amazon sidewalk

Amazon Sidewalk Specification

Protocol Stack 1.0, Document Revision A.1

February 8, 2024

Use of these Amazon Sidewalk specifications (the "Specifications") is subject to your compliance with the AWS Customer Agreement and the Service Terms (collectively, the "Agreement"), including all disclaimers and limitations as to such use contained therein.

All statements, information, and data contained herein is subject to change without further notice to improve reliability, function, or design. Certain parameters may vary in different applications and performance may vary over time. It is your responsibility to validate that Amazon Sidewalk is suitable for your particular device or application.

No license, express or implied, by estoppel or otherwise, to any intellectual property rights are granted by this document.

Amazon Sidewalk is not intended for use in, or in association with, the operation of any hazardous environments or critical systems that may lead to serious bodily injury or death or cause environmental or property damage, and you are solely responsible for all liability that may arise in connection with any such use.

This document is Non-Confidential.

Contents

Chapter 1

Overview

The Amazon Sidewalk network connects Endpoints to Amazon Sidewalk Cloud servers. The Endpoints connect through Amazon Sidewalk Gateways. Gateways and Endpoints communicate using the Amazon Sidewalk protocol. The Amazon Sidewalk protocol can be carried across the following radios:

- Amazon Sidewalk SubG, referred to as SubG-FSK
- The LoRa ® radio, referred to as SubG-CSS
- Bluetooth ® Low Energy, referred to as BLE

Amazon Sidewalk supports synchronous and asynchronous connection modes. In synchronous mode, the Endpoint must synchronize with a single Gateway before communicating through that Gateway. In asynchronous mode, an Endpoint transmits messages which can be received by multiple Gateways.

Amazon Sidewalk Endpoints must register with the Amazon Sidewalk network before they can begin communicating. Registration requires keys that are built into the Endpoint at the time of manufacture.

Amazon Sidewalk Endpoints and Gateways communicate using frames. Frames can carry commands to control the connection, or data to communicate with the Amazon Sidewalk Cloud server.

Endpoints that are not actively communicating with Gateways can enter a low power mode. Power Profiles are used to manage low power behavior.

1.1 Scope

This document defines the wireless protocol used by Endpoints in the Amazon Sidewalk network. This document covers the following:

- Protocol stack layers
- The SubG-FSK radio
- Interactions between Endpoints and other elements of the system (commands and procedures)
- Power Profiles
- Security
- Architecture
- Frame formats and detailed descriptions of frame fields

This document does not include the following:

- Detailed specification of Gateways and the Amazon Sidewalk Cloud
- Interactions between Gateways and the Amazon Sidewalk Cloud

Chapter 2

Network architecture

Amazon Sidewalk is a wireless network providing low power and low cost IoT connectivity both around and beyond the home. Amazon Sidewalk accomplishes this via Amazon Sidewalk Gateways and Endpoints. Amazon Sidewalk Gateways route traffic from Endpoints to the Internet. Amazon Sidewalk Sidewalk supports different data rates from 2 kbps on SubG-CSS, 50 kbps on SubG-FSK, to 1 Mbps on BLE. Each Gateway may handle multiple Amazon Sidewalk-enabled IoT Endpoints like security sensors, trackers and motion sensors. In addition to the SubG-FSK and SubG-CSS radios, the Amazon Sidewalk network and architecture also encompass Bluetooth technologies.

Amazon Sidewalk Endpoints operate under two distinct radio-agnostic modes or configurations: Wide Area Network (WAN) and Personal Area Network (PAN). In the Wide Area Network configuration, also known as roaming, there is no association made between the Gateway and Endpoints. Each Endpoint has its own unique identity, and security credentials used to connect to the Amazon Sidewalk network. In the Personal Area Network configuration, there is an association between the Endpoint and the Gateway belonging to the same user. This allows Amazon Sidewalk to set the appropriate data usage limits and routing prioritization.

Each of these configurations comes with its own security model. A Gateway has to be configured to participate in the Amazon Sidewalk network. The process of configuring a Gateway to participate in the Amazon Sidewalk network is governed by the Amazon Sidewalk consent agreement. The maximum bandwidth of an Amazon Sidewalk Gateway to the Amazon Sidewalk Cloud is 80 kbps. When a registered Gateway user shares their Gateway's bandwidth with Amazon Sidewalk, the total monthly data used by Amazon Sidewalk is capped at 500 MB per account.

Endpoints working in the Amazon Sidewalk network may use three wireless technologies for communication - see Figure 2.1 which shows the corresponding ranges. The short range uses BLE over a 1 or 2 Mbps link. The medium range is based on SubG-FSK. The long range is based on SubG-CSS. Currently, Endpoints may use only one of these wireless technologies. Dynamic switch between these three modulations and consequently data rates are not currently supported on the Endpoint side. Gateways support all three of these technologies.

Endpoints may use one of two connection modes to connect to the network: asynchronous or synchronous.

Connection configuration includes the following options:

1. The following wireless technologies and communication ranges are enabled on Endpoints:
   (a) Short range communication using BLE
   (b) Medium range communication using SubG-FSK
   (c) Long range communication using SubG-CSS

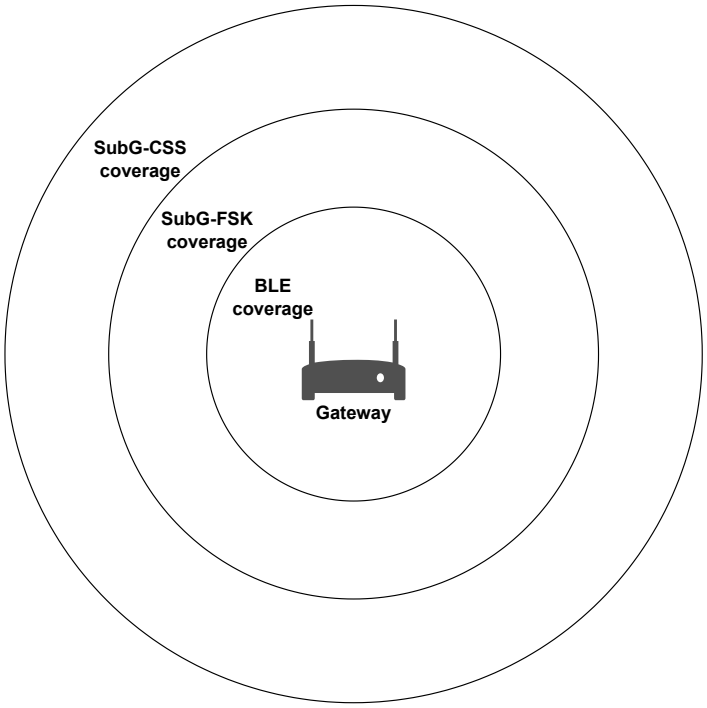


Fig. 2.1 Sidewalk wireless technologies coverage

2. Modulation usage has an impact on data rate traffic:
   (a) CSS modulation supports the lowest data rate on the SubG-CSS link (see Section 5.7.2 for details).
   (b) GFSK supports higher data rates on the SubG-FSK, and BLE links (see Section 5.7.3 for details).
3. There are two connection modes:
   (a) Asynchronous mode (see Section 5.5.2 for details).
   (b) Synchronous mode (see Section 5.2 for details).
4. There are two types of sub-network in which the Endpoint can operate:
   (a) Personal Area Network, for details see Section 2.2.
   (b) Wide Area Network, for details see Section 2.3.

Theoretically there are many possible ways to configure Amazon Sidewalk Endpoints, but the specification supports three configurations:

1. SubG-FSK - synchronous - WAN
2. SubG-CSS - asynchronous - WAN
3. BLE

Dynamic switching between configurations at the mac layer is not possible.

The following diagram shows the full Amazon Sidewalk network component end-to-end (E2E) software architecture.

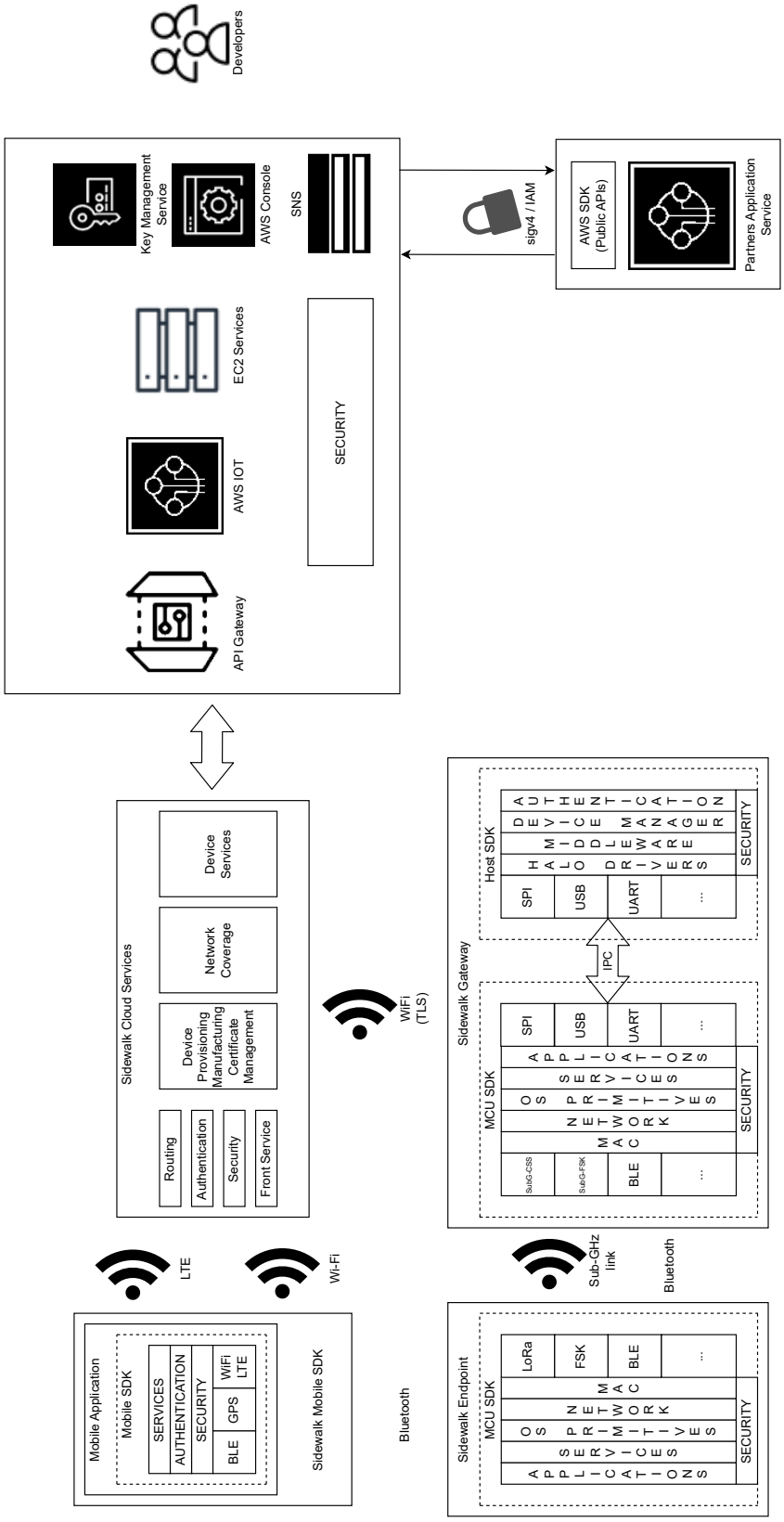


Fig. 2.2 Sidewalk E2E Architecture

2.1 Entities and roles

The main actors in the Amazon Sidewalk network are as follows:

1. End user interface, which could be placed in a mobile phone as an application or in a web service.
2. The cloud, which is the central entity of the whole system.
3. Gateways, which are nodes providing Amazon Sidewalk connectivity to Endpoints like security sensors, trackers and motion sensors.
4. An Endpoint, which provides specific functionality to its owner.

All of these elements create the Amazon Sidewalk network. A high level description of each element may be found in subsequent sections.

The specification defines interaction between Endpoints and Gateways. Cooperation between neighbor Gateways is described from a high level point of view to facilitate a better understanding of the whole system. A detailed description of Gateways interaction is outside the scope of this document. The description of interaction between cloud and Endpoint is provided in the Application Layer level from the protocol stack perspective.

2.1.1 End user interface

User interaction with the whole system is provided via the mobile app and/or via the web interface. User changes in the user interface related to selected Endpoints trigger message flow which is described in Figure 2.3.

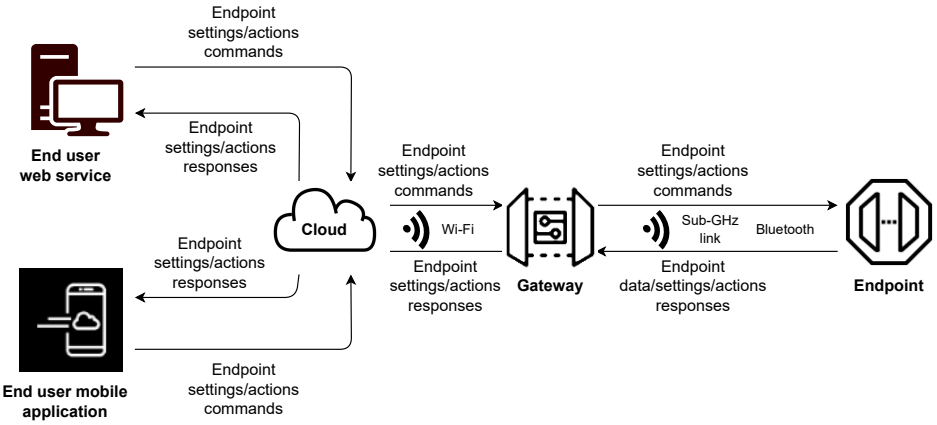


Fig. 2.3 End user interaction with an Endpoint

2.1.2 Cloud

The cloud is the central component of the Amazon Sidewalk network. It hosts the Amazon Sidewalk Network Server responsible for packet routing. The cloud also provides application level services for different types of Endpoints, participates in communication between end user applications and Endpoints, handles different security aspects, and many other functions. The cloud also works as a database, incorporates a central

routing repository, and stores different information about Endpoints, like connection profile settings and radio conditions. The cloud also has knowledge about the network topology, coverage and Gateways supporting roaming.

The cloud schedules downlink messages to Endpoints using the best available route. In synchronous mode, the Connection Profile determines the downlink transmit schedule. In asynchronous mode, downlink uses ad hoc scheduling. The cloud is responsible for scheduling downlink transmissions to avoid storing data on the Gateway for long periods. The Amazon Sidewalk Cloud selects the best Gateway, based on RSSI and other heuristics, to route downlink traffic, especially in cases when Endpoints are roaming.

2.1.3 Gateways

Amazon Sidewalk Gateways are special nodes of the Amazon Sidewalk network. These Nodes provide access to the external network for Endpoints organized in PAN and WAN networks. A single Gateway may support one PAN and one WAN network (for more detailed information about network types and topology, see Section 2.2 and Section 2.3). The traffic from an Endpoint to the cloud is handled by the Gateway as a passthrough node. The Gateway has no access to the application data sent from an Endpoint or received from the cloud because the application data are encrypted at the network and application levels.

Gateways may work in different configurations mainly impacting used modulations and subsequently different data rates. SubG-CSS and SubG-FSK modes of operation use the same radio, dynamically switching between the modes at runtime. SubG-CSS allows asynchronous while SubG-FSK allows synchronous communication. SubG-FSK-only or SubG-CSS-only modes of operation are also possible. All of these possible configurations are described in Section 5.4.

BLE uses a separate radio so works independently of SubG-CSS or SubG-FSK usage. A Gateway may receive data from SubG-CSS or SubG-FSK and BLE links simultaneously. The downlink transmission to an Endpoint is done either by SubG-CSS, SubG-FSK or BLE link based on the cloud's decision.

A Gateway having SubG-CSS or SubG-FSK capability has its own limited database which stores information about TX opportunities for operating Endpoints. Dependent on the user configuration, roaming support may be enabled or disabled. Based on stored information, a Gateway is able to recognize roaming Endpoints and subsequently allow or reject traffic coming from Endpoints. A Gateway also stores Endpoint network measurements reports so it may control Endpoints' power transmission.

As well as functioning as network service providers, Gateways also aid network topology map construction and maintenance in the cloud. Gateways perform periodic scans of their neighboring nodes and relay this information to cloud.

Occasionally a Gateway may lose its connection to the cloud. This information is provided to all surrounding Endpoints in the Beacon frame (see Section 5.1.3.1). When a Gateway link to a cloud is malfunctioning then it may try to get access to the external network through neighbor Gateways. This mode is called "distressed mode" and is described in Section 5.6.5. Information about the Gateway-cloud link health is not available for SubG-CSS Endpoints.

2.1.4 Endpoints

An Endpoint is a device that does not have responsibility for routing data or maintaining connection with the Amazon Sidewalk network for any device other than itself. These Endpoints provide different types of functionalities for the end user such as tracking devices, motion sensors and many others. These Endpoints have no direct Internet connectivity and are connected to the network through Amazon Sidewalk Gateways.

Endpoints operating using the SubG-FSK mode achieve synchronization during the Gateway discovery and subsequently the Beacon detection process. The procedure is described in Section 5.2.1. When a Beacon is received, the Endpoint gets uplink and downlink offset and periodicity. The Endpoint may exchange data as long as it is synchronized. A lack of synchronization results in a new Gateway discovery process.

An Endpoint operating with SubG-CSS may work in asynchronous mode only and WAN network type. The Endpoint does not need to be time-synchronized or associated with a Gateway. The Endpoint starts its transmission when required and does not specify which Gateway may handle that traffic. The uplink (UL) may be detected by various Gateways and the data is sent by them to the cloud. The decision about which Gateway shall handle downlink to the Endpoint is taken by the Amazon Sidewalk Network Server. The process of sending and receiving data between Endpoint and Gateway is described in Section 5.3.

Endpoints operating in asynchronous access mode are configured in one of two available Power Profiles. Basically, profiles configure the frequency of RX opportunities. The selected Endpoint profile have a direct impact on power consumption and also latency in both the uplink and downlink directions. Details related to Power Profiles are described in Section 5.5.

An Endpoint using BLE preserves behavior specified in the Bluetooth Low Energy Specification v 4.2. Chapter 6 provides the specification for Beacon and data frames. It also provides all other information required for appropriate BLE interaction of the Endpoint with the Amazon Sidewalk system that is complementary to the BLE specification.

2.2 PAN

Personal Area Network (PAN) integrates Endpoints and Gateways which belong to the same registered Amazon or Ring user account and allow them to bypass the opt-out and data limit restriction. All Endpoints within the PAN area share the same security key called a PAN key or broadcast key. This key is provided to an Endpoint at the time of device provisioning.

PAN can have one or more Gateways. These are central nodes of the network and can serve one or multiple Endpoints. Figure 2.4 shows the case when the PAN has one Gateway as a central node and a few Endpoints which are connected to the cloud via the Gateway.

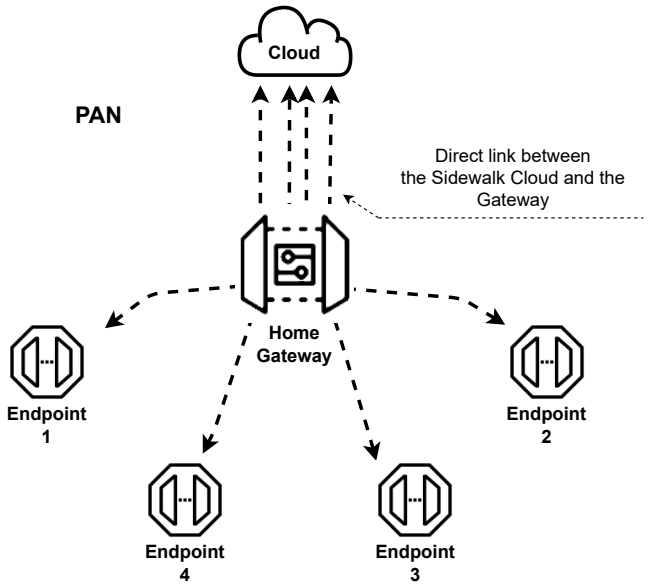


Fig. 2.4 Endpoint—cloud communication

Endpoints on the Amazon Sidewalk network shall limit the number of messages transmitted on the PAN BLE uplink. For the limit see Table 2.1. If an Endpoint exceeds the limit on the number of messages transmitted on the PAN uplink, then it may be disconnected from the Amazon Sidewalk network.

PHY	Maximum thoughput/messages
BLE	600/min

Table 2.1 Maximum throughput for each Endpoint on PAN uplink

2.3 WAN

Wide Area Network (WAN) integrates Endpoints and Gateways independent of their ownership. Each Endpoint within the WAN area has its own unique device key which changes periodically. The cloud is the entity which takes responsibility for ensuring authenticity and integrity of the incoming messages from the Endpoints. The security model adopted by WAN Endpoints is called WAN Network Topology.

WAN has multiple Gateways which serve one or multiple Endpoints. Unless roaming support is disabled by the user, all the Gateways are by default configured to serve any Endpoints. In this case, Gateways provide connectivity beyond the limit of the Personal Area Network. The Gateway mostly serves as a passthrough device. Endpoints may use SubG-FSK, SubG-CSS modes and BLE to communicate over the WAN network.

The end user may restrict usage of its own Gateway from supporting other users' Endpoints in the WAN network topology. Endpoints working in SubG-FSK mode may get the indication whether Gateway supports roaming Endpoints. This information is provided in the Beacon frame format on the Link Layer level. When an Endpoint is in WAN mode outside its own Gateway coverage then should refrain from sending a message to the Gateway when it does not support roaming.

Endpoints working in SubG-CSS mode have no information about which Gateway supports roaming or not. The SubG-CSS Endpoint sends the message and each Gateway receiving that message takes a decision whether the data should be forwarded to the cloud. Due to Amazon Sidewalk network's spatial diversity, it is possible that two or more Gateways may forward the same message to the cloud. It is the cloud's task to recognize duplicated packets and also take a decision about which Gateway would handle downlink transmission. Communication in the WAN network is shown in Figure 2.5.

The end user may restrict usage of its own Gateway from supporting other users' Endpoints over all PHYs.

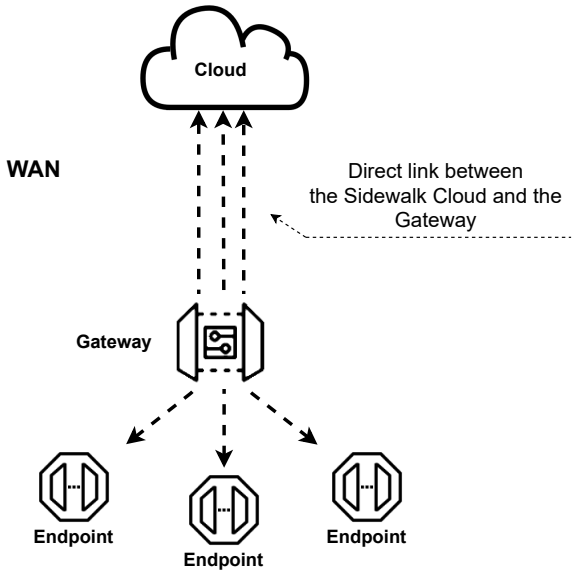


Fig. 2.5 WAN network

The Amazon Sidewalk network enforces uplink maximum traffic rates for endpoints. Please refer to Sidewalk End Device Maximum Uplink Traffic Rates in the Sidewalk documentation for details.

Chapter 3

Application

This chapter describes the Application layer role in the Amazon Sidewalk protocol stack.

3.1 Application layer characterizes the role of this layer and frame format.

3.2 Application commands describe application commands that are collected in separate groups.

3.1 Application layer

Application layer is the top layer of the Amazon Sidewalk protocol. This layer supports realization of procedures defined for different activities of an Endpoint. These activities may be Endpoint registration, time synchronization, and data exchange with the cloud. Procedures are executed by sending and receiving a sequence of messages between Endpoint and the other side of communication link. One message shall contain one application command only. The layer takes care of the right order of messages flow, handling exceptions and others that require proper finalization of a procedure.

A user application placed on an Endpoint provides a service to a user. This level application does not have direct access to commands defined and described in this and following sections. A user application interacts with the Amazon Sidewalk protocol stack application layer through the SDK interface. SDK interface description is outside the scope of this technical specification.

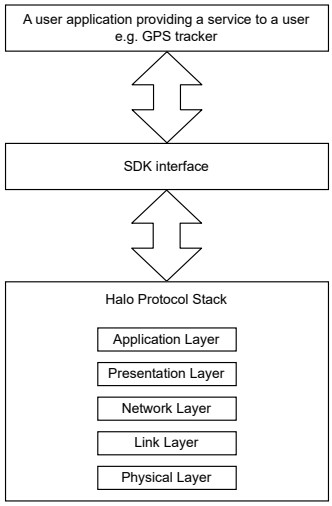


Fig. 3.1 Application layer - SDK interface

3.1.1 Application layer frame format

Application layer provides a simple frame format for coding of application commands. This frame contains three fields: Command class, Command ID and Command data.

Fig. 3.2 Application layer - Frame format

A Command class field identifies a high level functionality area. Each Command class contains set of commands that perform a task in the corresponding area. The following Command classes are defined in the Amazon Sidewalk protocol:

1. management - group of commands that are dedicated for managing Endpoints in the Amazon Sidewalk network
2. custom - group of commands that provide data exchange service for a user application
3. metrics - group of commands that are related to various statistics and measurements of communication link
4. security - group of commands that implement Amazon Sidewalk security and registration

The Command ID field identifies a command in a Command class. It also indicates an interpretation of Command data bytes.

The Command data field contains data associated with a Command. The size of the field is Command-dependent.

The Command class and Command ID fields may have different lengths. This specification defines five base cases where different sizes and range of values of these fields are determined. Both the Command class and Command ID shall be coded on the same field size and the higher value of either Command class or Command ID specifies the correct case. For example, when Command class is equal to 0x0 and Command ID is equal to 0x108 then both shall be encoded on 12b field size (Case 4).

Case	Command class range of values	Command ID range of values	Command class and command ID fields size	Comment
1	0x0 - 0x7	0x0 - 0x3	3b	see Figure 3.3
2	0x8 - 0xF	0x4 - 0xF	4b	see Figure 3.4
3	0x10 - 0xFF	0x10 - 0xFF	8b	see Figure 3.5
4	0x100 - 0xFFF	0x100 - 0xFFF	12b	see Figure 3.6
5	0x1000 - 0xFFFF	0x1000 - 0xFFFF	16b	see Figure 3.7

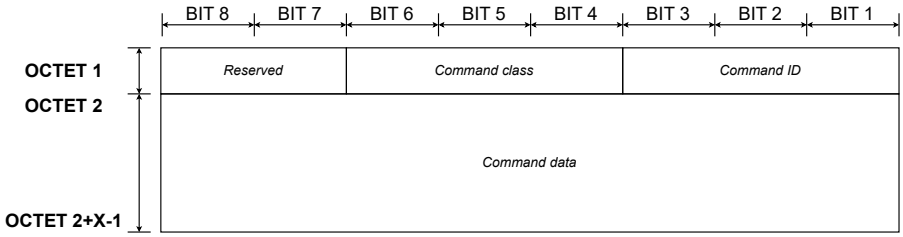


Fig. 3.3 Application layer - Command format - Case 1

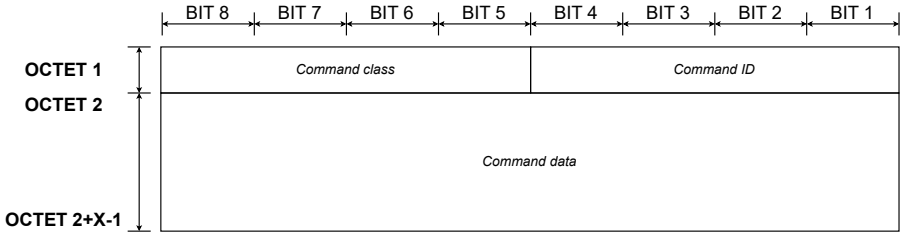


Fig. 3.4 Application layer - Command format - Case 2

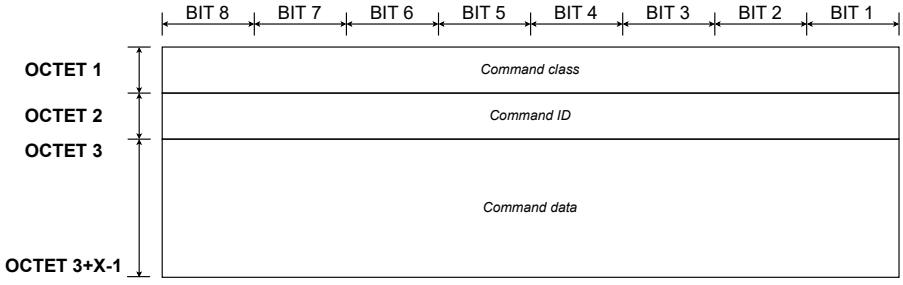


Fig. 3.5 Application layer - Command format - Case 3

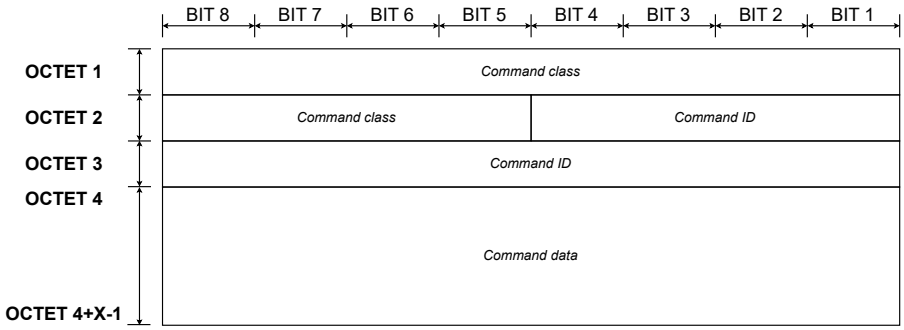


Fig. 3.6 Application layer - Command format - Case 4

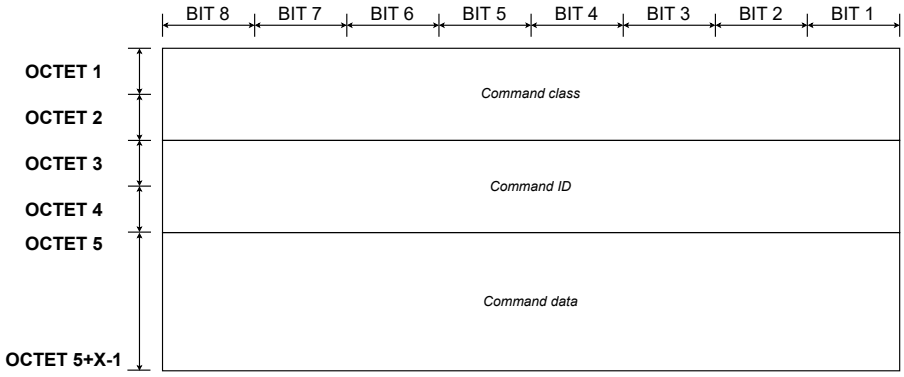


Fig. 3.7 Application layer - Command format - Case 5

3.2 Application commands

Application commands are divided into four types that are coded in the Presentation layer in the *opc* field. The type and the main purpose of each one are as follows:

1. Get/Read - the sender requests data to be returned from the receiver
2. Set/Write - the sender requests an action to be performed by the receiver
3. Notify - the sender provides an information to the receiver
4. Response - the receiver uses it to respond to Get/Read, Set/Write or Notify command types

The sender shall always provide information whether the response from the receiver is required (see *resp_req* field in the Presentation layer frame format, shown in Figure 5.3). The response type command may be requested in the following cases:

1. It shall be always requested in case of Get/Read command. The response command shall include requested data.
2. It may be requested for Set/Write command type. The response may not include the command payload but it shall include the status of the performed operation (see *status_hdr_inc*, *status_code_inc* and *status_code* fields in the Presentation layer frame format, shown in Figure 5.3).
3. It may be requested for Notify command type.

Each command is uniquely identified by: Command class, Command ID and sequence number (see *seq_num* field in the Presentation layer frame format, shown in Figure 5.3). A new sequence number shall be generated for each Get/Read, Set/Write and Notify command. The response command type shall have a copy of the sequence number that was included in the related Get/Read, Set/Write or Notify command.

Commands are sent in an asynchronous manner. The sender may not wait for a response before another command is sent to the recipient - e.g. it may send a few commands subsequently that are uniquely identified and wait for responses that are related to the sent commands (see Figure 3.8).

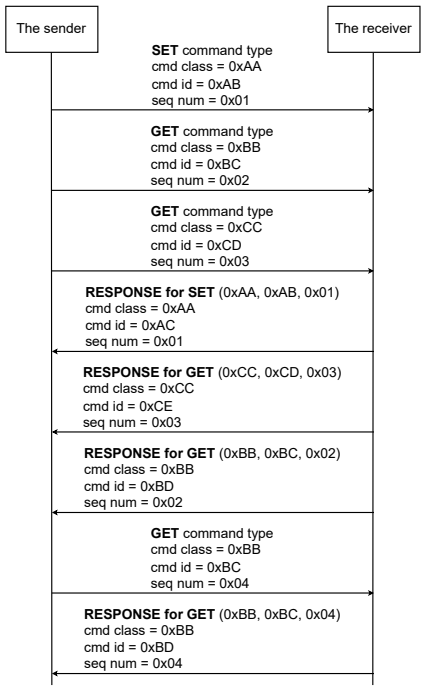


Fig. 3.8 Commands and responses

3.2.1 Custom commands

Custom commands are used to exchange data between a user application running on an Endpoint and an application service that is in the cloud. The user application data is included in the command data field.

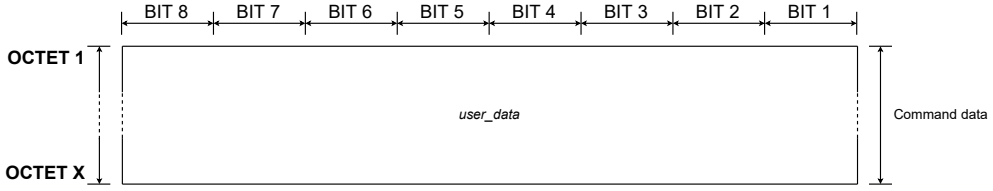


Fig. 3.9 Custom command - Command data format

1. Command class

Mnemonic	Length	Value	Description
cmd_class	3b	0x01	The value is common for all custom commands.

2. Command ID

Mnemonic	Length	Value	Type	Command name	Command direction
cmd_id	3b	0x00	Read	CUSTOM_GET_CMD	UL, DL
		0x00	Write	CUSTOM_SET_CMD	UL, DL
		0x00	Notify	CUSTOM_NOTIFY_CMD	UL, DL
		0x00	Response	CUSTOM_RESP_CMD	UL, DL

3. Command data - User application data

Mnemonic	Length	Value	Description
user_data	X B	data bytes	A user application data.

The field contains user application data. The included data length shall not exceed:

(a) 200 B for SubG-FSK Endpoints

(b) 19 B for SubG-CSS Endpoints

3.2.2 Management commands

3.2.2.1 Time management commands

The set of time management commands contains the following ones: GET_TIME and RESP_TIME. Commands are used by an Endpoint for time synchronization with the Amazon Sidewalk Cloud. For a detailed description of Global Clock Synchronization [GCS], see Section 4.2.

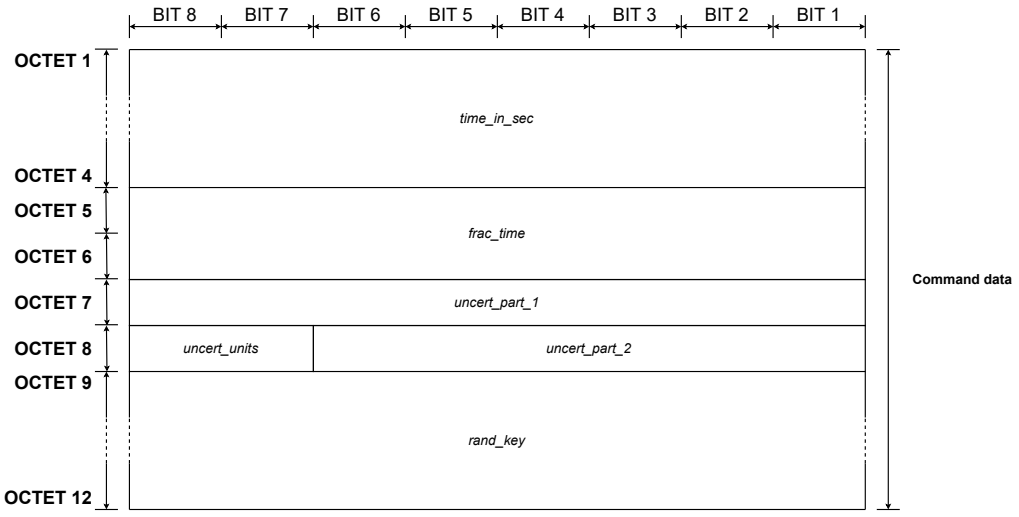


Fig. 3.10 Time commands - Command data format

1. Command class

Mnemonic	Length	Value	Description
cmd_class	12b	0x00	The value is common for all management commands.

2. Command ID

Mnemonic	Length	Value	Type	Command name	Command direction
cmd_id	12b	0x108	Read	GET_TIME	UL
		0x108	Response	RESP_TIME	DL

3. Command data - Time in seconds

Mnemonic	Length	Value	Description
time_in_sec	4B	time	The value is presented in seconds.

The field indicates the time since the GPS epoch (00:00:00 Jan 6, 1980).

4. Command data - Fractional time

Mnemonic	Length	Value	Description
fract_units	2B	fractional time	The value is expressed in units of 1/32768 s (30.5 us).

The field indicates the remaining fractional time.

5. Command data - Uncertainty units

Mnemonic	Length	Value	Description
uncert_units	2b	0x0	1/32768 s (30.5us)
		0x1	32/32768 s (0.97 ms)
		0x2	1024/32768 s (31.25 ms)
		0x3	8192/32768 s (250 ms)

The field indicates the error/uncertainty units of a single bit of uncertainty in time that is presented by two fields: *uncert_part_1* and *uncert_part_2*.

The maximum value of error/uncertainty in time based on units is as follows:

(a) for 0x0 maximum is 0.5 s

(b) for 0x1 maximum is 16 s

(c) for 0x2 maximum is 8.5 min

(d) for 0x3 maximum is 1.13 h

6. Command data - Uncertainty

Mnemonic	Length	Value	Description
uncert_part_1	1B	Less significant part of current error/uncertainty.	This part presents bits 0 - 7 of uncertainty
uncert_part_2	6b	Most significant part of current error/uncertainty.	This part presents bits 8 - 13 of uncertainty

The field indicates the current error/uncertainty. The total of the error/uncertainty is 14b. The value is split into two fields *uncert_part_1* and *uncert_part_2* in the frame. The value is expressed in units specified by *uncert_units*.

7. Command data - Random key

Mnemonic	Length	Value	Description
rand_key	4B	random value	The value is generated by the sender device.

RESP_TIME command shall include the copy of random key received in GET_TIME command.

3.2.2.2 Join commands

Join command *Join-Req Message* uses Amazon Sidewalk CMD ID 'SET_JOIN_REQ'. Join command *Join-Resp Message* uses Amazon Sidewalk CMD ID 'SET_JOIN_RESP'. They are used for initial connection establishment, namely Network Join procedure as well as periodic connection refreshes, namely Network Sync procedures as detailed in Section 8.3. The information content carried with these commands vary depending on use case, i.e. whether used for Network Join or Network Sync processes, as well as the communication mode and profile configuration of the Endpoint. The contents of *Join-Req Message* and *Join-Resp Message* are listed in Table 3.1 and Table 3.2, and detailed in section 3.2.2.2.3.

Profile	Use Case	Tag and Length Identifier (TL)	Profile Attributes
Async Connection Mode Profiles A, B, D	Network Join	0xC407	1. Device Profile ID (1B) 2. Window validity period in seconds (2B) 3. Window count (1B) 4. Receive window interval (2B) 5. Periodic uplink interval (Provable reachability interval) (1B)
	Network Sync	0xC407	1. Device Profile ID (1B) 2. Window validity period in seconds (2B) 3. Window count (1B) 4. Receive window interval (2B) 5. Periodic uplink interval (Provable reachability interval) (1B)
Sync Connection Mode Profile 1	Network Join	0xC305	ID of the selected GW (5B)
	Network Sync	0xC305	ID of the selected GW (5B)
Sync Connection Mode Profile 2	Network Join	0x4D	UL Link Latency (1B) and Load (1B)
		0x4C	DL Link Latency (1B) and Load (1B)
		0xC305	ID of the selected GW (5B)
		0x09	Link Duration (1B)
	Network Sync	0xC305	ID of the selected GW (5B)

Table 3.1 Information content carried in the JOIN-REQ CMD Payload

Profile	Use Case	Tag and Length Identifier (TL)	Profile Attributes
Async Connection Mode Profiles A, B, D	Network Join	0x00	Success or Failure (1B)
	Network Sync	0x00	Success or Failure (1B)
Sync Connection Mode Profile 1	Network Join	0x00	Success or Failure (1B)
	Network Sync	0x00	Success or Failure (1B)
Sync Connection Mode Profile 2	Network Join	0x08	Request Accept Reject (1B)
		0x46	UL Schedule (2B)
		0x41	DL Schedule (2B)
		0x47	Contention Param (2B)
		0x09	Link Duration (1B)
	Network Sync	0x00	Success or Failure (1B)

Table 3.2 Information content carried in the JOIN-RESP CMD Payload

3.2.2.2.1 Command class

Mnemonic	Length	Value	Description
cmd_class	12b	0x00	The value is common for all management commands.

3.2.2.2.2 Command ID

Mnemonic	Length	Value	Type	Command name	Command direction
cmd_id	12b	0x22F	Write	SET_JOIN_REQ	UL
		0x22F	Response	RESP_JOIN_REQ	DL

3.2.2.2.3 Command Payload

The command payloads of the 'Join Commands' carry several different fields of information content as listed in Tables 3.1 and 3.2. Each payload can carry several different information fields, each of which encoded following a TLV format (type-length-value) differentiating information from one another. The type and length fields indicate the type and size of the information content in the field followed by a value field carrying the information. In the case of asynchronous connection mode profiles, 'SET_JOIN_REQ' payload contains a single TLV field, namely 'Async profile attributes'. In the case of synchronous connection mode profile 2's 'Join Process' usage, 'SET_JOIN_REQ' CMD payload contains four different individually encoded TLV fields. All other usages include only a single TLV encoded field.

- Async profile attributes (0x04) This information field has 7 B of information payload requiring 2 B TL field with a value of '0xC407' making the total CMD payload size of 9 B. The information content is listed in order as follows:

  1. Device Profile id

Mnemonic	Length	Value	Description
profile_id	1B	0x80	Profile A
		0x81	Profile B

  The field indicates the Power Profile ID used by an Endpoint.

2. Window Validity Period

Mnemonic	Length	Value	Description
val_period	2B	0x0	Unlimited validity period.
		0x0001 - 0xFFFF	Validity period in seconds.

The field indicates the time period when an Endpoint is accessible by the cloud. The time counter starts after the last transmission received by the cloud. Downlink transmission shall not be performed after the validity period is expired by the cloud. The cloud shall wait until the next uplink frame to transmit its data in the downlink direction.

3. Number of downlink opportunities

Mnemonic	Length	Value	Description
nb_dl_slot	1B	0, 5, 10, 15, 20	The number of downlink opportunities.

The field indicates the number of downlink opportunities that are available after uplink transmission. Value 0 means infinite downlink opportunities. For more information about *nb_dl_slot* see Section 5.3.1.

4. Downlink opportunities separation

Mnemonic	Length	Value	Description
dl_separation	2B	5000	DL opportunities separation in ms.

The field indicates the time period between two consecutive downlink opportunities. For detailed information about *dl_separation* parameter, see Section 5.3.1, Section 5.5.2.1, and Section 5.5.2.2.

5. Uplink frequency of synchronization packets

Mnemonic	Length	Value	Description
ul_sync_interval	1B	0x00	Profile A
		0x05	Profile B

The field indicates the time interval of between subsequent uplink synchronization packets in minutes. Uplink frequency of synchronization packets may be considered also as Provable Reachable Interval (PRI) that identifies the longest interval since last uplink when the cloud may expect uplink data. It does not limit the ability of an Endpoint to send more uplink packets. The value may differ for selected Endpoints' Connection Profiles. For detailed information about *ul_sync_interval* parameter, see see Section 5.5.2.1, and Section 5.5.2.2.

Value 0 means that the time interval is undefined. It means that the cloud does not know when an uplink packet may come.

- Synchronous Connection Mode Profile 2 UL Link Characteristics (0xD)

  This information field has 2 B of payload requiring 1 B TL field with a value of '0x4C'. The first byte of the value contains indicates the desired interval between UL opportunities in increments of 63 ms. The second byte of the information contains an enum indicating the expected load of the node.

- Synchronous Connection Mode Profile 2 DL Link Characteristics (0xD)

  This information field has 2 B of payload requiring 1 B TL field with a value of '0x4C'. The first byte of the value contains indicates the desired interval between DL opportunities in increments of 63 ms. The second byte of the information contains an enum indicating the expected load of the node.

- Synchronous Connection Mode - Node ID (0x3)

  This information field has 5 B of value, i.e. requiring an extra byte of explicit length field, making the TL field give a value of '0xC305'. The information contains the ID of the Gateway selected as the *Sidewalk Relay*, as discovered by its 'Beacon' packet contents.

- Synchronous Connection Mode Link Duration (0x9)

  This information field has 1 B of payload requiring 1 B TL field with a value of '0x09'. The information contains link reservation duration at the *Sidewalk Relay* without requiring any transmissions activity. Prior to expiration of the reservation, the Endpoints can extend this reservation via an uplink packet for another full reservation period. In 'SET_JOIN_REQ', the information field indicates the desired wake-up characteristics of the Endpoint. In 'SET_JOIN_RESP', the information field indicates the requirement set by the Gateway keeping the reservation. The Endpoint adopts to the value in the 'SET_JOIN_RESP'.

- Status response (0x0) The field indicates whether SET_JOIN_REQ command is either accepted or rejected. The field is commonly used in Synchronous and Asynchronous connection modes.

Mnemonic	**Length**	**Value**	**Description**
status_resp	1B	0x00	accepted
		0x01	rejected

3.2.2.3 Device profile commands

Device profile commands are applicable to SubG-CSS devices only. Commands purpose is:

1. To get information related to the current configuration of the Connection Profiles used by the Endpoint (GET_DEVICE_PROFILE, RESP_DEVICE_PROFILE)
2. To request the Endpoint to set the selected Connection Profile using the configuration provided (SET_DEVICE_PROFILE)

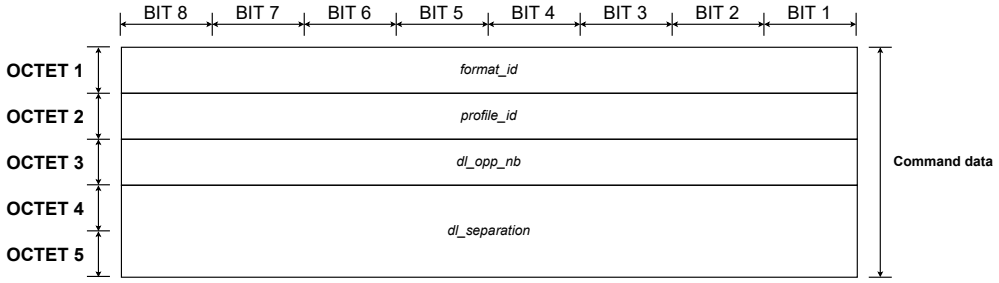


Fig. 3.11 SET-DEVICE-PROFILE and RESP-DEVICE-PROFILE commands - Command data format

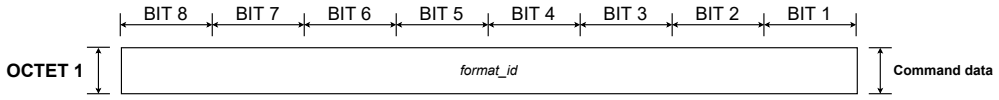


Fig. 3.12 GET-DEVICE-PROFILE command - Command data format

1. Command class

Mnemonic	Length	Value	Description
cmd_class	12b	0x00	The value is common for all management commands.

2. Command ID

Mnemonic	Length	Value	Type	Command name	Command direction
cmd_id	12b	0x230	Read	GET_DEVICE_PROFILE	DL
		0x230	Write	SET_DEVICE_PROFILE	DL
		0x230	Response	RESP_DEVICE_PROFILE	UL

3. Command data - Format ID

Mnemonic	Length	Value	Description
format_id	1B	0x01	SubG-CSS device Connection Profile settings.

4. Commands data - Profile ID

Mnemonic	Length	Value	Description
profile_id	1B	0x80	Profile A
		0x81	Profile B

The field indicates the Connection Profile ID used by an Endpoint.

5. Command data - Number of downlink opportunities

Mnemonic	Length	Value	Description
nb_dl_slot	1B	0, 5, 10, 15, 20	The number of downlink opportunities.

The field indicates the number of downlink opportunities that are available after uplink transmission. Value 0 means infinite downlink opportunities. For detailed information about *nb_dl_slot* parameter, see Section 5.3.1.

6. Command data - Downlink opportunities separation

Mnemonic	Length	Value	Description
dl_separation	2B	5000	DL opportunities separation in ms.

The field indicates the time period between two consecutive downlink opportunities.

3.2.2.4 Reset commands

The reset command is used by the Amazon Sidewalk Cloud to request a reset type to be performed by the Endpoint. The SET_RESET command is used for this purpose. The Endpoint confirms reset execution using NOTIFY_RESET command.

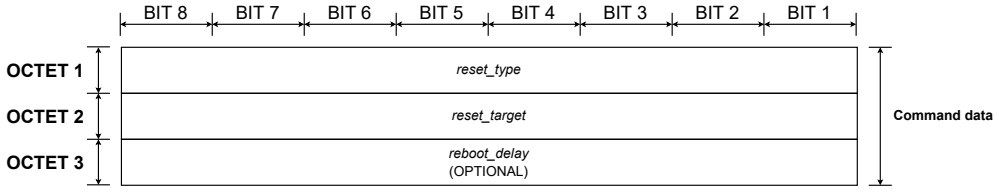


Fig. 3.13 SET-RESET command - Command data format

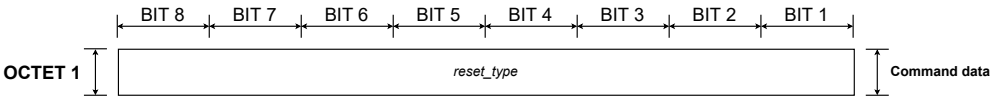


Fig. 3.14 NOTIFY-RESET command - Command data format

1. Command class

Mnemonic	Length	Value	Description
cmd_class	12b	0x00	The value is common for all management commands.

2. Command ID

Mnemonic	Length	Value	Type	Command name	Command direction
cmd_id	12b	0x112	Write	SET_RESET	DL
		0x112	Notification	NOTIFY_RESET	UL

3. Command data - Reset type

Mnemonic	Length	Value	Description
reset_type	1B	0x00	Soft reset clears the persistent configuration. Requires factory reset to be performed.
		0x01	Hard reset is the same as Soft reset.
		0x02	Reboots the device.

4. Command data - Reset target

Mnemonic	Length	Value	Description
reset_target	1B	0x00	RESET_TARGET_ALL
		0x01	RESET_TARGET_MCU
		0x02	RESET_TARGET_HOST

5. Command data - Reboot delay

Mnemonic	Length	Value	Description
reboot_delay	1B	0x00 - 0xFF	Reboot delay in seconds.

3.2.3 Metrics commands

Metrics Commands report transmit and receive metrics for synchronous and asynchronous connections. Values are determined by the Endpoint, and sent to the cloud frequently. RESP_ENHANCED_LDR_METRICS shall be used in asynchronous access mode only (SubG-CSS). RESP_ENHANCED_P2P_METRICS and NOTIFY_P2P_METRICS shall be used in synchronous access mode only (SubG-FSK). Metrics included in RESP_ENHANCED_LDR_METRICS are also available in RESP_ENHANCED_P2P_METRICS.

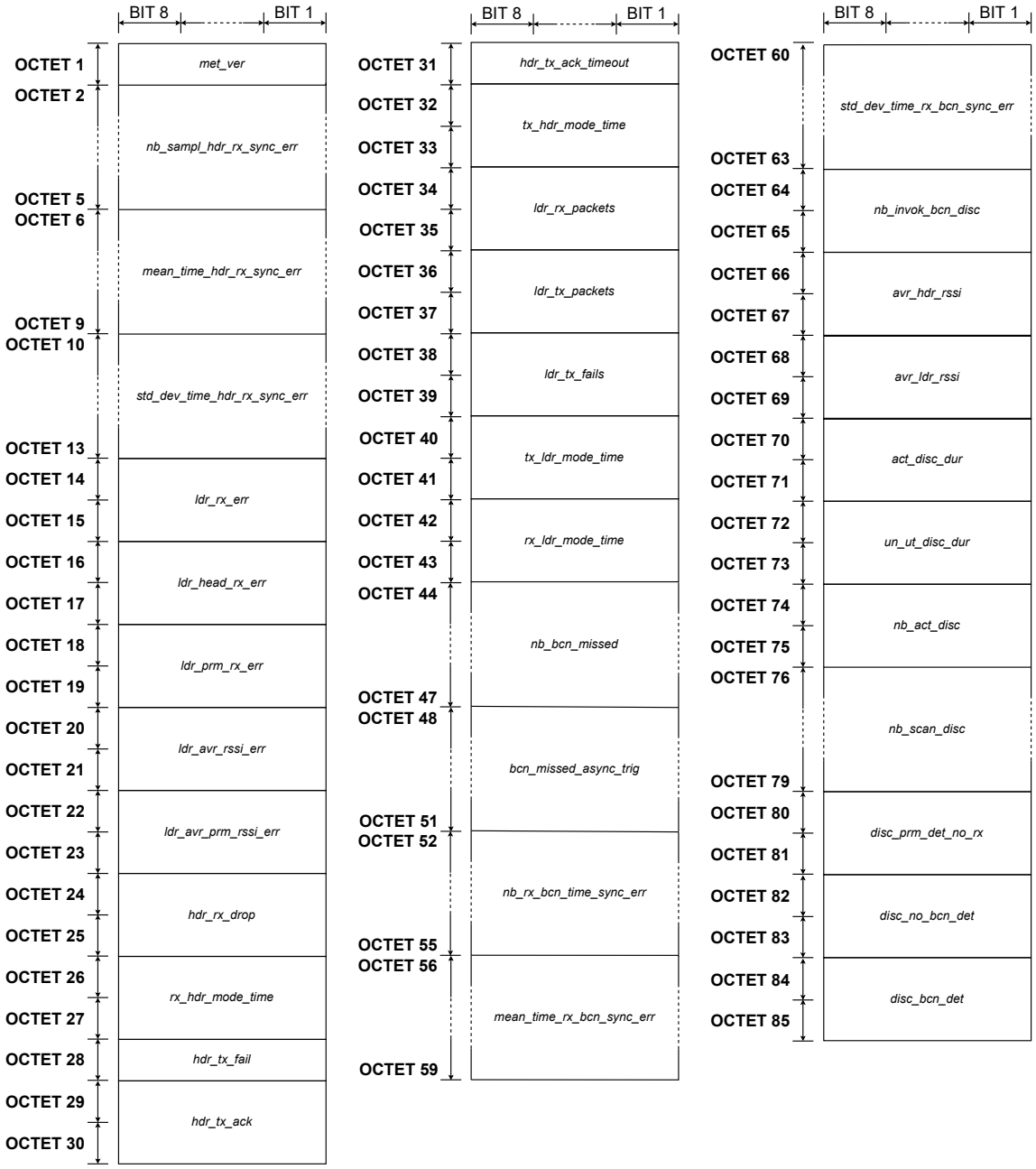


Fig. 3.15 RESP-ENHANCED-P2P-METRICS command - Command data format

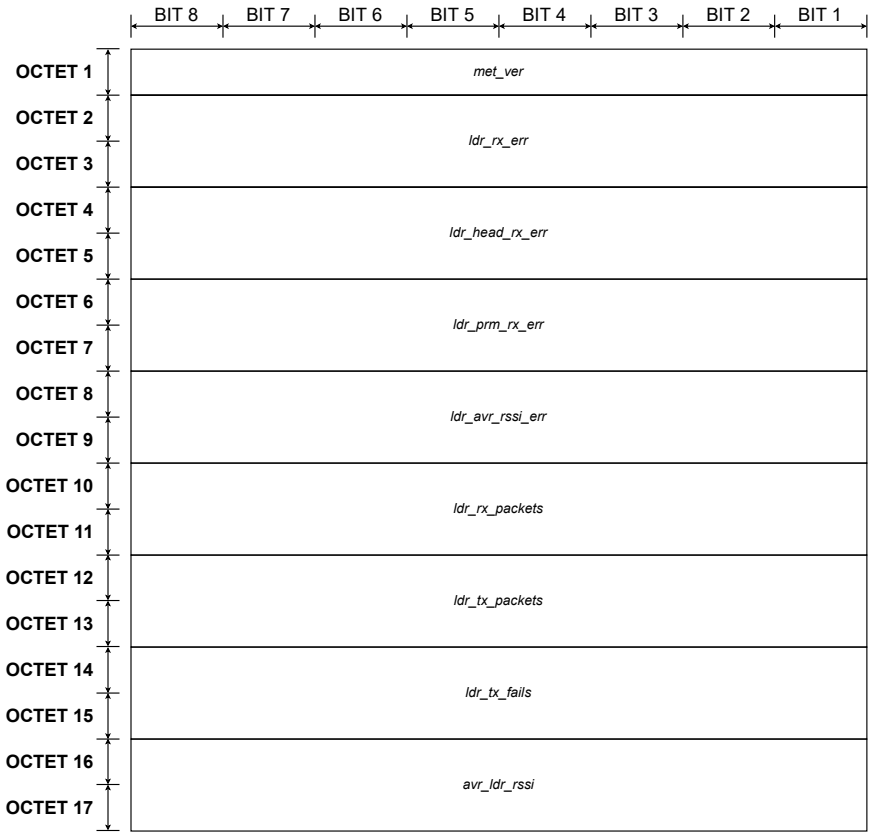


Fig. 3.16 RESP-ENHANCED-LDR-METRICS command - Command data format

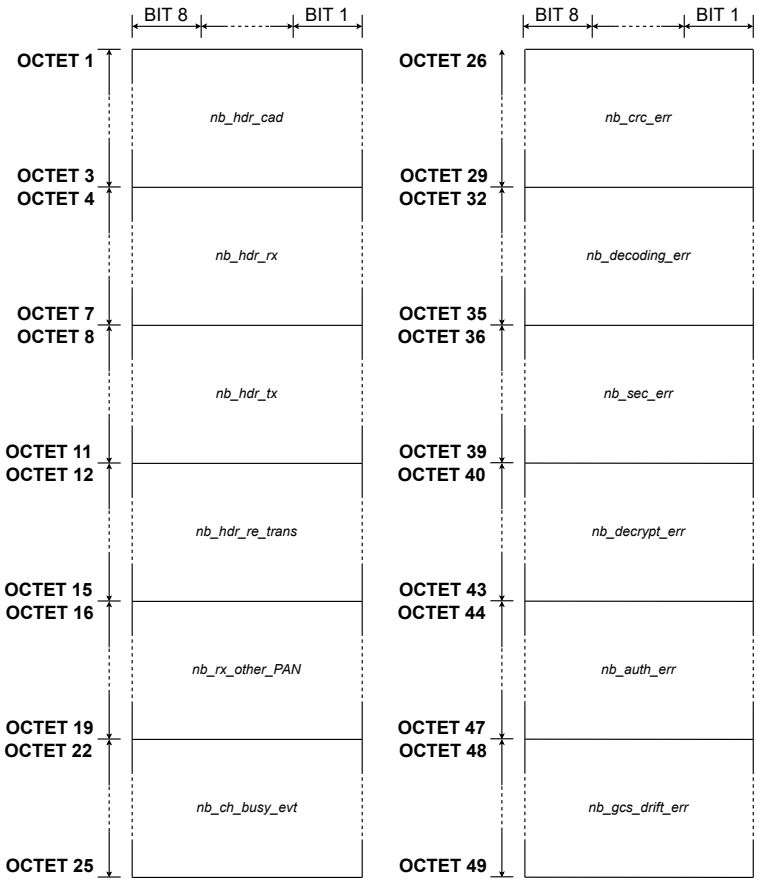


Fig. 3.17 NOTIFY-P2P-METRICS command - Command data format

1. Command class

Mnemonic	Length	Value	Description
cmd_class	4b	0x0B	The value is common for all metrics commands.

