Amazon Sidewalk Multi-link Application Note

Protocol Stack 1.0, Document Revision A

December 21, 2023

Contents

Chapter 1

Scope

The scope of this document is to explain the implementation of the multi-link feature in the Amazon Sidewalk end point stack. The document also explains the behavior of the feature for different multi link policy settings exercised through the SID API. This document does not attempt to discuss the details of Amazon Sidewalk features and functionality. Detailed explanation of Amazon Sidewalk features and functionality is documented by feature specific application notes, specifications, developer guides and the Amazon Sidewalk white paper.

Chapter 2

Purpose

The purpose of the feature is to abstract the connection establishment and connection maintenance behavior of the links supported by Amazon Sidewalk stack to the developer of the Amazon Sidewalk stack. The developer is provided with varied degrees of flexibility to not only control the behavior of the connection of the links and also control the transfer of messages over the links.

Chapter 3

Overview of the Functionality

Amazon Sidewalk supports three link types illustrated below. The multi-link feature implements a connection policy framework which influences the behavior of the connection and uplink of messages using the supported links.

The multi-link feature exerts influence at the application layer of the stack. The behavior of the links at Network, Security, MAC and PHY layers are not impacted by this feature. The feature also plays a role even when only one link is enabled by the developer's application. It influences the message's uplink attributes and link's connection timeout parameters to send the message. The multi-link parameters are explained in detail section 3.2 Link Control Attributes.

The feature indirectly impacts the link on which the downlinks are received by end point. Sidewalk is an uplink based protocol, the downlink from the Amazon Sidewalk cloud services always follow the link on which the last uplink is received from the end point.

The following table provides a brief description of the connection establishment and connection maintenance for the three link types supported by Amazon Sidewalk.

3.1 Connection Establishment and Connection Maintenance on Links

3.1.1 Bluetooth LE

LINK	**Bluetooth LE**
SID_LINK_TYPE	SID_LINK_TYPE_1
Description	• The end point performs the role of a Bluetooth LE peripheral and the gateway the role of Bluetooth LE Central. • Amazon Sidewalk Bluetooth LE advertisements are sent by the end point periodically indicating its presence to the Amazon Sidewalk Bluetooth LE gateways and to mobile application leveraging the Amazon Sidewalk Mobile SDK. The advertisements are sent to allow the Amazon Sidewalk cloud to discover the presence of the Amazon Sidewalk end point which is in the range of any Amazon Sidewalk Bluetooth LE gateway to send a downlink or indicate the reachability of the end point to the AWS IoT Wireless managed service. • When the end point wants to send an uplink, it indicates its intent to connect to the Amazon Sidewalk network by changing the payload in the Bluetooth LE advertisements. The Amazon Sidewalk Bluetooth LE gateway which is in the range of the end point is instructed by the Amazon Sidewalk cloud to start the Bluetooth LE connection process and establish connection. The Amazon Sidewalk Bluetooth LE end point then sends the uplink. • The Amazon Sidewalk Bluetooth LE gateways do not keep the connection persistent for a longer duration. Unless there is an activity (uplink or downlink) in the connected state, the Amazon Sidewalk Bluetooth LE gateways terminate the connection after a certain time (currently configured to 5 minutes). The end point Sidewalk stack also disconnects the Bluetooth LE connection with the Sidewalk Bluetooth LE gateways when the developer's application does not send any messages to AWS IoT Wireless managed service. The timeout for Sidewalk stack for end point to disconnect the Sidewalk Bluetooth LE gateway is configured to 30 seconds. • Acknowledgements at link level are handled at the link layer by Bluetooth LE protocol stack. The Bluetooth LE protocol stack is responsible to establish connection, maintenance of the connection and reporting of connected and disconnected events to the Amazon Sidewalk stack. The Amazon Sidewalk Bluetooth LE stack maintains connection with Amazon Sidewalk Network while the Bluetooth LE protocol maintains the Bluetooth LE connection between the end point and Amazon Sidewalk cloud.
Connection	Connection on demand. Loss and establishment of connection is notified by the underlying Bluetooth LE Stack.
References	Amazon Sidewalk Specification SID API User Guide

Table 3.1: Bluetooth LE

3.1.2 SubG-FSK

LINK	**SubG-FSK**
SID_LINK_TYPE	SID_LINK_TYPE_2
Description	• The Amazon Sidewalk SubG-FSK gateways send beacons periodically (configured every 10 seconds). • The Amazon Sidewalk SubG-FSK endpoints synchronize with the beacons to establish and maintain connection. • The Amazon Sidewalk end point scans passively for Amazon Sidewalk beacons to sync(connect) to a Amazon Sidewalk gateway. • End points can detect connection loss with a particular SubG-FSK gateway if it fails to receive a pre-configured (currently 3) number of beacons from the currently connected gateway consecutively. • Acknowledgements are present at the link layer when messages are exchanged between end points and gateways thus providing link layer reliability.
Connection	Connected. Loss of connection is detected by loss of synchronization of beacons from the connected Sidewalk SubG-FSK gateway. Failure to receive a join response for the join request sent by the end point even when the end point is in synchronization with Sidewalk SubG-FSK gateway's beacons results in disconnection state.
References	Amazon Sidewalk Specification Device Profiles Application Note

Table 3.2: SubG-FSK

3.1.3 SubG-CSS

LINK	**SubG-CSS**
SID_LINK_TYPE	SID_LINK_TYPE_3
Description	• There are periodic messages that in Profile B that are required to be sent by the end point stack periodically while in Profile A there are no periodic messages that are required to be sent by end point stack. • Link layer acknowledgements are not present on this link type.
Connection	Connectionless protocol. Status of availability of link is determined only after sending a message.
References	Amazon Sidewalk Specification Device Profiles Application Note

Table 3.3: SubG-CSS

3.2 Link Control Attributes

The following are the attributes that are controlled by the multi-link feature to pick an optimum link for sending messages.

