amazon sidewalk

# Amazon Sidewalk Sub-GHz Device Profiles Application Note

Protocol Stack 1.0, Document Revision A

March 3, 2026

# Contents

# Chapter 1

# Scope

This application note is intended for application developers who want to build their IoT solution using Amazon Sidewalk. This application note does the following:

1. Explains device profiles supported by the Amazon Sidewalk sub-GHz protocol
2. Enables developers to make an educated choice of the profile and the corresponding profile parameters to suit their application needs
3. Provides APIs and configuration settings for developers to change the profile and the corresponding parameters

# Chapter 2

# Device Profiles

## 2.1 Introduction

Amazon Sidewalk creates a low-bandwidth wireless network with the help of Amazon Sidewalk Gateway devices including select Echo and Ring devices. There are three main components to Amazon Sidewalk - Endpoints, Gateways and the Amazon Sidewalk Cloud. Gateway devices act as bridges and connect to Endpoints on one side and to the Amazon Sidewalk Cloud on the other, thus providing an end-to-end path. Endpoints which are sensor devices, such as a mailbox sensor, a tracker, a smoke detector or a motion detector connect over wireless channels to the Gateways. The Amazon Sidewalk Sub-GHz Protocol is a proprietary protocol developed by Amazon that enables message exchange between Gateways and Endpoints over the Sub-GHz channel. The protocol offers two different modes of communication - SubG-CSS (asynchronous) and SubG-FSK (synchronous).

IoT applications are sensitive to the following parameters: range, data rate, reliability, power consumption, mobility, latency and reachability. The Amazon Sidewalk protocol offers IoT application developers the flexibility to configure an Amazon Sidewalk profile through power, mobility and reachability parameters according to their application needs. To this end, we have introduced the concept of profiles. A profile in the Amazon Sidewalk protocol context is a configuration in the protocol that lets developers choose the best fit power, mobility, latency, and reachability settings for their application. Profiles are applicable only to Endpoints.

Let us explore three IoT applications to understand the relationship between power, mobility, reachability and latency:

- **Mailbox Sensor**: A mailbox sensor is a device that will notify the owner when mail arrives in their mailbox. The sensor sends the notification as an uplink message over Amazon Sidewalk to the application residing on the cloud. The mailbox sensor may send its health metric once in 24 hours. Almost all of the data exchanged for this use case originates from the sensor device and terminates at the application logic on the cloud. The use case does not have strict downlink (DL) constraints, as data from the application on the cloud to the application on the device is extremely rare. These devices are typically powered by AAA batteries and hence need to be extremely power-efficient.
- **Car Tracker**: A car tracker is a device that keeps the owner updated about the status of their car. Users are alerted if their car is bumped, or door movement occurs. Users will have the option to turn on live tracking through their application to get real time updates about the vehicle's location. Upon an alert of possible intrusion, a user may be able to activate the camera (if available) on the tracker to capture a picture or turn on video recording. While the device should be able to immediately send the uplink data upon an event, it should also be able to receive a downlink command in real time and perform any necessary action. These devices also need to be power-efficient.
- **Delivery Truck Tracker**: A delivery truck tracker is a device that keeps the owner updated about

the whereabouts of the truck and the delivery points the truck has visited. It differs from the car tracker due to the fact that a delivery truck is required to stop at specific delivery locations along its journey. The owner is alerted every time the truck stops at a delivery point. If necessary, the owner can turn on live tracking through their application to get real time updates about the truck's location. While the sensor device should be able to immediately send the uplink data upon an event, it should also be able to receive a downlink command in real time and perform necessary action.

See more examples in the Appendix.

The following section describes in detail the profiles offered in the Amazon Sidewalk protocol for both synchronous (SubG-FSK) and asynchronous (SubG-CSS) modes of operation.

## 2.2 Device Profile Definitions

### 2.2.1 Asynchronous Mode (SubG-CSS)

Asynchronous connection mode allows devices to connect to the network using the ALOHA channel access protocol. A frame can be transmitted by an Endpoint when needed without the need to synchronize with a Gateway.