2. Command ID

Mnemonic	Length	Value	Type	Command name	Command direction
cmd_id	4b	0x01	Notify	NOTIFY_P2P_METRICS	UL
		0x08	Response	RESP_ENHANCED_P2P_METRICS	UL
		0x09	Response	RESP_ENHANCED_LDR_METRICS	UL

NOTE: RESP_ENHANCED_P2P_METRICS and RESP_ENHANCED_LDR_METRICS are Response type commands but act as Notify commands types. This means that there is no get/read command type requesting a metrics report to be sent.

3. Command data - Metrics version

Mnemonic	Length	Value	Description
met_ver	1B	0x00 - 0xFF	Metrics version number.

4. Command data - Number of samples for mean time of RX synchronization error calculation

Mnemonic	Length	Value	Description
nb_sampl_hdr_rx_sync_err	4B	0x00000000 - 0xFFFFFFFF	Number of samples.

The number of samples is used for calculation of the mean time synchronization error (see field *mean_time_hdr_rx_sync_err*) and the variance of time sync error (see *std_dev_time_hdr_rx_sync_err*) on synchronously received messages.

5. Command data - Mean time synchronization error

Mnemonic	Length	Value	Description
mean_time_hdr_rx_sync_err	4B	0x00000000 - 0xFFFFFFFF	Mean time sync error on synchronously received messages.

The value is expressed in units of 33 us, e.g. 0x00000004 is equal to 132 us.

6. Command data - Variance of time synchronization error

Mnemonic	Length	Value	Description
std_dev_time_hdr_rx_sync_err	4B	0x00000000 - 0xFFFFFFFF	N times the variance of time sync error on synchronously received messages.

The number of N value is indicated by *nb_sampl_hdr_rx_sync_err* field.

7. Command data - Number of SubG-CSS RX errors

Mnemonic	Length	Value	Description
ldr_rx_err	2B	0x0000 - 0xFFFF	Total number of reception errors.

8. Command data - Number of header RX errors for SubG-CSS

Mnemonic	Length	Value	Description
ldr_head_rx_err	2B	0x0000 - 0xFFFF	Total number of SubG-CSS RX header CRC errors.

9. Command data - Number of preamble timeouts for SubG-CSS

Mnemonic	Length	Value	Description
ldr_prm_rx_err	2B	0x0000 - 0xFFFF	Total number of SubG-CSS Preamble timeouts. It is incremented when radio detects a preamble but fails to get a sync word.

10. Command data - SubG-CSS average RSSI of reception errors

Mnemonic	Length	Value	Description
ldr_avr_rssi_err	2B	0x0000 - 0xFFFF	The average RSSI of all reception errors for SubG-CSS (see *ldr_rx_err*).

The field is mapped into real RSSI value in the following way:

(a) all bits of *ldr_avr_rssi_err* shall be negated

(b) MSB is dropped and LSB value plus 1 indicates the RSSI value reported in the metric

e.g. 0xFF74 indicates RSSI value -140 dBm:

(a) NOT(0xFF74) = 0x008B

(b) 0x8B = 139, 139+1 = 140, RSSI = -140 dBm

11. Command data - SubG-CSS average RSSI of preamble errors

Mnemonic	Length	Value	Description
ldr_avr_prm_rssi_err	2B	0x0000 - 0xFFFF	The average RSSI of all pramble timeout errors for SubG-CSS (see *ldr_prm_rx_err*).

Coding of RSSI value is described in the bullet "Command data - SubG-CSS average RSSI of reception errors".

12. Command data - SubG-FSK message drops

Mnemonic	Length	Value	Description
hdr_rx_drop	2B	0x0000 - 0xFFFF	The number of dropped messages on SubG-FSK. The value is incremented when a duplicated message is detected.

13. Command data - Reception time in SubG-FSK mode

Mnemonic	Length	Value	Description
rx_hdr_mode_time	2B	0x0000 - 0xFFFF	The total time spend for reception in SubG-FSK mode by the Endpoint.

The value is expressed in seconds.

14. Command data - Number of transmission failures in SubG-FSK

Mnemonic	Length	Value	Description
hdr_tx_fail	1B	0x00 - 0xFF	The value of TX failures. The value is incremented when Link layer ACK is not received after a message is transmitted.

15. Command data - Number of acknowledges in SubG-FSK

Mnemonic	Length	Value	Description
hdr_tx_ack	1B	0x00 - 0xFF	The number of Link layer acknowledgments. The value is incremented when ACK is either sent or received.

16. Command data - Number of acknowledge timeouts in SubG-FSK

Mnemonic	Length	Value	Description
nb_hdr_ack_timeout	1B	0x00 - 0xFF	The total numbers of Link layer acknowledgment timeouts.

17. Command data - Transmission time in SubG-FSK mode

Mnemonic	Length	Value	Description
tx_hdr_mode_time	2B	0x0000 - 0xFFFF	The total time spend for transmission in SubG-FSK mode by the Endpoint.

The value is expressed in seconds.

18. Command data - Number of received messages in SubG-CSS mode

Mnemonic	Length	Value	Description
ldr_rx_packets	2B	0x0000 - 0xFFFF	The total number of successfully received messages in SubG-CSS mode.

19. Command data - Number of transmitted messages in SubG-CSS mode

Mnemonic	Length	Value	Description
ldr_tx_packets	2B	0x0000 - 0xFFFF	The total number of transmitted messages in SubG-CSS mode. The value is incremented when transmission is successful.

20. Command data - Number of transmission failures in SubG-CSS mode

Mnemonic	Length	Value	Description
ldr_tx_fails	2B	0x0000 - 0xFFFF	The total number of transmission failures.

21. Command data - Transmission time in SubG-CSS

Mnemonic	Length	Value	Description
tx_ldr_mode_time	2B	0x0000 - 0xFFFF	The total time spend for transmission in SubG-CSS mode by the Endpoint.

The value is expressed in seconds.

22. Command data - Reception time in SubG-CSS

Mnemonic	Length	Value	Description
rx_ldr_mode_time	2B	0x0000 - 0xFFFF	The total time spend for reception in SubG-CSS mode by the Endpoint.

The value is expressed in seconds.

23. Command data - Number of Beacons missed

Mnemonic	Length	Value	Description
nb_bcn_missed	4B	0x00000000 - 0xFFFFFFFF	Total number of Beacons missed by the Endpoint.

24. Command data - Number of switches from synchronous to asynchronous mode

Mnemonic	Length	Value	Description
bcn_missed_async_trig	4B	0x00000000 - 0xFFFFFFFF	Total number of switches from from synchronous mode to asynchronous mode.

25. Command data - Number of samples for mean time of RX synchronization error calculation for Beacons

Mnemonic	Length	Value	Description
nb_rx_bcn_time_sync_err	4B	0x00000000 - 0xFFFFFFFF	Number of samples used for calculation of Beacons mean time synchronization error.

The number of samples is used for calculation of the following:

- The mean time synchronization error of Beacons (see field *mean_time_rx_bcn_sync_err*) on synchronously received messages.
- The variance of time sync error of Beacons (see field *std_dev_time_rx_bcn_sync_err*) on synchronously received messages.

26. Command data - Mean time synchronization error of Beacons

The value is expressed in units of 33 us e.g. 0x00000004 is equal to 132 us.

Mnemonic	Length	Value	Description
mean_time_rx_bcn_sync_err	4B	0x00000000 - 0xFFFFFFFF	Mean time sync error on synchronously received Beacons.

Mnemonic	Length	Value	Description
std_dev_time_rx_bcn_sync_err	4B	0x00000000 - 0xFFFFFFFF	N times the variance of time sync error on received Beacons.

27. Command data - Variance of time synchronization error of beacons

    The number of N value is indicated by *nb_rx_bcn_time_sync_err* field.

28. Command data - Number of Beacon discovery invokes

Mnemonic	Length	Value	Description
nb_invok_bcn_disc	2B	0x0000 - 0xFFFF	The total number of times the Beacon discovery is invoked.

    The Beacon discovery is initiated every time when the synchronization with the Gateway is not present (see Section 5.2.2, and Section 5.2.1). The Beacon discovery may be invoked multiple times if the Beacon is not found. The parameter contains the number of Beacon discovery procedure starts.

29. Command data - SubG-FSK average RSSI

Mnemonic	Length	Value	Description
avr_hdr_rssi	2B	0x0000 - 0xFFFF	The average RSSI for SubG-FSK mode.

    The parameter contains the average RSSI of successfully received messages.

30. Command data - SubG-CSS average RSSI

Mnemonic	Length	Value	Description
avr_ldr_rssi	2B	0x0000 - 0xFFFF	The average RSSI for SubG-CSS mode.

    The parameter contains the average RSSI of successfully received messages.

31. Command data - Beacon discovery time

Mnemonic	Length	Value	Description
act_disc_dur	2B	0x0000 - 0xFFFF	The total amount of time spend in Beacon discovery procedure.

    The field contains cumulative value from all Beacon discovery procedures. The value is expressed in seconds.

32. Command data - No radio usage by discovery sub-module while active

33. Command data - Number of the discovery procedure activation

Mnemonic	Length	Value	Description
un_ut_disc_dur	2B	0x0000 - 0xFFFF	The amount of time, expressed in ms, discovery submodule did not use the radio while it was active. This duration includes scheduling overhead as well as radio time spent for other functions.

Mnemonic	Length	Value	Description
nb_act_disc	2B	0x0000 - 0xFFFF	The number of times when the Beacon discovery process was started.

Mnemonic	Length	Value	Description
nb_scan_disc	2B	0x0000 - 0xFFFF	Total number of scanned channels in the Beacon discovery process.

Mnemonic	Length	Value	Description
disc_prm_det_no_rx	2B	0x0000 - 0xFFFF	Total number of the preamble detections not finalized with the Beacon reception during the Beacon discovery procedure

34. Command data - Number of channels scanned by discovery procedure
35. Command data - Number of preamble detection in unsuccessful Beacon receptions
36. Command data - Number of non-Beacon messages reception

Mnemonic	Length	Value	Description
disc_no_bcn_det	2B	0x0000 - 0xFFFF	Total number of the packet receptions that were not recognized as Beacon during the discovery procedure.

37. Command data - Number of Beacon messages reception

Mnemonic	Length	Value	Description
disc_bcn_det	2B	0x0000 - 0xFFFF	Total number of the Beacon packet receptions during the Beacon discovery procedure.

38. Command data - Number of SubG-FSK preamble iterations

Mnemonic	Length	Value	Description
nb_hdr_cad	4B	0x00000000 - 0xFFFFFFFF	The number of SubG-FSK preamble cycle iterations.

39. Command data - Number of received messages in SubG-FSK mode

Mnemonic	Length	Value	Description
nb_hdr_rx	4B	0x00000000 - 0xFFFFFFFF	The total number of received messages in SubG-FSK mode. The value is incremented when reception is successful.

Mnemonic	Length	Value	Description
nb_hdr_tx	4B	0x00000000 - 0xFFFFFFFF	The total number of transmitted messages in SubG-FSK mode. The value is incremented when transmission is successful.

40. Command data - Number of transmitted messages in SubG-FSK mode

41. Command data - Number of retransmissions

Mnemonic	Length	Value	Description
nb_hdr_re_trans	4B	0x00000000 - 0xFFFFFFFF	Incremented when the TX in SubG-FSK has been rescheduled.

42. Command data - Number of detected busy channels in contention mechanism

Mnemonic	Length	Value	Description
nb_ch_busy_evt	4B	0x00000000 - 0xFFFFFFFF	The value indicates the number of busy channel during TX attempt (contention mechanism).

43. Command data - Number of header RX errors for SubG-FSK

Mnemonic	Length	Value	Description
nb_crc_err	4B	0x00000000 - 0xFFFFFFFF	Total number of SubG-FSK RX header CRC errors.

44. Command data - Number of message decoding errors

Mnemonic	Length	Value	Description
nb_decoding_err	4B	0x00000000 - 0xFFFFFFFF	Total number of decoding errors of received messages.

45. Command data - Number of messages without MAC included

Mnemonic	Length	Value	Description
nb_sec_err	4B	0x00000000 - 0xFFFFFFFF	Incremented when received message does not contain authentication tag (MAC).

46. Command data - Number of message decryption errors

Mnemonic	Length	Value	Description
nb_decrypt_err	4B	0x00000000 - 0xFFFFFFFF	Total number of decryption errors of received messages.

Mnemonic	Length	Value	Description
nb_auth_err	4B	0x00000000 - 0xFFFFFFFF	Total number of authentication errors of received messages.

47. Command data - Number of message authentication errors
48. Command data - Number of the Global Clock Synchronization time drift errors

Mnemonic	Length	Value	Description
nb_gcs_drift_err	4B	0x00000000 - 0xFFFFFFFF	Incremented when RESP_TIME is received and the time drift is greater than a certain value.

Chapter 4

Security

4.1 Architecture

Amazon Sidewalk is a shared network developed by Amazon to allow developers outside Amazon to create and bring to market all types of consumer, enterprise, and public sector smart and connected devices and services.

Amazon Sidewalk security features include: device authentication, data encryption, and more. These features encrypt and protect the network, maintain customer privacy and protect customer data.

The following key components are referenced throughout this chapter: Amazon Sidewalk Gateways, Amazon Sidewalk Endpoints, Amazon Sidewalk Network Server, and Application Servers. These key components are defined as follows:

- An **Endpoint** is a source and termination point of data for the RF protocol. As a hardware entity, an Endpoint contains safe storage for keys (including device and application keys and credentials) and a memory area for security procedures. The Endpoint security software is responsible for the ID obfuscation process.
- The **Endpoint Application Firmware** is a dedicated external application that is loaded onto an Endpoint for handling application business logic tasks. Software is signed using the device manufacturer's firmware signing key during the manufacturing process.
- The **Gateway** is a hardware entity in the Amazon Sidewalk network. A Gateway provides storage for keys and credentials and a memory area for security procedures.
- The Amazon Sidewalk **Gateway Software** is an Amazon Sidewalk-dedicated internal software module for routing Amazon Sidewalk protocol messages and configuring local resources. Endpoint to Application Server traffic (payload) passes through the Gateway software. For SubG-CSS and SubG-FSK, the Flex Layer has a third layer of encryption to encrypt the communication between the Gateway MCU and the Amazon Sidewalk Cloud.
- The Amazon Sidewalk **Network Server** is a dedicated cloud service responsible for time-based ID obfuscation, rotation and network key management. The service is also responsible for managing device primary keys and the certificate chain, and for terminating the Flex Layer and the Network Layer. The Amazon Sidewalk Network Server is the source of the time in GCS functionality that is part of each ciphering procedure used within the Amazon Sidewalk system. The routing table for Endpoint devices is stored in the Network Server.
- The **Application Server** is a dedicated cloud service. The Application Server is responsible for routing application traffic between external applications and applications in the Endpoint. AWS IoT Wireless is responsible for maintaining Endpoint application credentials for the Application Server.

Encrypted links in the Amazon Sidewalk system are listed below:

- **Endpoint** to **Gateway** content is handled by Amazon Sidewalk protocol. Security aspects are described in Section 4.6.1
- **Gateway** to Amazon Sidewalk **Network Server** link is managed by the Flex Layer, see Section 4.8.1. An encrypted link is established during the Gateway provisioning process, see Section 4.4.1.1. Transmission is encrypted by the Gateway network security key. Keys are derived and rotated on an hourly basis, see Section 4.5.2. The Gateway primary key is refreshed on each Gateway device reboot.
- **Endpoint** to Amazon Sidewalk **Network Server** security content is handled by the Network Layer protocol, command messages are provided in the Presentation Layer protocol. See Section 5.1.1; for security aspects see section 5.1.2.2.2, and for multirate see Section 5.4.3. Encrypted Third Party Application traffic is encapsulated in flex frames. The Gateway uses the Amazon Sidewalk protocol to route flex frames to the Endpoint.
- **Endpoint** to **Application Server** communication is handled by the Application Layer protocol. Application key encryption is used to encrypt traffic between the Endpoint and the AWS IoT Wireless managed service. See Section 3.1; for key exchange procedures see Section 4.5; for encoding, decoding and related security aspects, see Section 4.7.
- Communication between **AWS IoT Wireless** service and **Third Party Application** is handled by AWS Cloud Services. The operation of AWS Cloud Services is outside the scope of this document.

The end-to-end system has an Amazon Sidewalk system and a Third party (Web-based) application.

The Amazon Sidewalk system has two hardware modules (Endpoint and Gateway) and five software modules (Endpoint Application, Gateway application, Mobile application, Network, and Application Servers in the cloud). Figure 4.1 shows a simplified view of the system components. AWS IoT asset services support Application Server with auxiliary data (device and user-related).

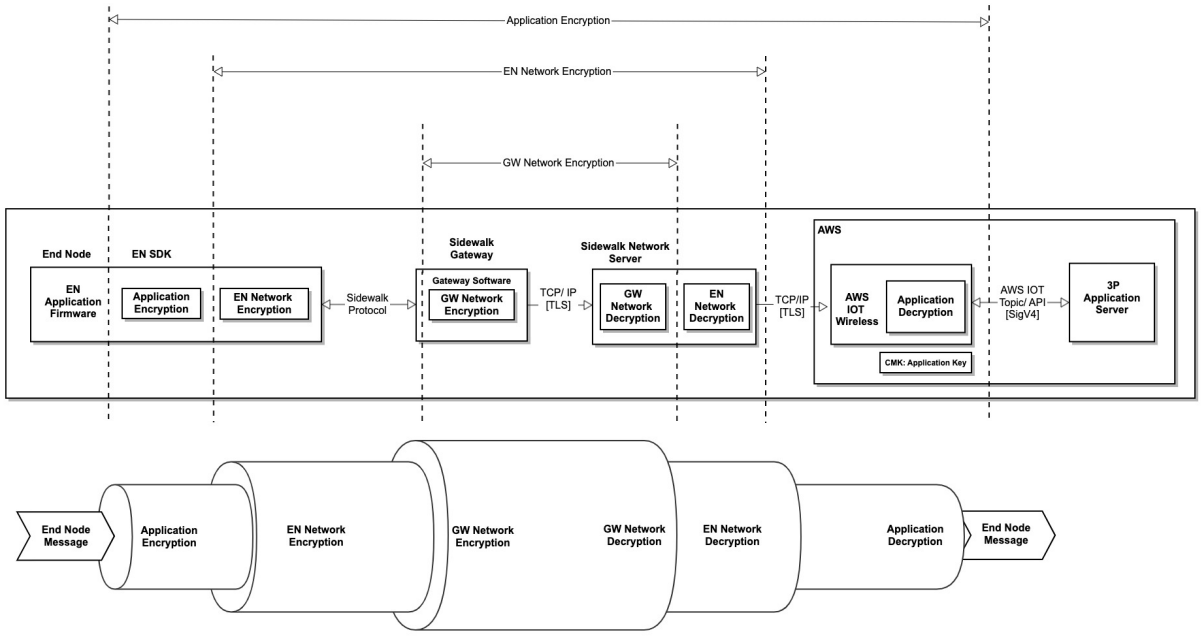


Fig. 4.1 Security Spec architecture diagram

4.1.1 Security model

The Amazon Sidewalk protocol ensures the security of transmitted data by encrypting the data. The Amazon Sidewalk protocol also supports device registration, authentication and authorization when connecting to the Amazon Sidewalk network.

Authorization in Amazon Sidewalk is based on Amazon Sidewalk's Public Key Infrastructure Certificate Authority system. A multilevel certificate chain is maintained, based on a self-signed root certificate authority (CA). Certificate issuing and signing and exchange of cryptographic keys is carried out using the asymmetric algorithm ECDH (elliptic curve Diffie-Hellman). Two sets of elliptic cryptographic curves (Ed25519 and p256r1) are used for keys and certificate generation. The Ed25519 certificate is used to define the identity of the device. The p256r1 is used to define the signature of the device. Both are needed to register a device with the certificates and to show proof of possession. Network data confidentiality and integrity are provided by the AES-GCM algorithm. Application data confidentiality is achieved using the symmetric key AES-CTR algorithm.

Endpoint pseudo-anonymity is achieved by Device ID obfuscation, and by address translation in the Network server. Each of the entities (device, service) has a set of credentials associated with it. Credentials consist of asymmetric keys, certificates and their signatures - see Section 4.1.3. Security aspects are implemented across the whole of the system's lifetime starting with device production, then FW management, connection to the user's network, maintaining transmission confidentiality, and ensuring pseudo-anonymity of devices in the network.

NOTE: End user identification and authentication are outside the scope of this document.

4.1.2 Certificate chain

The terms used to describe the certificate chain are described in Table 4.1.

Term	Definition
Certificate	An electronic document used to prove the validity of a public key. In Sidewalk, a certificate consists of three parts: the public key, the certificate serial number, and the certificate signature.
Certificate Chain	The certificate chain is a list of certificates used to authenticate an entity. The chain, begins with the certificate of that entity, and each certificate in the chain is signed by the entity identified by the next certificate in the chain.
Certificate Authority	A Certificate Authority (CA) is a certificate that issues (validates and signs) other certificates in the certificate chain.
Root Certificate Authority	A Root Certificate Authority (CA) forms the basis for the certificate chain. The root CA has the property that it is self-signed, issuing its own certificate, thus generating its own signature.
Intermediate Certificate Authority	An Intermediate Certificate Authority (CA) is a certificate in the chain that was issued (validated and signed) by another CA, and has the authority to issue other certificates (in contrast to the root CA which issues itself).
Device Attestation Key	The Device Attestation Key (DAK) is a private key issued by a Certificate Authority (CA) used to sign and issue device certificates during device provisioning and manufacturing.
Hardware Security Module	The Hardware Security Module (HSM) is a physical computing device that safeguards and manages digital keys. The HSM is used for computing device certificate signatures.

Table 4.1 Certificate chain

Amazon Sidewalk maintains two multilevel certificate chains using Ed25519 and p256r1 elliptic curves. Each Certificate Authority (CA) issues a certificate to the next level in a certificate chain. The certificate chain begins with the Amazon Root CA. The Device Attestation Key (DAK) is hosted on a Hardware Security Module (HSM). The HSM issues device certificates during the manufacturing provisioning process, see Section 4.4.1.1. Cloud certificates (Network Server certificate) are derived from the Amazon Root CA. Application server certificates are derived from the Manufacturer's certificate. Figure 4.2 shows a simplified certification chain.

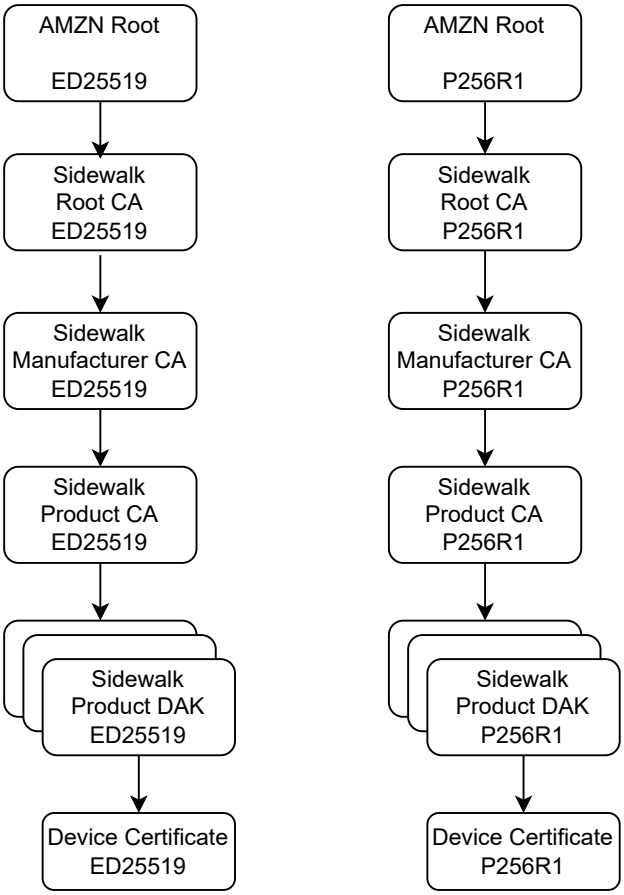


Fig. 4.2 Certification path

4.1.3 Certificates and device serial numbers

Amazon Sidewalk Certificates contain three fields - Serial, Public Key, and Signature. Amazon Sidewalk Certificates use the following layout:

Device certificates use the Amazon Sidewalk Manufacturing Serial Number (SMSN) as their serial number in the Serial field. Intermediate and root CAs use the following table to generate a 4 B serial number for the Serial field. A detailed fields description is in Table 4.2.

For more information on device serial number content see Section 4.3, for more information on the process of assigning numbers, see Section 4.4.1.1.

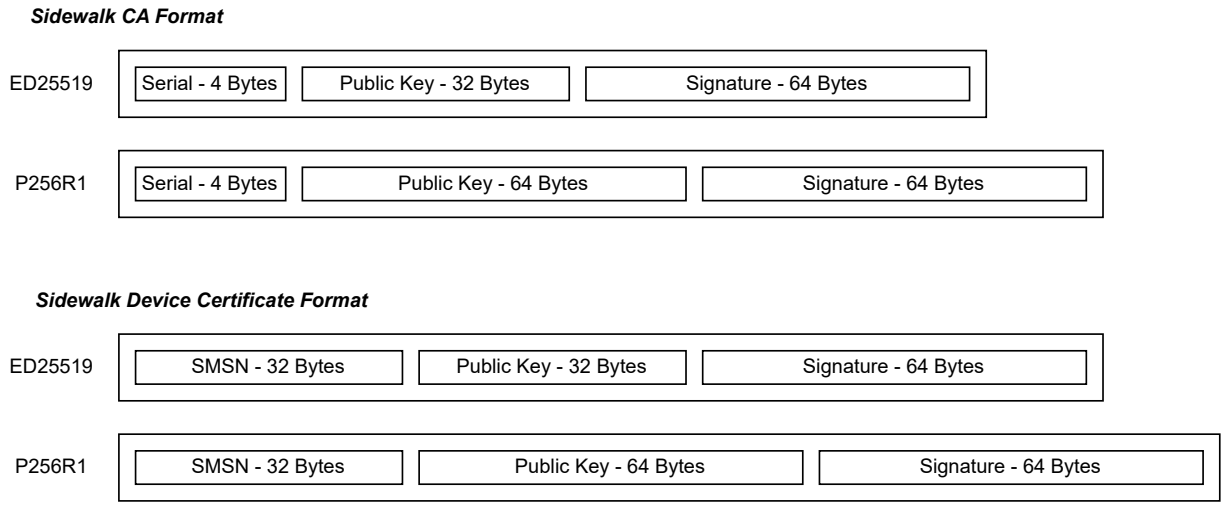


Fig. 4.3 Sidewalk certificate formats

Bit #	Content	Description	Notes
31..30	Type of certificate	0 -root, 1- intermediate, 2 - cloud	
29..28	Chain size	1-3 - depth of the full cert chain	
27..26	Depth in chain	0 -root (or cloud) 1 -3 - intermediate certificate	
25..23	Signing rights	0 - intermediate only 1 - intermediate and Endpoint 2 - intermediate and cloud 3 - intermediate, Endpoint and cloud cert 4 - Endpoint cert only 5 - cloud cert only 6 - Gateway 7 - All	Note: cloud can only sign messages
22 : 0	Subset of device serial number		Allows assignment of around 8.3M serial numbers

Table 4.2 Certificate serial number

4.1.4 Key types

There are two types of keys in the Amazon Sidewalk system: Asymmetric keys and Symmetric keys. Asymmetric keys are used to sign and verify the message. Symmetric keys are used to encrypt and decrypt the message. Asymmetric keys are built during the factory provisioning process, for more information, see Section 4.4.1.1. Symmetric keys are periodically derived from the primary symmetric key. Primary symmetric keys are derived during the device registration process, for more information, see Section 4.4.2.

Each symmetric key is presented in two versions: counter-based (CTRREF) and global clock-based.

Key Type	Key name	Usage	description	Notes
Asymmetric Key	DEVICE_ED25519_PRIVATE	Generate message signature using EDDSA	Ed25519 private key	Built during factory provisioning process
	DEVICE_ED25519_PUBLIC	Verify message signature EDDSA	Ed25519 public key	Built during factory provisioning process
	DEVICE_SECP256R1_PRIVATE	Generate message signature using ECDSA	secp256r1 private key	Built during factory provisioning process
	DEVICE_SECP256R1_PUBLIC	Verify message signature ECDSA	secp256r1 public key	Built during factory provisioning process
Symmetric Key	NETWORK_PRIMARY	Generate network symmetric security keys	Primary network key	Shared during device registration or key refresh process by ECDH
	APPLICATION_PRIMARY	Generate application symmetric security keys	Primary application key	Shared during device registration by ECDH-E

Table 4.3 Ephemeral key types: Symmetric security keys are derived in network nodes from a primary symmetric key and rotated periodically.

Link	Key name	Usage	Derivation	Rotation interval
Network Server	TX_ID_KEY	Generate TX_ID for WAN data packet	CMAC(WAN_Primary key, "Network_KDF_TX_UUID" |GCS time reference)	1 hour
	GCM_KEY	Encrypt/Decrypt data WAN packet with time reference	CMAC(Primary key, "Network_KDF_Device_Network_GCM" |GCS time reference)	1 hour
	GCM_CTRREF_KEY	Encrypt/Decrypt data WAN packet with counter-based reference	CMAC(Primary key, "Network_KDF_Dev_Ntw_GCM_COUNTER" |counter)	Whenever counter is updated
	TX_ID_CTRREF_KEY	Generate TX_ID for WAN time sync packet based on counter	CMAC(Primary key, "Network_KDF_Time_Sync_TX_UUID" |Time_Sync_Counter)	Whenever Time_Sync_Counter is increased
	CMAC_KEY	Generate authentication tag for time sync packets	CMAC(Primary key, "Network_KDF_Device_Network_CMAC" |GCS time reference)	1 hour
	CMAC_CTRREF_KEY	Generate authentication tag with counter-based reference	CMAC(Primary key, "Network_KDF_Dev_Ntw_CMAC_COUNTER" |counter)	Whenever counter is updated
	TSYNC_RESP_KEY	Generate time-based CMAC tag for WAN mode RESP_TIME command	CMAC(Primary key, "Network_KDF_Dev_Ntw_Tsync_Resp" |GCS time reference)	10 min
Application Server	TSYNC_RESP_CTRREF_KEY	Generate counter-based CMAC tag for WAN mode RESP_TIME command	CMAC(Primary key, "Network_KDF_Dev_Ntw_Tsync_Resp_Counter" |Time_Sync_Counter)	Whenever counter is updated
	JOIN_RESP_GCM_KEY	Encrypt/Decrypt WAN JOIN RESPONSE packet	CMAC(Primary key, "Network_KDF_Join_Resp_Dev_Ntw" |GCS time reference)	5 min
	APP_CTR_KEY	Encrypt/Decrypt (AES-CTR) application payload	CMAC(Application_Primary key, "Application_KDF_Device_Application_CTR" |GCS time reference)	1 hour
	APP_GCM_CTRREF_KEY	Encrypt/Decrypt (AES-GCM) application payload with counter-based reference	CMAC(Application_Primary key, "Application_KDF_Registration_GCM_COUNTER" |counter)	Whenever counter is updated

Table 4.4 Keys specification

4.1.5 The procedure of generating device private keys during MFG process

Each device requires two unique key pairs corresponding to the Ed25519 and p256r1 elliptic curves that are generated at the time of device manufacturing. The private keys are then stored in the device's persistent memory area. The device public key and the Amazon Sidewalk Manufacturing Serial Number (SMSN) are

signed using the DAK private key. The device public key and the Amazon Sidewalk Manufacturing Serial Number form the device certificate signature. Note the signing operation takes place in the HSM device, avoiding exposure of the DAK.

- ECDSA Signature = ECDSA(DAK p256r1 private key, Device p256r1 Public Key || SMSN)
- EdDSA Signature = EdDSA(DAK Ed25519 private key, Device Ed25519 Public Key || SMSN)

4.2 Global clock synchronization

Global clock synchronization is a key component in Amazon Sidewalk security. Each device in the network shall have its clock synchronized with the Amazon Sidewalk Cloud. The synchronized clock is the base for encrypted messages between Endpoints and the cloud. Devices that do not have a valid time synchronization can only exchange time synchronization commands with the Amazon Sidewalk Cloud.

The absolute time clock used in Amazon Sidewalk is Global Positioning System (GPS) Epoch time in seconds. The source of the correct time value is the Amazon Sidewalk Cloud. GCS time is verified by CMAC authentication between the Gateway and the Endpoint. For more information, see Figure 4.4.

The Gateway's clock is synchronized by exchanging GET_TIME / RESP_TIME messages using an encrypted tunnel to the Network server. For more information, see Section 8.1 and Section 4.2. An Endpoint in synchronous mode that has successfully finished the Beacon discovery process uses an uplink slot to send a GET_TIME command with a counter-based CMAC. An Endpoint in asynhronous mode sends a GET_TIME command after initialization. The GET_TIME command is received by the Gateway and forwarded by the Gateway plugin to the Network Server. The Network Server decodes and verifies the GET_TIME command. If the Network Server successfully verified the GET_TIME command the Network server returns a Notify command to the Gateway. The Notify Command from the Network Server contains a flex counter-based CMAC. The Gateway builds a RESP_TIME response command with the Gateway's GCS time. The data frame containing the RESP_TIME response command is encrypted with a counter-based CMAC and sent back to the Endpoint. If the Endpoint successfully received the RESP_TIME response command, the Endpoint synchronizes its own clock with the Gateway's clock source. Further data exchange uses GCS time-based security procedures.

Application commands used for the time synchronization are: GET_TIME, RESP_TIME. For more information on GET_TIME and RESP_TIME formats, see Section 4.2. End-to-end communications that use time synchronization commands shall use keys that are generated as follows:

1. If the Endpoint clock is not synchronized, the counter is used to generate keys.
2. If the Endpoint clock is synchronized, the Endpoint time is used to generate keys.

For more information on key management and key rotation see: Section 4.4. For more information on encrypted messaging see Section 4.7.

Configuration parameters of the time synchronization operations are:

1. *gcs_ret_num* - the value indicates the number of subsequent GET_TIME commands that may be sent with the first *gcs_resp_timeout* interval value if the RESP_TIME command is not received. When *gcs_ret_num* is reached then *gcs_resp_timeout* is increased.

   $gcs\_ret\_num = 3$

2. *gcs_resp_timeout* - the time interval that an Endpoint shall wait for a RESP_TIME command. The *gcs_resp_timeout* value is constant until the count of transmitted GET_TIME commands is less than or equal to *gcs_ret_num*. When the count of transmitted GET_TIME commands reaches *gcs_ret_num*, then the interval between subsequent GET_TIME commands is increased up to a defined maximum

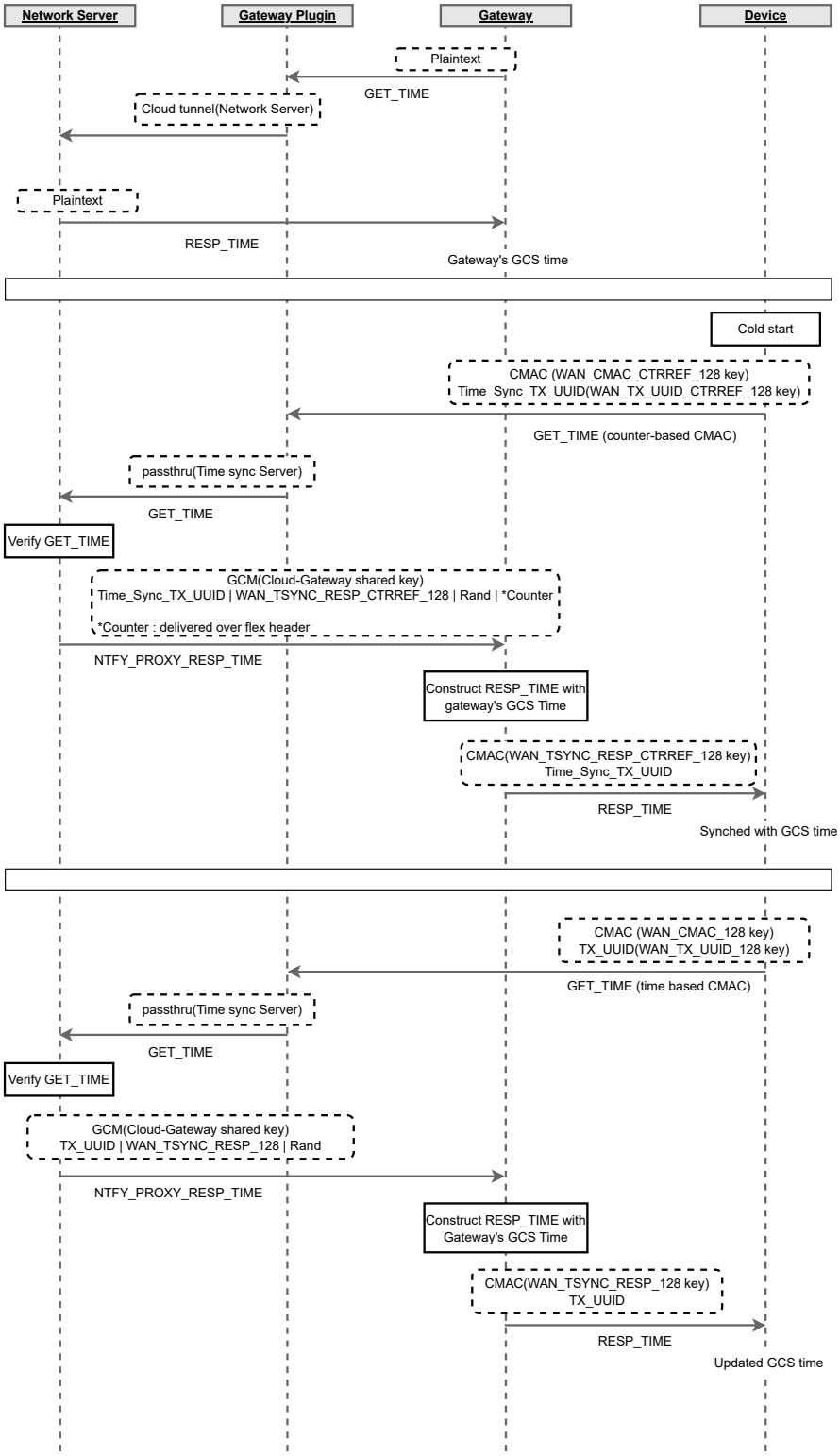


Fig. 4.4 GCS WAN message sequence

value. If the RESP_TIME is not received and *gcs_resp_timeout* reaches the maximum value then a GET_TIME command shall be sent at the maximum defined interval until the RESP_TIME command is received.

gcs_resp_timeout the first interval until gcs_ret_num is reached is 30 s, and then the timeout value is 3 min, then 5 min, then 8 min, then 10 min, then 15 min.

3. *gcs_sync_int* - the value indicates the frequency of sending GET_TIME command for the clock synchronization maintenance

   gcs_sync_int = 2 h

The clock shall be synchronized with the Amazon Sidewalk Cloud in the following cases:

1. The device is powered on or reset. Time synchronization shall be done as soon as possible. SubG-CSS, and SubG-FSK devices synchronize the clock at different points in time as follows:

   (a) A SubG-CSS device shall synchronize the clock after the protocol stack is on so that transmit and receive are possible

   (b) A SubG-FSK device shall synchronize the clock after the protocol stack is on, and after Beacon synchronization is achieved. NOTE: Clock synchronization could happen on other PHYs prior to the SubG-FSK Beacon synchronization.

   For flow charts of all procedures from the Amazon Sidewalk application viewpoint, see Chapter 8.

2. Maintenance of the correct clock value on a device side. The device shall synchronize the clock frequently at an interval of *gcs_sync_int*. Frequent clock synchronization is required because oscillator drift can cause the device's clock to float away from the Amazon Sidewalk network clock. The clock drift on the device side shall be less than +/- gcs_clock_drift seconds. When the time drift value received in RESP_TIME is greater than the time drift of the Endpoint then the Endpoint shall not update its clock value.

   gcs_clock_drift = 60 s

For values of *gcs_ret_num*, *gcs_resp_timeout*, *gcs_sync_int* and *gcs_clock_drift* see Appendix B.3.

4.3 Entity identification

Devices and services in Amazon Sidewalk have unique identification numbers. Endpoints are identified by a Device ID number, Cloud Services are identified by a Cloud Service ID.

4.3.1 Device ID

Device ID is a number that uniquely identifies a device in the Amazon Sidewalk network. Amazon Sidewalk has three types of Device ID:

1. 32 B SMSN
2. 5 B Sidewalk ID
3. 5 B TXID

4.3.2 Sidewalk Manufacturing Serial Number (SMSN)

For registration, the Amazon Sidewalk protocol shall use a subset(Size encoded 5B) of the Amazon Sidewalk Manufacturing Serial Number (SMSN) as the Device ID. The SMSN is constructed by calculating a hash over numbers of device-unique fields, using the following formula:

- SMSN=SHA256(DMS-DeviceType, **([DSN** | **unique serial number]**, APID))

where:

- SMSN - Unique identifier generated from values controlled by the customer. The unique identifier is the result of a SHA256 hashing calculation
- DMS-DeviceType - device type issued by DMS for the customer.
- DSN or Unique Serial Number - customer provided identifier that is unique to a particular device. If the customer has the concept of a Device Serial Number (DSN) this can be used, otherwise a UUID is sufficient.
- APID- Advertised Product ID - shortened product ID delivered by DMS present in the MFG page.

4.3.3 Amazon Sidewalk ID

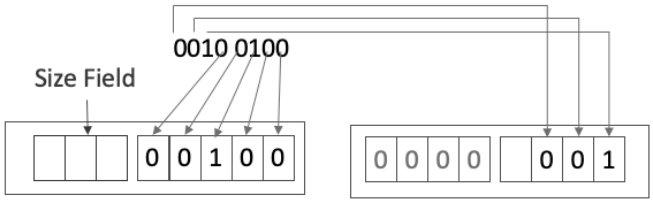
The Amazon Sidewalk ID is onboarded to the Endpoint during the registration process. After the registration process, the Amazon Sidewalk ID is not used for communication within the Amazon Sidewalk network. The Amazon Sidewalk ID is not used after registration to help protect user privacy. Instead of the Amazon Sidewalk ID, TXID is used in the Amazon Sidewalk network. The process of assigning a "random ID" (TXID) to a device is called obfuscation. For more information, see Section 4.6.

Device ID Type	Sub-Type	Usage	Periodicity
SMSN		Device Registration	Pre-registration
Sidewalk ID		Device Registration	During registration
TX_ID	Counter-based	1. Time sync on device when device has no notion of time 2. Deregistration from cloud Service when device has no notion of time	During time synchronization and factory reset
	Time Based	1. Regular Sidewalk data path communication 2. Periodic time sync for error correction in deviation when device has notion of time 3. Deregistration from cloud Service when device has notion of time	Every 15 min

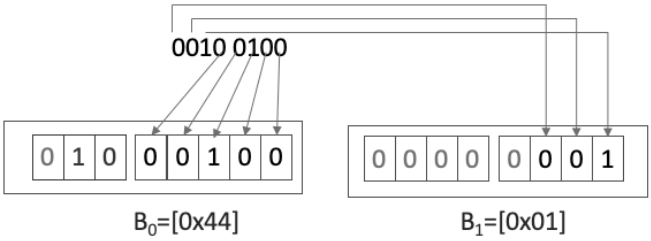
Table 4.5 Device ID types

For the Endpoint, the first 3 bits are defined as 0b101 and the remaining 37 bits allow more than 137 billion devices to be addressed. Device identity (Device ID and credentials) is built during the manufacturing provisioning process. During device provisioning, device control records containing the device serial number (SMSN) and public keys issued are stored in control logs, and uploaded to AWS. For more information, see Section 4.4.2.

- 36 = 0x24 = 0b 0010 0100
- Address Space (in 1st byte, 5 bits are available to store effective address, the remaining bits will go to the next byte):



- Size Encoded Address Space: In this example, the address occupied 0x2 = 0b10 bytes.



- Hence, the size encoded address array representing 36=0x24, looks like, {0x44, 0x01}

Fig. 4.5 Service ID

4.3.4 Service ID

Cloud services are identified by a unique Cloud Service ID number. Because Cloud Service ID can have a value from 0 to 2^{37} - 1, it is not required to send 5 B size-encoded Cloud Service ID address over the air. Figure 4.5 is an example of Cloud Service ID 36 (0x24) being converted to a size-encoded address.

There are four types of devices in the Amazon Sidewalk network (Endpoint, Gateway, Amazon Sidewalk Network Server and Application Server). Each device contains a unique ID (identity). IDs are secured by identity credentials built into the device during device manufacturing/releasing. Keys and certificates are issued based on the PKI security model using PKI certificates stored in the devices. Devices are connected by encrypting network and application data with separate keys. Links between termination points at each layer are secured by encryption with an ephemeral key exchange.

Endpoint 'K' (one of k Endpoints connected to a Gateway 'M') contains credentials related to its own ID, Link Layer related to temporary Endpoint's TXID, application layer and PKI infrastructure. Gateway 'M' (one of m Gateways connected to a Network Server) contains credentials related to its own ID, Network Server credentials associated with its own ID and PKI certificates. Network Server contains its own credentials, a list of credentials related to each of the Gateways connected to the Network Server at the Network Layer, and a list of credentials of each Endpoint (addressed by TXID and Device ID) at the Link Layer. Application Server contains its own credentials and also Endpoint credentials.

Figure 4.6 shows an overview of certificate and key locations within the system. Associated credentials are presented in dotted boxes.

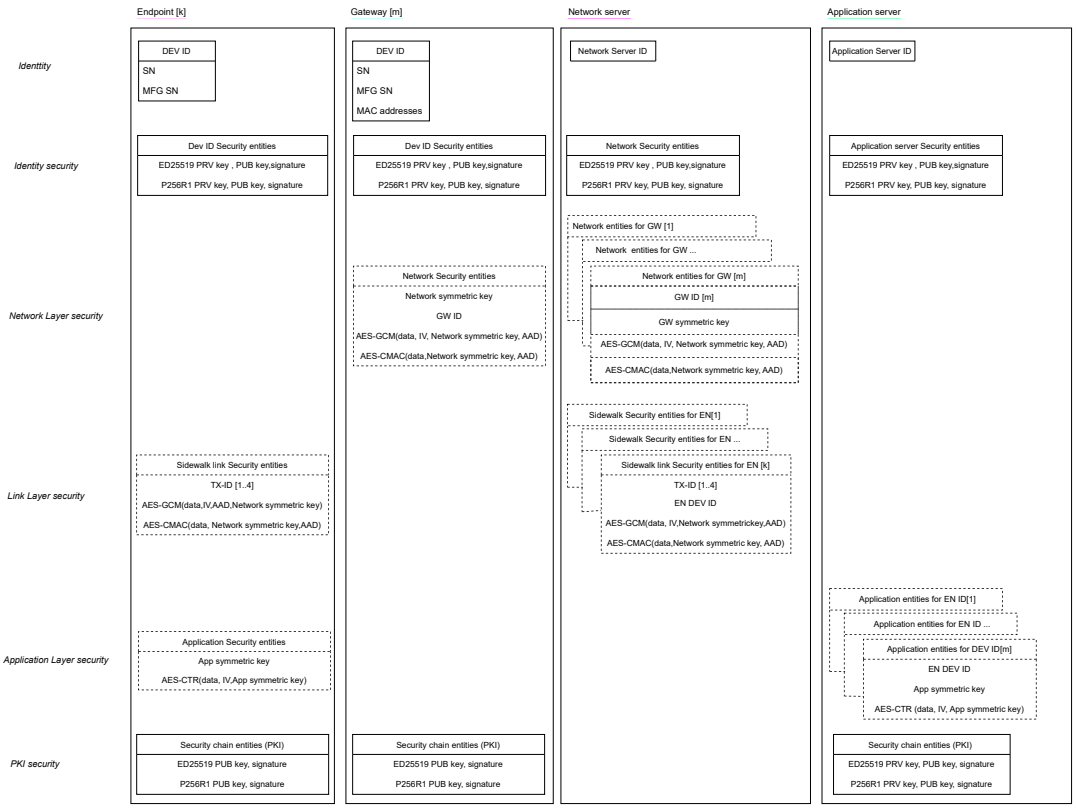


Fig. 4.6 Sidewalk network security

4.4 Key establishment

The key establishment process contains two stages: factory provisioning and registration. During the factory provisioning stage, hardware devices (Gateways and Endpoints) are provisioned with Amazon Sidewalk certificates and initial credentials. During the registration stage, hardware devices (Gateways and Endpoints) are associated and registered to a user.

4.4.1 Key assignment stages (initiating, storing discarding)

During obligatory device provisioning in the factory, initial credentials are stored on the device and the device is logged in manufacturing control logs. AWS consumes the control logs to allow devices to register with the network. The next mandatory stage for each device is owner provisioning. During owner provisioning the device is authenticated and its status is updated in AWS Cloud Services. During the device's lifetime its ownership can be changed, device software can be updated, and the device can be deregistered, or blocked. If any of the above changes are made to the device, the relevant credentials are updated or removed.

Separate security procedures are implemented for each of the following cases:

- Factory (Gateway, Endpoint) provisioning: During this stage all credentials are flashed on the device.
- Gateway owner provisioning: During this stage the device is authenticated in the network and network credentials are updated.
- Endpoint owner provisioning: During this stage device is authenticated in the network.
- Endpoint changing ownership (this is beyond the scope of this specification).

- Device deregistration, blocklist (this is beyond the scope of this specification).
- Application software upgrade. (This is beyond the scope of this specification.)
- Firmware upgrade (This is beyond the scope of this specification.)

4.4.1.1 Factory provisioning

A provisioning process in the mass production flow is triggered by the Contract Manufacturer that manufactures devices. This flow is driven by a number of factory tools and is designed for high volume manufacturing scenarios. This flow combines Factory Diagnostic Firmware, Amazon Sidewalk Signing Tool, Amazon Sidewalk Provisioning Tool, and Amazon Sidewalk Control Logs to issue Amazon Sidewalk certificates to devices and register them with Amazon Sidewalk Cloud Services. A device being provisioned is connected to the Contract Manufacturer's configuration system (hardware and software for data load, inventory management and unit calibration) and is also connected by an encrypted link to the Amazon Sidewalk Cloud. Device keys are signed by the Amazon Sidewalk signing tool using the HSM. Depending on the manufacturing needs, signing device keys can be done directly from the line PC, or by establishing a signing server on the local area network. The Amazon Sidewalk provisioning signing tool requests certificates after reading the Device Serial Number. Device certificates and signatures are generated based on received data EC key pairs (Ed25519 and p256r1). Device certificates and signatures are stored in device-persistent memory (flash). After certificates and signatures have been stored in the device, the Amazon Sidewalk Manufacturing Serial Number (SMSN), public keys and signatures are registered in the cloud by Amazon Sidewalk Cloud Services.

Each of the keys are signed by the Amazon Sidewalk Provisioning Tool during provisioning and flashed to the device by J-Link along with the rest of the certificate chain. Figure 4.7 shows the process flow.

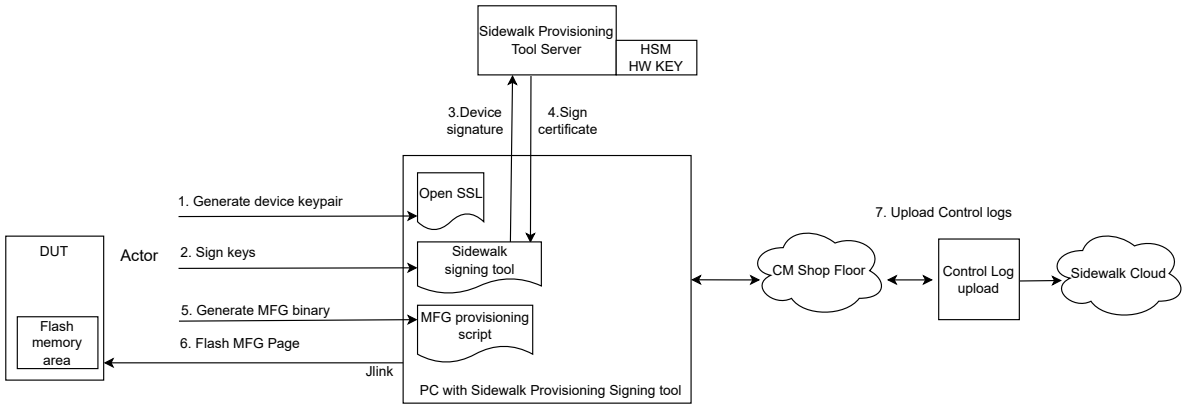


Fig. 4.7 Factory provisioning

The manufacturing page is a region in the internal memory that is reserved for various values to be written during the manufacturing of a device. Fields may be written to the manufacturing page either using a diagnostic version of the firmware or using the J-Link interface. Command line tools provided by the silicon vendor are used to write on the J-Link interface.

The following table summarizes the list of security items that need to be present on the device manufacturing page:

Required Field	Description
SID_MFG_STORE_VERSION	Version of the manufacturing storage block format.