3.2.1 Message Uplink Attributes

The message uplink attributes are as follows:

- Request Acknowledgment from AWS IoT Wireless managed service
- Number of Retries
- Time to Live

These attributes and their usage is described in detail in SID API User Guide's **Chapter 2.8.2.1 Sidewalk Message Transmit Attributes**.

3.2.2 Link Connection Control Attributes

3.2.2.1 Link Priority

When more than one link is enabled and no link is in connected state, the order of priority that needs to be followed to attempt connection is determined by the link priority attribute. The value has a range from 0 to 2. Value 3 and greater will cause undefined behavior. Lower the value higher the priority.

3.2.2.2 Maximum Time to Attempt Connection for a Link

The maximum time in seconds for which the Amazon Sidewalk stack attempts to establish connection on a link.

3.3 Link Connection Policy

There are two link connection policies that are available to the developer. The link policies are mutually exclusive. Only one policy can be enabled at a time. These policies can also be compiled out at build time.

The two available policies are Auto Connection Policy and Multi-link Connection Policy.

The option to the select link connection policy that should be used by the Amazon Sidewalk stack can be configured using the **sid_option** API as shown below.

The default link connection policy that stack is configured with is SID_LINK_CONNECTION_POLICY_NONE. In SID_LINK_CONNECTION_POLICY_NONE policy setting the stack does not apply any connection policy settings.

The stack is required to be initialized using **sid_init** before setting link connection policy using **sid_option**. The link connection policy can be configured successfully before the Sidewalk stack on the end point is registered and time is acquired.

The link connection policy configuration does not persist within the stack across reboots. The developer is expected to configure the preferred link connection policy on every reboot of the device.

SID_ERROR_INVALID_ARGS is returned if the stack is not initialized or invalid arguments are passed to **sid_option** API.

```
/**
 *  Describes the connection policy setting
 */
enum sid_link_connection_policy {
    /** Default setting. Sidewalk stack does not apply any
```

```
     *  connection policy. */
    SID_LINK_CONNECTION_POLICY_NONE = 0,
    /** Auto connection policy, Sidewalk stack applies auto connection policy
     *  based on the parameters
     *  configured through auto connection policy parameters. */
    SID_LINK_CONNECTION_POLICY_AUTO_CONNECT,
    /** Multi-link connection policy, Sidewalk stack applies multi-link
     *  connection policy based on the policy
     *  configured through multi-link connection policy. */
    SID_LINK_CONNECTION_POLICY_MULTI_LINK_MANAGER,
    /** Delimiter to enum sid_link_connection_policy */
    SID_LINK_CONNECTION_POLICY_LAST
};

// IOCTL to set link connection policy

// Set link connection policy to SID_LINK_CONNECTION_POLICY_AUTO_CONNECT
enum sid_link_connection_policy set_policy =
                                SID_LINK_CONNECTION_POLICY_AUTO_CONNECT;
sid_error_t ret = sid_option (handle, SID_OPTION_SET_LINK_CONNECTION_POLICY,
                              &set_policy, sizeof(set_policy));

// IOCTL to get link connection policy

// Get configured link connection policy
enum sid_link_connection_policy get_policy;
sid_error_t ret = sid_option(handle, SID_OPTION_GET_LINK_CONNECTION_POLICY,
                             &get_policy, sizeof(get_policy));
```

3.3.1 Default Link Connection Policy

When the link connection policy is configured to SID_LINK_CONNECTION_POLICY_NONE, the stack does not apply any connection policy algorithm to the messages enqueued with **sid_put_msg** API.

The developer is expected to establish the connection before enqueuing the message. If the link is not connected and the message is enqueued, the Amazon Sidewalk stack would fail the message with SID_ERROR_INVALID_STATE error code.

3.3.2 Auto Connection Policy

The auto connection policy can be enabled at build time by setting the build flag

SID_SDK_CONFIG_ENABLE_AUTO_CONNECT.

The auto connection policy provides flexibility to the developer to configure the link connection control attributes mentioned above.

The Amazon Sidewalk stack is required to be initialized using **sid_init** before setting link auto connection policy parameters using **sid_option**. The link auto connection policy parameters can be configured successfully before the Sidewalk stack on the end point is registered and time is acquired.

The Amazon Sidewalk stack triggers connection on a link only when a message is required to be transmitted by the developer using the **sid_put_msg** API.

The link auto connection policy parameters do not persist within the stack across reboots. The developer is expected to configure the preferred link auto link connection policy parameters on every reboot of the device.

SID_ERROR_INVALID_ARGS is returned if the stack is not initialized or invalid arguments are passed to **sid_option** API.