- **Device Profile A**: An Endpoint operating in Profile A is characterized by limited reachability from the Amazon Sidewalk Cloud since it opens a limited number of RX (receive) windows separated by 5 s, after the device has sent an uplink packet. This profile is best suited to devices that have strict power budget and sporadic uplink data requirements. Uplink is triggered when the application on the Endpoint wants to send data to the application hosted on the cloud. Following the uplink transmission, the Endpoint opens up a configured number of receive windows separated by 5 s. These windows are receive opportunities for downlink data from the application logic on the cloud in response to the uplink. Profile A does not perform periodic network sync.
  **Attributes**: limited downlink reachability, Very low power consumption e.g.: mailbox sensor, water leak detector, smoke detector


Figure 2.1: Device Profile A

- **Device Profile B**: This profile is suited to Endpoint devices that have strict downlink reachability requirements as opposed to Profile A Endpoints. These Endpoints are less constrained by power. The device opens up continuous RX windows for possible downlink data from the application logic residing on the cloud every 5 s. Endpoints operating in this profile perform a network sync procedure every 5 min (configurable from 1-240 mins), to advertise their presence to the Amazon Sidewalk Cloud.
  **Attributes**: downlink reachable every 5 s, higher power consumption due to continuous RX wake-ups - e.g.: car tracker, motion detector
- **Device Profile C**: This profile is suited to Endpoint devices that require the lowest possible downlink latency and are not constrained by power (typically line-powered devices). The device operates in continuous RX mode, meaning it stays in receive mode continuously without intervals between RX windows. Profile C does not perform periodic network sync (same as Profile A). The `rx_window_count` is automatically set to `SID_RX_WINDOW_CONTINUOUS` (0xFF) and `async_rx_interval_ms` is set to `SID_LINK3_RX_WINDOW_SEPA` (0).

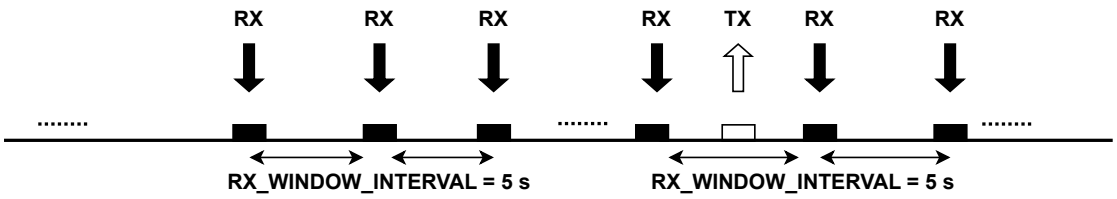


Figure 2.2: Device Profile B

**Attributes**: lowest downlink latency, highest power consumption, no periodic network sync - e.g.: line-powered gateways, always-on monitoring devices

- **Device Profile D**: Reserved for future use.

### 2.2.2 Synchronous Mode (SubG-FSK)

Endpoints working in synchronous connection mode shall be synchronized with a Gateway before an uplink or downlink transmission is performed. Synchronization is achieved through the process of discovering Beacon frames sent by the Gateways.

- **Device Profile 1**: This profile is best suited to moving Endpoints such as a delivery truck tracker or an elderly person tracker that keeps switching Gateways. This is the default profile of operation for a device operating in synchronous mode (SubG-FSK). The device automatically establishes an ad hoc mode of communication after discovering a Beacon from a nearby Gateway. In this Profile, there are a limited number of CS-UL (Common Slot UL) and CS-DL (Common Slot DL) slots defined in one Beacon interval (10.08 s), for uplink and downlink transmissions respectively. The Gateway makes the choice of the CS-UL slots and the CS-DL slots. The Endpoint does not have control over the choice of slots and will adhere to the slots chosen by the Gateway. The device decodes every Beacon from the Gateway to keep track of the ad hoc slots for that Beacon interval (10.08 s). Due to the nature of the profile, the Endpoint can go to power save mode for most of the Beacon interval.
  **Attributes**: high downlink latency, high uplink latency, low power consumption - e.g.: delivery truck tracker

