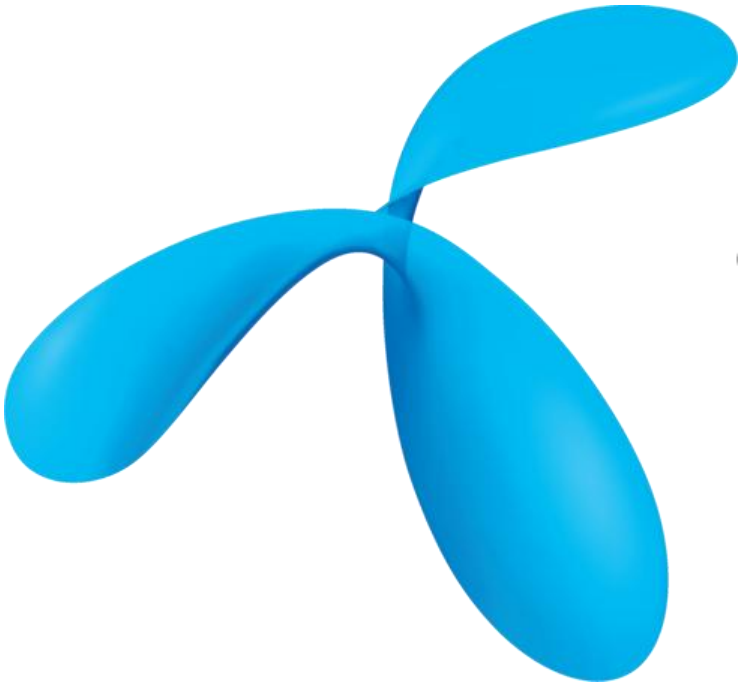


Telenor SMC API 2.0
User Change Documentation V2.0

Table of contents

1. Introduction

This document provides information to the new serverless version of the SMC API, developed to replace the legacy system. The new API is built on AWS cloud-native services, offering a modern, scalable, and secure infrastructure for accessing and managing sports-related data such as leagues, matches, events, players, and live statistics.

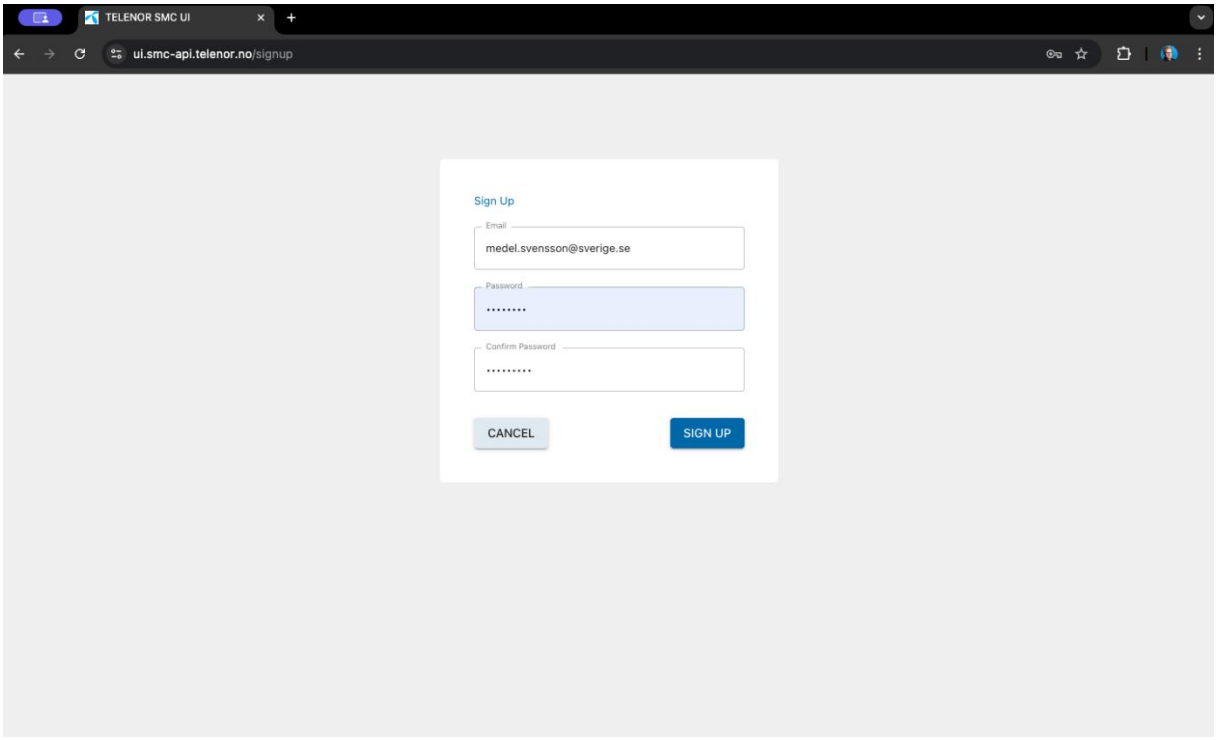
The primary goal of this documentation is to support developers and integrators in understanding and adopting the updated API. While the core structure and endpoints remain largely consistent with the legacy version to ensure backward compatibility, several critical enhancements have been introduced:

