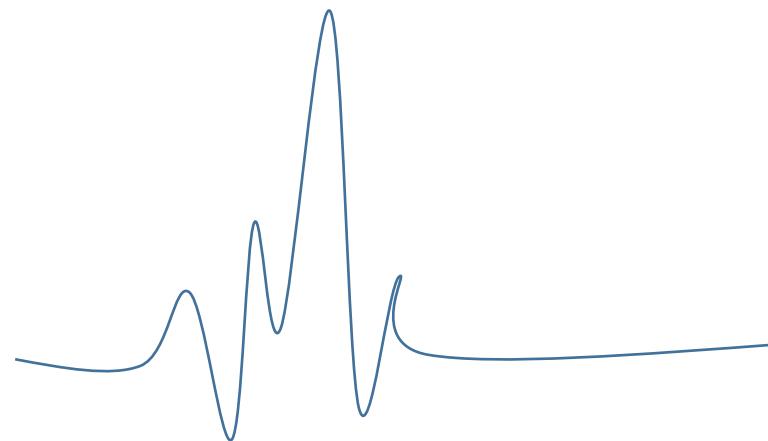


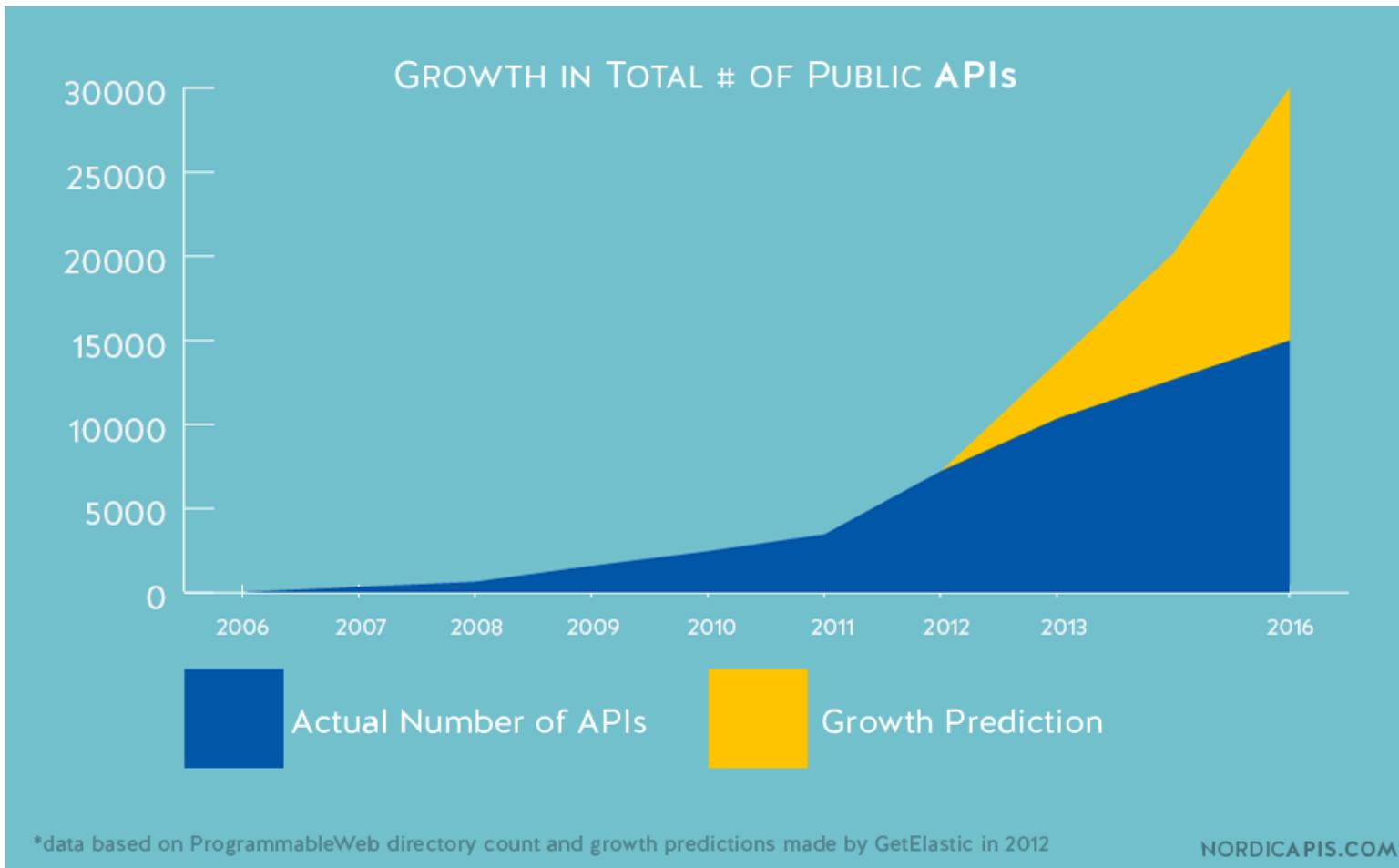
# Understanding and Executing on API Developer Experience

