FLIGHT INFO ALERTS

Getting Started

www.oag.com

Contents

Product Overview

www.oag.com

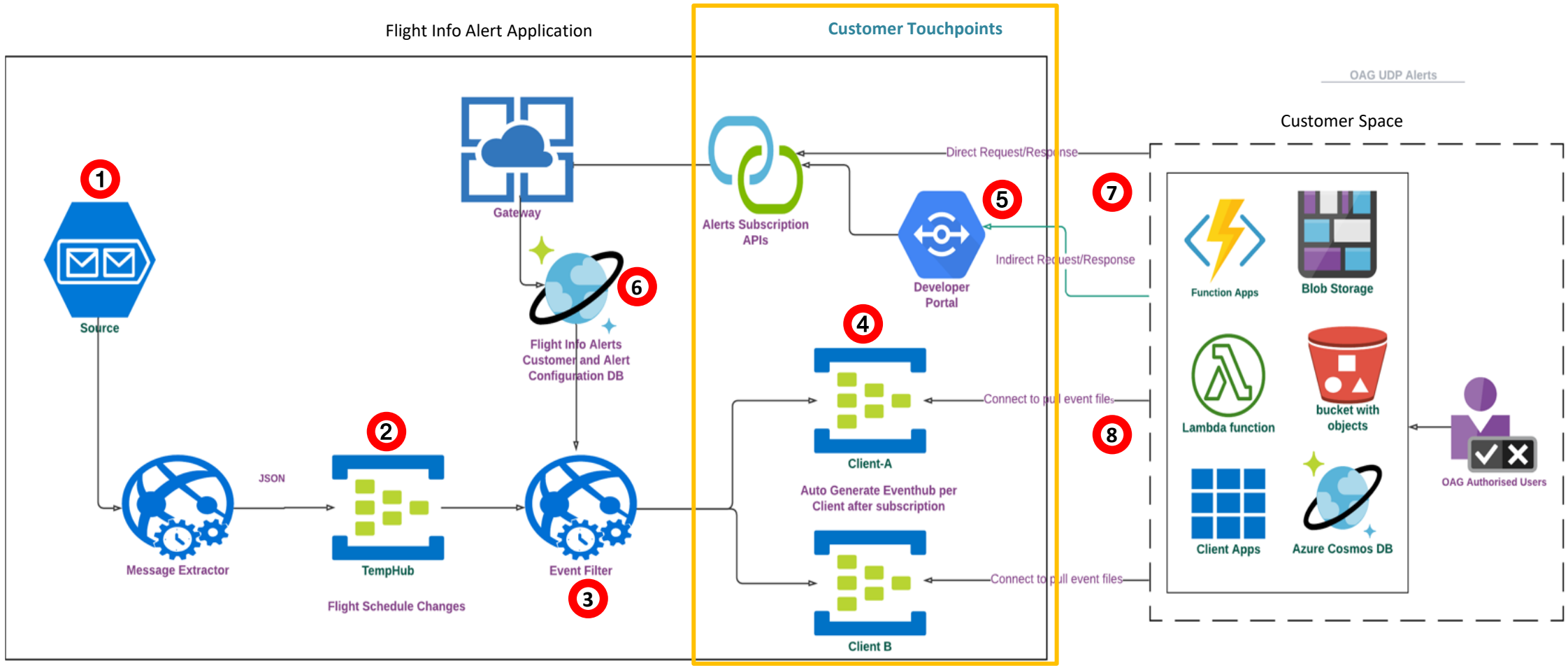
About Flight Info Alerts

- Alert customers to the changes to Flight data such as schedules and status changes, and deliver the information to them without the need for them to look up on the API
- Deliver changes in a fast and modern way, making it simple and easy for customers to see the updated schedules
- Give customers options about what changes they see, and in what frequency
- Delivery via universal Azure Event Hubs allowing customers to access the data quickly and easily
- Receive changes in real time, with flexibility on how often the data is collected to suit customer requirements

Key Functionality & Data

- Change events relating to both schedules and status changes
- Core schedules and status data fields - being updated as we add more features and functionality; full updated list can be found in our Knowledge Hub
- Notification and receipt of updated flight Information based on minimum set of alert criteria-
  - Carrier Code
  - Flight Number
  - Flight Number Range
  - Departure and/or Arrival Airport
  - Departure and/or Arrival Date
  - Schedules or Status information

Technical Overview



Note: Key on following page.

Technical Overview

1. The Flight Info Alerts application is connected to our METIS platform to source change events as they happen
2. Each change event is extracted and converted into JSON
3. Events are matched to customer-defined filters called ‘Alert Configurations’
4. Matched events are then streamed to the customer’s Event Hub for collection
5. New customers will subscribe to the Flight Info Alerts product through the OAG Developer Portal
6. Customer profiles and alert configurations are used to filter events to the customer’s Event Hub
7. Customers can use the available endpoints to manage their alert configuration directly on the Developer Portal or indirectly using their choice tool or application
8. Customers can connect to their Event Hub to collect flight change events at their desired frequency using their application

Note: Alerts are held in the Event Hub for a rolling 7 days.

Getting Started

Accessing the OAG Developer Portal

Pre-requisite

You will need an OAG Developer Portal account by going to Home – OAG Developer Portal. Also, you can read about the Flight Info Alerts product.

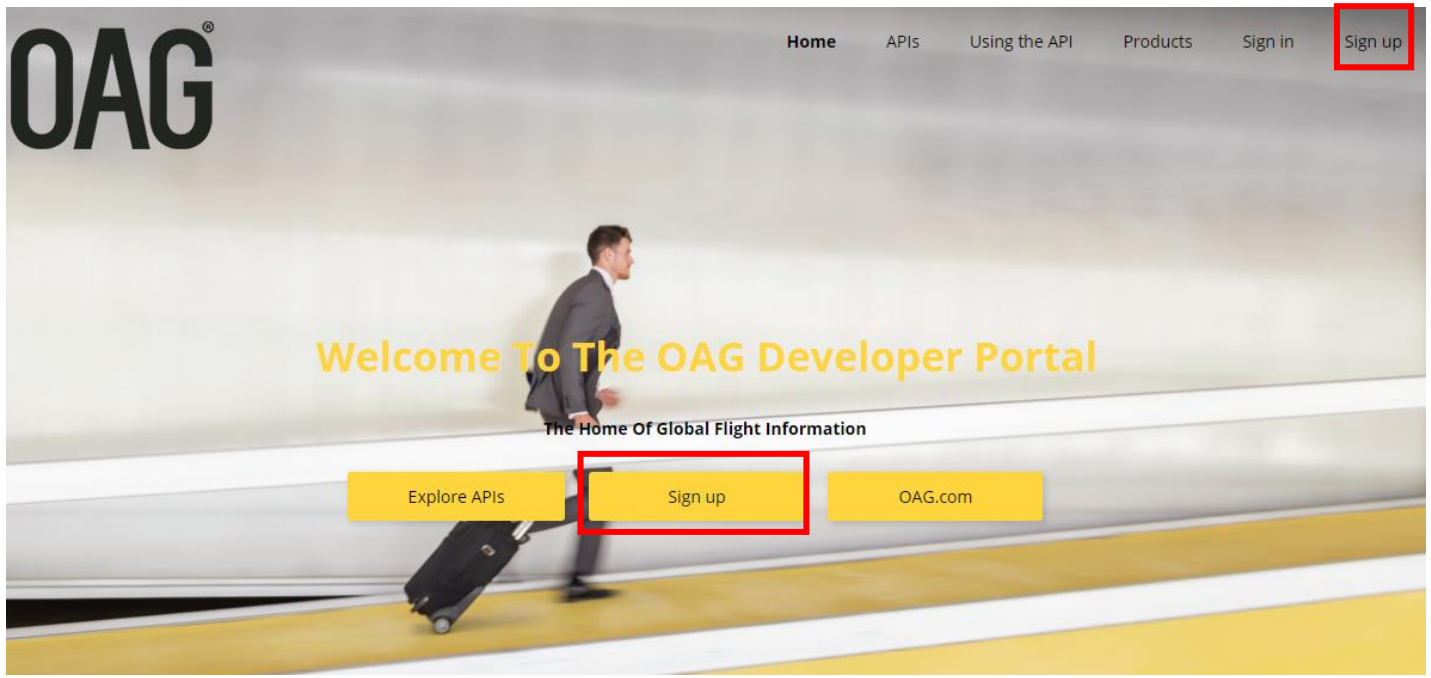
- The dedicated address for the OAG Developer Portal:

https://developers.oag.com/

Signing up

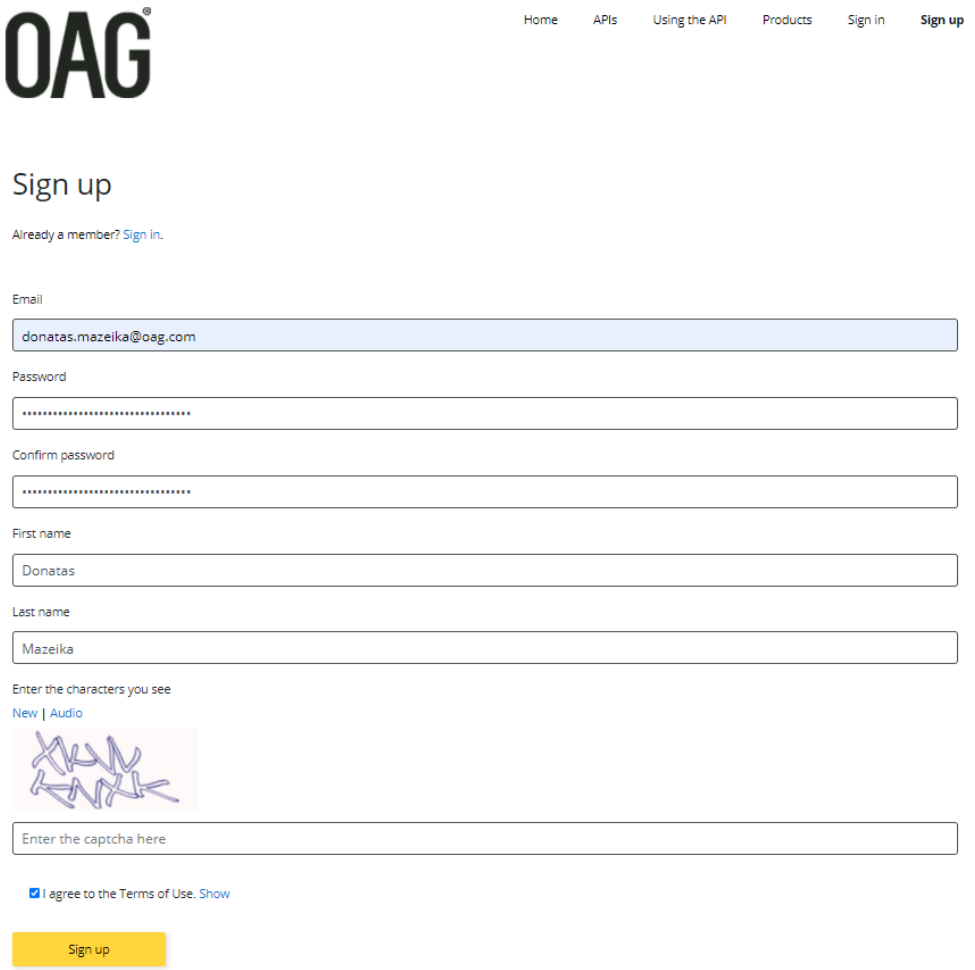
1

Select the “Sign Up” button from the main page



2

Fill your details, agree to the Terms of Use and press the “Sign Up” button



After registration, you will receive the confirmation email.

