools/scripts/public/sid_pc_link/apps/device_registration* and edit it as follows.

- Set the *ENDPOINT_ID* attribute equal to the "smsn" value found in the *certificate_[SIDEWALK_ID].json* file.
- If you are on Linux, the *BLUETOOTH_ADAPTER* attribute must be set to the Bluetooth dongle that you purchased as part of the setup. The `hcitool` command will read out your Bluetooth interfaces, and will likely be in the form `hciX` (most likely `hci0`, like the default setting in *app_config.json*). If you are on macOS, you do not need to touch this field.
  You can leave the rest of the attributes to their default values.
- Save and close the *app_config.json* file.
- Run the following command:

```
python3 main.py -r
```

You then see the logs of a series of exchanges on your terminal and on your EFR32 logging interface. After a minute or so, you should see "INFO Device registration succeeded". You have successfully registered your EFR32 device onto the Sidewalk network!

If your device is not detected by the script, it may be because your device connected automatically with the touchless registration using Sidewalk FFN.

If you are using the **Amazon Sidewalk - SoC Bluetooth Hello Neighbor** application, a main board button press is required to connect with the gateway. No button press is needed with the **Amazon Sidewalk - SoC Bluetooth Sub-GHz Hello Neighbor** application.

With the logging console connected, press main board button PB0. You should see the following status message.

```
[I] App - set connection request
[I] BLE State: CONNECTED

[I] App - sidewalk internal event
...
[I] App - sidewalk status changed: 0
[I] App - registration status: 0, time sync status: 0, link status: 1
[I] Delete rx_buffer :: Gateway [f1 76] :: Stream [7] :: Transaction [11]
```

Deregistration

You may want to de-register your endpoint for debug purposes.

> ⚠ WARNING ⚠: Successful de-registration requires a two-way message exchange between the endpoint, gateway, and cloud. Make sure your device is registered and time-synced with your gateway. You can check with the log output below:

```
<info> app: Registration Status = 0, Time Sync Status = 0 and Link Status = 0
```

To de-register your device run the command below by replacing `YOUR-WIRELESS-DEVICE-ID` with the device ID of the device you want to de-register.

```
aws iotwireless deregister-wireless-device --identifier "YOUR-WIRELESS-DEVICE-ID" --wireless-device-type "Sidewalk"
```

Successful de-registration of the device is indicated by the following log output:

```
<info> app: Registration Status = 1, Time Sync Status = 1 and Link Status = 1
```

Wait a few seconds after the de-registration of your device to see Automatic Touchless registration taking place to register your device.

> **INFO**: When the endpoint is registered but not time-synced and the de-registration API is called, the device is de-registered on the cloud side. The device does not receive the information because it is not time-synced. However, on its request message for time sync, the cloud will issue a message to notify the device it is not registered anymore. Upon receiving this new message, the device will remove its credentials and move to the deregistered state.

FSK Configuration

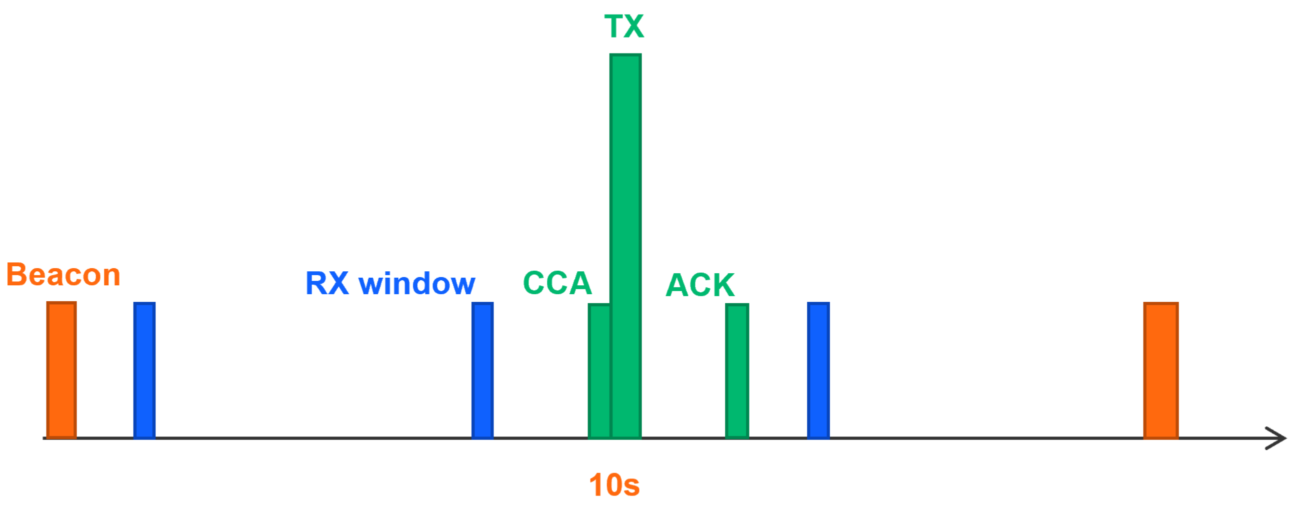
FSK Configuration

For more detailed information about FSK and its power profiles, see the "Sub-GHz Protocol Stack" > "Amazon Sidewalk Endpoint connection modes" > "SubG-FSK connection profiles" section in the Amazon Sidewalk specification.

Typical Behavior

FSK is a synchronous protocol, which means the endpoint keeps the connection with the gateway alive using beacons. Beacons are sent by the gateway every 10 seconds. Messages can be sent and received in between beacons.

By default the connection is kept alive and there are 3 RX opportunities every 10 seconds. Transmissions are preceded by a Clear Channel Assessment (CCA) and followed by an Acknowledgment message (ACK). See the figure below for an example:



When the endpoint boots, it connects to the gateway using discovery; the endpoint listens on every channel to detect the gateway preamble. Upon detection of the preamble, the endpoint retrieves its first beacon and starts communicating with the gateway (touchless registration, configuration negotiation, and time synchronization).

Power Consumption and Energy Modes

FSK Power Profiles

Two connection profiles are available for FSK referred to as "power profiles" 1 and 2. Power profile 1 consists of smaller messages exchanged during synchronization with the gateway. Protocol parameters are chosen by the gateway. Power profile 2 is a full synchronization with parameters chosen by the endpoint and negotiated with the gateway. In both power profiles it is possible to choose the time separation between two RX windows.

In the Hello Neighbor example application, power profile 2 is the default, with an RX window every 3150 ms.

Possible Parameters

The FSK power profile can be modified by the `sid_option` API call using option `SID_OPTION_900MHZ_SET_DEVICE_PROFILE`. Then the structure `sid_device_profile_unicast_params` can be updated with the following values:

- The device profile ID possible values `SID_LINK2_PROFILE_1` and `SID_LINK2_PROFILE_2` contained in `sid_device_profile_id` enum.
- The number of RX windows is always `SID_RX_WINDOW_CNT_INFINITE` for both power profiles.
- The windows separation can be any of the values of enum `sid_link2_rx_window_separation_ms`.
- The type of event for which the device wakes up values are contained in enum `sid_unicast_wakeup_type`.

Finally, while Profile 1 has infinite windows, the application can still turn the protocol off shortly after transmitting a packet to achieve similar operating characteristics of limited number of RX windows. When the protocol is turned off, the wakeups for beacon and data will not be executed. Whenever uplink data needs to be sent, the application can turn the protocol back on, re-establish the Sidewalk connectivity, transmit its data, and turn it back off. Hence, even though the protocol does not automate this process for the application, similar operating characteristics are still achievable.