# What is an API?

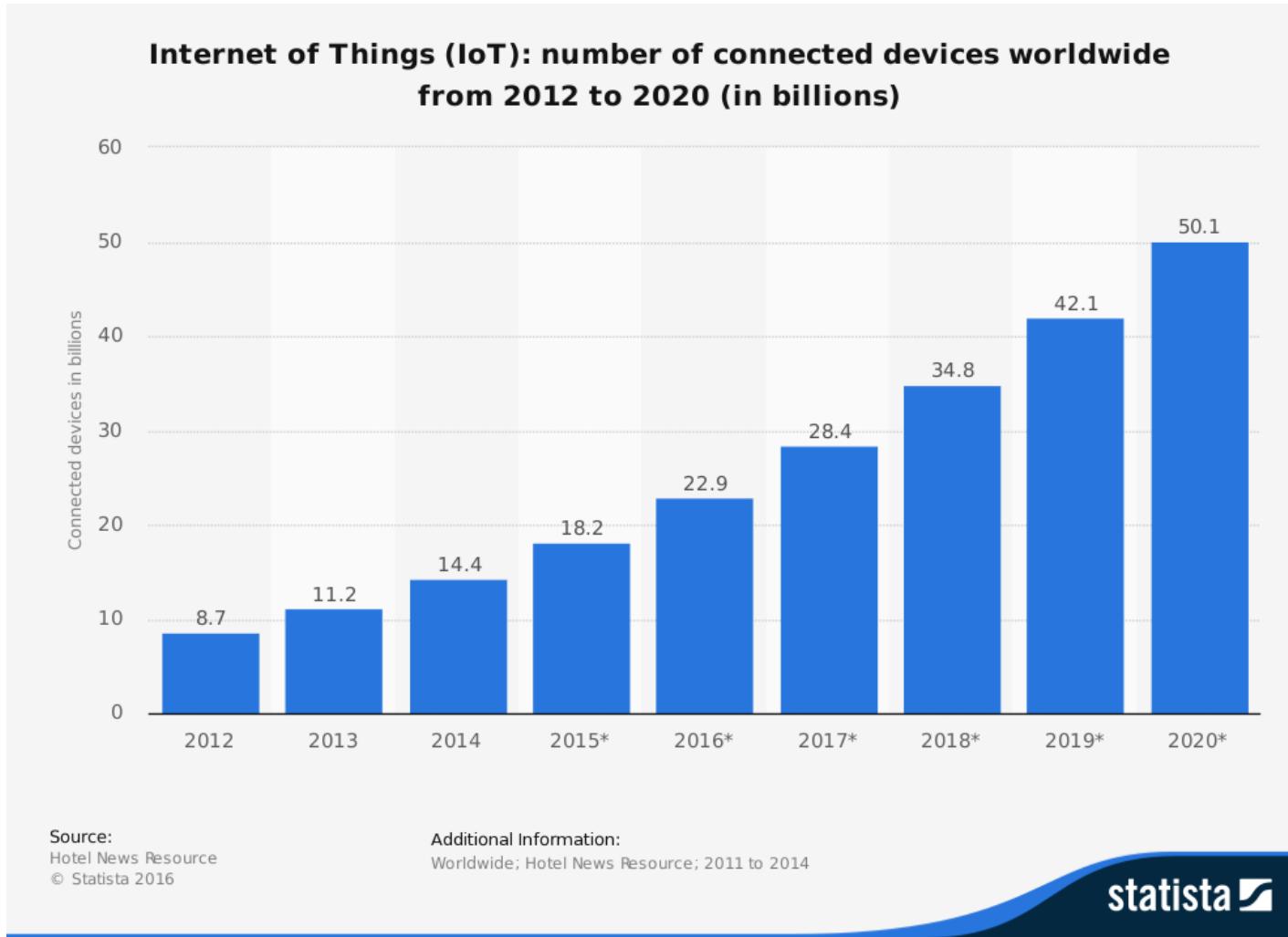
---



# APIs are Everywhere!



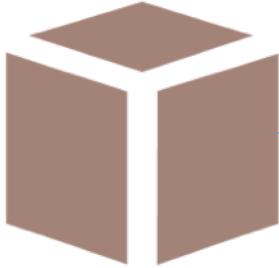
# Number of Connected Devices



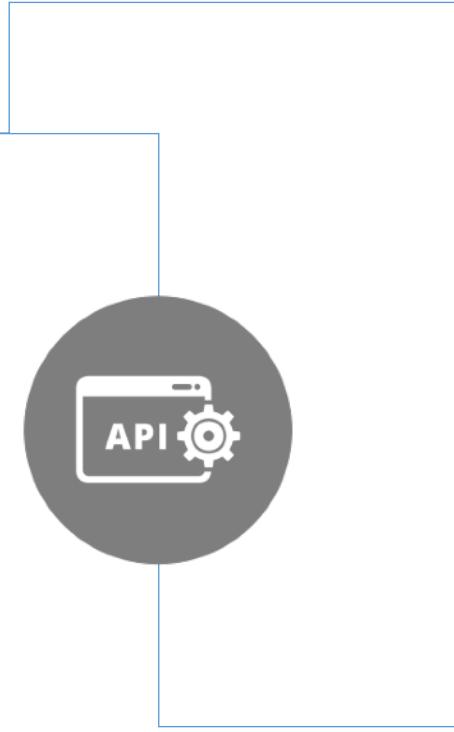
No. of devices/  
person  
**7 devices**

# The Multi Platform Ecosystem

---



A **platform** is a **product** that can be extended by users for the benefit of others

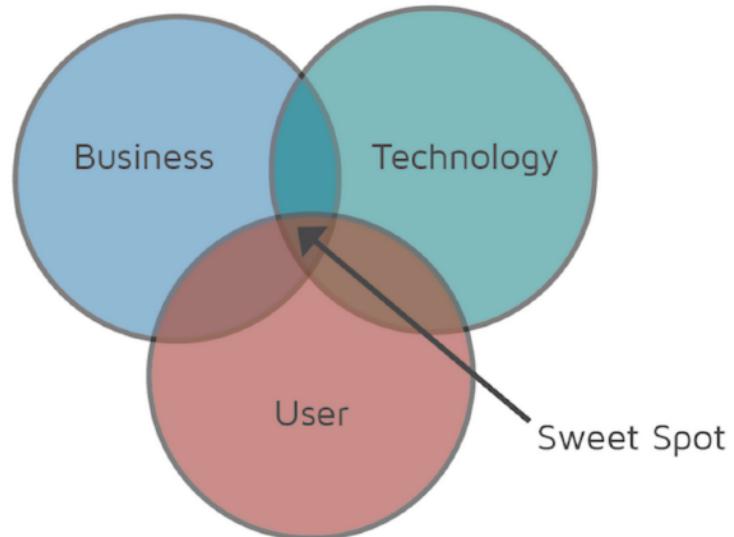


Average consumer

Third party developer

# Developers Are Humans Too

**DX is the aggregate of all the experiences a developer has when interacting with your Platform, usually through an API**