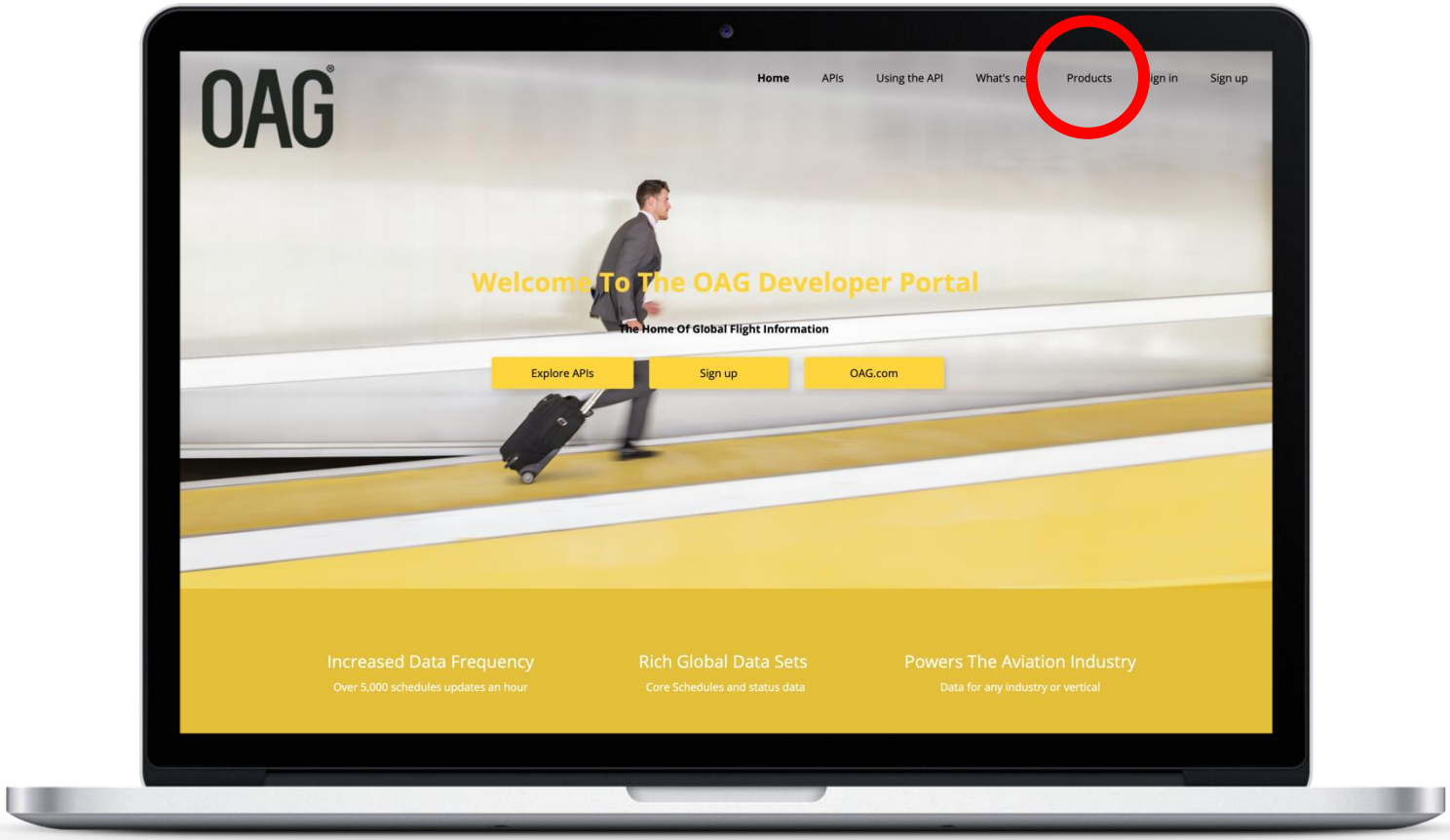
Subscribe to Flight Info Alert Product

Pre-requisite

You will need an OAG Developer Portal account by going to Home – OAG Developer Portal. Also, you can read about the Flight Info Alerts product.

Step one

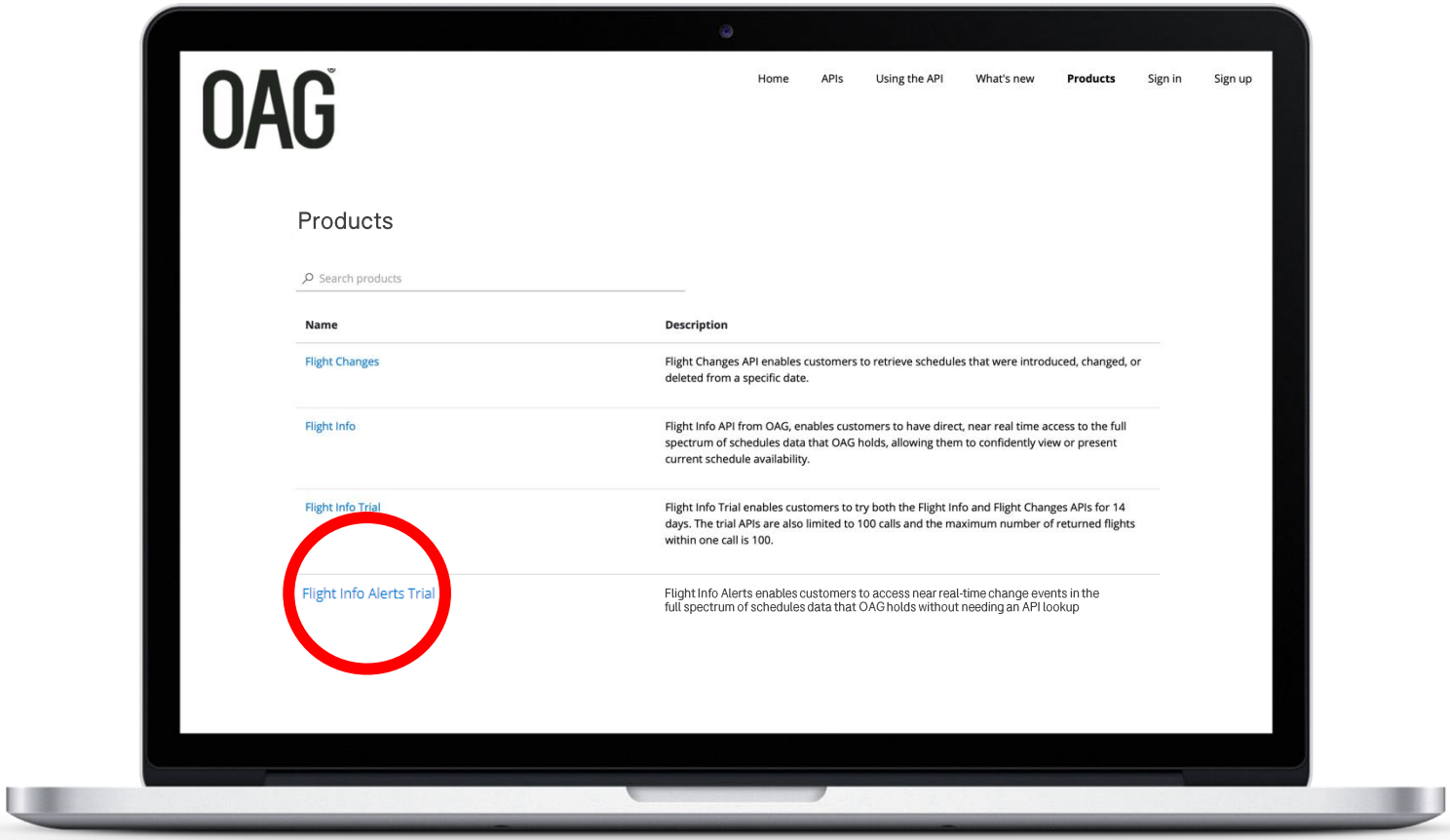
On the home page you could either; select “products” from the top right menu or scroll down to “Our APIs” and select “Try it out”.



Subscribe to Flight Info Alert Product

Step two

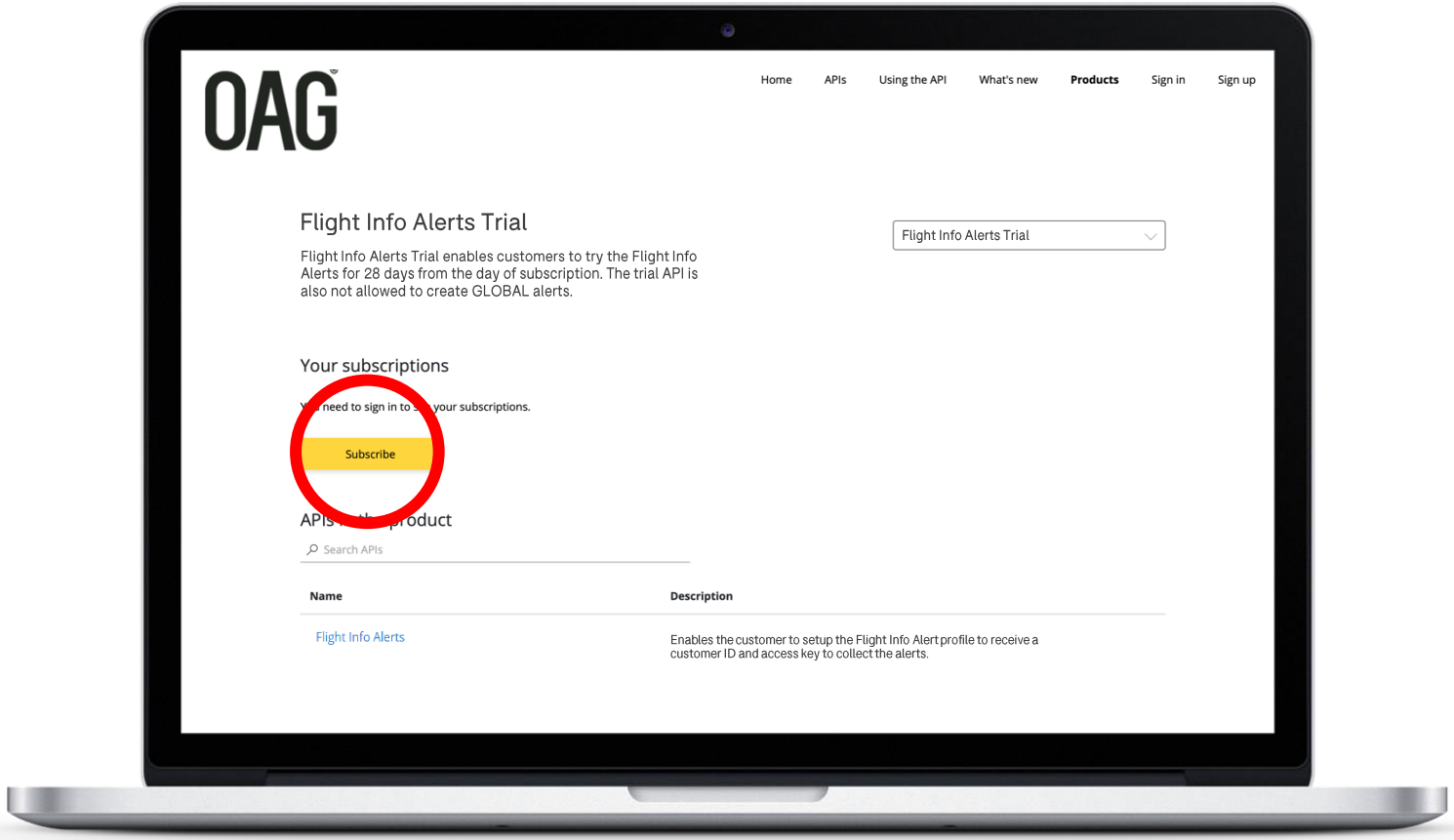
Select “Flight Info Alerts” on the product page.



