SKYLINK® MESSAGING HUB PUBLIC APIs REFERENCE GUIDE

Ver. 1.1

NAL Research, operating under the brands NAL Research and Blue Sky Network

11100 Endeavor Ct, STE 300, Manassas, VA, 20109, United States

Support@NALResearch.com | +1 858-551-3894 | www.NALResearch.com

LEGAL DISCLAIMER AND CONDITION OF USE

This guide is published and copyrighted by NAL Research (NAL). All information and specifications in this document are subject to change without notice. Nothing in this document is intended to create additional or separate warranties or guarantees.

NAL Research, operating under the brands NAL Research and Blue Sky Network
11100 Endeavor Ct, STE 300, Manassas, VA, 20109, United States
Support@NALResearch.com | +1 858-551-3894 | www.NALResearch.com

TABLE OF CONTENTS

PAGE INTENTIONALLY LEFT BLANK

INTRODUCTION

This guide explains how to use the SkyLink® Messaging Hub, including how to create and manage API Keys and channels in the SkyLink Portal.

IMPORTANT: Before using these optional Messaging Hub features, the SkyLink terminal must be remotely enabled by our activation team. Please contact your salesperson/account manager for details on the monthly fee associated with these services.

About the Messaging Hub

The SkyLink Messaging Hub is a message relay and delivery service for SkyLink devices. Mobile-originated (MO) messages are delivered from devices either via IP data over the internet (via cellular) or over IMT. Mobile-terminated (MT) messages are sent to devices via IMT only.

NOTE: If IMT is not available, it will use IP data over satellite if the routing and firewall configurations allow it.

Messaging Hub messages are text only. When sending binary data, it must be encoded as text. Common binary-to-text encodings are Base64 and hex string.

About Channels

Messaging Hub is built off the concept of channels, a list of delivery endpoints that allow each device to support multiple messaging use cases. Multiple channels can be configured per device profile.

As an example, a temperature sensor could be configured to communicate on channel 1 and a precipitation sensor could be configured to communicate on channel 2. In the SkyLink Portal, these 2 channels can be configured to deliver messages to different endpoints (one that processes temperature data and another that processes precipitation data).

The opposite is true as well – MT messages are available on their respective channels for whichever SkyLink-connected processors need them.

NOTE: To configure channels and their respective endpoints, see SkyLink Portal - Channel Management.

DELIVERY ENDPOINTS

Channels allow you to configure delivery endpoints to which the Messaging Hub system will deliver messages received from devices. We currently support the following endpoints:

- **Email** – messages will be emailed to the specified address as JSON file attachments.
- **HTTP(S)** – the message JSON will be delivered via POST to the specified URL.
- **TCP** - the message JSON will be delivered to the TCP address and port as text.

Delivery Message JSON Format

```
{
  "serial": string,
  "payload": {
    "cid": number,
    "data": string
  }
```

SKYLINK DEVICE API

SkyLink has 2 APIs currently available: a REST API and a WebSocket API. The REST API requires a connected service to repeatedly check for or send messages. The WebSocket API can send messages and deliver new ones to clients as they arrive.

Received messages will not be deleted from the queue automatically; be sure to delete a message once your application has received and processed it.

Sent messages will be deleted from the queue automatically. Messages are deleted once they have passed the channel's configured TTL value. Sent and failed messages will be deleted when the TTL time has expired or after 1 week, whichever happens first.

There are 2 types of message priorities: standard and immediate. Standard priority messages are sent to a queue that is processed every 30 seconds, allowing multiple sensor data reports to be collected and sent at the same time for optimization of the minimum Iridium data session size. Immediate priority messages by-pass the queue and are immediately sent to the modem for transmission.

NOTE: Immediate priority messaging is only applicable on the device side and not valid for the Messaging Hub Send API. The Send API does not queue messages internally and sends them immediately (Iridium may queue the messages once they are received). Attempting to send an immediate priority message via the Send API will result in an error.

REST API

The REST API is documented in both an OpenAPI specification and a Swagger UI that can be used to interact with the API. To access, connect to a SkyLink device and navigate to these URLs:

- OpenAPI specification: http://my.skylink.net/hub/openapi
- Swagger UI: http://my.skylink.net/hub/ui/swagger

The device specifications at the above URLs will always show the current firmware version. The endpoint listings on the next page may be out-of-date or incomplete.