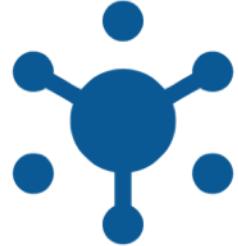
# Why DX Matters

We do better with products and platforms we enjoy using



# Why DX Matters

---



Growth and Adoption: The value of an API increases when users reach critical mass



Prevents Switch: Makes APIs differentiable and competitive



Helps drive business and technological goals

# Speaking of Business and Technological Goals

---



## Public APIs

Priority: High Adoption



## Private APIs

Priority: Low Cost



## Partner APIs

Priority: High Adoption, Low Cost

# How To Think About DX

# Goals for Good DX

---



Solve a Consumer Need



Help the Business



Built for sustainability

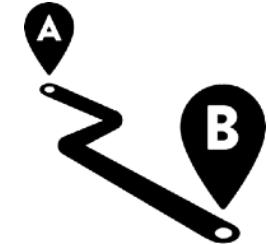
# Three Step Process for Good DX

---



## Step 1

Understand the API's audience



## Step 2

Understand the audience journey

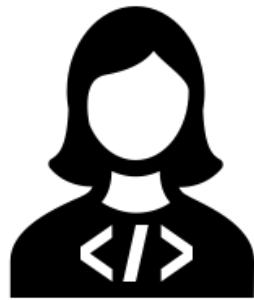
## Step 3

Map the audience to the journey

# Step 1: Understand the API's Audience

---

## API Users



Newcomer -  
First Time  
User



Debugger –  
addressing a  
specific issue

## API Decision Makers



Evaluator-  
Deciding whether  
to use the API



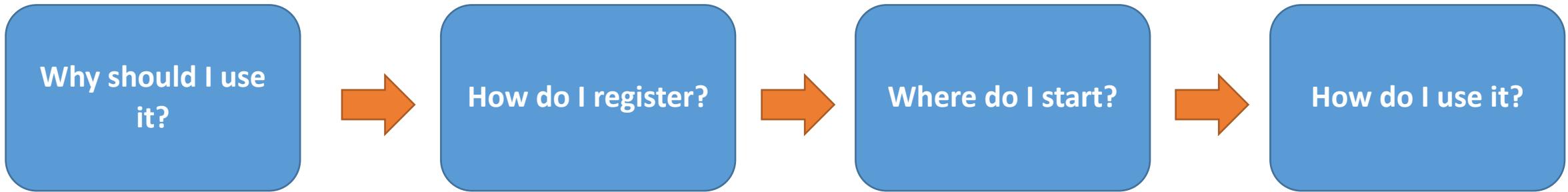
Problem Solver –  
Is X possible with  
this API

Eg: CTO

Eg: PM

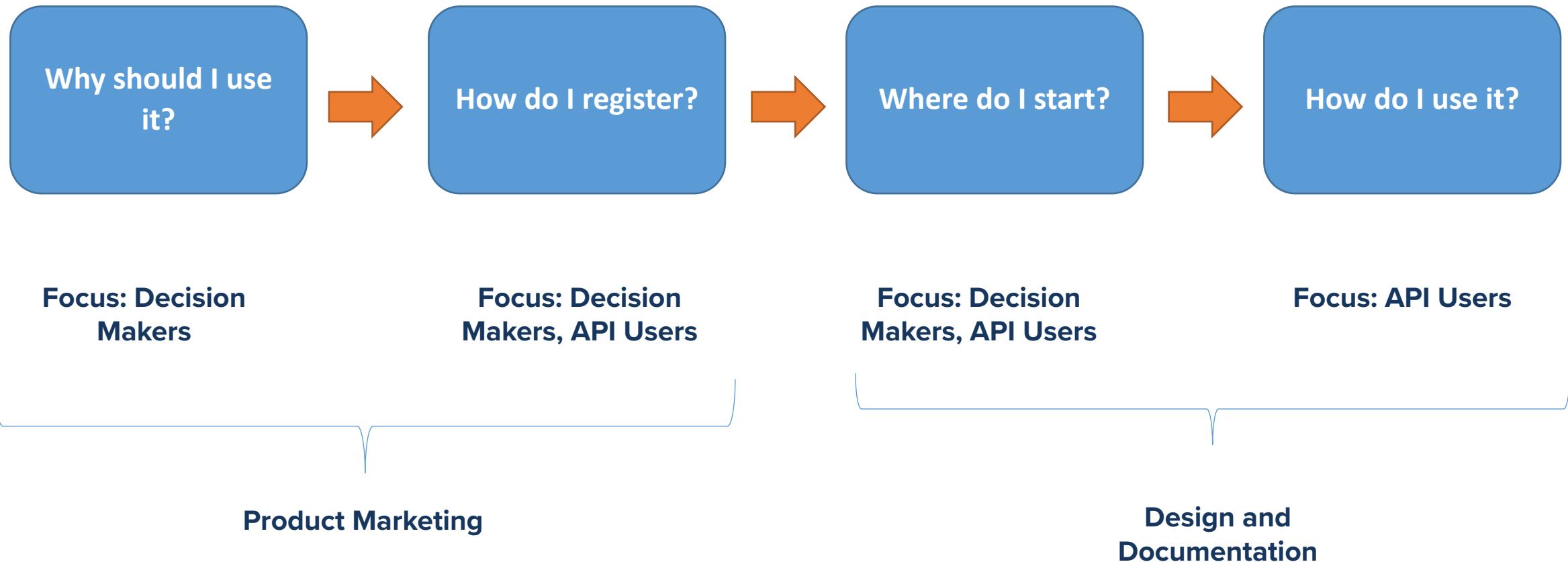
## **Step 2: Understand their Journey**

---



## Step 3: Map Journey to the Right Audience

---



# Recommendations to Optimize Each Phases

# Follow the 3:30:3 Rule

---

## 3:30:3 Rule

- 3 seconds to understand API's purpose
- 30 seconds to identify entry point into system
- 3 minutes to use the API's result

## 3.a. “Why Should I Use It”?

Why should I use  
it?