Subscribe to Flight Info Alert Product

Step three

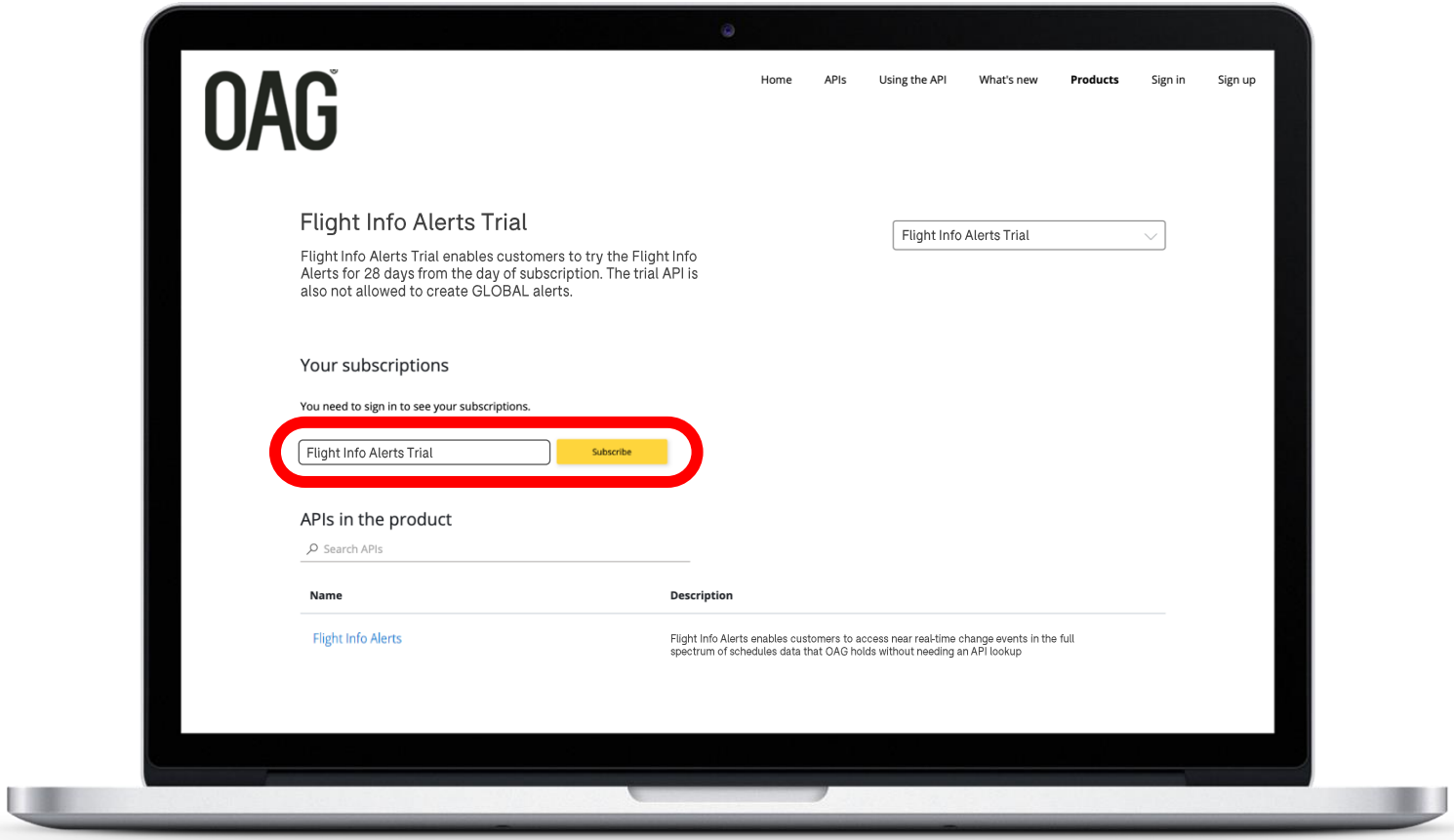
Click the “Subscribe” button to subscribe to Flight Info Alert.



Subscribe to Flight Info Alert Product

Step four

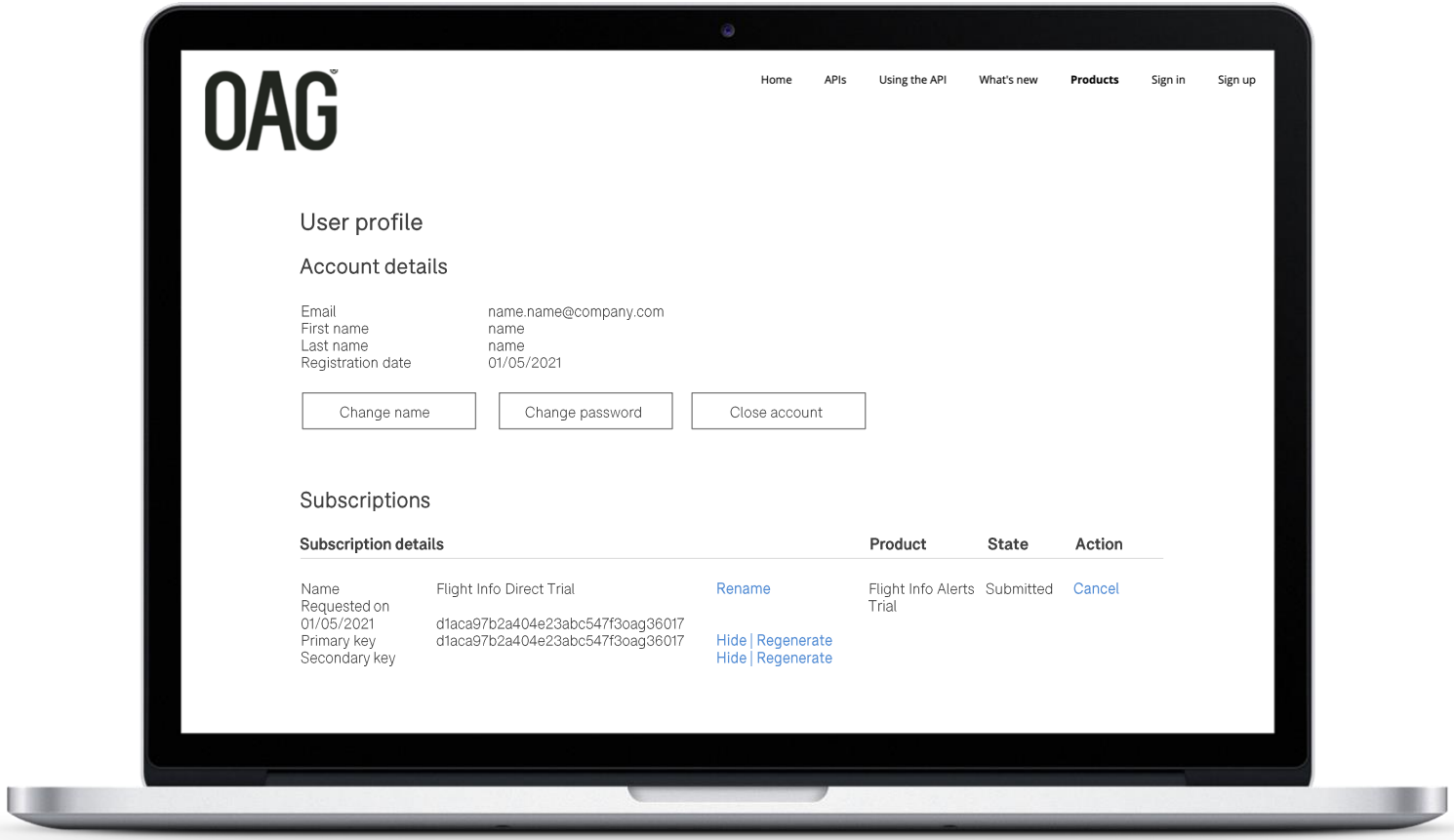
Enter your company name to subscribe and select the subscribe button.



Subscribe to Flight Info Alert product

Step five

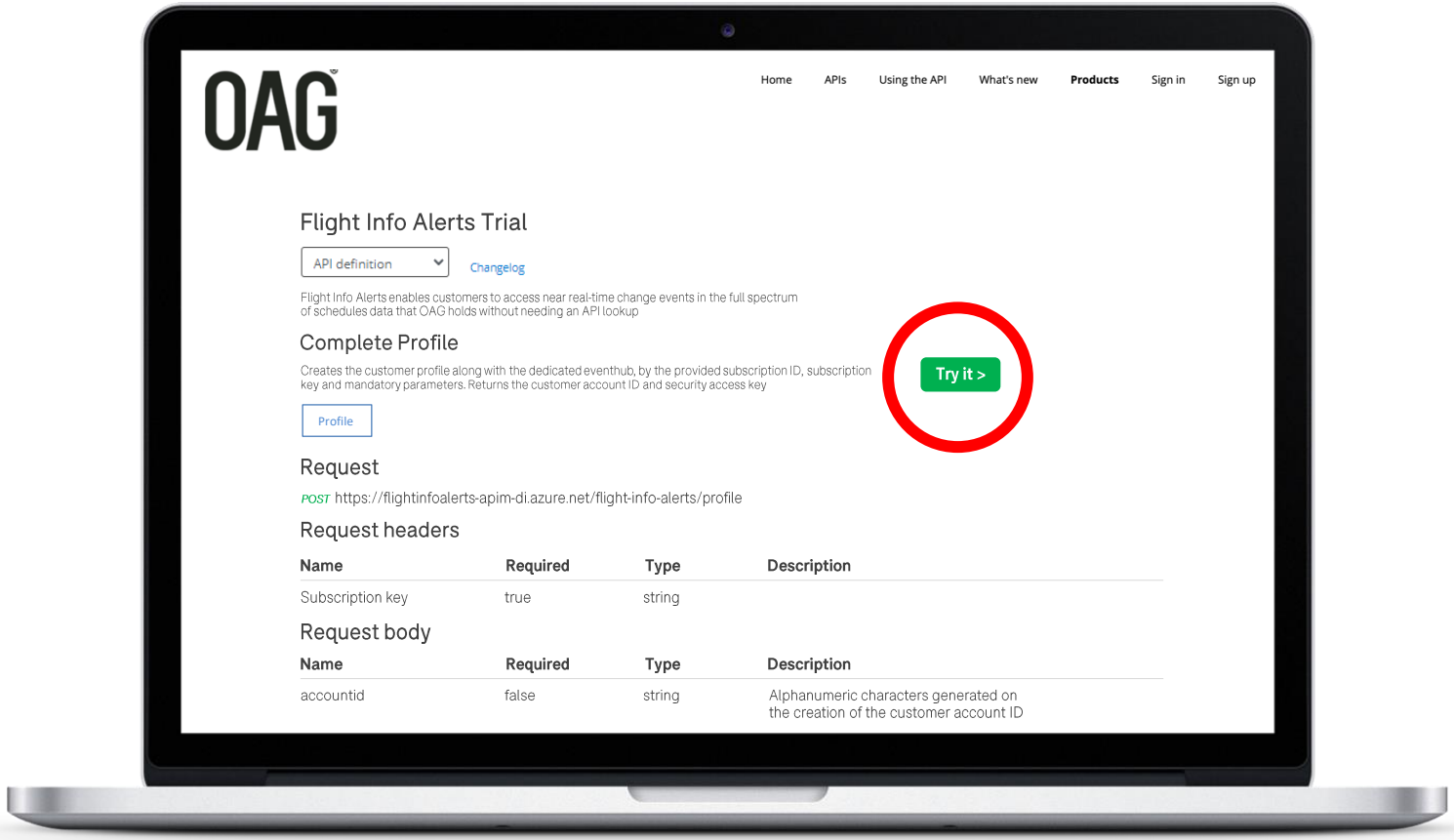
You will be redirected to the profile page, where you will see your subscription keys. Subscription keys are deactivated, pending approval. Also, you will receive an acknowledgement email confirming your subscription and a welcome email upon the activation of the subscription keys.