Endpoint URL	HTTP Action	Request Data Format	Response	Notes

http://my.skylink.net/hub/v1/messages	GET		200 – Success
			{
			messages: [
			{
			channel_id: number,
			data: string,
			message_id: number,
			priority: ["standard" , “immediate”],
			status: ["received", "queued", "sent", "send_failed"]
			}
			]
			}
			
			401 - Not Authorized
			
			500 - Server Error
				Gets all received messages for all channels.

http://my.skylink.net/hub/v1/messages/send	POST	{
		channel_id: number,
		data: string,
		priority: ["standard", “immediate”]
		}
			201 - Message Sent
			{
			message_id: number
			}
			400 - Data Validation Error
			401 - Not Authorized
			404 - Channel Not Found
			422 - Request Validation Failed
			424 - IMT Topic Not Subscribed
			500 - Server Error
				Sends a message.

http://my.skylink.net/hub/v1/messages/<message-id>	DELETE		204 - No Content - Deleted successfully 401 - Not Authorized 404 - Message Not Found 500 - Server Error	Deletes the message with provided ID from the message queue.
			200 – Success { channels: [{ channel_id: number, name: string, ttl: number, num_received: number, num_queued: number, num_sent: number, num_failed: number	

http://my.skylink.net/hub/v1/channels	GET		}] } 401 - Not Authorized 404 - No channels configured 500 - Server Error	Gets a list of all configured channels. Includes number of received messages.

http://my.skylink.net/hub/v1/channels/<channel-id>	GET		200 – Success
			{
			channel_id: number,
			name: string,
			ttl: number,
			num_received: number,
			num_queued: number,
			num_sent: number,
			num_failed: number
			}
				Gets details of a specific channel.

			401 - Not Authorized 404 - Channel Not Found 500 - Server Error	Includes number of received messages.

http://my.skylink.net/hub/v1/channels/<channel-id>/messages	GET		200 – Success
			{
			received_messages: [
			{
			channel_id: number,
			message_id: number,
			data: string,
			priority: ["standard" , “immediate”],
			inserted_at: date,
			updated_at: date
			}
			],
				Gets messages for specified channel ID.

			sent_messages: [
			{
			channel_id: number,
			id: number,
			data: string,
			imt_message_id: number,
			last_attempt: date,
			modality: ["imt", "http"],
			priority: ["standard" , “immediate”],
			status: ["queued", "queued_imt", "sent", "error", "expired", "failed"],
			error_message: string | null,
			retry_count: number,
			inserted_at: date,
			updated_at: date
			}
			]
			}
			401 - Not Authorized
				Includes both received and sent messages.
				
				Sent messages will be automatically deleted after being sent or retried for the maximum number of times.

			404 - Channel Not Found 500 - Server Error	

http://my.skylink.net/hub/v1/channels/<channel-id>/messages/send	POST	{
		data: string,
		priority: ["standard", “immediate”]
		}
			201 – Success
			{
			message_id: number
			}
			400 - Data validation error
			401 - Not Authorized
			404 - Channel Not Found
			422 - Request Validation Failed
			424 - IMT Topic Not Subscribed
			500 - Server error
				Sends a message for the specified channel ID.

http://my.skylink.net/hub/v1/channels/<channel-id>/messages	DELETE		204 - No Content - Deleted successfully 401 - Not Authorized 404 - Channel Not Found 500 - Server error	Deletes all messages for the channel.

WebSocket API

WebSocket enables a two-way interactive communication session between client and server without having to poll the server for a reply. SkyLink utilizes the Phoenix Framework's channels for its WebSocket implementation.

WEBSOCKET CLIENT LIBRARIES

Libraries exist for many programming languages to interact with this framework. It is highly recommended to use one of these libraries when connecting to SkyLink's WebSocket endpoint. Some example libraries are:

- Swift (iOS)
  - SwiftPhoenix
- Java (Android)
  - JavaPhoenixChannels
- Kotlin (Android)
  - JavaPhoenixClient
- C#
  - PhoenixSharp
- Elixir
  - phoenix_gen_socket_client
- NodeJS
  - phoenix-channels
- JavaScript (Browser)
  - phoenix (docs)
    - via npm

If a library for your language does not exist, or you cannot use a library, here is the documentation to write your own client.