Focus: Decision  
Makers

### Recommendations



Position based on  
Industry



Concise messaging of  
core value

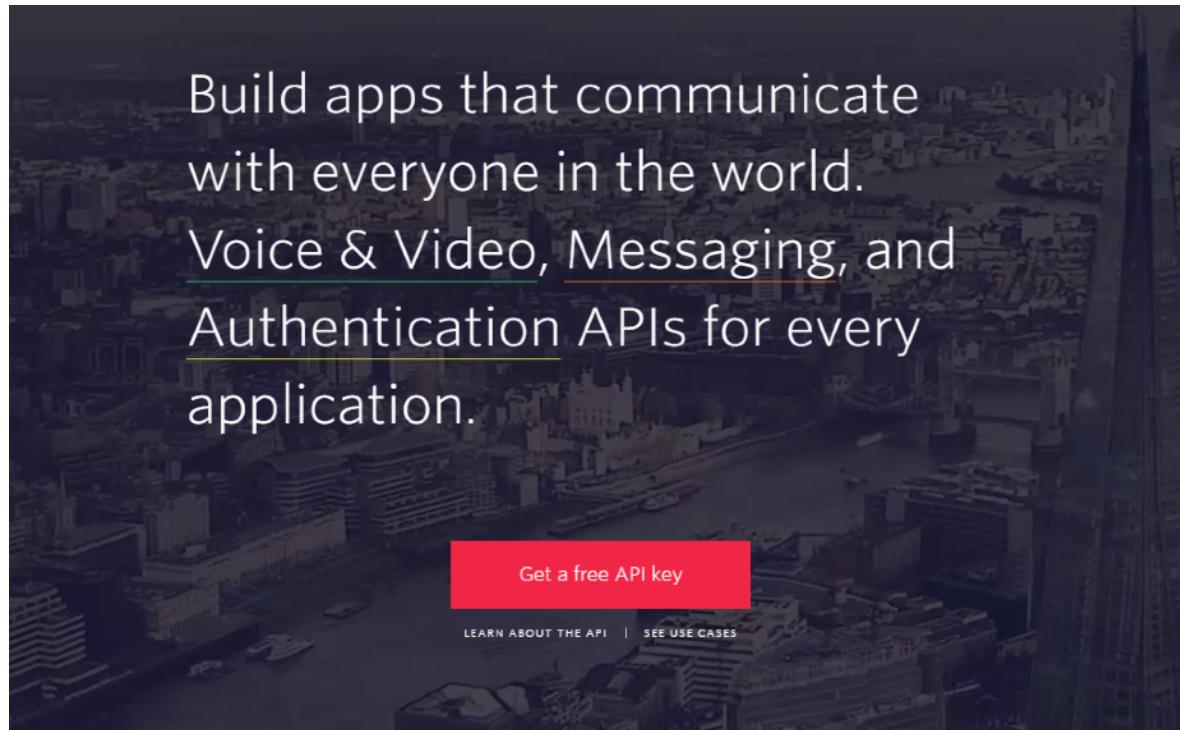


Understand value in 3  
seconds

# Examples

## Twilio

Know exactly what the value is, and what it's mean to be used for



The image is a screenshot of the Twilio website. At the top, it says "TWILIO IS A CLOUD COMMUNICATIONS PLATFORM". Below this are three main sections: "PROGRAMMABLE COMMUNICATIONS" (with a red icon), "SUPER NETWORK" (with a red icon), and "BUSINESS MODEL FOR INNOVATORS" (with a red icon). To the right of each section is a brief description. Further down the page are three main categories: "VOICE & VIDEO", "MESSAGING", and "AUTHENTICATION". Each category has a description and a "EXPLORE" button. At the bottom, there is a section titled "EXPLORE WAYS TO MAKE YOUR COMMUNICATIONS GREAT" with several smaller cards, each featuring an icon and a brief description: "TWO-FACTOR AUTHENTICATION", "APPOINTMENT REMINDERS", "MASKED PHONE NUMBERS", "CALL TRACKING", "INSTANT LEAD ALERTS", and "CLICK-TO-CALL". At the very bottom right, there is a call-to-action button "FIND THE USE CASE FOR YOUR NEED".

# Examples

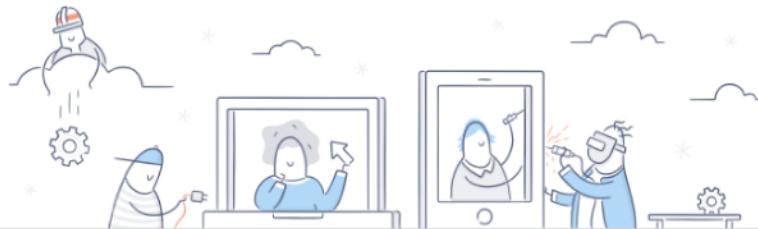
---

## Dropbox

Pleasant to read, easy to understand, with example use cases, all in the first section of the landing page

Build your app on the Dropbox platform

A powerful API for apps that work with files.



### Read our docs

Docs are organized by language, from .NET to Swift.

### Create your app

Getting started is simple and quick from the App Console.

### Test your ideas

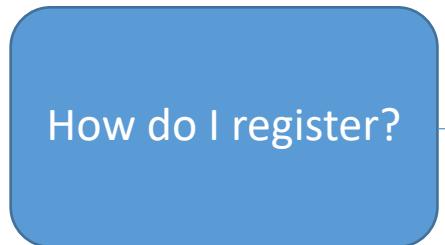
It's easy to prototype and test examples with our API Explorer.

Learn from our examples

Photo Watch uses our Swift SDK to let users see their Dropbox photos on Apple Watch.