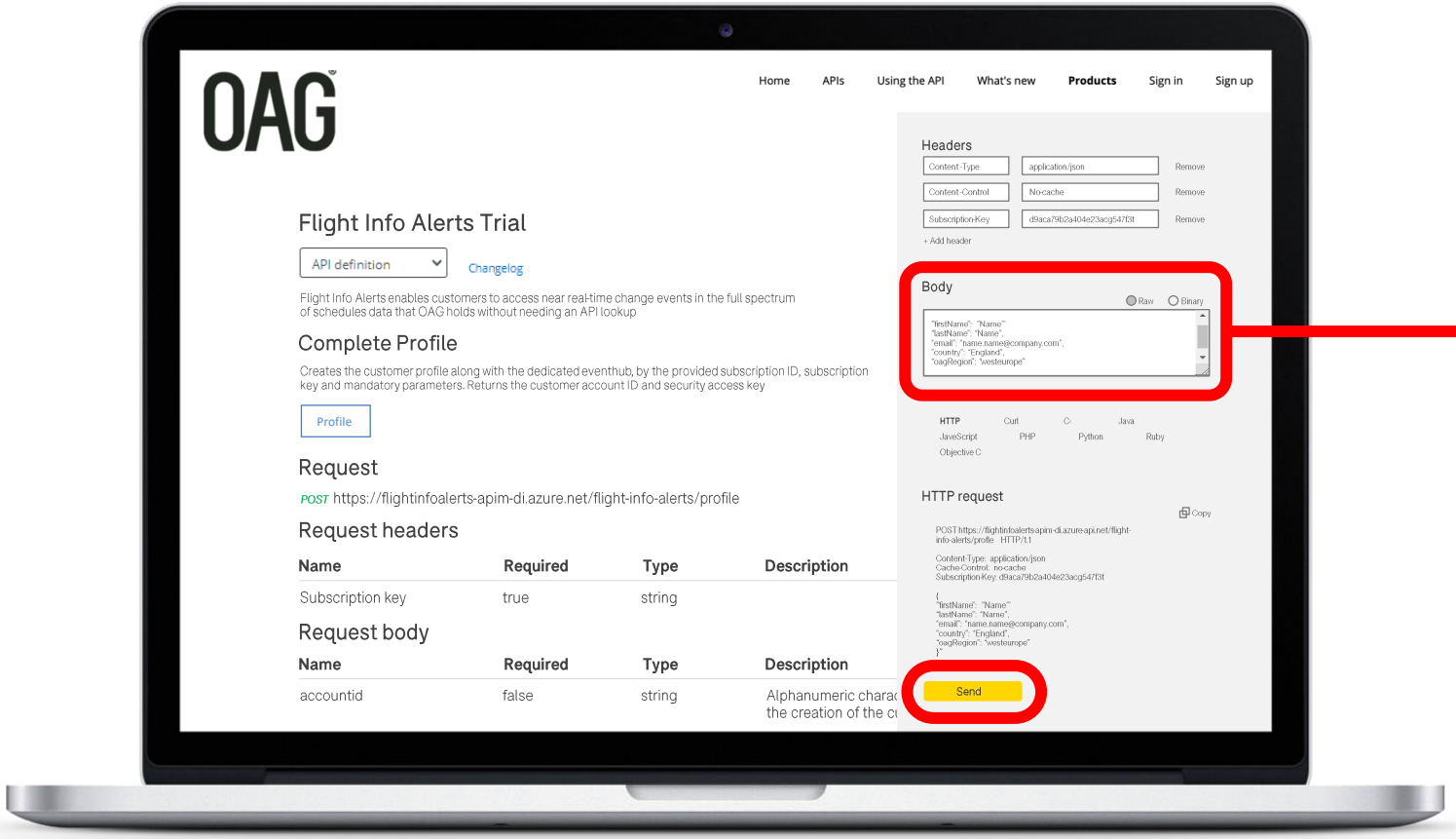
Profile Completion

Step six

When the subscription is confirmed, you will need to complete your profile by navigating to Products > Flight Info Alert > Complete Profile



Profile Completion

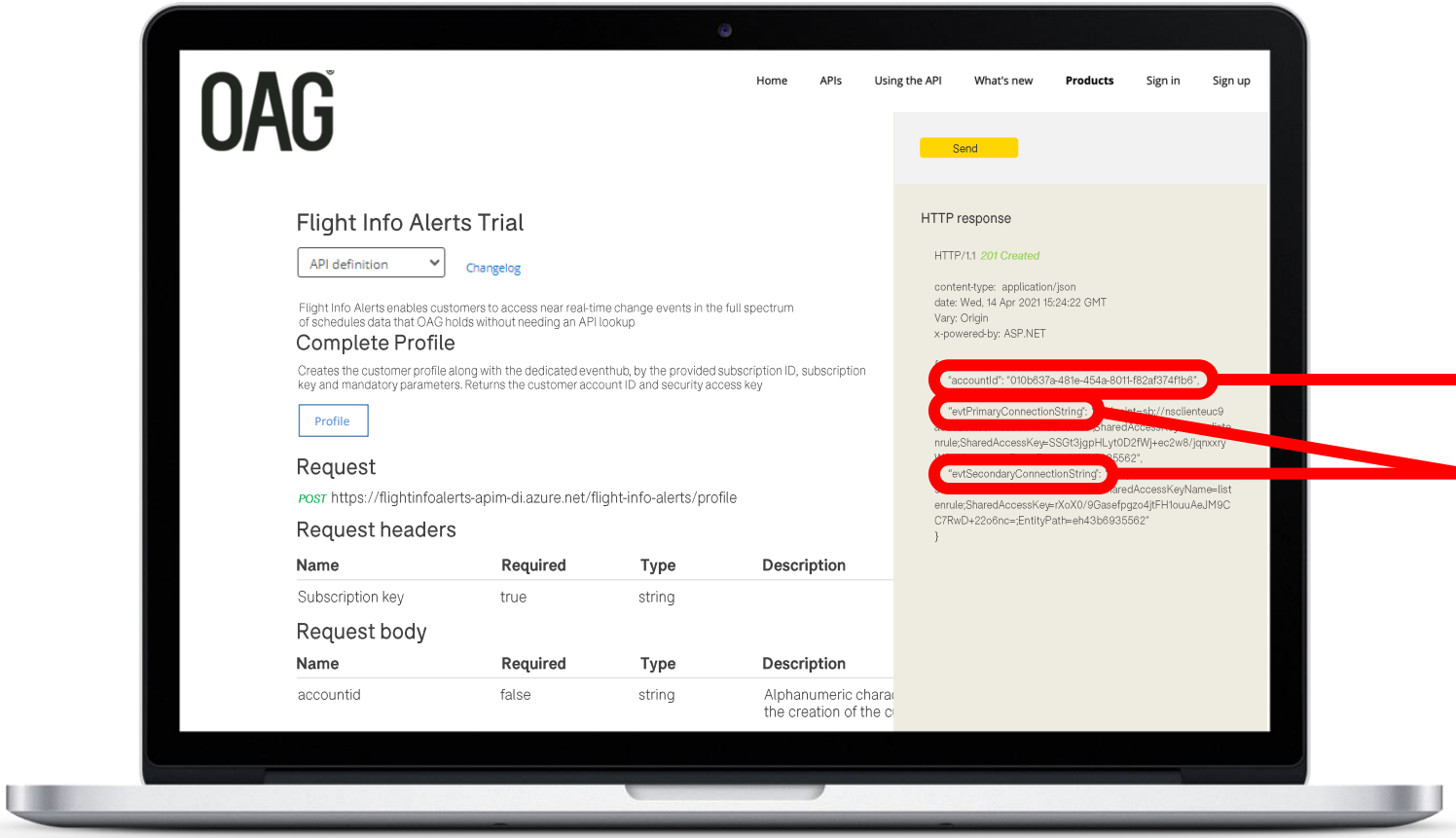


Step seven

You can complete your profile by entering your details into the provided request “body” fields.

```
{
  "firstName": "string",
  "lastName": "string",
  "email": "string",
  "secondaryEmail": "string",
  "country": "string",
  "content": "string",
  "companyName": "string"
}
```

Profile Completion

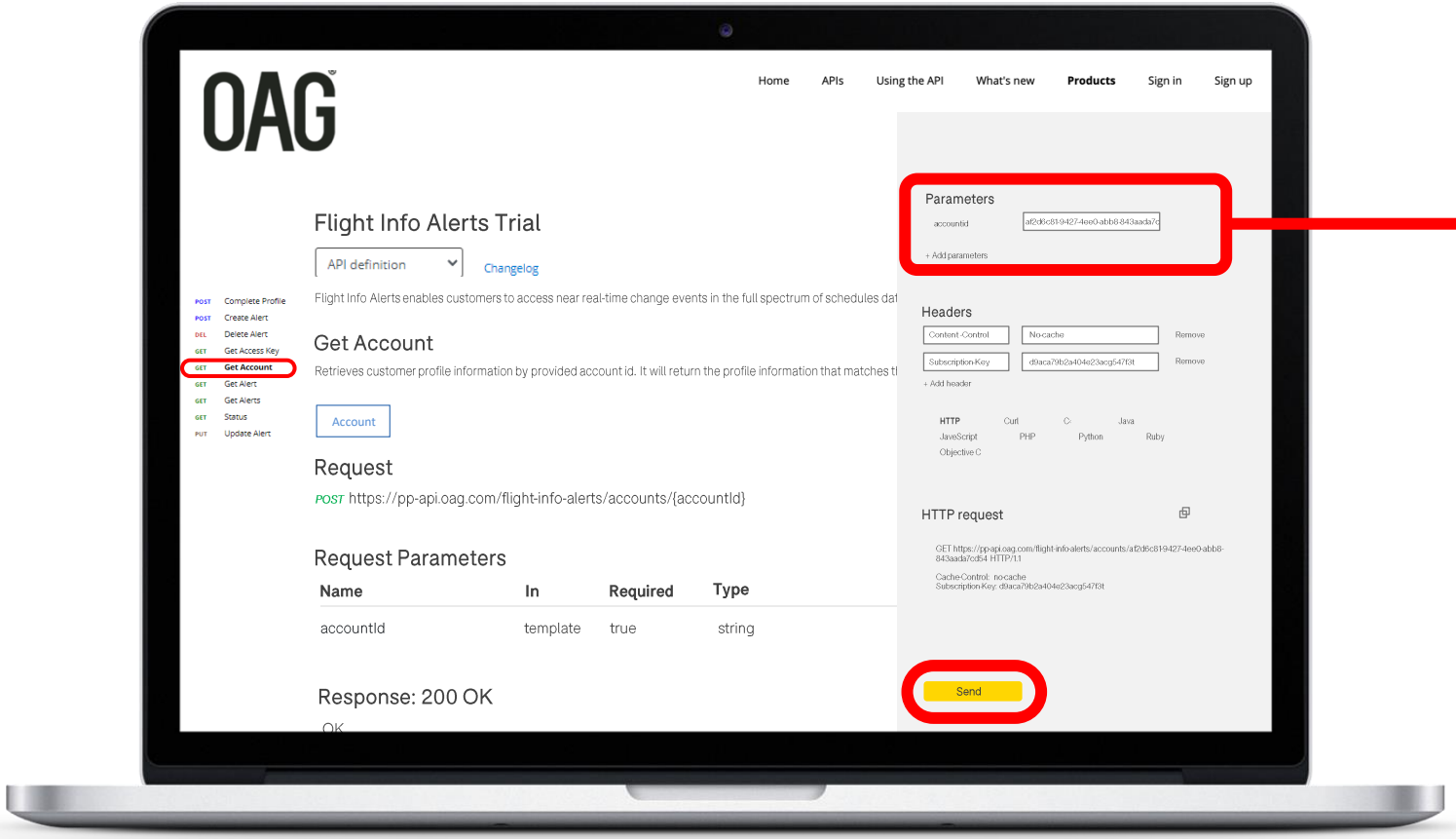


Step eight

You'll need your account id to create, update, delete and retrieve your alerts.

Primary and secondary access keys to connect to your dedicated Event Hub.