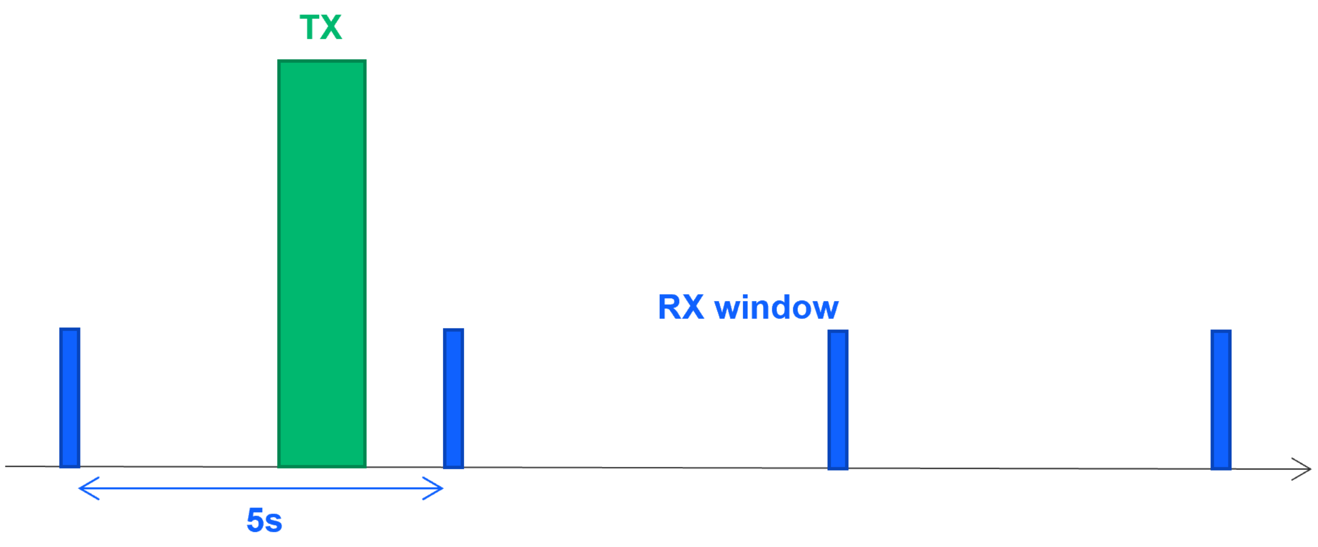
CSS Configuration

CSS Configuration

For more detailed information about CSS and its power profiles, see the "Sub-GHz Protocol Stack" > "Amazon Sidewalk Endpoint connection modes" > "Asynchronous access mode profiles" section in the Amazon Sidewalk specification.

Typical Behavior

CSS is an asynchronous protocol, which means it does not keep a connection with the gateway. Though transmits are asynchronous, during an active downlink phase listening windows for reception occur every 5 seconds as shown below. These windows either repeat indefinitely or after a transmission for a limited time, depending on the active connection profile (also referred to as "power profiles" A and B).



Power Consumption and Energy Modes

CSS Power Profiles

Power profile A opens a limited number of listening windows after a transmission. Power profile B opens listening windows periodically while also assuring regular transmission to keep the connection alive. In power profile B, the gateway considers the endpoint inactive after 5 minutes without transmission and stops sending messages. To prevent this, the Sidewalk stack implements a keep-alive mechanism for power profile B that creates a power consumption overhead not present in power profile A. There is a trade-off between this additional overhead and lower latency, as in power profile A a device must re-perform time synchronization before transmission if too much time has elapsed since the last transmit. Power profile B is the default connection profile in the Sidewalk stack.

Possible Parameters

The CSS power profile can be modified by the `sid_option` API call using option `SID_OPTION_900MHZ_SET_DEVICE_PROFILE`. Then the structure `sid_device_profile_unicast_params` can be updated with the following values:

- The device profile ID possible values `SID_LINK3_PROFILE_A` and `SID_LINK3_PROFILE_B` contained in `sid_device_profile_id` enum.
- The number of RX windows is either `SID_RX_WINDOW_CNT_INFINITE` for power profile B and any of the other values in enum `sid_rx_window_count` for power profile A.
- The windows separation can be any of the values of enum `sid_link3_rx_window_separation_ms`.
- Values for the type of event for which the device wakes up are contained in enum `sid_unicast_wakeup_type`.

Developer's Guide

Silicon Labs Developer's Guide for Amazon Sidewalk

This developer's guide presents all the concepts needed to develop your own application leveraging Amazon Sidewalk.

- **Stack Structure**: Presentation of Silicon Labs' solution integrating Amazon Sidewalk.
- **Application Development**: Instructions and documentation to allow you to develop your own application including automating device creation, and Secure Element and memory usage.
- **Testing and Debugging**: Solutions for debugging including logging.
- **Energy Modes and Power Optimization**: Introduction to Amazon Sidewalk power consumption.
- **Move to Custom Hardware**: Information on how to move to your custom hardware.
- **Troubleshooting**: Troubleshooting guide with all common issues.

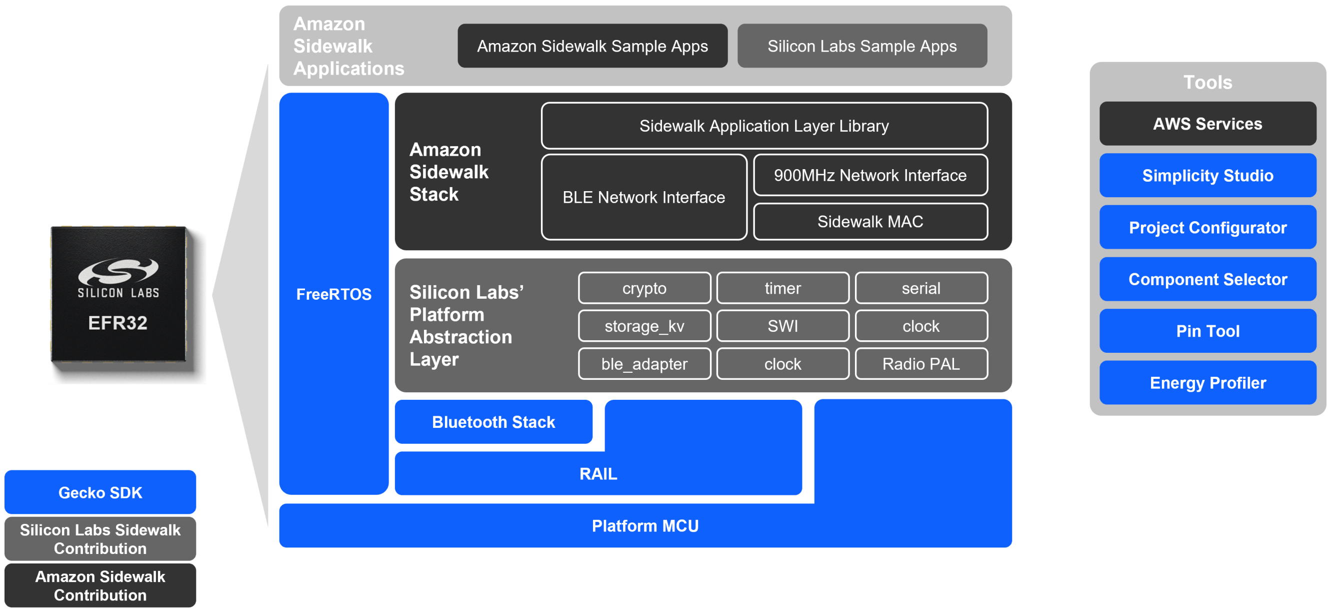
Useful links to Amazon documentation:

- Amazon Sidewalk General Documentation
- API Reference (PDF)
- AWS API Reference for Amazon Sidewalk
- Amazon Sidewalk Requirements
- Sidewalk Qualification
- Mobile SDK Developer Guide
- Manufacturing

Stack Structure

Stack Structure

The following figure describes the firmware structure that integrates the Amazon Sidewalk protocol stack and enables Amazon Sidewalk connectivity for the end device. The Platform Abstraction Layer assures communication between Silicon Labs Gecko SDK and the Amazon Sidewalk protocol stack. The developer creates applications on top of the stack, which Silicon Labs provides as a precompiled library.



The Amazon Sidewalk stack contains the following blocks:

- Application Layer Library: Contains API call definitions to be used in application development.
- BLE Network Interface: Contains Amazon Sidewalk-specific configuration for the Bluetooth LE (BLE) link layer.
- Sub-GHz Network Interface: Contains Amazon Sidewalk-specific configuration for sub-GHz link layer and Semtech drivers.
- MAC Layer: Amazon Sidewalk implementation of the MAC layer.

Dependencies

The following software components are used by the stack:

- FreeRTOS: Used as a base to run the stack.
- PSA crypto: Used for all cryptographic operation.
- BLE stack: Used to access the BLE stack.
- SPI driver: Used only in sub-GHz-enabled applications, for communication with a third-party sub-GHz radio module.
- RAIL: Used to access the radio and high-precision timers.

Library Files

Amazon Sidewalk extension contains three library files: one for the Sidewalk stack, one for the BLE stack and the last one for the Semtech chip driver. The Silicon Labs Platform Abstraction Layer is available in source in the extension.