Table 4.6 continued from previous page

Required Field	Description
SID_MFG_STORE_DEV_ID	Sidewalk ID. The Sidewalk ID includes a size field in the first 3 most significant bits, so the unique Device IDs should be no larger than (2^37 - 1) and the first 3 most significant bits should be hard coded to 0b101.
SID_MFG_STORE_HW_MODEL_AND_VERSION	Hardware model and version number. This is a silicon vendor-specific value that is used by the boot loader to determine if a firmware package that is about to be loaded is compatible with the device. Each model of hardware shall use a different value. If a new hardware revision of the same model needs to run a different version of firmware, the hardware version byte shall be increased. | 3B hardware model | 1B hardware version |
SID_MFG_STORE_BOARD_ID	Board ID
SID_MFG_STORE_SERIAL_NUM	Serial number of device.
SID_DEVCFG_DEVICE_PRIV_ED25519 SID_DEVCFG_DEVICE_PUB_ED25519 SID_DEVCFG_DEVICE_PUB_ED25519_SIGNATURE	Unique per device curve Ed25519 key pair, and signature (see: SID device authentication, SID key and signature generation)
SID_DEVCFG_DEVICE_PRIV_P256R1 SID_DEVCFG_DEVICE_PUB_P256R1 SID_DEVCFG_DEVICE_PUB_P256R1_SIGNATURE	Unique per device curve p256r1 key pair, and signature (see: SID device authentication, SID key and signature generation)
SID_DEVCFG_MODEL_PUB_ED25519 SID_DEVCFG_MODEL_PUB_ED25519_SIGNATURE SID_DEVCFG_MODEL_ED25519_SERIAL	Model device curve Ed25519 key pair, and signature (see: SID device authentication, SID key and signature generation)
SID_DEVCFG_MODEL_PUB_P256R1 SID_DEVCFG_MODEL_PUB_P256R1_SIGNATURE SID_DEVCFG_MODEL_P256R1_SERIAL	Model curve p256r1 key pair and signature (see: SID device authentication, SID key, and signature generation)
SID_DEVCFG_MAN_PUB_ED25519 SID_DEVCFG_MAN_PUB_ED25519_SIGNATURE SID_DEVCFG_MAN_ED25519_SERIAL	Product Manufacturer curve Ed25519 key pair and signature (see: SID device authentication, SID key and signature generation)
SID_DEVCFG_MAN_PUB_P256R1 SID_DEVCFG_MAN_PUB_P256R1_SIGNATURE SID_DEVCFG_MAN_P256R1_SERIAL	Product Manufacturer curve p256r1 key pair, and signature (see: SID device authentication, SID key and signature generation)
SID_AMZN_PUB_ED25519	Amazon Ed25519 public key
SID_AMZN_PUB_P256R1	Amazon p256r1 public key
SID_MFG_STORE_FW_KEY1 SID_MFG_STORE_FW_KEY1_Signature	Production Public Key #1 and signature used to check FW signatures. This key must be the same for every device that runs the same firmware. see: SID key and signature generation

Table 4.6 continued from previous page

Required Field	Description
SID_MFG_STORE_FW_KEY2 SID_MFG_STORE_FW_KEY2_SIGNATURE	Production Public Key #2 and signature used to check FW signatures. This key must be the same for every device that runs the same firmware. see: SID key and signature generation
SID_MFG_STORE_FW_DEV_KEY SID_MFG_STORE_FW_DEV_SIGNATURE	Development Public Key and signature used to check FW signatures. This key must be the same for every device that runs the same firmware. see: SID key and signature generation
SID_DHAv2_REGION_ADDRESS	DHAv2 region start address. The address should be at the beginning of the page
SID_DHAv2_REGION_SIZE	DHAv2 Region Size
SID_DHAv2_PID	Defined by Amazon and assigned to a product during product onboarding. SID_DHAv2_PID is unique to every product even for multiple products from the same manufacturer. This is alphanumeric and 4 characters in length. ASCII encoded. This location is used for devices that do not have the DHAv2 page extension.
SID_DHAV2_PIN	ASCII encoded secret PIN used for DHAv2 Registration. This location is used for for devices that do not have the DHAv2 page extension.
SID_MFG_STORE_TEST_MASTER_HW_ID	Lower 32 bits of Bluetooth MAC of master device that controlled the RF testing of this device.
SID_MFG_STORE_REG_ISO_COUNTRY	String containing two characters that determines the default country where the device is certified to operate. In ISO 3166-2
APP_MFG_STORE	Device-specific manufacturing value storage not specified by Sidewalk. (ie calibration values for sensors, etc.)
SID_MFG_STORE_CRC	Calculated over Amazon Sidewalk manufacturing storage. (words 0-1022)

Table 4.6 Security items in hardware device

4.4.2 Device registration

Devices are required to register before using Amazon Sidewalk. During registration the device is authenticated.

4.4.2.1 Gateway registration

As Amazon Sidewalk Gateways are based on Amazon and Ring product devices, an Amazon or Ring user account is required to register the product. To use an Amazon or Ring account requires configuring local Internet access. Amazon Sidewalk registration happens automatically once the user has signed in and registered the Gateway product. As part of Amazon Sidewalk registration, the Gateway establishes an encrypted connection with the Amazon Sidewalk Cloud, providing an authenticated connection.

4.4.2.2 Endpoint registration

The release described in this specification supports Endpoint registration via BLE and FSK can also be used.

An Endpoint can be registered using a mobile application leveraging the Amazon Sidewalk mobile SDK.

An Amazon registered user with a dedicated mobile application connects to Sidewalk Cloud Services and to the device while it is in Out Of the Box Experience (OOBE) mode and provides the device ID to register.

During this stage security is set up and a connection with the Client's API is established. The Endpoint connects to the Mobile Application with default settings and further configuration proceeds.

Then the authenticated link between AWS Cloud Services and the Endpoint is associated, the Endpoint is bound to the user Gateway and necessary credentials are established.

The following diagram summarizes the sequence flow:

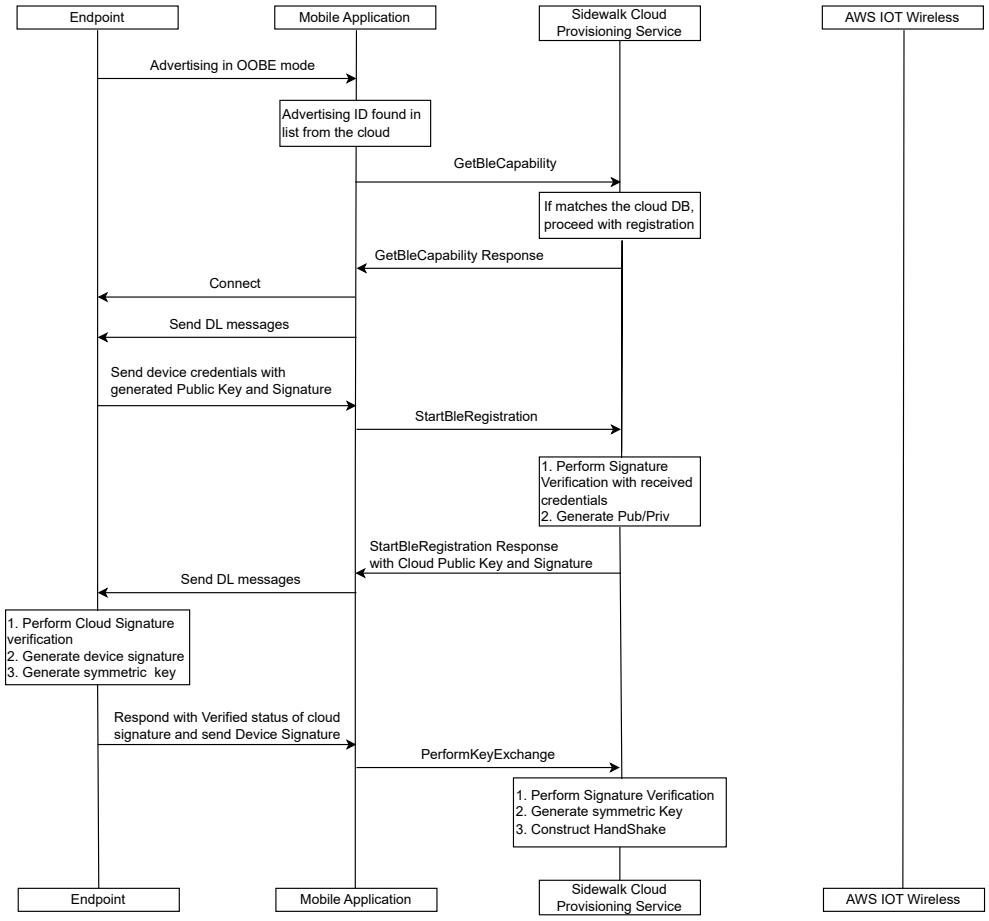


Fig. 4.8 EN registration part 1

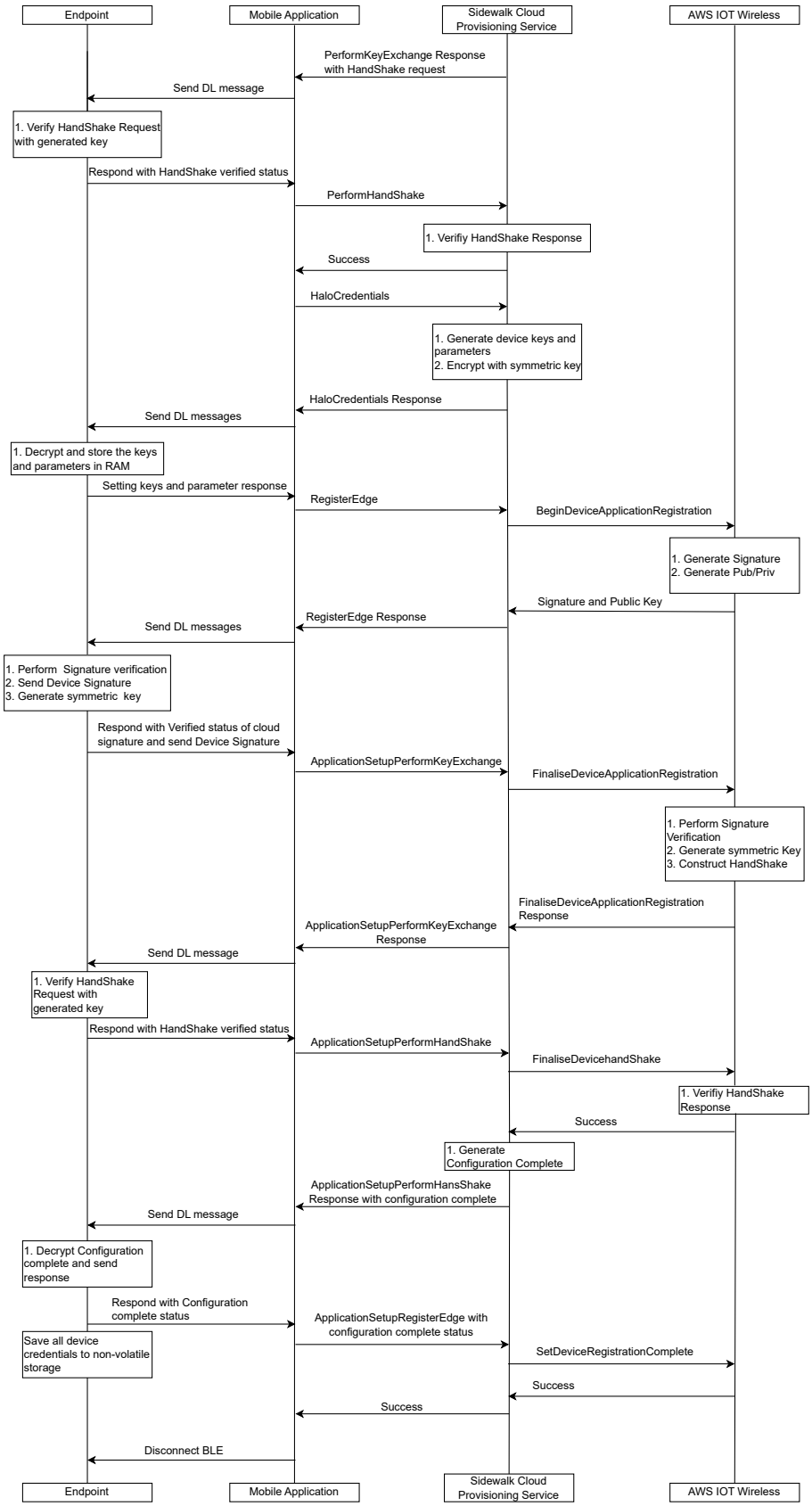


Fig. 4.9 EN registration part 2

4.5 Key management

Each of the links is protected by an ECDH key exchange process. Initially key pairs and certificates are built and stored in the device and in the relevant link termination point during the manufacturing provisioning process. For more information, see Section 4.4.1.1. For the application link, the Application Server initiates key exchange. For the Flex Layer link and the Network Layer link that terminate in the Network Server, the Network Server initiates key exchange.

4.5.1 Session Key exchange

The Network Server generates the ephemeral key pair and exchanges its public key with the Endpoint. The Endpoint generates the ephemeral key pair and exchanges its public key with the Network Server. the Network Server and the Endpoint generate the shared secret key using ECDH. The Network Server and the Endpoint use a predefined formula to generate the symmetric secret key from the shared secret. This symmetric secret key is the NETWORK_PRIMARY_KEY. The Network Server generates a temporary key from the NETWORK_PRIMARY_KEY. The Network Server uses the temporary key to encrypt a payload, then the payload is sent to the Endpoint for validation. The Endpoint decrypts the encrypted packet and verifies the content. If the Endpoint successfully decrypts and verifies the content, the Endpoint sends an uplink message to the Network Server with an encrypted packet indicating successful verification. If the Network Server receives the uplink packet, the Network Server decrypts the encrypted portion and validates it against the payload content sent on the downlink. If verification is successful, both the Network Server and the Endpoint save the NETWORK_PRIMARY_KEY to a database record.

4.5.2 Gateway Primary Key Negotiation

For SubG-FSK and SubG-CSS, the Gateway primary key is derived during Gateway registration. The Gateway primary key is used to encrypt the exchange between a Gateway and the Amazon Sidewalk Network Server. The Gateway primary key is used to encrypt the inner Flex Layer. The outer Flex Layer is not encrypted.

For BLE-enabled Gateways, a TLS session is used instead of the outer Flex Layer.

The following diagram describes the key exchange flow between a Gateway and the Amazon Sidewalk Network Server:

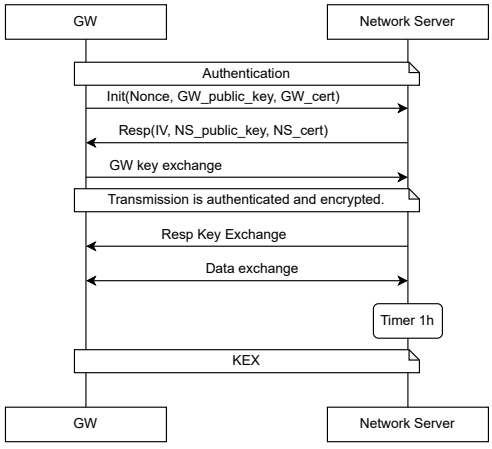


Fig. 4.10 Key exchange (KEX) flow

4.6 Obfuscated TX(Transmission) ID

Due to pseudo-anonymity requirements, the Device ID may not be exposed for unauthorized tracking in the radio protocol. An obfuscation process is implemented to avoid exposing the Device ID.

4.6.1 Device ID obfuscation

ID obfuscation is a method for increasing device anonymity/privacy by generating a temporary ID to help prevent brute force attacks.

The Network Server key generated during registration is used to create a Device ID obfuscation key. The Device ID obfuscation key is used to generate TXID using the equation below. The TXID rotation window is negotiated during registration. The default value for the TXID rotation window is 900 s.

- TXID=HMAC(DeviceIDobfuscationKey, sha256(Time_Reference, Device ID))[0:4]

where

- DeviceIDobfuscationKey: TXID_CTRREF for counter-based procedure and TXID for GCS time-based procedure
- Time_Reference: counter or GCS time. Time size: 4 B are allocated.
- [0 ..4] - TXID is five LSB from AES-HMAC algorithm result.

The mapping from the TXID to the customer ID is maintained by the Application Server.

Due to possible differences between Endpoint local time and network server time (time drift), three TXIDs are generated with Time_Reference, Time_Reference+900 s and Time_Reference - 900 s which are called the Current, Next and Previous TXIDs. Three TXIDs are generated to validate the obfuscated Device ID received in the message. Packets with an obsolete TXID are discarded by the receiver.

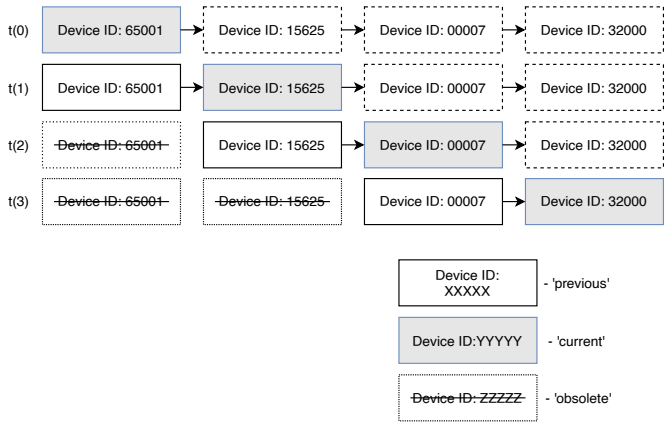


Fig. 4.11 ID rotation

TXIDs are calculated in both the Network Server and the Endpoint.

4.7 Encrypted messaging

Encrypted messaging protects communication between an Endpoint and the Amazon Sidewalk Cloud. Encrypted messaging has two layers of security: Network Layer security and Application Layer security. Each encrypted layer is used to encrypt different parts of the Message frame. The Network Encryption Layer encrypts the Network Layer frame payload. The Application Layer security may encrypt the "Application command" part of the Presentation Layer frame. Link Layer frame parts such as "Frame control", "Source address", "Destination address", "Sequence number" are always sent as plaintext (see Section 5.1.1, and Section 5.1.1.1). Network Layer frame parts such as "Frame control" and "Authentication tag" are always sent as plaintext (seeSection 5.1.2, Section 5.1.2.1, and Section 5.1.2.2) Figure 4.12 shows which security layers encrypt the different parts of the Link Layer and Network Layer frames. Information on whether Network Layer encryption is enabled can be found in the Network Layer frame in the *encr_ena* field of the Frame control part (see section 5.1.2.2.1). Information on whether Application Layer encryption is enabled can be found in the Presentation Layer frame in the *app_encr_ena* field of the Application Support Layer header part (see Section 5.1.3). Network-only Messages do not use Application Layer encryption.

Beacon frame (see Section 5.1.3.1) is always sent as plaintext. The Authentication tag that is included in the Beacon frame format is not used (see section 5.1.3.1.9).

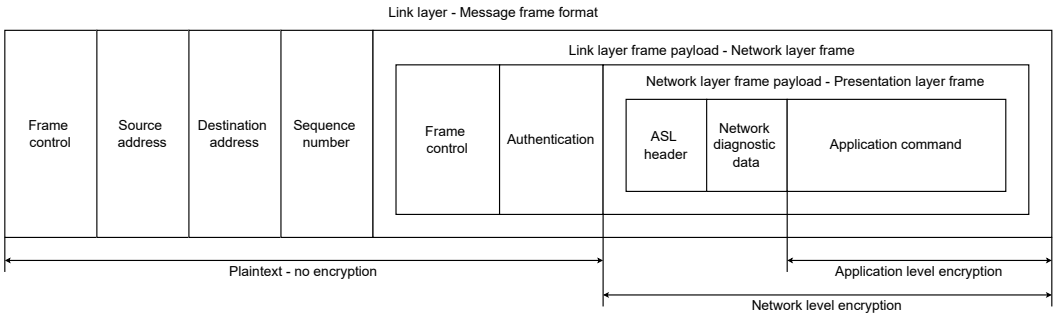


Fig. 4.12 Message frame format encryption

Network Layer encryption and Application Layer encryption use different algorithms for encryption, decryption and MAC generation. Network Layer security shall use the AES-GCM algorithm for encryption, decryption, MAC generation and verification. Application Layer security shall only use the AES-CTR algorithm for encryption and decryption. Application Layer security shall not use the MAC for verifying the sender. During time synchronization, MAC generation shall use the AES-CMAC algorithm. Time sync messages shall be authenticated.

Encrypted messaging requires the Endpoint clock to be synchronized with the Amazon Sidewalk Cloud before end-to-end communication is established. Algorithms AES-GCM and AES-CTR use symmetric keys that are generated based on the global clock for all Amazon Sidewalk application commands. When the Endpoint clock is out of synchronization, AES-CMAC uses symmetric keys that are generated based on the counter for time synchronization commands (GET_TIME, RESP_TIME). When the Endpoint clock is synchronized, AES-CMAC also use symmetric keys that are generated based on the global clock for time synchronization commands. The end-to-end message is not encrypted and MAC verification is applied only for time synchronization commands. For more information on clock synchronization feature, see Section 4.2. For more information on keys used for encryption, decryption and MAC generation, see Section 4.7.1

The AES-GCM algorithm used for encryption and MAC generation requires the following input parameters - see Figure 4.13):

The output of the algorithm is:

1. N bytes of the ciphertext

Inputs	Size in bytes	Description
Security key	16	symmetric key
IV	12	Initialization Vector used in Network Layer security
AAD	8	Additional Authenticated Data used for encryption in Network Layer security
Byte array	N	data to be encrypted

Table 4.7 Input data for AES-GCM packet encryption

2. MAC size is 4 B for sub-GHz encryption

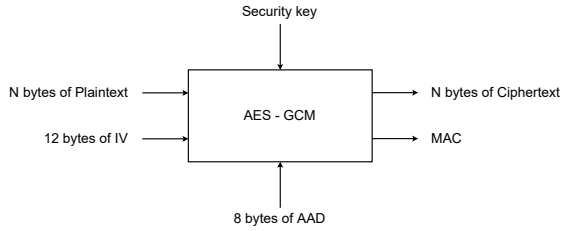


Fig. 4.13 Network layer security - packet encryption

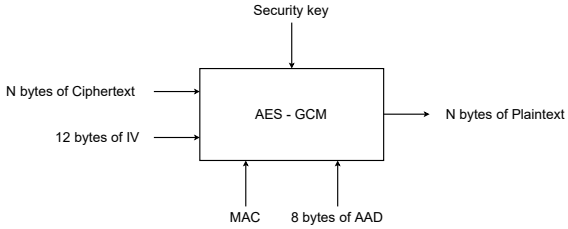


Fig. 4.14 Network layer security - packet decryption

The AES-GCM algorithm used for decryption and MAC verification requires the following input parameters (see Figure 4.14):

Inputs	Size in bytes	Description
Security key	16	symmetric key
IV	12	Initialization Vector used in the Network Layer security
AAD	8	Additional Authentication Data used for encryption in the Network Layer security
Byte array	N	N bytes of the received ciphertext
MAC	4	received MAC for decryption in sub-GHz

Table 4.8 Input data for AES-GCM packet decryption

The output of the algorithm is N bytes of plaintext.

The AES-CTR algorithm used for encryption requires the following input parameters (Figure 4.15):

Inputs	Size in bytes	Description
Security key	16	symmetric key
IV	16	Initialization Vector used in the Application Layer security
Byte array	N	bytes of the string to be encrypted

Table 4.9 Input data for AES-CTR packet encryption

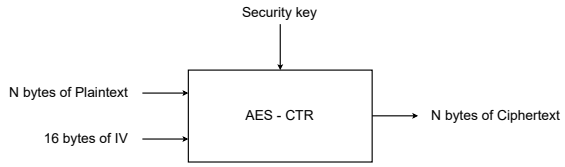


Fig. 4.15 Application layer security - packet encryption

The output of the algorithm is N bytes of the ciphertext.

The AES-CTR algorithm used for decryption requires the following input parameters (see Figure 4.16):

Inputs	Size in bytes	Description
Security key	16	symmetric key
IV	16	Initialization Vector used in the Application layer security
Byte array	N	received ciphertext

Table 4.10 Input data for AES-CTR packet decryption

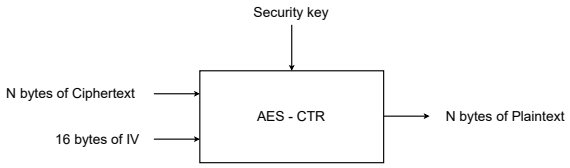


Fig. 4.16 Application layer security - packet decryption

The packet decryption algorithm outputs bytes of plaintext.

The AES-CMAC algorithm used for MAC generation requires the following input parameters (see Figure 4.17):

Inputs	Size in bytes	Description
Security key	16	symmetric key
AAD	16	Additional Authentication Data used for MAC generation
Byte array	N	message to be authenticated
MAC	4	

Table 4.11 Input data for AES-CMAC

The output of the algorithm is the MAC. The MAC size for SubG-CSS devices is 4 B. The MAC size for SubG-FSK devices is 12 B.

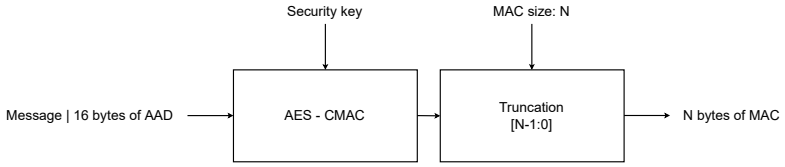


Fig. 4.17 MAC generation

NOTE: The message to be authenticated is the Network Layer frame payload. The input to the AES-CMAC algorithm that contains "Message |16 B of AAD" is the concatenation of the Message and Additional Authenticated Data (AAD).

For more information on the Symmetric keys that are used in the AES-GCM, AES-CMAC and AES-CTR algorithms, see Section 4.7.1.

For more information on Initialization Vector generation and format, see Section 4.7.2.

For more information on Additional Authentication Data generation and formats, see Section 4.7.3.

4.7.1 Symmetric keys used in encrypted messaging

Symmetric keys are used for encrypted messaging by Network Layer encryption, and Application Layer encryption. For more information on Symmetric key generation and rotation rules, see:Section 4.4.

Keys used by the Network Layer security for encrypted messaging are shown in Table 4.12.

The following keys are used in encrypted messaging on the Application security level:

NOTE: Application Layer security is not applicable when the Endpoint clock is not synchronized.

4.7.2 Initialization Vector

The Initialization Vector (IV) has different formats for the Network Layer security and Application Layer security. Network Layer security uses 12 B IV format that is presented in Figure 4.18. Application Layer security uses a 16 B IV format, as shown in Figure 4.19. The IV is never reused with the same encryption key as the monotonically increasing sequence number has 14 bits of entropy within a 1-min time reference window. In the sub-GHz link, the 14 bits of the sequence number will never be repeated within 1 min.

Key	Description
GCM_KEY	used as AES-GCM input (encryption, decryption, MAC generation and verification) when the Emdpoint clock is synchronized. It may be used for all commands except time synchronization commands
CMAC_CTRREF_KEY	used as AES-CMAC input (MAC generation) for GET_TIME command when the Endpoint clock is not synchronized
TSYNC_RESP_KEY	used as AES-CMAC input (MAC generation) for RESP_TIME command when the Endpoint clock is synchronized
TSYNC_RESP_CTRREF_KEY	used as AES-CMAC input (MAC generation) for RESP_TIME command when the Endpoint clock is not synchronized
JOIN_RESP_GCM_KEY	used as AES-GCM input (encryption, decryption, MAC generation and verification) for RESP_JOIN_REQ command

Table 4.12 Symmetric keys used in encrypted messaging - Network Layer

Key	Description
APP_CTR_KEY	used as AES-CTR input (encryption and decryption) when the Endpoint clock is synchronized. It may be used for all commands except time synchronization commands

Table 4.13 Symmetric keys used in encrypted messaging - Application Security

4.7.2.1 Network encryption layer IV

The 12 B IV format is a concatenation of the following values:

1. *Source ID* is the Amazon Sidewalk ID of the Endpoint and the Cloud Service ID of the sender
2. *Sequence number* is in the Link Layer header which is a monotonically incremented number (see Section 5.1.3)

   NOTE: Link Layer packet sequence number size may be 1, 2 or 3 B. The IV generation shall use a 3 B format. When sequence number size is less than 3 B in the Link Layer then 0x00 value shall be added as MSB byte/s to compose a sequence number for IV generation e.g. when the sequence number is encoded in 2 B "12 34" then "00 12 34" shall be used for IV generation.

3. *Reference time* expressed in seconds is calculated based on the following formula:
   $Time = ROUNDDOWN(T\_current / T\_IV\_ref) * T\_IV\_ref$
   where

$T\_current$ - current network time
T_IV_ref - validity of reference time in IV (see Appendix B.3 for the exact value of T_IV_ref)
$T\_IV\_ref = 60$ s

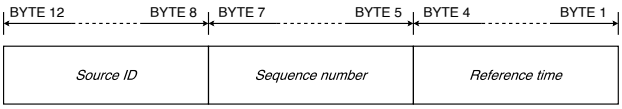


Fig. 4.18 IV format for the Network security layer

4.7.2.2 Application Layer security IV

The 16 B IV format is a concatenation of the following values:

1. *Source ID* is the Amazon Sidewalk ID of the Endpoint and the Cloud Service ID of the sender
2. *UL/DL* indicates the message direction. Bit set to "0" indicates the downlink, bit set to "1" indicates the uplink
3. *Application layer sequence number* is the same as in *sequence number* field present in the ASL header / Presentation Layer

   NOTE: Application sequence number size may have 1 or 2B. IV generation shall use a 2 B format. When the application sequence number size is less than 2 B then 0x00 value is added as MSB byte to compose the sequence number for IV generation e.g. Sequence number is 0x2, the sequence number portion of IV is 0x000002 for the downlink and 0x800002 for the uplink.

4. *Reference time* is the same as the Network Layer encryption reference time
5. 4 B, the LSB is reserved and the default value is 0x00000000

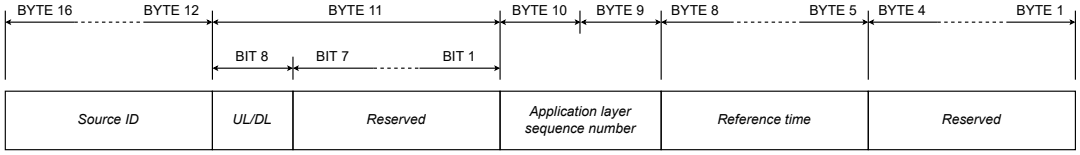


Fig. 4.19 IV format for the Application security layer

The generated IVs are rotated in intervals equal to T_IV_ref.

4.7.3 AEAD - Authenticated Encryption with Associated Data

The Authenticated Encryption with Associated Data (AEAD) has a different format of Additional Authentication Data (AAD) for Network Layer security and MAC generation. Network Layer security uses an 8 B format that is shown in Figure 4.20. MAC generation uses a 16 B format that is shown in Figure 4.21.

AEAD is the concatenation of the following parameters:

1. *Sequence number* is the Link Layer packet sequence number (see Section 5.1.3.2).

   The Sequence number is only used in packet authentication AEAD generation.

NOTE: The Link Layer packet sequence number size may be 1, 2 or 3B.

The IV generation shall use a 3 B format.

When the Link Layer sequence number is less than 3 B then 0x00 value shall be added as MSB byte/s to compose a sequence number for IV generation e.g. when the sequence number is encoded as 2 B "12 34" then "00 12 34" shall be used.

2. *Source ID* is sender entity ID. Source ID depends on the message source. Source ID can be the Device ID or the Cloud Service ID - see Section 4.3.

3. *Destination ID* is the destination identity used in the Link Layer frame format. The Destination ID can be the Device ID or the Cloud Service ID.

NOTE: Source ID and Destination ID size may be in the range 1 to 5 B. AEAD generation shall use a 5 B format. When Source ID or Destination ID size is less than 5 B in the Link Layer then the value 0x00 shall be added as LSB byte/s to compose an ID for AEAD generation e.g. when Destination ID is encoded in 2 B "12 34" then "12 34 00 00 00" shall be used.

4. *Security reference* is included in the Network Layer frame format (*sec_ref* field) when *ext_hdr* field is set to 1. When *ext_hdr* field is set to 0 and *sec_ref* is not included then the value shall be set to 0 by default.

5. *Security topology* is included in the "Frame control" block of the Network Layer frame format (*sec_top* field)

NOTE: The current release of the protocol supports only one security topology. This bit shall be set to 1 by default.

6. *Authentication code size* is included in the "Frame control" block of the Network Layer frame format (*auth_code_size* field).

7. *Encryption enabled* is included in the "Frame control" block of the Network Layer frame format (*encr_ena* field).

8. *Destination format* and *Source format* are included in the "Frame control" block of the Link Layer frame format (*src_frmt* and *dst_frmt* fields).

9. *RF frame protocol version* is included in the "Frame control" block of the Link Layer frame format (*prot_ver* field).

NOTE: The Link Layer contains a 4 bit prot_ver field that contains the protocol version, but AEAD requires a 1 B value. The value taken from the Link Layer shall have 4 bits with 0 value prepended to prot_ver field to fit the AEAD protocol version size.

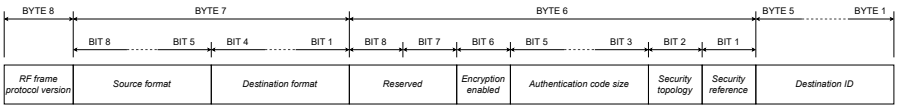


Fig. 4.20 8B AAD for the Network security layer

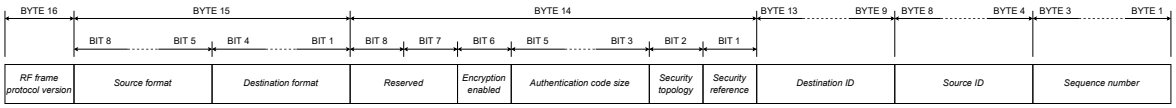


Fig. 4.21 16B AAD for MAC generation

4.8 Flex Layer messaging

4.8.1 Flex Layers

Flex Layer headers use a Type, Length Value (TLV) format. The Flex Layer header format is used for Network Layer header content and Link Layer header content in the Flex headers for Gateway to cloud communication and for cloud to Gateway communication. Flex headers are also used to achieve different security layers for communications. Single and Double flex headers are supported in Amazon Sidewalk communication.

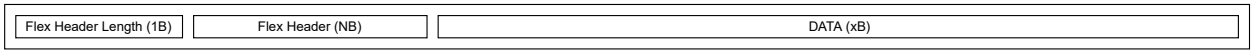


Fig. 4.22 Single Flex Layer

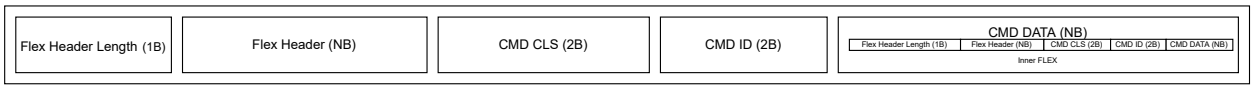


Fig. 4.23 Double Flex Layer

4.8.2 Flex Layers message flow

It is assumed that the Gateway is registered in the Amazon Sidewalk network. See Section 4.4.2.1. It is assumed that the Endpoint has successfully finished the Join procedure, and the Endpoint time is synchronized.

A packet received from the Endpoint (with TXID) is encrypted with the Gateway's GCM_KEY and encapsulated in an inner flex packet (inner flex headers are added).

The whole inner flex packet is encapsulated in an outer flex frame (with header and security related metadata) and sent over a TCP/IP secured link towards the Network Server in the Sidewalk Cloud. The Network server, recovers the Device ID from the TXID, decrypts the flex data using the GCM_KEY key and routes the encrypted message to the Application server. The Application server decrypts the payload data and redirects the payload to the relevant Third party Application ID.

Downlink sequence: A message with the destination Device ID is encrypted (using APP_CTR_KEY) and is sent to Network server. A TXID is generated from the Device ID, based on an internal map for device obfuscation. The Inner Flex headers are added and the payload is encrypted with the GCM_KEY. The message is encapsulated in an outer Flex message (headers and and security related data are added) and the outer Flex message is sent over secured TCP/IP network to the Gateway. The Gateway decrypts the Flex payload using the GCM_KEY and places the packet with the TXID in a queue (the App message in the Flex payload is still encrypted with the APP_CTR_KEY). The packet is scheduled and sent over the air

with preamble and PHY headers. If the Endpoint successfully receives the packet, the Endpoint decrypts the packet using the APP_CTR_KEY (and metadata). For flow diagrams see Figure 4.24 and Figure 4.25.

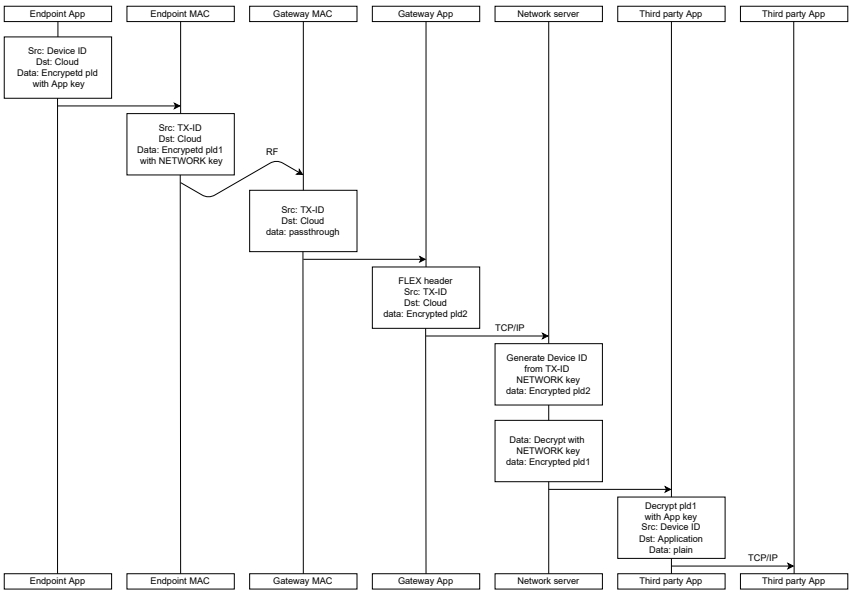


Fig. 4.24 EN to Cloud

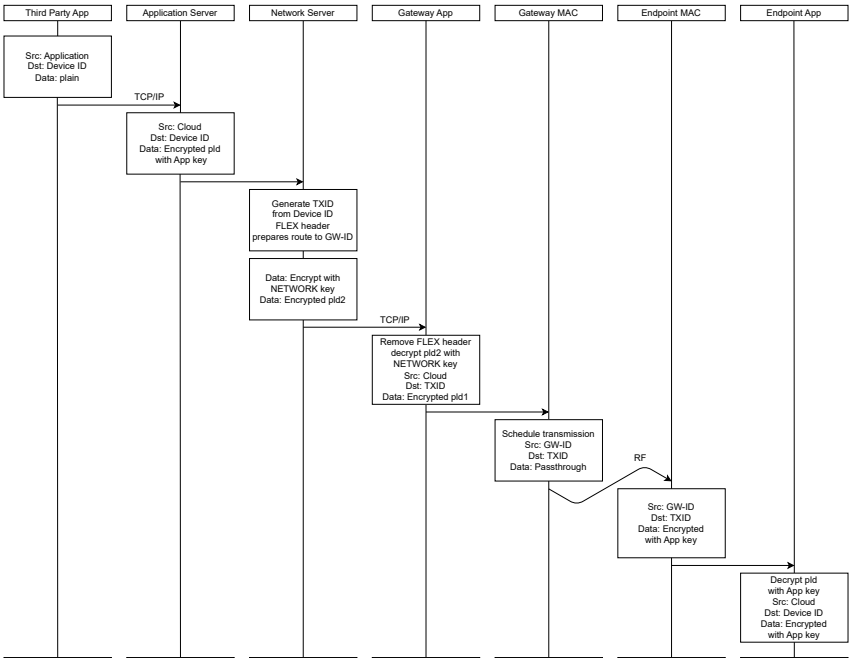


Fig. 4.25 Cloud to EN

Chapter 5

Sub-GHz Protocol Stack

5.1 Protocol Stack Layers and Structure

The protocol stack is illustrated in Figure 5.1.

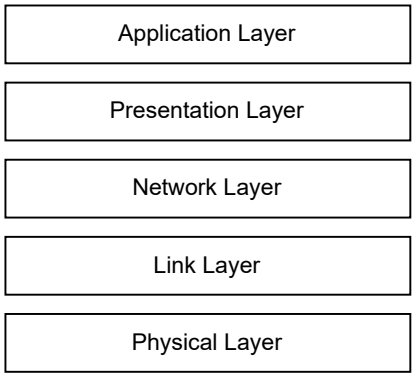


Fig. 5.1 Protocol stack

A detailed description of each layer's functions, supported features, and frame format can be found in the following chapter and subsections:

1. The Application Layer is described in Chapter 3 Application. This chapter includes application commands format definition, a detailed description of available commands, and flow charts showing the base scenarios with command usage.

2. The Presentation Layer is described in Section 5.1.1. This subsection provides information on how application commands are included in the layer. This subsection also defines the format of diagnostic data that may be included in a command or response.

3. The Network Layer is described in Section 5.1.2. This subsection describes the network security settings of the message.

4. The Link Layer is described in Section 5.1.3. This subsection defines the Beacon frame format and message frame format and their functions.

5. The Physical Layer is described in Section 5.1.4. This subsection defines the RF frame format, configuration and airtime values. This subsection also includes considerations of the time drift aspects.

5.1.1 Presentation Layer

The Presentation Layer supports exchange of Application Layer commands between two entities. The Presentation Layer also supports Application Layer security including encryption and frame size randomization. Presentation Layer frames may also contain diagnostic data that are typically sent by an Endpoint in the uplink direction.

5.1.1.1 Presentation Layer Frame format

The frame is divided into three main blocks as follows:

- The first block is the "Application Support Layer header". This block provides information for decoding and interpreting the frame.
- The second block is "Network diagnostic data". This block contains two main parts. The first part contains diagnostic data. The second part contains padding bytes.
- The third block is "Application command". This block contains the application command and its payload.

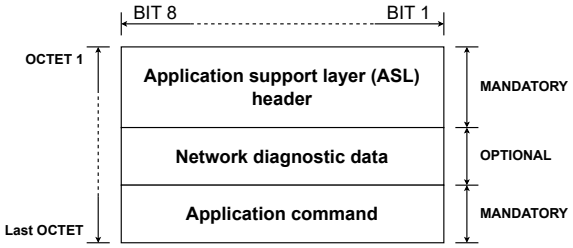


Fig. 5.2 Presentation layer - Blocks

5.1.1.1.1 Application Support Layer (ASL) header

The Application Support Layer header includes flags indicating the presence or absence of fields in the rest of the frame, the encryption status of the rest of the Network Diagnostic and Application Command blocks, and the Application Support Layer header version number. The Application Support Layer header version has an impact on whether a received message is processed or rejected. The last value in the Application Support Layer header is related to the uplink or downlink message number that allows control of the correct sequence of data transmit and receive.

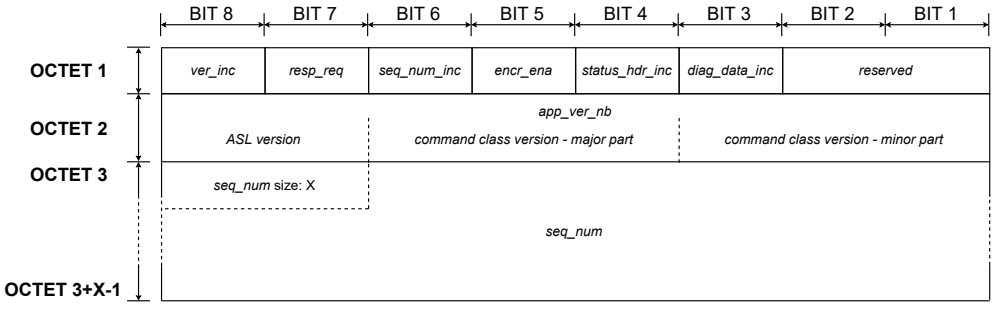


Fig. 5.3 Presentation layer - ASL header

1. Application version number flag (version_inc)

Mnemonic	Presence M/O/C	Length	Value	
ver_inc	M	1b	Bit value	Description
			0	version number field not included
			1	version number field included

The value of the Application version number flag indicates whether the "Application frame version number" field is included in the frame.

(a) The Application version number is optional for messages exchanged with the Amazon Sidewalk Cloud.

(b) The Application version number shall be included in all destination addresses that are not in the cloud.

2. Response request flag (resp_req)

Mnemonic	Presence M/O/C	Length	Value	
resp_req	M	1b	Bit value	Description
			0	response not requested
			1	response requested

The value of the response request flag indicates whether the sender requests a response from the destination.

3. Sequence number flag (seqn_inc)

Mnemonic	Presence M/O/C	Length	Value	
seq_num_inc	M	1b	Bit value	Description
			0	sequence number field not included
			1	sequence number included

The value of the Sequence number flag indicates whether the "Sequence number" field is included in the frame.

4. Application level encryption flag (app_enc_ema)

Mnemonic	Presence M/O/C	Length	Value	
encr_ena	M	1b	Bit value	Description
			0	encryption disabled
			1	encryption enabled

The value of the Application level encryption flag indicates whether the "Application command" block is encrypted with application level encryption.

5. Status header flag (status_hdr_inc)

Mnemonic	**Presence M/O/C**	**Length**	**Value**	
status_hdr_inc	M	1b	Bit value	Description
			0	status header field not included
			1	status header field included

The value of the Status header flag indicates whether the "Status code" and/or "Additional application data" fields are included in the frame.

6. Network diagnostic data flag (nw_data_blob_inc)

Mnemonic	**Presence M/O/C**	**Length**	**Value**	
diag_data_inc	M	1b	Bit value	Description
			0	diagnostic data not included
			1	diagnostic data included

The value of the Network diagnostic data flag indicates whether the "Network diagnostic data" block is included in the frame.

7. Application frame version number (version)

Mnemonic	**Presence M/O/C**	**Length**	**Value**	
app_ver_nb	C *version_inc* is set to 1	1B	Bit number	Description
			8:7	ASL version
			6:4	command class version - major part
			3:1	command class version - minor part

The value of the Application frame version number indicates the version of the Application Support Layer. This identifies the revision of the command class that is in use.

The meaning of selected parts of the field is as follows:

- ASL version: specifies the ASL header format. One format is specified. The value indicating the specified ASL header is 0x0.
- Command class version - major part: indicates significant changes introduced in the command class. All commands from a command class that have different major versions are not backward compatible. Devices that can only support major versions that are lower than the received command class version shall reject the incoming message.
- Command class version - minor part: indicates small changes introduced to the command class. Devices supporting only lower minor versions shall accept an incoming message and process only the parts of the command that can be understood. Fields that cannot be understood by the device shall be ignored.

Mnemonic	Presence M/O/C	Length	Value
seq_num	C *seqn_inc* is set to 1	1 to 2 B	counter value

8. Sequence number (seqn)

   The value of the Sequence number indicates the sequence number of the message. The field contains two parts:

   (a) the 2 most significant bits of the first byte specify the size of the field in bytes

   (b) the remaining bits are the sequence number:

      i. 1 B size: remaining 6 bits allow a sequence number range between 0 and 63

      ii. 2 B size: remaining 14 bits allow a sequence number range between 0 and 16383

5.1.1.1.2 Network diagnostic data (nw_data_blob)

The Network diagnostic data field contains information related to signal conditions and message timing. Network diagnostic data are grouped in several formats by Format ID. Each Format ID indicates the type of diagnostic data that are included in the frame. The Format ID also informs whether padding bytes may be included in this part of the frame. The number of padding bytes is random. Adding padding bytes randomizes the packet size. Randomizing the packet size prevents an attacker from using statistical analysis to deduce the message type.

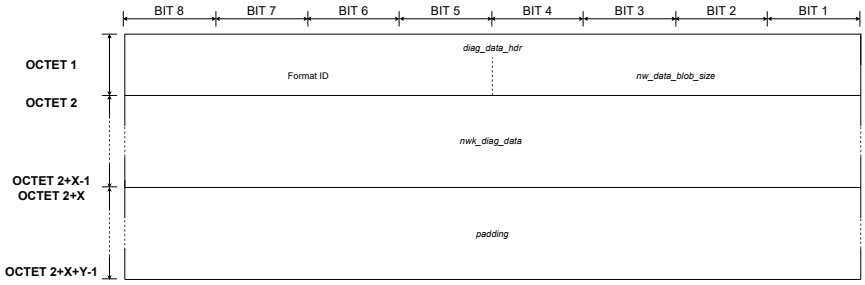


Fig. 5.4 Presentation layer - Network diagnostic data

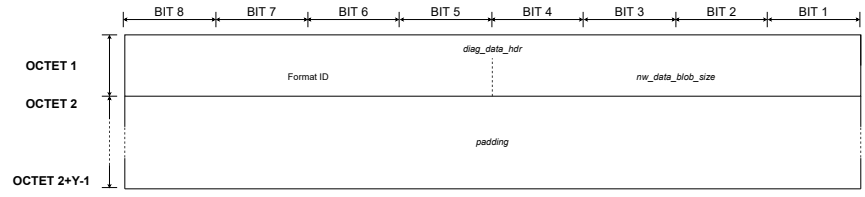


Fig. 5.5 Presentation layer - Network diagnostic data - Format ID 0

1. Network diagnostic data header (diag_data_hdr)

Mnemonic	**Presence M/O/C**	**Length**	**Value**	
diag_data_hdr	C *diag_data_inc* is set to 1	1B	Bit number	Description
			8:5	Format ID
			4:1	*nw_data_blob_size*

The Network diagnostic data header contains the Format ID of the network diagnostic data. The Network diagnostic data header also contains the *nw_data_blob_size*. The *nw_data_blob_size* indicates the total size of the *nwk_diag_data* and *padding* fields.

2. Diagnostic data (nwk_diag_data)

Mnemonic	**Presence M/O/C**	**Length**	**Value**
nwk_diag_data	C *diag_data_inc* is set to 1 Format ID > 0	see NOTE	diagnostic data

The Diagnostic Data field contains additional diagnostic data. The data length and presence are Format ID-dependent.

The following table shows supported Format IDs.

Format ID number	**Diagnostic data presence**	**Format ID size** *format_id_size*
0	not present	0
3	present	length of total TLV

NOTE: Based on the Format ID, the length will be different. The length of Format ID 0 is 0 as it does not include any "Diagnostic data".

(a) Format ID 0 No diagnostic data.

(b) Format ID 3 Diagnostic data of this format is encoded in the format of TLV.

TLV key name	**TLV key ID**	**TLV length**	**Description**
Synchronization request	0x01	1B	TLV for network synchronization in 5.4. Value of TLV is not used.

3. Padding (padding)

Mnemonic	**Presence M/O/C**	**Length**	**Value**
padding	C *diag_data_inc* is set to 1 (see NOTE 1)	*padding_size*	padding bytes

The *padding* field contains additional padding bytes to randomize the packet size. Randomizing the packet size improves packet security against size tracking.

NOTE 1: The presence of the field is based on Format ID. The padding field presence is mandatory for Format ID 0.

The length of the *padding* field is *padding_size*. The receiver calculates the number of padding bytes based on the Format ID of the network diagnostic data and *nw_data_blob_size*. Both values are provided in *diag_data_hdr* in the received message. The receiver shall use the following formula based on the Format ID:

(a) Format ID 0
All bytes in the "Network diagnostic data" block shall be padding bytes. The number of padding bytes is calculated using the formula:
padding_size = *nw_data_blob_size* * 2

The sender calculation of *padding_size* is more complex and is constrained as follows:

(a) The total number of bytes included in the "Network diagnostic data" block (*format_id_size*) and Presentation Layer (*frame_pld_size*) shall not exceed *max_packet_size*. *max_packet_size* has different values for synchronous and asynchronous link modes.

Connection mode	***maximum application payload size***
SubG-CSS	19
SubG-FSK	200
BLE	255

The first step in calculating the padding field size is to check that *packet_size* does not exceed *max_packet_size*:

packet_size = *format_id_size* + *frame_pld_size*

If padding bytes may be added, then the initial random length of padding field is calculated:

rand_pad = rand()%(*max_packet_size* - *packet_size*) + *packet_size*

next the number of padding bytes is calculated as a multiple of *padding_block_size*:

pad_mod = *rand_pad* MOD *padding_block_size*

If *pad_mod* is not a multiple of *padding_block_size* then *pad_mod* is recalculated:

IF *pad_mod* != 0 THEN

pad_mod = *padding_block_size* - *pad_mod*

Next, the number of allowed bytes in the frame is taken into account using the following formula:
total_size = MIN(*rand_pad* + *pad_mod*, *max_packet_size*)

Then, the initial value of the padding bytes length is calculated:
padding_size = *total_size* - *packet_size*

The length of padding field is limited by *max_padding_size*:
padding_size = MIN(*padding_size*, *max_padding_size*)

Finally, the sum of *padding_size* and *format_id_size* is divided by 2:
IF (*padding_size* + *format_id_size*) MOD 2 != 0 THEN
padding_size = *padding_size* - 1

The *nw_data_blob_size* included in the frame shall be calculated from the following formula:
nw_data_blob_size = (*padding_size* + *format_id_size*)/2

5.1.1.1.3 Application command

The format of the "Application command" part of the frame is determined by the presence of the "Status code", "Additional application data" and "Command" fields as follows:

1. Case 1A: "Status code", "Additional application data" and "Command" fields are included in the frame (see Figure 5.6)
2. Case 1B: "Status code" and "Additional application data" are included in the frame (see Figure 5.7)
3. Case 2A: "Additional application data" and "Command" fields are included in the frame (see Figure 5.8)
4. Case 2B: Only "Additional application data" is included in the frame (see Figure 5.9)
5. Case 3A: "Status code" and "Command" fields are included in the frame (see Figure 5.10)
6. Case 3B: Only "Status command" field is included in the frame (see Figure 5.11)
7. Case 4A: Only "Command" field is included in the frame (see Figure 5.12)
8. Case 4B: No "Status code", "Additional application data" and "Command" fields are included in the frame (see Figure 5.13)

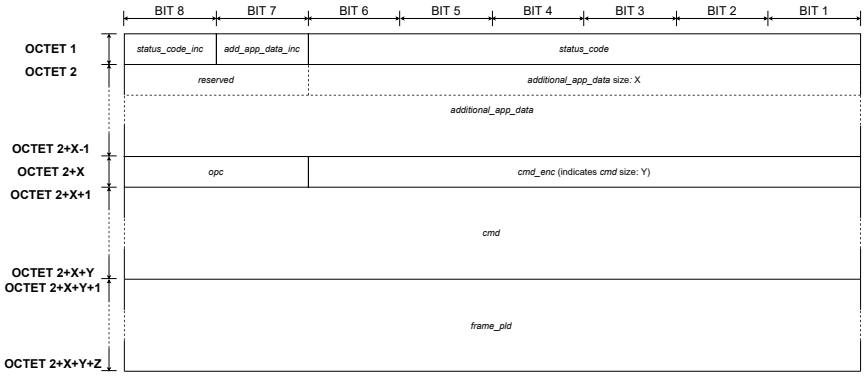


Fig. 5.6 Presentation layer - Application command - CASE 1A

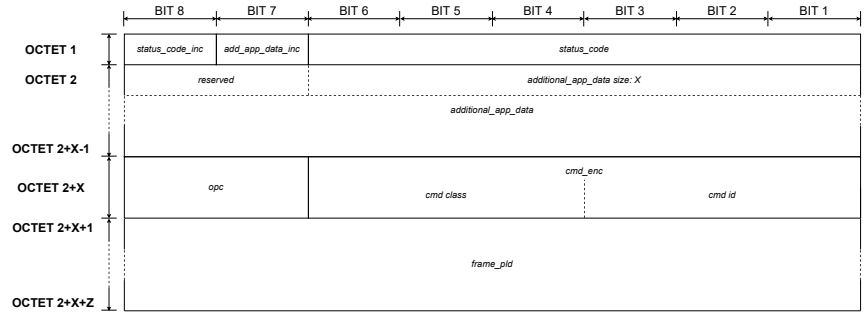


Fig. 5.7 Presentation layer - Application command - CASE 1B

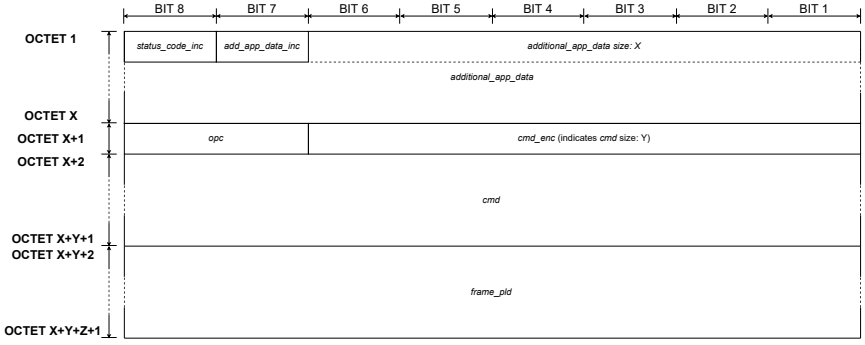


Fig. 5.8 Presentation layer - Application command - CASE 2A

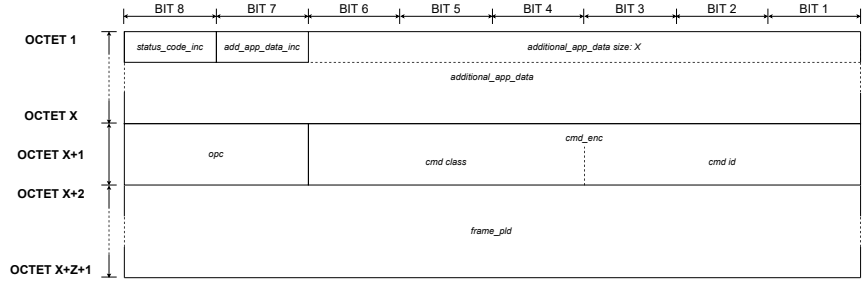


Fig. 5.9 Presentation layer - Application command - CASE 2B

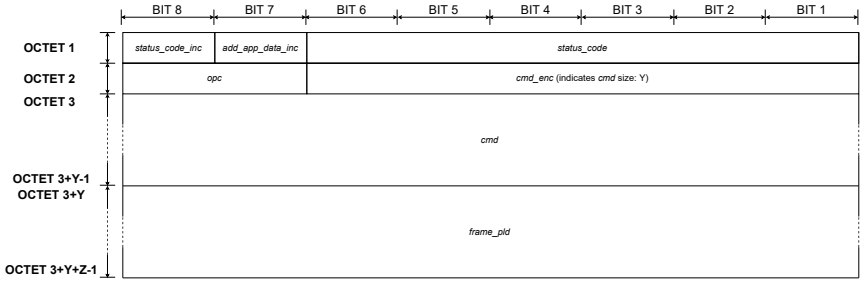


Fig. 5.10 Presentation layer - Application command - CASE 3A

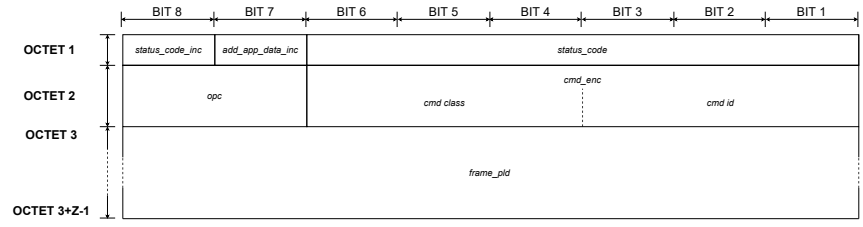


Fig. 5.11 Presentation layer - Application command - CASE 3B

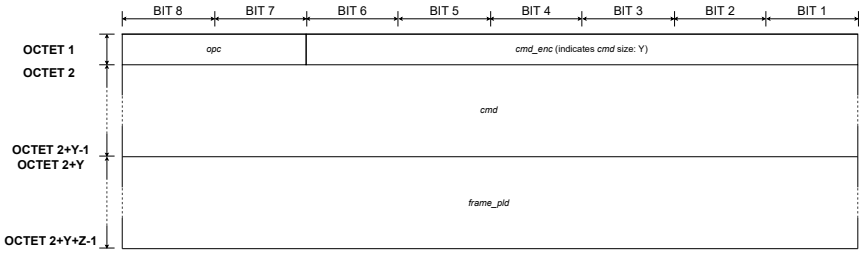


Fig. 5.12 Presentation layer - Application command - CASE 4A

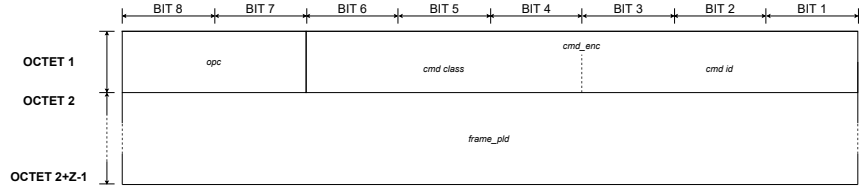


Fig. 5.13 Presentation layer - Application command - CASE 4B

1. Status code flag (status_code_inc)

Mnemonic	**Presence M/O/C**	**Length**	**Value**	
status_code_inc	C *status_hdr_inc* is set to 1	1b	Bit value	Description
			0	status code not included
			1	status code is included

The value of the status code flag indicates whether the frame includes the *status_code* field.