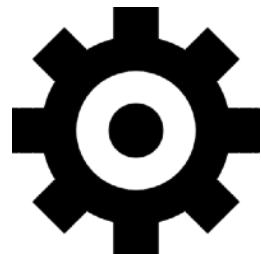


### 3.b. “How Should I Register?”

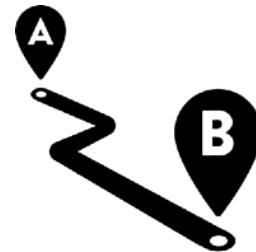


Focus: Decision Makers, API  
Users

#### Recommendations



Explain setup in  
comprehensive fashion



Minimal steps to  
register – 30 seconds



Provide API console

# Examples

## Make Sign up straightforward

Dropbox asks you for your name, email and password for the free API key. It also allows Google sign in. Straightforward process with Mailchimp as well.



Create an account [or log in](#)

First name

Last name

Email

Password

I agree to [Dropbox terms](#). [Create an account](#)

or

[Sign up with Google](#)



### Get started with a free account

Create a free MailChimp account to send beautiful emails to customers, contributors, and fans. Already have a MailChimp account? [Log in here](#)

Email

Username

Password  [Show](#)

One lowercase character  One uppercase character  
 One number  One special character  
 8 characters minimum

[Get Started!](#)

By clicking this button, you agree to MailChimp's  
[Anti-spam Policy & Terms of Use](#).

# Bad Examples

---

## Jumping through hurdles - SoundCloud

SoundCloud asks users to fill up a 3 page Google form, with details on the app idea, before awarding the API key. Waiting period is typically 2 weeks

SoundCloud Application Registration

\* Required

About you

Please provide us with information about you.

SoundCloud Profile URL \*

This profile must already exist. This profile will own the app. Example:  
[https://soundcloud.com/your\\_permalink\\_here](https://soundcloud.com/your_permalink_here)

Your answer

Company (if applicable)

Your answer

Developer Contact Name \*

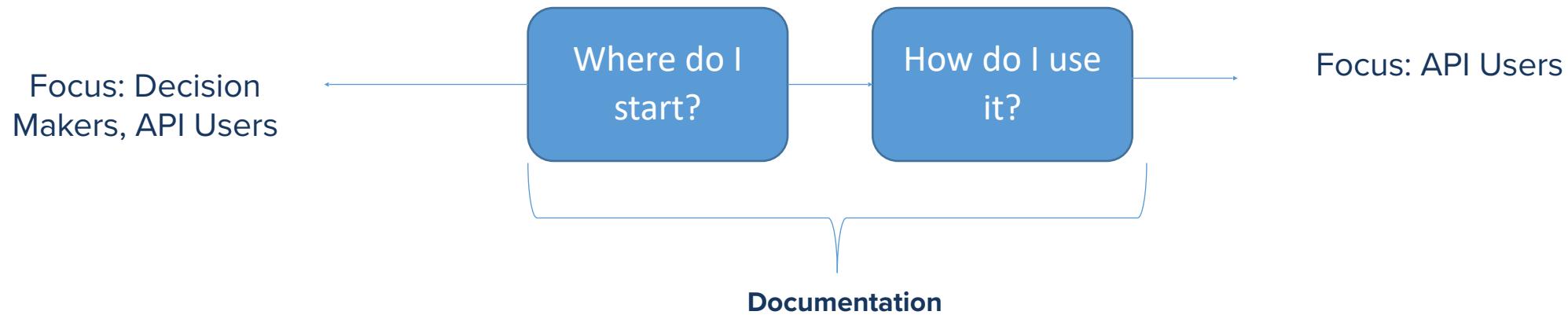
Your answer

Contact Email Address \*

Your email address \*must\* be associated with the SoundCloud profile that will own the app.

### 3.c. “Where and How Do I Use it?”

---



# API Documentation and the OpenAPI Specification

# What is API Documentation

---

- Documentation is a deliverable of technical content, containing instructions on how to effectively use your API → usage manual
- Good documentation can be a determining factor in API adoption and growth

An API is only as good as its documentation

# Examples

<a href="#">Overview</a>
» <a href="#">Activities</a>
» <a href="#">Captions</a>
» <a href="#">ChannelBanners</a>
» <a href="#">Channels</a>
» <a href="#">ChannelSections</a>
» <a href="#">Comments</a>
» <a href="#">CommentThreads</a>
» <a href="#">GuideCategories</a>
» <a href="#">i18nLanguages</a>
» <a href="#">i18nRegions</a>
» <a href="#">PlaylistItems</a>
» <a href="#">Playlists</a>
» <a href="#">Search</a>
» <a href="#">Subscriptions</a>
» <a href="#"> Thumbnails</a>
» <a href="#">VideoAbuseReportReasons</a>
» <a href="#">VideoCategories</a>
» <a href="#">Videos</a>
» <a href="#">Watermarks</a>
» <a href="#">Standard Query Parameters</a>
» <a href="#">YouTube Data API Errors</a>

## API Reference

The YouTube Data API lets you incorporate functionality from the YouTube platform into your own website or application. The lists below identify the different resources available through the API, and each resource supports methods to insert, update, or delete many different types of data.

This reference guide explains how to use the API to interact with YouTube's data. A resource represents a type of item that can be created, updated, or deleted. For each resource type, the guide lists the fields that make up the resource, along with its associated JSON objects. The guide also lists one or more sample requests and explains how to use those methods in your application.

### Calling the API

The following requirements apply to YouTube Data API requests:

1. Every request must either specify an API key or use OAuth 2.0 authentication. An API key is available in the [Developer Console's API Access page](#).
2. You **must** send an authorization token for every request that retrieves data. You must include an OAuth 2.0 authorization token for any request that retrieves data from a user's account.
3. The API supports the OAuth 2.0 authentication protocol.

In addition, some API methods for retrieving data from a user's account may return additional metadata when requests are made. For example, when requesting a user's uploaded videos, the API may also contain private videos if the user has granted permission to do so.

# Tumblr API

Welcome to the Tumblr API! There isn't anything we enjoy more than seeing talented designers and engineers using Tumblr to invent whole new forms of creative expression. We've put a tremendous amount of care into making this API functional and flexible enough for any projects you throw at it. Join us in our [discussion group](#) to talk about how to use it, what could be better, and all the amazing things you're going to build with it. Follow our [Developers Blog](#) for important news and announcements. Please use the API responsibly, and [send us your feedback](#). Enjoy!