MESSAGING HUB WEBSOCKET API

NOTE: The Phoenix Framework's WebSocket implementation – and therefore this guide - uses the term "channels". Other implementations may use terms such as "topics".

Use a client as described above to connect to the WebSocket endpoint.

WebSocket URL: ws://my.skylink.net/hub/socket

Once connected, the WebSocket client should provide the ability to join a channel after you provide the channel name. WebSocket channel names follow this pattern: "channel:<channel-id>" where <channel-id> is the Messaging Hub channel ID.

Only messages sent to this ID will be delivered to the channel. If you would like to receive messages for any or all channels, join "channel:all".

Upon joining a channel, all of its received messages will be delivered. After this, only new messages will be delivered. To receive all messages again, leave and rejoin the channel.

Message Structure

```
{
 "messages": [
  {
   channel_id: number,
   message_id: number,
   data: string,
   priority: ["standard"],
   inserted_at: date,
   updated_at: date
  }
 ]
}
```

MESSAGING HUB EVENTS

The following events will be delivered by the Messaging Hub:

Event	Data	Notes

mo_message:new	{
	channel_id: number,
	data: string,
	inserted_at: date,
	last_attempt: date,
	imt_message_id: uuid | null,
	modality: null | 'imt' | 'http',
	priority: ["standard" , “immediate”],
	status: 'queued' | 'queued_imt' | 'sent' | 'error' | 'expired' | 'failed'
	attempts: number,
	error_message: null | string,
	updated_at: date
	}
		A new message has been added to the send queue.
		Statuses:
		• queued - message has been added to the internal send queue
		• queued_imt - message has been queued in the modem’s IMT buffer
		• sent - message was sent
		• error - an error has occurred while sending; see error_message
		• expired - send timeout reached
		• failed - system has failed to send the message

mo_message:update	{
	channel_id: number,
	data: string,
	inserted_at: date,
	last_attempt: date,
	imt_message_id: uuid | null,
	modality: null | 'imt' |
	'http',
	priority: ["standard" , “immediate”],
	status: 'queued' | 'queued_imt' | 'sent' | 'error' | 'expired' | 'failed'
	attempts: number,
	error_message: null | string,
	updated_at: date
	}
		Send queue message has been updated. This will happen as the message’s status gets updated.

	{ channel_id: number, data: string, inserted_at: date, last_attempt: date, imt_message_id: uuid | null,	

mo_message:delete	modality: null | 'imt' | 'http', priority: ["standard" , “immediate”], status: 'queued' | 'queued_imt' | 'sent' | 'error' | 'expired' | 'failed' attempts: number, error_message: null | string, updated_at: date }	Send queue message was deleted.

mt_message:new	{
	channel_id: number,
	message_id: number,
	data: string,
	priority: ["standard"],
	inserted_at: date,
	updated_at: date
	}
		A new message has been received.

	{ channel_id: number, message_id: number, data: string,	Received message has been updated. This event may never be received as received messages are not typically

mt_message:update	priority: ["standard" , “immediate”], inserted_at: date, updated_at: date }	updated at this time after the new event has been sent. May be used in the future.

mt_message:delete	{
	channel_id: number,
	message_id: number,
	data: string,
	priority: ["standard" , “immediate”],
	inserted_at: date,
	updated_at: date
	}
		Received message was deleted.

MESSAGING HUB API COMMANDS

The WebSocket API currently supports the following commands:

Command	Data	Response	Notes

send:message	{
	channel_id: number,
	data: string,
	priority: ["standard" , “immediate”]
	}
		Success:
		{
		id: number,
		channel_id: number,
		imt_message_id: number | null,
		data: string,
		last_attempt: date | null,
		modality: "http" | "imt" | null,
		priority: "standard" | “immediate”,
		status: "queued" | "queued_imt" | "sent" | "error" | "expired" | "failed",
		error_message: string | null,
		attempts: number,
		inserted_at: date,
		updated_at: date
		}
			Sends a message to the specified channel ID. An error is returned if the channel is not configured.
			
			A successful send command will generate a new MO message event for the channel. Subsequent events will be generated when the message is updated (modality and status). All events will include the same data as the success message.