- API Key Management is now handled through AWS API Gateway, enabling automated provisioning, usage plans, and rate limiting.
- Access Control has been refined with support for granular access levels (e.g., read-only, write, admin) and league-specific permissions.
- Production Access is separated from sandbox environments, requiring explicit approval for live data usage.
- League Access Requests are now formalized, ensuring that only authorized users can retrieve or modify data for specific leagues.
- Data Format Improvements include the transition from integer-based IDs to string-based identifiers, improving compatibility with distributed systems and UUID standards.
- A queueing service was set up as a “Push API” to which customers can subscribe in order to get the events from the SMC API.

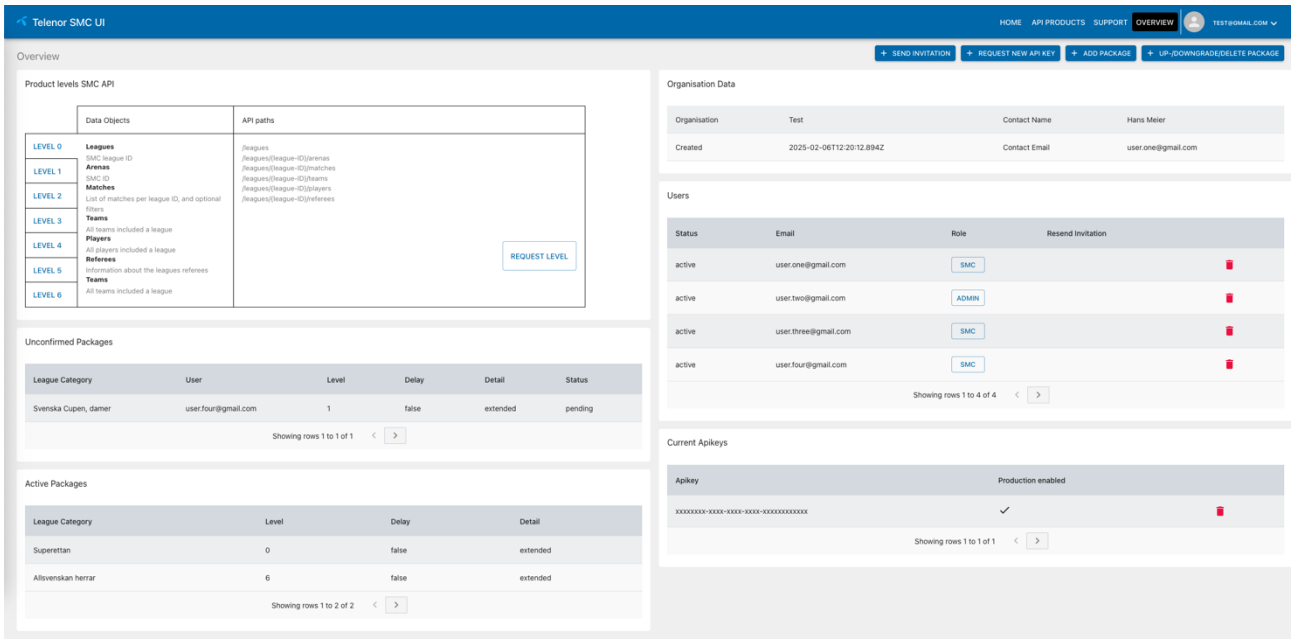
This documentation describes the changes and guides through the UI to get started with the new API.