The following shows the parameters that are available to the developer to configure the auto link connection policy. The developer can select the link types, priority among the links and connection attempt timeout seconds.

```
/**
 *  Describes link auto connect parameters
 */
struct sid_link_auto_connect_params {
    /** Sid link type */
    enum sid_link_type link_type;
    /** Auto connect policy if set enable is equal to 0 */
    bool enable;
    /** Priority of the link type when more than one link is specified
     *  using #sid_put_msg
     *  0 is highest priority, 2 is lowest priority */
    uint8_t priority;
    /** Maximum period upto which the stack attempts to form a connection */
    uint16_t connection_attempt_timeout_seconds;
};

// IOCTL to set auto connect parameters

// Set auto connection params for SID_LINK_TYPE_1 with priority 1 and
// connection_attempt_timeout_seconds to 30
struct sid_link_auto_connect_params params_link_1 = {
    .link_type = SID_LINK_TYPE_1,
    .enable = true,
    .priority = 0,
    .connection_attempt_timeout_seconds = 30
};

sid_error_t ret = sid_option(handle,
                             SID_OPTION_SET_LINK_POLICY_AUTO_CONNECT_PARAMS,
                             &params_link_1, sizeof(params_link_1));

// Set auto connection params for SID_LINK_TYPE_2 with priority 2 and
// connection_attempt_timeout_seconds to 120
struct sid_link_auto_connect_params params_link_2 = {
    .link_type = SID_LINK_TYPE_2,
    .enable = true,
    .priority = 1,
    .connection_attempt_timeout_seconds = 120
};
ret = sid_option(handle, SID_OPTION_SET_LINK_POLICY_AUTO_CONNECT_PARAMS,
                 &params_link_2, sizeof(params_link_2));

// Set auto connection params for SID_LINK_TYPE_3 with priority 3 and
// connection_attempt_timeout_seconds to 90
struct sid_link_auto_connect_params params_link_3 = {
    .link_type = SID_LINK_TYPE_3,
    .enable = true,
    .priority = 2,
    .connection_attempt_timeout_seconds = 90
};
ret = sid_option(handle, SID_OPTION_SET_LINK_POLICY_AUTO_CONNECT_PARAMS,
                 &params_link_3, sizeof(params_link_3));

// Disable auto connection on SID_LINK_TYPE_2 link
struct sid_link_auto_connect_params params_link_2_disable = {
    .link_type = SID_LINK_TYPE_2,
    .enable = false,
};
ret = sid_option(handle, SID_OPTION_SET_LINK_POLICY_AUTO_CONNECT_PARAMS,
                 &params_link_2_disable, sizeof(params_link_2_disable));

// IOCTL to get auto connect parameters

// Get auto connection parameters configured for SID_LINK_TYPE_1
struct sid_link_auto_connect_params get_params_link_1 = {
    .link_type = SID_LINK_TYPE_1,
};
ret = sid_option(handle, SID_OPTION_GET_LINK_POLICY_AUTO_CONNECT_PARAMS,
                 &get_params_link_1, sizeof(get_params_link_1));
```

3.3.2.1 Connection Attempt Timeout

The following table gives the minimum value of **connection_attempt_timeout_seconds** that can be configured for each of the link type.

SID_ERROR_INVALID_ARGS is returned if a value for **connection_attempt_timeout_seconds** is configured to be lower than the value indicated in the table below.

link_type	**minimum connection attempt timeout in seconds**
SID_LINK_TYPE_1	10
SID_LINK_TYPE_2	30
SID_LINK_TYPE_3	20

Table 3.4: The minimum value of connection attempt timeout in seconds for each link type.

3.3.2.2 Link Auto Connection Policy Algorithm

The following conditions are required to be met for the Sidewalk stack to trigger connection on a link.

- The link is already started using the **sid_start** API.
- The end point is already registered to the Amazon Sidewalk network and time is acquired successfully.
- The link connection policy is configured to SID_LINK_CONNECTION_POLICY_AUTO_CONNECT.
- The auto connection policy parameters are enabled on the link.
- The link mask present in the **sid_put_msg** has this link enabled in the **sid_msg_desc**.
- The size of the message that is required to be transmitted is less than the MTU of the links that are started and have the auto connection policy parameters as enabled.

- If a Sidewalk stack has at least one link in connected state and the connected link can send the message, the link auto connection policy algorithm does not get triggered as the connected link can successfully send the message.
- When the connection cannot be established on any of the links the policy is enabled, the policy will progress through the links in the order of priority configured by the developer multiple times until all the messages that are enqueued by the user have expired. When there are no messages in the end point Amazon Sidewalk stack's send queue, the algorithm will stop attempting to establish connection with the Amazon Sidewalk network.

The following examples are used to explain the behavior of the link auto connection policy algorithm.

Example 1

Conditions	**Algorithm Operation**
• Sidewalk stack initialized and started with all links. • At least one link is in connected state, SID_LINK_TYPE_3 is connected. • Auto connect policy is enabled and auto connection policy parameters on all three links. • Link priority order is SID_LINK_TYPE_1 > SID_LINK_TYPE_2 > SID_LINK_TYPE_3. • Message sent using **sid_put_msg** with message tx attributes of acknowledgment required, retries and time to live set by the developer. • Message sent using **sid_put_msg** has link_type set to pick any of the three links. (link_type = (SID_LINK_TYPE_1 | SID_LINK_TYPE_2 | SID_LINK_TYPE_3)) • The payload of the message that is requested to be transmitted can be sent on all the three links.	The auto connection policy does not attempt connection on any links as the link which is in connected state will send the message on the connected link.

Table 3.5: Example 1

Example 2

Conditions	Algorithm Operation
• Sidewalk stack initialized and started with all links. • No link is in connected state. • Auto connect policy is enabled and auto connection policy parameters on all three links. • Link priority order is SID_LINK_TYPE_1 > SID_LINK_TYPE_2 > SID_LINK_TYPE_3. • Message sent using **sid_put_msg** with message tx attributes of acknowledgment required, retries and time to live set by the developer. • Message sent using **sid_put_msg** has link_type set to pick any of the three links. (link_type = (SID_LINK_TYPE_1 | SID_LINK_TYPE_2 | SID_LINK_TYPE_3)) • The payload of the message that is requested to be transmitted can be sent on all the three links.	The auto connection policy starts attempting link connection for a period of connection timeout specified by the auto connection policy for that link. The order of priority to pick links for connection in this case is SID_LINK_TYPE_1 followed by SID_LINK_TYPE_2 and SID_LINK_TYPE_3. During the course of connection attempts, if any of the link gets connected, the message is sent over the connected link. If any of the links do not get connected within the configured time to live setting in the message's transmit attributes the message will be failed with SID_ERROR_TIMEOUT error code.