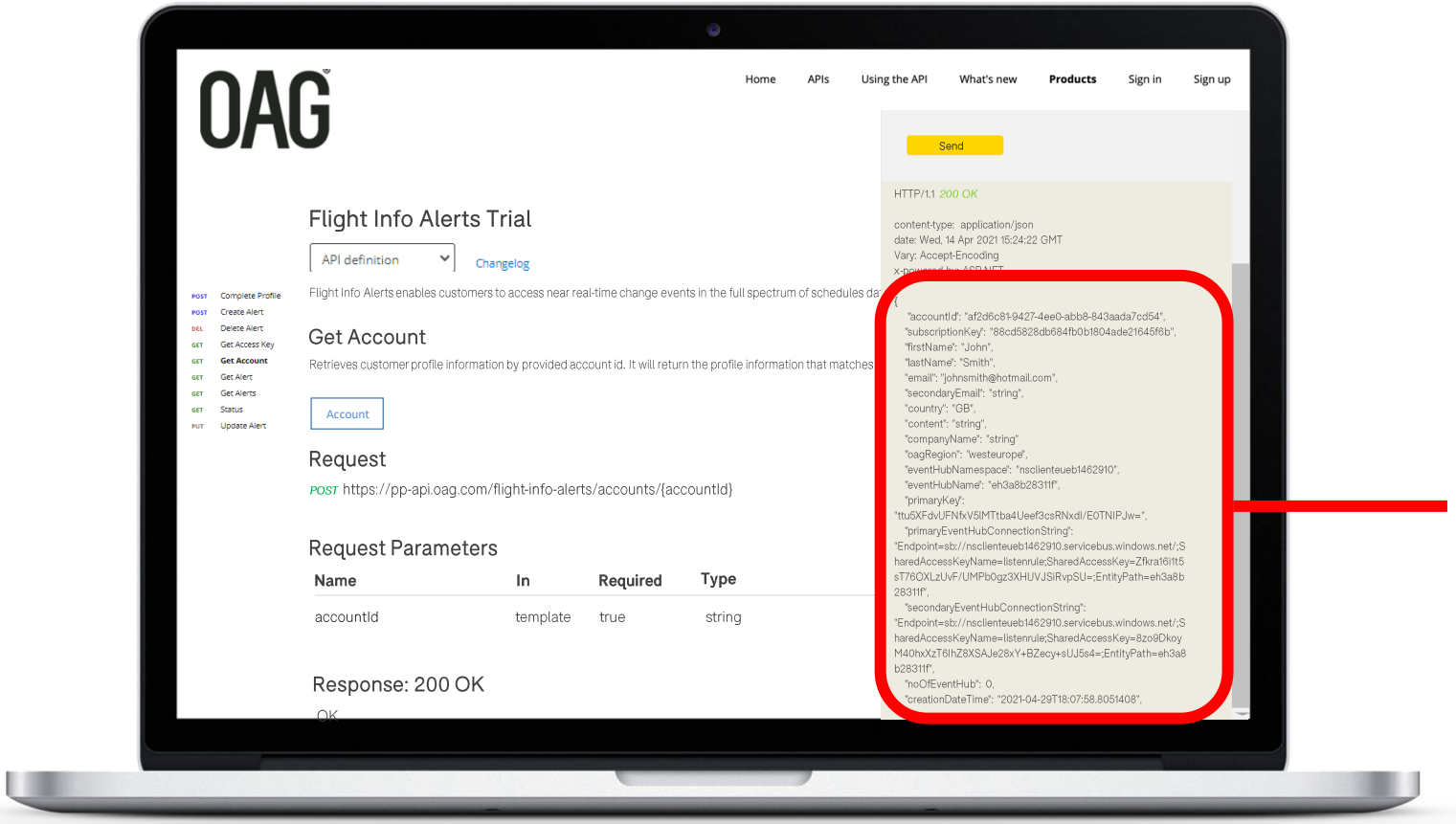
Get Account Profile



Step nine

You can retrieve your account details using the “Get Account Profile” method, by providing your account id

Get Account Profile



Your full account details including your account status, type of account, whether you have created alert configurations, and includes seat information will be returned

Creating Alerts

www.oag.com

Alert Configuration Types

Global Alert

Alerts will be received on all schedule changes. No additional parameter is required

Note: The global alert configuration encompasses all other alert types. It is also not possible to create other alerts when a global alert is active, and vice versa.

Port Alert

Alerts will be received on all changes for either the departure or arrival port, with departure or arrival date as optional parameters. No additional flight data parameters are required and if given, will be ignored.

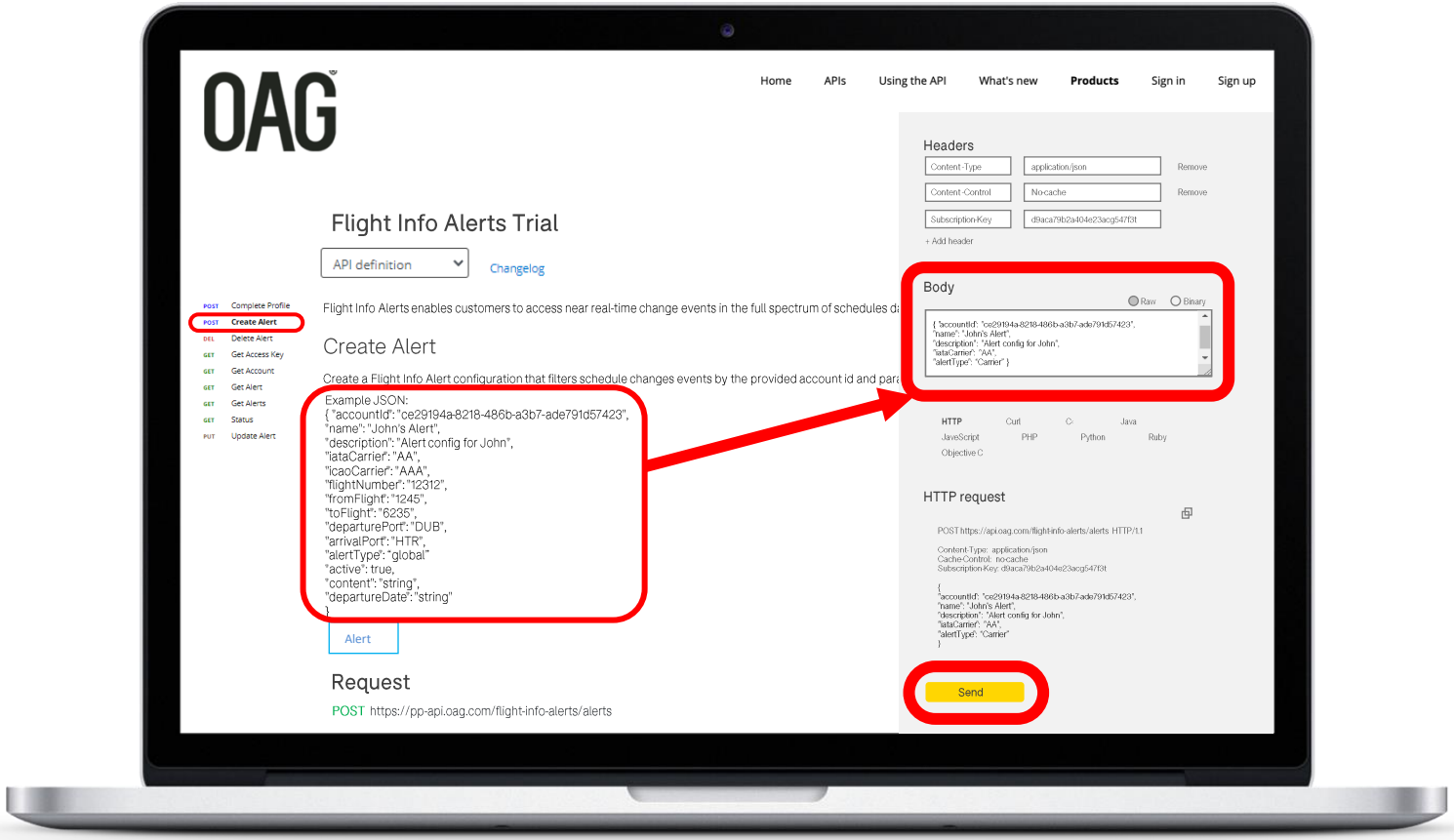
Carrier alert

Carrier Alert : Primarily change events by the provided carrier code. Further filtering of carrier-related changes can be set with additional parameters.

Note: When a flight alert is created, no checks are performed to determine if the alert represents an actual flight. As such, no alert will be received unless it matches an event.

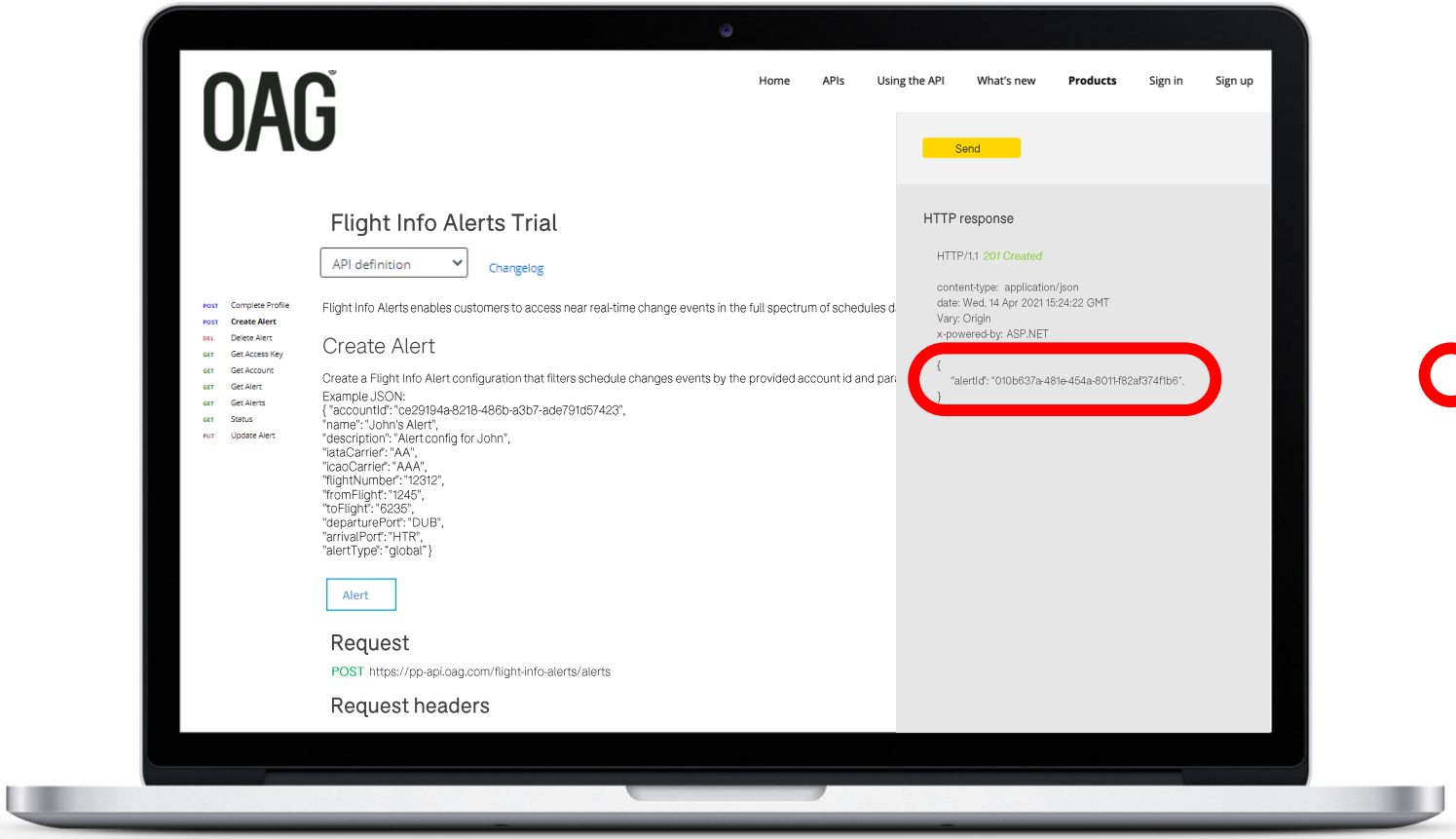
For more information on creating Alerts, please refer to this page.