2. Additional data flag (add_app_data_inc)

Mnemonic	**Presence M/O/C**	**Length**	**Value**	
add_app_data_inc	C *status_hdr_inc* is set to 1	1b	Bit value	Description
			0	additional application data not included
			1	additional application data included

The value of the Additional data flag indicates whether the frame includes the *add_app_data* field.

3. Status code (status_code)

Mnemonic	**Presence M/O/C**	**Length**	**Value**
status_code	C *status_hdr_inc* is set to 1 *status_code_inc* is set to 1	6b	status code

The value of status code indicates the status code of the operation. Only selected types of messages may contain a status code, as follows:

(a) Response type messages shall always include a status code.

(b) Notification type messages may or may not include a status code.

The status code is not applicable to read or write type messages, so the status code field shall not be present in "Application command header".

Values of status codes can be found in Section 3.1 describing the application level frame format.

4. Additional application data (add_app_data)

Mnemonic	**Presence M/O/C**	**Length**	**Value**
add_app_data	C *status_hdr_inc* is set to 1 *additional_app_data_inc* is set to 1	1 to 63B	application additional data

The additional application data field contains additional application information which may be included by a device's Application Layer. Additional data shall not be added if the resulting frame exceeds the MTU size.

The 6 least significant bits of the first byte represents the length of the application additional data.

The 2 most significant bits of the first byte shall be encoded using the following rules:

(a) If "Status code" is included, the bits shall be set to 0.

(b) If "Status code" is not included, the bits contain the value of "Status code flag" and "Additional data flag".

5. Operation Code (opc)

Mnemonic	**Presence M/O/C**	**Length**	**Value**		
			Bits value		Description
opc	M	2b	0	0	read from destination
			0	1	write to destination
			1	0	notify destination with data from sender
			1	1	respond to previous read/write from destination (possibly includes data in response)

The value of "Operation Code" indicates the operation code that identifies the application command type. The application command type can be: get, set, notify, or respond. For more information, see Section 3.2.

6. Command encode (cmd_enc)

Mnemonic	**Presence M/O/C**	**Length**	**Value**
cmd_enc	M	6b	0x0 - 0x3F

The value of the Command encode field indicates the coding format of the command class; the coding format of the command ID; whether the Command field is included in the Presentation layer frame; and the size of the Command field.

The Command encode field is interpreted as follows:

(a) 0x0 to 0x3B - *cmd* field is not included. Command class and command ID are encoded in 6 bits.

(b) 0x3C to 0x3F - *cmd* field is included in the frame. The value indicates the size of the Command field as follows:

i. 0x3C indicates 1B size

ii. 0x3D indicates 2B size

iii. 0x3E indicates 3B size

iv. 0x3F indicates 4B size

7. Command (cmd)

Mnemonic	Presence M/O/C	Length	Value
cmd	C *cmd_enc* > 0x3B	1 to 4B	command class and command ID

The Command field contains the command class and command ID. For more information see Section 3.1.

8. Payload (frame_pld)

Mnemonic	Presence M/O/C	Length	Value
frame_pld	O	see NOTE	Data to be written, arguments for a read, notify payload data or response data

The Payload field contains command data. For more information, see Section 3.1.

NOTE: The field does not contain an explicit payload length. Length can be derived using the following formula: Payload length = Total payload received over the air – ASL header with optional additional application data and/or optional network diagnostic data header

5.1.2 Network Layer

5.1.2.1 Network Layer overview

The Network Layer supports the network level security aspects of the frame, like authentication, encryption, and security type.

5.1.2.2 Network Layer Frame format

The Network Layer frame is divided into three main blocks. The first block is "Frame control". The Frame control block contains information that provides information for decoding the rest of the frame. Frame control also contains information related to network security aspects and frame fragmentation. The second block is "Authentication". The Authentication block includes information used in the authentication process. The third block is "Frame payload". The Frame payload block contains higher layer data.

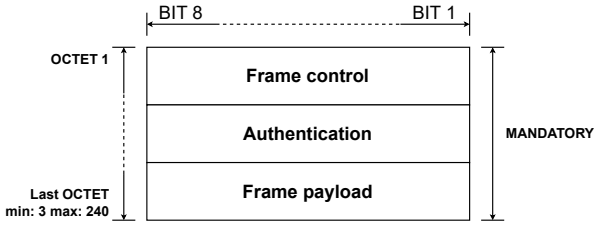


Fig. 5.14 Network layer - Blocks

5.1.2.2.1 Network Layer Frame Control Fields

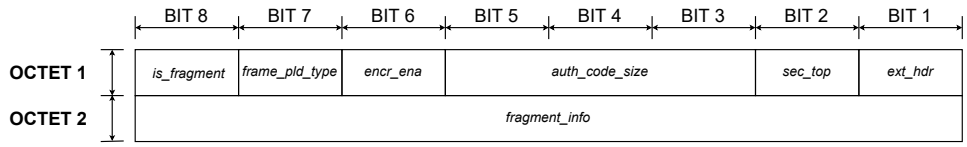


Fig. 5.15 Network layer - Frame control

1. Packet fragmentation flag (is_fragment)

Mnemonic	Presence M/O/C	Length	Value	
is_fragment	M	1b	Bit value	Description
			0	not fragmented
			1	fragmented

The value of the Packet fragmentation flag indicates whether or not the Network Layer frame is fragmented.

NOTE: The current release of the protocol does not support frame fragmentation. The field shall be set to 0.

2. Payload type flag (frame_pld_type)

Mnemonic	Presence M/O/C	Length	Value	
frame_pld_type	M	1b	Bit value	Description
			0	reserved
			1	Presentation Layer frame

The value indicates the type of payload available in the field *frame_pld*.

NOTE: The current release of the protocol supports one payload type only and this is the Presentation Layer frame. This bit shall be set to 1 by default.

3. Encryption flag (encr_ena)

Mnemonic	Presence M/O/C	Length	Value	
encr_ena	M	1b	Bit value	Description
			0	encryption disabled
			1	encryption enabled

The value indicates whether the network frame payload (*frame_pld*) field is encrypted.

4. Authentication code size (auth_code_size)

Mnemonic	Presence M/O/C	Length	Value			
			Bits value			Description
auth_code_size	M	3b	0	0	0	no auth code
			0	0	1	2 B
			0	1	0	4 B
			0	1	1	8 B
			1	0	0	12 B
			1	0	1	16 B
			1	1	0	reserved
			1	1	1	reserved

The value indicates the size of the authentication code field (*auth_code*). The maximum allowed size is:

(a) 16 B when *ext_hdr* is set to 1 and *sec_ref* is set to 0

(b) 12 B when *ext_hdr* is set to 1 and *sec_ref* is set to 1

(c) 16 B when *ext_hdr* is set to 0.

5. Network security topology (sec_top)

Mnemonic	Presence M/O/C	Length	Value	
			Bit value	Description
sec_top	M	1b	0	reserved
			1	WAN security topology

The value indicates the security topology.

NOTE: The current release of the protocol supports only one security topology. This bit shall be set to 1 by default.

6. Extended header flag (ext_hdr)

Mnemonic	Presence M/O/C	Length	Value	
			Bit value	Description
ext_hdr	M	1b	0	*sec_ref* field is not present
			1	*sec_ref* field is present

The value indicates extended configuration of the "Authentication" part of the network frame.

When *ext_hdr* is set to 0, then the GCS security reference shall be used.

When *ext_hdr* is set to 1, then the GCS security reference is indicated by *sec_ref* field.

7. Fragmentation info (fragment_info)

Mnemonic	Presence M/O/C	Length	Value
fragment_info	C *is_fragment* is set to 1	1B	reserved

The value indicates the fragment number and the total number of fragmented packets.

NOTE: The current release of the protocol does not support frame fragmentation. The field is_fragmented shall be set to 0 so fragment_info is not included in the network frame format.

5.1.2.2.2 Authentication fields

"Authentication" part of the frame may have three different formats based on the configuration included in the "Frame control" block and also the *sec_ref* field which is included only in case *ext_hdr* is set to 1. It is possible for the "Authentication" part not to be included in the Network Layer. The possible configurations are as follows:

1. *ext_hdr* is set to 0 and:
   (a) *auth_code_size* > 0 then the configuration includes *auth_code* field only in "Authentication" block (see Figure 5.16).
   (b) *auth_code_size* is set to 0 then the configuration excludes the whole "Authentication" block from the Network Layer frame.
2. *ext_hdr* is set to 1, *auth_code_size* > 0 this configuration applied in "Frame control" indicates the presence of *sec_ref* in the "Authentication" block. This configures the final format of the "Authentication" block:
   (a) *sec_ref* is set to 0 then the configuration includes *auth_code* field (see Figure 5.17).
   (b) *sec_ref* is set to 1 then the configuration includes *counter_val* and *auth_code* fields (see Figure 5.18).

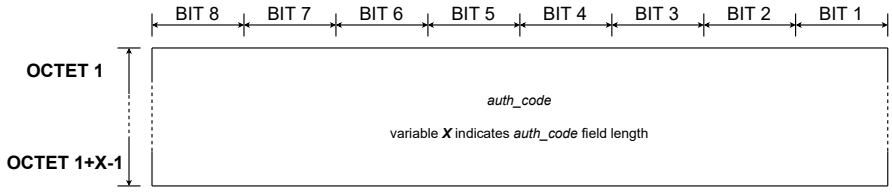


Fig. 5.16 Network layer - Authentication - Authentication code

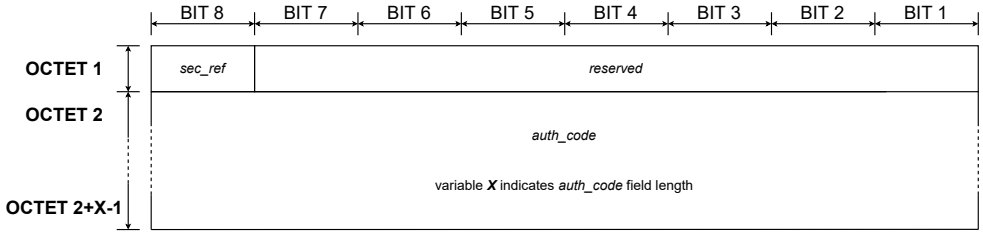


Fig. 5.17 Network layer - Authentication - Security reference GCS

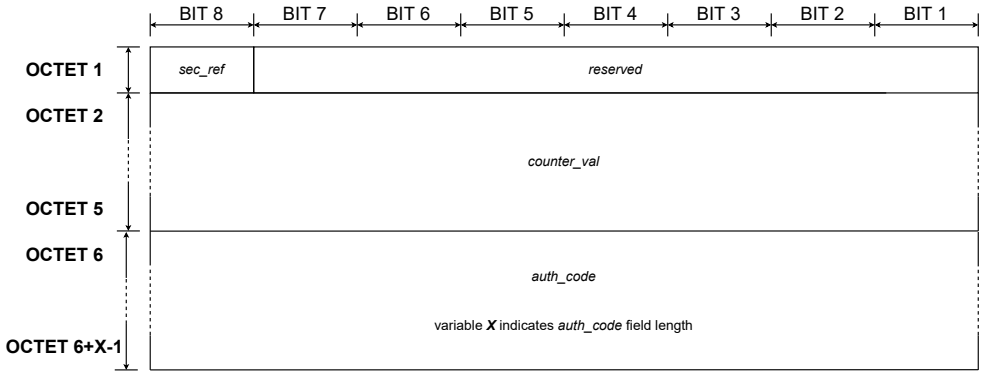


Fig. 5.18 Network layer - Authentication - Security reference COUNTER

1. Authentication code (auth_code)

Mnemonic	Presence M/O/C	Length	Value
auth_code	C *ext_hdr* is set to 0 *auth_code_size* >0	see *auth_code_size* field	authentication code

The field contains the value of the authentication code.

2. Security reference information (sec_ref)

Mnemonic	Presence M/O/C	Length	Value		
sec_ref	C *ext_hdr* is set to 1	1B	Bit number	Bit value	Description
			7	0	security reference GCS
				1	security reference COUNTER
			0 to 6		reserved

The field indicates the security reference configuration. The default value of the GCS security reference is 0.

3. Counter value (counter_value)

Mnemonic	Presence M/O/C	Length	Value
counter_val	C *ext_hdr* is set to 1 *auth_code* > 0 *sec_ref* is set to 1	4B	counter value

The field contains the counter value.

5.1.2.2.3 Network Layer Frame payload field (frame_pld)

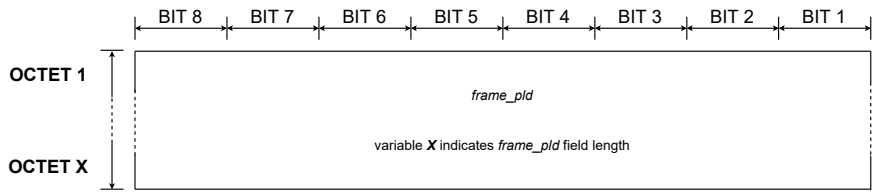


Fig. 5.19 Network layer - Payload

The field contains Presentation Layer frame data.

Mnemonic	Presence M/O/C	Length	Value
frame_pld	M	1 — 221B	Presentation layer frame

NOTE: The current release of the protocol shall have the Presentation Layer frame included in the Network Layer payload.

5.1.2.2.4 Network Layer configuration

Three generic formats of the Network Layer are possible based on the "Frame control" part:

1. The security information is not included in the frame: *auth_code_size* = 0, *ext_hdr* = 0. The frame contains only "Frame control" and "Frame payload" parts (see Figure 5.20).

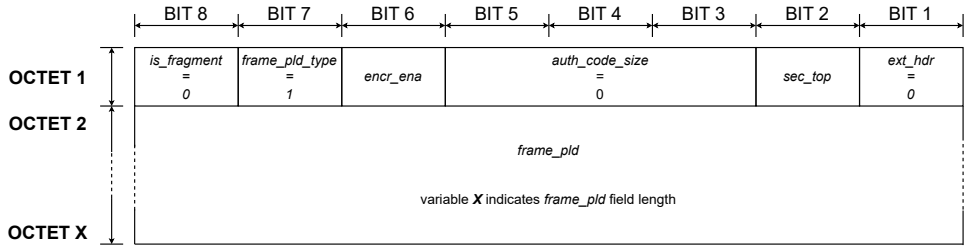


Fig. 5.20 Network layer - Frame format - Case 1

2. The security information is included but an extended header is not present: *auth_code_size* > 0, *ext_hdr* = 0. The frame contains the following parts: "Frame control" , "Authentication" and "Frame payload" (see Figure 5.21).

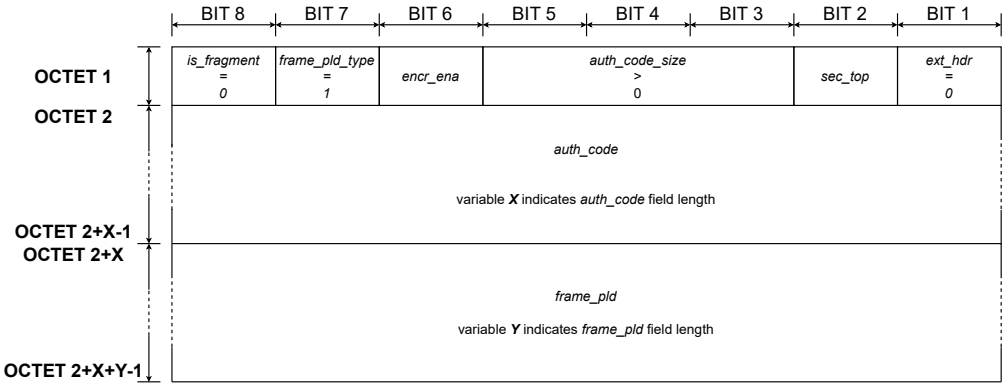


Fig. 5.21 Network layer - Frame format - Case 2

3. The security information is included in the extended header: *auth_code_size* > 0, *ext_hdr* = 1. Additionally *sec_ref* configures the format of "Authentication" block. Figure 5.22 shows the case when *sec_ref* is set to 0 and Figure 5.23 shows the case when *sec_ref* is set to 1. The frame contains all parts: "Frame control", "Authentication" and "Frame payload".

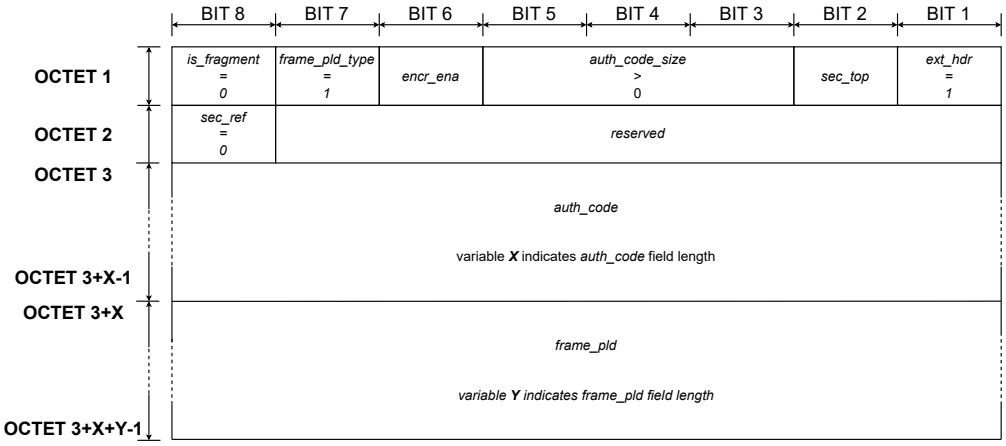


Fig. 5.22 Network layer - Frame format - Case 3A

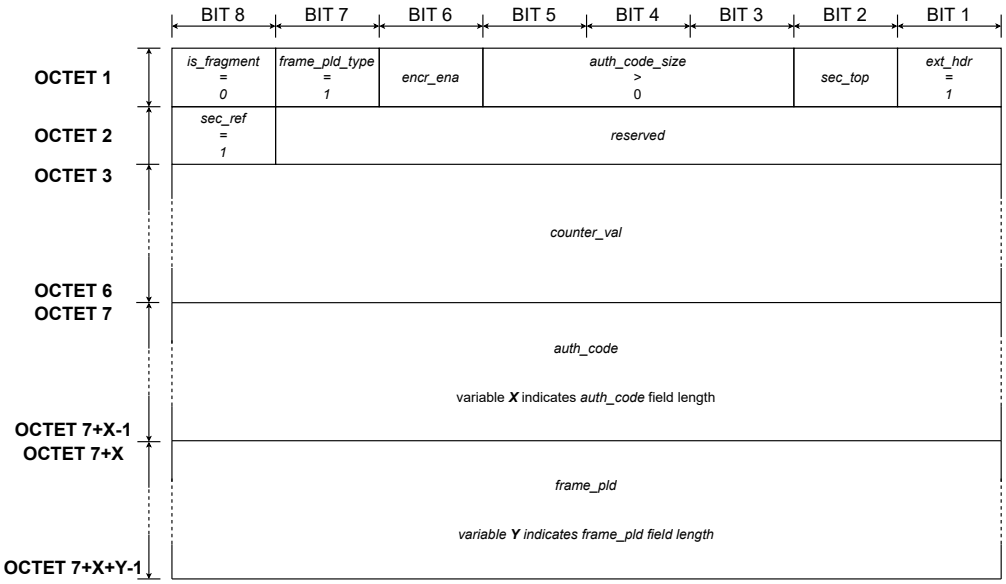


Fig. 5.23 Network layer - Frame format - Case 3B

5.1.3 Link Layer

The Link Layer is involved in Gateway discovery, maintenance, and link establishment for message exchange between two Endpoints. These functions are handled by two separate frame formats: Beacon frame format and message frame format.

The Beacon frame format is used for broadcasting status, and for broadcasting radio configuration parameters. Endpoints use the information in the Beacon frame to choose a Gateway. Information from the Beacon frame is used for maintenance of the Gateway-Endpoint connection, such as power control or channel hopping. The Beacon frame also informs about the state of the Gateway - for example, indicating congestion.

The message frame format is used for data exchange and data reception acknowledgement at the Link Layer level. The frame used for data exchange includes higher layer frames. The frame used for acknowledgement is terminated at the Link Layer level and does not contain a frame payload. Acknowledgements are sent between a Gateway and an Endpoint. Acknowledgements are only used with synchronous transmission. Asynchronous transmission type shall not use Acknowledgements.

5.1.3.1 Beacon Frame format

The frame is divided into different parts containing the Beacon-specific information as depicted in Figure 5.24. The first one is “Frame control”, that provides information that allows correct decoding and interpretation of the frame. “Source address” and “Destination address” identify the sender of the Beacon and the PAN ID, which are part of the Gateway’s owner identification. “Connection status” provides information related to Gateway connection status to the cloud, its load, operation mode (synchronous/asynchronous or synchronous only) and roaming settings. “Power control” provides information on the antenna and power settings used by the Gateway. “Hopping parameters” contain values needed for calculation of an active channel used for uplink or downlink transmission. “Data and Security control” identifies the presence of optional “Data” part and also data size included in “Security”. “Data” is a container of all optional data that could be transmitted by the Gateway. “Security” contains a tag authenticating the Gateway for devices registered to the Gateway owner’s account.

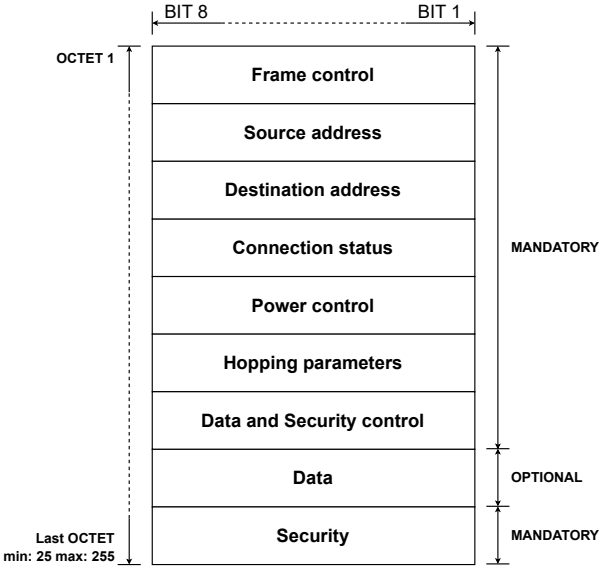


Fig. 5.24 Link layer - Beacon - Blocks

5.1.3.1.1 Link Layer Frame Control Fields

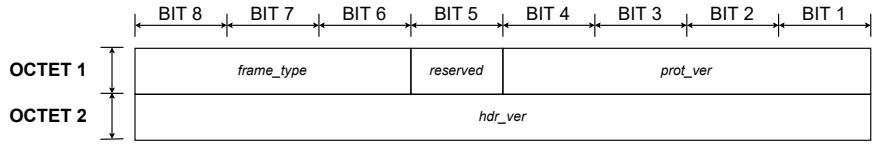


Fig. 5.25 Link layer - Beacon - Frame control

1. Frame type (frame_type)

Mnemonic	Presence M/O/C	Length	Value			
frame_type	M	3b	Bits value			Description
			0	0	0	Beacon

The value indicates the frame type.

2. Protocol Version (prot_ver)

Mnemonic	Presence M/O/C	Length	Value				
prot_ver	M	4b	Bits value				Description
			0	1	0	0	Halo 2.0 protocol version

The value of the protocol version indicates the Amazon Sidewalk Protocol version number. An Endpoint that receives the Beacon frame with a protocol version which is different from the supported one shall neither decode the frame nor join to the Gateway.

3. SubG-FSK version (hdr_ver)

Mnemonic	Presence M/O/C	Length	Value
hdr_ver	M	1B	0x0

The value of the SubG-FSK version indicates the version of SubG-FSK supported by the Gateway. SubG-FSK version identifies the Beacon, hopping pattern and PHY version supported by the Gateway.

5.1.3.1.2 Beacon Source address field (src)

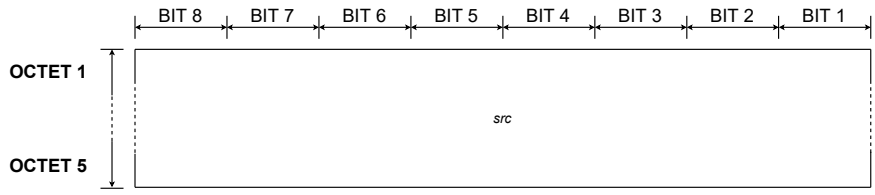


Fig. 5.26 Link layer - Beacon - Source address

Mnemonic	Presence M/O/C	Length	Value
src	M	5B	global device ID

The value indicates the device ID of the Gateway that is sending the Beacon.

5.1.3.1.3 Beacon Destination address field (dst)

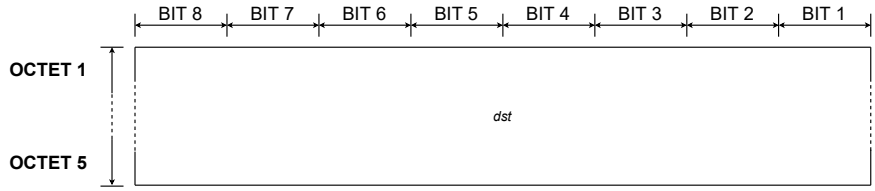


Fig. 5.27 Link layer - Beacon - Destination address

Mnemonic	Presence M/O/C	Length	Value
dst	M	5B	PAN ID

The value indicates the globally unique PAN ID.

5.1.3.1.4 Beacon Connection status fields

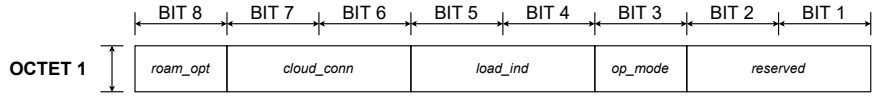


Fig. 5.28 Link layer - Beacon - Connection status

1. Roaming Support (roam_opt)

Mnemonic	**Presence M/O/C**	**Length**	**Value**	
roam_opt	M	1b	Bit value	Description
			0	roaming forbidden
			1	roaming allowed

The value of the roaming support indicates whether the Gateway transfers packets between roaming Endpoints and the Amazon Sidewalk Cloud.

> *NOTE: An Endpoint device that is roaming shall not attempt to connect to a Gateway that does not support roaming. If a Gateway has roaming disabled, then the Gateway can recognize roaming devices and drop packets from roaming devices.*

2. Cloud connection indicator (cloud_conn)

Mnemonic	**Presence M/O/C**	**Length**	**Value**		
cloud_conn	M	2b	Bits value		Description
			0	0	disconnected
			0	1	experiencing recent intermittent disconnects
			1	0	connected in “Distress Gateway” mode
			1	1	connected

The value of the cloud connection indicator indicates the Gateway-cloud connection condition. Endpoints may use this indicator while selecting a Gateway.

For more information on the “Distress Gateway” mode, see Section 5.6.5.

3. Load indicator (load_ind)

Mnemonic	**Presence M/O/C**	**Length**	**Value**		
			Bits value		Description
load_ind	M	2b	0	0	light traffic
			0	1	
			1	0	
			1	1	heavy traffic

The value of the load indicator indicates the traffic load of the Gateway.

4. Gateway operation mode (op_mode)

Mnemonic	**Presence M/O/C**	**Length**	**Value**	
			Bit value	Description
op_mode	M	1b	0	synchronous and asynchronous mode
			1	synchronous mode

The value of the Gateway operation mode indicates the modes of communication actively supported by the Gateway. The Gateway may be in synchronous transmission mode and not supporting asynchronous transmission mode, or the Gateway may be in preamble cycling mode. In preamble cycling mode, the Gateway cycles between synchronous and asynchronous transmission modes. (See Section 5.2 and Section 5.3.)

5.1.3.1.5 Beacon Power control fields

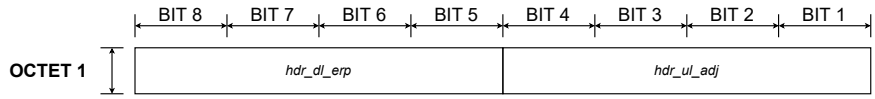


Fig. 5.29 Link layer - Beacon - Power control

1. SubG-FSK downlink EIRP (hdr_dl_erp)

Mnemonic	Presence M/O/C	Length	Value
hdr_dl_erp	M	4b	0x0 – 0xF

The value of the power control fields indicates the effective isotropic maximum radiated power of the Gateway used for the current Beacon round.

EIRP in dBm = $hdr\_dl\_erp * 4$

2. SubG-FSK uplink power adjustment (hdr_ul_adj)

Mnemonic	Presence M/O/C	Length	Value
hdr_ul_adj	M	4b	0x0 – 0xF

The value of the SubG-FSK uplink power adjustment indicates the offset from ideal sensitivity in dB due to interference the Gateway is receiving.

Adjustment in dB = $hdr\_ul\_adj * 4$

5.1.3.1.6 Beacon Hopping parameters fields

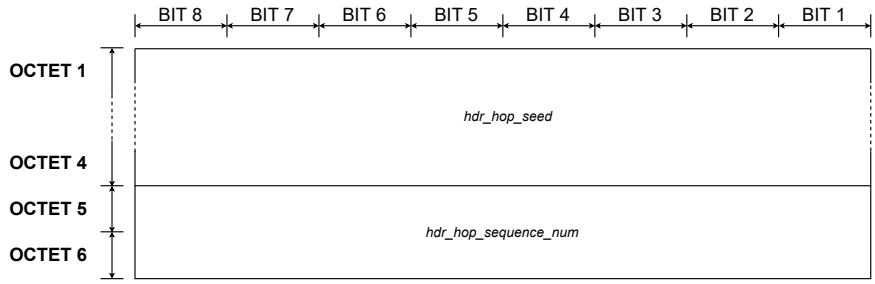


Fig. 5.30 Link layer - Beacon - Hopping parameters

1. Hopping seed

Mnemonic	Presence M/O/C	Length	Value
hdr_hop_seed	M	4B	0x0 – 0xFFFFFFFF

The Hopping seed field contains the value of the seed used in the channel hopping algorithm, described in Section 5.2.4.

2. Hopping sequence number

Mnemonic	Presence M/O/C	Length	Value
hdr_hop_sequence_num	M	2B	0x0 – 0xFFFF

The value of the Hopping sequence number field contains the value of the sequence number used in the channel hopping algorithm, described in Section 5.2.4.

5.1.3.1.7 Beacon Data and Security control fields

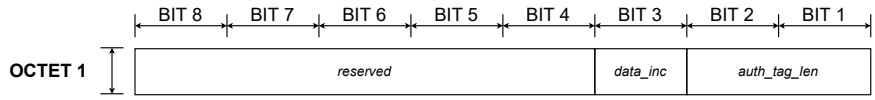


Fig. 5.31 Link layer - Beacon - Data and Security control

1. Data flag (data_inc)

Mnemonic	**Presence M/O/C**	**Length**	**Value**	
data_inc	M	1b	Bit value	Description
			0	Data is not included
			1	Data is included

The value of the Data flag indicates the existence of optional data in the frame.

2. PAN authentication tag length (auth_tag_len)

Mnemonic	**Presence M/O/C**	**Length**	**Value**		
auth_tag_len	M	2b	Bits value		Description
			0	0	4B
			0	1	8B
			1	0	12B
			1	1	16B

The value indicates the size of *pan_auth_tag* in the Security part of the frame.

5.1.3.1.8 Beacon Data fields

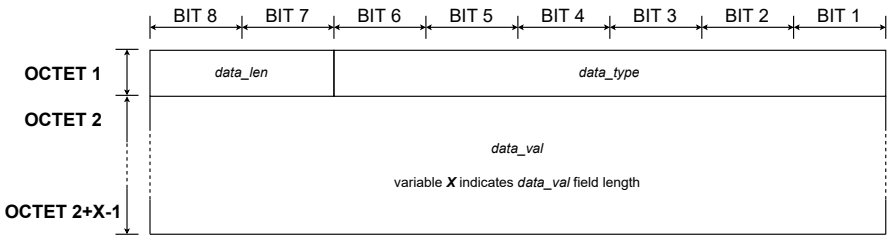


Fig. 5.32 Link layer - Beacon - Data - Case 1

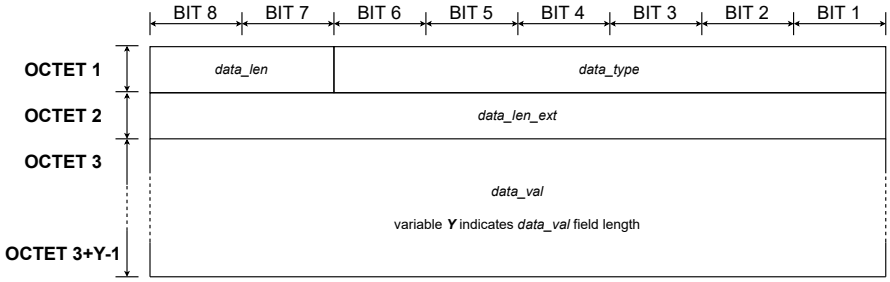


Fig. 5.33 Link layer - Beacon - Data - Case 2

1. Data length (data_len)

Mnemonic	Presence M/O/C	Length	Value		
			Bits value		Description
data_len	C *data_inc* is set to 1	2b	0	0	1B
			0	1	2B
			1	0	4B
			1	1	data size is indicated in *data_len_ext* field

The value of Data length indicates the size of the *data_val* field.

2. Data type (data_type)

Mnemonic	Presence M/O/C	Length	Value	
data_type	C *data_inc* is set to 1	6b	Value	Description
			0x1	SubG-FSK-synchronous CS schedule

The value of the Data type field indicates the data type included in the *data_val* field.

3. Data length extension (data_len_ext)

Mnemonic	Presence M/O/C	Length	Value
data_len_ext	C *data_inc* is set to 1 and *data_len* is set to 0x3	1B	size of *data_val* field

The value of the Data length extension field indicates the size of the *data_val* field.

4. Data value (data_val)

Mnemonic	Presence M/O/C	Length	Value
data_val	C *data_inc* is set to 1	see *data_len* or *data_len_ext*	optional data values

The value of the Data value field contains the optional data value. The length of the field is indicated in *data_len* or *data_len_ext*.

The field value for "SubG-FSK-Synchronous CS schedule" for the ad hoc access model is coded as follows:

1B: *CS-UL* offset

1B: *CS-UL* periodicity

1B: CS-DL offset

1B: CS-DL periodicity

5.1.3.1.9 Beacon Security field (pan_auth_tag)

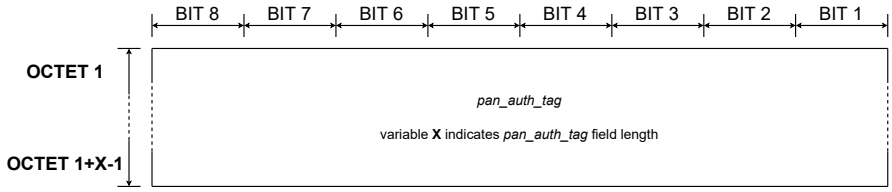


Fig. 5.34 Link layer - Beacon - Security

Mnemonic	Presence M/O/C	Length	Value
pan_auth_tag	M	see *auth_tag_len* field	authentication tag value

The security field contains the value of the authentication tag.

5.1.3.2 Message frame format

The frame format has the following components:

- "Frame control" provides information that allows decoding of the frame and also controls the frame reception acknowledgment.
- "Source address" and "Destination address" identify both ends of the communication link.
- "Sequence number" contains the uplink or downlink message number that allows control of the correct sequence of data transmission and reception.
- "Frame payload" contains data from upper layers.

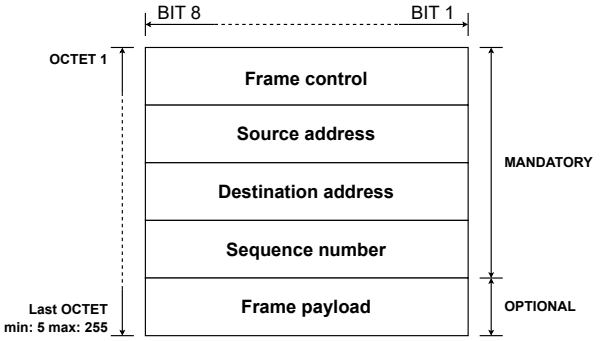


Fig. 5.35 Link layer - Message - Blocks

5.1.3.2.1 Message Frame Control Fields

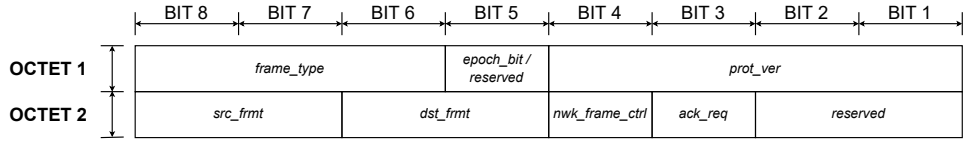


Fig. 5.36 Link layer - Message - Frame control

1. Frame type (frame_type)

Mnemonic	Presence M/O/C	Length	Value			
frame_type	M	3b	Bits value			Description
			0	0	1	Data
			0	1	0	Network Data
			0	1	1	ACK

The value of Frame type indicates the frame type. Based on this parameter, other fields in the frame may be optional or not present.

2. Epoch frame flag (epoch_bit/reserved)

Mnemonic	Presence M/O/C	Length	Value	
epoch_bit/reserved	M	1b	Asynchronous mode uplink only	
			Bit value	Description
			0	Non-Epoch frame
			1	Epoch frame
			Synchronous uplink or downlink and asynchronous downlink mode	
			0	reserved

The Epoch frame flag field is valid for the data frame format, with uplink direction in asynchronous mode only. The Epoch frame flag indicates whether the frame is used for the Endpoint to provide a time reference to the Gateway to synchronize Target Transmission Times.

The field is reserved and should be set to 0 for all other cases. Specifically:

(a) Data frame format in synchronous mode and asynchronous mode in the downlink direction.

(b) Network Data frame format for synchronous mode and asynchronous mode.

(c) ACK frame format.

NOTE: The Epoch frame flag field shall always be set to 0 for synchronous mode.

3. Protocol version (prot_ver)

The Protocol version value indicates the Amazon Sidewalk Protocol version number.

Mnemonic	Presence M/O/C	Length	Value				
prot_ver	M	4b	Bits value				Description
			0	1	0	0	Halo 2.0 protocol version

NOTE: The current version of the specification supports only one protocol version.

4. Source field format (src_frmt)

   The Source field format value indicates the sender of the data packet and additionally gives information about the *src* field format.

Mnemonic	Presence M/O/C	Length	Value		
src_frmt	M	2b	Bits value		Description
			0	0	cloud
			0	1	global device ID
			1	0	reserved
			1	1	reserved

5. Destination field format (dst_frmt)

Mnemonic	Presence M/O/C	Length	Value		
dst_frmt	M	2b	Bits value		Description
			0	0	cloud
			0	1	global device ID
			1	0	reserved
			1	1	reserved

   The Destination field format value indicates the recipient of the data packet and gives information about the *dst* field format. When the cloud is the target, the Link Layer ACK response is sent by the Gateway. The message reception confirmation done by the cloud is handled in higher layers.

6. Network frame flag (nwk_frame_ctrl)

Mnemonic	Presence M/O/C	Length	Value	
nwk_frame_ctrl	M	1b	Bit value	Description
			0	*frame_pld* field is not present
			1	*frame_pld* field is present

   This Network frame flag field indicates whether the Link Layer frame includes the Network Layer frame.

   The field is valid for “Data” and “Network Data” frame types only and shall be set to 1.

   ACK frame type shall always be set to 0. Acknowledgement does not transport any data from higher layers.

7. Message acknowledge request (ack_req)

Mnemonic	Presence M/O/C	Length	Value	
ack_req	M	1b	Bit value	Description
			0	ACK not requested
			1	ACK requested

The Message acknowledge request field indicates whether a Link Layer acknowledgement of the message reception is requested. The acknowledgement may be requested for messages sent in synchronous connection mode only. The field shall always be set to 0 in messages sent in asynchronous connection mode.

The Message acknowledge request field is valid for data and network data frame formats only. The Message acknowledge request field is set to 1 by default for the following destinations: *cloud* and *global device ID*.

ACK frame type should always be set to 0 and has no meaning in this case.

5.1.3.2.2 Message Source address field (src)

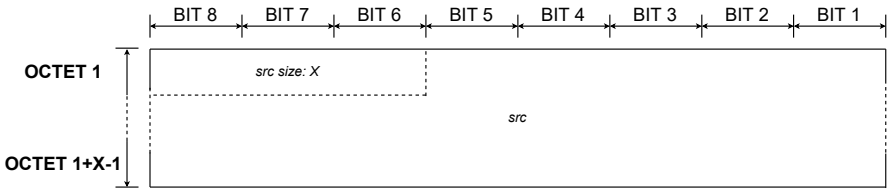


Fig. 5.37 Link layer - Message - Source address

Mnemonic	Presence M/O/C	Length	Value
src	M	1 – 5B	3 most significant bits of the first byte specify the field size.

The Source address field contains the source address of the frame. The Source address format is indicated by the *src_frmt* field.

5.1.3.2.3 Message Destination address field (dst)

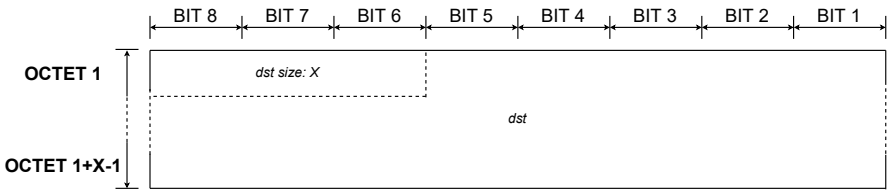


Fig. 5.38 Link layer - Message - Destination address

Mnemonic	Presence M/O/C	Length	Value
dst	M	1 – 5B	3 most significant bits of the first byte specify the field size.

The Destination address field contains the destination address of the frame. The Destination address format is indicated by the *dst_frmt* field.

5.1.3.2.4 Message Sequence number field (seq_num)

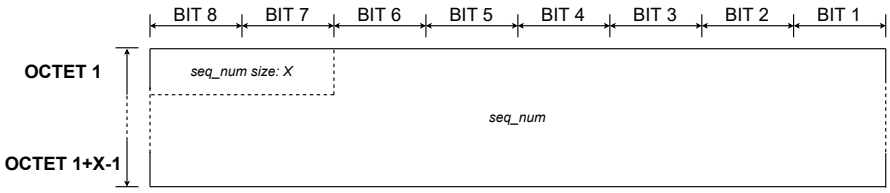


Fig. 5.39 Link layer - Message - Sequence number

Mnemonic	Presence M/O/C	Length	Value
seq_num	M	1 – 3B	2 most significant bits of the first byte specify the field size.

The sequence number field indicates the sender's packet sequence number.

The ACK frame type shall have a *seq_num* copy of the frame where acknowledgement was requested.

5.1.3.2.5 Message Link Layer Frame payload field (frame_pld)

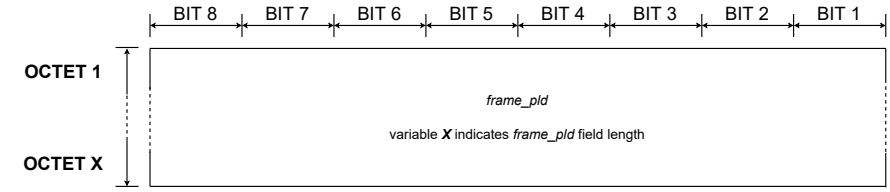


Fig. 5.40 Link layer - Message - Frame payload

Mnemonic	Presence M/O/C	Length	Value
frame_pld	C *nwk_frame_ctrl* is set to 1	0 – 240B	Network Frame bytes

This field contains the frame payload which includes the Network Layer Frame. This field can be included only if *nwk_frame_ctrl* is set to 1.

This field is not present in the ACK frame format.

5.1.4 Physical layer

The Amazon Sidewalk protocol uses a sub-GHz UHF band, ITU [1] ISM band 9 for signal transmission. Two modulation techniques in sub-GHz are defined by 802.15.4 standard [2] as follows:

GFSK (Gaussian Frequency Shift Keying) modulation is used in SubG-FSK, and LoRa [3] modulation is used in SubG-CSS.

For more information, see Section 5.7. In both cases, half-duplex (HDX) transmission schemas are implemented to allow multi-user access.

In both cases, the RF frame consists of preamble, sync word, header, payload (Link Layer Frame) and optional Frame Check Sequence (FCS).

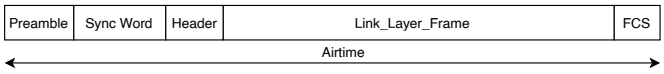


Fig. 5.41 RF frame

The RF frame length is limited by regional certification authority requirements for airtime occupation. See ETSI EN 300 220-2 and FCC part 15.247 for details. Allowed airtime and power levels depend on the region and operating bands. For more information, see Section 5.7. An additional timing correction path is implemented because transmitter and receiver clocks are not synchronized and their frequencies drift. For more information, see Section 5.1.4.3.

5.1.4.1 Data order

Bit order: i.e. byte 'e' (0x65, 01100101 B) is presented at the baseband bus timeline as data sequence (see Figure 5.42).

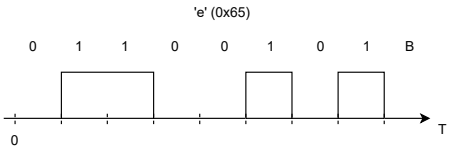


Fig. 5.42 Bits order

Byte order: Little endian is defined in the PHY frame [2].

5.1.4.2 RF frame formats

The physical frame is built as follows: The physical header (optionally precoded) is attached to a predefined preamble. Then, RF channel code is specified then the buffered data packet is channel encoded. The Frame Check Sequence is calculated and attached to the end of the frame. The baseband RF frame is sent to the modulator and up-converted to the preselected RF channel.

NOTE: Preamble, channel coding and FCS calculation are HW implemented.

Four steps are necessary for proper RF data reception. First, the receiver must synchronize its demodulator with preamble signal. The next step is to demodulate and decode header to gather information regarding receiving frame. Then, payload can be properly received. Correctness of transmission is confirmed by

calculation of Frame Check Sequence at the end. After FCS evaluation link layer data content can be passed for further processing or discarded.

RF frame consists of five segments:

1. Preamble - for bit synchronization, can be repeated for improving receiver synchronization and probability of reception (due to physical aspects: time drift, multipath effect, etc.);
2. Sync Word - mark for start of data block;
3. PHY Header - contains PHY data control bits;
4. PHY payload - Link Layer Frame;
5. FCS (Frame Check Sequence) - data block checksum.

The content and length of each segment depends on predefined modulation scheme.

5.1.4.2.1 Sub-G FSK frame format details

Bit '1' is represented as positive frequency shift (+dev/2 Hz), bit '0' - as negative frequency shift against center frequency F0.

The SubG-FSK PHY frame format follows IEEE 802.15.4. Preamble is followed with PHY layer header and data field. The frame is completed with a Frame Check Sequence (FCS). The field description details are as follows:

1. Preamble

   Preamble consists of multiple bytes: 0x55, 01010101 B. Preamble can be extended for time drift compensation.

2. Synchronization Word

   Synchronization Word (defined in [2] as SFD - Start Frame Delimiter). SFD is 2 B length.

   FEC disabled 0x904E (default value).

3. PHR PHY header

   PHY header consists of 16 bits (2 B).

   0-2 RSVD (0 default);

   3 FCS type: 0-32 bits, default value: 1-16bit;

   4 Data whitening: 0- disabled; 1-PN9 sequence with 0xF as starting condition is used (default value);

   5-15 PSDU length (only 251 B are available in the present version of the specification).

4. Payload

   The payload length is defined by PSDU length of PHY header, the payload is scrambled if Data whitening of PHY header is set. The maximum payload length is 251 B.

5. FCS Frame check sequence

   FCS type 1: Polynomial x16+x12+x5+1 (CCITT) is used for 16 bit FCS, default value;

   FCS type 0: In case of 32bit FCS then ANSI X3.66-1979 shall be used.

5.1.4.2.2 SubG-FSK Airtime summary

RF data rate is: 50 kbps, Application layer MTU size is up to 200 B;

PHY frame length is up to 251 B, < 55.12ms (14ms preamble).

5.1.4.2.3 SubG-CSS frame format details

LoRa symbols are used for SubG-CSS. LoRa symbols are composed of chirps. Detailed LoRa uplink PHY frame format is described in Application Note AN1200.22 [3]. Preamble is followed by PHY layer header (PHDR), then calculated PHDR_CRC is placed. The next data field is attached. Uplink frame is completed with a CRC. With current implementation, only SF11 with Coding Rate (CR) 4/5 is used. The field description details are as follows:

1. Preamble

   The number of preamble symbols is programmable. Downlink Preamble can vary between 8 - 23 symbols in order to compensate for any timing error. Uplink Preamble contains 16 symbols.

2. Synchronization Word contains value 0x34.

3. PHY header

   PHDR - contains 6 symbols.

   PHDR_CRC - 16 bit CRC.

   PHY header is protected by 4/8 CRC.

4. PHY Payload

   PHY payload - contains encapsulated higher layer frame. Max Payload size is 44 B for SF 11.

5. CRC

   Polynomial x16+x12+x5+1 is used for 16 bit CRC calculation. If 8 bit CRC is used, then last 8 bits of 16 bit CRC are used. The default value for CRC is 16 bits.

Upchirps are used in the uplink, downchirps in the downlink. In the downlink, the symbols are IQ inverted. This is done to achieve orthogonality between uplink and downlink signals.

5.1.4.2.4 SubG-CSS Airtime summary

RF data rate is < 1760 bps, PDU size up to 44 B;

Uplink PHY frame duration is: 279.552 ms (preamble 16 symbols, data 44 B);

Downlink PHY frame duration is: 246.784 ms (preamble 8 symbols, data 44 B), 308,224 ms (preamble 23 symbols, data 44 B).

5.1.4.3 Time drift

In synchronous mode the transmitter chooses timing of packet's preamble in accordance with expected receivers listening window.

Due to oscillator (XO) frequency offsets and drifts multiplied by MCU clock generator internally calculated in Endpoint RF frame timings can be biased with error which exceed system requirements. When frequency disparity between two nodes exceeds the specifications of +/-100 ppm, the Endpoint may not have enough preamble symbols, resulting in a packet loss. The time drift between the nodes is impacted by calibration errors between the hardware and temperature-dependent oscillation frequency variations. The Amazon Sidewalk SDK provides two mechanisms for bounding the drift:

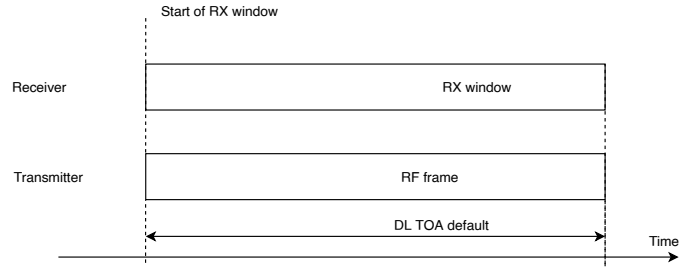


Fig. 5.43 No timedrift

1. Temperature compensation: Internal Endpoint RTC compensation based on calibration table and measuring circuit temperature in 10 s intervals. Procedure is valid for each state of connection (asynchronous and synchronous).
2. Link layer offset compensation: Compensate for any timing error between the Gateway and the Endpoints.

   Measured drift constitutes time offset to apply for an Endpoint's upcoming link layer operations.

For SubG-CSS, TX drift compensation is implemented (additional 30 ms preamble length is added). RX window size must cover possible worst case scenarios: 30 ms TX time advance due to clock rushing out and 30 ms additional TX time extension due to extended preamble.

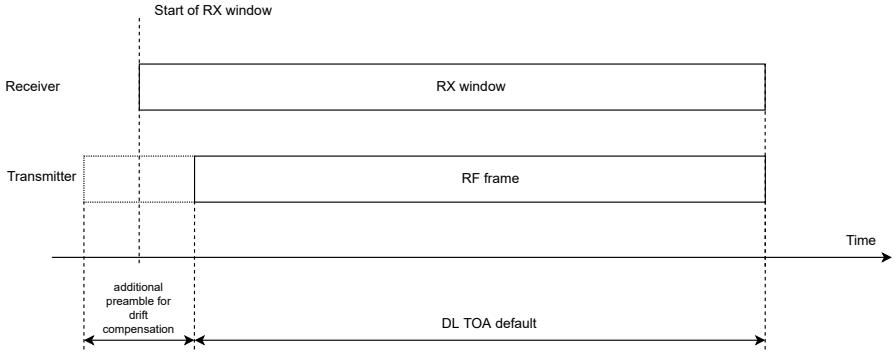


Fig. 5.44 Timedrift compensation

5.2 SubG-FSK Connection

Endpoints working in SubG-FSK mode shall be synchronized with a Gateway before an uplink or downlink transmission is performed. The synchronization mechanism works by searching for a periodic frame sent by Gateways, namely Beacon frames. Endpoints shall detect eligible Gateways by searching and receiving Beacons. In addition to its role in synchronization, Beacon payload carries basic information about the state of the Gateway, such as the Gateway's cloud connection status, hopping seed or sequence number. The Beacon payload is not encrypted, allowing discovery and connection initiation by all Amazon Sidewalk devices. For a detailed description of the Beacon frame format and information carried by the Beacon, see Section 5.1.3.1. Section 5.2.1 describes the Beacon discovery mechanism that Endpoints use in initial synchronization with a Gateway.

Section 5.2.2 describes how Endpoints maintain synchronization with a Gateway.

Section 5.2.3 describes Endpoints' data packet transmission scheduling in both directions.

Section 5.2.4 describes the mechanism of the channel number calculation.

Section 5.2.5 describes the mechanism that reduces/eliminates over the air collision.