2. Registration to the new API

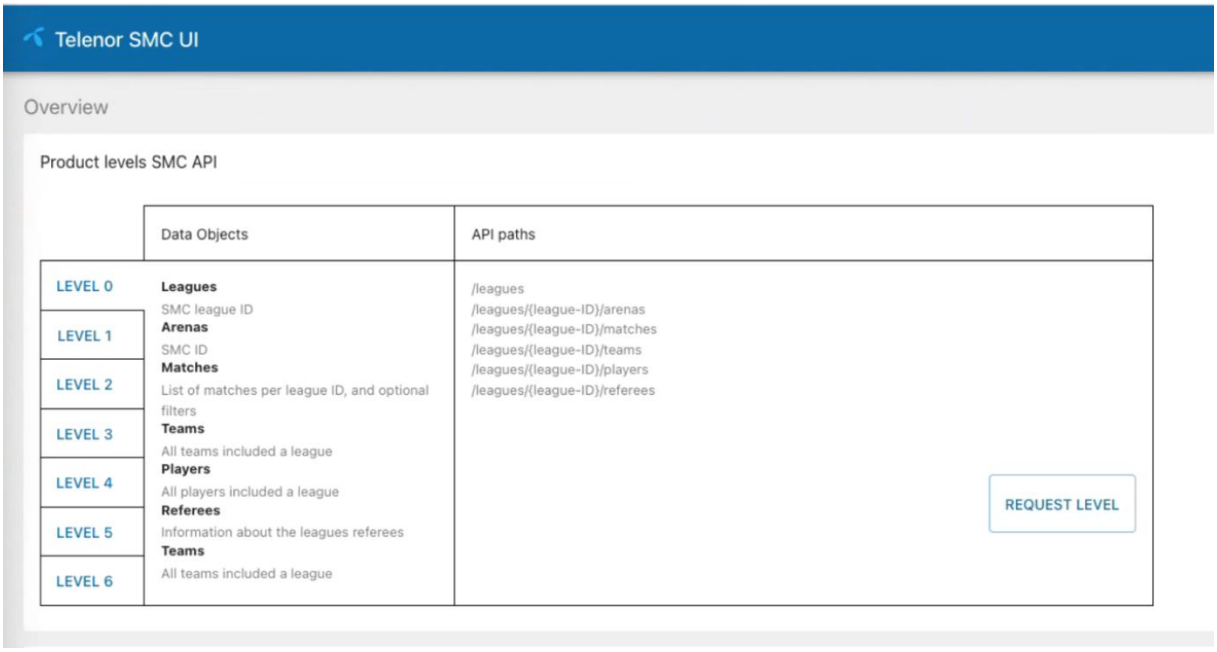
To get access to the new API please register at https://ui.smc-api.telenor.no/. You will receive a confirmation e-Mail from Telenor SMC<noreply@ui.smc-api.telenor.no> with a code, that you need to provide in the next step. If you do not see the e-Mail in your Inbox, please also check your spam folder.



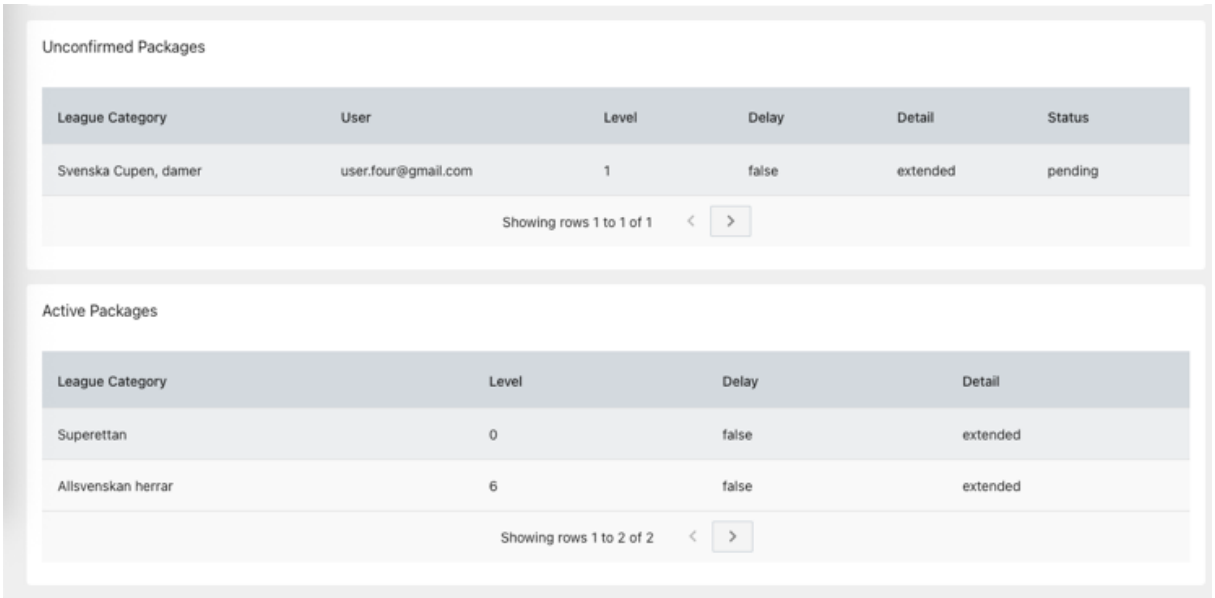
Once registered you can Access the APIs UI portal to administrate access to leagues, request further detail levels, and manage API Keys.