Application Development

Application Development

Prototyping APIs

To automate device creation for prototyping, scripts are available that create the necessary objects in the cloud and the manufacturing page. These scripts are provided in the Silicon Labs extension for Amazon Sidewalk Github repository. To set up a prototype device and register it to the network the following steps are necessary:

1. Deploy your application in AWS
2. Compile and flash an application for your embedded device
3. Create a device profile in AWS
4. Create a wireless device in AWS
5. Generate a manufacturing page
6. Flash the manufacturing page to your embedded device

You can deploy your AWS application automatically using dedicated tools like Amazon CloudFormation. You can compile and flash your application using Simplicity Studio 5. To create the device profile, wireless device, and manufacturing page you can use Silicon Labs' prototyping script.

Remember to install the requirements listed in the `requirements.txt` file under `amazon_dependencies` with `pip3 install -r requirements.txt`.

The script configuration is handled by a JSON file like the following:

```
{
  "awsAccount": {
    "awsRegion": "us-east-1",
    "awsProfile": "default"
  },
  "commanderPath": "C:\\SiliconLabs\\SimplicityStudio\\v5_4\\developer\\adapter_packs\\commander\\commander.exe",
  "generationRequests": [
    {
      "deviceProfileId": null,
      "deviceName": null,
      "destinationName": "CFSDestination",
      "targetPart": null,
      "quantity": 1
    }
  ]
}
```

An example configuration file is provided at the root of the repository.

The first block `awsAccount` is used to configure the connection to AWS cloud:

- `awsRegion` : The AWS region you would like to add your devices to (Note that only the `us-east-1` region is supported at the moment).
- `awsProfile` : Used to choose the AWS CLI profile linking to the correct credentials. If you do not use profiles, you can leave the default.

The second block defines the toolchain used to create the manufacturing page:

- `commanderPath` : Should link to the the Simplicity Commander executable file.

The third block `generationRequests` controls the target device details:

- `deviceProfileId` : If this is your first time running the script, it should be empty.
- `deviceName` : This is optional, you can choose to give a custom name to your device.
- `destinationName` : Should be the name of the destination used by your AWS cloud application.
- `targetPart` : The OPN of the target part. If empty, manufacturing pages for all supported radio boards will be generated.
- `quantity` : The quantity of devices you want to create (default value is 1).

On the first run, the script creates a prototyping device profile and fills the `deviceProfileId` with the resulting device ID. All subsequent wireless devices will be created using this device profile. Note that during prototyping, a device profile can be linked to a maximum of 1000 wireless devices.

To start the script, execute the following:

```
python3 generate_prototype.py --input <.> --output <output_folder> --config example_config.json
```

`number_of_instances` is the number of devices you want to create (default value is 1).

The script creates a directory structure as follows in the chosen output folder under `mfg_output` :

```
DeviceProfile_7c51bc6b-0556-2083-6f0d-aeb750c94508
├── DeviceProfile.json
└── WirelessDevice_db57b1c9-22ad-c052-07ae-1e1cae9bb384
    ├── SiLabs_MFG.nvm3
    ├── Silabs_xG21.s37
    ├── Silabs_xG23.s37
    ├── Silabs_xG24.s37
    ├── Silabs_xG28.s37
    └── WirelessDevice.json
```

The first directory is named with the device id of your device profile and contains `DeviceProfile.json` , which contains your device profile information. Then a directory is created for every wireless device you created. One wireless device contains manufacturing pages for all supported platforms and a `WirelessDevice.json` file containing your device information (including private keys).

To use the script with command-line arguments instead of a configuration file, simply drop the `--config` parameter.

```
usage: generate_prototype.py [-h] [-i INSTANCES] -in INPUT -out OUTPUT [-p AWS_PROFILE] [-n NAME] [-d DST_NAME] [-t TARGET] [-c
COMMANDER] [-cfg CONFIG]

optional arguments:
  -h, --help            show this help message and exit
  -i INSTANCES, --instances INSTANCES
                        Number of instances to generate (default: 1)
  -in INPUT, --input INPUT
                        Path of the input directory
  -out OUTPUT, --output OUTPUT
                        Path of the output directory
  -p AWS_PROFILE, --aws-profile AWS_PROFILE
                        Name of your AWS profile from .aws/credentials (default: default)
  -n NAME, --name NAME  Specified name for the newly created device (default: sidewalk_[user]_device)
  -d DST_NAME, --dst-name DST_NAME
                        Destination name used for uplink traffic routing (default: CFSDestination)
  -t TARGET, --target TARGET
                        Target part number for which the MFG page generation has to be done (default: all)
  -c COMMANDER, --commander COMMANDER
                        Path to commander executable, not needed if already in system PATH
  -cfg CONFIG, --config CONFIG
                        Configuration file, if provided other arguments are ignored
```

Non-Volatile Memory Use

Amazon Sidewalk example applications use non-volatile memory to store the application code, registration information, and manufacturing data. This section describes how this memory is used for each type of data stored.

In addition to the application code, three other NVM3 instances are used in an Amazon Sidewalk project:

- Default
- Manufacturing
- Key-value storage

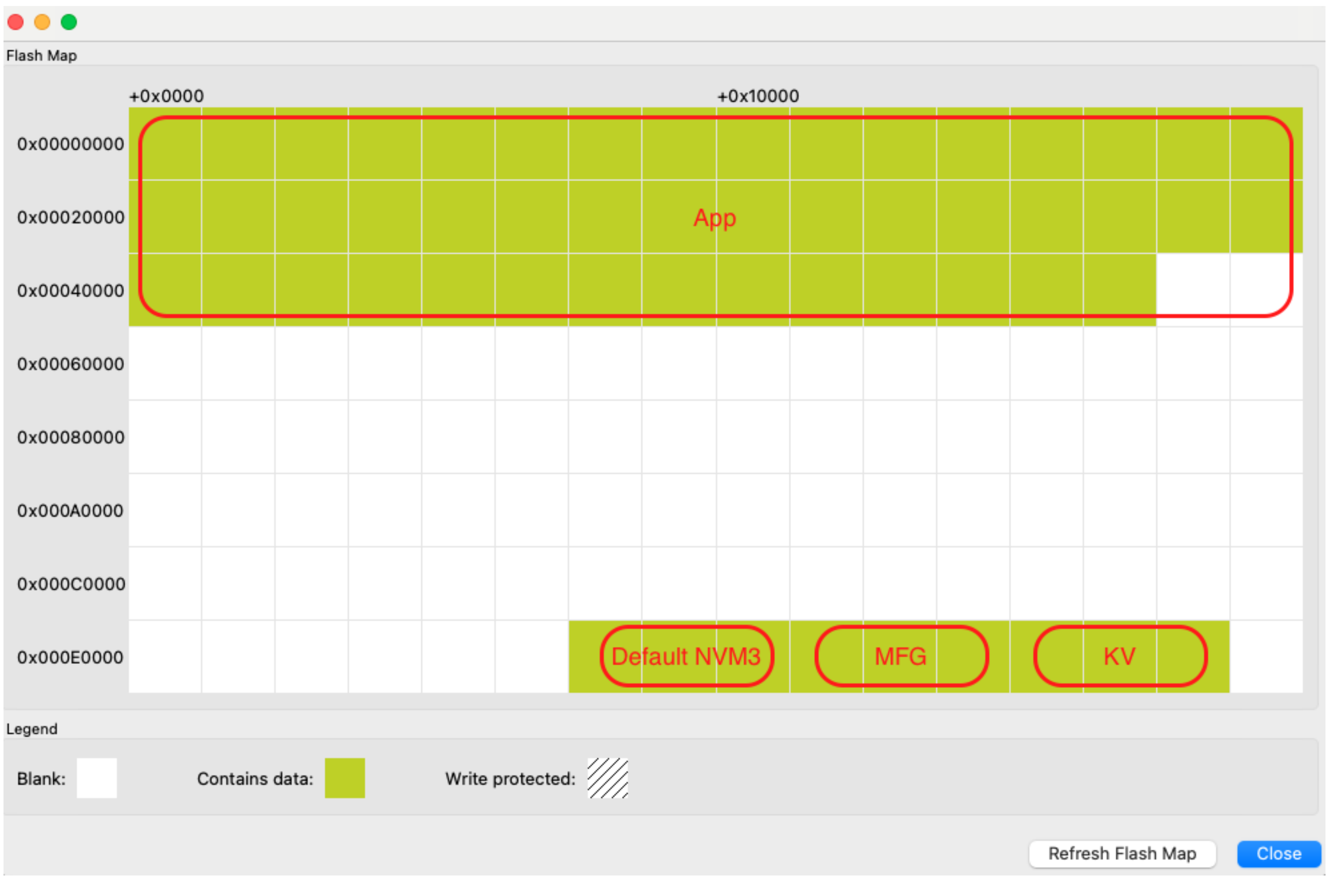
The default instance is mostly used by the Gecko SDK Suite (GSDK) for BLE stack status and cryptographic storage. This is where the wrapped private keys are stored when using Secure Vault. For more information see [the dedicated page] (place holder link for platform resources).