5.2.1 Beacon discovery

The procedure of discovering nearby Gateways through their periodic Beacon transmissions is called the Beacon discovery procedure. The Beacon discovery procedure is based on traversing the channel list and performing a preamble detection on each channel. The Beacon discovery procedure is divided into two phases: preamble detection in the currently selected channel and channel change operation. If a preamble is not detected on a channel, then the Endpoint shall switch to the next channel from the list and repeat the operation. If a preamble is detected, then the Endpoint shall receive the PHY frame and check if the received frame is a Beacon frame from a Gateway eligible for connections. The duration of each phase is defined by the following:

1. *tr_det_listening* - time period used for preamble detection
2. *ch_change* - time period used for switching to the next channel from the list

Note that the Beacon preamble is long enough to ensure detection regardless of the channel on which the Beacon packet is transmitted. The preamble duration is calculated by *(nb_channels + 1)*(tr_det_listening + ch_change)* and is greater than or equal to *prm_dur*. The whole PHY frame duration of the Beacon is equal to *bcn_dur*.

The Beacon discovery algorithm schedule is as follows:

1. Select channel 0.
2. Listen to the medium on selected channel for *tr_det_listening*
   (a) if the preamble is detected, start receiving the PHY frame for a maximum duration of *bcn_dur* time
      i. if the whole PHY frame is not received within *bcn_dur* time, stop receiving and go to bullet 3
      ii. if the whole PHY frame is received within *bcn_dur* time, then go to (b)
   (b) frame is checked against being a Beacon and fitting the Beacon criteria
      i. if the frame is accepted and the Beacon is received, then stop the algorithm
      ii. if the frame is not accepted, then go to bullet 3
   (c) if traffic is not detected, then go to bullet 3
3. Switch to the next channel within the *ch_change* time
   (a) if all supported channels are used, then go to bullet 1
   (b) if not all supported channels are used, then increment the current channel by 1 and continue from bullet 2

Figure 5.45 shows how the algorithm works.

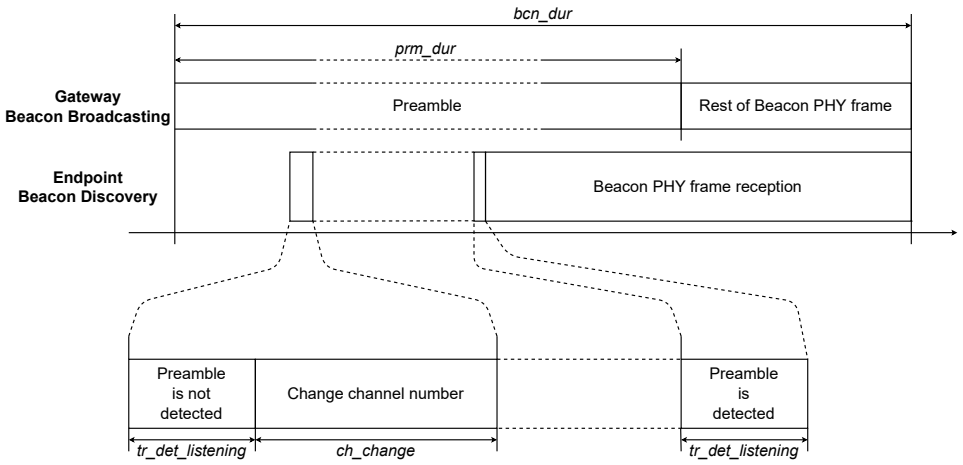


Fig. 5.45 SubG-FSK connection - Beacon Discovery

5.2.2 Synchronization maintenance

Based on the channel hopping algorithm, synchronized devices are capable of calculating the channel on which the next Beacon should be transmitted. Additionally, Beacons are broadcast every *bcn_interval* time interval so a synchronized Endpoint knows the channel and the approximate timing of the next Beacon packet. This leads to less time consumed receiving the Beacon as the synchronized device receives just the tail of the Beacon preamble transmitted by the Gateway. The Endpoint opens a downlink detection slot for each *bcn_interval* and listens for ongoing transmission for *tr_det_listening* time. When the transmission is detected, the device starts to receive the whole PHY frame and validate whether the received data is the Beacon. Figure 5.46 shows the Beacon reception process on the synchronized Endpoint.

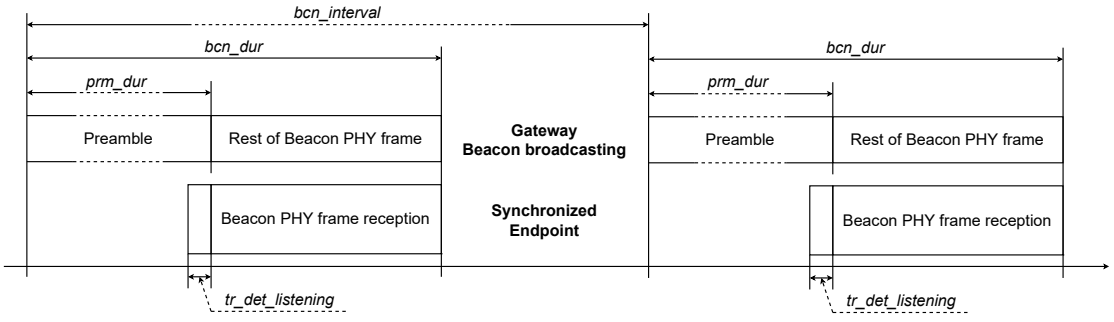


Fig. 5.46 SubG-FSK connection - Beacon reception

If a synchronized Endpoint fails to receive a Beacon, the Endpoint continues Link Layer operations and attempts to receive the next Beacon while the number of consecutive failures is less than 4. The third consecutive failure triggers the Beacon discovery procedure.

Figure 5.47 shows the Beacon timing from the Gateway, synchronized Endpoint and the device detecting the Beacon.

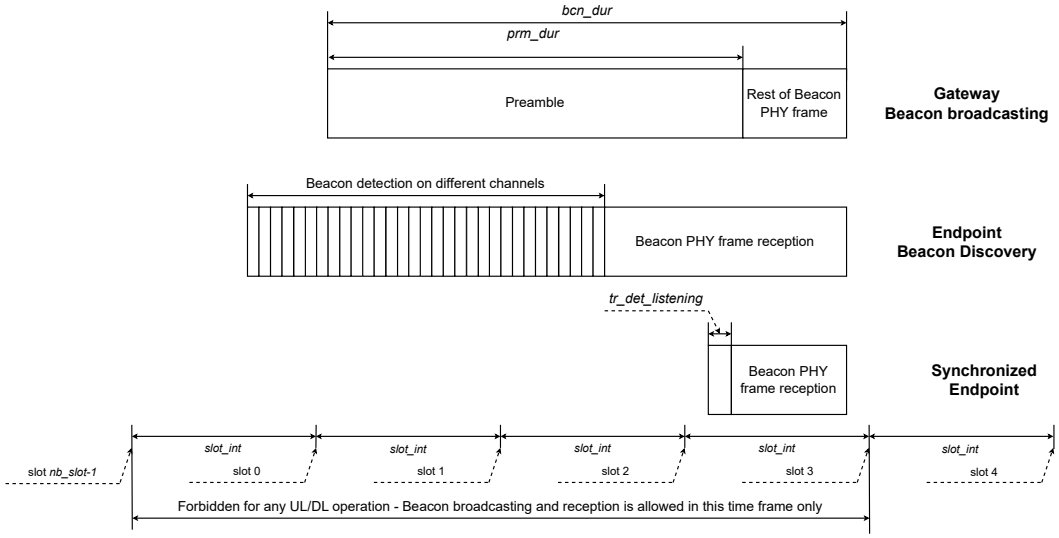


Fig. 5.47 SubG-FSK connection - Beacon operations

5.2.3 Uplink and downlink

An Endpoint in SubG-FSK mode shall send and receive data using Target Transmission Times. Target Transmission Times are separated by the time interval *slot_int*.

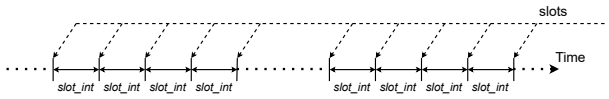


Fig. 5.48 SubG-FSK connection - Opportunities

Gateways broadcast Beacons frequently, every *bcn_interval*. The number of slots can be calculated based on the following equation:

$$nb\_slot = bcn\_interval \,/\, slot\_int$$

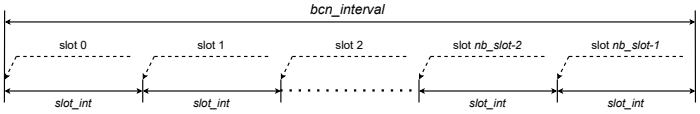


Fig. 5.49 SubG-FSK connection - Opportunities and Beacon

There is one Target Transmission Time in each slot. Target transmission times can be used for both uplink and downlink. The Gateway further specifies a subset for an uplink, i.e. from an Endpoint to a Gateways, for one or a set of nodes and similarly another subset of slots for downlink transmissions from a Gateway to an Endpoint. The following slots are not used for uplink and downlink slots: 0, 1, 2 and *nb_slot - 1*. These are reserved for the Beacon broadcasting. Hence, the index of the first and last uplink and downlink opportunities are *last_ul_dl_nb_opp_index* is 3 and *first_ul_dl_nb_opp_index* is *nb_slot - 2*.

The subset of slots assigned for each Endpoint's uplink and downlink transmissions are characterized by two parameters: offset and periodicity. The offset is the number of slots between slot 0 and the first slot that is scheduled for uplink or downlink. The periodicity is the time interval between subsequent slots that are scheduled for uplink or downlink. The maximum number of slots that may be used for uplink or downlink activity has the value *ul_dl_nb_slot*. This value can be calculated by the following equation:

ul_dl_nb_slot = *nb_slot* - 4 (number of slots that shall not be used for uplink and downlink slots)

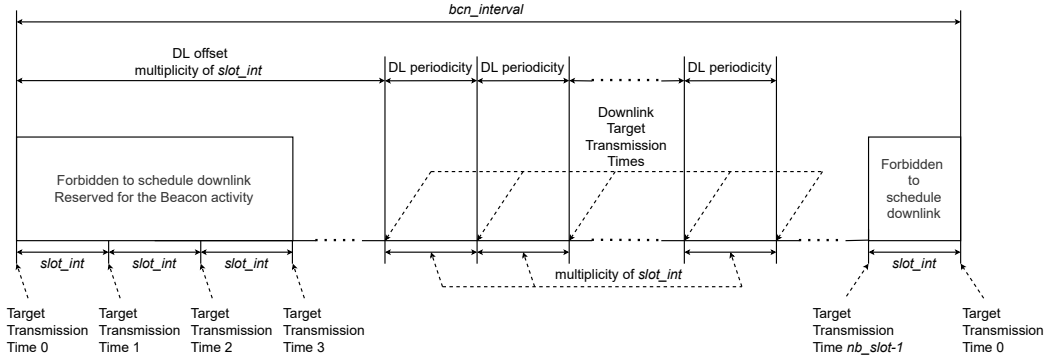


Fig. 5.50 SubG-FSK connection - DL scheduling

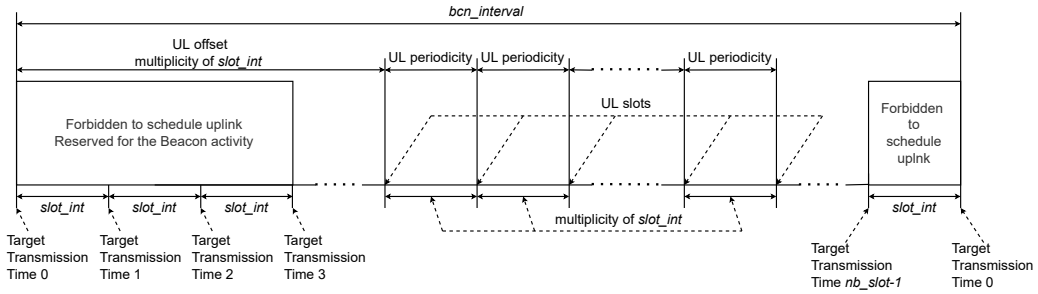


Fig. 5.51 SubG-FSK connection - UL scheduling

The device starting its transmission shall use a contention mechanism to sense whether the medium is free and uplink transmission may be performed.

The Endpoint shall sense the medium for the *tr_det_listening* duration for each Target Transmission Time and start receiving data if traffic is detected. If traffic is not detected, then the Endpoint may switch off the radio until the next scheduled uplink or downlink Target Transmission Time.

Devices working in SubG-FSK mode shall use a channel hopping mechanism for all network activities.

5.2.4 Channel hopping

The channel hopping mechanism calculates a channel number for each transmit and receive activity. The number of supported channels may differ for different regions and the number of supported channels is *nb_channels*. Transmission that starts in a selected slot on a channel shall finish on the same channel, even if the transmission spans following slots. The channel calculations for the Beacon broadcasting and Endpoint uplink or downlink transmissions are calculated separately. The result of the calculations is a hopping sequence that contains channel numbers for upcoming transmissions. The Beacon hopping sequence contains *nb_bcn_channels* elements. The uplink or downlink hopping sequence contains *nb_ul_dl_channels*. The hopping sequence calculations are described in Section 5.2.4.3 and Section 5.2.4.4. Both sequences are derived from the channel shuffler. The channel shuffler uses a pseudo random number generator. For further details, see Section 5.2.4.1.

5.2.4.1 Pseudorandom number generator

The pseudorandom number generator (PRNG) produces pseudorandom values that are used in an iterative process by the channel shuffler and linear feedback shift register (LFSR) is used for that purpose. The *lfsr_value*[*n*] is produced by the linear-feedback shift register where n = 0, 1, 2, 3, ... , *nb_channels*-1. LFSR is implemented as a feedback polynomial in the following shape:

$x^{16} + x^{14} + x^{13} + x^{11} + 1$

The feedback polynomial is used in an iterative process where output of $n-1$ step is the base for calculations in step n. The output value of the feedback polynomial can be calculated as follows:

$msb = lfsr\_value[n-1]$ **XOR**
$(lfsr\_value[n-1] \gg 2)$ **XOR**
$(lfsr\_value[n-1] \gg 3)$ **XOR**
$(lfsr\_value[n-1] \gg 5)$

$lfsr\_value[n] = (lfsr\_value[n-1] \gg 1)$ **OR** $(msb \ll 15)$

The LFSR is initialized with *seed* and *seq_id*:

1. 16 most significant bits of the *seed* are taken: *msb_seed*
2. 16 least significant bits of the *seed* are taken: *lsb_seed*
3. the pseudo-random number generator is initialized as:
   lfsr_init = *msb_seed* **XOR** *lsb_seed* **XOR** *seq_id*

The first feedback polynomial value for $n = 0$ is calculated based on the following:

$msb = lfsr\_init$ **XOR**
$(lfsr\_init \gg 2)$ **XOR**
$(lfsr\_init \gg 3)$ **XOR**
$(lfsr\_init \gg 5)$

$lfsr\_value[0] = (lfsr\_init \gg 1)$ **OR** $(msb \ll 15)$

The *lfsr_value*[*0*] shall be a non-zero value. In case the *lfsr_value*[*0*] value is zero, then 0xACE1 shall be used instead.

5.2.4.2 Channel shuffler

The channel shuffler produces a randomized sequence of *nb_channels* numbers. The channel shuffler involves a random number generator to randomize the sequence and the randomization algorithm is as follows:

1. Initialize RNG with *seed*, *seq_id*
2. Initialize the sequence of numbers with values: 0, 1, 2, 3, ..., *nb_channels*-1
3. Initialize the iteration number n with value 0.
4. Calculate random value: *rng_out_value*[*n*] = *lfsr_value*[*n*] **MOD** *nb_channels*
5. Swap numbers from the sequence pointed out by *rng_out_value*[*n*] and n indexes
6. Increase the iteration number $n = n + 1$

(a) if $n <= nb\_channels$-1 continue from bullet 4

(b) if $n > nb\_channels$-1 stop the algorithm

The schematic process of the sequence randomization is shown in Figure 5.52.

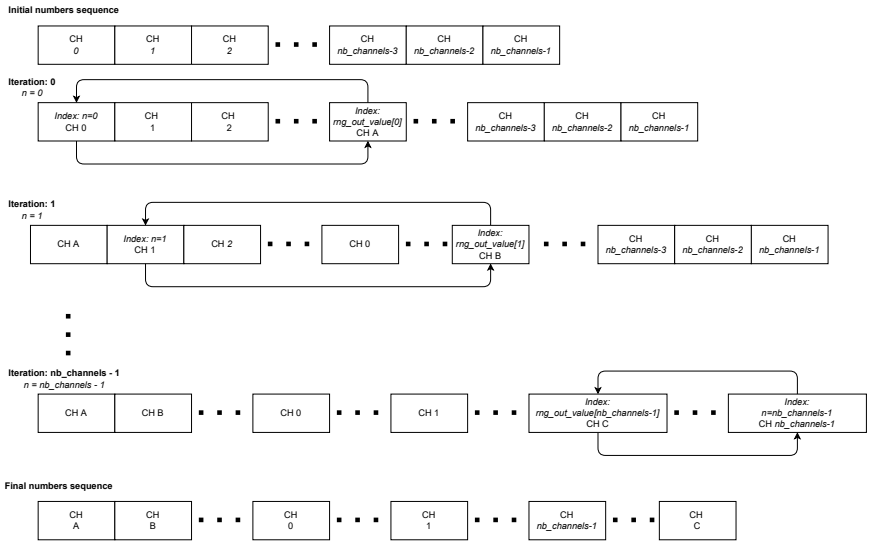


Fig. 5.52 SubG-FSK connection - Channels randomization process

5.2.4.3 Beacon hopping sequence calculation

Each subsequent Beacon is transmitted on a different channel. The channel number is taken from the channel hopping sequence that contains *nb_channels* elements. Values in the channel hopping sequence are calculated by the channel shuffler. The channel shuffler input *seed* is broadcast by the Beacon and can be found in the field *hdr_hop_seed* (see Section 5.1.3.1). The channel shuffler input *seq_id* indicates the number of channels sequence and is calculated based on two parameters:

1. *bcn_hop_sequence_num* - the value is broadcast in the Beacon (see Section 5.1.3.1)
2. *nb_channels* - number of supported channels

The equation for the *seq_id* is as follows:

seq_id = *bcn_hop_sequence_num* / *nb_channels*

The index in the numbers sequence that points out the value indicating the channel of the received Beacon is *bcn_hop_sequence_num* **MOD** *nb_channels* while the upcoming Beacon channel value is pointed out by index (*bcn_hop_sequence_num* +1) **MOD** *nb_channels*.

The Beacon channel usage example is shown in Figure 5.53. The assumptions for the figure are *nb_channels* = 69, *bcn_hop_sequence_num* starts from 0.

5.2.4.4 Uplink or downlink hopping sequence calculation

Subsequent slots that may be used for Endpoint uplink or downlink activity shall have different channels assigned (see Section 5.2.3). The channel assignment is defined by the channel hopping sequence list that contains *nb_ul_dl_channels* elements. Each element indicates the channel number is seen once in the values

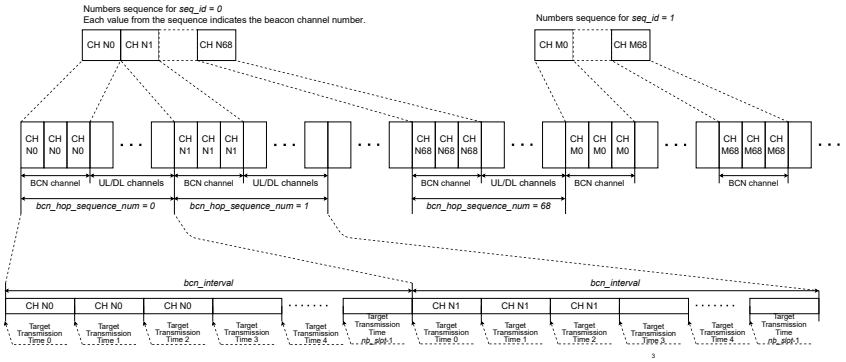


Fig. 5.53 SubG-FSK connection - Beacon channels

sequence. Additionally, the channel used by Beacon broadcasting for the current *bcn_interval* period, channels used for *nb_neighb_bcn* past and future Beacons shall not be present in the uplink or downlink hopping sequence. The value of *nb_ul_dl_channels* can be calculated using the following equation:

*nb_ul_dl_channels = nb_channels - 2*nb_neighb_bcn - 1*

The base for the uplink or downlink channels sequence calculation is the channel shuffler and the Beacon hopping sequence. The channels sequence calculation starts with the channel shuffler that has two parameters: *seed* and *seq_id*. The seed value is broadcast by the Beacon and can be found in the field *hdr_hop_seed* (see Section 5.1.3.1). The *seq_id* shall be calculated based on four parameters:

1. *bcn_hop_sequence_num* - the value is broadcasted in the Beacon
2. *nb_ul_dl_channels* - number of channels used for uplink or downlink activity
3. *nb_slot* - number of slots in *bcn_interval*
4. *slot_nb* - slot number (range is from 0 to $nb\_slot - 1$)

The equation for the *seq_id* is as follows:

$seq\_id = ((bcn\_hop\_sequence\_num * nb\_slot) + slot\_nb) \ / \ nb\_ul\_dl\_channels$

The output of the channel shuffler is the sequence of *nb_channels* values. The next step is to remove the last three channels used for broadcasting Beacons, *nb_neighb_bcn* and the next two channels that will be used for broadcasting Beacons, *nb_neighb_bcn* based on the calculated Beacon hopping sequence returned by the channel shuffler. If the calculation happens during a Beacon slot, that Beacon slot is counted with the last three channels used for broadcasting Beacons. After removing Beacon channels, the final sequence of *nb_ul_dl_channels* values is the uplink or downlink hopping sequence.

The channel value that shall be used for the uplink or downlink slot is given by *hop_seq_index* value. The channel value indicates the index in the hopping sequence that corresponds to the channel number. The index value can be calculated based on the following equation:

*hop_seq_index = ((bcn_hop_sequence_num * nb_slot) + slot_nb)* **MOD** *nb_ul_dl_channels*

An example of uplink, downlink and the Beacon channel hopping sequence usage is shown in Figure 5.54. The example has the following parameters *nb_channels* = 69, *nb_ul_dl_channels* = 64, *bcn_interval* = 10080 ms, *slot_int* = 63 ms, *nb_slot* = 160.

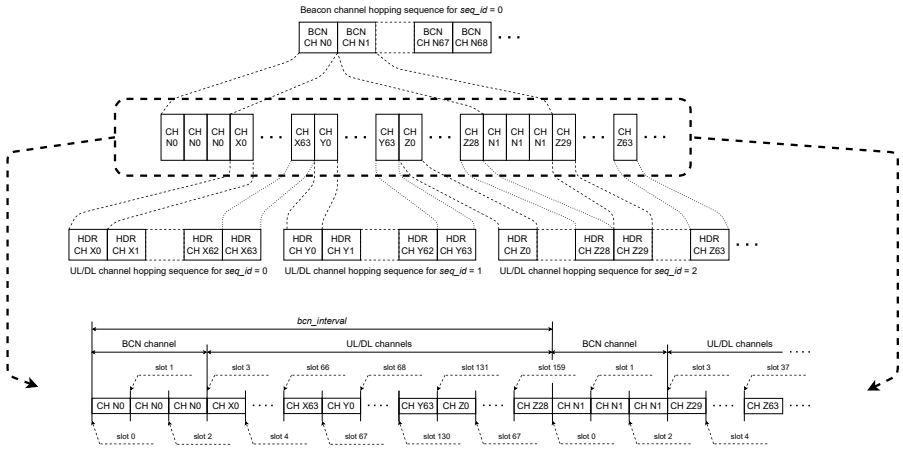


Fig. 5.54 SubG-FSK connection - Channel hopping usage

5.2.5 Contention mechanism

The role of the contention mechanism is to prevent a collision when two or more Endpoints start sending data in the same transmission slot. At a random time slightly before the Target Transmission Time, the device shall check whether the medium is free before transmission is started, and for that purpose a listen-before-transmit solution is used. The transmission activity is forbidden if the channel is busy and the device shall defer the transmission to a later Target Transmission Time. The transmission activity is allowed if the channel is free.

The check of medium availability is done in a contention window which precedes the target transmission time. The contention window duration has a value *contend_time*. The contention window is divided into 6 sub-slots. Each sub-slot has duration sub_slot_time. The number of sub-slots is defined by *sub_slot_nb* and the duration of each sub-slot is defined by *sub_slot_time*. The Endpoint has to select one sub-slot randomly. Gateways select deterministically, and always select *sub_slot_nb*-1. The selected slot is used for the detection of activity from other devices in the network. Figure 5.55 shows the timing of contention sub-slots and uplink slots.

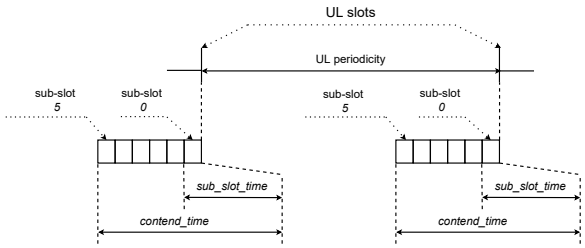


Fig. 5.55 SubG-FSK connection - Contention window

If the device does not detect any activity on the selected sub-slot, the device starts transmission. The transmission shall start in a subsequent sub-slot. All the remaining subsequent sub-slots shall be used for transmission of additional preamble symbols prepended to the Physical Layer frame (see Section 5.1.4). If sub-slot 0 is used to detect activity, additional preamble symbols shall not be added. The actual Physical Layer frame can be started at the Target Transmission time. Figure 5.56 shows when the device can start sending the data.

If the device detects activity on the selected sub-slot, then the device shall not start its transmission. The device shall select the next transmission slot randomly from the set of the next 4 subsequent uplink slots. At

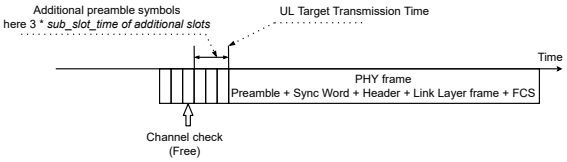


Fig. 5.56 SubG-FSK connection - Data transmission - Allowed

the next scheduled transmission slot, the channel availability check process needs to be performed again with a new random sub-slot selection. This process can be repeated up to *tr_ret_nb* times before transmission failure is concluded. Figure 5.57 shows the situation when transmission needs to be deferred to a later time.

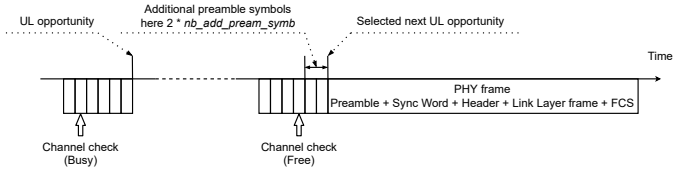


Fig. 5.57 SubG-FSK connection - Data transmission - Forbidden

5.3 SubG-CSS connection

The SubG-CSS mode allows devices to connect to the network using the ALOHA channel access protocol. The frame can be transmitted by an Endpoint when it is needed.

Section 5.2.3 describes scheduling of transmission and time dependencies between downlink and uplink.

Section 5.3.1.1 contains a detailed description of a randomization algorithm used for scheduling uplink Endpoint activities.

Each Endpoint shall communicate with the Amazon Sidewalk network to indicate its presence to the cloud and also give a Target Transmission Time to receive data. The device presence indication may be either frequent, with a specified time interval, or unscheduled, when uplink transmission is executed occasionally. Endpoints may receive data within a regulated period of time with specified intervals after uplink transmission. For more details, see Section 5.5.

5.3.1 Downlink and uplink

The device working in SubG-CSS mode may start its transmission randomly. All uplink transmissions are not synchronized. The moments in time when uplink may happen are called uplink Target Transmission Times. These events are separated by a specified time interval called uplink slot separation and its value is *ul_separation*. Downlink shall happen after uplink transmission only. The Endpoint receive is synchronized. The moments in time when a downlink may happen are called downlink Target Transmission Times. These events are separated by a specified time interval called downlink slot separation and its value is *dl_separation*. Uplink transmissions are asynchronous with respect to other devices but follow a pseudo-random pattern for each device. Figure 5.58 shows the downlink and uplink slots scheduling.

The number of uplink Target Transmission Times followed by downlink has the value *nb_ul_opp*.

The value *nb_ul_slot* has to be a power of 2

$nb\_ul\_slot = 2^i$ where i= 0, 1, 2, ...

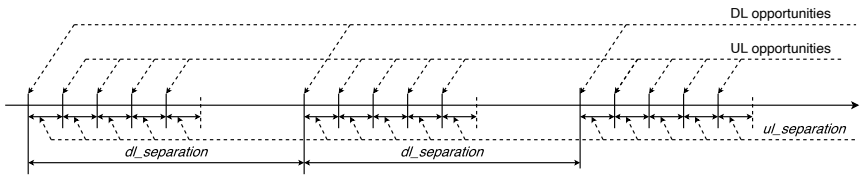


Fig. 5.58 DL and UL timing

the total amount of downlink and uplink slots *nb_slot* in *dl_separation* period is:

$$nb\_slot = nb\_ul\_opp + 1$$

and the time consumed for all *nb_slot* has to be less than *dl_separation* time:

$$nb\_slot * ul\_separation < dl\_separation$$

The number of downlink slots used for network data reception has the value *nb_dl_slot*. The number is counted from the last uplink transmission and the Endpoint does not schedule any further downlink when *nb_dl_slot* is reached. This value and *dl_separation* are configured by Connection Profile settings (see Section 5.5).

Each uplink slot has its own index *ul_slot_index* in the *dl_separation* period and possible values are as follows:

$$ul\_slot\_index = 0, 1, ..., nb\_ul\_slot - 1$$

The method of selection of an uplink slot for a transmission is based on Lehmer's random number generator and the result of the calculation is *ul_slot_index*. The index value allows calculation of the offset of an uplink slot from a downlink slot:

$$ul\_offset = (ul\_slot\_index + 1) * ul\_separation$$

Each uplink transmission is scheduled but not synchronized, so the time drift is not taken into account and the preamble size is constant.

Figure 5.59 shows the *ul_offset* and *ul_opp_index* values. Section 5.3.1.1 describes values evaluation.

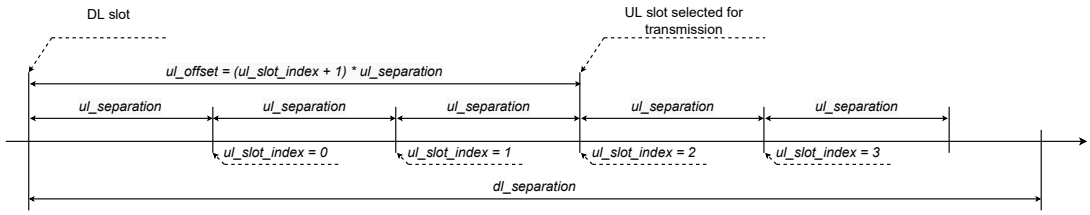


Fig. 5.59 UL opportunity selection

Downlink slots following an uplink slot are scheduled and synchronized. The synchronization between the Endpoint and the associated Gateway is based on the uplink Epoch frame. The position in time of subsequent downlink slots is based on the uplink transmission. The Endpoint calculates *ul_slot_index* of the downlink slot used for transmission and *ul_offset* from the last downlink slot preceding the transmission. The time to the downlink slot is calculated as follows:

$$dl\_offset = dl\_separation - ul\_offset$$

Figure 5.60 shows scheduling of downlink slots.

An Endpoint which has been sleeping for a long time starts its transmission in case data is available to be sent. The first uplink transmission is the base for the calculation of subsequent downlink slots scheduling. Also when additional data transfer is required, the device uses scheduled downlink slots for subsequent uplink estimation. Figure 5.61 shows such a case.

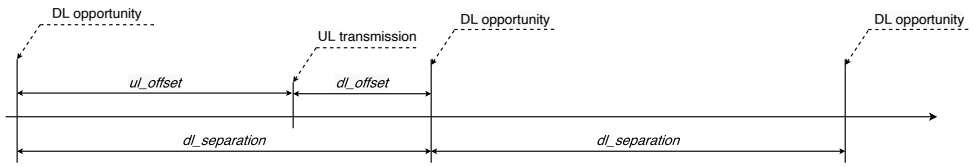


Fig. 5.60 DL opportunity estimation

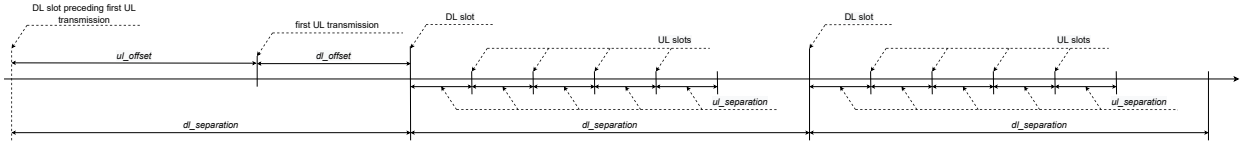


Fig. 5.61 UL and DL scheduling

5.3.1.1 Uplink offset computation algorithm

Endpoints in close proximity can have data transmitted at a similar time. This leads to a situation where uplink activity of devices can overlap. This causes an uplink collision and a packet loss due to interference, and could happen in the case of tracking devices, for example. The uplink offset computation algorithm provides a mechanism which randomizes scheduling of uplink activities. The algorithm purpose is to calculate the *ul_slot_index* value which points out an uplink slot used for transmission activity.

The algorithm input for variable calculations is *seed* and *seq_id*. The *seed* is the table containing all UUID digits of the Endpoint and the size of the table is equal to *nb_uuid* of UUID digits. Figure 5.62 shows this table.

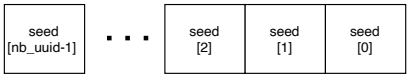


Fig. 5.62 Seed table

The *seq_id* is the Link Layer sequence number of the Epoch frame to be transmitted (see *seq_num* field in Data Frame format of Link Layer in section 5.1.3.2.4).

The core of the algorithm is Lehmer's random number generator. The base equation of the generator is:

$$x[n] = (a * x[n-1] + b)\ MOD\ m$$

where n indicates the iteration number and $x[n]$ indicates the uplink slot. Parameters m, a and b are common for all iterations. Variable m is equal to *nb_ul_slot*:

$$m = nb\_ul\_slot = nb\_slot - 1$$

Variables b and a are based on the seed and calculated based on the following equations:

$$b = (seed[nb\_uuid - 1]\ MOD\ m)\ OR\ 1$$

$$a = (seed[nb\_uuid - 3] + (5 - (seed[nb\_uuid - 3]\ MOD\ 4)))\ MOD\ m;$$

also the initial value $x[n]$ where $n = 0$ is based on seed and is calculated in the following way:

$$x[0] = seed[nb\_uuid - 2]\ MOD\ m$$

When UUID is changed, these three variables also need to be recalculated and the algorithm needs to be reinitialized.

The Endpoint needs to calculate a number of iterations *nb_it* which need to be executed for the provided *seq_id*. This is based on two variables *curr_it* and *prev_it* value. Variable *prev_it* is set to 0 during algorithm initialization:

$prev\_it = 0$

the value *curr_it* is always calculated from equation

$curr\_it = seq\_id \ MOD \ m$

as this is an MOD operation; then it could happen that

$curr\_it < prev\_it$

so in this case requires recalculation of *curr_it* in the following way:

$curr\_it = (seq\_id \ MOD \ m) + m$

the number of iterations which needs to be executed for specified *seq_id* is:

$nb\_it = curr\_it - prev\_it$

then the n value is increased by 1 up to *nb_it* in Lehmer's equation:

$n = n + nb\_it$

so the final $x[]$ value for *seq_id* is

$x[n + nb\_it]$

where n is the final value of all iterations executed till current *seq_id*. After this the *prev_it* is overwritten by *curr_it* and *curr_it* and is calculated again for the next *seq_id*.

Each new $x[n]$ value calculated for subsequent *seq_id* indicates the uplink slot index *ul_opp_index* so *ul_offset* and is calculated using the following equation:

$ul\_offset = (ul\_slot\_index + 1) * ul\_separation$

This identifies the uplink slot scheduled for transmission relative to the downlink slot.

5.4 Gateway connection modes

In the sub-GHz band, a Gateway operates in one of four modes:

1. SubG-FSK-only mode (see Figure 5.63)
2. SubG-CSS-only mode (see Figure 5.64)
3. SubG-FSK+SubG-CSS mode, also called multirate mode (see Figure 5.65)
4. Offline operation mode (see Figure 5.66)

The first three modes support different Endpoint types with communication to the Amazon Sidewalk Cloud. Gateways transfer Endpoints' packets to the cloud through the direct communication link that may be handled by different communication technologies like Wi-Fi, LAN or cellular connectivity. These Gateways operation modes are described in detail in Section 5.4.1, Section 5.4.2, and Section 5.4.3

Offline operation mode shall be used by the Gateway when a direct link to the Amazon Sidewalk Cloud is lost. The switch to this mode shall be done automatically and the Gateway behaves like a SubG-CSS Endpoint. The SubG-CSS connection is used in this case. The connectivity to the cloud is provided by a neighborhood Gateway that is called an assisting Gateway. Endpoints cannot connect to the network through the Gateway operating in this mode.

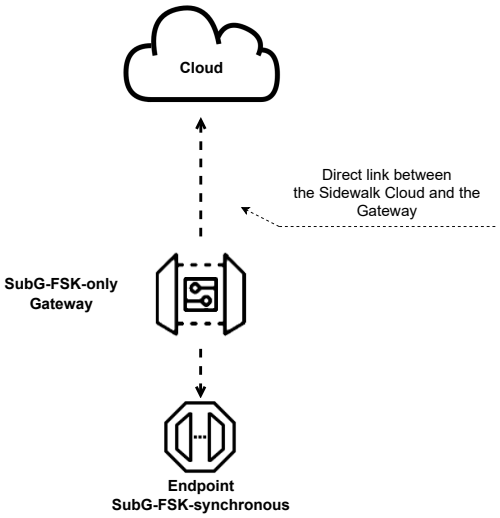


Fig. 5.63 SubG-FSK-only gateway

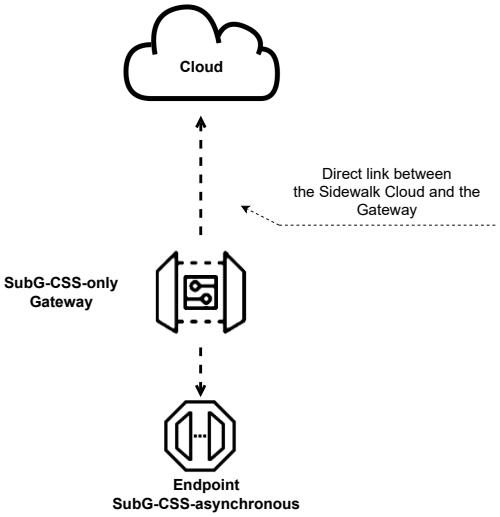


Fig. 5.64 SubG-CSS-only gateway

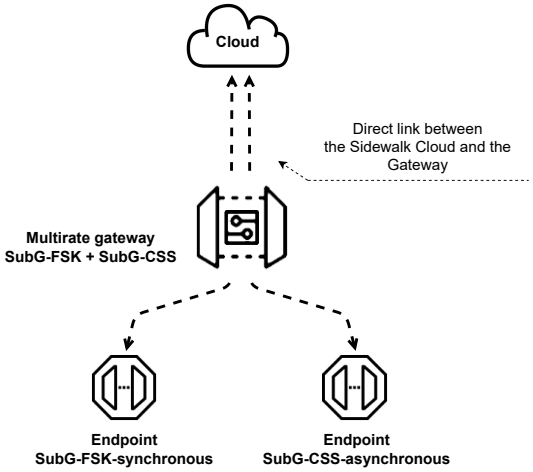


Fig. 5.65 Multirate gateway

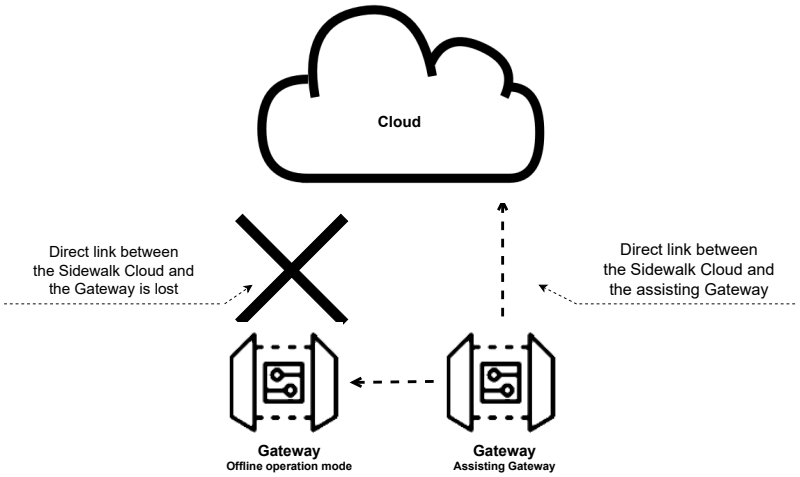


Fig. 5.66 Offline mode gateway

5.4.1 SubG-FSK-only operation mode

In this mode, the Gateway's responsibility is to send the Beacon frequently and to handle SubG-FSK Endpoints' traffic in uplink and downlink directions.

The Gateway not transmitting data shall listen to the medium to detect uplink traffic. The Gateway shall start an uplink preamble detection in slots for *hdr_listening_time* duration (see Figure 5.67). When a transmission of a preamble is detected in *hdr_listening_time* period, then reception of the uplink packet shall be started. The Gateway shall continue listening to the medium in the upcoming slot after the reception of an uplink packet is completed (see Figure 5.68). The device not detecting traffic in *hdr_listening_time* shall switch the radio to the inactive state till upcoming selection activity.

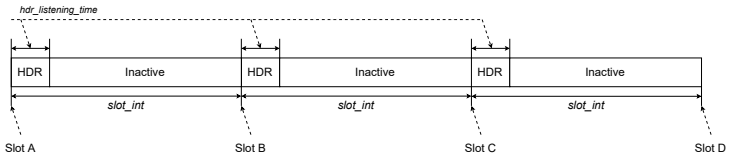


Fig. 5.67 Gateway timing - SubG-FSK-only

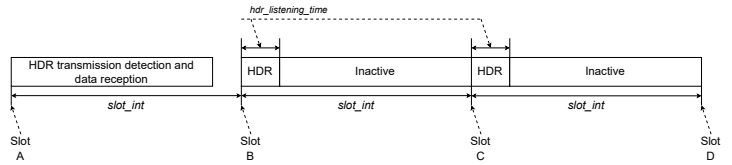


Fig. 5.68 SubG-FSK-only gateway - data reception

The downlink transmission is possible only during the instance of time when the Endpoint starts listening in the assigned downlink slots. The Gateway shall start the downlink transmission on the upcoming downlink slot. The Gateway does not check the medium availability but starts its transmission on the *sub_slot_nb*-1 detection sub-slot. This promotes the Gateway over Endpoints and gives higher priority to a downlink transmission. The mechanism is described in Section 5.2.5.

5.4.2 SubG-CSS-only operation mode

In this mode, the Gateway responsibility is to handle SubG-CSS Endpoints' traffic in the uplink and downlink directions.

The Gateway not transmitting data shall listen to the medium to detect uplink traffic. Listening to the medium is done periodically each *slot_int* for *ldr_listening_time* duration. The radio shall be inactive for the remainder of the *slot_int* period (see Figure 5.69).

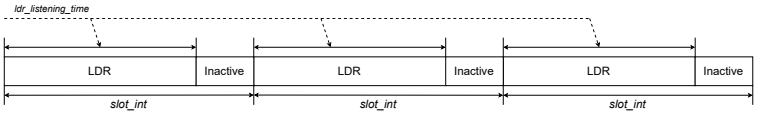


Fig. 5.69 Gateway timing - SubG-CSS-only

When an Endpoint transmission of a preamble is detected in *ldr_listening_time* period then reception of the uplink packet shall be started. The Gateway reception is spanned over a few *slot_int* (see section 5.1.4.2.4).

The Gateway shall continue listening to the medium in the upcoming slot after the reception of an uplink packet is completed (see Figure 5.70).

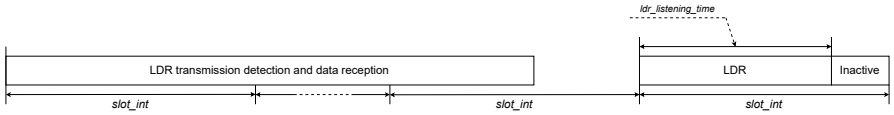


Fig. 5.70 SubG-CSS-only gateway - data reception

The SubG-CSS downlink transmission is scheduled in a synchronous way. The synchronization is based on each uplink Epoch frame (see Section 5.3). The Amazon Sidewalk Cloud stores information when the SubG-CSS Endpoint can receive data and provides these parameters to the Gateway. This is based on the Join procedure and Connection Profile settings of the Endpoint. The start of a downlink transmission may not be aligned with the listening schedule (see Figure 5.71 and Section 5.5 when downlink transmission may happen). The Gateway is also responsible for applying the correct preamble size to the PHY level based on the possible time drift determined by the crystal quality.

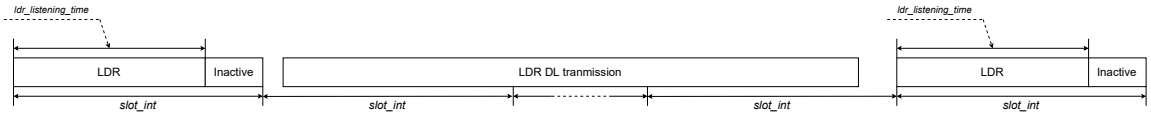


Fig. 5.71 SubG-CSS-only gateway - data transmission

5.4.3 Multirate operation mode

In this mode, the Gateway's responsibility is to send the Beacon frequently and to handle SubG-CSS and SubG-FSK Endpoints traffic in uplink and downlink directions. This Gateway mode is a combination of SubG-CSS-only and SubG-FSK-only modes.

The Gateway listens to SubG-FSK and SubG-CSS uplink traffic sequentially. The listening schedule is determined in the same way as for the SubG-FSK-only Gateway. SubG-FSK traffic detection starts in each slot for *hdr_listening_time* duration. When SubG-FSK traffic is not detected, then SubG-CSS traffic detection is started for *ldr_listening_time* duration. When SubG-CSS traffic is not detected, then the cycle is repeated (see Figure 5.72).

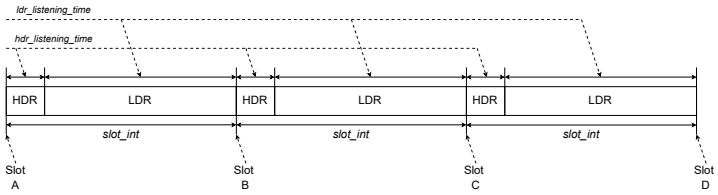


Fig. 5.72 Gateway timing - SubG-CSS and SubG-FSK

The Gateway switches between modulations till the moment a transmission is detected in one of the modulations, which triggers reception of the uplink packet. The data reception may span a few slots. When the packet reception is completed, then the Gateway shall continue to listen to the medium starting from upcoming either SubG-FSK or SubG-CSS mode (see example on Figure 5.73).

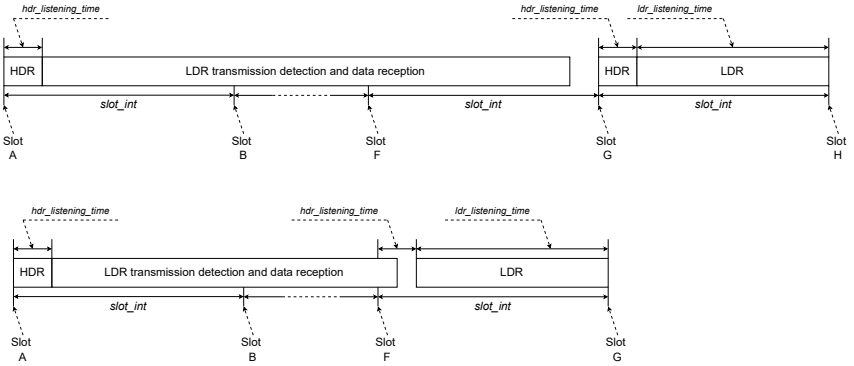


Fig. 5.73 Multirate gateway - data reception

The SubG-FSK downlink transmission in case of multirate mode has the same scheduling and rules as described in Section 5.4.1. The SubG-CSS downlink transmission rules are the same as described in Section 5.4.2. In SubG-CSS-only mode, the start of a downlink transmission is not aligned with the listening schedule and may span a few slots (see example in Figure 5.74).

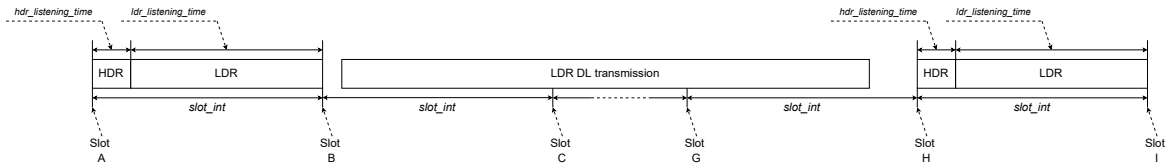


Fig. 5.74 Multirate gateway - data transmission

5.5 Amazon Sidewalk Endpoint connection modes

The Amazon Sidewalk MCU SDK is configurable to the needs of applications using it through different connection profiles it offers over both Synchronous and Asynchronous link types. Each link type has various connection profiles that further expose a set of configurable parameters. The combination of link types, profiles and parameters allow configuring the MCU-SDK to adopt the needs of the application by trading off between KPIs such as power consumption and latency. The following sections describe the connection profiles for each of the SubG-FSK and SubG-CSS link types.

5.5.1 SubG-FSK connection profiles

For SubG-FSK connection mode, the Amazon Sidewalk SDK defines two connection profiles, *Connection_profile 1* and *Connection_profile 2*. These connection profiles differ in terms of the properties of the 900 MHz link that the Endpoint forms with the Gateway, through which Amazon Sidewalk connectivity is achieved. As defined in Section 5.2.3, Endpoints operate over a certain subset of *ul_dl_nb_opp* transmission opportunities. This subset differs between *Connection_profile 1* and *Connection_profile 2*.

More explicitly, while all *Connection_profile 1* Endpoints operate over a common subset of transmission slots defined by the Gateways, each *Connection_profile 2* Endpoint defines its own subset based on its profile parameters. This configuration directly influences the trade-off between link layer latency and energy consumption. The following sections present the details of Amazon Sidewalk Cloud connection establishment and maintenance of each profile.

Dynamic switching between the profiles is not supported. However, parameter switching within profiles is supported.

5.5.1.1 Connection Profile 1

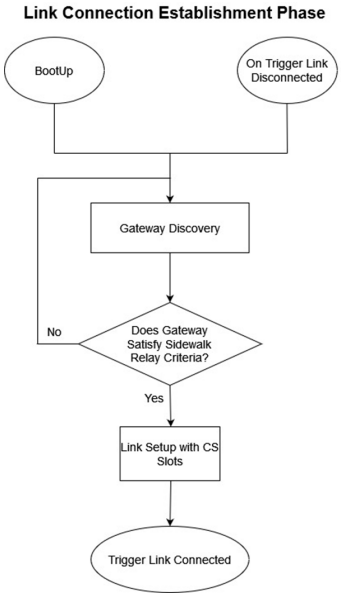
SubG-FSK connection mode of transmission defines *ul_dl_nb_opp* transmission opportunities in the interval between two consecutive Beacons numbered as presented in Section 5.2.3. The channel and timing of these opportunities are derived based on the Beacons transmitted by the Gateways. The Gateway further selects two subsets of these opportunities, namely *CS-UL* and *CS-DL*, together forming Common Slots. These two subsets of transmission opportunities are announced by the Gateway via Beacon packets and are shared by all *Connection_profile 1* Endpoints connecting to this Gateway.

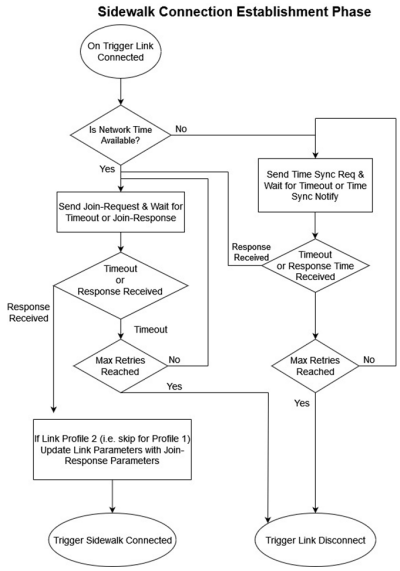
When transmitting uplink data, all *Connection_profile 1* Endpoints contend and transmit packets only over *CS-UL* slots. Similarly, the Gateway transmits the downlink data towards *Connection_profile 1* Endpoints only on *CS-DL* slots, thereby letting the Endpoint sleep for the rest of the transmission opportunities and save energy. The transmission opportunities in the *CS-UL* and *CS-DL* slots are periodic and defined by a pair of integers, namely offset and periodicity as shown in section 5.1.3.1.8. The period can be as small as one transmission opportunity and no maximum is defined either for *CS-UL* or for *CS-DL*. Each Gateway selects *CS-UL* and *CS-DL* slots to prevent sustained link quality degradation on negotiated links, i.e. Connection Profile 2. The Gateways can announce disabling of *Connection_profile 1* connectivity in the corresponding direction by choosing an empty subset. The default values used by the Gateways are available in Appendix B.1.

The operation of MCU-SDK of *Connection_profile 1* devices can be can be divided into three phases:

1. Link Connection Establishment Phase, which establishes the connection with a chosen Amazon Sidewalk Gateway;
2. Amazon Sidewalk Connection Establishment Phase, which establishes the connection with the Amazon Sidewalk Cloud; and
3. Connection Maintenance Phase, which maintains and periodically checks for connectivity with the Amazon Sidewalk Cloud.

The application is notified with a connected status after completing the first phase. As long as the protocol is kept on after the connection establishment, Amazon Sidewalk SDK operation maintains both cloud and link connectivity through its Connection Maintenance Phase. During any of these phases, if a connection failure is detected, the MCU SDK notifies the application that it is disconnected from Amazon Sidewalk, and attempts to reestablish an alternative connection starting from the link connection establishment phase. These phases are depicted in Figure 5.75.





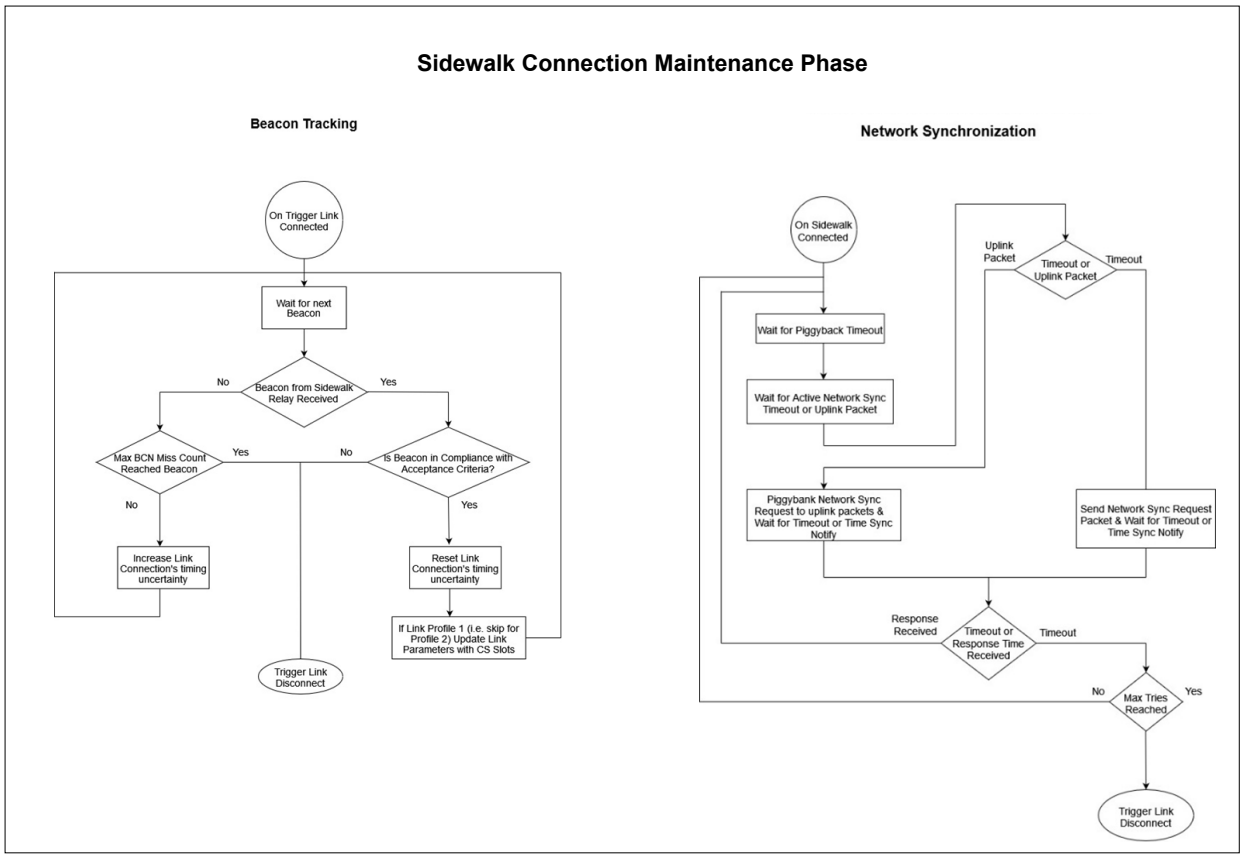


Fig. 5.75 SubG-FSK connection phases for both connection profile 1 and 2

5.5.1.1.1 Link Connection Establishment Phase

The link layer connection is initiated by the Endpoints. The Endpoints discover nearby Amazon Sidewalk Gateways through the Beacon Discovery operation as described in Section 5.2.1. The contents of each discovered Beacon is evaluated for eligibility of the Gateway sending it to serve as *Sidewalk Relay* to/from the Amazon Sidewalk Cloud by investigating the fields of the Beacon payload detailed in section 5.1.3.1.4. In order for the Gateway to be chosen as *Sidewalk Relay*:

- *roam_opt* must indicate roaming allowed,
- *CS-UL* must define at least one *CS-UL* slot, and
- *CS-DL* must define at least one *CS-DL* slot.