Table 3.6: Example 2

Example 3

Conditions	Algorithm Operation
• Sidewalk stack initialized and started with all links. • No link is in connected state. • Auto connect policy is enabled and auto connection policy parameters on all three links. • Link priority order is SID_LINK_TYPE_3 > SID_LINK_TYPE_2 > SID_LINK_TYPE_1. • Message sent using **sid_put_msg** with message tx attributes of acknowledgment required, retries and time to live set by the developer. • Message sent using **sid_put_msg** has link_type set to pick any of the three links. (link_type = (SID_LINK_TYPE_3 | SID_LINK_TYPE_2 | SID_LINK_TYPE_1)) • The payload of the message that is requested to be transmitted can be sent on all the three links.	The auto connection policy starts attempting link connection for a period of connection timeout specified by the auto connection policy for that link. The order of priority to pick links for connection in this case is SID_LINK_TYPE_3 followed by SID_LINK_TYPE_2 and SID_LINK_TYPE_1. During the course of connection attempts, if any of the link gets connected, the message is sent over the connected link. If any of the links do not get connected within the configured time to live setting in the message's transmit attributes the message will be failed with SID_ERROR_TIMEOUT error code.

Table 3.7: Example 3

Example 4

Conditions	**Algorithm Operation**
• Sidewalk stack initialized and started with all links. • No link is in connected state. • Auto connect policy is enabled and auto connection policy parameters on all three links. • Link priority order is SID_LINK_TYPE_1 > SID_LINK_TYPE_2 > SID_LINK_TYPE_3. • Message sent using **sid_put_msg** with message tx attributes of acknowledgment required, retries and time to live set by the developer. • Message sent using **sid_put_msg** has link_type set to pick any of the three links. (link_type = (SID_LINK_TYPE_1 | SID_LINK_TYPE_2 | SID_LINK_TYPE_3)) • The payload of the message that is requested to be transmitted can only be sent on only SID_LINK_TYPE_1.	The auto connection policy attempts link connection only on SID_LINK_TYPE_1 as SID_LINK_TYPE_2 and SID_LINK_TYPE_3 cannot send the message as the payload of the message exceeds the MTU of the two links. If the connection can be established within the time to live of message's transmit attributes, the message will be sent over SID_LINK_TYPE_1. Failure to send would result in message failing with SID_ERROR_TIMEOUT error code.

Table 3.8: Example 4

Example 5

Conditions	**Algorithm Operation**
• Sidewalk stack initialized and started with all links. • No link is in connected state. • Auto connect policy is enabled and auto connection policy parameters on all three links. • Link priority order is SID_LINK_TYPE_1 > SID_LINK_TYPE_2 > SID_LINK_TYPE_3. • Message sent using **sid_put_msg** with message tx attributes of acknowledgment required, retries and time to live set by the developer. • Message sent using **sid_put_msg** has link_type set to pick only one link. (link_type = SID_LINK_TYPE_2) • The payload of the message that is requested to be transmitted can be sent on all the three links.	The auto connection policy attempts link connection only on SID_LINK_TYPE_2 as SID_LINK_TYPE_1 and SID_LINK_TYPE_3 are not in the link_type setting in **sid_put_msg** API. If the connection can be established within the time to live of message's transmit attributes, the message will be sent over SID_LINK_TYPE_1. Failure to send would result in message failing with SID_ERROR_TIMEOUT error code.

Table 3.9: Example 5

Example 6

Conditions	Algorithm Operation
• Sidewalk stack initialized and started with only one link SID_LINK_TYPE_2. • No link is in connected state. • Auto connect policy is enabled and auto connection policy parameters on all three links. • Link priority order is SID_LINK_TYPE_1 > SID_LINK_TYPE_2 > SID_LINK_TYPE_3. • Message sent using **sid_put_msg** with message tx attributes of acknowledgment required, retries and time to live set by the developer. • Message sent using **sid_put_msg** has link_type set to pick any of the three links. (link_type = (SID_LINK_TYPE_1 | SID_LINK_TYPE_2 | SID_LINK_TYPE_3)) • The payload of the message that is requested to be transmitted can be sent on all the three links.	The Sidewalk stack is started with only one link SID_LINK_TYPE_2. Even though the link auto connection policy parameters are configured for all links, the auto connection policy algorithm will function only on one link, in this case SID_LINK_TYPE_2 as the other two links are not started. If the connection can be established within the time to live of message's transmit attributes, the message will be sent over SID_LINK_TYPE_2. Failure to send would result in message failing with SID_ERROR_TIMEOUT error code.

Table 3.10: Example 6

Example 7

Conditions	Algorithm Operation
• Sidewalk stack initialized and started with two links (SID_LINK_TYPE_2 | SID_LINK_TYPE_3). • No link is in connected state. • Auto connect policy is enabled and auto connection policy parameters on all three links. • Link priority order is SID_LINK_TYPE_1 > SID_LINK_TYPE_2 > SID_LINK_TYPE_3. • Message sent using **sid_put_msg** with message tx attributes of acknowledgment required, retries and time to live set by the developer. • Message sent using **sid_put_msg** has link_type set to pick any of the three links. (link_type = (SID_LINK_TYPE_1 | SID_LINK_TYPE_2 | SID_LINK_TYPE_3)) • The payload of the message that is requested to be transmitted can be sent on all the three links.	The Sidewalk stack is started with two links SID_LINK_TYPE_2 and SID_LINK_TYPE_3. Even though the link auto connection policy parameters are configured for all links, the auto connection policy algorithm will function only on two links, in this case SID_LINK_TYPE_2 and SID_LINK_TYPE_3 as SID_LINK_TYPE_1 is not started. If the connection can be established within the time to live of message's transmit attributes, the message will be sent over SID_LINK_TYPE_2 or SID_LINK_TYPE_3. Failure to send would result in message failing with SID_ERROR_TIMEOUT error code.