		Error:
		{
		"reason": string
		}	

delete:message	{
	"message_id": number
	}
		Success:
		{
		message_id: number,
		channel_id: number,
		data: string,
		priority: "standard" | “immediate”,
		inserted_at: date,
		updated_at: date
		}
		
		Error:
		{
		"error": string
		}
			Deletes the specified received message and returns the message object that was deleted.

delete:sent	{
	"message_id": number
	}
		Success:
		{
		id: number,
		channel_id: number,
		imt_message_id: number | null,
		data: string,
		last_attempt: date,
		modality: "http" | "imt",
		priority: "standard" | “immediate”,
		status: "queued" | "queued_imt" | "sent" | "error" | "expired" | "failed",
		error_message: string | null,
		attempts: number,
		inserted_at: date,
		updated_at: date
		}
		
		Error:
		{
		"error": string
		}
			Deletes the specified message from the queue. If the message is in the queue or has an error or expired status, it will not be sent or retried.

clear:channel	{
	"channel_id": number
	}
		Success: No data returned
		
		Error:
		{
		"error": string
		}
			Deletes all received messages for the specified channel.

MESSAGING HUB SEND API

The Messaging Hub API provides a programming interface to send Messaging Hub messages to IMT devices from the internet. There are 2 endpoints available: one to send messages and another to retrieve the status of the sent message.

API Keys, which are created in the SkyLink Portal, are required for authentication with the Messaging Hub Send API to send messages.

Endpoint URL	HTTP Action	Request Data Format	Response	Notes

https://messaging.skylink.net/api/v1/mt/send/serial/<serial_number>/priority/<priority>	POST	Message Body:
		{
		cid: number,
		data: string
		}
		
		Message Header:
		X-api-key : <api-key>
		
		Message Parameters:
		serial_number - Device serial number displayed on SkyLink Portal. It has the following format <XXXX-XXXX>. i.e., 0000-03DE
			201 - Message Sent
			{
			"status": "OK",
			"messageId": string,
			"requestReference": string
			}
			
			400 - Data Validation Error
			{
			"messageId": string,
			"status": "Failed",
			"message": string
			}
				Sends a message to the device with specified serial number with the specified channel ID and priority.

		priority - Acceptable values: "standard"	401 – Unauthorized
			
			403 - Forbidden
			{
			"messageId": string,
			"status": "Forbidden",
			"message": string
			}
			
			404 - Channel Not Found
			{
			"messageId": string,
			"status": "NotFound",
			"message": string
			}
			
			500 - Server Error
			{
			"messageId": string,
			"status": "Failed",	

			"message": string }	

https://messaging.skylink.net/api/v1/mt/status/serial/<serial_number>/requestReference/
<request_reference>
	GET	Message Header:
		X-api-key : <api-key>
		
		Message Parameters:
		serial_number - Device serial number displayed on SkyLink Portal. It has the following format <XXXX-XXXX>. i.e., 0000-03DE
		
		request_reference - Request reference UUID returned in the send message response.
			201 - Message Status
			{
			"requestReference": string,
			"status": "OK",
			"data": {
			"type": "mt",
			"serial": string,
			"messagePending": false | true,
			"status": "queued_with_aws" |"success" | "message_discarded_on_overflow" | "message_expired" | "message_timed_out" | "message_cancelled" | "unprovisioned" | "subscription_invalid" | "topic_invalid" | "no_response_to_ring" | "not_registered" | "message_failed" | "crc_error_in_transfer" | "local_crc_error" | "delayed_access" | "service_disabled" | "registration_expired" | "malformed_json" , "messageId": string
				Retrieves status of previously sent MT message.

			}
			}
			
			401 – Unauthorized
			
			403 - Forbidden
			{
			"requestReference": string,
			"status": "Forbidden",
			"message": string
			}
			
			404 - Message was not found
			{
			"requestReference": string,
			"status": "NotFound",
			"message": string
			}
			
			500 - Server Error	

			{
			"requestReference": string,
			"status": "Failed",
			"message": string
			}	