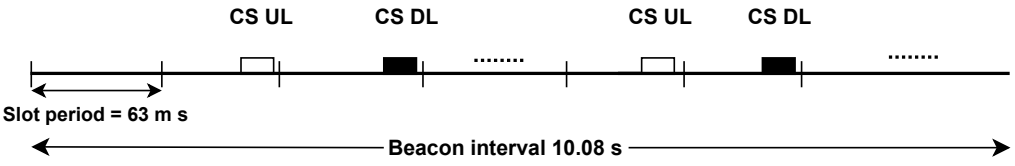


Figure 2.3: Device Profile 1

- **Device Profile 2**: This profile is suitable for Endpoints that have strict uplink and downlink latency requirements. Endpoints that are not mobile, resulting in a long period of association with a particular Gateway, should choose this profile. In this profile, the Endpoint negotiates for DL and UL allocation from the Gateway based on the application needs, thus ensuring that latency requirements are met. Based on the negotiation with the Gateway, the Endpoint can operate at a higher power consumption with lower latency or lower power consumption with higher latency. The timing diagram below shows the Rx latency configured to 5 slots = 63*5 ms = 315 ms. The Rx latency can be configured from 1 slot (= 63 ms) up to 80 slots (=5040 ms).
  **Attributes**: suited to static Endpoints - Endpoint negotiation with Gateway can ensure low power consumption - e.g.: mailbox sensor, smoke detector, water leak sensor

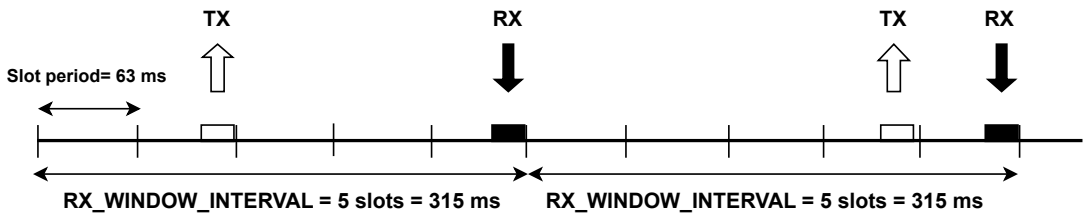


Figure 2.4: Device Profile 2

### 2.2.3 Dynamic Profile Switching

1. For asynchronous (SubG-CSS) profiles, the Amazon Sidewalk protocol allows dynamic profile switching between Profile A, Profile B, and Profile C.
2. For synchronous (SubG-FSK) profiles, the Amazon Sidewalk protocol allows dynamic profile switching Profile 1 to Profile 2.

Dynamic profile switching applies only after JOIN responses are received, and the change of profile will be persistent, that is, the switched profile remains after device reboot.

## 2.3 Sid API

The `sid_option` API call is used to change the device profiles.

```
sid_error_t sid_option(struct sid_handle h,
                       enum sid_option option,
                       void *data, size_t len)
```

The `sid_option` takes the following input parameters:

- `h` is the IOCTL handle returned by the call to function `sid_init`() at the time of Amazon Sidewalk initialization by the application
- `option` takes the value `SID_OPTION_900MHZ_SET_DEVICE_PROFILE` (=1)
- `*data` is a pointer to type `sid_device_profile`

The `data` and `len` parameters have the following values:

- `data->unicast_params.device_profile_id` is the profile ID of type `sid_device_profile_id`
- `data->unicast_params.rx_window_count` is of type `sid_rx_window_count`
  Note: For `SID_LINK2_PROFILE_1` device there is no effect of `rx_window` and may be defaulted to 0
- `data->unicast_params.async_rx_interval_ms` defaults to 5000 ms in this version of the protocol for Asynchronous Mode (SubG-CSS)
- `data->unicast_params.sync_rx_interval_ms` is of type `sid_link2_rx_window_separation_ms` for Synchronous Mode (SubG-FSK)
- `data->unicast_params.beacon_interval_unit` specifies the beacon interval unit for FSK communication.
  Valid values are `SID_LINK2_BEACON_INTERVAL_UNIT_1`, `SID_LINK2_BEACON_INTERVAL_UNIT_2`, and `SID_LINK2_BEACON_INT`
  (applicable only to Synchronous Mode SubG-FSK)
- `data->unicast_params.l2_rx_duration_sec` defines the Layer 2 receive duration in seconds for FSK links. Valid range is 0 to 65535 seconds (applicable only to Synchronous Mode SubG-FSK)
- `data->unicast_params.wakeup_type` shall be set to `SID_TX_AND_RX_WAKEUP`(= 3)