Create Alert



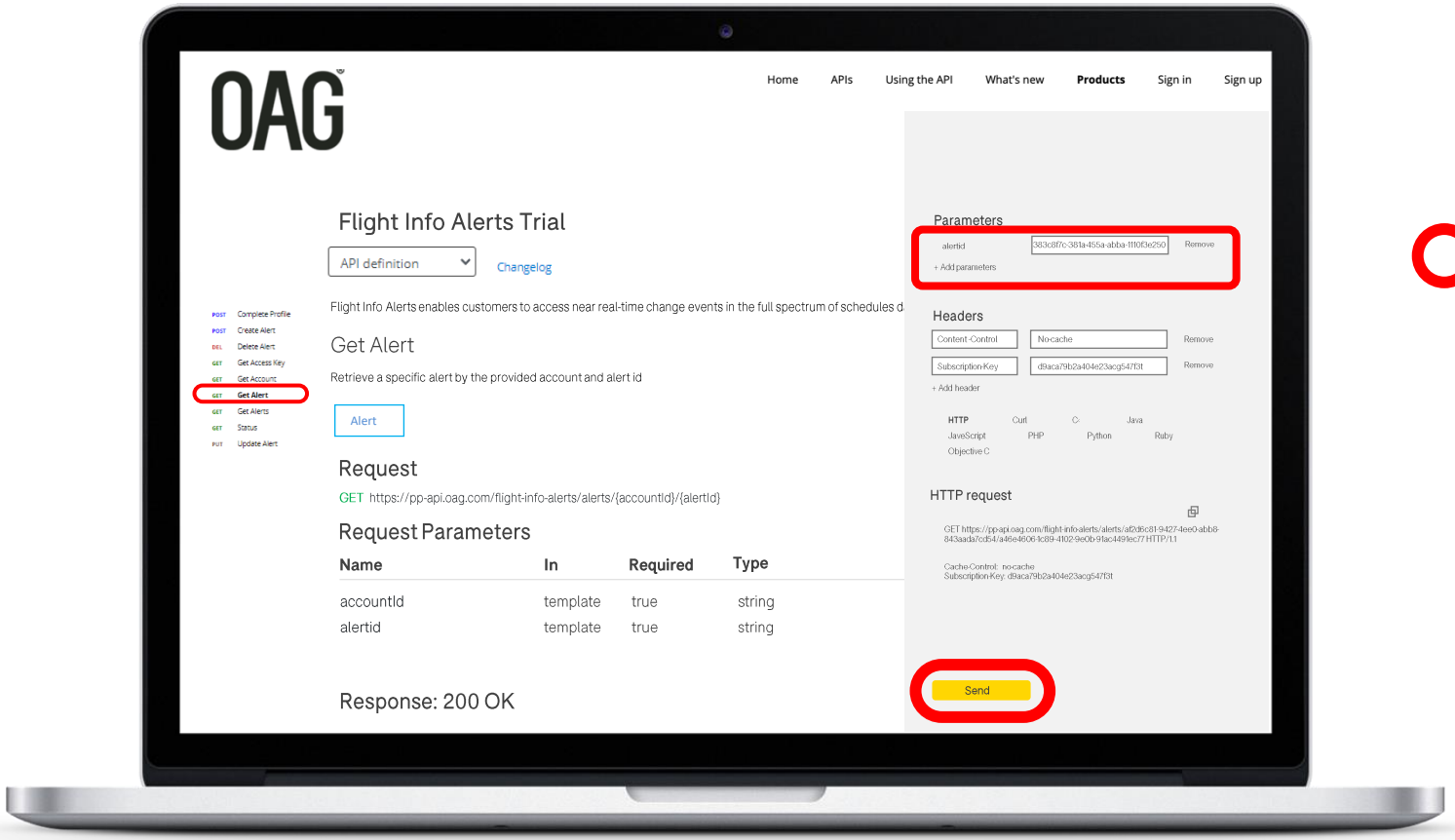
- Select the “Create Alert” endpoint. An example the JSON body including all parameters is provided. With it you can create any of the 3 alert types (i.e. Global, Carrier, Port) by removing unwanted parameters. select “Try it” and delete the parameters as relevant.

Create Alert



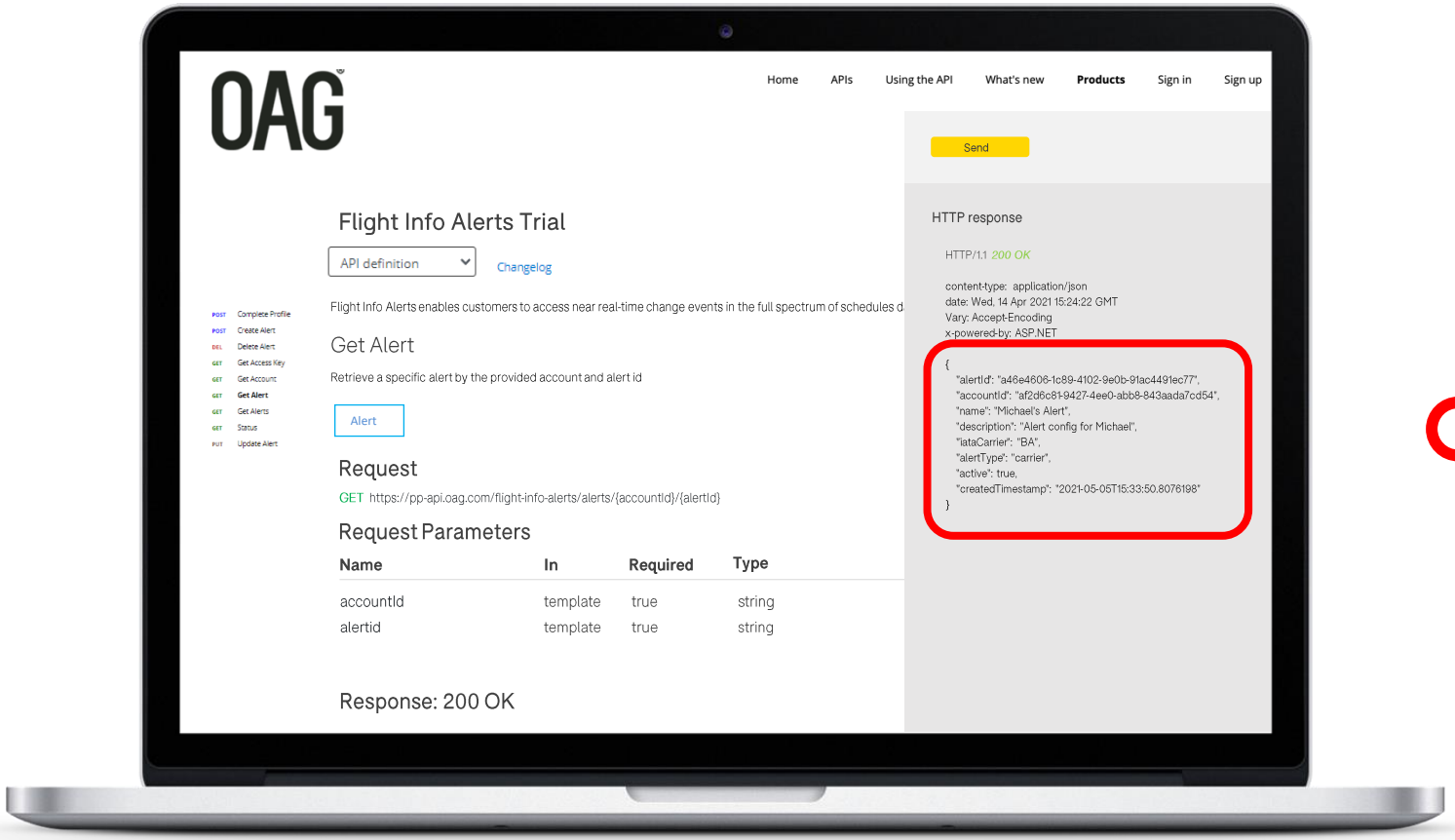
- You'll need your alert id to be able to retrieve the alert.

Get Alert



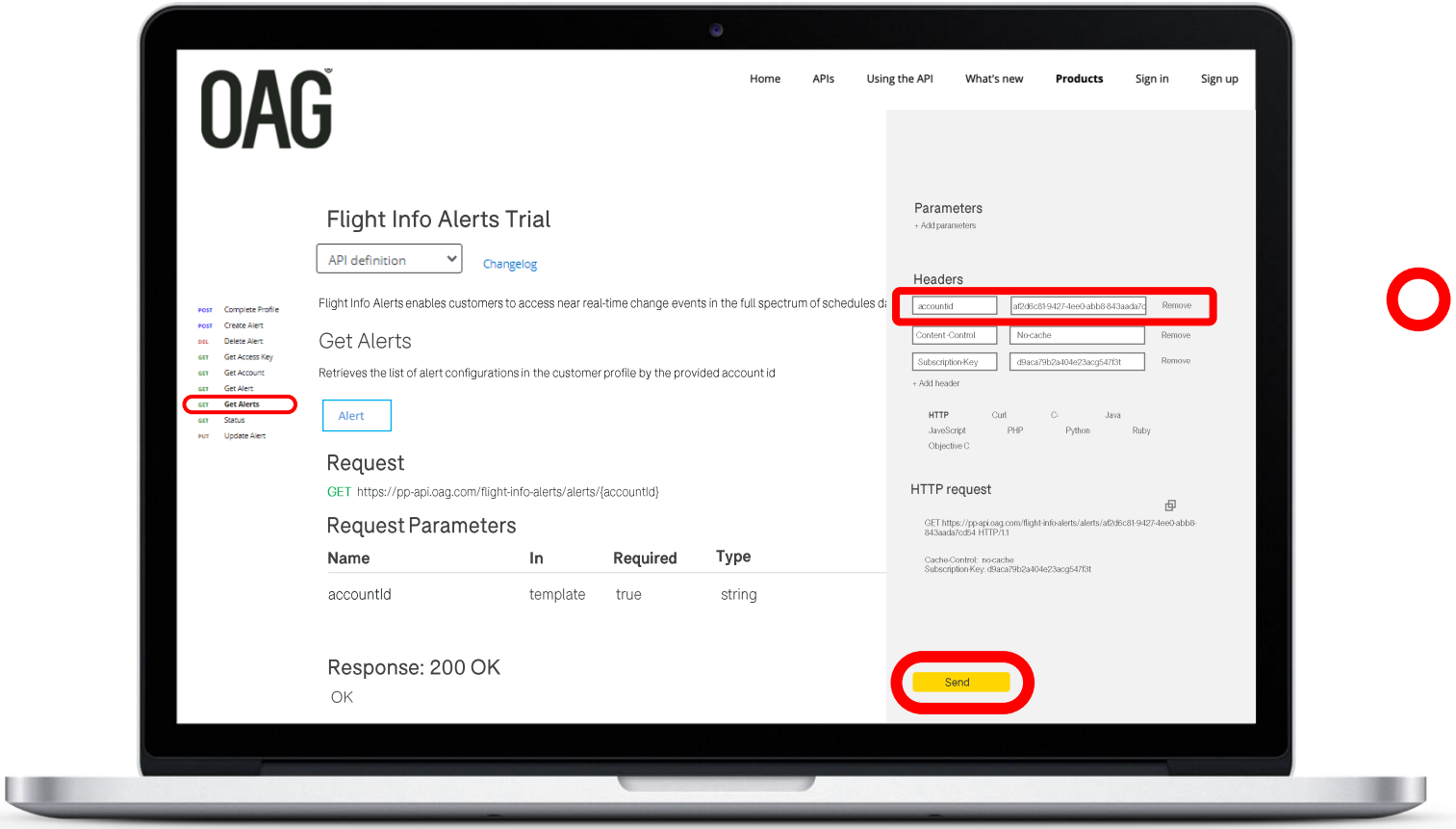
- Enter the alert id. Note you can only retrieve one alert at a time