SKYLINK PORTAL

Create an API Key

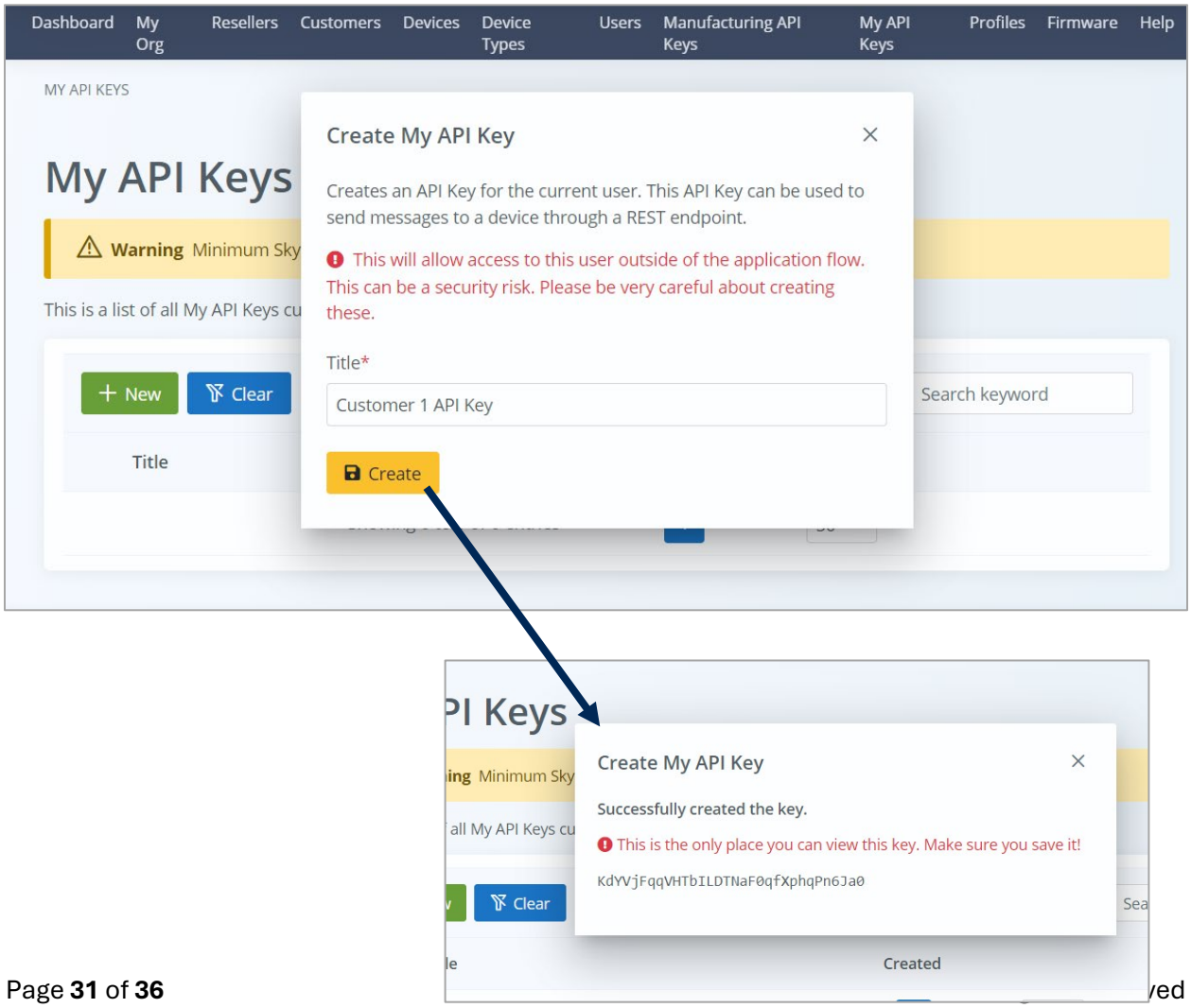
An API Key is used to authenticate the message sent from the customer's application to a device(s).

The Messaging Hub Send API section describes *all* endpoints that need an API Key to be passed in the Message Header.