The manufacturing instance is used to store the manufacturing page generated for every device. The manufacturing page contains various information such as device public keys, signatures, and Sidewalk Manufacturing Serial Number (SMSN). This instance is read-only from application code.

The Key-value storage contains information about the Amazon Sidewalk stack. The way the key-value pairs are stored on NVM3 is a bit different from the way the manufacturing information is stored. Objects are stored under groups, so there is actually one NVM3 object (so-called group) to store different information. The size of this instance depends on the needs of the stack (depending on the link layer). The values stored in the instance are not part of the public Amazon Sidewalk APIs. This instance is accessible in read-write from application code.

For more information on Silicon Labs NVM driver, see the dedicated documentation NVM3 - NVM Data Manager. For background on NVM3, see the Platform Resources section.

Memory Map for EFR32xG21 Series

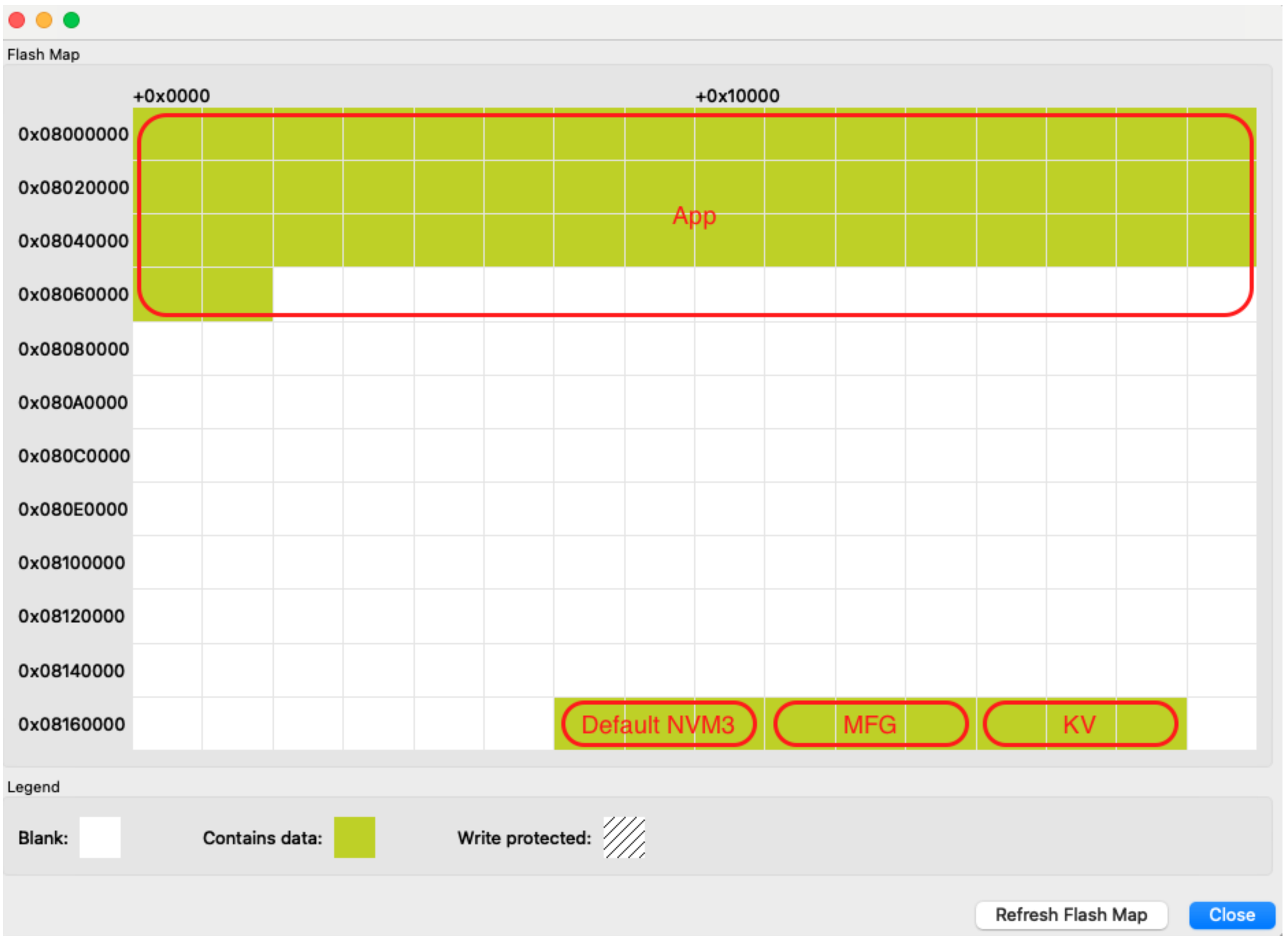


Region	Base Address	End Address	Size	Description
Application	0x00000000	-	-	Application code - size depends on application
Default NVM3 Instance	0x000EC000	0x000F2000	0x6000	Used for BLE stack. Wrapped keys are stored here when using Secure Vault.
Manufacturing page NVM3 Instance	0x000F2000	0x000F8000	0x6000	Manufacturing data - Read-only from Sidewalk Application context
Key-value Storage NVM3 Instance	0x000F8000	0x000FE000	0x6000	Sidewalk stack and registration status - Read/Write from Sidewalk Application context

Those addresses are given as an example for an EFR32xG21 with a flash size of 1024 kB. Base addresses can change to adapt to smaller flash sizes. You can use Simplicity Commander to display the memory layout of your application:

- With your kit selected, go to **Device Info** panel.
- Select **Flash Map** in **Main Flash** panel.

Memory Map for EFR32xG24 Series



Region	Base Address	End Address	Size	Description
Application	0x08000000	-	-	Application code - size depends on application
Default NVM3 Instance	0x0816C000	0x08172000	0x6000	Used for BLE stack. Wrapped keys are stored here when using Secure Vault.

Region	Base Address	End Address	Size	Description
Manufacturing page NVM3 Instance	0x08172000	0x08178000	0x6000	Manufacturing data - Read-only from Sidewalk Application context
Key-value Storage NVM3 Instance	0x08178000	0x0817E000	0x6000	Sidewalk stack and registration status - Read/Write from Sidewalk Application context

Those addresses are given as an example for an EFR32xG24 with a flash size of 1536 kB. The base addresses can change to adapt to smaller flash sizes. You can use Simplicity Commander to display the memory layout of your application:

- With your kit selected, go to **Device Info** panel.
- Select **Flash Map** in **Main Flash** panel.

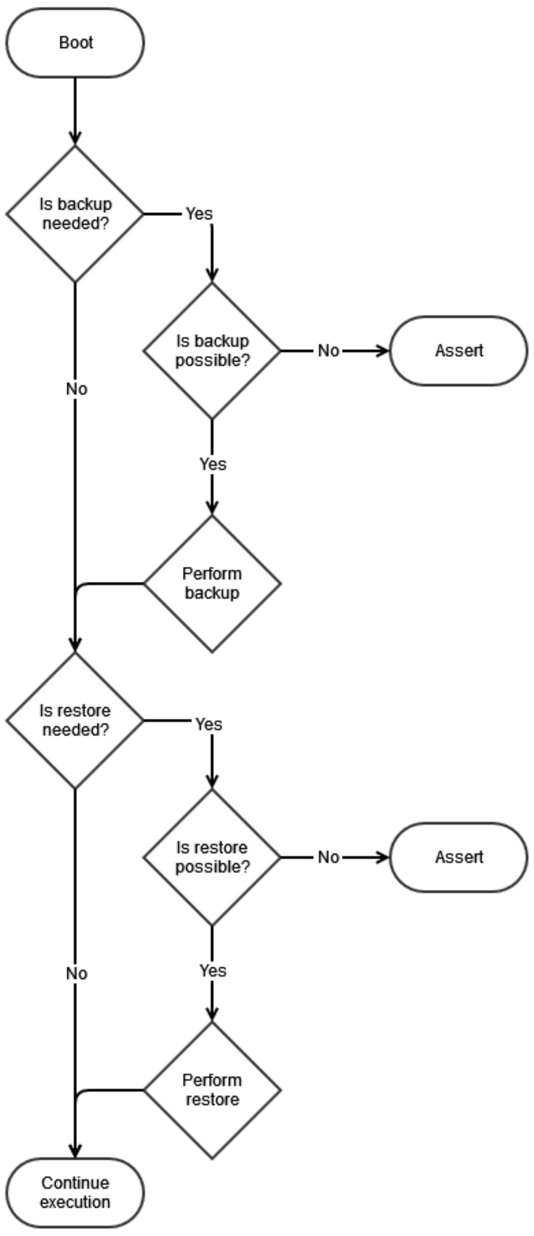
Credentials Backup/Restore Feature

The out-of-the-box (OOB) application provided with the Pro Kit has a feature that allows users to erase the flash memory without losing the device private keys and other device-specific information. The feature relies on the user data partition that is not affected by mass-erase but it can still be erased. This feature can be used alongside Secure Vault seamlessly.