Discovered Beacons not satisfying all three criteria are dropped and the Beacon Discovery procedure continues as described in Section 5.2.1. With the first discovered Beacon satisfying all three requirements, each Endpoint chooses the Gateway sending the Beacon as its *Sidewalk Relay*, stops the Beacon Discovery operation, and starts following the timing and hopping pattern indicated in the Beacon, establishing 900 MHz connectivity with the chosen *Sidewalk Relay*. Any subsequent communication to/from the *Sidewalk Relay* follows the uplink and downlink packet transmission as described in Section 5.2.3 over the subsets of slots defined by *CS-UL* and *CS-DL*.

Link layer connectivity is maintained through tracking and receiving the Beacons as described in Section 5.2.2. Each subsequent Beacon reception also updates the transmission opportunities available for uplink and downlink for the Endpoints with the new *CS-UL* and *CS-DL*, respectively. On the third consecutive failure on Beacon reception, the MCU-SDK triggers disconnect operation, i.e. returning to the beginning of the Link Connection Establishment Phase in which the SDK auto triggers the Beacon Discovery operation to find a new *Sidewalk Relay*.

5.5.1.1.2 Amazon Sidewalk Connection Establishment Phase

Following link connectivity, the next phase is to establish connectivity with Amazon Sidewalk. This phase involves encrypted packet exchange with the Amazon Sidewalk Cloud following *Network Security Level* with *Sidewalk TX-ID* addressing for the Endpoint. Both of these security features require *Global Network Time* as described in Section 4.5. If the network time is not previously available or is not within acceptable bounds, prior to the Connection Establishment Phase, the Endpoint first goes through *Global Clock Synchronization* procedure to acquire/refresh network time. This time sync operation requires another pair of uplink and downlink messages and the process is covered in Section 4.2. If *Global Network Time* is already available, the Endpoint skips the *Global Clock Synchronization* and directly proceeds to the Amazon Sidewalk Connection Establishment Phase.

Figure 5.76 depicts the message pair used in this phase, namely *Join-Req Message* and *Join-Resp Message*. The messages are opaque to the *Sidewalk Relay* due to the *Network Security Level* encryption. The purpose of encrypted message exchange in the Amazon Sidewalk Connection Establishment Phase is to validate connectivity with the Amazon Sidewalk Cloud and to establish the routing of the downlink packets from the application server through the Amazon Sidewalk Gateway chosen as *Sidewalk Relay*. This process is repeated after each completion of the Link Connection Establishment Phase as well as periodically thereafter as long as the SubG-FSK link connection with the *Sidewalk Relay* is maintained. This initial packet exchange is called Network Join, while the periodic checks are referred to as Network Synchronization and are described in Section 5.6.

The contents of *Join-Req Message* and *Join-Resp Message* are listed in Table 3.1 and Table 3.2, and detailed below in section 3.2.2.2.3. Following *Join-Req Message*, the MCU-SDK waits for a *Join-Resp Message* for a random duration uniformly chosen in the range [20 s, 30 s]. In a case where no response is received within this interval, MCU-SDK sends another *Join-Req Message* followed by a new random interval chosen

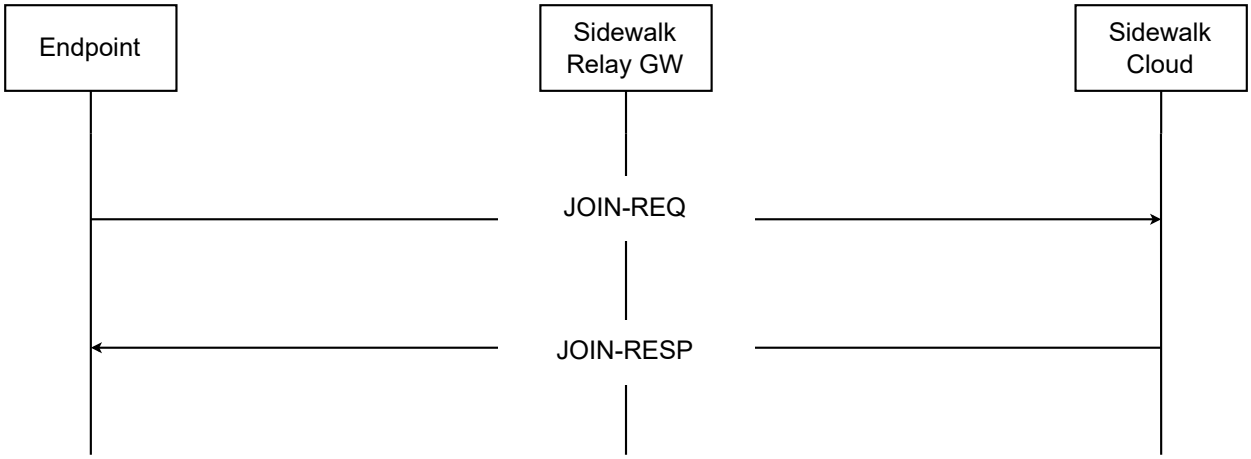


Fig. 5.76 Profile 1 join msg exchange diagram

from the same range. After a maximum of three retries, the Endpoint triggers the disconnect operation, i.e returning to the beginning of the Link Connection Establishment Phase in which the SDK auto triggers Beacon Discovery to find a new *Sidewalk Relay*. Upon the successful reception of the *Join-Resp Message* the status field inside the the command payload is investigated. A negative status acts as a disconnect trigger returning the Endpoint to the beginning of the Link Connection Establishment Phase.

Finally, the reception of *Join-Resp Message* with a positive status indicator completes the Amazon Sidewalk Connection Establishment Phase, after which the Amazon Sidewalk Connection Maintenance Phase starts. The Amazon Sidewalk SDK notifies the application that Amazon Sidewalk connectivity is established. In this state, the data generated by the application are served in an FIFO manner following the link characteristics described in section 5.5.1.1.1. All further data communications use network layer encryption between the Amazon Sidewalk Cloud and the Endpoint in addition to any application level encryption over ASL frames as described in Section 4.7. In the Amazon Sidewalk Connection Maintenance Phase, alongside the serving of application data, the SDK maintains Amazon Sidewalk Connectivity as described in Section 5.5.1.3.

5.5.1.2 Connection Profile 2

The choice of wake-up frequency of the Endpoint is a trade-off between the expected downlink delay and the power draw on the Endpoint. A frequent wake-up interval drives up the energy consumption of the Endpoint while reducing the latency that the packet waits before being transmitted over the SubG-FSK link. Conversely, an infrequent wake-up interval allows energy savings via longer sleep durations while increasing the Link Layer delay in the downlink.

Amazon Sidewalk SDK exposes a configurable parameter, namely *link_rx_int* for *Connection_profile 2* that allows arbitration of this trade-off based on the application requirements. Applications selecting this profile additionally configure *link_rx_int* parameter that defines the desired time interval between consecutive wake-up instances. The range of values defined for this parameter is listed in Table 5.1.

The operation of MCU-SDK of *Connection_profile 2* devices can be investigated in three phases:

- initial Link Connection Establishment Phase, which establishes the initial connection to be used for the control of traffic with a chosen Amazon Sidewalk Gateway;
- Amazon Sidewalk Connection Establishment Phase and Link Connection Update Phase, which establish the connection with the Amazon Sidewalk Cloud and update the SubG-FSK link properties based on *link_rx_int* configuration; and

Profile	Parameter Name	Parameter Description	Value Range
Profile 1	-	-	-
Profile 2	*link_rx_int*	The requested period of downlink opportunities	1 Slot (63 ms), 5 Slots (315 ms), 10 Slots (630 ms), 15 Slots (945 ms), 40 Slots (2520 ms), 50 Slots (3150 ms), 80 Slots (5040 ms)

Table 5.1 Synchronous Connection Mode Link Profile Parameters

- Connection Maintenance Phase, which maintains and periodically checks for connectivity with the Amazon Sidewalk Cloud.

The flow diagram of these phases is depicted in Figure 5.75.

5.5.1.2.1 Link Connection Establishment Phase

The Endpoint operation starts at a point identical to that of *Connection_profile 1*, going through the Link Connection Establishment Phase to discover and select the first Gateway as the *Sidewalk Relay* matching the criteria described in section 5.5.1.1.1. During this phase, the SubG-FSK link characteristics follow *"Common Slots"* rather than the configuration of the Endpoint. More explicitly, during this phase, uplink messages are sent over the *CS-UL* slots and the Endpoint wakes up on the *CS-DL* slots to receive the responses regardless of the *link_rx_int* setting configured.

5.5.1.2.2 Amazon Sidewalk Connection and Link Connection Update

The initial link formed with the Gateway has characteristics with identical with those of *Connection_profile 1* Endpoints. The Endpoint uses this initial link during the Amazon Sidewalk Connection Establishment Phase. However, unlike *Connection_profile 1* nodes, *Connection_profile 2* update the link characteristics to match their configuration with the completion of the Amazon Sidewalk Connection Establishment Phase.

Amazon Sidewalk Connection Establishment involves an ECDHE Key Exchange with the Amazon Sidewalk Cloud following the Network Security level that requires the Global Network Time as described in Section 4.5. If the network time is not available or is not within acceptable bounds prior to the Connection Establishment Phase, the Endpoint first goes through the Global Clock Synchronization procedure to acquire/refresh network time. This time sync operation requires another pair of uplink and downlink messages as described in Section 4.2. If the Global Network Time is already available, the Endpoint skips the time sync process and directly proceeds to the Amazon Sidewalk Connection Establishment Phase.

The Amazon Sidewalk Connection Establishment Phase is depicted in Figure 5.77 and involves three message exchanges, namely *Join-Req Message*, *Auth Response Message*, *Join-Resp Message*. The message contents and encryption levels of these messages are tabulated in Appendix A.1.

The Amazon Sidewalk Connection Establishment Phase is initiated by the Endpoint by sending a *Join-Req Message* that includes *Connection_profile 2* setting configured by the Endpoint. The contents of the *Join-Req Message* is not visible to the *Sidewalk Relay* and is forwarded to the Amazon Sidewalk Cloud. The Amazon Sidewalk Cloud decrypts the message and in response sends a *Auth Response Message* back to the *Sidewalk Relay*. Semantically, *Auth Response Message* notifies the Gateway that the Endpoint identified with the

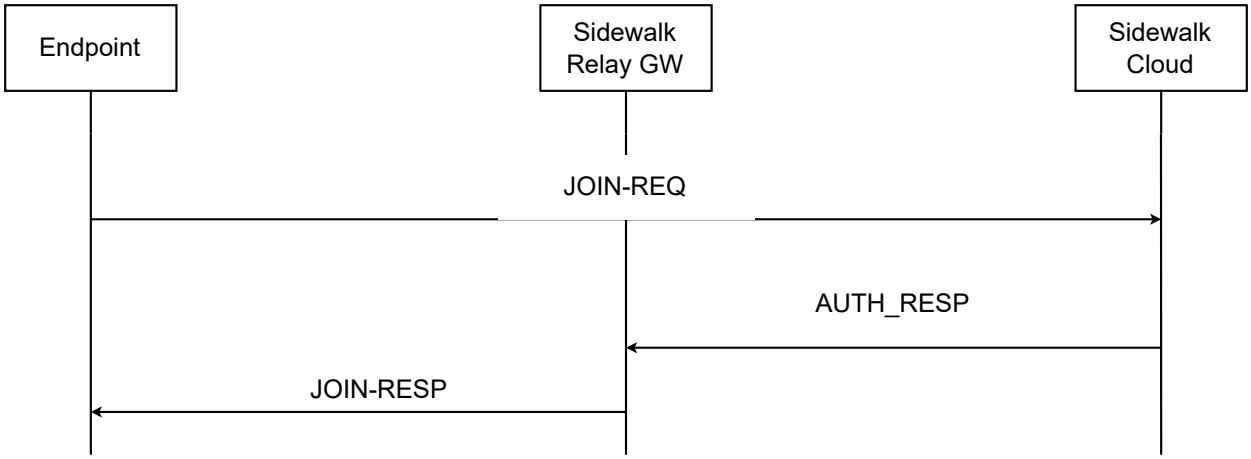


Fig. 5.77 Profile 2 join msg exchange diagram

Sidewalk TX-ID requests a *Connection_profile 2* connection with the Gateway has a downlink period of *link_rx_int*. *Auth Response Message* encapsulates the contents of the *Join-Resp Message* as well as a limited duration key *Join Response Key* that is derived from the Endpoint's credentials registered in the cloud. Based on the encapsulated contents of the *Join-Resp Message*, the Gateway runs an algorithm described in section section 5.5.1.2.2 to make a local decision to accept or reject the incoming request. If the request is accepted, a downlink schedule, namely the schedule indicating the set of slots the Endpoint should wake up to expect downlink packets, is assigned for this Endpoint. The details of the algorithm for selecting the downlink schedule at the Gateway side are described in section 5.5.1.2.2. The resulting decision of the Gateway is conveyed to the Endpoint via a *Join-Resp Message* sent over the SubG-FSK link following the *CS-DL* slots. This message is encrypted with the key received in the *Auth Response Message*. As described in Section 4.5, this key has limited usage duration and is only used by the Endpoint to receive *Join-Req Message*. Any other message type using this key type is rejected by the Endpoint. All other communications on the uplink and downlink follow the Network Security model providing encryption between the Amazon Sidewalk Cloud and the Endpoint, hiding the message contents and types from the *Sidewalk Relay*. Similarly, if the Join Request is rejected by the Gateway, a *Join-Resp Message* with status code content indicating this rejection is sent to the Endpoint with the same encryption and link schedule.

Upon reception of a *Join-Resp Message* indicating that the connection is accepted, the SubG-FSK link characteristics between the Gateway and the Endpoint are updated with the schedules in its contents. The *Join-Resp Message* contents indicate two schedules: (i) a schedule indicating the set of slots the Endpoint is allowed to use on its uplink, and (ii) a schedule indicating the set of slots the Endpoint should wake up to expect downlink packets. While the assigned schedules are dependent on the policy implemented in the Gateways, as of the GW-SDK 1.13, the chosen policy is

- On uplink allows all *ul_dl_nb_opp* slots between beacons;
- On downlink allows a schedule with a period equal to *link_rx_int* configured by the Endpoint and a random offset as detailed in section 5.5.1.2.2.

Hence, after successful joining, the application layer connection status is updated to "connected" state, and the Endpoint starts using all *ul_dl_nb_opp* slots between Beacons for its uplink transmissions, and starts following the wake-up schedule matching the desired *link_rx_int* assigned by the Gateway. The Gateway stores the previously assigned downlink schedule for the Endpoint and routes the downlink packets received from the cloud considering the slots matching the wake-up schedule of the Endpoint.

The Gateway preserves allocation through *Sidewalk TX-ID* rotations by the Endpoint. The downlink messages from the Amazon Sidewalk Cloud are forwarded to the Endpoint following the assigned DL schedule even after a *Sidewalk TX-ID* rotation and even if no prior message from the Endpoint with the given TXID

is received. The saved allocation is removed from the Gateway database if no uplink message is received from the Endpoint for longer than 1 h.

Schedule Assignment and Maintenance at Gateways

The incoming *Auth Response Message* trigger SLOT-ASSIGN-REQ input of the algorithm depicted in Figure 5.78. The Gateway first checks if the incoming TXID already has an existing assignment with the same value of the *link_rx_int* parameter. In the case of a match, the previously assigned schedule is reassigned to the Endpoint. If there is no existing assignment or if the new request parameter is different from the previous one, then an incoming request is classified as a new request.

For the request classified as new, the Gateway next checks if the assignment request is made for an Endpoint registered in the same account with the Gateway. Both of these checks are done via querying the Amazon Sidewalk Cloud with the incoming TXID. The Endpoints registered on the same account have privileges and are assigned a schedule without further checks.

If the request is made for an Endpoint registered on an account different than the account the Gateway is registered on, the Gateway further checks the total number of allocations actively maintained. If the total is below the threshold, K, a Gateway configured for the incoming request is accepted and a schedule is generated. Otherwise, the incoming request is rejected.

To accept the new request, the Gateway creates a periodic downlink schedule for the incoming Endpoint. The period of the schedule is set as *link_rx_int* configuration of the Endpoint that is sent with *Join-Req Message* and relayed in *Auth Response Message*.

The offset is selected randomly. The random selection creates a subset of opportunities numbered in the range: [*last_ul_dl_nb_opp_index*, *first_ul_dl_nb_opp_index*] as defined in Section 5.2.3.

The random offset is selected to create a subset with the largest number of elements possible. The offset is selected randomly in the range: [*last_ul_dl_nb_opp_index*, (*last_ul_dl_nb_opp_index*+(*first_ul_dl_nb_opp_index* - *last_ul_dl_nb_opp_index*) % *link_rx_int*))].

The assignments are preserved by the Gateway and matched with the upcoming R-TXID. The Gateway preserves the reservation unless no uplink packet from the device is received for a period of 1 h.

5.5.1.3 Link Connection Maintenance Phase

For both *Connection_profile 1* and *Connection_profile 2*, MCU SDK maintains both the 900MHz connectivity and the Amazon Sidewalk Cloud connectivity through two mechanisms, (i) link connection loss detection mechanism and (ii) Amazon Sidewalk connection loss detection mechanism. Connection loss detection through either mechanism triggers switching to the disconnected state, in which case the Amazon Sidewalk SDK automatically attempts to reestablish Amazon Sidewalk connectivity starting from the Initial Link Connection Establishment phase as presented in sections 5.5.1.1.1 and 5.5.1.2.1.

5.5.1.3.1 Link Connection Loss Detection

The first connection loss detection mechanism depends on periodically received beacons from the Gateway as described in Section 5.2.2. Three consecutive failures on Beacon reception indicate a connectivity loss.

5.5.1.3.2 Amazon Sidewalk Connection Loss Detection

The behavior of the network sync, including the frequency of message exchanges and piggybacking, is identical for both profiles and is described in Section 5.6. Similar to application layer-generated data packets, the network synchronization messages, whether piggybacked to the data or sent as stand-alone packets, are sent following the updated link properties achieved at the end of the Amazon Sidewalk Connection Establishment Phase. In other words, *Connection_profile 1* Endpoints use *"Common Slots"* link characteristics.

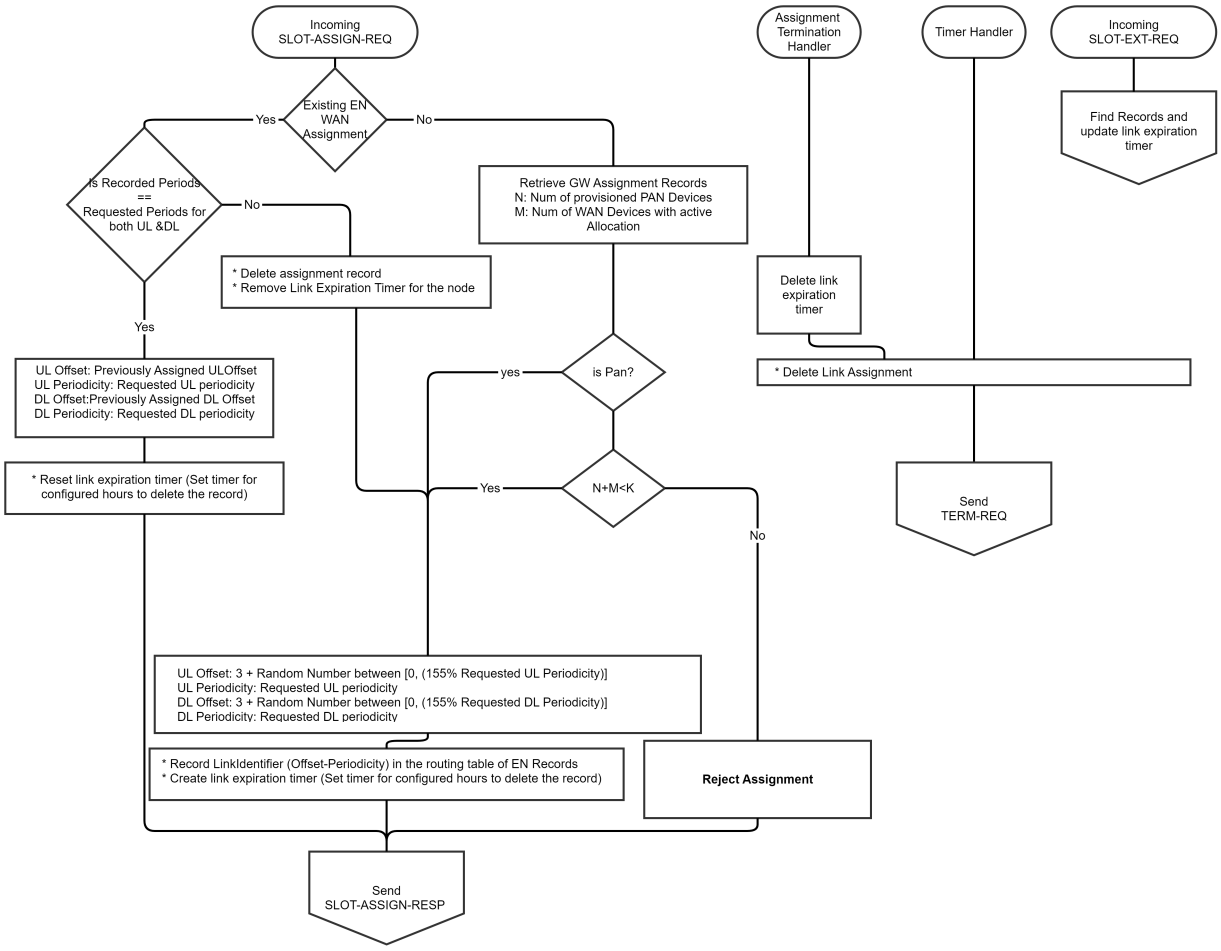


Fig. 5.78 SubG-FSK-WAN Connection Profile 2 Schedule Assignment Algorithm

Connection_profile 2 Endpoints use the link characteristics updated in the Amazon Sidewalk Connection Establishment Phase. The updated link characteristics on uplink allow the use of any slots among *ul_dl_nb_opp* slots for reduced link layer latency, and on the downlink limit the alternatives to the assigned schedule, whose period is chosen by the Endpoint's configuration arbitrating the energy consumption latency tradeoff. Details of network synchronization are presented in Section 5.6.

5.5.2 Asynchronous access mode profiles

Asynchronous Connection profiles' focus is on downlink transmission configuration. This impacts Endpoint availability in the network and amount of downlink data that may be sent from the Amazon Sidewalk Cloud. Uplink transmission is not regulated by Connection Profiles A and B, and both have the same behavior. The following connection profiles are defined for asynchronous connection mode:

1. Profile A: This profile maximizes the device radio interface inactivity time, making it the most power-efficient profile in this connection mode. The downlink latency may be significant as downlink may only happen after uplink. The number of downlink opportunities is limited by *dl_opp_nb* (see Connection Profiles Parameters below). This value indicates the Endpoint availability period in the network for the Amazon Sidewalk Cloud. When the last downlink opportunity is reached, the reception operation on an Endpoint shall be suspended till the next uplink transmission. The profile is suitable for uplink-oriented use cases, where the device application can tolerate relatively longer delay in downlink and

tolerate some degree of uncertainty on the Endpoint's downlink availability.

2. Profile B: This profile has greater power consumption than Connection Profile A. The DL latency is reduced compared to Profile A and may be estimated. The Profile may handle larger amounts of data in the downlink direction.

Connection profiles in SubG-CSS mode are defined by two parameters:

1. Number of downlink opportunities - *dl_opp_nb* - the parameter indicates how many downlink opportunities are available after the last uplink transmission.

2. Downlink opportunities separation - *dl_separation* - the parameter indicates the time between two consecutive downlink opportunities.

5.5.2.1 Profile A Configuration

Profile A parameters:

1. Number of downlink opportunities - *dl_opp_nb* - the profile has limited downlink opportunities. The number of downlink opportunities shall be counted after each uplink transmission. The counter shall be reset after each uplink.

2. Downlink opportunities separation - *dl_separation* - the value is constant and is not configurable.

Parameter	Value	Note
dl_opp_nb	5, 10, 15, 20	
dl_separation	5s	constant not configurable

Table 5.2 Profile A parameters

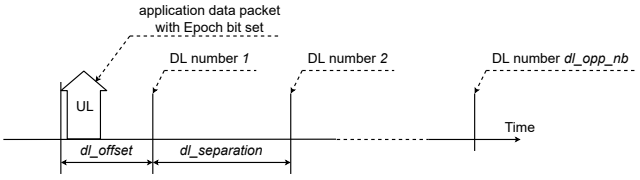


Fig. 5.79 Profile A - UL of synchronization packet and DL schedule

5.5.2.2 Profile B Configuration

The Profile B parameters:

1. Number of downlink opportunities - *dl_opp_nb* - the profile has infinite downlink opportunities which are scheduled continuously with downlink opportunity separation.

2. Downlink opportunities separation - *dl_separation* - the value is constant and is not configurable.

Parameter	Value	Note
dl_opp_nb	infinite	
dl_separation	5s	constant not configurable

Table 5.3 Profile B parameters

This profile has infinite downlink opportunities. The downlink transmission shall be synchronized based on uplink synchronization packets. Only selected uplink packets shall be used for downlink synchronization. These are sent in the uplink direction no later than around *ul_sync_interval* since the last uplink packet used for downlink synchronization but no earlier than *ul_sync_guard_interval*. The procedure for selecting an uplink packet used for synchronization defines two cases:

1. when a user application data packet is being sent within *ul_sync_guard_interval* then this packet shall have Epoch bit set (see Figure 5.80)

2. when a user application data packet is not planned to be sent within *ul_sync_guard_interval* then Join Request message with Epoch bit set shall be sent after *ul_sync_interval* time (see Figure 5.81)

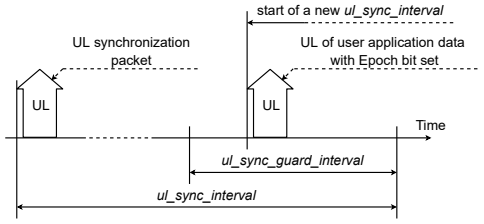


Fig. 5.80 Profile B - DL synchronization - user data in UL

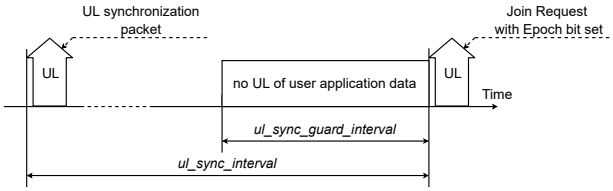


Fig. 5.81 Profile B - DL synchronization - no user data in UL

5.6 Connection maintenance with network synchronization

Endpoints in a WAN network are not always connected to a Gateway that belongs to the same account. Endpoints may roam from one Gateway to another Gateway or lose the connection to it, and thus to the cloud. Therefore, Endpoints in the WAN network keep monitoring the connection with the cloud by sending synchronization requests to the cloud either periodically or sporadically in specific states. There are three types of synchronization requests, included in two application layer commands for network synchronization,

1. Time management command described in Section 3.2.2.1.

2. JOIN command described in Section 3.2.2.2.

3. In the specified piggybacking state, uplink data is piggybacked with synchronization request TLV, described in network diagnostic data Format ID in section 5.1.2.2.3. When the cloud receives uplink data with this format ID, it replies JOIN command with type "Notify" to Endpoints as a synchronization response. This process is described in Section 3.2.2.2.

In any state, when Endpoints receive responses to the above commands, Endpoints will be set as synchronized for a certain period. See Figure 5.82 - an Endpoint sends a synchronization request and a synchronization response is received. In this way, the Endpoints are able to know the status of connectivity with the network. This also ensures Endpoints are updating the routing table in the cloud, so that the cloud can deliver the downlink packet to the correct Gateway. When the network is detected as not synchronized, Endpoints working in SubG-FSK connection mode will trigger Beacon discovery, looking for a new Gateway to which a synchronization request can be sent. For Endpoints working in SubG-CSS connection mode, uplink might still be connected. Hence, Endpoints will stay in the piggybacking state, any uplink data in this state will be synchronization request, receiving a synchronization response can bring device back to synchronized. Based on the Endpoint's connection profile setting, the behavior of network synchronization will be slightly

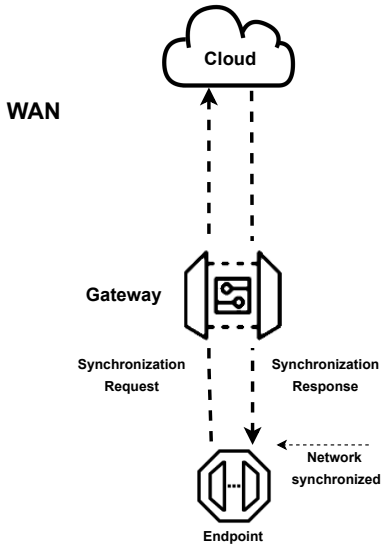


Fig. 5.82 Network synchronization

different. For further information, see the following subsections.

5.6.1 Connection Profiles 1 and 2

The behavior of time synchronization is the same for Connection Profiles 1 and 2. Periodic synchronization is activated for them. Endpoints are set as synchronized for 840 s when receiving a synchronization response and enter the piggybacking state at the halfway point of the synchronized period, 420 s. If no synchronization responses are received within the synchronized period, JOIN command described in Section 3.2.2.2 is sent due to activation of periodic synchronization. Endpoints trigger Beacon discovery autonomously when they have failed to synchronize with the network. See Figure 5.83.

Endpoints may leave the piggybacking state earlier and are set as synchronized for another 840 s when receiving a synchronization response in the piggybacking state. See Figure 5.84.

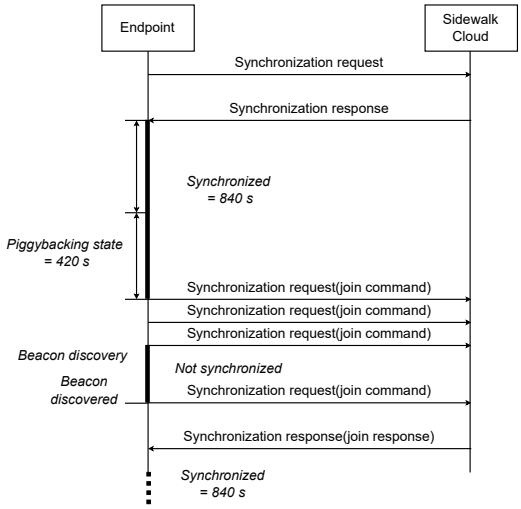


Fig. 5.83 Profile 1 and 2 network synchronization

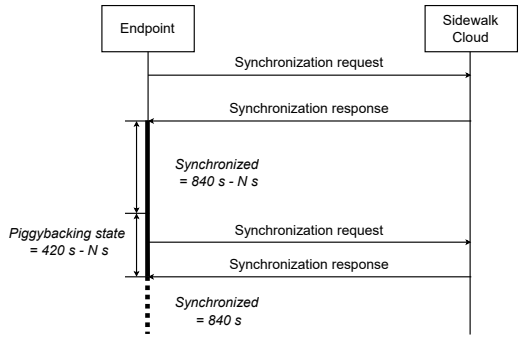


Fig. 5.84 Profile 1 and 2 synchronized in the piggybacking state

5.6.2 Connection Profile A

To save energy, there is no periodic synchronization required for Profile A. Most of the time, Endpoints stay in the piggybacking state. When a synchronization response is received, Endpoints are set as synchronized for a period of 300 s. Endpoints enter the piggybacking state again at the halfway point of the synchronized period, 150 s. If no synchronization responses are received in the synchronized period, Endpoints are set as not synchronized and remain in the piggybacking state. See Figure 5.85.

Similar to Profiles 1 and 2, Endpoints leave the piggybacking state earlier and are set as synchronized for another 300 s when receiving a synchronization response in the piggybacking state.

5.6.3 Connection Profile B

Periodic synchronization is activated for Profile B. As in Profile A, Endpoints are set as synchronized for 300 s when receiving a synchronization response and enter the piggybacking state at the halfway point of the synchronized period, 150 s. If no synchronization responses are received within the synchronized period, JOIN command described in Section 3.2.2.2 is sent due to the activation of periodic synchronization.

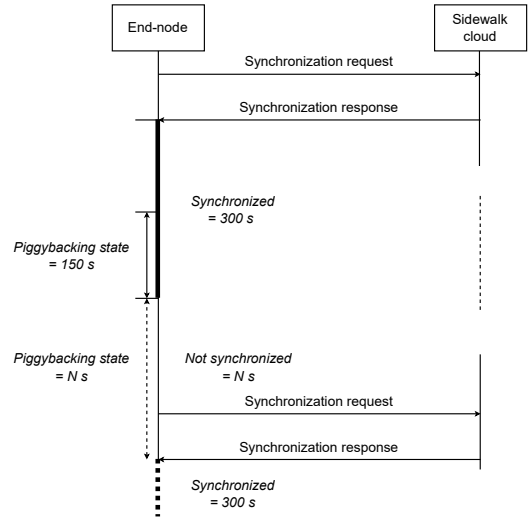


Fig. 5.85 Profile A network synchronization

On failure of the first periodic synchronization, the Endpoints are set as not synchronized and periodic synchronization will be retried three times, then Endpoints stop it and stay in the piggybacking state. See Figure 5.86.

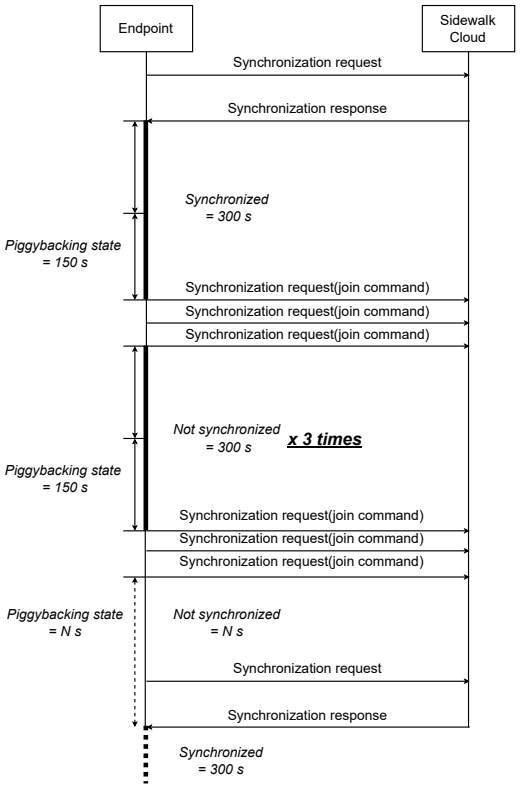


Fig. 5.86 Profile B network synchronization

Same as Profile A, Endpoints leave the piggybacking state earlier and are set as synchronized for another 300 s when receiving a synchronization response in the piggybacking state.

5.6.4 Connection Profile C

Connection Profile C is not specified in this version of the Amazon Sidewalk specification.

5.6.5 Connection Profile D - Offline operation mode

The offline Gateway operation, Connection Profile D, is also called "Distress Gateway" mode. The Gateway that loses the direct internet connectivity to the cloud shall automatically switch to Distress Gateway mode. The Gateway shall establish Amazon Sidewalk Cloud connectivity through another Gateway using LoRa modulation. Gateways that provide connection to the Distress Gateway are called assisting Gateways. When the link is recovered then the Gateway switches to a normal operation state automatically.

During the distress mode of operation, the Distress Gateway may send Beacons periodically but this is dependent on the initial operation mode:

1. multirate mode - the Gateway sends a Beacon periodically;
2. SubG-FSK-only mode - the Gateway sends a Beacon periodically;
3. SubG-CSS-only mode - the Gateway does not send a Beacon periodically.

The Gateway notifies the cloud that the link is lost using a distress message. The message is sent frequently with different time intervals between two consecutive messages. The first interval has minimum value *min_dis_mess_int* and increases exponentially for subsequent distress messages up to *max_dis_mess_int*. When an ACK is received from the cloud, the Gateway continues sending the message with the maximum defined interval *max_dis_mess_int*. When ACK is not received and the maximum defined time interval between two consecutive messages is reached then the Gateway shall reset the interval to *min_dis_mess_int* and repeat the exponential increase of the interval between distress messages up to *max_dis_mess_int*.

The Distress Gateway has two ways to communicate with the cloud:

1. common channel - this shall be used for sporadic message exchange.
2. dedicated link over separate channel - this method uses a different channel from a common one for bulky data transfer - e.g. file transfer. The Gateway shall start distress mode using the common channel and may optionally switch to a dedicated channel with a message from the Amazon Sidewalk Cloud over the common channel. The channel selection and criteria for switching to a dedicated channel is controlled by the cloud. The assisting Gateway which handles the connection over a dedicated link cannot support traffic from other SubG-CSS devices but is still able to support SubG-FSK Endpoints.

The transmission between two Gateways, the Distress Gateway and the assisting Gateway, is fully asynchronous in both uplink and downlink directions. This is the only difference when compared to the communication scheme between an SubG-CSS Endpoint and a Gateway where downlink is synchronized based on the uplink Epoch frame. The Epoch frames are not used in distress mode communication between Gateways. Additionally, a long preamble is used in both directions for downlink and uplink on the PHY Layer.

5.7 Sub-GHz physical layer modulations

The sub-GHz ISM band has two physical layer modulation techniques: CSS and GFSK. The modulation process consists of two stages: digital modulation and analog modulation. During digital modulation, the data payload (link layer frame) is attached to preamble and the CRC is placed at the end. The frame layout

is defined by IEEE in 802.15.4 [1]. For more information, see Section 5.1.4. The analog part of modulation is applied afterwards (CSS or FSK respectively). During this process, data bits are assigned to symbols according to a modulation scheme. The result is analog signal in baseband. For more information on CSS modulation, see Section 5.7.2; for more information on FSK modulation, see Section 5.7.3.

The next step is to upconvert the baseband (BB) signal to either the working RF channel (in case of single channel mode SubG-CSS) or to the scheduled channel allocated (for SubG-FSK). For more information on frequency hopping, see Section 5.7.1.

Finally, the RF-modulated signal is amplified and filtered to achieve the required RF power spectrum. The RF signal must be compliant with normative requirements for bandwidth and power density. The Normative requirements are defined by relevant administrative bodies: ETSI for ITU Region 1, FCC for ITU Region 2., and others for ITU Region 3. Available channel lists with modulation parameters and duty cycles are shown in Table 5.4 and Table 5.5 for ITU Region 2.

Modulation	Channel	Frequency	Signal Bandwidth	SF	Duty cycle
CSS	#	MHz	kHz		
SubG-CSS 2 kbps	0	902.5	500	11	NA
	...	...			
	30	926.5			

Table 5.4 SubG-CSS Channel allocation for Region 2

NOTE 1: Channel spacing for SubG-CSS is 800 kHz.

NOTE 2: Channel 26 is the default working channel.

Modulation	Channel	Frequency	Frequency Deviation	Modulation index	BT
FSK	#	MHz	kHz		
GFSK 50 kbps	0	902.2	25	1	1
	...	...			
	128	927.8			

Table 5.5 FSK Channel allocation for Region 2.

NOTE: Channel spacing for SubG-FSK is 200 kHz. Sixty-nine channels are used in Amazon Sidewalk.

Half-duplex (HDX) transmission modes are implemented in both SubG-CSS and SubG-FSK due to the requirement for multi access on a single channel and limitations for radiated power. Both modulation methods can use any of the available channels available in the regional channel grid. As examples in ITU Region 2. 915 MHz band is available.

5.7.1 Frequency hopping

There are two reasons for developing a frequency hopping system. Firstly, the system becomes more immune against fades and single channel interferences. Secondly, the system fulfills requirements for channel-radiated power density normative limits. Slow frequency hopping is implemented in the Amazon Sidewalk protocol. There is a precalculated hopping pattern based on random *seed* and table-based *seq_id* for each entity. An algorithm implements Beacon channel collision avoidance. Details are described in Section 5.2.4.

5.7.2 LoRa modulation for SubG-CSS

LoRa modulation is a kind of spread spectrum modulation where a frequency chirp is implemented as a constant envelope frequency sweep within the working channel. Symbol rate (SR) is defined as:

$SR = BW/(2^{SF})$,

where BW is signal bandwidth and SF is spreading factor. SF 11 is used in Amazon Sidewalk rel. 1.10.

Bit rate (Rb) is calculated as:

$Rb = (SF * CR * BW)/(2^{SF})[bits/s]$, where CR code rate 4/5 ... 4/8.

> *NOTE: Higher SF provides better processing gain per symbol and hence requires less SNR. Furthermore, higher SF provides more immunity to co-channel interference.*

Frequency tolerance: $<$ +/-90.25kHz @500kHz BW; RX sensitivity: $\leq$ -128 dBm @ SF11/500kHz BW. [2]

5.7.3 GFSK modulation for SubG-FSK

In GFSK modulation, a Gaussian filter (pulse shaper with predefined bandwidth) is applied after the FSK modulator. This module reduces the occupied bandwidth of the modulated signal. NOTE: Lower BT increases inter-symbol interference. Occupied bandwidth (OBW) can be estimated as:

$OBW = Rb + 2 * Fdev$, where Rb is the PHY bitrate, and Fdev is the frequency deviation corresponding to the chosen modulation index. Rb equals symbol rate for GFSK modulation.

TX frequency clock accuracy $\leq$ +/- 40PPM [1],[4]; RX sensitivity: $\leq$ -107 dBm [3];

Blocking and selectivity in accordance with local requirements [5].

References:

[1] IEEE 802.15.4g (GFSK).

[2] Semtech SX 1272/73 application note.

[3] TI CC1070 application note.

[4] TI CC1200 application note.

[5] FCC CFR 47.

Chapter 6

BLE Protocol Stack

BLE is used as a transport medium to provide connectivity for an Endpoint to exchange encrypted messages with the Amazon Sidewalk Cloud via compatible BLE-enabled Amazon Sidewalk Gateways. This chapter provides details of the BLE protocol stack and use cases that are specific to Amazon Sidewalk. This chapter covers selected security aspects that are specific to BLE and are not covered by Chapter 4.

BLE usage in Amazon Sidewalk emphasizes power efficiency and low latency for Endpoint communication. BLE gives the most efficient and highest data rate but has a shorter range than the SubG-FSK and SubG-CSS links.

BLE devices may play central or peripheral roles. A peripheral device (Endpoint) advertises its presence by broadcasting an advertisement packet. A central device scans for advertisements and may choose to allow connection with a peripheral device based on the advertisement data. A central device may connect with multiple peripheral devices simultaneously. Smartphones with a mobile application installed and Gateways act as central devices while devices like sensors act as peripheral devices.

Amazon Sidewalk BLE devices use the Generic Attribute Profile (GATT) defined in Bluetooth Low Energy Specification v 4.2 with the following GATT concepts:

- **Client:** A device that initiates connection. Typically, the central device (smartphone/Gateway) is a GATT client.
- **Server:** A device that receives connection. Typically, a BLE peripheral is also the GATT server.
- **Service:** A collection of related characteristics is a BLE GATT service.
- **Characteristic:** Contains data to be exchanged between the client and server. Each characteristic can support one or more operations like read, write, notify or indicate. A client can subscribe to notifications when connecting with a GATT server.
- **Descriptor:** GATT Descriptors are defined attributes that describe a characteristic value. ”Client Characteristic Configuration Descriptor 0x2902” is used as a descriptor value.

6.1 Amazon Sidewalk BLE requirements

An Endpoint hosting an Amazon Sidewalk mobile application or working as a Gateway shall support the following functionalities:

Functionality	**A device hosting mobile application or a Gateway requirement**
BLE spec support	Minimum: BLE 4.2 (Not leveraging security features)
Number of concurrent connections	Miniumum: 4 Preferred: 6
Supported PHY layers	1.0 Mbps (minimum) 2.0 Mbps

6.1.1 Endpoint software requirements

Functionality	**Endpoint requirement**
BLE spec support	Minimum: BLE 4.2 (Not leveraging security features)
Profile	ATT (GATT) and GAP
Support for peripheral role (and advertiser)	Yes
Support for central role (and scanner)	No
Number of concurrent connections	N/A - only one connection support
Supported PHY layers	1.0 Mbps (minimum) 2.0 Mbps
Support for configurable ATT MTU size	Minimum: 185B Recommended: 512B
Support for extended advertising	No
Support for data length extension	Yes - 251B
Support for L2CAP with LE credit-based flow control mode	No
Support for LL privacy	Yes
Advertisement Interval	Alternating 160 ms and 30 s and 1 s [+/- 10 ms] for the rest of the time
Advertisement delay/window	0-10 ms
Connection Interval	30 ms - 4 s (based for use case) - enforced by central
Peripheral latency	Default 0

6.1.2 BLE KPIs

KPIs Type	Target	Definition	Notes
Concurrent connection	Miniumum: 4 Preferred: 6	Minimum simultaneous connection	KPIs for the Gateway or cellular phone with end user mobile application
Packet latency	Less than 2 s	Latency for beacon detection	Based on scanning interval KPIs for the Gateway or cellular phone with end user mobile application
Latency to connection	Less than 1s	Latency to get connected via central after beacon detection	

6.1.3 Traffic patterns

Pattern	Occurrence	Description	Size	Comments
Advertisement	160 ms for 30 s, 1 s afterwards	Standard Advertisement packet	31B	
Connection management	Upon each connection establishment	AMA stack handshake messages	Less than 32 Bytes each	The procedure contains two TX and two RX transmissions
		Authentication Beacon packet message on BLE connection	Fragmentation dependent. AMA fragmentation size depends on negotiated BLE MTU between central and peripheral	
		Time synchronization message	Up to 255 Bytes of BLE payload	

6.2 Amazon Sidewalk BLE protocol stack

Amazon Sidewalk BLE supports two types of messages that play the main role in the BLE communication between Endpoints in the Amazon Sidewalk environment.

The first message type is the Beacon that is used for communication initiation, e.g. data transfer, registration, time synchronization. The protocol stack for the Beacon is shown in Figure 6.1

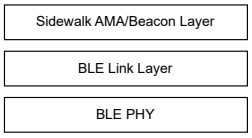


Fig. 6.1 BLE Beacon stack

The Beacon frame format and a detailed description of the different advertisement payloads is included in section Section 6.2.1.

The second message type is the data message that is used for data transmission and reception by the user application (custom commands) and Amazon Sidewalk management and security class commands used for registration, time synchronization, factory reset, and challenge request. The protocol stack of the data message is shown in Figure 6.2.

The Application Layer frame format and application commands are described in Chapter Chapter 3. The Presentation Layer frame format is described in section Section 5.1.1.1. The Flex Layer is described in Section 6.2.1.6.

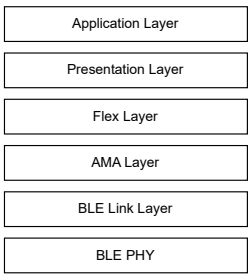


Fig. 6.2 BLE Data stack

The AMA (Alexa Mobile Accessory) Layer is common to the Beacon and Data formats. Details of the AMA frame and AMA frame types may be found in Section 6.2.1.9.

The BLE Link Layer and BLE PHY are described in the BLE Specification v 4.2.

6.2.1 Beacon frame format

The BLE Beacon is used for communication initiation. The following types of Beacons are defined based on the communication purpose:

1. Beacon for initiating registration of a new Endpoint to the Amazon Sidewalk network.
2. Beacon for initiating counter-based time synchronization procedure from the Amazon Sidewalk Cloud.
3. Beacon for initiating Endpoint-encrypted connection.
4. Beacon for presence detection and determining device state (ONLINE/OFFLINE).

The Beacon frame format is common to all Beacons, and selected fields in the Beacon frame define the purpose of the Beacon. The Beacon frame format contains two main blocks: Header and Manufacturing Data. The Header includes information related to the BLE Ad Type and BLE UUID of the service. Manufacturing Data contains information about the Amazon Sidewalk application, network service, device state and Application-Specific Data.

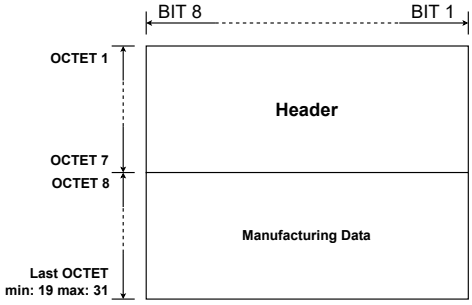


Fig. 6.3 BLE Beacon - Blocks

Every time the SDK starts advertising, the device advertises every 160 ms for the first 30 s after boot and 1 s afterwards, independently of the advertisement type. These values are user-configurable in the SDK. An Endpoint shall move to perform periodic time synchronization advertising mode for 60 s once every two (default) hours.

Beacons advertised by the Endpoint shall be forwarded by a Gateway to the cloud using three types of policies:

1. Upload - Beacons are forwarded to the cloud without a delay.
2. Accumulate - the Gateways maintain the latest Beacon information for each type of Beacon received and forward the first Beacon received of a particular type. The accumulated Beacon is sent after the defined accumulation period.
3. Drop - unknown or unsupported Beacon formats are dropped by the Gateway.

Beacon type	Policy name	Accumulation period
Registration/Provisioning	Upload	N/A
Counter-based time synchronization	Accumulate	30 s
Time-based connection request	Accumulate	30 s
Presence detection and determining device state	Accumulate	10min
Unknown	Drop	N/A

6.2.1.1 Beacon Header

The Beacon header block is coded in Tag, Length, Value format. All fields of the Beacon header are mandatory. Each field has only one value that is defined and supported by the specification. The header fields are presented in Figure 6.4.

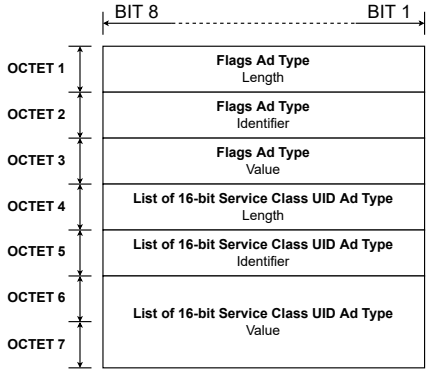


Fig. 6.4 BLE Beacon - Header

The header field values are constant and not configurable. The following values shall be applied in each header field:

Field name	Field size	Field value	Description
Flags Ad Type Length	1B	0x02	
Flags Ad Type Identifier	1B	0x01	
Flags Ad Type Value	1B	0x06	BLE only accessory
List of 16-bit Service Class UID Ad Type Length	1B	0x03	
List of 16-bit Service Class UUID Ad Type Identifier	1B	0x03	
List of 16-bit Service Class UUID Ad Type Value	2B	0x03FE	AMA Service UUID

6.2.1.2 Beacon Manufacturing Data

Manufacturing Data fields are presented in Figure 6.5.

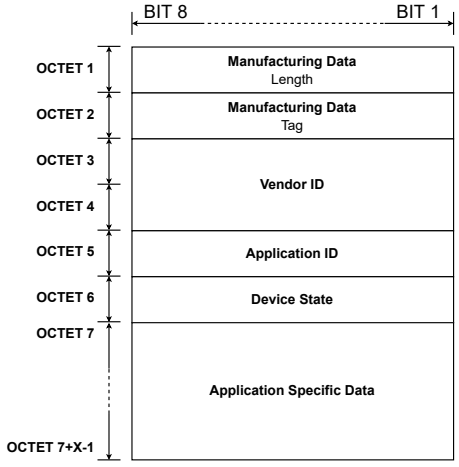


Fig. 6.5 BLE Beacon - Manufacturing Data

The length of the Manufacturing Data is mainly determined by application-specific data that contain information used for Beacon sender identification and authorization. The Manufacturing Data Tag and Vendor ID are constant and not configurable.

Field name	Field size	Field value	Description
Manufacturing Data Length	1B	0xB - 0x17	Size of subsequent fields of the Manufacturing Data Block
Manufacturing Data Tag	1B	0xFF	
Vendor ID	2B	0x0171	Amazon.com Services LLC (value assigned by Bluetooth SIG)

Service Type	Service value	Service Description
Sidewalk Services	0x00 - 0x06	Sidewalk Cloud Services
	0x07 - 0x1F	reserved
	0x20	Amazon Device Application
	0x21	IoT Wireless Application
	0x22 - 0x25	Amazon Device Application
	0x26 - 0xFF	reserved

6.2.1.3 Manufacturing data - Application ID

The Application ID field contains the identification of the Beacon receiver.

6.2.1.4 Manufacturing data - Device State

Device State includes information related to the device of the Beacon sender.

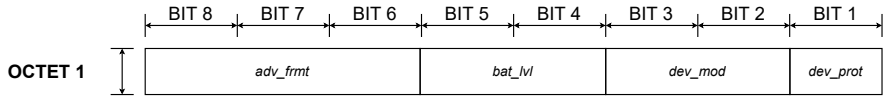


Fig. 6.6 BLE Beacon - Device State

Mnemonic	Length	Value	Description	
dev_prot	1b	0x0	reserved	Device protocol
		0x1	BLE	
dev_mod	2b	0x0	Normal mode	Device mode
		0x1	Registration Mode	
		0x2	Connection Request	
		0x3	reserved	
bat_lvl	2b	0x0	Critical level	Battery level
		0x1	Low level	
		0x2	Normal level	
		0x3	reserved	
adv_frmt	3b	0x0 - 0x1	Sidewalk Beacon version	Advertisment format
		0x2 - 0x7	reserved	

6.2.1.5 Manufacturing data - Application Specific Data

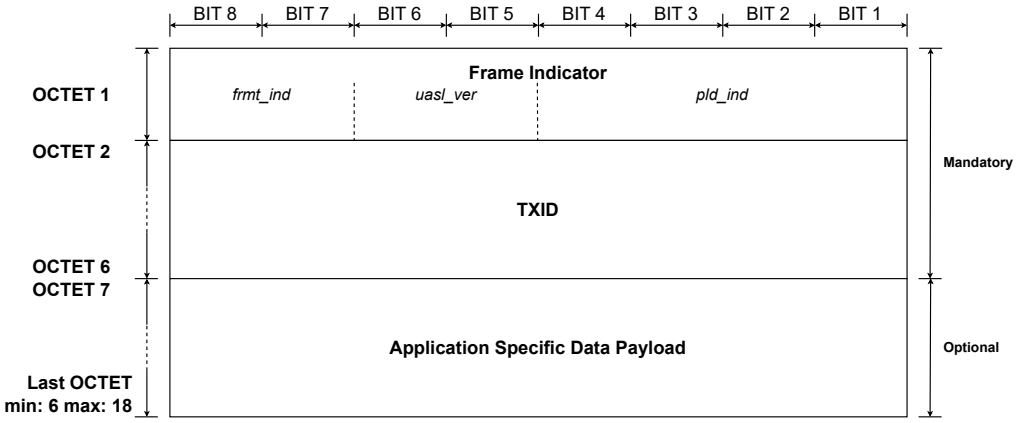


Fig. 6.7 BLE Beacon - Application Specific Data

Application-Specific Data format may differ for different Beacon types.

Frame Indicator contains information related to the type of device identification that is included in the TXID field (see *frmt_ind* description). It contains information about the Beacon/frame format version that is supported by the sender of the Beacon (see Section 5.1.1 and *uasl_ver* description).

Frame indicator Field name	**Frame indicator Field value**	**Frame indicator Field meaning**
frmt_ind	0x0	TXID time based frame
	0x1	TXID counter based frame
	0x2	Sidewalk ID / SMSN based frame
	0x3	reserved
uasl_ver	0x0	ASL version
	0x1 - 0x3	reserved
pld_ind	0x0	No Data
	0x1 - 0xF	reserved

The Frame Indicator has the following values defined for the different types of Beacon:

Application ID	**Device State**	**Frame indicator Value**	**Beacon type**
0x04	0x23, 0x2B, 0x33	0x80	Registration/Provisioning
0x03	0x21, 0x29, 0x31	0x40	Counter-based time synchronization
0x06	0x35, 0x2D, 0x25	0x00	Time-based connection request
0x21	0x21, 0x29, 0x31	0x00	Presence detection and determining device state

The Application-Specific Data format for different Beacon types is presented in figures 6.8, 6.9, and 6.10.

- Beacon for initiating registration of a new device to the Amazon Sidewalk network.

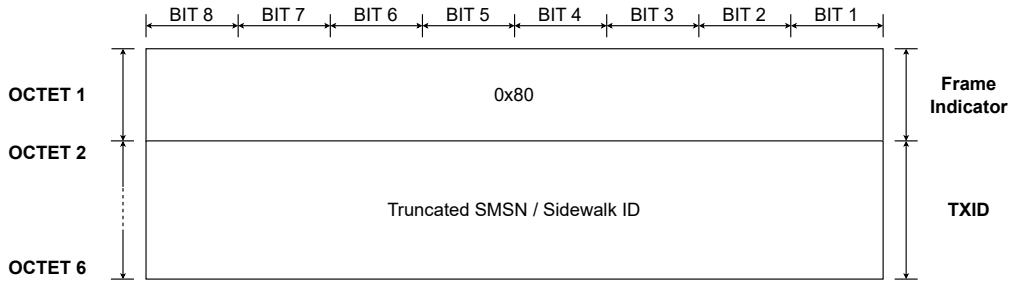


Fig. 6.8 BLE Beacon - Application Specific Data - Registration and Provisioning

- Beacon for initiating counter-based time synchronization.