Get Alert



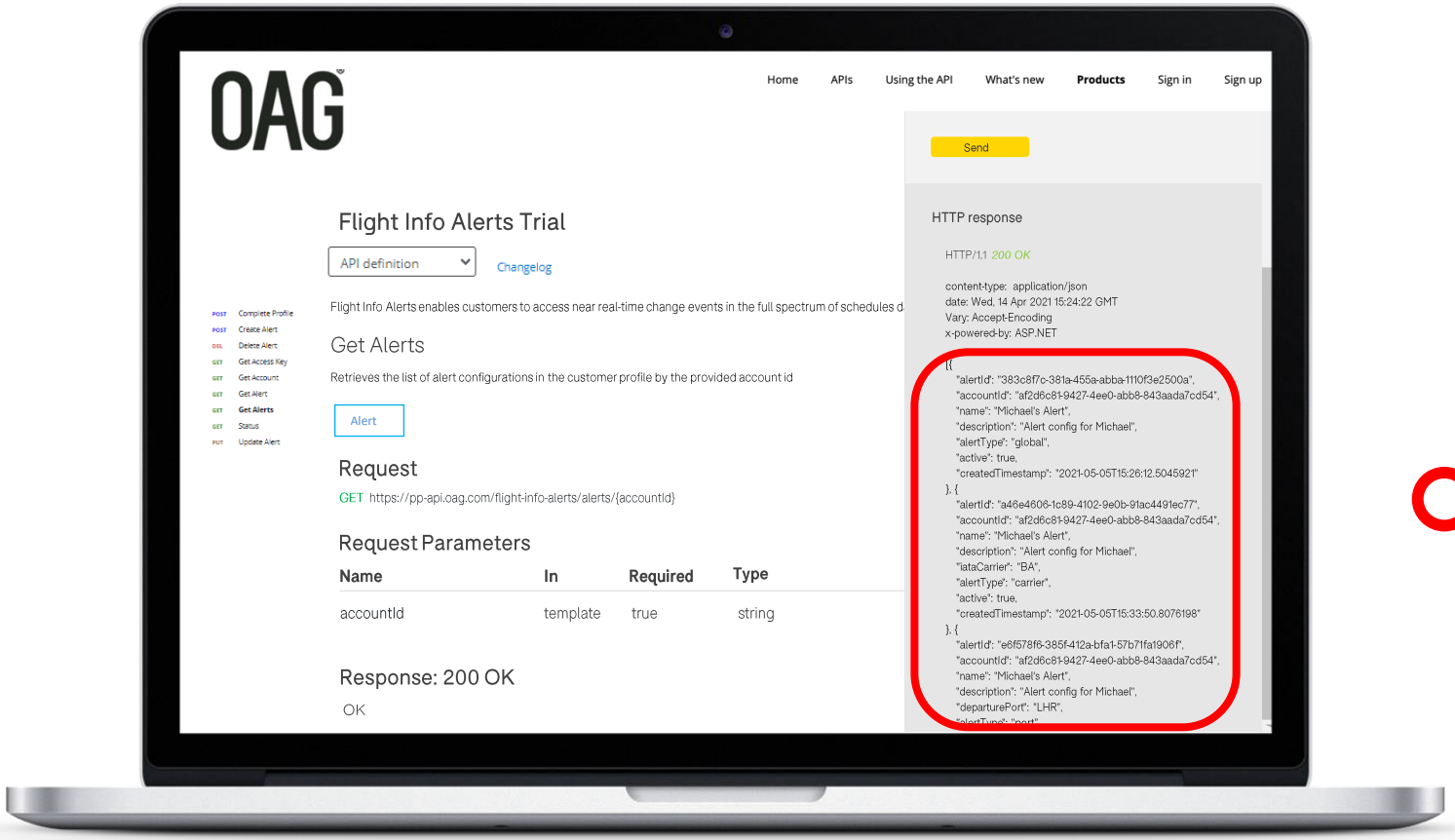
- The alert for the provided alert id is returned.

Get Alerts



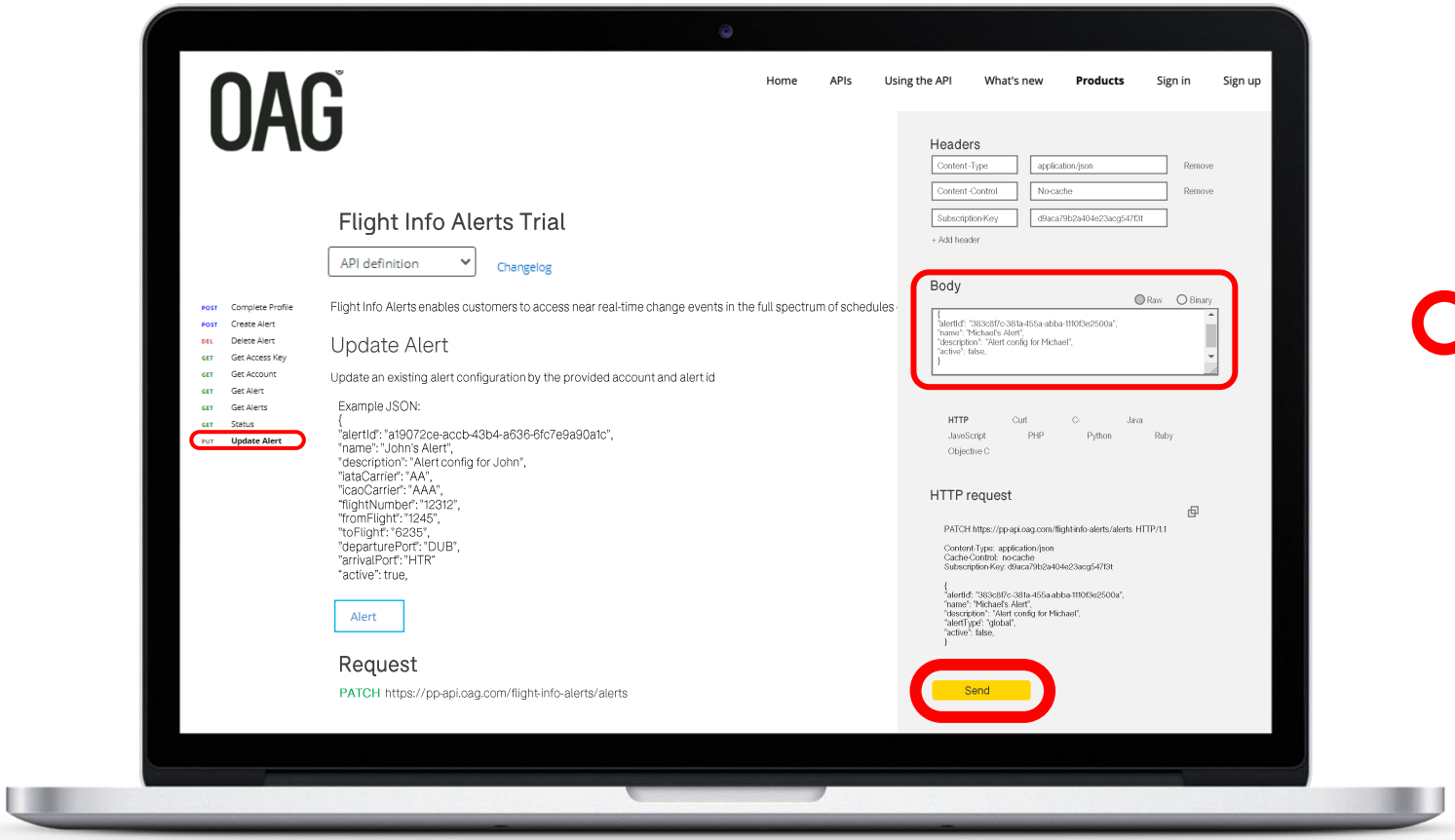
Enter your account id

Get Alerts



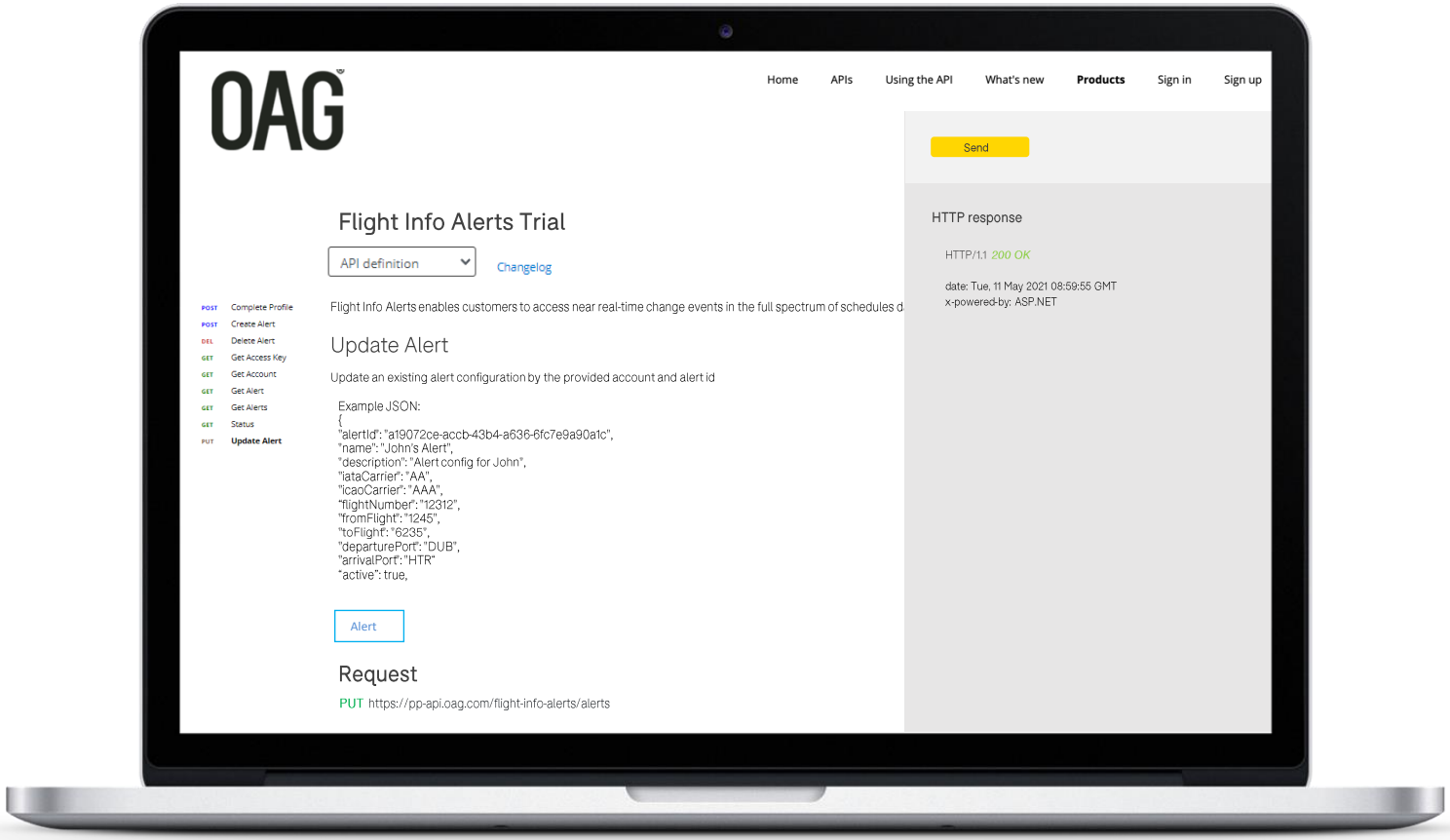
- All the alerts created for the provided account id are returned