The manufacturing page needs to persist in order for the application to work in the Amazon Sidewalk network. This data can be divided on two groups, one for device-specific information and the other for information common to all devices created under the same device profile. As the capacity of the user data partition is quite limited, the common information is stored in the application code and only the device-specific information is backed up on the user data partition.

To generate the header file that must be included in the application, use the `nvm3_to_c_array_converter` script.

The following flow diagram shows the backup/restore feature workflow.



The Backup feature is used to copy device-specific information from the manufacturing page onto the user data partition. There is no need to copy common information as it is hard-coded in the application. Backup takes place when the application cannot detect any backed up data on the user data partition. This implies that, if the user data partition is erased but not the application, then the application will perform a backup process after each boot.

The Restore feature is used to copy device-specific information from the user data partition and common information from the OOB application code onto the manufacturing page. Restore takes place when the application cannot detect any valid manufacturing page. This implies that, if the device is mass-erased, then the application will perform a restore process after each boot and restore the whole manufacturing page.

Use Cases

1. If you erase the main flash: The device should recover the credentials on first boot after flashing the application binary.
2. If you erase the user data partition: The device should recover by itself on first boot.
3. If you erase both the main flash and user data partition: Your device credentials are not recoverable.

This is especially important if you have one of Silicon Labs Pro Kits for Amazon Sidewalk. If you wish to use the radio boards with an application other than the demo, you can erase the main flash and use any application you like with your own manufacturing page. As long as you have not erased the user data partition, your device will recover the demo when you flash the out-of-the-box application binary back on your device. You should never erase the user data partition of your Pro Kit's radio boards.

Leveraging Secure Vault

On Silicon Labs' EFR32 Series 2 platforms, you can leverage Secure Vault to store the device's sensitive data (private keys). A set of scripts, called Dynamic Data Provisioning (DDP), is provided to use the Secure Element in the Amazon Sidewalk context.

The goal of DDP scripts is to facilitate dynamic data provisioning by providing ready-to-use scripts and a way of communicating with the embedded device from a host computer. The DDP flow for development and manufacturing setups stays the same, which makes the transition from development to manufacturing transparent.

Scripts consist of two functions: initialization and provisioning.

The initialization script generates an image with the static data of a Sidewalk device (the common part of the Amazon Sidewalk certificate) and optionally a Sidewalk application. The generated image is identical for all the devices in question.

The provisioning script, on the other hand, flashes the initialization image generated in the previous step, flashes the dynamic data provisioner RAM application, and provisions device-specific information via this application. Each device is provisioned with a unique set of security credentials (device-specific part of the Sidewalk certificate).

Leveraging Secure Vault does not use the manufacturing page (MFG) generated by the prototyping or manufacturing scripts. Instead it uses the WirelessDevice.json and DeviceProfile.json files provided as output of the prototyping scripts described in Prototyping APIs section. The steps are as follows:

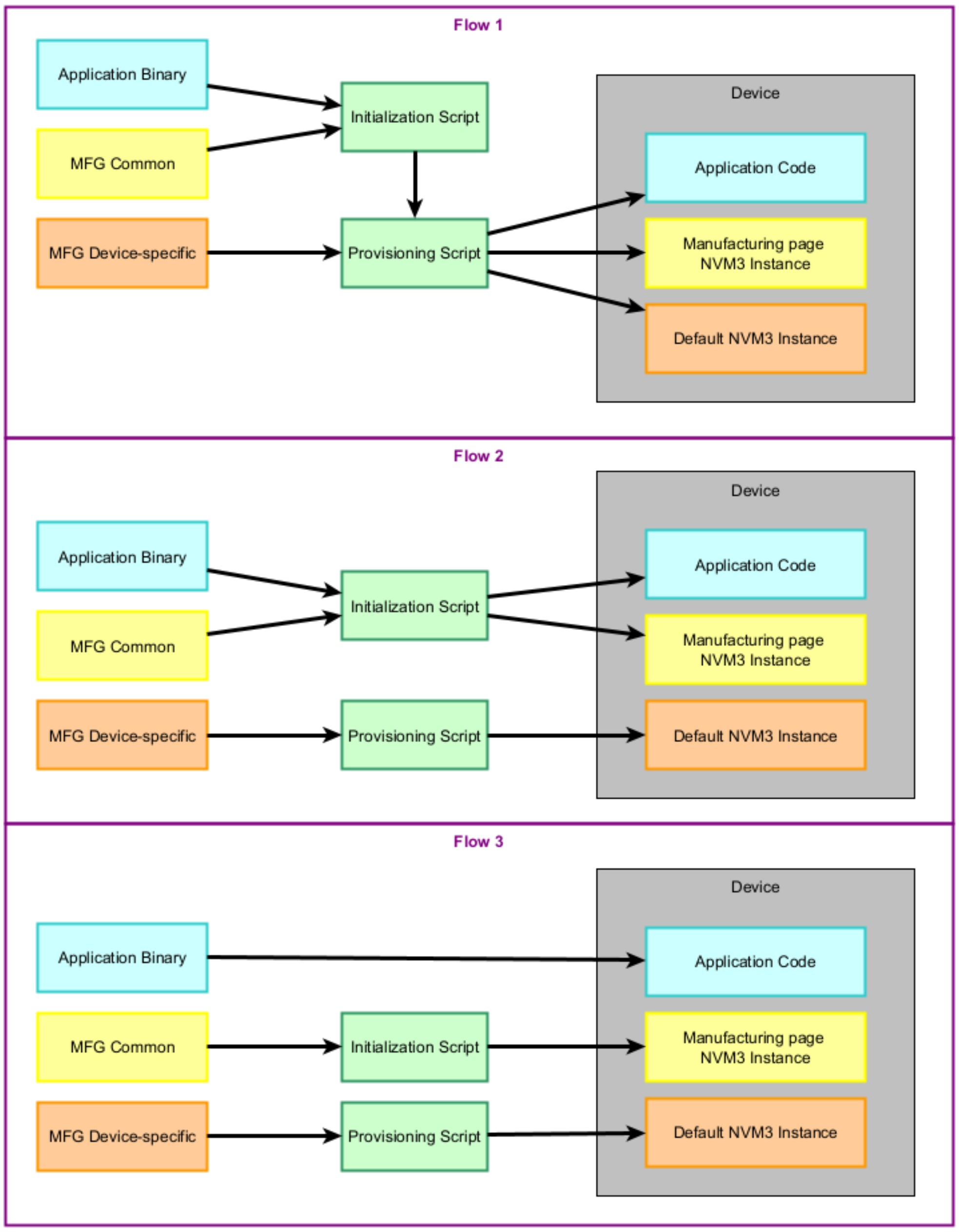
1. The initialization script creates an output file (.s37) containing a part of the manufacturing page that is identical for all the devices created using the same Device Profile.
2. The output of this initialization script can optionally include the Sidewalk application binary.
3. The device-specific information is provisioned to the device using the provisioning script. Here, the rest of the manufacturing page on the device will be completed with the device-specific information.

To summarize, the initialization script generates an output file that can be flashed to all devices. Then the provisioning script is used to provision the device credentials. The credentials are wrapped by keys contained in the Secure Vault and stored in the Default NVM3 Instance as described in the Non-Volatile Memory Use section.