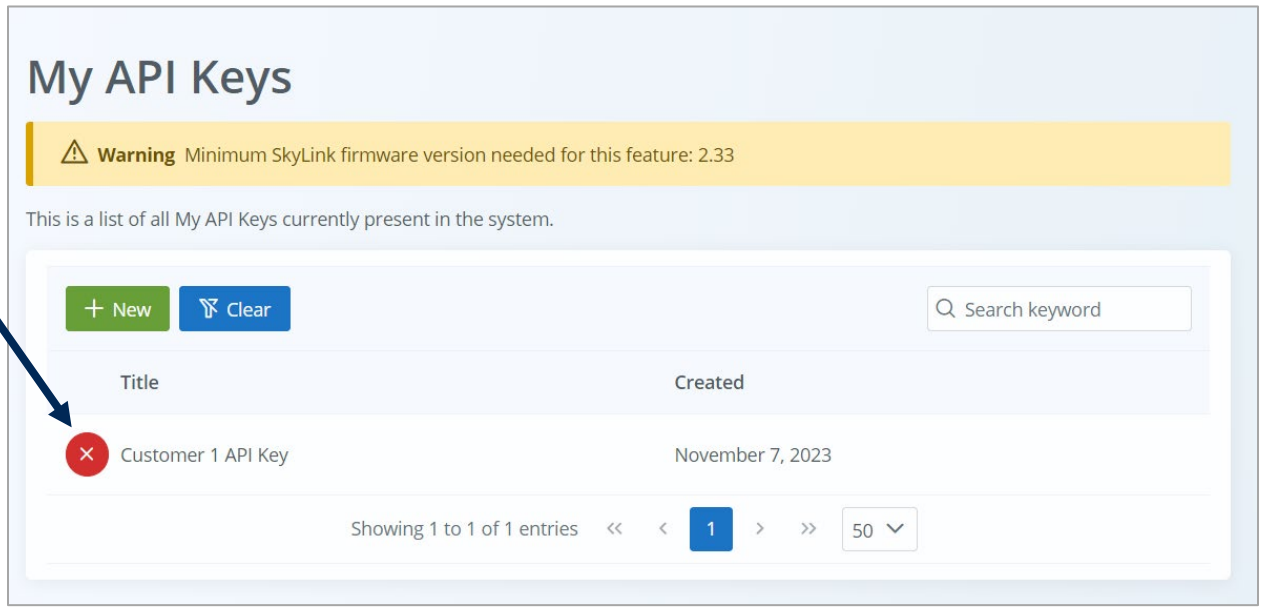
To create an API Key:

1. Under the desired user account, log in to the SkyLink Portal
2. Navigate to the My API Keys screen, where you can create, delete, and see a list of names for all existing API Keys.
3. Click on New, complete the mandatory Title field, then press 'Create.'

WARNING: Once you click 'Create,' you will see that particular API Key ONLY once. Be sure to save it!



To remove or revoke an API Key, click the red X next to the desired API Key title.