- `len` indicates the size of `*data` and should be set to size of(`sid_device_profile`) and evaluates to 8

### 2.3.1 Examples

- Setting the Asynchronous (SubG-CSS) profile `PROFILE_A with rx_window_count` parameter to 5. Suitable for the mailbox sensor described above.

```
sid_option(h,
           SID_OPTION_900MHZ_SET_DEVICE_PROFILE,
           *data,
           sizeof(sid_device_profile))
```

  The `sid_option` takes the following input parameters:

  - `h` is the IOCTL handle received in the call to `sid_init`() when the Amazon Sidewalk protocol is initialized
  - `*data` is a pointer of type `sid_device_profile`

  The `data` and `len` parameters have the following values:

  - `data->unicast_params.device_profile_id = SID_LINK3_PROFILE_A`
  - `data->unicast_params.rx_window_count = 5`
  - data->unicast_params.unicast_window_interval.async_rx_interval_ms = 5000
  - data->unicast_params.wakeup_type = SID_TX_AND_RX_WAKEUP
  - `len` evaluates to 8 which is the size of `sid_device_profile`

- Setting the Asynchronous (SubG-CSS) profile `PROFILE_B with rx_window_count` parameter to 0 (INFINITE). Suitable for the car tracker and the delivery truck tracker described above.

```
sid_option(h,
           SID_OPTION_900MHZ_SET_DEVICE_PROFILE,
           *data,
           sizeof(sid_device_profile))
```

  The `sid_option` takes the following input parameters:

  - `h` is the IOCTL handle received in the call to `sid_init`() when the Amazon Sidewalk protocol is initialized
  - `*data` is a pointer of type `sid_device_profile`

  The `data` and `len` parameters have the following values:

  - `data->unicast_params.device_profile_id = SID_LINK3_PROFILE_B`
  - `data->unicast_params.rx_window_count = 0`
  - `data->unicast_params.unicast_window_interval.async_rx_interval_ms = 5000`
  - `data->unicast_params.wakeup_type = SID_TX_AND_RX_WAKEUP`
  - `len` evaluates to 8 which is the size of `sid_device_profile`

  Note: rx_window_count for Profile B shall be set to 0 always

- Setting the Asynchronous (SubG-CSS) profile `PROFILE_C` for continuous RX mode. Suitable for line-powered devices requiring lowest downlink latency.

```
sid_option(h,
           SID_OPTION_900MHZ_SET_DEVICE_PROFILE,
           *data,
           sizeof(sid_device_profile))
```

The `sid_option` takes the following input parameters:

- `h` is the IOCTL handle received in the call to `sid_init`() when the Amazon Sidewalk protocol is initialized
- `*data` is a pointer of type `sid_device_profile`

The `data` and `len` parameters have the following values:

- `data->unicast_params.device_profile_id = SID_LINK3_PROFILE_C`
- `data->unicast_params.wakeup_type = SID_TX_AND_RX_WAKEUP`
- `len` evaluates to 8 which is the size of `sid_device_profile`

Note: For Profile C, `rx_window_count` and `async_rx_interval_ms` are automatically set by the stack to `SID_RX_WINDOW_CONTINUOUS` (0xFF) and `SID_LINK3_RX_WINDOW_SEPARATION_CONTINUOUS` (0) respectively.

- Setting the Synchronous (SubG-FSK) profile `PROFILE_1`, `rx_interval_ms` has no consequence here and may be defaulted to 0. Suitable for the delivery truck tracker described above.

```
sid_option(h,
           SID_OPTION_900MHZ_SET_DEVICE_PROFILE,
           *data,
           sizeof(sid_device_profile))
```

The `sid_option` takes the following input parameters:

- `h` is the IOCTL handle received in the call to `sid_init`() when the Amazon Sidewalk protocol is initialized
- `*data` is a pointer of type `sid_device_profile`

The `data` and `len` parameters have the following values:

- `data->unicast_params.device_profile_id = SID_LINK2_PROFILE_1`
- `data->unicast_params.rx_window_count = 0`
- `data->unicast_params.beacon_interval_unit = SID_LINK2_BEACON_INTERVAL_UNIT_1` (default)
- `data->unicast_params.l2_rx_duration_sec = 0` (default)
- `data->unicast_params.wakeup_type = SID_TX_AND_RX_WAKEUP`
- `len` evaluates to 8 which is the size of `sid_device_profile`

Note: `data->unicast_params.unicast_window_interval.sync_rx_interval_ms` does not apply to `PROFILE_1`.