The scripts can be used in three different configurations:

1. You can include the application binary in the initialization script and use the output of the script in the provisioning script. This flashes both binaries containing the application, MFG common section, and MFG device-specific section.
2. You can call the initialization script with the application binary. Flash the resulting image, which contains the application and MFG common section, on your device. Then call the provisioning script without the initialization image (init-img) option to flash the remaining MFG device-specific information.
3. You can omit the application binary in the initialization script (sid-app option). You will have to flash the output of the initialization script (MFG common section) before calling the provisioning one. Then you can call the provisioning script (MFG device-specific section). The application binary can be flashed at any point or added as an argument of the provisioning script (using the init-img option).

In all the flows, the only hard requirement is to flash the output of initialize_silabs.py (sid.s37) **before** calling silabs_provision.py. The common information in the manufacturing page must be on the device before the device-specific one. See the diagram below for more details.



The DDP scripts are located in `ddp/scripts` folder of the extension. The folder contains two python scripts (initialization_silabs.py and provision_silabs.py) and a binary file for the secure vault control (ddp_xg2x.bin one for each supported platform). In the following instructions on each script, two commands are shown, one for prototyping and one for manufacturing. The prototyping one can be used with the output files of Amazon prototyping flow (DeviceProfile.json and WirelessDevice.json) while the manufacturing one can be used with Amazon manufacturing flow output files containing the private keys.

Running the Initialization Script

The initialization script takes an Amazon Sidewalk device certificate, SoC family and Sidewalk application binary as input arguments and generates a binary file that contains the static (common) part of the Sidewalk certificate and the Sidewalk application provided as input arguments. The output binary file is common to all the product instances so it does not contain any device-specific information (device-specific information is provisioned in the next step). This script can be run once to generate the common binary in the production, following these steps:

- Navigate to `ddp/scripts` directory
- Run the commands below

```
pip3 install -r requirements.txt

# For manufacturing
python3 initialize_silabs.py --cert-file certificate.json --soc <bg21_or_mg21_or_mg24> --sid-app <sidewalk_application.s37> --out-file sid.s37 --prod

# For prototyping
python3 initialize_silabs.py --cert-file WirelessDevice.json --dev-profile DeviceProfile.json --soc <bg21_or_mg21_or_mg24> --sid-app <sidewalk_application.s37> --out-file sid.s37
```

At the end of this step, an output file (in `ddp/scripts/out`) that contains the static (common) part of the Sidewalk certificate and the Sidewalk application is generated. It will be used as an input argument in the next step.

Running the Provisioning Script

The provisioning script takes a Sidewalk device certificate, SoC family, initialization image (generated in the previous step) and provisioning image (ddp_mg24.bin) as input arguments and provisions the end device with device credentials extracted from the Sidewalk device certificate. This script must be run per device in production, using the following steps:

- Connect a main board and a supported radio board (BRD4181C, BRD4187C, or BRD4332A)
- Navigate to `ddp/scripts` directory (it must have been done previously in theory)
- Run the following commands

> ⓘ INFO ⓘ: If you prefer to flash the initialization image before the provisioning, you can skip the `--init-img` argument.

```
# For manufacturing
python3 provision_silabs.py --cert-file certificate.json --soc <bg21_or_mg21_or_mg24> --init-img out/sid.s37 --prov-img <sidewalk_dynamic_data_provisioner.bin> --prod

# For prototyping
python3 provision_silabs.py --cert-file WirelessDevice.json --soc <bg21_or_mg21_or_mg24> --init-img out/sid.s37 --prov-img <sidewalk_dynamic_data_provisioner.bin>
```

At the end of this step, the Sidewalk end device is provisioned with device-specific credentials.

Modify Application Code

To enable Secure Vault in the application, you must modify your code as follow:

In file `app_init.c`, add function `silabs_crypto_enable_sv` extern definition.

```
/* Enable Secure Vault for secure key storage */
extern void silabs_crypto_enable_sv(void);

void app_init(void)
{
```

In the same file, call function `silabs_crypto_enable_sv` during `app_init` before crypto module initialization.

```
  app_assert(ret == SID_ERROR_NONE, "sidewalk key-value storage init failed");

  silabs_crypto_enable_sv();

  // Initialize PAL crypto module
  ret = sid_pal_crypto_init();
```

Testing and Debugging

Testing and Debugging

RTT Logs

The UART interface is not always available to report traces from Sidewalk example applications. Instead, reporting leverages the J-Link RTT interface. To set up the communication between your PC and the EFR32, follow these instructions.

- Install the J-Link RTT Viewer.
- Open the J-Link RTT Viewer.
- In the **Configuration** panel, **Connection to J-Link** section, select **USB**.
- In the **Specify Target Device** list, select the connected part (for example EFR32BG21BxxxF1024 or EFR32MG24AxxxF1536).
- In the **Target Interface & Speed panel**, select **SWD** and **4000 kHz**.
- In the **RTT Control Block** panel, select **Auto Detection**.
- Click **OK**.

> ⓘ INFO ⓘ: For the KG100S module, select EFR32BG21BxxxF1024 as the target device.

A terminal opens and the Sidewalk application traces are output as shown below (Bluetooth application).

```
> [I] MFG NVM3 start info:
> OK
>
> [I] KV NVM3 start info:
> OK
>
> [I] MFG Store opened with 0 objects
```

> ⚠ WARNING ⚠: J-Link RTT and Simplicity Studio use the same channel to communicate with the board. If you do not see logs in J-Link RTT, try closing and re-opening Simplicity Studio to reset the connection.

Cloud Application Debugging

To debug your cloud application, several AWS objects can be use to monitor the events in your account. See here for more information.

Energy Modes and Power Optimization

Energy Modes and Power Optimization

For more information on Energy modes for Silicon Labs platforms, see the dedicated documentation on Power Manager.

BLE

The BLE configuration can be seen in the file `app_ble_config.c` of your sample application. Default device BLE name is "Sl", MTU size is 247 bytes, and MAC address type is public. BLE connection parameters are chosen by the gateway and the connection timeout is 30 seconds.

Advertising is divided into two behaviors:

- Fast advertising: transmit beacons every 160 ms for 30 seconds after boot
- Slow advertising: transmit beacons every 1 s after fast advertising

You can check the following structures in the API Reference:

- `sid_ble_cfg_adv_param_t` : for advertising parameters
- `sid_ble_cfg_conn_param_t` : for the connection parameters
- `sid_ble_cfg_gatt_profile_t` : for the GATT profile parameters
- `sid_ble_config_t` : for more generic configuration parameters

For more information on Silicon Labs BLE stack and power consumption, see the following pages:

- BLE General Overview
- Optimizing Current Consumption in Bluetooth Low Energy Devices
- Current Consumption Variation with TX Power

FSK

The FSK radio configuration can be seen in the file `app_subghz_config.c` of your sample application. While running with FSK, your EFR32 will be either in EM2 or EM0 energy modes depending on radio events. During radio events like beacons, RX, or TX your EFR32 will run in EM0 energy mode and in EM2 outside of those events.

For more power optimization, you can check Amazon Sidewalk power profiles for FSK here.

CSS

The CSS radio configuration can be seen in the file `app_subghz_config.c` of your sample application. While running with CSS, your EFR32 will be either in EM2 or EM0 energy modes depending on radio events. During radio events like RX or TX your EFR32 will run in EM0 energy mode and in EM2 outside of those events.

For more power optimization, you can check Amazon Sidewalk power profiles for CSS here.

Move to a Custom Hardware

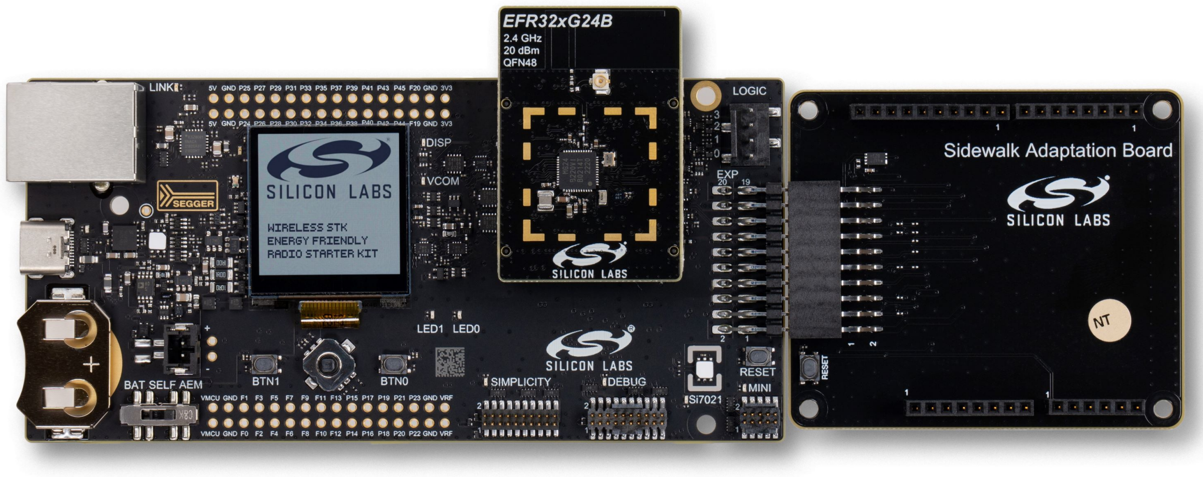
Move to Custom Hardware

Configure Transceiver GPIOs

Using the Sidewalk protocol on the sub-GHz radio band with an EFR32 radio board requires the Semtech SX1262MB2CAS LoRa shield. The Semtech shield can be connected to the main board using either an adapter board (recommended) or 10 female-to-female jumper wires (if adapter board is not available). Using this third-party sub-GHz radio module enables sub-GHz communication from EFR32xG21 and EFR32xG24 radio boards. The Semtech shield is not necessary with the KG100S, as it already includes the Semtech chip.