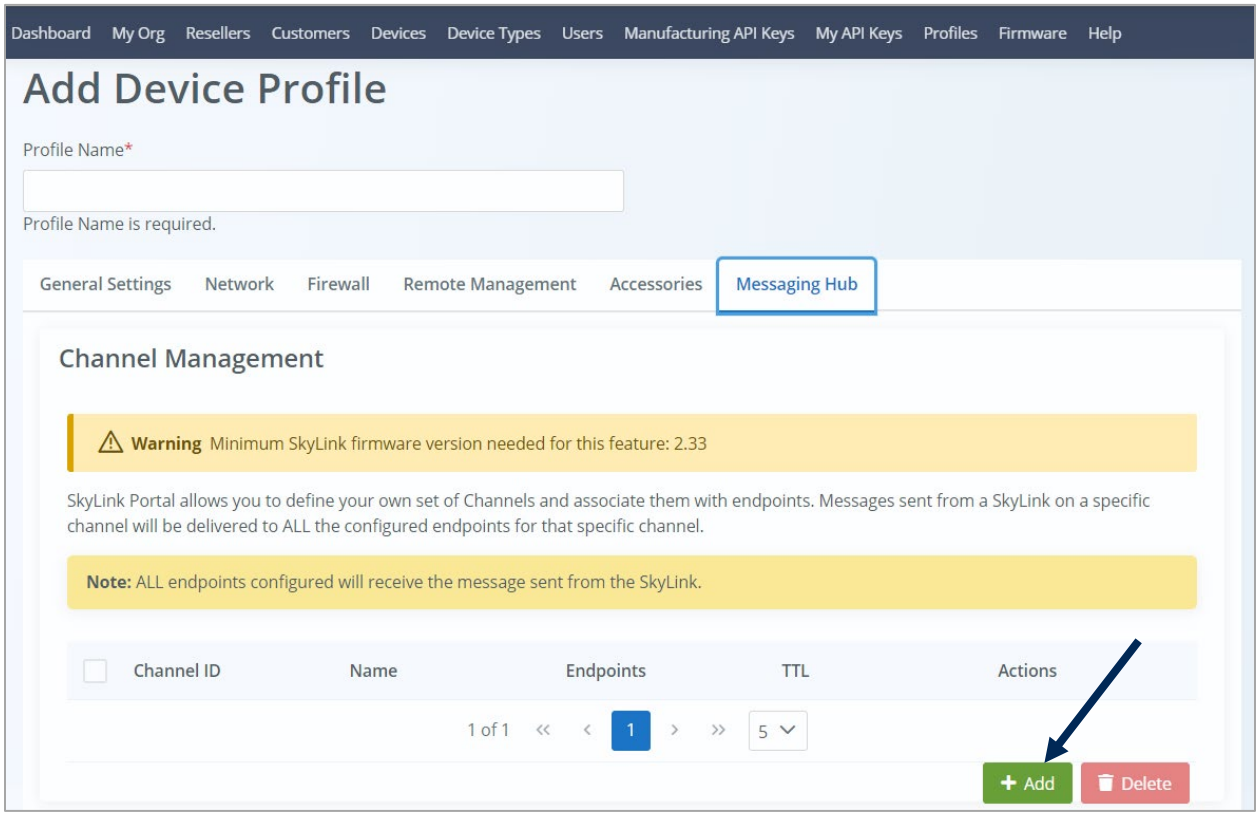


Channel Management

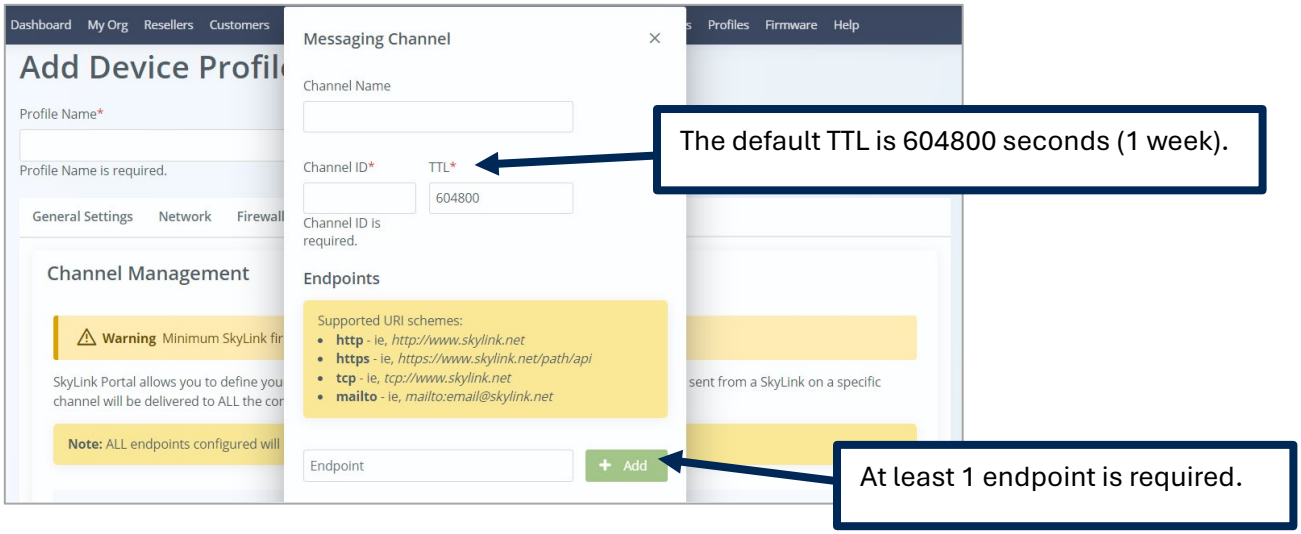
Messaging Hub channels can be added, edited, and deleted in the SkyLink Portal on the Profiles screen. Navigate to 'Device Profiles' and click on the 'Messaging Hub' tab.