Table 3.11: Example 7

Example 8

Conditions	Algorithm Operation
• Sidewalk stack initialized and started with two links (SID_LINK_TYPE_2 | SID_LINK_TYPE_3). • No link is in connected state. • Auto connect policy is enabled and auto connection policy parameters on all three links. • Link priority order is SID_LINK_TYPE_1 > SID_LINK_TYPE_2 > SID_LINK_TYPE_3. • Message sent using **sid_put_msg** with message tx attributes of acknowledgment required, retries and time to live set by the developer. • Message sent using **sid_put_msg** has link_type set to pick only one link that is not started. (link_type = SID_LINK_TYPE_1) • The payload of the message that is requested to be transmitted can be sent on all the three links.	The Sidewalk stack will not attempt connection as the developer has requested the message to be sent on a link SID_LINK_TYPE_1 which is not started. The message will be immediately failed with error code SID_ERROR_NO_ROUTE_AVAILABLE.

Table 3.12: Example 8

Example 9

Conditions	Algorithm Operation
• Sidewalk stack initialized and started with two links (SID_LINK_TYPE_2 | SID_LINK_TYPE_3). • No link is in connected state. • Auto connect policy is enabled and auto connection policy parameters on all three links. • Link priority order is SID_LINK_TYPE_1 > SID_LINK_TYPE_2 > SID_LINK_TYPE_3. • Message sent using **sid_put_msg** with message tx attributes of acknowledgment required, retries and time to live set by the developer. • Message sent using **sid_put_msg** has link_type set to pick any of the three links. (link_type = (SID_LINK_TYPE_1 | SID_LINK_TYPE_2 | SID_LINK_TYPE_3)) • The payload of the message that is requested to be transmitted can be sent only one link which is SID_LINK_TYPE_1 which is not started.	The Sidewalk stack will not attempt connection as the developer has requested the message whose payload cannot be sent over the links that are started in this case SID_LINK_TYPE_2 and SID_LINK_TYPE_3. The message will be immediately failed with error code SID_ERROR_NO_ROUTE_AVAILABLE.

Table 3.13: Example 9

Example 10

Conditions	Algorithm Operation
• Sidewalk stack initialized and started with all links. • No link is in connected state. • Auto connect policy is disabled on all links. • Message sent using **sid_put_msg** with message tx attributes of acknowledgment required, retries and time to live set by the developer. • Message sent using **sid_put_msg** has link_type set to all links. (link_type = (SID_LINK_TYPE_1 | SID_LINK_TYPE_2 | SID_LINK_TYPE_3)) • The payload of the message that is requested to be transmitted can be sent on all the three links.	The Sidewalk stack will not attempt connection as the link auto connection policy parameters are disabled on all links. The message will be immediately failed with error code SID_ERROR_NO_ROUTE_AVAILABLE.

Table 3.14: Example 10

3.3.3 Multi-link Connection Policy

The multi-link connection policy can be enabled at build time by setting the build flag SID_SDK_CONFIG_ENABLE_MULTI_LINK.

The multi-link connection policy determines the message uplink attributes and link connection attributes and does not expect the developer to configure these parameters while sending the message.

The Amazon Sidewalk stack is required to be initialized using **sid_init** before setting link multi-link connection policy using **sid_option**. The link multi-link connection policy can be configured successfully before the Sidewalk stack on the end point is registered and time is acquired.

The Amazon Sidewalk stack triggers connection on a link only when a message is required to be transmitted by the developer using the **sid_put_msg** API.

The multi-link connection policy do not persist within the stack across reboots. The developer is expected to set the preferred multi-link connection policy on every reboot of the device.

SID_ERROR_INVALID_ARGS is returned if the stack is not initialized or invalid arguments are passed to **sid_option** API.

The following shows the policies that are available to the developer to configure the multi-link connection policy.

```
/** multi-link policy modes */
enum sid_link_multi_link_policy {
    /** Default policy, Default Setting, All links are enabled */
    SID_LINK_MULTI_LINK_POLICY_DEFAULT = 0,
    /** Policy optimized for better power consumption */
    SID_LINK_MULTI_LINK_POLICY_POWER_SAVE = 1,
    /** Policy optimized for performance, better throughput */
    SID_LINK_MULTI_LINK_POLICY_PERFORMANCE = 2,
    /** Policy optimized for latency, faster uplinks */
    SID_LINK_MULTI_LINK_POLICY_LATENCY = 3,
    /** Policy optimized for reliability, messages have a longer Time to live
     *  and more retries */
    SID_LINK_MULTI_LINK_POLICY_RELIABILITY = 4,
    /** Delimiter to enum sid_link_power_policy_type */
    SID_LINK_MULTI_LINK_POLICY_LAST
};

// IOCTL to set multi-link policy

// Set multi-link policy to SID_LINK_MULTI_LINK_POLICY_PERFORMANCE
enum sid_link_multi_link_policy policy =
     SID_LINK_MULTI_LINK_POLICY_PERFORMANCE;
sid_error_t ret = sid_option(handle,
                             SID_OPTION_SET_LINK_POLICY_MULTI_LINK_POLICY,
                             &policy, sizeof(policy));

// IOCTL to get multi-link policy

// Get current configured multi-link policy
enum sid_link_multi_link_policy get_policy;
sid_error_t ret = sid_option(handle,
                             SID_OPTION_GET_LINK_POLICY_MULTI_LINK_POLICY,
                             &get_policy, sizeof(get_policy));
```

The following conditions are required to be met for the Sidewalk stack to trigger connection on a link.