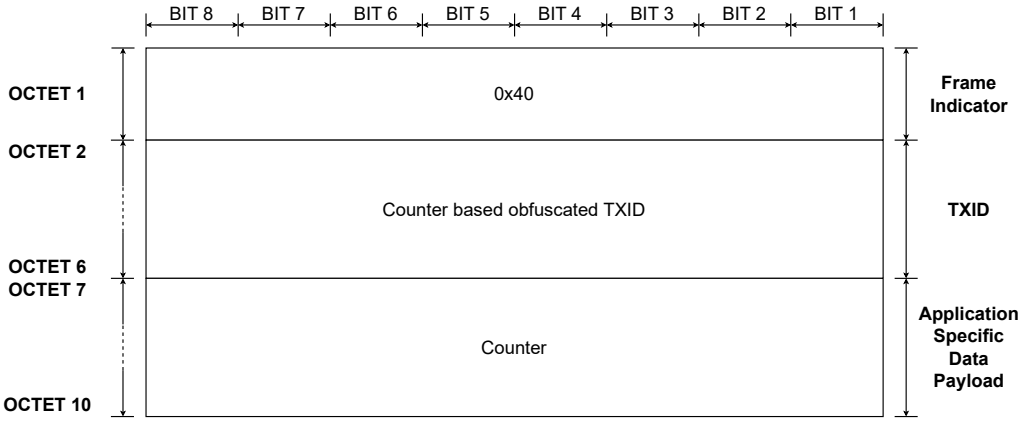


Fig. 6.9 BLE Beacon - Application Specific Data - Counter based time synchronization

- Beacon for initiating Endpoint connection and presence detection.

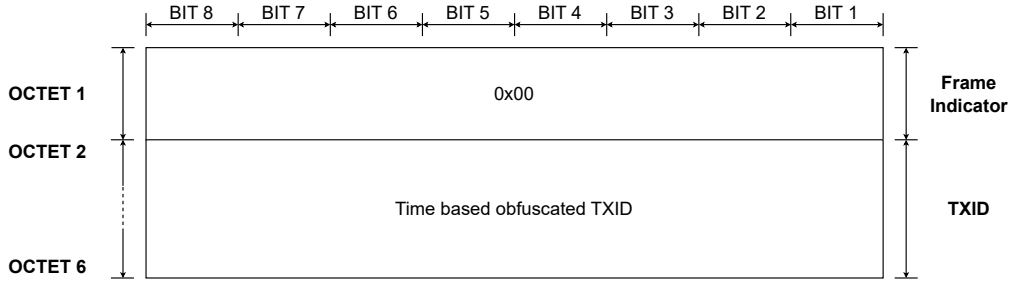


Fig. 6.10 BLE Beacon - Application Specific Data - Time based connection request and presence detection

6.2.1.6 Flex frame format

Amazon Sidewalk data transmission over BLE uses the single flex frame format. The frame format is presented in Figure 6.11. The frame contains two main blocks:

1. Flex Header Length that indicates the length of subsequent fields.

2. Flex Header that contains the sequence of several flex keys that include flex key tags, length and their associated values. Flex key tags, values and description of each flex key are described in Section 6.2.1.8. Flex key shall be coded in either Standard format or Optimized format (see Section 6.2.1.7).

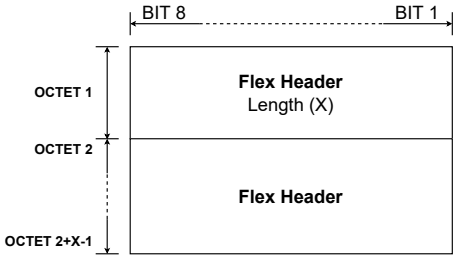


Fig. 6.11 BLE Data - Flex layer frame format

6.2.1.7 Flex key coding format

Flex keys are encoded in either Standard or Optimized format. The most significant 2 bits of the first octet in the flex key, define the encoding format used.

Bits 8 and 7 value	Coding format	Comment
0x3	standard	The flex key value size is indicated in the Flex Key Value Length field (see Figure 6.12).
0x2	4B optimized format	The flex component value is coded on 4 B. (see Figure 6.13)
0x1	2B optimized format	The flex component value is coded on 2 B. (see Figure 6.14)
0x0	1B optimized format	The flex component value is coded on 1 B. (see Figure 6.15)

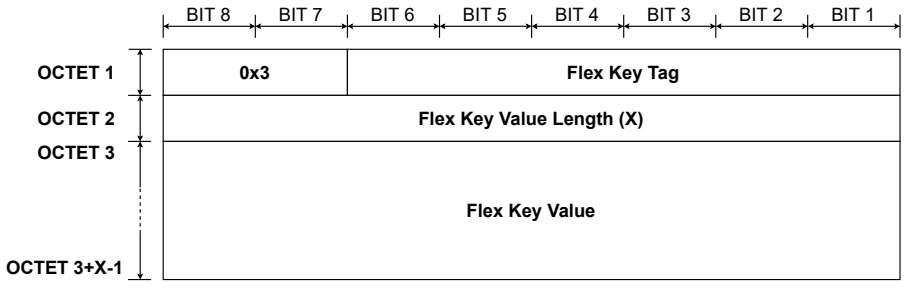


Fig. 6.12 Flex key coding - Standard format

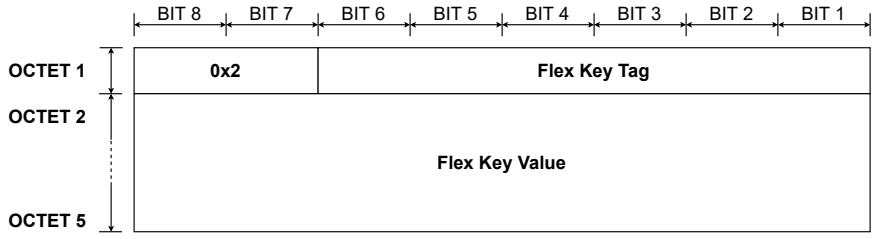


Fig. 6.13 Flex key coding - 4 B optimized format

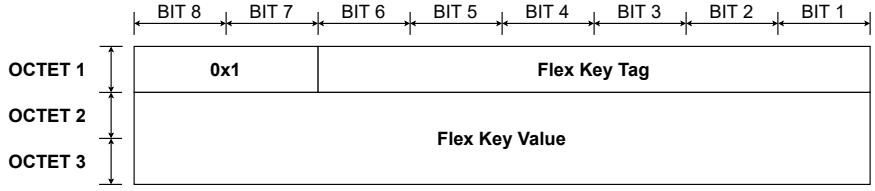


Fig. 6.14 Flex key coding - 2 B optimized format

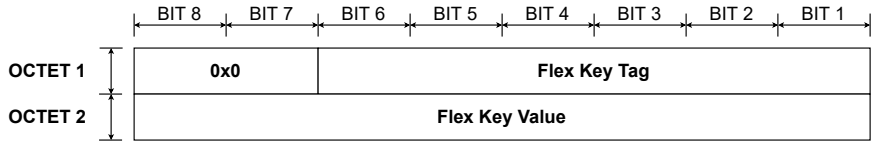


Fig. 6.15 Flex key coding - 1 B optimized format

6.2.1.8 Flex keys

The flex keys supported by Amazon Sidewalk data transmission over BLE are shown in Table 6.1

Flex Key Name	Flex Key Tag	Flex Key Value Length	Presence M/O/C	Description
FLEX_KEY_DST	0x01	1 - 5B	M	Destination address. The coding is the same as *dst* field in Link Layer - Message frame format (see Section 5.1.3.2).
FLEX_KEY_DST_FRMT	0x02	1B	M	Destination address format (see FLEX_KEY_DST). The coding is the same as *dst_frmt* field in Link Layer - Message frame format (see Section 5.1.3.2).
FLEX_KEY_ENCR_ENA	0x04	1B	M	Network level encryption flag. Value 0x00 and absence of the flex components means that encryption is disabled. Value 0x01 means that encryption is enabled. The Flex Payload is encrypted.
FLEX_KEY_SRC	0x10	1 - 5B	M	Source address. The coding is the same as *src* field in Link Layer - Message frame format (see Section 5.1.3.2).
FLEX_KEY_SRC_FRMT	0x11	1B	M	Source address format (see FLEX_KEY_SRC). The coding is the same as *src_frmt* field in Link Layer - Message frame format (see Section 5.1.3.2).
FLEX_KEY_PROTOCOL_VER	0x12	1B	M	Flex Protocol Version Number Value 0x01 is the only value supported by the specification.
FLEX_KEY_SEC_REF	0x1B	1B	M	Security reference configuration. The coding is the same as *sec_ref* field in Network Layer frame format (see section Section 5.1.2.2). Absence of this flex component means that security reference is GCS.
FLEX_KEY_SEC_REF_COUNTER_VAL	0x1C	4B	C	Counter value to be used as security reference. The field is mandatory when FLEX_KEY_SEC_REF indicates security reference COUNTER.
FLEX_KEY_SEC_TOPOLOGY	0x1E	1B	M	Network security topology. The coding is the same as *sec_top* field in Network Layer frame format (see section Section 5.1.2.2).
FLEX_KEY_LL_SEQ_NUM	0x21	1 - 3B	M	BLE Link Layer sequence number. 2MSbs specify size of field in bytes.
FLEX_KEY_AUTH_TAG	0x23	4B	M	Authentication tag.
FLEX_KEY_IS_DATA_ASL_FORMATTED	0x2B	1B	M	Flex Payload formating flag. Value 0x00 or absence of this flex component means that Presentation Layer is not included. The value is not supported by the specification. It is used for backward compatibility only. Value 0x01 means the ASL frame format.

Table 6.1 Flex keys

6.2.1.9 AMA frame format

The Amazon Sidewalk frame format is encapsulated within an Amazon proprietary (AMA - Alexa Mobile Accessory) format for which the specification will be publicly available in the near future.

6.3 Amazon Sidewalk BLE security

The Amazon Sidewalk security model, architecture and certificate chain are described in Chapter 4, Section 4.1, Section 4.1.1, Section 4.1.2. The frame structure is described in section Section 4.8. The key structure and names, key exchange and authentication procedure are the same as those described for sub-GHz technology in Section 4.5. The following sections of the document focus on the differences in implementation for the specificity of the BLE medium (high bandwidth and available frame length 255 B). The procedures for synchronization, registration and authentication of an Endpoint in the Amazon Sidewalk system are described in Chapter 4, and in Section 4.2 and Section 4.4.2.2.

6.3.1 Amazon Sidewalk Application Layer credentials

Detailed specification for the Amazon Sidewalk Application Layer credentials for Network Server and Application Server are presented in the following subsections.

6.3.1.1 Network Server Encryption/Decryption

Counter-based credentials are used only after a factory reset when an Endpoint has not yet synchronized time. Time-based credentials are used for regular data traffic.

Symmetric keys with 128 bit length are used for coding the Network Server link. Authentication tag is 4 B size. Initialization vector is 12 B length. Additional Authentication Data are a concatenation of Flex parameters listed in Table 6.2 below.

Credentials	Time-based Commands	Counter-based Commands	Registration (Touchless FFN / Mobile)
Key type	GCM_KEY	GCM CTRREF KEY	GCM CTRREF KEY
Authentication tag	4B	4B	4B
AAD	| 1B Flex Header length | FLEX_KEY_LL_SEQ_NUM | FLEX_KEY_SRC_FRMT | FLEX_KEY_SRC | FLEX_KEY_DST_FRMT | FLEX_KEY_DST | FLEX_KEY_ENCR_ENA | FLEX_KEY_PROTOCOL_VER | FLEX_KEY_SEC_REF_COUNTER_VAL | FLEX_KEY_SEC_TOPOLOGY | FLEX_KEY_SEC_REF | FLEX_KEY_IS_DATA_ASL_FORMATTED |	| 1B Flex Header length | FLEX_KEY_LL_SEQ_NUM | FLEX_KEY_SRC_FRMT | FLEX_KEY_SRC | FLEX_KEY_DST_FRMT | FLEX_KEY_DST | FLEX_KEY_ENCR_ENA | FLEX_KEY_PROTOCOL_VER | FLEX_KEY_SEC_REF_COUNTER_VAL | FLEX_KEY_SEC_TOPOLOGY | FLEX_KEY_SEC_REF | FLEX_KEY_IS_DATA_ASL_FORMATTED |	| 1B Flex Header length | FLEX_KEY_LL_SEQ_NUM | FLEX_KEY_SRC_FRMT | FLEX_KEY_SRC | FLEX_KEY_DST_FRMT | FLEX_KEY_DST | FLEX_KEY_ENCR_ENA | FLEX_KEY_PROTOCOL_VER | FLEX_KEY_SEC_REF_COUNTER VAL | FLEX_KEY_SEC_TOPOLOGY | FLEX_KEY_SEC_REF | FLEX_KEY_IS_DATA_ASL_FORMATTED |
IV	| 5B Source ID | 2b Link type | 22b Sequence number | 4B Reference time —	| 5B Source ID | 2b Link type | 22b Sequence number | 4B counter reference|	| 5B Source ID | 2b Link type | 22b Sequence number | 4B counter reference—
	Note: Link type : 0x0 : Radio, 0x1 : BLE		

Table 6.2 Encryption and Decryption parameters

6.3.1.2 Counter-based network server security

Counter-based network security is used in the registration procedure and the deregistration procedure when an Endpoint has not yet synchronized time. Payload is processed according to the formula:

AES-GCM (Encrypted Payload, AAD, IV, Authentication TAG, GCM CTRREF KEY)

where

GCM CTRREF KEY =
CMAC (NETWORK_PRIMARY_KEY, "Network_KDF_Dev_Ntw_GCM_COUNTER"
| Counter Reference),

and Counter Reference = Counter value supplied in the flex header.

6.3.1.3 Network Server Authentication parameters (Only TIME_SYNC command)

Counter-based time synchronization (TIME SYNC command) is used when an Endpoint reboots and/or over the periodic time sync.

The only difference between the time-based and counter-based time sync commands is the key type used. Details are presented in Table 6.3.

Credentials	Time-based	Counter-based
Key type	TSYNC RESP KEY (Not Supported)	TSYNC RESP CTRREF KEY
Authentication Tag (MAC) size	4B	4B
IV (Initialization Vector)	None	None
AAD (Additional Authentication Data)	| 1B Flex Header length | FLEX KEY LL SEQ NUM | FLEX KEY SRC FRMT | FLEX KEY SRC | FLEX KEY DST FRMT | FLEX KEY DST | FLEX KEY ENCR ENA | FLEX KEY PROTOCOL VER | FLEX KEY SEC REF COUNTER VAL | FLEX KEY SEC TOPOLOGY | FLEX KEY SEC REF | FLEX_KEY_IS_DATA_ASL_FORMATTED |	| 1B Flex Header length | FLEX KEY LL SEQ NUM | FLEX KEY SRC FRMT | FLEX KEY SRC | FLEX KEY DST FRMT | FLEX KEY DST | FLEX KEY ENCR ENA | FLEX KEY PROTOCOL_VER | FLEX KEY SEC REF COUNTER VAL | FLEX KEY SEC TOPOLOGY | FLEX KEY SEC REF | FLEX_KEY_IS_DATA_ASL_FORMATTED |

Table 6.3 BLE key types

6.3.2 Application Server Encryption/Decryption

AES-CTR cryptography is used for Application Server data processing. Details are presented in Table 6.4

Credentials	Time based	Registration
Key type	APP_CTR_KEY	APP_GCM_CTRREF_KEY
AAD (Additional Authentication Data)	NONE	| 1B Flex Header length | FLEX KEY LL SEQ NUM | FLEX KEY SRC FRMT | FLEX KEY SRC | FLEX KEY DST FRMT | FLEX KEY DST | FLEX KEY ENCR ENA | FLEX KEY PROTOCOL VER | FLEX KEY SEC REF COUNTER VAL | FLEX KEY SEC TOPOLOGY | FLEX KEY SEC REF | FLEX_KEY_IS_DATA_ASL_FORMATTED | 2B - 4B of plain text ASL header |
Authentication Tag (MAC) size	NONE	4B
IV (Initialization Vector)	| 5B Source ID | 1b uplink/downlink | 7b Zeros | 2B app layer seqn | 4B time reference | 4B Zeros | Source ID : Component generating the packet UL : Sidewalk ID DL : Cloud component generating the packet time reference will be same as Network Layer encryption time reference 1b uplink/downlink : 1 = uplink, 0 = downlink 4B Zeros : Padding to align 16Bytes IV	| 5B Source ID | 1b Link type | 7b null | 2B app layer seqn | 4B counter reference | Source ID : Component generating the packet UL : Sidewalk ID DL : Cloud component generating the packet (Provisioning Service) link type 1 - uplink 0 - downlink

Table 6.4 Encryption/Decryption parameters

6.3.3 D2D-related parameters

D2D feature uses a security procedure similar to that used for registration. D2D_PRIMARY_KEY is used for mobile to Endpoint communication. Derived keys with their credentials and derivation method are in Table 6.5.

Credentials	D2D_GCM KEY	D2D_GCM CTRREF KEY
Key type	time based	counter based
Authentication tag	4B	4B
AAD	| 1B Flex Header length | FLEX_KEY_LL_SEQ_NUM | FLEX_KEY_SRC_FRMT | FLEX_KEY_SRC | FLEX_KEY_DST_FRMT | FLEX_KEY_DST | FLEX_KEY_ENCR_ENA | FLEX_KEY_PROTOCOL_VER | FLEX_KEY_SEC_REF_COUNTER_VAL | FLEX_KEY_SEC_TOPOLOGY | FLEX_KEY_SEC_REF | FLEX_KEY_IS_DATA_ASL_FORMATTED ||	| 1B Flex Header length | FLEX_KEY_LL_SEQ_NUM | FLEX_KEY_SRC_FRMT | FLEX_KEY_SRC | FLE_KEY_DST_FRMT | FLEX_KEY_DST | FLEX_KEY_ENCR_ENA | FLEX_KEY_PROTOCOL_VER | FLEX_KEY_SEC_REF_COUNTER_VAL | FLEX_KEY_SEC_TOPOLOGY | FLEX_KEY_SEC_REF | FLEX_KEY_IS_DATA_ASL_FORMATTED |
IV	| 5B Source ID | 2b Link type | 22b Sequence number | 4B counter reference|	| 5B Source ID | 2b Link type | 22b Sequence number | 4B counter reference|
Key derivation	CMAC(D2D_PRIMARY_KEY, "Mobile_KDF_Device_Mobile_GCM"|time reference for this key)	CMAC(D2D_PRIMARY_KEY, "Mobile_KDF_Device_Mobile_GCM_COUNTER"|[0x00, 0x00, 0x00, 0x00])

Table 6.5 Device to device keys

6.4 Amazon Sidewalk BLE features

6.4.1 Endpoint-initiated Encrypted Connection

It is assumed that an Endpoint is registered and time synchronized. Process flow is presented in Figure 6.16. Obfuscated TXID and Amazon Sidewalk internal server ID are used in the advertisement packet. Details of frames content are presented in Section 6.2.

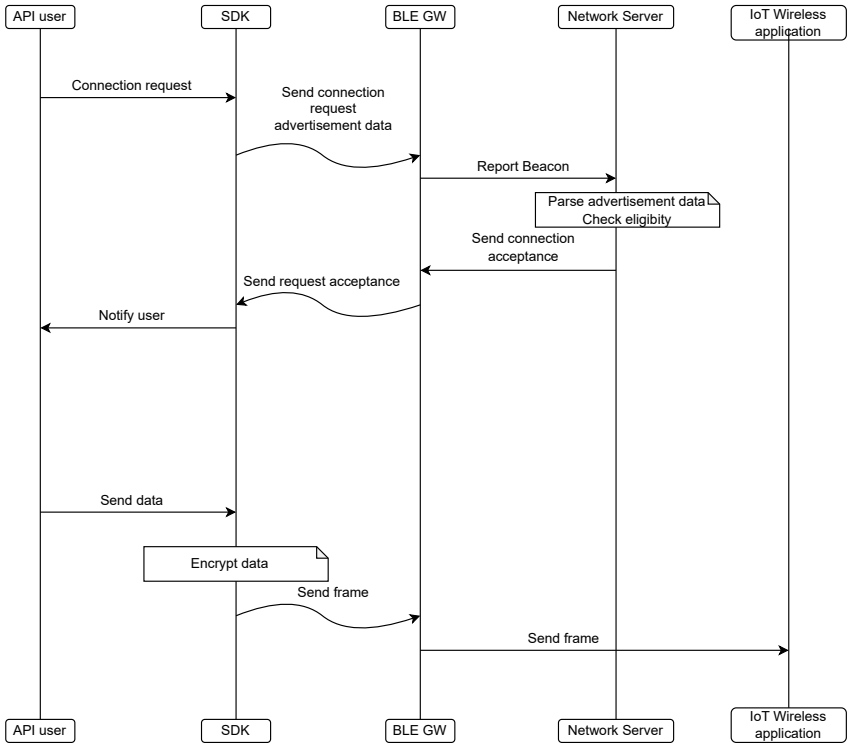


Fig. 6.16 BLE Endpoint initiated encrypted connection

The connection request advertisement is an on-demand procedure when an application needs to send an uplink packet but no active connection to the Amazon Sidewalk Cloud is available. A connection request from an Endpoint application API is handled by an Endpoint SDK to translate into a connection request

advertisement. The Amazon Sidewalk BLE Gateway forwards the connection request advertisement to the Amazon Sidewalk internal server as a Report Beacon message. The Amazon Sidewalk internal server, based on decoded content, builds a challenge request and sends it back to a BLE Gateway which then forwards the challenge request to the Endpoint over RF. After successful reception and authentication, the Endpoint SDK notifies the application of confirmation of link existence after which the application can send uplink data traffic over the Amazon Sidewalk network.

The encrypted link is disconnected due to time-outs and message decryption errors.

6.4.2 Device-to-Device Communication (D2D)

Device to device (D2D) functionality describes encrypted connection of the Amazon Sidewalk Mobile SDK hosted on a mobile phone and an Endpoint. D2D functionality encompasses Amazon Sidewalk Mobile SDK authentication to the Amazon Sidewalk Cloud and encrypted channel establishment between the Mobile Application and an Endpoint.

The core of D2D functionality is the Mobile Application to Endpoint device key (D2D_PRIMARY_KEY) that is established on the Endpoint and on the Mobile Application. The customer owning an Endpoint can have more than one instance of the Mobile Application in the Amazon Sidewalk environment. Two basic cases are possible:

1. 1x1 configuration - one Mobile Application handles the Endpoint.
2. Nx1 configuration - multiple Mobile Applications instances with the Endpoint belonging to the customer. All instances of the Mobile Application have the same D2D_PRIMARY_KEY.

D2D_PRIMARY_KEY is generated independently on the Mobile Application and the Endpoint in case of 1x1 configuration. When an additional instance of the Mobile Application is created (Nx1 configuration) then this additional instance of the Mobile Application gets the D2D_PRIMARY_KEY from the Endpoint using a session key.

D2D_PRIMARY_KEY is the input parameter for derivation of the D2D_GCM_KEY, and for derivation of the D2D_GCM_CTRREF_KEY that are used in encrypted communication between the Mobile Application and the Endpoint with either time-based reference or counter-based reference (see Chapter 6.3 for the derivation formula).

D2D connection establishment is the base for data exchange between the Mobile Application and the Endpoint. D2D session establishment, Custom class, Challenge request/response and Metrics commands are supported on D2D communication.

6.4.2.1 D2D communication scenarios

D2D communications between the Mobile Application, the Amazon Sidewalk Cloud and the Endpoint include the following:

1. **The Endpoint is not registered**
   - Mobile activity
     (a) Detects Endpoint proximity based on Beacon data.
     (b) Starts establishing the encrypted channel with the cloud and the Endpoint.
     (c) At the end of establishing the encrypted channel:
         i. Mobile Application instance will have the D2D_PRIMARY_KEY and the frequency of D2D_GCM_KEY key rotation.
     (d) If channel establishment fails or the BLE link disconnects, and a D2D_PRIMARY_KEY was generated, that key is discarded.

- Cloud activity

  Supports mobile to establish the encrypted channel establishment by validating the Endpoint with signature verification from the Endpoint.

- Endpoint activity

  (a) Performs signature verification from the cloud via a Mobile Application to verify it is communicating with a trusted source.

  (b) At the end of the procedure of establishing the channel:

  i. The Endpoint will have the D2D_PRIMARY_KEY and the frequency of D2D_GCM_KEY key rotation.

  (c) In case of failure or disconnection of BLE link, D2D_PRIMARY_KEY is discarded if generated.

2. **The Endpoint is registered and Mobile Application has Mobile to device key**

- Mobile activity

  (a) Checks if the Endpoint is registered with the Beacon data and:

  i. if authentication material is present, authenticates the obfuscated TXID with the authentication material for the Amazon Sidewalk ID.

  ii. if authentication material is not present, requests that the cloud to identify the device and receives the authentication material.

  iii. if Endpoint is requesting time sync, Mobile Application shall request that the Amazon Sidewalk Cloud provide time to the Endpoint

  (b) Establishes direct BLE connection to exchange data using time-based encryption (D2D_GCM_KEY key).

- Cloud activity

  Performs the Endpoint authentication on the mobile request and provides authentication material to the Mobile Application.

- Endpoint activity

  Decrypts the packet (using D2D_GCM_KEY key) and provide the payload to application.

3. **The Endpoint is registered and mobile does not have Mobile to device key**

- Mobile activity

  (a) Checks if the Endpoint is registered with the Beacon data and:

  i. if authenticated material is present, authenticates the obfuscated TXID with the authentication material for the Amazon Sidewalk ID.

  ii. if authenticated material is not present, requests that the cloud identify the device and receives the authentication material.

  iii. if Endpoint is requesting time sync, Mobile Application shall request that the Amazon Sidewalk Cloud provide time to the Endpoint

  (b) Requests the cloud to send a challenge request to the Endpoint. The Endpoint shall respond with its challenge response to the cloud for authentication and the Endpoint shall open up the encrypted session channel.

  (c) Starts the procedure of establishing the encrypted channel and derives the session key. A derived key from the session key shall be used to get the D2D_PRIMARY_KEY from the Endpoint.

- Cloud activity

  (a) Performs the Endpoint authentication on the mobile request and provides authentication material to the Mobile Application.

  (b) Sends the challenge request to the Endpoint via the Mobile Application.

  (c) Validates the challenge response from the Endpoint.

- Endpoint activity
  (a) Validates the challenge request. When the validation is successful then sends the challenge response to the cloud.
  (b) Opens up the gate to honor commands required for establishing an encrypted session and derivation of session key.
  (c) Derived key from session key shall be used to share the D2D_PRIMARY_KEY to the Mobile Application.

4. **The Endpoint is registered and both the Endpoint and Mobile Application do not have primary key**

- Mobile activity
  (a) Checks if the Endpoint is registered with the Beacon data and:
    i. if authenticated material is present, authenticates the obfuscated TXID with the authentication material for the Amazon Sidewalk ID.
    ii. if authenticate material is not present, requests that the cloud identify the device and receives the authentication material.
    iii. if the Endpoint is requesting time sync, the Mobile Application shall request that the Amazon Sidewalk Cloud provide time to the Endpoint.
  (b) Requests that the cloud send a challenge request to the Endpoint. The Endpoint shall respond with its challenge response to the cloud for verification.
  (c) Starts the procedure of establishing the encrypted channel and derives the session key. Since the Endpoint does not have the D2D_PRIMARY_KEY, the Endpoint shall store the session key as D2D_PRIMARY_KEY. D2D_PRIMARY_KEY is shared using a derived key from session key.
- Cloud activity
  (a) Performs the Endpoint authentication on the mobile request and provides authentication material to the Mobile Application.
  (b) Sends the challenge request to the Endpoint via the Mobile Application.
  (c) Validates the challenge response from the Endpoint.
- Endpoint activity
  (a) Validates the challenge request. When the validation is successful then sends the challenge response to the cloud.
  (b) Opens up the gate to honor commands required for establishing an encrypted session and derivation of session key.
  (c) Derived key from session key shall be used to share the D2D_PRIMARY_KEY to the Mobile Application.

5. **The Endpoint is registered to another user**

   The Amazon Sidewalk Cloud shall prevent D2D communication from the Mobile Application to the Endpoint by not providing the authentication material to the Mobile Application.

6.4.3 Touchless Registration (Frustration-Free Network)

Frustration-Free Network (FFN) is a procedure where an Endpoint is registered onto the Amazon Sidewalk network without user interaction (Touchless). Simplified architecture diagram is presented in Figure 6.17.

There are three types of user accounts possible to provide FFN setup:

Setup options are presented in Figure 6.18.

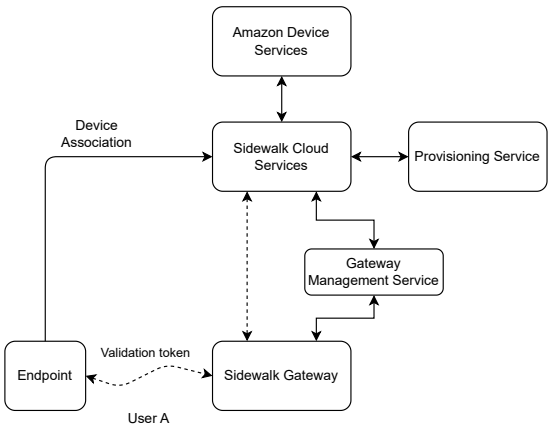


Fig. 6.17 BLE FFN architecture

User account type	Registration type
Amazon account	Type A registration
Enterprise/Partner account	Type B registration
No account (WAN use cases)	Type C registration

The FFN process consists of two preliminary steps: Amazon Sidewalk registration and initialization session and actual registration in the Amazon Sidewalk system. During the FFN procedure the Endpoint is verified and authenticated by the Amazon Sidewalk Cloud and the credentials, and the key and configuration are stored in the Endpoint and in the cloud registry. Following completion of the FFN procedure, the BLE link is disconnected. The Endpoint is then ready to establish a link with its updated key and configuration. Using the link, the Endpoint can obtain time information and perform any required Amazon Sidewalk functionality.

For detailed flow diagrams see: Figure 6.19, and Figure 6.20.

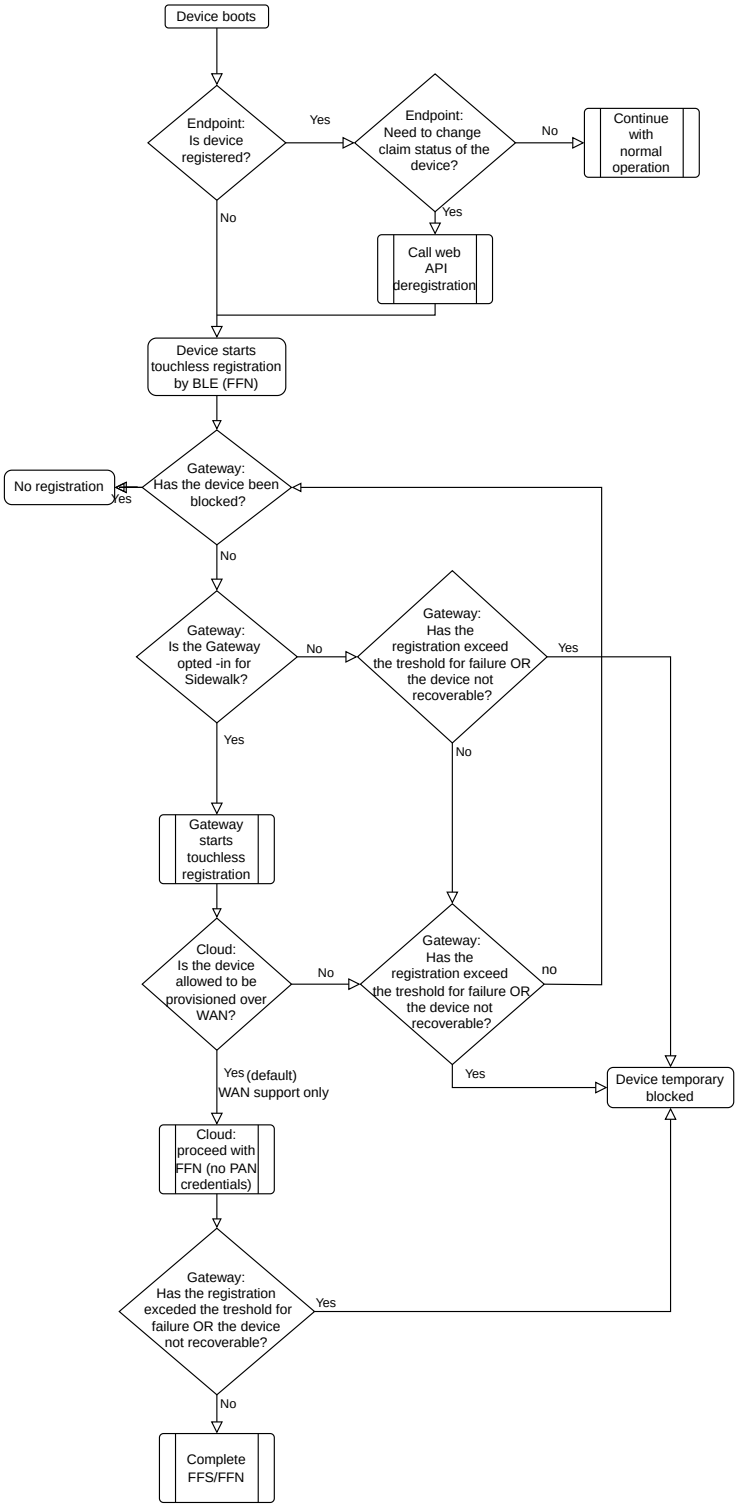


Fig. 6.18 BLE FFN touchless registration flow

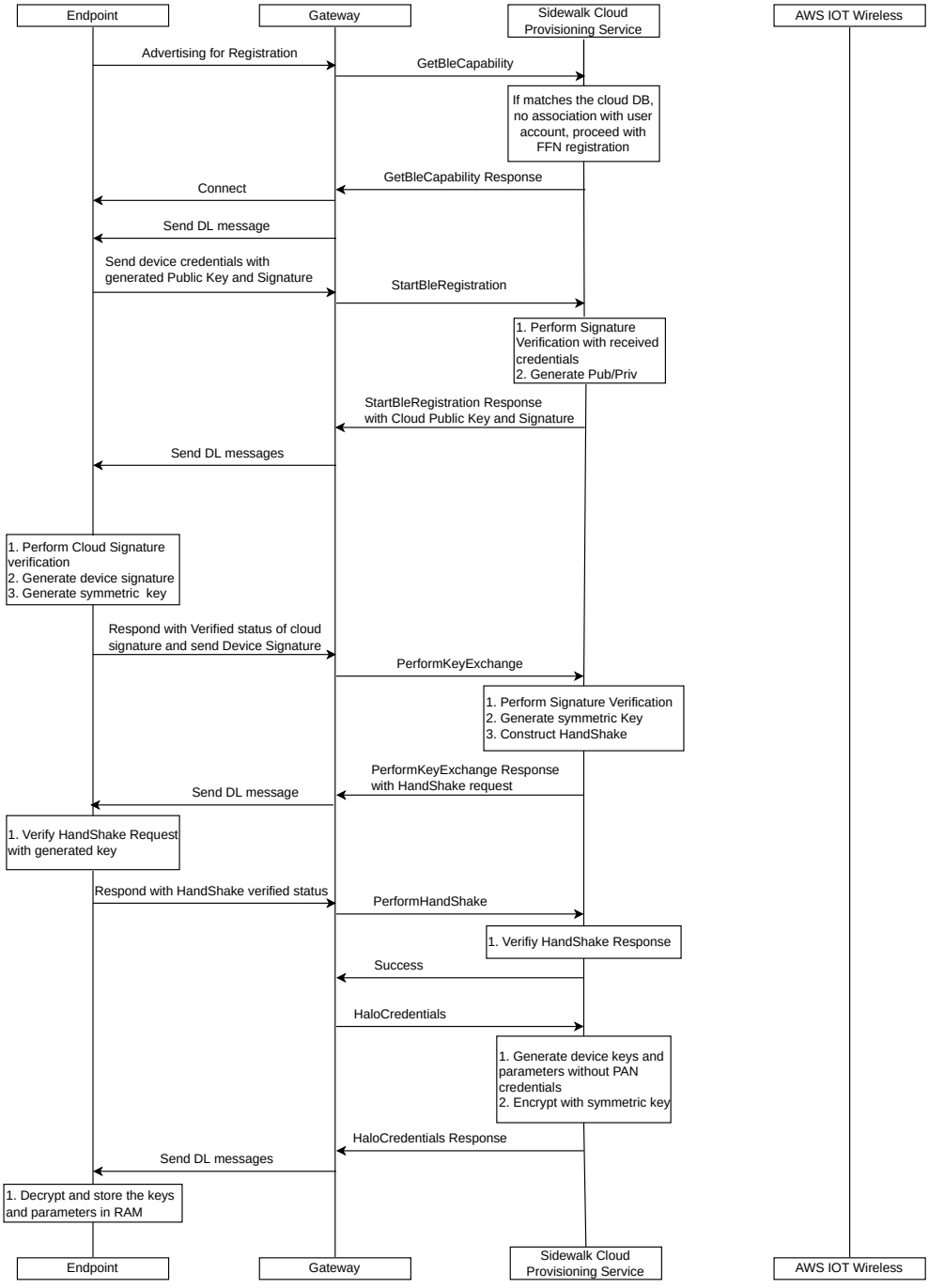


Fig. 6.19 BLE Endpoint registration 1of2

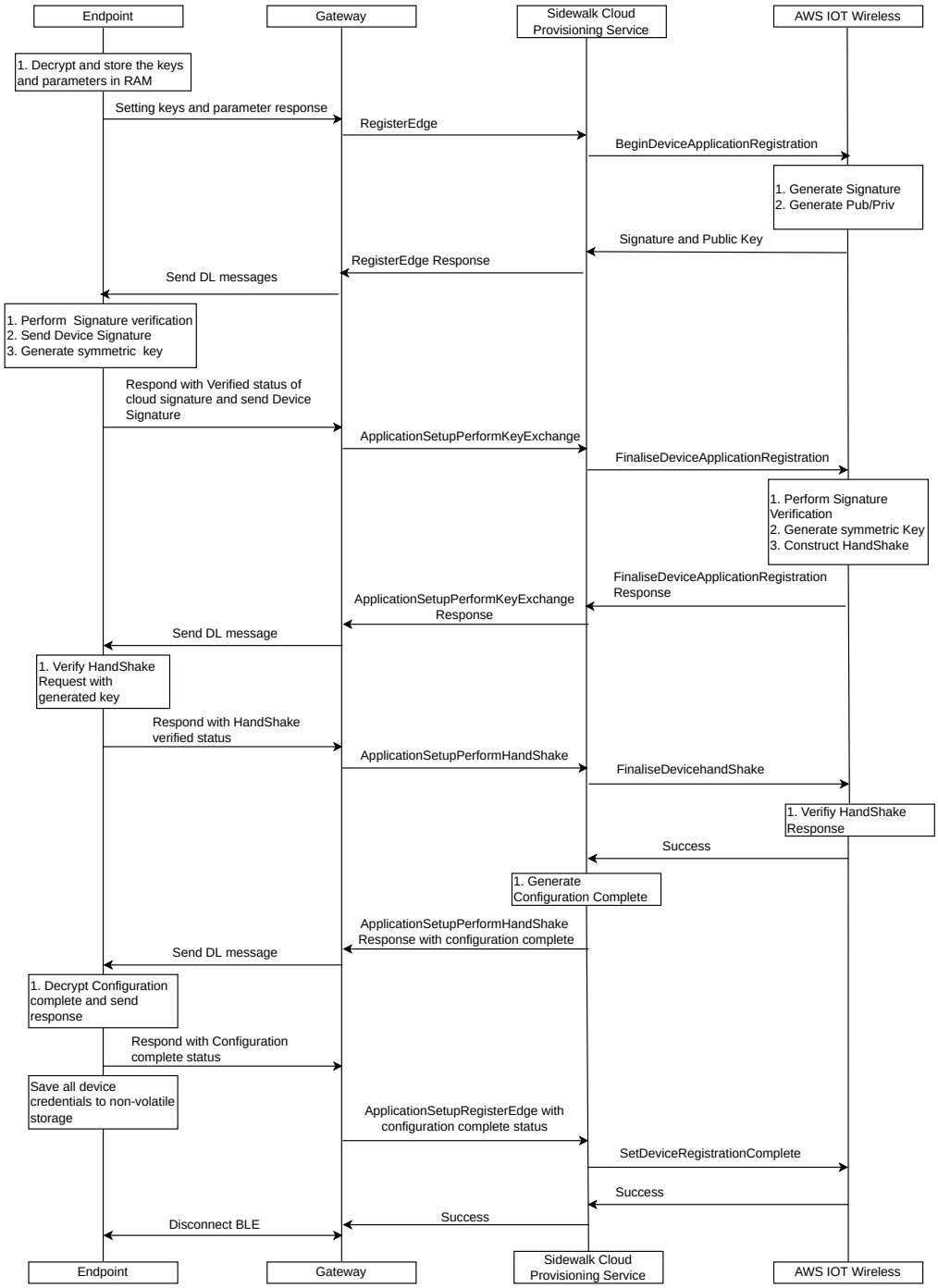


Fig. 6.20 BLE Endpoint registration 2of2

6.4.4 Time Sync trigger over Amazon Sidewalk BLE Beacon

Time Synchronization with a common time source is a prerequisite for the operation of time-based authentication materials and encryption of messages in normal operation mode. During the boot-up process, the internal clock in the Endpoint is not synchronized with the NTP source, therefore counter-based synchronization is carried out in the first stage of establishing the connection. The counter is represented by 4 B length TimeSync.

The synchronization process is as follows: An Endpoint sends the TimeSync Beacon to the Gateway and the Gateway forwards the TimeSync Beacon once every 30 s to the Amazon Sidewalk TimeSync Server. The Amazon Sidewalk TimeSync server, based on the received Beacon, authenticates the counter-based TXID and sends back to the Gateway a notify Response Time with one time authentication key. Gateway provides the reference time to the Endpoint. After authenticating the response time, the Endpoint disconnects the BLE link and changes its state to obfuscated time-based TXID.

During normal operation mode, the Endpoint will move to periodic TimeSync mode once every 2 h for 60 s to correct time drift. If the time cannot be synced, then the Endpoint shall remain time synced and Beacon with obfuscated time-based TXID for normal operation. When the Endpoint detects the drift to be more than 60 s, then the Endpoint will reset the network time and more into TimeSync request mode till the Endpoint receives a time from the Amazon Sidewalk Cloud.

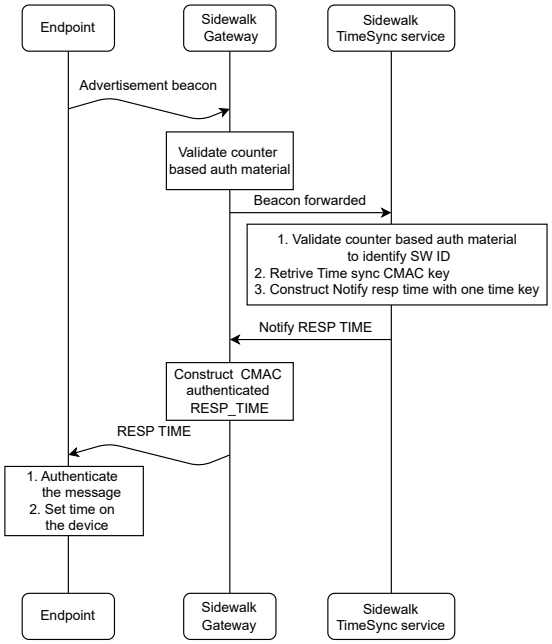


Fig. 6.21 BLE TimeSync flow

For details of the events used for time synchronization, and the periodicity of events, see Table 6.6.

TimeSync events	Periodicity	Comments
Beaconing Interval	160ms	for the first 30s after boot and 1s afterward
Time sync beacons Upload from gateway	30sec	
Periodic time sync request (counter-based)	Every 2 hours	For 60 seconds

Table 6.6 Periodicity of TimeSync events

6.4.5 Network Key Refresh

6.4.5.1 Introduction

Sufficiently frequent key exchange is essential to avoid system compromise. Based on estimation of traffic and keys' complexity, the key refresh procedure is to be triggered periodically for the Endpoint. This periodicity is configurable by the Amazon Sidewalk Cloud. The network key refresh process is similar to the registration procedure. The network key refresh process is triggered by the cloud service in the Amazon Sidewalk system. Due to available bandwidth in the BLE channel, the impact for the system performance during executing network key refreshing procedure can be neglected.

Due to lack of a direct TCP link between Endpoint and Network Server an intermediate state of the state machine maintaining both established (previous) keys and current (new) keys is necessary.

6.4.5.2 Key Refresh Flow

1. Key refreshing scheduler initiates Key Refresh process using a time-based policy for cloud-initiated Key Refresh request.
2. Keep track of Key Refresh state and update flag Key Refresh pending if the Endpoint is not in a pending reset state.
3. Perform the Key Refresh if the Endpoint is time-synchronized.
   (a) Check Key Refresh Initiated flag for device refresh state
   (b) Provisioning Service will send a cloud-authenticated message to start Key Refresh process on the Endpoint
4. Upon handshake verification, Amazon Sidewalk Provisioning Service shall indicate to the Network Server to delete the old authentication material and calculate the new authentication material.
5. On completion of the Key Refresh procedure, the Endpoint shall Beacon with the TXID generated with the new key.

6.4.5.3 Sequence Diagram

The process follows the diagram shown in Figure 6.22

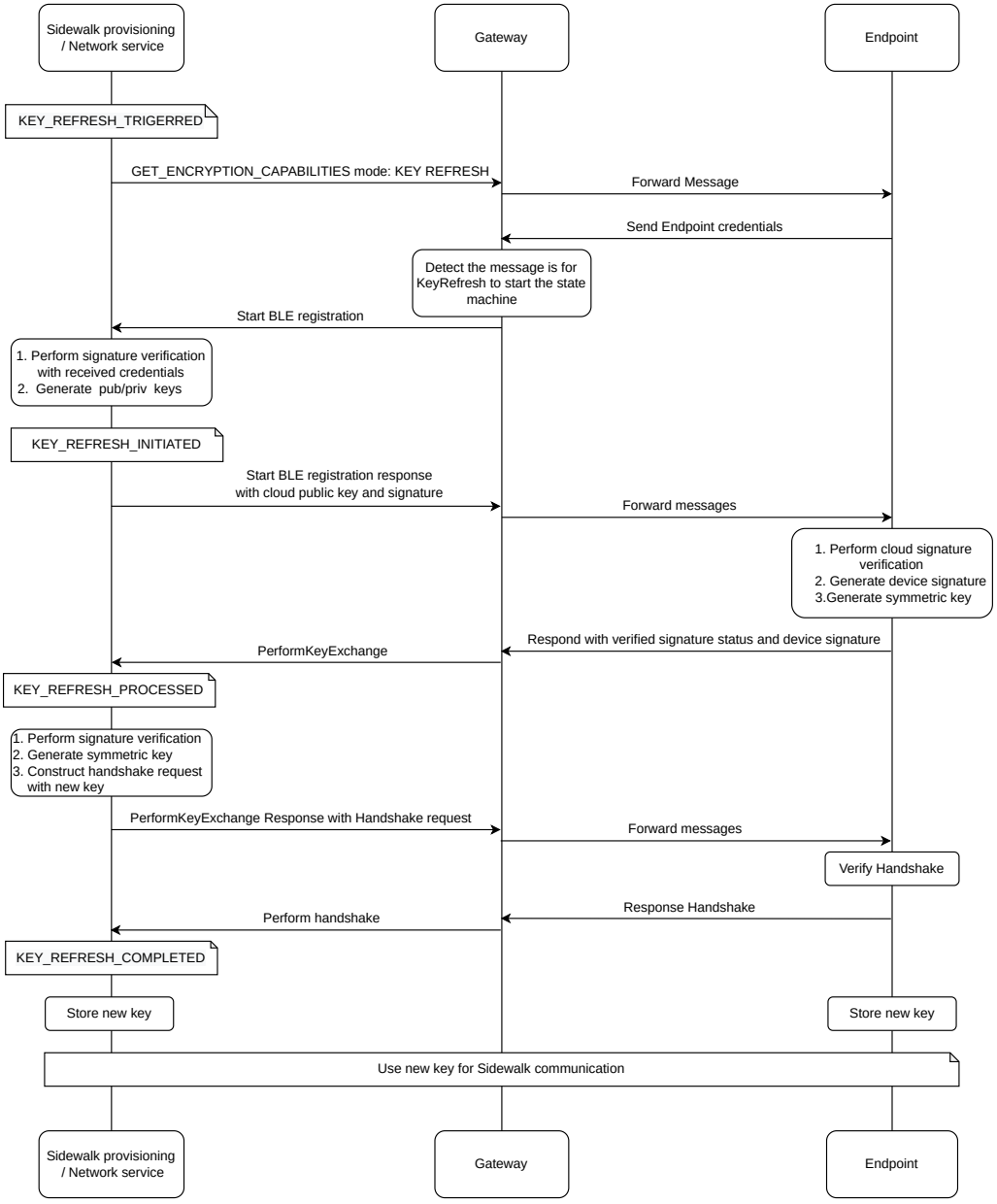


Fig. 6.22 BLE Network key refresh

Details about state machine flags are presented below.

6.4.5.4 States for Key Refresh

- KEY_REFRESH_TRIGGERED: When the scheduler trigger event is received for an Endpoint in the provisioning service and GET_ENCRYPTION_CAPABILITIES is sent to the Endpoint from the cloud.
- KEY_REFRESH_INITIATED: When the Initial commands are received for authentication with Key Refresh flag.
- KEY_REFRESH_UNSUCCESSFUL: If any error occurs during the Key Refresh flows and the current state of the device is one of the above, set retry count accordingly.
- KEY_REFRESH_PROCESSED: When Endpoint signature is verified and processed successfully (cloud has old and new keys stored for the device).
- KEY_REFRESH_COMPLETED: When the Perform Handshake is successful, the device entry will have this state (old key will be cleared and only new key will be maintained).

For intermediate step failure, the Endpoint resets back to initialization state for the Key Refresh procedure to be restarted.

6.4.6 Deregistration

Deregistration of the Endpoint and Endpoint reset work together in BLE mode. When the Amazon account user or AWS API user initiates the deregistration process, its authorization is validated first with its Amazon LWA or AWS account credentials respectively. The next step is to update the Endpoint state in AWS IOT wireless from ”registered” to ”provisioned” and cancel Endpoint associations. The Amazon Sidewalk Cloud will mark the internal state of the Endpoint to be in Pending Reset state. The Amazon Sidewalk Cloud continues to hold device keys and authentication material until an Endpoint becomes reachable by the network and successfully resets. A cloud-authenticated RESET message is sent from the cloud to the Amazon Sidewalk Mobile SDK. After the Mobile SDK determines that the Endpoint is not in initialization mode, the Mobile SDK forwards the RESET command to an Endpoint. An Endpoint answers with a cloud-authenticated RESET_RESPONSE message and removes all the credentials and transitions to a provisioned state. The Endpoint starts sending initialization Beacon message, awaiting reregistration. During this time, the Amazon Sidewalk Mobile SDK is scanning for registration Beacon from the Endpoint. After successful registration and Beacon reception, the Amazon Sidewalk Mobile SDK sends Deregister Device message to the cloud proxy service to finalize the deregistration process in the cloud. The Endpoint status is published, credentials are removed, session information is removed and confirmation response is sent back to the Amazon Sidewalk Mobile SDK. The Endpoint state machine is presented in Figure 6.23.

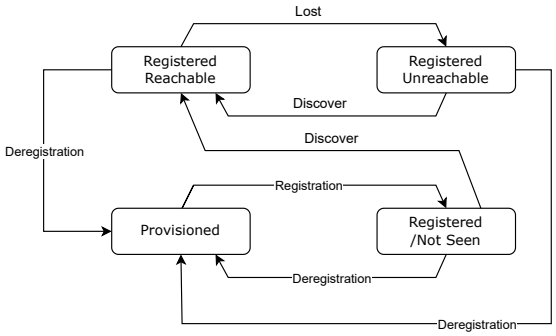


Fig. 6.23 BLE Endpoint state machine

A detailed process flow is presented in Figure 6.26.

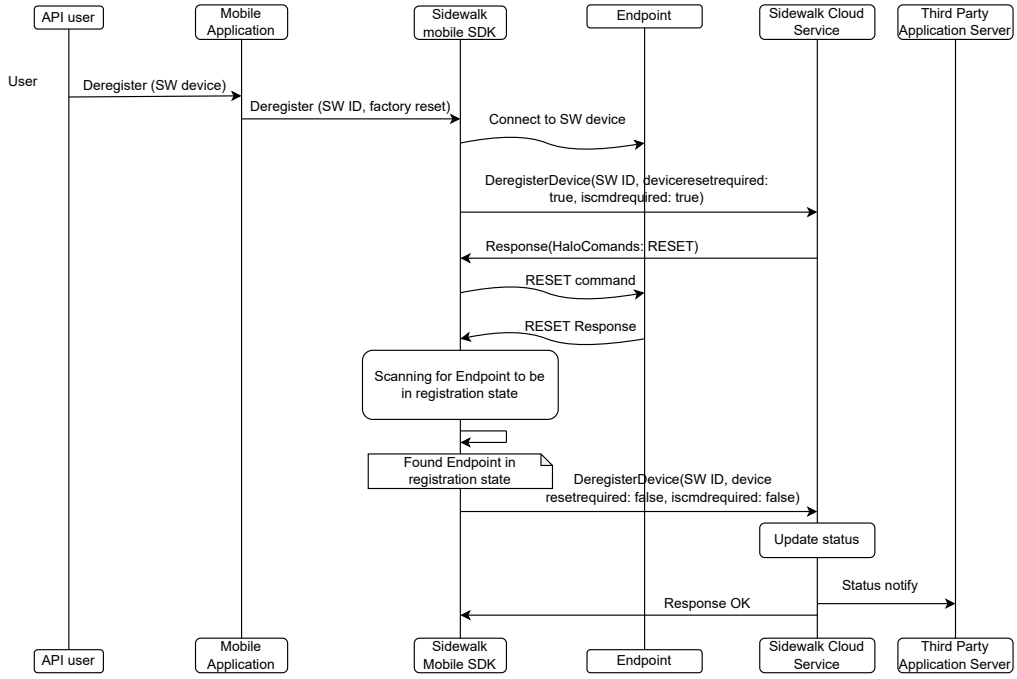


Fig. 6.24 BLE deregistration

If a Mobile is not in the proximity of the Endpoint, then the cloud marks the device state as pending reset and attempts to send a deregistration command via an available Gateway to reach the Endpoint when the Endpoint tries communicating with the Amazon Sidewalk Cloud.

A similar deregistration procedure is followed using a Gateway. When the user deregisters the Endpoint on the Amazon Sidewalk Cloud, the Amazon Sidewalk Cloud marks the Endpoint in Reset Pending state and waits for any uplink message from the Endpoint. When an uplink message is received and the Endpoint is in Reset Pending state, the authenticated RESET message is sent to the Endpoint via a reachable Gateway.

The Endpoint authenticates the RESET command and then sends the RESET_RESPONSE as best effort. The Endpoint shall then remove credentials and shall move to a deregistration state awaiting reregistration.

A detailed process flow is presented in Figure 6.25.

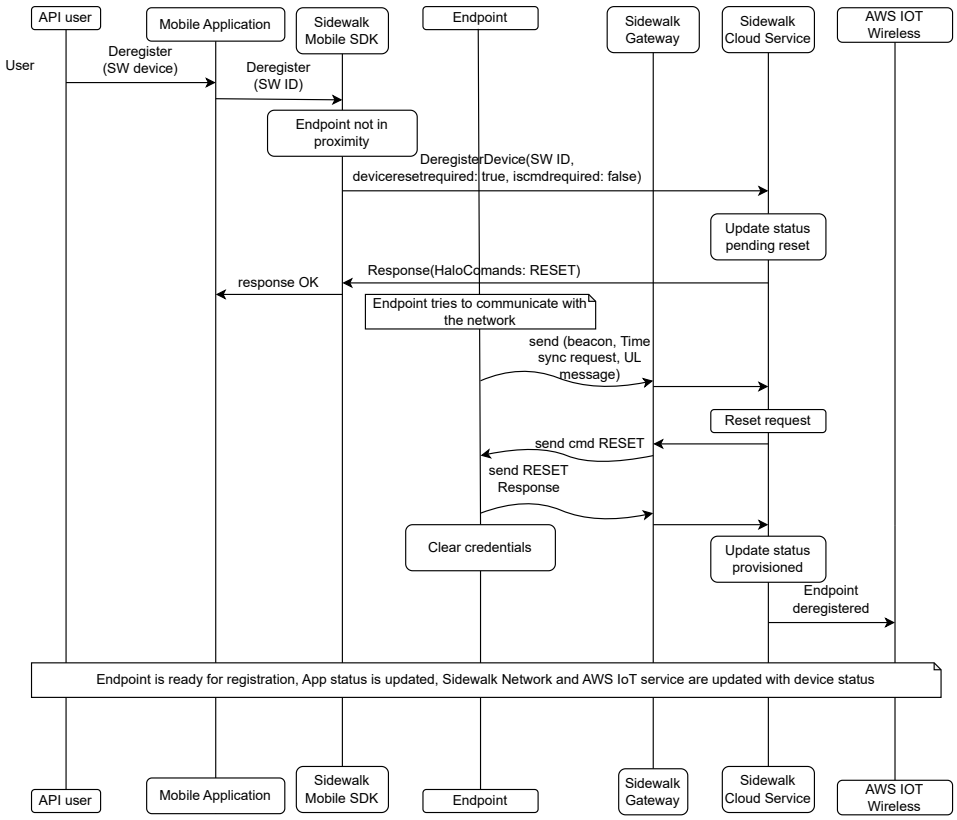


Fig. 6.25 Deregistration by Sidewalk Gateway when Endpoint is not in proximity of mobile application

Figure 6.26 shows the reset message sent might or might not reach the Endpoint. Even if the reset message does not reach the Endpoint, it will be resent upon receiving the first uplink from the Endpoint.