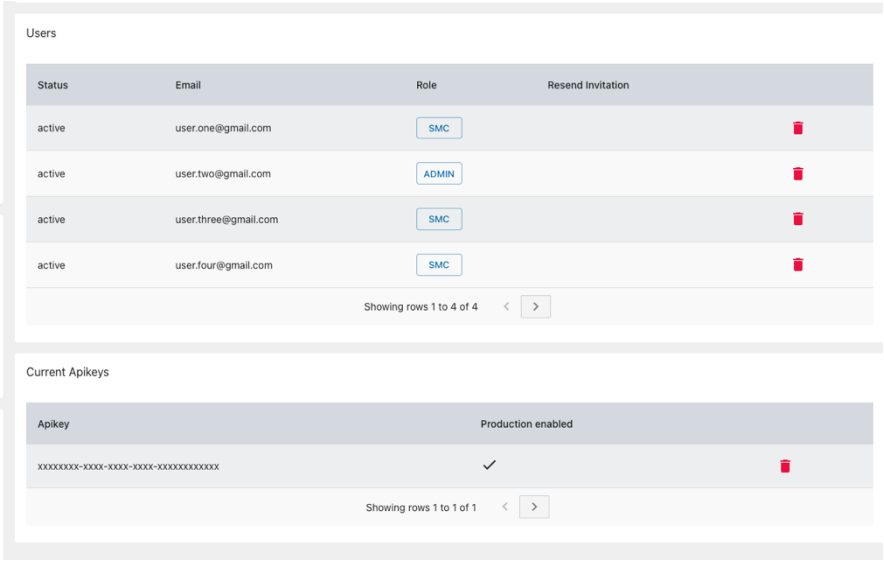
The landing-page to the portal gives you an overview to your organization and the booked league packages.



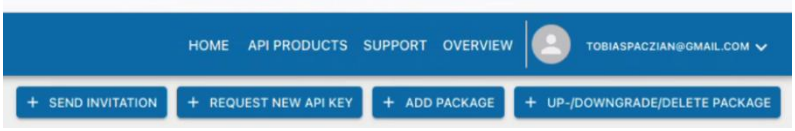
Via the levels in the upper left corner, you can see which Level includes what details and what endpoints of the API can be consumed at each Level. If a Level is not booked yet, you can easily request the level using the “request level” button. Levels include lower Levels. If you ask for Level 3 you get access to Level 0, 1, 2 and 3. But you need to ask for access to Level for each League.



Below the Product Levels you can see the league packages that you have requested but the confirmation is pending and the league packages that are active already, showing you to which leagues you have access and to which level.



To the right you can see your organization data as well the Users that already have access to the portal in order to manage the settings from your organization and the role they own. Below the users you can see the current API Keys that you have requested and if they have access to the production system. You can easily delete current API Keys or Users by via the bin icon.



Using the Buttons in the upper right corner you can send an invitation to your team members in order to get access to the UI Portal as well or request a new API Key (e.g. for production access) and add or manage further league packages.

Note: If you run into problems during the setup please contact Lars.Carlsson@telenor.no from Telenor. We will help you to get started.