Tumblr also supports an API that delivers content according to the [oEmbed standard](#). Our oEmbed API endpoint is <https://www.tumblr.com/oembed>, which supports post URLs in the format [https://\\*.tumblr.com/post/\\*](https://*.tumblr.com/post/*).

If you're looking for documentation for the old API, you can find it [here](#).

## What You Need

Get an OAuth key: [register an application](#)

You'll need this to get your API key, even if you don't ever need to use a fully signed OAuth request.

For more details, see [Authentication](#) below.

## Contents

<a href="#">What You Need</a>
<a href="#">Overview</a>
<a href="#">Console</a>
<a href="#">Clients</a>
<a href="#">Authentication</a>
<a href="#">Blog Methods</a>
» <a href="#">Blog Info</a>
» <a href="#">Avatar</a>
» <a href="#">Likes</a>
» <a href="#">Following</a>
» <a href="#">Followers</a>
<a href="#">Posts</a>
» <a href="#">Text</a>
» <a href="#">Photo</a>
» <a href="#">Quote</a>
» <a href="#">Link</a>
» <a href="#">Chat</a>
» <a href="#">Audio</a>
» <a href="#">Video</a>
» <a href="#">Answer</a>
<a href="#">Queue</a>
<a href="#">Drafts</a>
<a href="#">Submissions</a>
<a href="#">Posting</a>

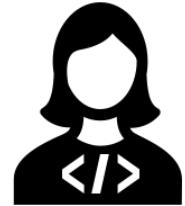
# Traditional API Development and Documentation



Architect



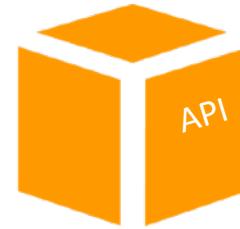
Technical  
Writer



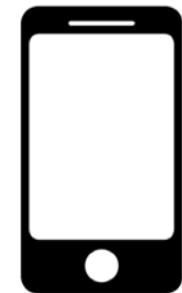
Developer



Tester



Request  
Response



# Major Problems

---



Maintaining documentation, tests and implementation is hard



Implementation comes at a cost of user experience



Communication between services built in different programming languages is not easy

# RESTful Interface



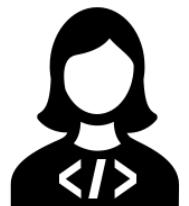
Architect



Technical  
Writer



Keeps in  
Sync

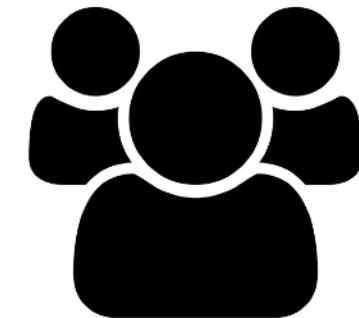
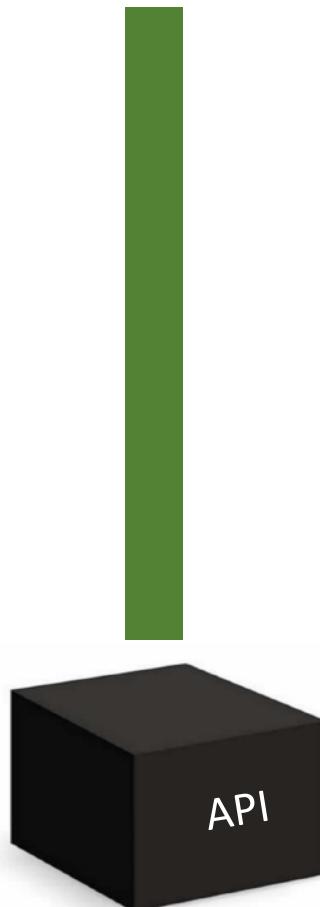


Developer



Tester

Restful  
Interface



# OpenAPI Specification is the Answer

---

Allow machines/tools to integrate  
with API



Allow humans to implement API code



Allow humans to read and generate API  
documentation and test cases



# The OpenAPI Spec is the API Contract

---

- The OAS is an API description format to Design and Document REST APIs
- The OAS defines your API's contract
- Connects computers, technology stacks and humans in one unified language

# Evolution of the OAS

---

- 2010 – Wordnik created and open sourced Swagger Specification
- 2010 to 2015 – Massive adoption of the Swagger Specification, with Swagger tooling developed by community
- 2015 - Swagger Spec and core tooling acquired by **SmartBear**
- 2015 – **Open API Initiative** was created to standardize REST design. Swagger Spec was renamed as **OAS**