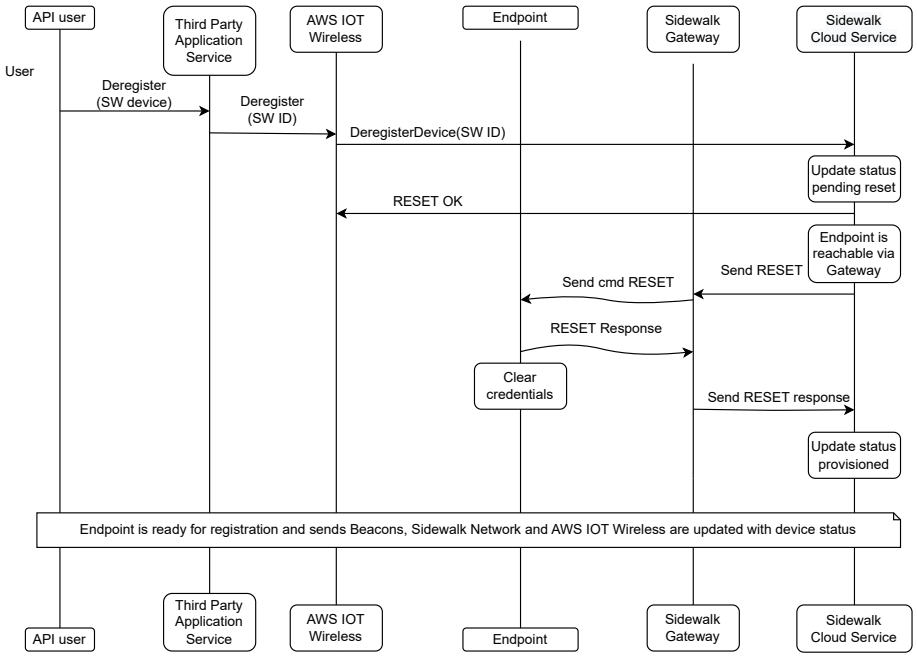


Fig. 6.26 BLE GW deregistration

An Endpoint can trigger deregistration using sid_set_factory_reset API provided to the application. Amazon Sidewalk SDK as best effort shall attempt to send Factory Reset Notify using an available link to the Amazon Sidewalk Cloud via a Gateway and then clear the credentials. The Endpoint shall move to waiting for reregistration state. A detailed process flow is presented in Figure 6.27

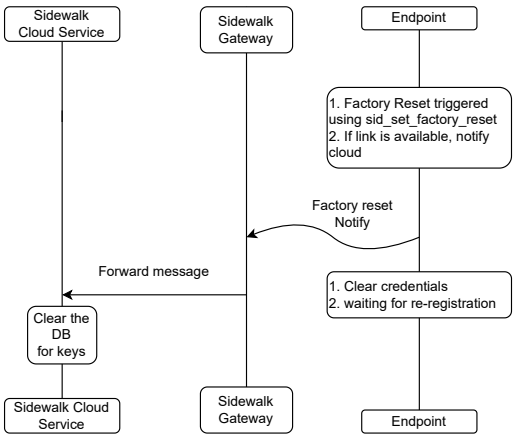


Fig. 6.27 BLE Endpoint Triggered deregistration

Chapter 7

Sidewalk Bulk Data Transfer

7.1 Overview

7.1.1 Purpose and Benefits

Sidewalk Bulk Data Transfer (SBDT) is a foundational software feature designed to support the end-to-end Amazon Sidewalk network architecture for large data transfer to Sidewalk devices. One of the main benefits of the feature is to allow for the operational maintenance and enhancement of Sidewalk devices, however applications are not limited to Over-the-air (OTA) firmware update and SBDT can be used to deliver other file types as well.

A key application of SBDT is enabling developers to remotely deploy new firmware versions onto their devices within the Sidewalk network. This capability is not just a convenience; it's a strategic feature that expands Sidewalk's functionality and network accessibility. With SBDT, file transfers can occur over the Sidewalk network, directly to the end devices, without customer interventions.

The incorporation of OTA updates within Sidewalk-connected devices brings multiple advantages, including enhanced security compliance and compliance to keep the devices up-to-date with the latest firmware while extending their lifecycle and capabilities, cost-efficiency in maintenance from remote operation, and rapid prototyping to introduce a new firmware to their Sidewalk devices.

This initial phase will focus on the fundamental capability to update devices individually, a critical step in establishing the framework's effectiveness. However, this phase will also have its limitations, including one device updating support at a time, no integration with AWS IoT fleet management configuration, and no handling of corner cases, such as recovery from power failures, network interruptions, or cancellations. Another benefit of the file transfer feature is that it can be used for configuration/setting download which provide the ability to change device behavior.

7.2 Design Architecture

The following chapter provides details about the design architecture, data flows, and specific mechanisms of the SBDT feature.

7.2.1 Sidewalk Data Transfer Supported Flows

A successful file transfer (for the purpose of OTA or generic file) requires complex coordination between devices, device firmware, network connectivity, and cloud. The file transfer architecture is chosen to ensure that devices receive the latest file/updates more efficiently and that the process saves both time and processing effort and needs to be scalable to handle fleets of devices. The architecture also depends on the devices

themselves (WAN vs PAN, internal policies) and the availability of the gateways as that dictates the file transfer/OTA flow that can be used to efficiently deliver the updates to the device.

Our approach emphasizes file integrity, safeguarding against image corruption end-to-end during file transfer. The architecture's flexibility is evident in features like real-time transfer status monitoring, allowing users to track and confirm successful updates. Additionally, the capability to schedule file transfers at a predetermined time optimizes network usage.

The diagram below demonstrates the current support flow of Sidewalk bulk data transfer by using the cloud driven flow over the Sidewalk BLE connectivity. The current flows doesn't require any Sidewalk Gateway involvement as it is treated as pass through.

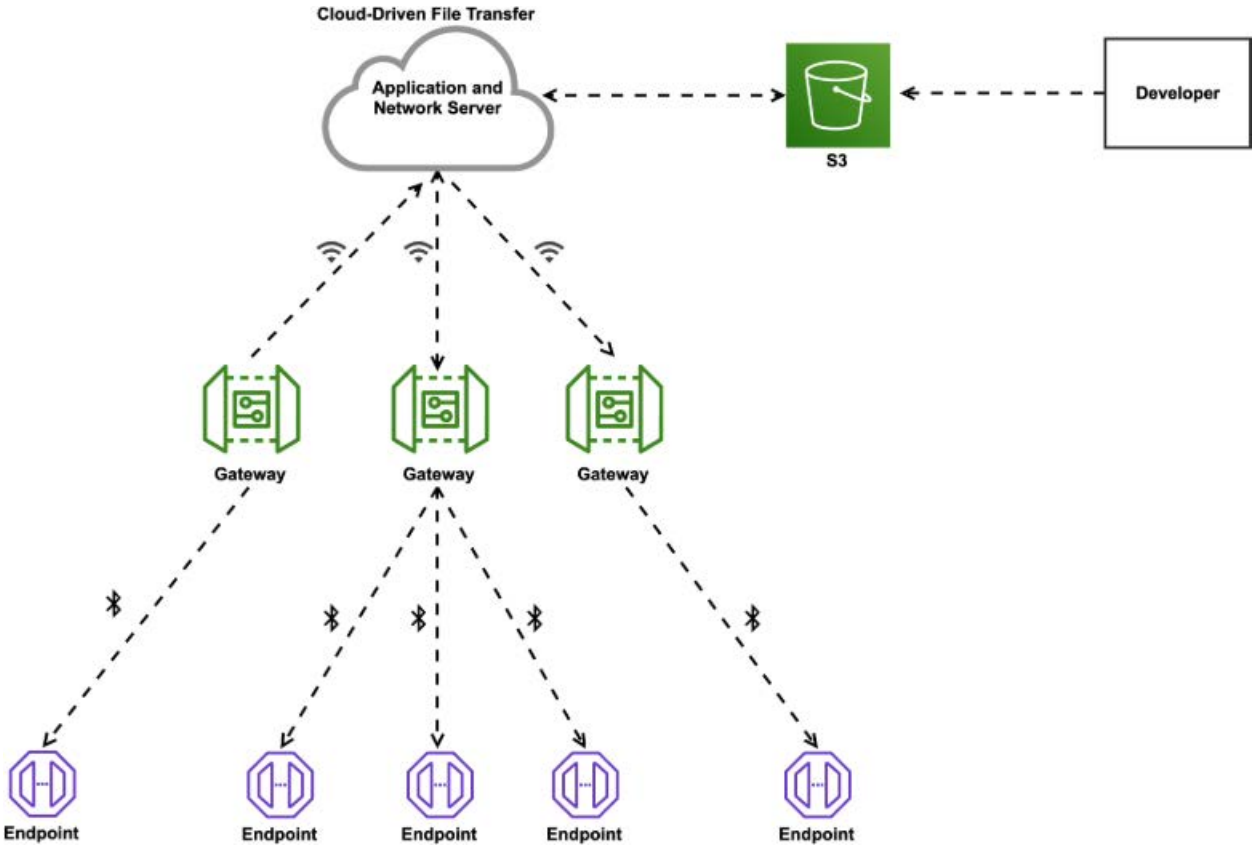


Fig. 7.1 End to End Architecture for SBDT

7.2.2 Messaging

The end to end feature uses a combination of application layer and network layer messaging to deliver the file from the developer application to the end device. All the down link and uplink messages are encrypted and sent uni-cast to each device. The application layer messages are control and handshake messages between the application server and the device and the network layer messages are the data transport messages in the Sidewalk file transfer between the network server and the device.

7.2.3 File Transfer Delivery Mechanisms

The (SBDT) feature incorporates a file transfer protocol designed to handle down link file transfers between cloud device and end nodes on the Sidewalk network. The file transfer is initiated by the developer's appli-

cation when it uploads the File into an S3 bucket and starts a File transfer session through the application server using the AWS IoT FUOTA APIs.

Cloud-to-device transfer are initiated through a File Updates Over The Air (FUOTA) task API, triggered by the cloud application service.

7.2.3.1 Cloud Driven Delivery

For this mechanism, files are delivered to reachable devices as soon as they are uploaded and triggered from AWS IoT Core for Amazon Sidewalk. The cloud immediately attempts to transfer files to target devices that are available at the time of release and devices that are constrained like performing critical activities, low on battery, flash and RAM limited will receive the files the next time they become available. This allows us to deliver to targeted reachable devices and at the same time avoid disrupting active devices.

7.3 End Device

The File transfer feature has a direct impact on end device resources, especially the flash and RAM. The amount of RAM needed depends on the fragmentation size selected by the user. The design of the feature is such that the device doesn't need to have the entire file size memory available to transfer the entire file. The end device also has some scratch internal buffers that are used for receiving the encrypted fragments and decrypted fragments.

The end device is expected to have available flash to host the entire file being transferred and the feature also requires scratch flash storage for storing meta data related to the file transfer.

7.4 Security and Privacy

The data payloads transported using the Sidewalk File Transfer protocol are encrypted and authenticated using the Application layer key and the network layer keys.

The application layer file fragments are encrypted with the per device application layer key and the network layer payload chunks are encrypted with the per device network keys.

This allows for added security even in cases where an end-device might been physically compromised. The protocol also uses a hash integrity checking at the application layer to ensure the overall integrity of the file delivered.

The diagram below describes the security flow of the file fragmented payloads transported between the cloud and the device.

The File is initially fragmented into fragment_size fragments and encrypted by AWS IoT using the per device application keys. A hash of the fragment is also calculated and passed to the network Server. The network server further splits each fragment into MTU size chunks and also encrypts each chunk with the per device network layer keys and sends each chunk to the gateway to the end device.

The received chunks on the end nodes are decrypted and re-assembled into a fragment and decrypted and integrity validated using the hash sent from AWS IoT. Upon successful validation, the fragment is sent to the user application to write to flash and re-assemble the file.

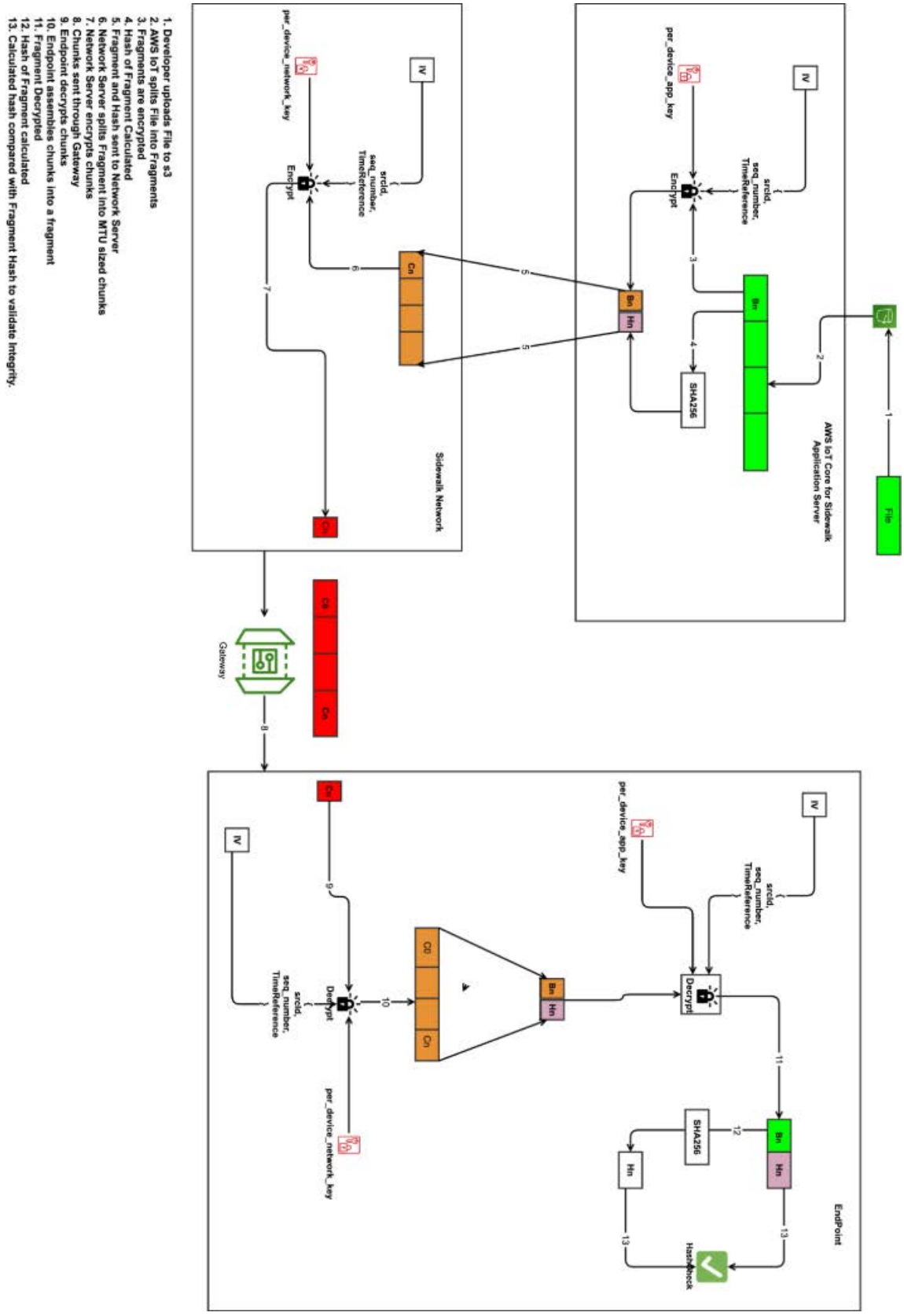


Fig. 7.2 File Fragmentation Flow from AWS IOT to End Device

7.5 Protocol Specification

The following are the control and data commands that are supported in this protocol specification for the transport of a blob of file from the application server and network server to the end device.

All commands are sent uni-cast to the end device.

The diagram below describes the high level messaging flow from Developer to the the device.

Fig. 7.3 High Level Messaging Flow from Developer to End Device

All messages follow the Sidewalk application layer message frame formats below refer to Chapter 3 of specification document for more details.

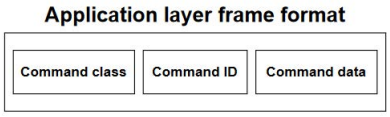


Fig. 7.4 Sidewalk Application Layer Format

7.5.1 SBDT Commands

All SBDT commands share the same command class

Mnemonic	Length	Value	Description
cmd_class	12b	0x03	This value is common for all sbdt control and data commands

Table 7.1 SBDT Commands

7.5.1.1 Application Layer Commands

TLV

Sidewalk TLV format is adopted to encode the command data. The format of the TLV is as below. In TLV encoding, value determines the length, even though type is UINT32, if the value is less than 255 then the length encoded in TLV will be only one byte.

Value Length	Key	Value
2b	6b	xB

Table 7.2 TLV Format 1.

Value Length(2b)	Length	Comments
0b00	1	
0b01	2	
0b10	3	
0b11	length encoded in next byte	[2b(11) — 6b key — 1B value length — xB value]

Table 7.3 TLV Format 2.

Key	Value (6b)	Size
TAG_ID_SBDT_FILE_ID	0x01	4B
TAG_ID_SBDT_NUM_OF_BLOCKS	0x02	2B
TAG_ID_SBDT_BLOCK_SIZE	0x03	2B
TAG_ID_SDBT_BLOCK_ID	0x04	2B
TAG_ID_SBDT_LAST_BLOCK_SIZE	0x05	2B
TAG_ID_SBDT_CHUNK_MTU	0x06	2B
TAG_ID_CHUNK_ID	0x07	1B
TAG_ID_SBDT_WINDOW_SIZE	0x08	1B
TAG_ID_SBDT_BIT_MASK_SIZE	0x09	4B
TAG_ID_SBDT_STATUS	0x0A	1B
TAG_ID_SBDT_AES_KEY	0x0B	16B
TAG_ID_SBDT_META_DATA_OFFSET	0x0C	1B-2B
TAG_ID_SBDT_DATA	0x0D	1B-nB
TAG_ID_SBDT_START_TIME	0x0E	4B
TAG_ID_SBDT_SEQUENCE_NUMBER	0x0F	2B
TAG_ID_SBDT_TIME_REF	0x10	4B
TAG_ID_SBDT_CUSTOMER_REASON	0x11	1B
TAG_ID_SBDT_ACTUAL_BLOCK_SIZE	0x12	2B
TAG_ID_SBDT_NUMBER_OF_CHUNKS	0x13	2B

Table 7.4 TLV Format 3.

Transfer Request

The application transfer request is sent by the application server to the end device to initiate the file transfer session. Below is a description of the frame format.

Mnemonic	Length	Value	Type	command_name	Direction
cmd_id	14b	0x01	Write	TRANSFER_APPLICATION_REQUEST	DL
	14b	0x03	Response	TRANSFER_APPLICATION_RESPONSE	UL

Table 7.5 Command ID

Mnemonic	Length	Value	TLV	Description
sbdt_version	1B	0	fixed	sdbt version
file_id	4B	0	0x81	uniquie file id crc32 of file
fragment_size	1B-2B	1K-8K	0x03	user supplied fragment size range from 1K to 8K
	1B-2B	1K-8K	0x43	
number_of_fragments	1B-2B	variable	0x02	number of fragment_size fragments in a file
	1B-2B	variable	0x42	
last_fragment_size	1B-2B	variable	0x05	size of the last fragment of the file
	1B-2B	variable	0x45	
start_time	4B	variable	0x8E	time to start the transfer
file_descriptor	1B-32B	variable	0x0	user supplied file meta data
	1B-32B	variable	0x4	
	1B-32B	variable	0x8	
	1B-32B	variable	0xc	

Table 7.6 Command Data - TRANSFER_APPLICATION_REQUEST

The application transfer request response is sent by the end device to application server as a response to the application transfer request. Below is a description of the frame format.

Mnemonic	Length	Value	TLV	Description
sbdt_version	1B	0		sdbt version
file_id	4B	0	0x81	uniquie file id crc32 of file
fragment_id	1B-2B	variable	0x04	current fragment index
	1B-2B	variable	0x44	
sbdt_status	1B	variable	0x0A	sdbt error status code

Table 7.7 Command Data - TRANSFER_APPLICATION_RESPONSE

Transfer File Finalize

The transfer file finalize command is sent by the application server to the end device to exit the file transfer session. Below is a description of the frame format.

Mnemonic	Length	Value	Type	command_name	Direction
cmd_id	12b	0x21	Write	TRANSFER_FILE_FINALIZE	DL
	12b	0x23	Response	TRANSFER_FILE_FINALIZE_RESPONSE	UL

Table 7.8 Command ID

Mnemonic	Length	Value	TLV	Description
sbdt_version	1B	0		sdbt version
file_id	4B	0	0x81	uniquie file id crc32 of file

Table 7.9 Command Data - TRANSFER_FILE_FINALIZE

The transfer file finalize response is sent by the end device to application server as a response to the transfer file finalize request. Below is a description of the frame format.

Mnemonic	Length	Value	TLV	Description
sbdt_version	1B	0		sdbt version
file_id	4B	0	0x81	uniquie file id crc32 of file
sbdt_status	1B	variable	0x0A	sdbt error status code

Table 7.10 Command Data - TRANSFER_FILE_FINALIZE_RESPONSE

Transfer Cancel

The transfer cancel command is sent by the application server to the end device to cancel an ongoing transfer session. Below is a description of the frame format.

Mnemonic	Length	Value	Type	command_name	Direction
cmd_id	12b	0x2D	Write	TRANSFER_CANCEL	DL
	12b	0x2F	Response	TRANSFER_CANCEL_RESPONSE	UL
	12b	0x2E	Notify	TRANSFER_CANCEL_NOTIFY	UL

Table 7.11 Command ID

Mnemonic	Length	Value	TLV	Description
sbdt_version	1B	0		sdbt version
file_id	4B	0	0x81	uniquie file id crc32 of file

Table 7.12 Command Data - TRANSFER_CANCEL

The transfer cancel response command is sent by the end device to the application server as a response to the cancel request. Below is a description of the frame format.

Mnemonic	Length	Value	TLV	Description
file_id	4B	0	0x81	uniquie file id crc32 of file
sbdt_status	1B	variable	0x0A	sdbt error status code

Table 7.13 Command Data - TRANSFER_CANCEL_RESPONSE

The transfer cancel notify command is sent by the end device to the application server to cancel an ongoing transfer session. Below is a description of the frame format.

Mnemonic	Length	Value	TLV	Description
file_id	4B	0	0x81	uniquie file id crc32 of file
sbdt_status	1B	variable	0x0A	sdbt error status code
cancel_reason	1B	variable	0x11	cancel reason

Table 7.14 Command Data - TRANSFER_CANCEL_NOTIFY

7.5.1.2 Network Layer Commands

Transfer Request

The transfer request command is sent by the network server to the end device to initiate the file transfer session. Below is a description of the frame format.

Mnemonic	Length	Value	Type	command_name	Direction
cmd_id	12b	0x05	Write	TRANSFER_REQUEST	DL
	12b	0x07	Response	TRANSFER_RESPONSE	UL

Table 7.15 Command ID

Mnemonic	Length	Value	TLV	Description
sbdt_version	1B	0		sdbt version
file_id	4B	0	0x81	uniquie file id crc32 of file
fragment_size	1B-2B	1K-8K	0x03	user supplied fragment size range from 1K to 8K
	1B-2B	1K-8K	0x43	
chunk_size	1B-2B	variable	0x06	size of link mtu chunks
	1B-2B	variable	0x46	
window_size	1B	variable	0x08	size of fragment window

Table 7.16 Command Data - TRANSFER_REQUEST

The transfer request response is sent by the end device to network server as a response to the transfer request. Below is a description of the frame format.

Mnemonic	Length	Value	TLV	Description
sbdt_version	1B	0		sdbt version
file_id	4B	0	0x81	uniquie file id crc32 of file
fragment_size	1B-2B	1K-8K	0x03	user supplied fragment size range from 1K to 8K
	1B-2B	1K-8K	0x43	
fragment_id	1B-2B	variable	0x04	current fragment index
	1B-2B	variable	0x44	
number_of_fragments	1B-2B	variable	0x02	number of fragment_size fragments in a file
	1B-2B	variable	0x42	
chunk_size	1B-2B	variable	0x06	size of link mtu chunks
	1B-2B	variable	0x46	
window_size	1B	variable	0x08	size of fragment window
number_of_chunks	1B-2B	variable	0x13	number of chunks per fragment
	1B-2B	variable	0x93	
sbdt_status	1B	variable	0x0A	sdbt error status code

Table 7.17 Command Data - TRANSFER_RESPONSE

Transfer Request Finalize

The transfer request finalize command is sent by the network server to the end device to finish the transfer request handshaking to initiate the file transfer session. Below is a description of the frame format.

Mnemonic	Length	Value	Type	command_name	Direction
cmd_id	12b	0x09	Write	TRANSFER_REQUEST_FINALIZE	DL
	12b	0x0B	Response	TRANSFER_REQUEST_FINALIZE_RESPONSE	UL

Table 7.18 Command ID

Mnemonic	Length	Value	TLV	Description
sbdt_version	1B	0		sdbt version
file_id	4B	0	0x81	uniquie file id crc32 of file
chunk_size	1B-2B	variable	0x06	size of link mtu chunks
	1B-2B	variable	0x46	
window_size	1B	variable	0x08	size of fragment window

Table 7.19 Command Data - TRANSFER_REQUEST_FINALIZE

The transfer request finalize response command is sent by the end device to the network server to complete the transfer request handshaking. Below is a description of the frame format.

Mnemonic	Length	Value	TLV	Description
sbdt_version	1B	0		sdbt version
file_id	4B	0	0x81	uniquie file id crc32 of file
sbdt_status	1B	variable	0x0A	sdbt error status code

Table 7.20 Command Data - TRANSFER_REQUEST_FINALIZE_RESPONSE

Transfer Block

The transfer block command is sent by the network server to the end device to transport a fragment of the file. Below is a description of the frame format.

Mnemonic	Length	Value	Type	command_name	Direction
cmd_id	12b	0x15	Write	TRANSFER_BLOCK_DATA	DL
	12b	0x17	Response	TRANSFER_BLOCK_DATA_RESPONSE	UL

Table 7.21 Command ID

Mnemonic	Length	Value	TLV	Description
header	1B			block header info ()
file_id	4B	0	0x81	uniquie file id crc32 of file
chunk_id	1B-2B	variable	0x07	current chunk index
	1B-2B	variable	0x47	
fragment_id	1B-2B	variable	0x04	current fragment index
	1B-2B	variable		
seq_num	12B	variable	0x4F	fragment sequence number
chunk_data	1B-nB	variable	0x0D	file payload
	1B-nB	variable	0x4D	
	1B-nB	variable	0x8D	
	1B-nB	variable	0xCD	

Table 7.22 Command Data - TRANSFER_BLOCK_DATA

The transfer block response command is sent by the end device to the network server to acknowledge the transfer of a fragment of the file. Below is a description of the frame format.

Mnemonic	Length	Value	TLV	Description
header	1B			block header info ()
file_id	4B	0	0x81	uniquie file id crc32 of file
fragment_id	1B-2B	variable	0x04	current fragment index
	1B-2B	variable		
seq_num	12B	variable	0x4F	fragment sequence number
bitmask_size	1B-4B	variable	0x09	bitmask size
	1B-4B	variable	0x49	
	1B-4B	variable	0x89	
	1B-4B	variable	0xC9	
sbdt_status	1B	variable	0x0A	sdbt error status code
bitmask	1B-nB	variable		received data bitmask
	1B-nB	variable		
	1B-nB	variable		
	1B-nB	variable		

Table 7.23 Command Data - TRANSFER_BLOCK_DATA_RESPONSE

Transfer Exit

The transfer exit command is sent by the network server to the end device to exit the transfer session on a completed transfer. Below is a description of the frame format.

Mnemonic	Length	Value	Type	command_name	Direction
cmd_id	12b	0x36	Notify	TRANSFER_FILE_EXIT	DL

Table 7.24 Command ID

Mnemonic	Length	Value	TLV	Description
file_id	4B	0	0x81	uniquie file id crc32 of file

Table 7.25 Command Data - TRANSFER_FILE_EXIT

7.5.2 Status

The following lists the status information returned by SBDT.

Value	Definition (FuotaDeviceStatus)	Description
0x00	`Successful`	Success
0x01	`Wrong_descriptor`	Genric Error
0x02	`Wrong_descriptor`	Version mismatch
0x03	`Package_Not_Supported`	File too Big
0x04	`Not_enough_memory`	No Memory
0x05	`Low_Battery`	Low Battery
0x06	`Security_Error`	Block Verification Failed
0x07	`MissingFrag`	Missing Chunks
0x08	`File_System_Error`	Write Failed
0x09	`Security_Error`	File Verification Failed
0x0A	`Restart_Transfer`	Restart
0x0B	`Already_Available`	File Exists
0x0C	`Link_Not_Supported`	Unsupported Link
0x0D	`Internal_Error`	Invalid File ID
0x0E	`Internal_Error`	Invalid Block Size
0x0F	`MissingFrag`	Invalid Block ID
0x10	`Initial`	Ready Status
0x11	`InProgress`	InProgress

Table 7.26 SBDT Status Information

Chapter 8

Procedures and flow charts

8.1 Power up

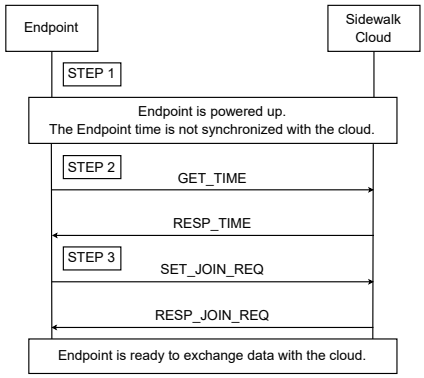


Fig. 8.1 Power up

The flow chart specifies the command sequence that shall be exchanged between the Endpoint and the Amazon Sidewalk Cloud to make an Endpoint fully operational. The procedure is as follows:

1. STEP 1 - The Endpoint powers up and initializes Amazon Sidewalk SDK with one of the three types of connection modes, i.e. BLE, SubG-CSS or SubG-FSK connection modes. The parameters of the selected connection mode, stored in the Endpoint's non-volatile memory, are loaded back. These parameters can be modified to desired values by resetting them through Amazon Sidewalk APIs. In the next boot-up,
2. STEP 2 - Amazon Sidewalk is started through the sid_start instruction. This instruction triggers the protocol to start forming the initial communication link. Depending on the connection mode that the Endpoint is initialized as follows:
   (a) in the case of BLE connection mode, the instruction initializes the components for BLE communication, starts sending out advertisement packages and waits for connection initiation requests from nearby Gateways. The BLE communication link is established via a handshake request by an Amazon Sidewalk Gateway responding to the advertisement package as described in Section 6.2;
   (b) in the case of the SubG-FSK connection mode, the instruction starts a Beacon Discovery operation as described in Section 5.2.1 "Beacon discovery". Upon discovering the Gateway, SubG-FSK connection with the discovered Gateway is established using *"Common Slots"* schedule as described in section 5.5.1.1.1;

(c) in the case of the SubG-CSS connection modes, the instruction initializes the components to send SubG-CSS packets but does not send or receive any packets.

3. STEP 3 - The Endpoint shall establish time synchronization with the Amazon Sidewalk Cloud by using GET_TIME and RESP_TIME command pairs. These commands use counter-based keys (see details in Section 4.2 “Global clock synchronization”, and Section 4.5 “Key management”, and further details of this process are described in . The Endpoint proceeds with Step 4 after establishing time synchronization.

4. STEP 4 - The Endpoint initiates the Amazon Sidewalk join procedure by using set_join_req and resp_join_req command pairs as described in Section 8.3. Upon completing this step, the Endpoint reports Amazon Sidewalk Connected state to its app. Once connected, the Endpoint allows uplink and downlink data message flows between application servers and the Endpoint through the Amazon Sidewalk Cloud.

5. STEP 5 - The Endpoint starts maintaining Amazon Sidewalk connectivity. This maintenance involves:

   (a) link layer connectivity between *Sidewalk Relay* and the Endpoint in the connection mode being used including BLE, 900 MHz synchronous or asynchronous modes.

   (b) GCS synchronization maintenance through periodic updates as described in Section 8.2.1,

   (c) Amazon Sidewalk connectivity maintenance through periodic Network Synchronization messages as described in Section 5.6.

8.2 Global Clock Synchronization

8.2.1 Global clock synchronization - Bootup and maintenance

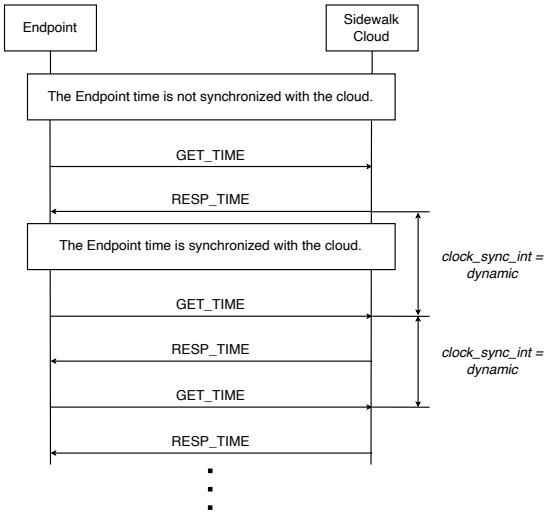


Fig. 8.2 Time sync maintenance

The flow chart specifies the commands sequence that shall be exchanged between the Endpoint and the Amazon Sidewalk Cloud to keep the Endpoint clock synchronization. Network clock synchronization is required for messaging features as described in Section 4.7. The clock is synchronized during procedures described at boot-up time and maintained throughout the operation. The maintenance of the established clock synchronization shall be done by using GET_TIME and RESP_TIME commands, each *clock_sync_int*. The Endpoint shall not update its clock value in case the received value does not differ more than *sec_clock_drift*

seconds. When there is no response for the GET_TIME command then the Endpoint shall perform a next try to synchronize the clock after the *clock_sync_int* period which is equal to two hours. When the response for the GET_TIME command is successfully received by the Endpoint, the *clock_sync_int* period dynamically adapts based on the time difference between the received time in RESP_TIME and the Endpoint's current time. This is described in Section 8.2.2. There is no special procedure to handle lack of RESP_TIME.

8.2.2 Global clock synchronization - maintenance interval

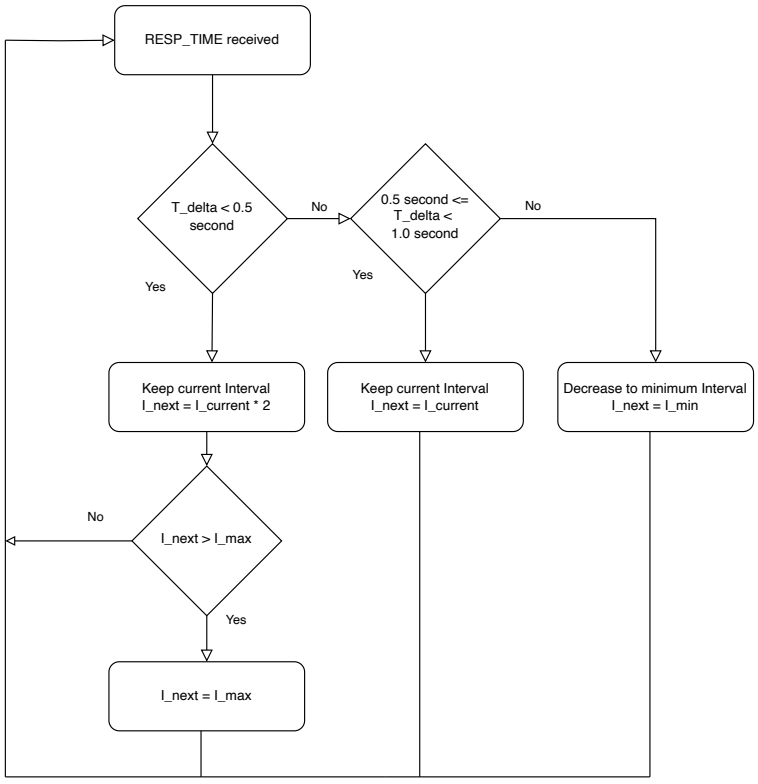


Fig. 8.3 Time sync maintenance Interval

The next time sync interval is updated dynamically based on the difference between the current time on the Endpoint and the time in RESP_TIME

$I\_next = I\_current * 2$, when $I\_next <= I\_current$, If $T\_delta <= 0.5$ second

$I\_next = I\_current$, If 0.5 second $< T\_delta <= 1$ second

$I\_next = I\_min$, If $T\_delta >= 1$ second

1. I_next: Next time sync interval.
2. $I\_current$: Current time sync interval.
3. I_max: Maximum time sync interval (12 hours).
4. I_max: Minimum time sync interval (2 hours).
5. T_delta: Time difference between current time on Endpoint and received in RESP_TIME.

The flow chart specifies the backoff procedure followed to determine the next *clock_sync_int* period only in the case of successfully receiving the time from RESP_TIME. The procedure for determining the next *clock_sync_int* period is as follows:

1. The Endpoint starts the clock synchronization process. if there is sucessful response RESP_TIME then *clock_sync_int* period is set initially to 2 hours.
2. After 2 hours clock synchronization process is attempted and if RESP_TIME is received and the difference between the current time and time in the RESP_TIME is less than one second, *clock_sync_int* period is set to 4 hours.
3. successful RESP_TIME reception at subsequent *clock_sync_int* periods would result in update of *clock_sync_int* period as described below
   (a) Initial *clock_sync_int* = 2 hours
   (b) Next *clock_sync_int* = 4 hours
   (c) Next *clock_sync_int* = 8 hours
   (d) Next *clock_sync_int* = 12 hours
4. *clock_sync_int* period does not exceed 12 hours and every 12 hours clock synchronization process is attempted in case of successful reception of RESP_TIME and the time difference between the received time in
5. At any step of the *clock_sync_int* period, the difference between the current time and time in the RESP_TIME is greater than 1 second, the *clock_sync_int* period is reset to 2 hours
6. Failure to receive RESP_TIME in response to GET_TIME command would start the Global Clock Synchronization Section 8.2.3.
   (a) The SubG-CSS device shall continue from STEP 3 of Section 8.1.
   (b) The SubG-FSK device shall complete Section 8.1.

8.2.3 Global clock synchronization - retry

The flow chart specifies the commands sequence that shall be exchanged between the Endpoint and the Amazon Sidewalk Cloud to synchronize the Endpoint clock in case there is no response from the cloud for the first GET_TIME command. The procedure is included in STEP 2 and STEP 3 of Section 8.1. The procedure for handling the clock synchronization failure is as follows:

1. STEP 1 - The Endpoint initializes the clock synchronization process but it fails. There is no response RESP_TIME for the GET_TIME command for *sync_resp_timeout* equal to 3 min.
2. STEPs 2 - 4 - *retry_count* is initialized with 1 value and the next GET_TIME command is sent. When the response is not received for *sync_resp_timeout* then the activity is repeated in STEP 3 and STEP 4. The *retry_count* is increased by 1 up to its maximum value and *sync_resp_timeout* is equal to 3 min. When the *retry_count* reached its maximum value then go to STEP 5.
3. STEPs 5 - 8 - the *sync_resp_timeout* time interval is increased in the subsequent steps. The Endpoint waits a longer time for the possible response from the cloud. The value of the *sync_resp_timeout* is as follows in the next steps:
   (a) STEP 5 - *sync_resp_timeout* = 5 min
   (b) STEP 6 - *sync_resp_timeout* = 8 min
   (c) STEP 7 - *sync_resp_timeout* = 10 min

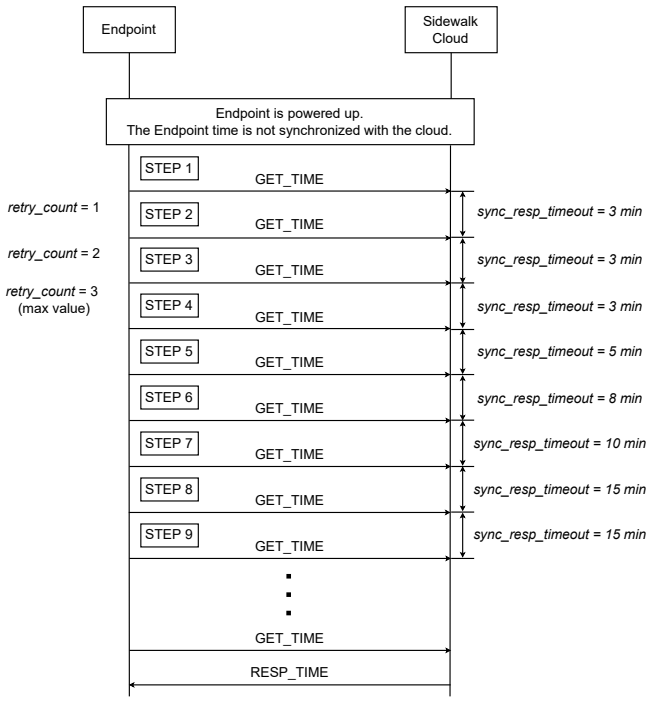


Fig. 8.4 Time sync - no response scenario

(d) STEP 8 - *sync_resp_timeout* = 15 min

4. STEP 9 - When there is still no response from the cloud then *sync_resp_timeout* is not increased. The Endpoint shall wait for a maximum value of *sync_resp_timeout* minutes for the response from the cloud for each subsequent GET_TIME command. The Endpoint shall send GET_TIME command each *sync_resp_timeout* till the moment the RESP_TIME command is received and the time is synchronized.

   (a) The SubG-CSS device shall continue from STEP 3 of Section 8.1.

   (b) The SubG-FSK device shall complete Section 8.1.

8.3 Join and Network Sync Procedures

Join process establishes a connection with the Amazon Sidewalk Cloud through the *Sidewalk Relay*. This process involves a secure command exchange, namely *Join-Req Message* on the uplink and *Join-Resp Message* on the downlink, following *Network Encryption Model*. Via this packet exchange, the Endpoint:

1. notifies the Amazon Sidewalk Cloud that the Endpoint is active and connected through *Sidewalk Relay* or *Sidewalk Relay*s;

2. notifies the Amazon Sidewalk Cloud that it is using a chosen communication mode, profile and parameters indicated through the payload of *Join-Req Message*;

3. checks the existence of a bidirectional communication path between the cloud and the Endpoint.

The *Join-Req Message* and *Join-Resp Message* exchange is performed not only the first time the Endpoints joins Amazon Sidewalk but also periodically thereafter. The initial exchange is called a Network Join and the periodical exchanges are referred to as Network Sync. The information carried in the command payload varies depending on the use case, i.e. Network Join or Network Sync, as well as the Amazon Sidewalk communication mode and profile configuration of the Endpoint. The contents of *Join-Req Message* and

Join-Resp Message are listed in Table 3.1 and Table 3.2, respectively. The command payloads are encoded using a TLV(type-length-value) format. The type field indicates the type of information content, the length field indicates the size of the information content, and the value is the payload.

8.3.1 Join Request

8.3.2 Join Request and retries

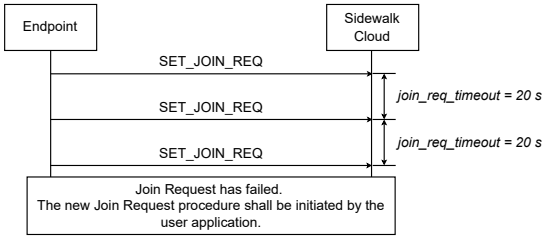


Fig. 8.5 Join Request no response

The Endpoint needs to complete the join process before reporting a connected status to the application. After sending a *Join-Req Message*, the Endpoint waits for a *Join-Resp Message* and if not received within a timeout interval, repeats the process. The timeout interval is 20 s for SubG-CSS connection mode and a random value [between 20 s and 30 s] for SubG-FSK connection mode.

The flow chart specifies the commands sequence that shall be exchanged between the Endpoint and the Amazon Sidewalk Cloud to handle the lack of RESP_JOIN_REQUEST for the SET_JOIN_REQ on the Endpoint side. The procedure is as follows:

1. The Endpoint transmits the SET_JOIN_REQ command.
2. The Endpoint waits for the RESP_JOIN_REQ:
   (a) if the response from the cloud is received by the Endpoint then it finishes the commands exchange successfully
   (b) if the response from the cloud is not received by the Endpoint then
      i. if the number of SET_JOIN_REQ commands sent subsequently is less than three then wait for *join_req_timeout* time and continue from bullet 1
      ii. if the number of SET_JOIN_REQ commands sent subsequently is equal to three then go to bullet 3
3. There is no response for three subsequent SET_JOIN_REQ commands so the procedure is stopped with failure.

A maximum of three consecutive failures or receiving a *Join-Resp Message* with a failure code leads to a failure to join indication. Handling of this failure varies depending on the connection mode:

- in the case of SubG-FSK connection mode, the device returns to the Link Connection Establishment Phase by triggering a new Gateway Discovery for finding a new *Sidewalk Relay*; and
- in the case of SubG-CSS connection mode, the device stays in a disconnected state until a new join procedure is triggered by the user application via 'sid_start' API.

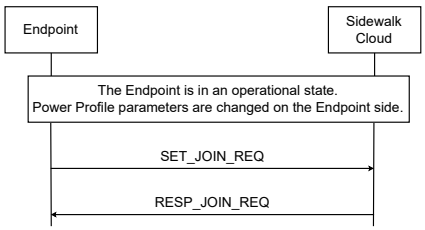


Fig. 8.6 Power profile parameter change

8.3.3 Connection Profile parameters change

Commands SET_JOIN_REQ and RESP_JOIN_REQ are not only used in Section 8.1. This set of commands shall be used every time the Power Profile configuration is changed on the Endpoint side. It shall also be used in case when the Power Profile is changed - e.g. the Endpoint switches from Profile A to Profile B.

Glossary

ACK	Acknowledgement
AES-GCM	Advanced Encryption Standard Galois/Counter Mode
AES-CTR	advanced encryption standard counter mode
AES-CMAC	Advanced Encryption Standard Cipher Message Authentication code
AMA	Alexa Mobile Accessory
APID/API	Advertised Product ID
ASL	Application Support Layer
BB	baseband
BLE	Bluetooth Low Energy
BoardID	type of device build
BT	bandwidth-time product
CA	Certificate Authority
Chunk	MTU Size Data Payload Transported over a Link.
CMAC	Cipher-based Message Authentication Code
CR	Coding Rate
CRC	Cyclic Redundancy Check
CS-UL	Cloud Server Uplink
CS-DL	Cloud Server Downlink
CTRREF	counter-based reference symmetric key
D2D	Device to Device
DAK	Device Attestation Key
DHAv2	Device Hardware Authentication - Version 2
DL	Downlink
DMS	Device Management Service
DSN	Device Serial Number
DUT	Device Under Test
EC	Elliptic Curve
ECDH	Elliptic Curve Diffie-Hellman
FCS	Frame Check Sequence
FEC	Forward Error Correction
FFN	Frustration-Free Network
FIFO	First In, First Out
Fragments	Smaller Blocks of a File.
FSK	Frequency Shift Coding
FW	firmware

GATT	Generic Attribute profile
GFSK	Gaussian Frequency-Shift Keying
GPS	Global Positioning System
HDR	High Data Rate
HDX	Half-Duplex
HSM	Hardware Security Module
ID	identifier
IoT	Internet of Things
ISM	radio bands reserved for Industrial, Scientific and Medical purposes
KEX	Key Exchange Flow.
LAN	Local Area Network
LDR	Low Data Rate
LFSR	Linear Feedback Shift Register
LoRa	LoRa (derived from Long Range) is a proprietary chirp spread spectrum radio modulation technique owned by Semtech.
MAC	Media Access Control
MCU-SDK	Micro Controller Unit Software Development Kit
MTU	Maximum Transmission Unit.
OpenSSL	Open Secure Sockets Layer, an open source software library
PAN	Personal Area Network
PHDR	Physical layer Header
PHY	Physical layer
PKI	Public Key Infrastructure
Rb	bit rate
RF	Radio Frequency
RNG	Random Number Generator
RSVD	Reserved
RX	Receive operation
SBDT	Sidewalk Bulk Data Transfer.
SDK	Software Development Kit
SF	Spreading Factor
SFD	Start of Frame Delimiter
SMSN	Sidewalk Manufacturing Serial Number
SR	Symbol Rate
SubG-CSS	Amazon Sidewalk asynchronous transport using Semtech's LoRa chirp spread spectrum (CSS) modulation in the sub-GHz ISM band. Note that this is also referred to as Low Data Rate (LDR), CSS, LoRa, and asynchronous. Variable names in code may reflect the alternative names.
SubG-FSK	Amazon Sidewalk synchronous transport using GFSK modulation in the sub-GHz ISM band. Note that this is also referred to as High Data Rate (HDR), 900MHz, FSK, GFSK and synchronous. Variable names in code may reflect the alternative names.
TLV	Tag Length Value.
TX	Transmit operation
TXID	Transmitter/Transmission Identifier
UL	Uplink
Wi-Fi	high speed wireless data transmission

Appendix A

Command List

A.1 SubG-FSK Mode Connection Profile 2 Command List

ABRV/ **Command Class/** **ID/** **OP Code/** **Source/** **Destination/** **Encryption**	**Information Content**
JOIN-REQ 0x0 / 0x22F / 0x1 / End Device / Destination/ SID_SEC_KEY_WAN_GCM_128	JOIN-REQ message carries requested link characteristics for both uplink and downlink messaging expected on the end device. These characteristics are MAC level parameters set for each product based on the application requirements and the device profile. 1. Uplink Link-Latency Characteristics: Describes the time interval between two uplink opportunities. The value is described in units of slot interval, i.e. 63 ms. 2. Uplink Load Characteristics: Expected traffic rate for each uplink opportunity. The load is mapped to a range of values in interval [0-255], where 0 indicates lowest load and 255 indicates highest load of 1 packet per uplink opportunity. 3. Downlink Link-Latency Characteristics: Describes the time interval between two downlink opportunities. The value is described in units of slot interval, i.e. 63 ms. 4. Downlink Load Characteristics: Expected traffic rate for each DL opportunity. The load is mapped to a range of values in interval [0-255], where 0 indicates lowest load and 255 indicates highest load of 1 packet per downlink opportunity. 5. GWID 6. Link Allocation duration
JOIN-RESP 0x0 / 0x22F / 0x3 GW\ End Device / SID_SEC_KEY_WAN_JOIN_RESP_GCM_128	JOIN-RESP message carries assigned schedule, i.e. A-UL and A-DL for the FSK-WAN-EN. A-UL parameters define the time instances and the channel the Endpoint is allowed to transmit on. Similarly, A-DL parameters define the time instances and channels the Endpoint should expect a downlink message. 1. Request Accepted/Rejected (If rejected no other information should be there) 2. Uplink Schedule: Offset and Periodicity 3. Uplink Contention Param: Sublot window size, Slot window size 4. Downlink Schedule: Offset and Periodicity 5. Link Allocation duration
AUTH-RESP 0xFF / 0xD / 0x1 Cloud / GW/ SID_SEC_KEY_WAN_GCM_128	1. A boolean accepting and rejecting the request 2. JOIN-REQ contents a. R-TX-ID of the requesting Endpoint b. UL 3. Uplink Link-Latency Characteristics: a. Uplink Load Characteristics b. DL 4. Downlink Link-Latency Characteristics 5. Downlink Load Characteristics 6. One time key to be used for downlink encryption/authentication 7. Link Allocation duration 8. SID_SEC_KEY_WAN_JOIN_RESP_GCM_128 key of the Endpoint

Table A.1 Information content of the synchronous mode profile's connection establishment commands

Appendix B

Regional parameters

B.1 Parameters for SubG-FSK connection mode with 50 kbps data rate

Parameters introduced in Section 5.2.

Name	Value	Description
slot_int	63 ms	The time interval between two subsequent Target Transmission Times.
bcn_interval	10080 ms	The time interval between start of the transmission of two subsequent beacons.
nb_slot	160	The number of Target Transmission Times (counted from 0 to 159). The number of Target Transmission Times is equal to the number of slots.
ul_dl_nb_slot	156	The number of Target Transmission Times that may be used for uplink and downlink scheduling. *ul_dl_nb_slot* = *nb_slot* - 4 The number of Target Transmission Times that are reserved for the Beacon broadcasting is equal to 4 (slot number 159, 0, 1, 2). These cannot be used for Endpoint uplink, downlink transmission.
last_ul_dl_nb_opp_index	3	The index of the first UL/DL opportunity within a *bcn_interval*.
first_ul_dl_nb_opp_index	158	The index of the last UL/DL opportunity within a *bcn_interval*.
nb_opp	160	The number of opportunities (counted from 0 to 159).
tr_det_listening	0.64 ms	Time period used for the preamble detection.
ch_change	1.36 ms	The time period used for channel change during the Beacon discovery.
nb_channels	69	The total number of channels used for SubG-FSK communication (counted from 0 to 68).
bcn_prm_dur	140 ms	The Beacon preamble duration.
nb_bcn_channels	69	The number of random channel indexes of selected channels for each Beacon hopping sequence. Each Beacon hopping sequence utilizes all available channels used in SubG-FSK communication (*nb_channels*).
nb_neighb_bcn	2	The number of channels used in before and after the current channel for the Beacon transmission.
nb_ul_dl_channels	64	The number of random channel indexes of selected channels for each uplink and downlink hopping sequence. *nb_ul_dl_channels* = *nb_channels* - 2**nb_neighb_bcn* - 1
sub_slot_nb	6	The number of sub-slots in the contention window.
sub_slot_time	2 ms	The duration of a sub-slot in the contention mechanism.
contend_time	12 ms	The contention window duration in the contention mechanism. *contend_time* = *sub_slot_nb* * *sub_slot_time*
nb_tx_retry	3	The default number of transmission retries in the contention mechanism.
CS-UL	offset 8, period 50	The common slot schedule defined by Sidewalk GWs in uplink direction.
CS-DL	offset 9, period 50	The common slot schedule defined in Sidewalk GWs in downlink direction.

B.2 Parameters for SubG-CSS connection with 2 kbps data rate

Parameters introduced in Section 5.3.

Name	Value	Description
ul_separation	367 ms	Separation of uplink Target Transmission Times.
dl_separation	5 s	Separation of downlink Target Transmission Times. The value of the parameter is also defined in Section 5.2
nb_ul_slot	8	The number of uplink slots.
nb_slot	9	The total number of downlink and uplink slots in a dl_separation period. *nb_slot* = *nb_ul_slot* + 1
nb_dl_slot	5, 10, 15, 20, infinite	The number of downlink slots after the last uplink. Numbers 5, 10, 15, 20 are applicable for Profile A. Infinite value is applicable for Profile B. The same as *dl_slot_nb* defined in Section 3.2.2.3.

Parameters introduced in Section 5.5 Amazon Sidewalk Endpoint connection modes.

Name	Value	Description
ul_sync_interval	5 min	The maximum value of the time since last uplink for sending the synchronization packet with Epoch bit set.
ul_sync_guard_interval	1 min	Time interval within which uplink packets shall be used for downlink synchronization. Data packet in uplink direction within *ul_sync_interval*.
nb_dl_slot	5, 10, 15, 20, infinite	See description of *nb_dl_slot*.

B.3 Parameters for use in security

Parameters introduced in Section 4.2.

Name	Value	Description
gcs_ret_num	3	The number of GET_TIME commands that are sent with the constant interval in case no response is received from the Sidewalk Cloud.
gcs_resp_timeout	30 s, 3, 5, 8, 10, 15 min	The interval between subsequent GET_TIME commands in case there is no reponse is received from the Sidewalk Cloud. The first interval value 30 s is used max_retry_count times. Time intervals 3, 5, 8, 10 are used once. 15 min is used till the moment the RESP_TIME is received.
gcs_sync_int	2 h	The time interval between two GET_TIME commands when the clock is synchronized on the Endpoint side. It is used for clock synchronization maintenance.
gcs_clock_drift	60 s	The maximum allowed clock drift on the Endpoint side.

Parameters introduced in Section 4.7.2.

Name	Value	Description
T_IV_ref	60 s	The validity of the reference time in Initialization Vector (IV).

Appendix C

Change History

Version	Summary of Changes
Protocol Stack 1.0, Document Revision A	First release of specification.
Protocol Stack 1.0, Document Revision A.1	Add Sidewalk File transfer.