3. Base URL

You can reach the new API under the following base-URL:
https://smc-api.telenor.no

4. API Key Handling

Once you get the API Key you need to provide it in your API Requests in the Header, as follows: *--header 'Authorization: 61xxxb6d-7d31-4f19-8xdd-6433xxxxx2bb'*

```
cURL
curl --location --request GET 'https://smc-api.telenor.no/leagues/01JQV8CGVWY99E0ZNJTZNN4G2K/matches/01JQV8GV5J0AR4W6T3W5Q4Q4SZ' \
--header 'Authorization: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx' \
--header 'Content-Type: application/json'
```

5. Onboarding Process for new customers

As a new customer please contact Lars.Carlsson@telenor.no after registering to request an organization which you will get assigned to. This is a one-time step. Once you are assigned to the organization you can invite further members and administrate the access to the platform yourself.

Note: Organizations/customers from the legacy API are migrated to the new API before the launch.

6. Changes to the API compared to the legacy SMC API

In the following segment the major changes are described that come with the switch from the legacy to the new SMC API.

6.1. IDs stored as Strings instead of Integers

All IDs generated by the new SMC API are now represented as strings instead of integers. This change affects all internal identifiers such as league_id, match_id, event_id, player_id, arena_id and team_id. The shift to string-based IDs improves compatibility with distributed systems and NoSQL databases that are used in the new API. It ensures greater flexibility and consistency across services.

Developers integrating with the new API should ensure that IDs are no longer parsed or handled as integers in their applications, and should update any validation logic or database schemas accordingly to support string-based identifiers.

The internal IDs for league, match, player, arena and team are unique for each league and season. Player_id for a player in Allsvenskan 2025 are not the same as for Allsvenskan 2026. Please use Fogis context to if consistent IDs needed.

Note: This change does not apply to the external FOGIS IDs - which retain their original format.

6.1.1. Internal Identifiers are scoped to a specific league

All internal identifiers (IDs) are scoped to a specific league. This means that entities such as arenas, which may be shared across multiple leagues, are represented separately for each league within the system. Each instance will have a distinct internal ID tied to its respective league. As a result, the same physical arena may appear multiple times, each with a unique internal ID corresponding to the league it belongs to. To ensure consistent identification and cross-league referencing, a stable external ID is provided in all contexts. This external ID serves as a universal reference point, enabling reliable resolution of entities regardless of their league-specific internal representations.

6.2. Endpoint Changes

6.2.1. Field Name Update in Medical Treatment Events

Endpoint: /leagues/{league-id}/matches/{match-id}/events/medical-treatment
Change: The field previously named match-phase has been renamed to phase.
Reason: This aligns the schema with other event types, which already use the phase field for consistency.

6.2.1. Extended Referee Information.

Endpoint: /leagues/{league-id}/referees
New Fields Added:

- Name
- referee-team-id
- referee-team-name

Note: While referee-team-id and referee-team-name are now part of the response schema, they currently return empty string values as the data is not yet available in the database.

6.2.2. New Event Type: Offside

A new event type, offside, has been introduced.

Endpoint: /leagues/{league-id}/matches/{match-id}/events/offside
Function: Returns all offside events for a given match.

Note: GET /leagues/{league-id}/matches/{match-id}/events will now include offside events in the aggregated event list.

6.2.3. New End-Point with Fogis Context

We have implemented support for Fogis IDs. See specified end-points below.
Context name = fogis
Ext-IDs are all Fogis IDs. League, players, match, team....

Type of request	End-Point
GET	/{context-name}/leagues
GET	/{context-name}/leagues/{ext-league-id}/referees