# Example of an OAS Defined API

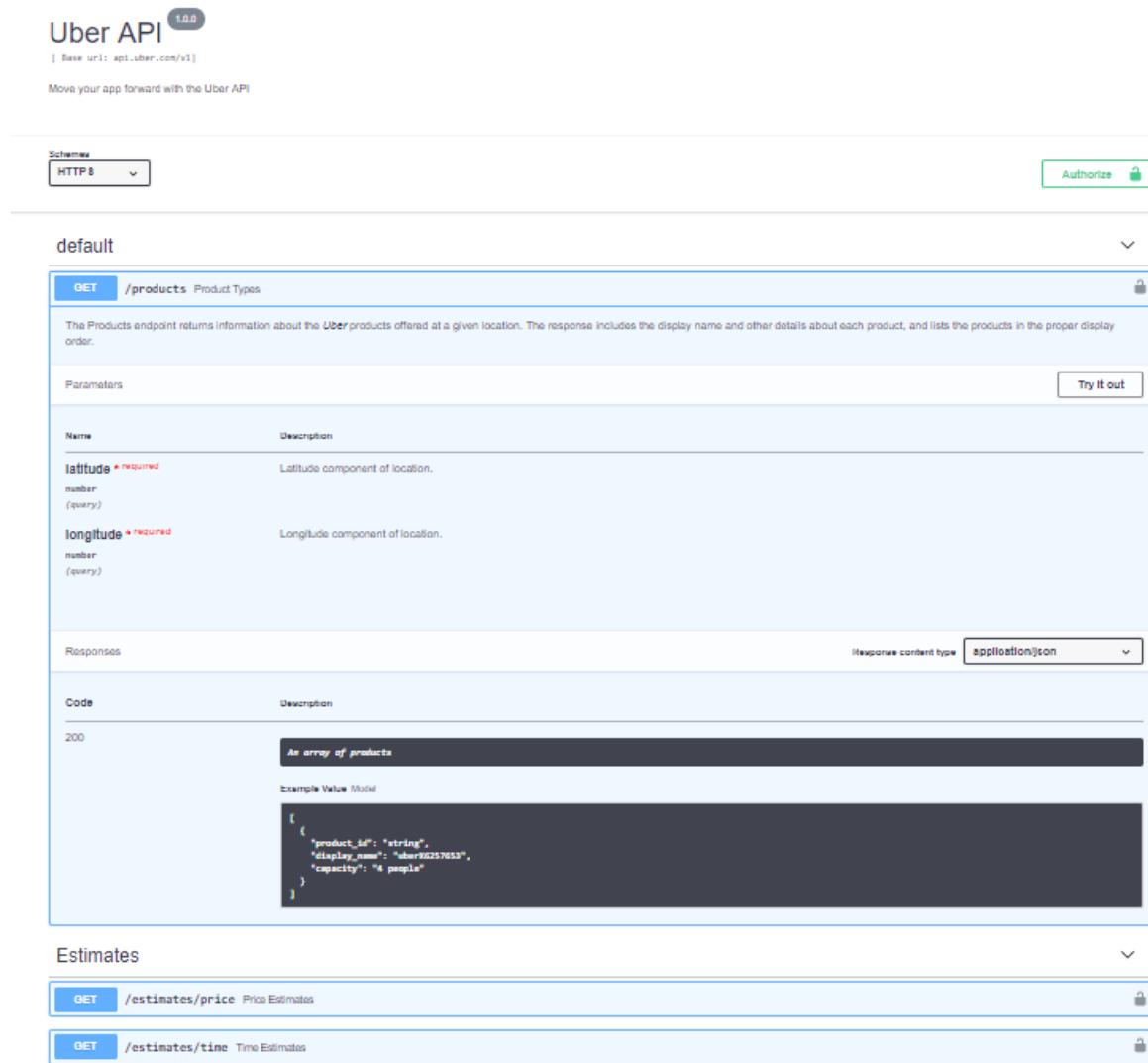
## API Overview

```
swagger: '2.0'
info:
  title: Uber API
  description: Move your app forward with the Uber API
  version: "1.0.0"
# the domain of the service
host: api.uber.com
# array of all schemes that your API supports
schemes:
  - https
# will be prefixed to all paths
basePath: /v1
produces:
  - application/json
paths:
  /products:
    get:
      summary: Product Types
      description: |
        The Products endpoint returns information about the *Uber* products offered at a given location. The response includes the display name and other details about each product, and lists the products in the proper display order.
      parameters:
        - name: latitude
          in: query
          description: Latitude component of location.
          required: true
          type: number
          format: double
        - name: longitude
          in: query
          description: Longitude component of location.
          required: true
          type: number
          format: double
      responses:
        200:
          description: An array of products
          schema:
            type: array
            items:
              type: object
              properties:
                product_id:
                  type: string
                  description: Unique identifier representing a specific product for a given latitude & longitude. For example, uberX in San Francisco will have a different product_id than uberX in Los Angeles.
                description:
                  type: string
                  description: Description of product.
                display_name:
                  type: string
                  description: Display name of product.
                capacity:
                  type: string
                  description: Capacity of product. For example, 4 people.
                image:
                  type: string
                  description: Image URL representing the product.
```

## Requests

## Responses

# Example of Documentation



The screenshot shows the Swagger interface for the Uber API. At the top, it says "Uber API 1.0.0" and "Save url: api.uber.com/v1". Below that is a sub-header "Move your app forward with the Uber API". A dropdown menu "Schemas" is set to "HTTP 1.1". A green "Authorize" button with a lock icon is on the right. The main content area has a title "default". Under "GET /products Product Types", there's a description: "The Products endpoint returns information about the Uber products offered at a given location. The response includes the display name and other details about each product, and lists the products in the proper display order." A "Try It Out" button is present. The "Parameters" section lists "latitude" (required, number, query) and "longitude" (required, number, query). The "Responses" section shows a 200 response with a description "An array of products" and a "Response content type" of "application/json". An example value model is shown as a JSON object:

```
{  
  "product_id": "string",  
  "display_name": "uberX257653",  
  "capacity": "4 people"  
}
```

Below this, under "Estimates", are two sections: "GET /estimates/price Price Estimates" and "GET /estimates/time Time Estimates", both with lock icons.



Hosted on  
SwaggerHub

# Creating Effective Documentation

# Effective Documentation Tips

---

- Documentation plays a central role in the way APIs are perceived by a User
- Tip 1: List the Fundamentals
- Tip 2: Write for Humans
- Tip 3: Explain your Request-Response Cycles
- Tip 4: Empower with Experimentation

# Tip 1: List the Fundamentals

---

## Fundamental sections in every documentation



Authentication



Errors



End Points



Examples



Terms of Use



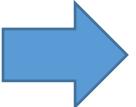
Changelog

# Authentication in OAS

Authorize 

- Basic Auth

```
securityDefinitions:  
  basicAuth:  
    type: basic
```



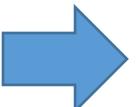
Available authorizations

Basic authorization ↪

Username:

Password:

- OAuth 2
  - Implicit
  - Password
  - Access Code



Available authorizations

Scopes are used to grant an application different levels of access to data on behalf of the end user. Each API may declare one or more scopes.