- Setting the Synchronous (SubG-FSK) profile `PROFILE_2` with `sync_rx_interval` parameter to 315 ms (=63*5 ms). Suitable for the mailbox sensor described above.

```
sid_option(h,
           SID_OPTION_900MHZ_SET_DEVICE_PROFILE,
           *data,
           sizeof(sid_device_profile))
```

The `sid_option` takes the following input parameters:

- `h` is the IOCTL handle received in the call to `sid_init`() when the Amazon Sidewalk protocol is initialized
- `*data` is a pointer of type `sid_device_profile`

The `data` and `len` parameters have the following values:

- `data->unicast_params.device_profile_id = SID_LINK2_PROFILE_2`
- `data->unicast_params.rx_window_count = 0`
- `data->unicast_params.unicast_window_interval.sync_rx_interval_ms = 315`
- `data->unicast_params.beacon_interval_unit = SID_LINK2_BEACON_INTERVAL_UNIT_1` (configurable)
- `data->unicast_params.l2_rx_duration_sec = 3600` (example: 1 hour)
- `data->unicast_params.wakeup_type = SID_TX_AND_RX_WAKEUP`
- `len` evaluates to 8 which is the size of `sid_device_profile`

# Chapter 3

# References


1. Amazon Sidewalk Specification
2. Amazon Sidewalk Sid API Developer Guide

# Chapter 4

# Appendix

- **Connected Smoke Detector**: A smoke detector is a device that will signal an alarm upon detecting smoke. These devices are expected to have extremely low uplink (UL) latency - i.e. latency of communication between the sensor device to the cloud application upon detection of smoke. Under normal circumstances, the sensor may send its health metric once in 24 hours. The use case does not demand strict downlink(DL) constraints, as data from the application on the cloud to the application on the smoke detector is extremely rare. These devices are typically powered by AAA batteries and hence need to be extremely power-efficient.
- **Pet Tracker**: A pet tracker is a device that keeps an owner updated about the location of their pet. The owner may define a geo-fence around their home where they do not expect real time data about the whereabouts of their pet. If the pet wanders away from the geo-fence, then the owner is notified. Upon receiving the notification the owner has the option to request real-time tracking, which will be a downlink command to the Endpoint and needs to be serviced immediately. In this example, both uplink latency and downlink latency are the driving factors. Reducing latency increases power consumption.

# Chapter 5

# Change History

| Version | Summary of Changes |
|---|---|
| Protocol Stack 1.0, Document Revision A | First release of Sub-GHz Device Profiles Application Note. |
| Protocol Stack 1.0, Document Revision A.1 | Add LoRa Profile C documentation for continuous RX mode. Add Profile D as reserved. Update dynamic profile switching. |

# Glossary

| | |
|---|---|
| CSS | Chirp Spread Spectrum |
| Downlink | Data sent from a Gateway to an Endpoint |
| Endpoint | A device that accesses services of Amazon Sidewalk to transport messages to and from the application services through AWS IoT |
| FSK | 2-Gaussian Frequency Shift Keying modulation scheme |
| Gateway | Connections between Endpoints and Amazon Sidewalk are established and maintained through these devices. Gateways are Amazon and Ring devices. Amazon Sidewalk messages are transported between Amazon Sidewalk and the AWS IoT service through these devices. For a list of Amazon Sidewalk-enabled Amazon/Ring Gateways and the links that they support. |
| IoT | Internet of Things |
| LoRa | Long Range proprietary protocol based on Chirp Spread Spectrum modulation scheme (CSS) |
| RX | Receive |
| Sid API | Amazon Sidewalk Application Programming Interface |
| TX | Transmit |
| Uplink | Data sent from an Endpoint to a Gateway |