- The Amazon Sidewalk stack is started successfully using **sid_start** API.
- The Sidewalk stack is already registered and time is acquired successfully.
- The link connection policy is configured to SID_LINK_CONNECTION_POLICY_MULTI_LINK_MANAGER. SID_LINK_MULTI_LINK_POLICY_ACTIVE is the default that is used when link connection policy is set to SID_LINK_CONNECTION_POLICY_MULTI_LINK_MANAGER.
- The size of the message that is required to be transmitted is less than the MTU of the links that are started and have the auto connection policy parameters as enabled.

The link connection attributes and message uplink attributes are determined by the multi-link policy and the developer configuration is not used to send the message.

The following table provides the configuration of these attributes for each of the multi-link policy.

The configuration shown in the below policies cannot be configured by the developer. The configuration is statically encoded in the mulit-link policy algorithm.

3.3.3.1 SID_LINK_POLICY_DEFAULT

Multi-link Policy	**SID_LINK_POLICY_DEFAULT**
Description	This multi-link policy has the default behavior of the stack except for configuring the message reliability parameters and connection attributes. The stack does not attempt to keep the connection with the Sidewalk cloud services when there are no messages enqueued to be sent by the stack.

Message Reliability Parameters	**Parameter**	**Bluetooth LE**	**SubG-FSK**	**SubG-CSS**
	Time to live in seconds	60	120	150
	Retries	3	3	3
	Transport acknowledgement requested	true	true	true

Link Priority for Uplink	**Link**	**Priority(lower value = high priority)**
	Bluetooth LE	0
	SubG-FSK	1
	SubG-CSS	2

LINK_TYPE_1 (Bluetooth LE)	**Connection Attributes**	**Setting**
	Link status	Enabled
	max connection attempt period in seconds	30
	Attempt to connect to gateway(uplink active)	false

LINK_TYPE_2 (SubG-FSK)	**Connection Attributes**	**Setting**
	Link status	Active
	Keep connection with gateway	true
	max connection attempt period in seconds	60

LINK_TYPE_3 (SubG-CSS)	**Connection Attributes**	**Setting**
	Link status	Active
	max connection attempt period in seconds	60

Table 3.15: Default

3.3.3.2 SID_LINK_POLICY_OPTIMIZE_FOR_POWER

Multi-link Policy	**SID_LINK_POLICY_OPTIMIZE_FOR_POWER**
Description	This multi-link policy stops the links that are enabled by the stack when the stack is not requested to send any messages. The policy behavior is to optimize the power consumption on the device.

Message Reliability Parameters

Parameter	**Bluetooth LE**	**SubG-FSK**	**SubG-CSS**
Time to live in seconds	60	120	150
Retries	1	1	1
Transport acknowledgement requested	true	true	true

Link Priority for Uplink

Link	**Priority(lower value = high priority)**
Bluetooth LE	0
SubG-FSK	1
SubG-CSS	2

LINK_TYPE_1 (Bluetooth LE)

Connection Attributes	**Setting**
Link status	Enabled
max connection attempt period in seconds	30
Attempt to connect to gateway(uplink active)	false

LINK_TYPE_2 (SubG-FSK)

Connection Attributes	**Setting**
Link status	Active
Keep connection with gateway	true
max connection attempt period in seconds	60

LINK_TYPE_3 (SubG-CSS)

Connection Attributes	**Setting**
Link status	Active
max connection attempt period in seconds	60

Table 3.16: Optimize for Power

3.3.3.3 SID_LINK_POLICY_OPTIMIZE_FOR_PERFORMANCE

Multi-link Policy	**SID_LINK_POLICY_OPTIMIZE_FOR_PERFORMANCE**
Description	This multi-link policy has the behavior to maximize throughput of the device. The stack attempts to keep at least one of the enabled links connected with the Amazon Sidewalk cloud service. The policy maintains the connected state even when no messages are enqueued.

Message Reliability Parameters:

Parameter	**Bluetooth LE**	**SubG-FSK**	**SubG-CSS**
Time to live in seconds	60	120	120
Retries	1	1	1
Transport acknowledgement requested	true	true	true

Link Priority for Uplink:

Link	**Priority(lower value = high priority)**
Bluetooth LE	0
SubG-FSK	1
SubG-CSS	2

LINK_TYPE_1 (Bluetooth LE):

Connection Attributes	**Setting**
Link status	Enabled
max connection attempt period in seconds	30
Attempt to connect to gateway(uplink active)	true

LINK_TYPE_2 (SubG-FSK):

Connection Attributes	**Setting**
Link status	Active
Keep connection with gateway	true
max connection attempt period in seconds	60

LINK_TYPE_3 (SubG-CSS):

Connection Attributes	**Setting**
Link status	Active
max connection attempt period in seconds	30

Table 3.17: Optimize for Performance

3.3.3.4 SID_LINK_POLICY_OPTIMIZE_FOR_LATENCY

Multi-link Policy	**SID_LINK_POLICY_OPTIMIZE_FOR_LATENCY**
Description	This multi-link policy has the behavior to optimize the latency of the uplink messages. The period for which the stack attempts to form a successful connection with the Amazon Sidewalk cloud services is low when compared to other policies.