> ⓘ INFO ⓘ: For superior signal integrity, ease of use, and a more robust development platform, Silicon Labs recommends the Sidewalk Adaptation Board (BRD8042A, included in the Silicon Labs Pro Kit for Amazon Sidewalk) instead of jumpered wire connections.

When using the Sidewalk Adaptation Board, connect it to the main board Expansion Header as shown below, and mount the Semtech shield to the female pin headers on the Adaptation Board.



When using jumper wires, connect the Semtech shield to the main board Expansion Header with the following scheme:

EFR32 Mainboard Exp. Pin	Semtech Shield Pin	Function
EXP_HEADER 1	J3-6	GND
EXP_HEADER 4	J2-4	SPI MOSI
EXP_HEADER 6	J2-5	SPI MISO
EXP_HEADER 8	J2-6	SPI SCK
EXP_HEADER 10	J1-8	SPI NSS

EFR32 Mainboard Exp. Pin	Semtech Shield Pin	Function
EXP_HEADER 11	J2-1	ANT_SW
EXP_HEADER 12	J1-6	DIO1
EXP_HEADER 13	J1-4	BUSY
EXP_HEADER 14	J4-1	SX NRESET
EXP_HEADER 2	J3-4	VMCU

If you wish to customize the wiring between your sub-GHz chip and the EFR32, you can implement such changes in your application with the Pin Tool (in Simplicity Studio). When using the KG100S module, you also need to edit the GPIO mappings in file `app_gpio_config.c` in the sub-GHz example applications.

> ⚠ **WARNING** ⚠: In KG100S applications Silicon Labs recommends not using the SPI peripheral, because the multi-chip module (MCM) design already leverages it for communication between the EFR32 and the Semtech radio transceiver. Sharing the SPI bus with additional devices can negatively impact time-critical radio control signals and lead to message failure in sub-GHz protocols.

Amazon Sidewalk API

Amazon Sidewalk API

Refer to the Amazon Sidewalk Sid API Developer Guide.

Troubleshooting

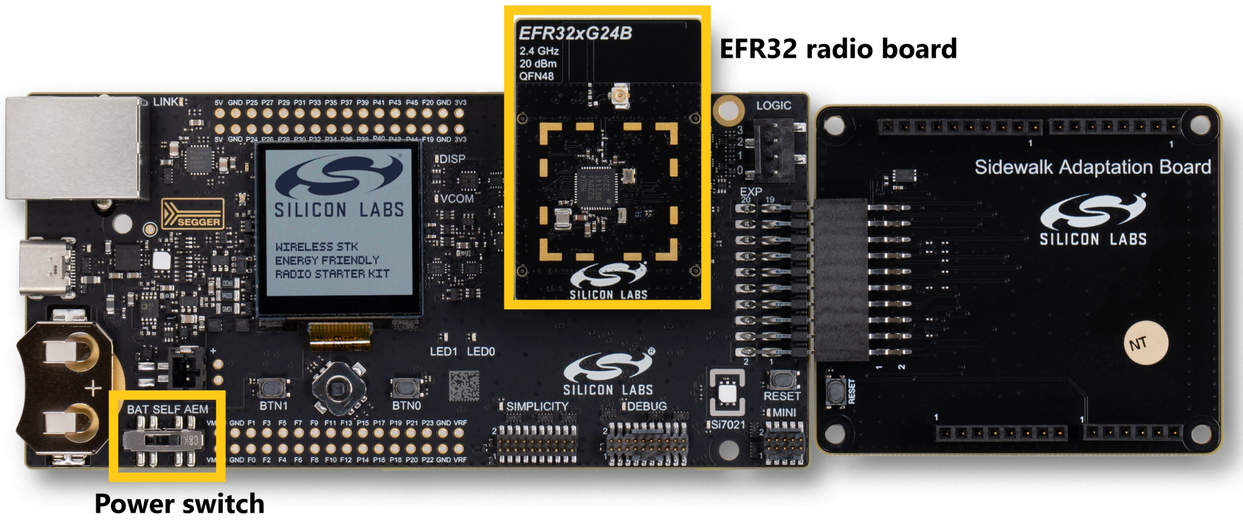
Troubleshooting

J-Link could not connect to the device (EFR32)

When using the Wireless Starter Kit main board (WSTK) along with a radio board, you should be able to connect with J-Link RTT Viewer using the parameters in the dedicated section. Your EFR32 should also be detected by Simplicity Studio 5.

If your EFR32 is not detected or you have difficulties connecting with J-Link RTT Viewer, you can check two points:

- Your WSTK power switch is on "AEM" mode.
- Your radio board is plugged in correctly (see the following)



CloudFormation Errors

Permission Denied

While executing the CloudFormation command to deploy your stack, you have an error showing missing authorization as follows:

```
<your user> is not authorized to perform: cloudformation:<some action> on resource aws:cloudformation:us-east-1:<some resource> because no identity-based policy allows the cloudformation:<some action> action
```

It is probably because your IAM user associated to your CLI does not have sufficient permissions. You should ask the person managing your AWS account to help you on this task.

Failed to Create Stack

CloudFormation stack creates objects and every user inside the same IAM policy has access to those objects. The stack only needs to be created once, and can be used for all your devices. If you try to deploy the CloudFormation stack but some objects already exist with the same name, your stack will not deploy with results like the following:

```
Waiting for changeset to be created..
Waiting for stack create/update to complete

Failed to create/update the stack. Run the following command
to fetch the list of events leading up to the failure
aws cloudformation describe-stack-events --stack-name NameOfYourStack
```

To get more details on this error, execute this command: `aws cloudformation describe-stack-events --stack-name NameOfYourStack`

This command displays all the actions attempted by CloudFormation. While reading the error messages, if you see an error log about an object that already exists then you should check what objects are already created inside your AWS account.

As an example you can see in the following log that the creation of `CFSRepublishLambdaRole` failed (value of `ResourceStatus` field) because it already exists (value of `ResourceStatusReason` field):

```
{
"StackId": "arn:aws:cloudformation:us-east-1:78468574544:stack/NameOfYourStack/4a54e4-4583-5435-f548-ad546584dd58",
"EventId": "CFSRepublishLambdaRole-CREATE_FAILED-2022-09-21T14:18:19.899Z",
"StackName": "NameOfYourStack",
"LogicalResourceId": "CFSRepublishLambdaRole",
"PhysicalResourceId": "",
"ResourceType": "AWS::IAM::Role",
"Timestamp": "2022-09-21T14:18:19.899000+00:00",
"ResourceStatus": "CREATE_FAILED",
"ResourceStatusReason": "CFSRepublishLambdaRole already exists in stack arn:aws:cloudformation:us-east-1:560019425582:stack/SidewalkStack/a58745f-5458-5742-125e-65a8b45a25",
"ResourceProperties": "{\"MaxSessionDuration\":\"3600\",\"RoleName\":\"CFSRepublishLambdaRole\",\"Description\":\"Allows IoT to call AWS services on your behalf.\",\"Policies\":[{\"PolicyName\":\"CFSRepublishPolicy\",\"PolicyDocument\":{\"Statement\":[{\"Action\":[\"iot:*\",\"sqs:*\",\"iotwireless:*\",\"logs:*\"],\"Resource\":[\"*\"],\"Effect\":\"Allow\"}]}}],\"AssumeRolePolicyDocument\":{\"Version\":\"2012-10-17\",\"Statement\":[{\"Action\":[\"sts:AssumeRole\"],\"Effect\":\"Allow\",\"Principal\":{\"Service\":[\"lambda.amazonaws.com\"]},\"Sid\":\"\"}]}}"
}
```

You should delete the duplicated resources or rename them.