GET	/{context-name}/leagues/{ext-league-id}/players
GET	/{context-name}/leagues/{ext-league-id}/players/{ext-playerId}
GET	/{context-name}/leagues/{ext-league-id}/players/{ext-playerId}/live-tracking/stats/individual
GET	/{context-name}/leagues/{ext-league-id}/players/{ext-playerId}/live-tracking/stats/individual/aggregated
	(Required query parameter aggregateFunction missing (must be SUM or AVG)) example:
	?aggregateFunction=SUM
GET	/{context-name}/leagues/{ext-league-id}/matches/{ext-match-id}/events/corner
GET	/{context-name}/leagues/{ext-league-id}/matches/{ext-match-id}/events/free-kick
GET	/{context-name}/leagues/{ext-league-id}/matches/{ext-match-id}/events/goal
GET	/{context-name}/leagues/{ext-league-id}/matches/{ext-match-id}/live-stats
GET	/{context-name}/leagues/{ext-league-id}/matches/{ext-match-id}/events/match-phase
GET	/{context-name}/leagues/{ext-league-id}/matches/{ext-match-id}/events/medical-treatment
GET	/{context-name}/leagues/{ext-league-id}/matches/{ext-match-id}/events/offside
GET	/{context-name}/leagues/{ext-league-id}/matches/{ext-match-id}/events/penalty
GET	/{context-name}/leagues/{ext-league-id}/matches/{ext-match-id}/events/red-card
GET	/{context-name}/leagues/{ext-league-id}/matches/{ext-match-id}/events/shot
GET	/{context-name}/leagues/{ext-league-id}/matches/{ext-match-id}/live-tracking/stats/individual
GET	/{context-name}/leagues/{ext-league-id}/matches/{ext-match-id}/events/substitution
GET	/{context-name}/leagues/{ext-league-id}/matches/{ext-match-id}/events/yellow-card
GET	/{context-name}/leagues/{ext-league-id}/teams/{ext-team-id}
GET	/{context-name}/leagues/standings
GET	/{context-name}/leagues/{ext-league-id}/live-goals-matches-played
GET	/{context-name}/leagues/{ext-league-id}/live-assists-matches-played
GET	/{context-name}/leagues/{ext-league-id}/live-gk-saves-matches-played
GET	/{context-name}/leagues/{ext-league-id}/live-public
GET	/{context-name}/leagues/{ext-league-id}/live-public-round
GET	/{context-name}/leagues/{ext-league-id}/live-public-team

GET	/{context-name}/leagues/{ext-league-id}/live-public-league

7. SMC Push API

In the SMC UI you can generate a queue and subscribe it to multiple topics. You will receive a queue URL, an Acccess Key and a Secret Access key. Using the code below, you can access the messages on the queue.

This is a nodeJS sample for polling (you have to set the QUEUE_URL, ACCESS_KEY and SECRET_ACCESS_KEY variable in the environment and do an npm install for @aws-sdk/client-sqs):

The new push contains the same information but is structured a little different.

```
// import the aws library
const { SQSClient, ReceiveMessageCommand, DeleteMessageCommand } = require("@aws-sdk/client-sqs");

// set the queue URL
const QueueUrl = process.env.QUEUE_URL;

// get the authentication from the environment
const AccessKey = process.env.ACCESS_KEY;
const SecretAccessKey = process.env.SECRET;

// create a client with the credentials
const client = new SQSClient({
credentials:{
accessKeyId: AccessKey,
secretAccessKey: SecretAccessKey
}
});

// main function
async function main () {

// construct the polling command
const command = new ReceiveMessageCommand({
QueueUrl: QueueUrl, // URL of the queue
MaxNumberOfMessages: 1, // number of messages until the receive is triggered
VisibilityTimeout: 3, // seconds to wait before another worker can see and pick up the message
WaitTimeSeconds: 20, // seconds to long poll (maximum 20)
});
```

```
// keep listening to the queue messages
while (true) {
try {
console.log('initiating long poll');

// this will wait a maximum of WaitTimeSeconds, but trigger immediately
// once MaxNumberOfMessages messages are received
const response = await client.send(command);
if (response) {
let messages = response.Messages;

// check if there are messages, will be undefined if there are none
if (messages) {

// iterate over the messages (maximum MaxNumberOfMessages)
for (let message of messages) {

// call the message handler
await handleMessage(message);
}
}
}
} catch (e) {
console.log('something went wrong, aborting');
console.log(e);
return;
}
}

}

// process the message
async function handleMessage (message) {

// the message content is in message.Body
console.log('received message:\n'+message.Body);

// if the message has been handled successfully, delete it from the queue
const command = new DeleteMessageCommand({
QueueUrl: QueueUrl,
ReceiptHandle: message.ReceiptHandle,
});
try {
await client.send(command);
console.log('message handled successfully');
} catch (e) {
console.log('something went wrong deleting the message');
}
}

main();
```

8. Support and Contact

If you have any questions regarding the onboarding or the handling of the API, please contact mathias.krispersen@telenor.no from Telenor.