To create a channel:

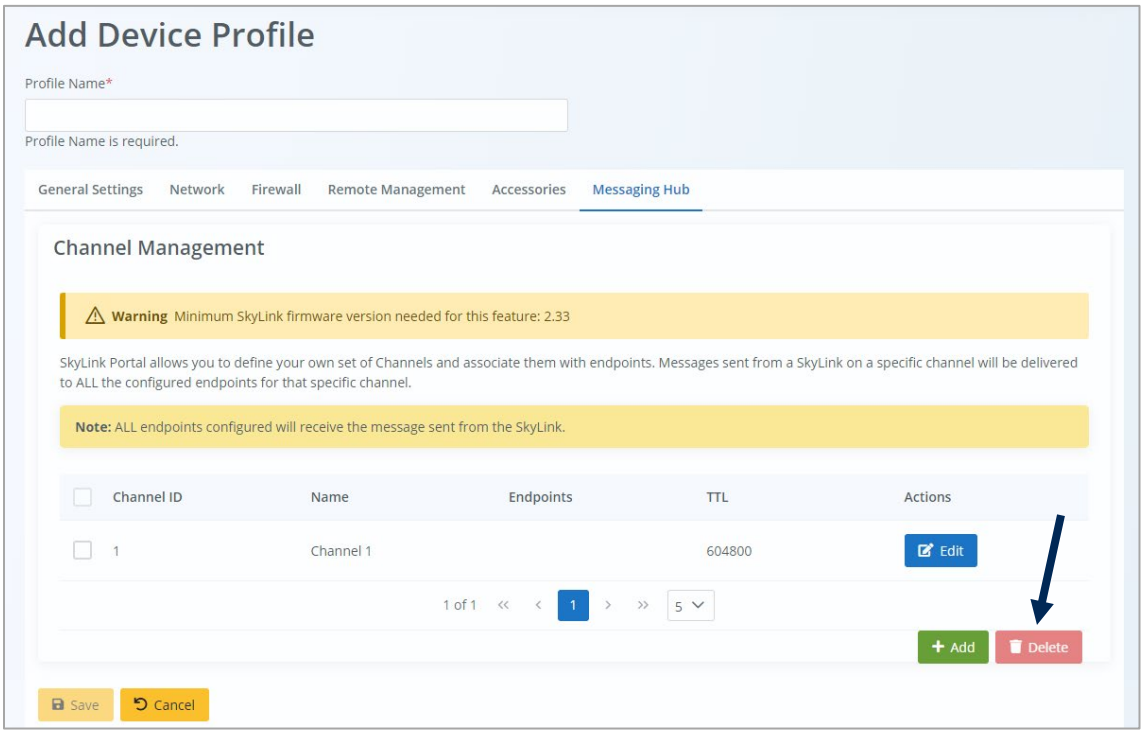
1. Click on the green 'Add' button.



2. Complete the optional and required fields.



3. Press ‘Save.’ A list of all channels will be displayed on the screen. To remove, select any or all desired channels and press ‘Delete.’ A dialog box will ask you to confirm the action.



REMINDER: After creating channels to a profile, you must associate that profile with a device so that the channels and other configurations are transmitted to it. Please see the SkyLink Cloud Services User Guide for more information.

APPENDIX A: REVISION HISTORY

Date	Ver.	By	Description
13 Nov 2023	0.0	MZ	Initial Draft
21 Nov 2023	0.1	MZ	Public Release Draft
28 Nov 2023	0.2	MZ	Corrected pg. 6 hyperlink
29 May 2024	0.3	MZ	Updated address
5 June 2024	0.4	MZ	Added 'Messaging Hub Events' section
15 August 2024	0.5	MZ	Corrected URLs under 'Messaging Hub Send API' section
12 March 2025	1.0	MZ	Removed high priority messaging; added content about immediate messaging; updated design into cobrand template
29 April 2025	1.1	MZ	Added 'IMPORTANT' message in the 'Introduction' section

TECHNICAL SUPPORT

NAL Research is committed to providing the highest level of service and support. If you have any questions or concerns, please feel free to contact us by email or phone; contact information is available at the bottom of this page.

Thank you for choosing NAL Research!

NAL Research, operating under the brands NAL Research and Blue Sky Network

11100 Endeavor Ct, STE 300, Manassas, VA, 20109, United States

Support@NALResearch.com | +1 858-551-3894 | www.NALResearch.com