Registration Errors

Failing During Device Registration

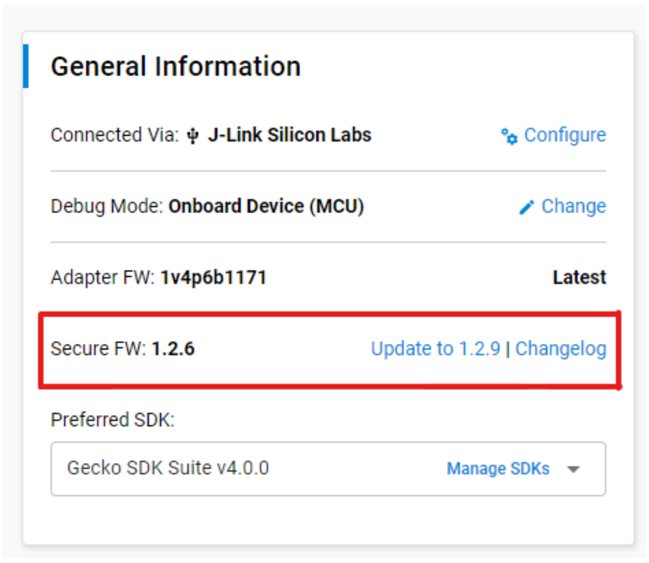
Once your device is detected by the registration script, it starts to exchange messages in order to register. During this message exchange, if you see the following errors on your endpoint logs, then your Secure Element Firmware is probably outdated.

```
<error> GET_DEVICE_ECDH_SIG: 3
<error> MASK NOT FOUND
<info> data_send send -8
<error> Dispatcher: Message handler returned unexpected result - -8
<error> MASK NOT FOUND
```

The Amazon Sidewalk examples require that the EFR32 uses a minimal version of Secure Element firmware depending on the radio board MCUs. See the table below to check your version:

Radio Board MCUs	SE minimal version
MG21	1.2.9
MG24	2.1.7
KG100S	1.2.9

You can verify this version in the Simplicity Studio 5 **Launcher** perspective. If the EFR32 is using an older firmware version, update it. If you do not see **Update to 1.2.9** (as shown below), disconnect and re-connect the board until it appears.



General Information

Connected Via: J-Link Silicon Labs — Configure

Debug Mode: **Onboard Device (MCU)** — Change

Adapter FW: **1v4p6b1171** — **Latest**

Secure FW: **1.2.6** — Update to 1.2.9 | Changelog

Preferred SDK: Gecko SDK Suite v4.0.0 — Manage SDKs

Time Synchronization

Once your device is registered, time synchronization between the gateway and the endpoint should take place within a few minutes. If the device does not attempt to synchronize with the gateway, check that your Amazon Echo device is connected to the Internet and localized in the US, then try rebooting your gateway and your endpoint.

Amazon Sidewalk - SoC Hello Neighbor Errors

When trying out the Hello Neighbor application with FSK radio layer, you may see logs about failing Disco module that resemble the following logs (the device is fully registered and attempts to discover the gateway):

```
<info> [00132619] App - sidewalk status changed: 1
<info> [00132619] App - registration status: 0, time sync status: 1, link status: 0[00132631] <info> RTC compensation is not supported.
[00132632] <info> swi thread created
[00132632] <info> Set region 1, country: US, num cfg: 2
[00132673] <info> P2P chnl loaded: 7
[00132673] <info> PAN ID loaded:
[00132673] <info> B9 80 96 21 00
[00132675] <info> Def mcast retries: 1
[00132675] <info> Pairing state loaded: 0
[00132675] <info> Config version 4
[00132675] <info> init_chnl_seed, chsid:9, seed:FFFFFA610000
[00132676] <info> rnet_mac_disco_init state = 0, req_auth = 0,cur_ev = 00000000
[00132676] <info> min ch symbols ms 151 bcn time ms 167
[00132677] <info> Starting GWD Process in: PASSIVE_SYNC/FFS_MODE
[00132677] <info> rnet_mac_disco_start_sampling state = 1, req_auth = 0, cur_ev = 00000000 f = 00052AC1 r = 0
[00132678] <info> Disco sampling start SUCCESS! state = 2, req_auth = 0, cur_ev = 00000000
[00132679] <info> 900MHz MAC Init: PRB:1 BCN:1 HDR:1, HDR_LORA:0 LDR:0
[00132680] <info> rnet_sec: Loading Setting (key=0xa) not on flash!
[00132682] <error> rnet_sec: File content for 2 mode cannot be trusted!
[00132683] <info> rnet_sec: Loading Setting (key=0x9) not on flash!
[00132685] <error> rnet_sec: File content for 1 mode cannot be trusted!
[00132686] <info> Ring-Net initialized
[00132686] <info> Disco stop sampling success. Time elapsed = 7 ms
[00132686] <info> [MET] B:0 N:0
[00132687] <info> rnet_mac_disco_init state = 1, req_auth = 0,cur_ev = 00000000
[00132687] <info> Starting GWD Process in: PASSIVE_SYNC/FFS_MODE
[00132688] <info> rnet_mac_disco_start_sampling state = 1, req_auth = 0, cur_ev = 00000000 f = 00052AC1 r = 0
[00132688] <info> Disco sampling start SUCCESS! state = 2, req_auth = 0, cur_ev = 00000000
[00132689] <info> Disco active. LSC=0, symb=7490, rx_to_ms=151
[00132690] <info> [MET] B:0 N:0
[00132857] <warning> Disco submodule ended prematurely! ch = 0
[00133025] <warning> Disco submodule ended prematurely! ch = 1
[00133193] <warning> Disco submodule ended prematurely! ch = 2
[00133362] <warning> Disco submodule ended prematurely! ch = 3
[00133530] <warning> Disco submodule ended prematurely! ch = 4
```

This error is generally linked to a missing or incorrectly plugged-in Semtech board.

Overview

Platform Resources

When you develop in the Silicon Labs GSDK, you have additional resources available to you through the Gecko Platform. This section includes information on the following topics.

- Bootloading: Bootloading allows you to update application firmware images on Wi-SUN devices. This section provides background information about bootloading using the Silicon Labs Gecko Bootloader.
- Non-Volatile Memory Use: This section offers an introduction to non-volatile data storage and describes how to use NVM3 data storage.
- Security: Silicon Labs offers a range of security features depending on the part you are using and your application and production needs.

Overview

Bootloading Amazon Sidewalk Applications

Bootloading allows you to update application firmware images on Sidewalk devices. This section provides background information about bootloading using the Silicon Labs Gecko Bootloader.

- Bootloader Fundamentals (PDF): Introduces bootloading for Silicon Labs networking devices. Discusses the Gecko Bootloader and bootloader file formats.
- Silicon Labs Gecko Bootloader User's Guide for GSDK 4.0 and Higher (PDF): Describes the high-level implementation of the Silicon Labs Gecko Bootloader for EFR32 SoCs and NCPs, and provides information on how to get started using the Gecko Bootloader with Silicon Labs wireless protocol stacks in GSDK 4.0 and higher.

Overview

Non-Volatile Memory Use

This section offers an introduction to non-volatile data storage and describes how to use NVM3 data storage.

- Non-Volatile Data Storage Fundamentals (PDF): Introduces non-volatile data storage using flash and the three different storage implementations offered for Silicon Labs microcontrollers and SoCs: Simulated EEPROM, PS Store, and NVM3.
- Using NVM3 Data Storage (PDF): Explains how NVM3 can be used as non-volatile data storage in various protocol implementations.

Overview

Security

Silicon Labs offers a range of security features depending on the part you are using and your application and production needs.

- IoT Security Fundamentals (PDF): Introduces the security concepts that must be considered when implementing an Internet of Things (IoT) system. Using the ioXt Alliance's eight security principles as a structure, it clearly delineates the solutions Silicon Labs provides to support endpoint security and what you must do outside of the Silicon Labs framework.
- Integrating Crypto Functionality with PSA Crypto vs. Mbed TLS (PDF): Describes how to integrate crypto functionality into applications using PSA Crypto compared to Mbed TLS.

Copyright © 2023 Silicon Laboratories. All rights reserved.