Message Reliability Parameters	**Parameter**	**Bluetooth LE**	**SubG-FSK**	**SubG-CSS**
	Time to live in seconds	30	60	120
	Retries	3	3	3
	Transport acknowledgement requested	true	true	true

Link Priority for Uplink	**Link**	**Priority(lower value = high priority)**
	Bluetooth LE	0
	SubG-FSK	1
	SubG-CSS	2

LINK_TYPE_1 (Bluetooth LE)	**Connection Attributes**	**Setting**
	Link status	Enabled
	max connection attempt period in seconds	15
	Attempt to connect to gateway(uplink active)	true

LINK_TYPE_2 (SubG-FSK)	**Connection Attributes**	**Setting**
	Link status	Active
	Keep connection with gateway	true
	max connection attempt period in seconds	30

LINK_TYPE_3 (SubG-CSS)	**Connection Attributes**	**Setting**
	Link status	Active
	max connection attempt period in seconds	60

Table 3.18: Optimize for Latency

3.3.3.5 SID_LINK_POLICY_OPTIMIZE_FOR_RELIABILITY

Multi-link Policy	**SID_LINK_POLICY_OPTIMIZE_FOR_RELIABILITY**
Description	This multi-link policy has the policy to optimize for reliable sending of the messages. This policy has configuration that has higher message retries and time to live configured for each message.

Message Reliability Parameters	**Parameter**	**Bluetooth LE**	**SubG-FSK**	**SubG-CSS**
	Time to live in seconds	30	120	240
	Retries	5	5	5
	Transport acknowledgement requested	true	true	true

Link Priority for Uplink	**Link**	**Priority(lower value = high priority)**
	Bluetooth LE	0
	SubG-FSK	1
	SubG-CSS	2

LINK_TYPE_1 (Bluetooth LE)	**Connection Attributes**	**Setting**
	Link status	Enabled
	max connection attempt period in seconds	60
	Attempt to connect to gateway(uplink active)	false

LINK_TYPE_2 (SubG-FSK)	**Connection Attributes**	**Setting**
	Link status	Active
	Keep connection with gateway	true
	max connection attempt period in seconds	60

LINK_TYPE_3 (SubG-CSS)	**Connection Attributes**	**Setting**
	Link status	Active
	max connection attempt period in seconds	120

Table 3.19: Optimize for Reliability

3.3.3.6 Multi-link Connection Policy Algorithm

The following conditions are required to be met for the Sidewalk stack to trigger connection on a link.

- The link is already started using the **sid_start** API.
- The Sidewalk stack is already registered and time is acquired successfully
- The link connection policy is configured to SID_LINK_CONNECTION_POLICY_MULTI_LINK_MANAGER.
- The size of the message that is required to be transmitted is less than the MTU of the links that are started.
- When the size of the messages that is required to be transmitted exceeds the MTU of the links that are started, the message is failed with SID_ERROR_NO_ROUTE_AVAILABLE by the **sid_put_msg** API.
- The message transmission attributes (time to live, retries and ack requested) of the connected link are applied to messages enqueued with **sid_put_msg** API.
- If more than one link is connected, the message transmission attributes of the link that has the higher throughput is applied to the message that is enqueued with **sid_put_msg** API. For example, If SID_LINK_TYPE_1 and SID_LINK_TYPE_2 are in connected state, the message transmission attributes of SID_LINK_TYPE_1 are applied for the enqueued message.

- If no link is connected, the messages transmission attributes of the link that has the lowest throughput is applied to the message that is enqueued with **sid_put_msg** API. For example, if SID_LINK_TYPE_1, SID_LINK_TYPE_2 and SID_LINK_TYPE_3 are enabled, and all the links are in disconnected state, the message that is enqueued with **sid_put_msg** API assumes the message transmission attributes of SID_LINK_TYPE_3.
- If a Sidewalk stack has at least one link in connected state and the connected link can send the message, the link auto connection policy algorithm does not get triggered as the connected link can successfully send the message.
- The message uplink attributes as well the link_type in **sid_msg_desc** setting used by the developer is ignored and no validation checks are performed on these uplink message attributes. These attributes are applied based on the multi-link policy configuration set in the table below.
- Multi-link policy functionality is enabled even when the Amazon Sidewalk stack is compiled with support for only one link or only when one link is initialized and started. When only one link is present, the multi-link policy manager will attempt connection only on the link which is enabled.
- For multi-link connection policies which do not have support of background connection maintenance, when the connection cannot be established on any of the links the policy is enabled, the policy will progress through the links in the order of priority configured in the policy table multiple times until all the messages that are enqueued by the user have expired. When there are no messages in the end point Amazon Sidewalk stack's send queue, the algorithm will stop attempting to establish connection with the Amazon Sidewalk network.

The following examples are used to explain the behavior of the link auto connection policy algorithm.

Example 1

Conditions	**Algorithm Operation**
• Sidewalk stack initialized and started with all links. • At least one link is in connected state, SID_LINK_TYPE_3 is connected. • Link connection policy is set to multi-link policy is enabled and SID_LINK_MULTI_LINK_POLICY_ACTIVE is selected. • The payload of the message that is requested to be transmitted can be sent on all the three links.	The message transmission attributes viz, time to live, ack requested and retries are applied from SID_LINK_TYPE_3 message attributes of the SID_LINK_MULTI_LINK_POLICY_ ACTIVE. The multi-link connection policy does not attempt connection on any links as the link which is in connected state will send the message on the connected link.