Update Alert

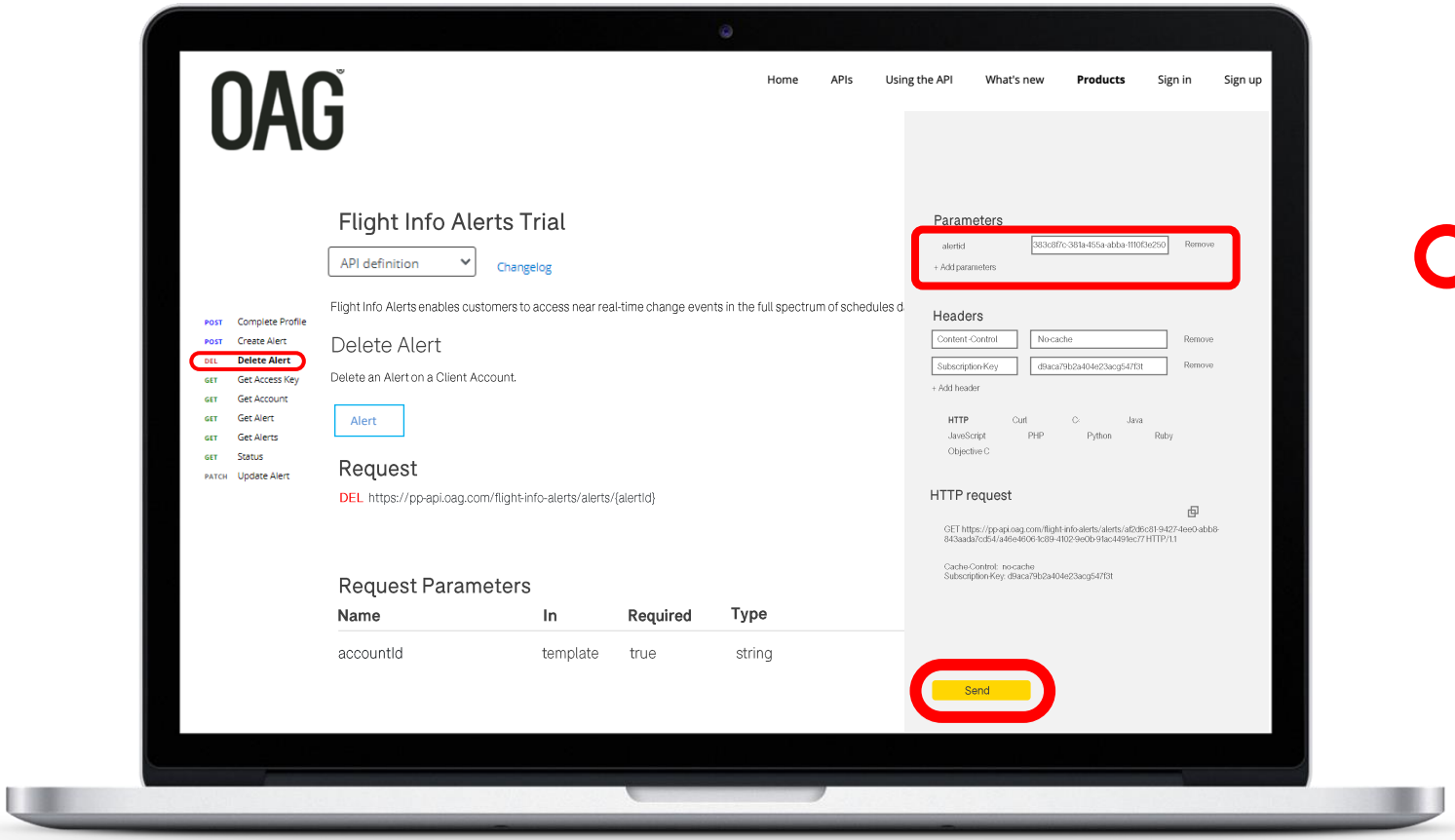


- Enter your alert body and update the parameters as necessary. Note that you cannot update the alert type.

Update Alert

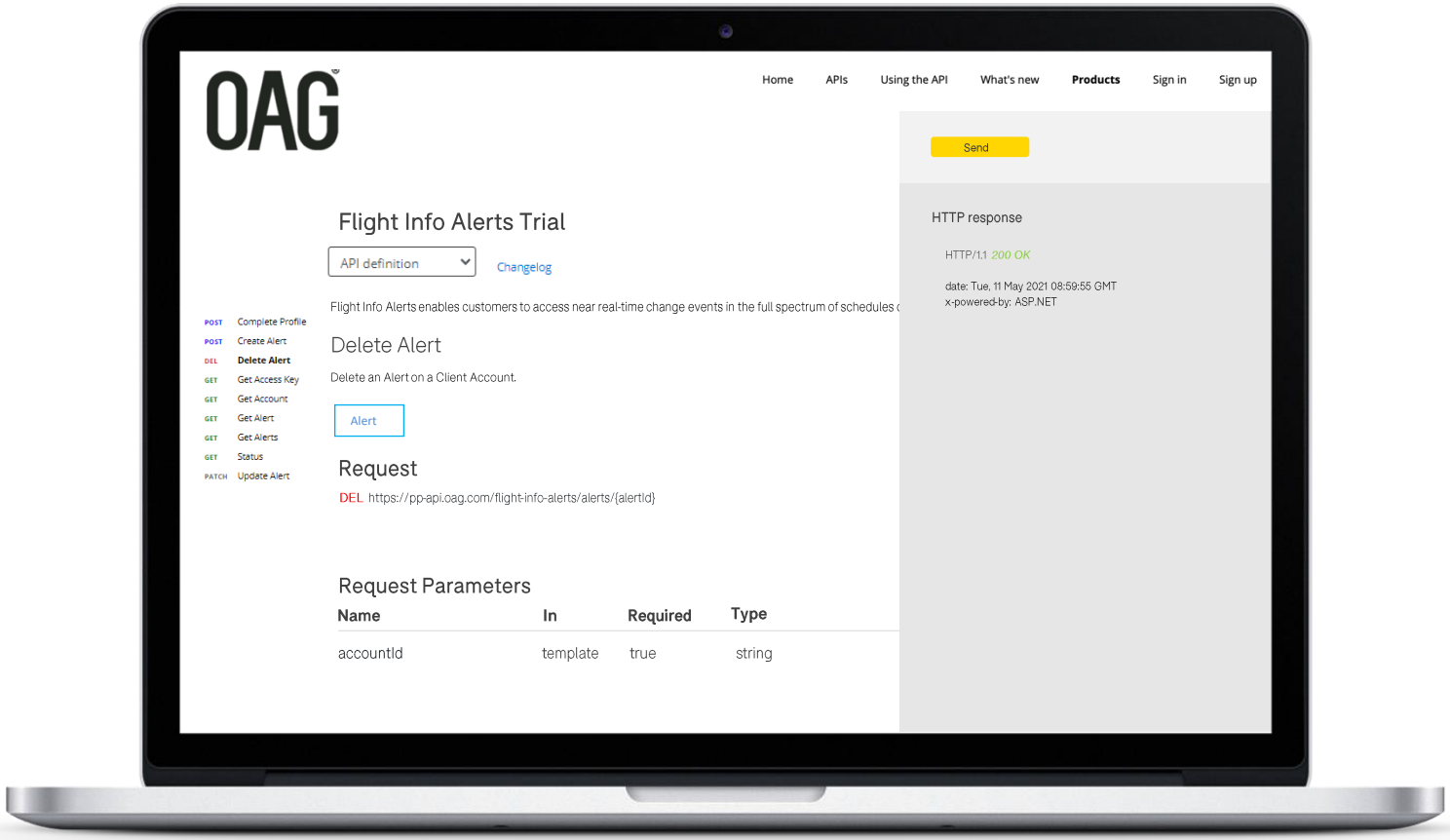


Delete Alert



Enter the alert id

Delete Alert



Connecting to the Event Hub

Accessing your Alerts Data

- Alert customers to the changes to schedules or status changes, and deliver the information to them without the need for them to look up on the API
- Now that you have configured your Alerts, OAG will begin streaming individual change Events to your Event Hub.
- These are accessible using the connection strings that you were provided with when you completed your profile (at the same time as receiving your Account ID).
- The next step is to retrieve the flight change events. On the next 2 pages you will find a table for language/platform support for MS Azure Event Hubs and links to example change Event JSON samples.

Language/Platform Support for Azure Event Hubs/OAG Alerts

Language	Azure		AWS		Google		On Prem	
	Function	App	Lambda	App	Function	App	Function	App
.Net (C#)	✓	✓	?	✓	?	✓	n/a	✓
Java	✓	✓	?	✓	?	✓	n/a	✓
Python	✓	✓	?	✓	?	✓	n/a	✓
JavaScript	✓	✓	?	✓	?	✓	n/a	✓
Go	✓	✓	?	✓	?	✓	n/a	✓
C/C++	✓	✓	?	✓	?	✓	n/a	✓
Azure Cli	✗	✓	✗	n/a	✗	n/a	n/a	✓
Azure Powershell	✗	✓	✗	n/a	✗	n/a	n/a	✓
Apache Kafka	✓	✓	✓	✓	?	✓	n/a	✓

Function: Cloud provided serverless function
App: Language created component that runs within an app service, container, VM or physical server, connecting to a hub via an SDK

Sample Event JSON

Please refer to the following pages to look at Sample events for both schedules and status data:

https://www.oag.com/en/knowledge/schedule-event-sample-flight-info-alerts

https://www.oag.com/en/knowledge/status-event-sample-flight-info-alerts

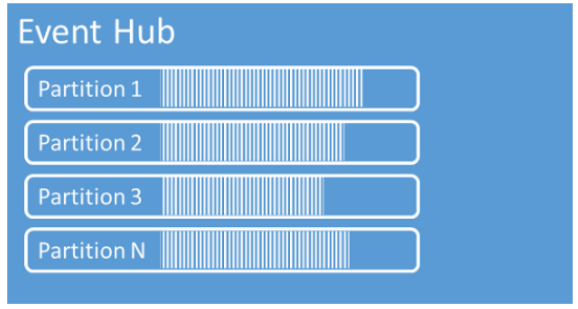
Key Things to Note

Event Hub

Event Hubs are a scalable event processing service that ingests and processes large volumes of events and data, with low latency and high reliability.

Event Hub Partitions

Customers Event hubs are 4 partitions by default. The customer's Event hub reader must be configured to read events from all partitions. As newer events arrive, they're added to the end of the sequence within the partition.

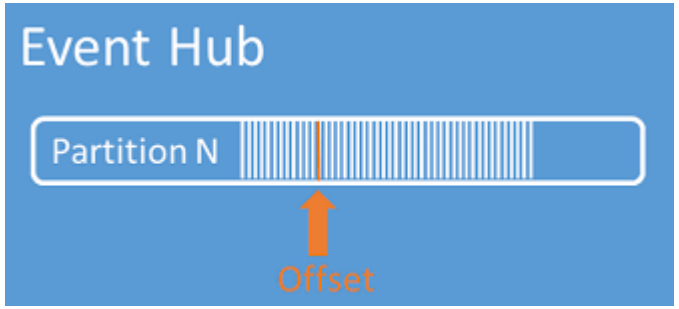


Partitions hold event data that contains body of the event. Each event is headed with metadata such as its offset in the partition, its number in the stream sequence, and service-side timestamp at which it was accepted.

Checkpointing

To keep track of your position in the event stream and know when the data stream is complete, you will need to store the position of the event sequence. Checkpointing is the responsibility of the consumer and occurs on a per-partition basis. If your reader disconnects and reconnects, you will be able to begin reading at the checkpoint stored from the last time you read from the event hub.

When your reader connects, it passes the offset to the event hub to specify a point in the event stream to start reading the events. The same mechanism can be used to provide failover resilience to return to older data by setting a lower offset using the checkpointing process, thereby replaying the event stream from a point before the most recent checkpoint.



We have provided some helpful resources on the following page.

Other Helpful Resources

OAG Knowledge Hub
https://www.oag.com/en/knowledge/flight-info-alerts

Microsoft Azure Event Hubs developer documentation
https://docs.microsoft.com/en-us/azure/event-hubs/

Understanding the consumer side of Azure Event Hubs
https://www.serverless360.com/blog/understanding-consumer-side-of-azure-event-hubs-checkpoint-initialoffset-eventprocessorhost
https://devblogs.microsoft.com/azure-sdk/eventhubs-clients/

Java specific examples (includes EventPosition) – available in all supported languages
https://github.com/Azure/azure-event-hubs/tree/master/samples/Java/Basic

To access events from a position in time, offset or sequence number - available in all supported languages
https://docs.microsoft.com/en-us/dotnet/api/microsoft.azure.eventhubs.eventposition?view=azure-dotnet
https://www.codota.com/code/java/classes/com.microsoft.azure.eventhubs.EventPosition

Checkpointing
https://docs.microsoft.com/en-us/java/api/overview/azure/messaging-eventhubs-checkpointstore-blob-readme?view=azure-java-stable
A custom checkpoint store can be created by overriding the Checkpointstore class.

Using AWS Lambda with self-managed Apache Kafka – use this to connect through to Azure Event Hub
https://docs.aws.amazon.com/lambda/latest/dg/kafka-smaa.html

Azure Event Hubs client library for Java - Version 5.7.0
https://docs.microsoft.com/en-us/java/api/overview/azure/messaging-eventhubs-readme?view=azure-java-stable