Table 3.20: Example 1

Example 2

Conditions	Algorithm Operation
• Sidewalk stack initialized and started with all links. • No link is in connected state. • Link connection policy is set to multi-link policy is enabled and SID_LINK_MULTI_LINK_POLICY_ RELIABILITY is selected. • The payload of the message that is requested to be transmitted can be sent on all the three links.	The message transmission attributes viz, time to live, ack requested and retries are applied from SID_LINK_TYPE_3 message attributes of the SID_LINK_MULTI_LINK_POLICY_ RELIABILITY as no link are in connected state and the link with lowest throughput capacity is SID_LINK_TYPE_3. The multi-link connection policy starts attempting link connection for a period of connection timeout specified by the multi-policy for that link. During the course of connection attempts, if any of the link gets connected, the message is sent over the connected link. If any of the links do not get connected within the configured time to live setting in the multi-link policy's message attributes setting or the message will be failed with SID_ERROR_TIMEOUT error code.

Table 3.21: Example 2

Example 3

Conditions	Algorithm Operation
• Sidewalk stack initialized and started with all links. • SID_LINK_TYPE_1 and SID_LINK_TYPE_2 are in connected state. • Link connection policy is set to multi-link policy is enabled and SID_LINK_MULTI_LINK_POLICY_ RELIABILITY is selected. • The payload of the message that is requested to be transmitted can be sent on all the three links.	The message transmission attributes viz, time to live, ack requested and retries are applied from SID_LINK_TYPE_1 message attributes of the SID_LINK_MULTI_LINK_POLICY_ RELIABILITY as SID_LINK_TYPE_1 and SID_LINK_TYPE_2 are in connected state and the link with highest throughput capacity is SID_LINK_TYPE_1. The multi-link connection policy does not attempt connection on any links as both SID_LINK_TYPE_1 and SID_LINK_TYPE_2 are in connected state and the message will be sent on SID_LINK_TYPE_1.

Table 3.22: Example 3

Example 4

Conditions	Algorithm Operation
• Sidewalk stack initialized and started with two links (SID_LINK_TYPE_1 | SID_LINK_TYPE_2). • No link is in connected state. • Link connection policy is set to multi-link policy is enabled and SID_LINK_MULTI_LINK_POLICY_POWER is selected. • The payload of the message that is requested to be transmitted can be sent on all the three links.	The message transmission attributes viz, time to live, ack requested and retries are applied from SID_LINK_TYPE_2 message attributes of the SID_LINK_MULTI_LINK_POLICY_POWER as no link are in connected state and the link with lowest throughput capacity is SID_LINK_TYPE_2. The multi-link connection policy starts attempting link connection for a period of connection timeout specified by the multi-policy for that link. During the course of connection attempts, if any of the link gets connected, the message is sent over the connected link. If any of the links do not get connected within the configured time to live setting in the multi-link policy's message attributes setting or the message will be failed with SID_ERROR_TIMEOUT error code.

Table 3.23: Example 4

Example 5

Conditions	Algorithm Operation
• Sidewalk stack initialized and started with only one link SID_LINK_TYPE_3. • No link is in connected state. • Link connection policy is set to multi-link policy is enabled and SID_LINK_MULTI_LINK_POLICY_PERFORMANCE is selected. • The payload of the message that is requested to be transmitted cannot be transmitted on SID_LINK_TYPE_3.	The Sidewalk stack will not attempt connection as the developer has requested the message whose payload cannot be sent over the links that are started in this case SID_LINK_TYPE_3. The message will be immediately failed with error code SID_ERROR_NO_ROUTE_AVAILABLE.

Table 3.24: Example 5

Example 6

Conditions	Algorithm Operation
• Sidewalk stack initialized and started with only one link SID_LINK_TYPE_2. • No link is in connected state. • Link connection policy is set to multi-link policy is enabled and SID_LINK_MULTI_LINK_POLICY_LATENCY is selected. • The payload of the message that is requested to be transmitted can be transmitted on SID_LINK_TYPE_2.	The Sidewalk stack will attempt connection on SID_LINK_TYPE_2 as this is the only link that is started. If the link gets connected in the time period present in the messages's uplink attributes, the message is sent successfully, or the message is failed with SID_ERROR_TIMEOUT error code.

Table 3.25: Example 6

Background Connection Maintenance

When the stack is configured with multi link policies SID_LINK_POLICY_OPTIMIZE_FOR_LATENCY and SID_LINK_POLICY_OPTIMIZE_FOR_PERFORMANCE, the multi-link algorithm attempts on SID_LINK_TYPE_1 link keeps connection (uplink active) with the Sidewalk cloud services. This connection maintenance is achieved by updating the advertisement payload to connect. For other policies, the algorithm maintains the default behavior of the link connection policy.

3.4 Updates to sid_put_msg API

When the link connection policy is not configured either to SID_LINK_CONNECTION_POLICY_AUTO_CONNECT or SID_LINK_CONNECTION_POLICY_MULTI_LINK_MANAGER, the **sid_put_msg** will accept messages only when the Sidewalk stack is registered, time is acquired and at least one of the configured links that the Amazon Sidewalk stack is started with, is in connected state.

When either of the connection policy is enabled, the **sid_put_msg** accepts messages even when Sidewalk stack is registered and time acquired. The links need not be in connected state for the **sid_put_msg** API to accept messages.

The Amazon Sidewalk stack will queue this messages in its internal queue waiting on the connection policy algorithm to successfully establish a connection.

It should be noted that the rate at which the messages are sent using **sid_put_msg** API should be dependent on the throughput capacity of the links on which the connection can be established. For example, the SID_LINK_TYPE_3 has a lower throughput for sending messages. Enqueuing messages at a rate faster than the link can successfully send messages will cause undefined behavior.

Glossary

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