API requires the following scopes. Select which ones you want to grant to Swagger UI.

OAuth2.0

Authorization URL: <http://example.com/oauth/auth>

Flow: `implicit`

client\_id:

Scopes:

write  
*allows modifying resources*

read  
*allows reading resources*

# End Points in OAS

---

```
paths:
```

```
  /products: 
```

```
  /estimates/price: 
```

```
  /estimates/time: 
```

```
  /me: 
```

```
  /history: 
```

# Errors and Examples in OAS

```
responses:
  200:
    description: An array of products
    schema:
      type: array
      items:
        type: object
        properties:
          product_id:
            type: string
            description: Unique identifier representing a
                         specific product for a given latitude & longitude.
                         For example, uberX in San Francisco will have a
                         different product_id than uberX in Los Angeles.

          display_name:
            type: string
            description: Display name of product.
            example: uberX6257653

          capacity:
            type: string
            description: Capacity of product. For example, 4
                         people.
            example: 4 people
```

# Changelog and Terms of Use

---

Box

-  TABLE OF CONTENTS
- 2017-07-13 Token Exchange
- 2017-06-07 Recent Items API
- 2017-06-05 Chunked Upload API
- 2017-06-01 New Metadata Template Editing Options; Changes to Enterprise Events API
- 2017-05-03 Auto-Generation of RSA Key Pairs and Exportable App Settings
- 2017-05-03 The New Box Developer Console is Now Generally Available
- 2017-05-01 Deprecation: TLS 1.0
- 2017-04-28 Increase in Metadata Template Limit for an Enterprise

Google



## Google APIs Terms of Service

Last modified: December 5, 2014 ([view archived version](#))

Thank you for using Google's APIs, other developer services, and associated software (collectively, "APIs"). By accessing or using our APIs, you are agreeing to the terms below. If there is a conflict between these terms and additional terms applicable to a given API, the additional terms will control for that conflict. Collectively, we refer to the terms below, any additional terms, terms within the accompanying API documentation, and any applicable policies and guidelines as the "Terms." You agree to comply with the Terms and that the Terms control your relationship with us. So please read all the Terms carefully. If you use the APIs as an interface to, or in conjunction with other Google products or services, then the terms for those other products or services also apply.

Under the Terms, "Google" means Google Inc., with offices at 1600 Amphitheatre Parkway, Mountain View, California 94043, United States, unless set forth otherwise in additional terms applicable for a given API. We may refer to "Google" as "we", "our", or "us" in the Terms.

### Section 1: Account and Registration

---

## Tip 2: Write for Humans

---

### Never Assume



Audience is 100% developers



Consumers are fully familiar with API/  
Domain Jargon

- Strive to make documentation readable by Decision Makers
- Start writing in plain English without jargon
- Provide context and information for jargon and domain specific words
- Maintain consistency of tone

# Examples

---

## Coupons

A coupon contains information about a percent-off or amount-off discount you might want to apply to a customer. Coupons only apply to [invoices](#); they do not apply to one-off [charges](#).

### The coupon object

#### ATTRIBUTES

id  
string

object  
string , value is "coupon"

## Invoices

Invoices are statements of what a customer owes for a particular billing period, including subscriptions, invoice items, and any automatic proration adjustments if necessary. Note that invoices at Stripe are part of the recurring billing process and are not intended for one-time charges.

Once an invoice is created, payment is automatically attempted. Note that the payment, while automatic, does not happen exactly at the time of invoice creation. If you have configured webhooks, the invoice will wait until one hour after the last webhook is successfully sent (or the last webhook times out after failing).

Any customer credit on the account is applied before determining how much is due for that invoice (the amount that will be actually charged). If the amount due for the invoice is less than 50 cents (the minimum for a charge), we add the amount to the customer's running account balance to be added to the next invoice. If this amount is negative, it will act as a credit to offset the next invoice. Note that the customer account balance does not include unpaid invoices; it only includes balances that need to be taken into account when calculating the amount due for the next invoice.

### Tip 3: Explain your Request-Response Cycles

---

- Your API Users should know exactly what to expect from a successful call
- Describe full sample response body in every supported format
- Focus not just on the response format, but also HTTP response headers and error codes
- Provide enough context using examples
  1. Examples in Use Cases
  2. Examples in Request-Response Cycles

# Examples

---

HTTP status code summary	
<b>200 - OK</b>	Everything worked as expected.
<b>400 - Bad Request</b>	The request was unacceptable, often due to missing a required parameter.
<b>401 - Unauthorized</b>	No valid API key provided.
<b>402 - Request Failed</b>	The parameters were valid but the request failed.
<b>404 - Not Found</b>	The requested resource doesn't exist.
<b>409 - Conflict</b>	The request conflicts with another request (perhaps due to using the same idempotent key).
<b>429 - Too Many Requests</b>	Too many requests hit the API too quickly. We recommend an exponential backoff of your requests.
<b>500, 502, 503, 504 - Server Errors</b>	Something went wrong on Stripe's end. (These are rare.)

## Stripe

All the Error Codes are described in a separate section for users to follow

# Examples

## YouTube

Well detailed parameters, with an overview section of every resource. These resources are all organized in an easy to follow navigation pane

Overview

- Activities
- Captions
- ChannelBanners
- Channels
- ChannelSections
- Comments
- CommentThreads
- GuideCategories
- I18nLanguages
- I18nRegions
- Playlists
- Playlists
- Search
- Subscriptions
- Thumbnails
- VideoAbuseReportReasons
- VideoCategories
- Videos
- Watermarks
- Standard Query Parameters
- YouTube Data API Errors

Captions: list

Returns a list of caption tracks that are associated with a specified video. Note that the API response does not contain the actual captions and that the [captions.download](#) method provides the ability to retrieve a caption track. [Try it now.](#)

★ Quota impact: A call to this method has a [quota cost](#) of 50 units in addition to the costs of the specified [resource parts](#).

Request

HTTP request

```
GET https://www.googleapis.com/youtube/v3/captions
```

Authorization

This request requires authorization with at least one of the following scopes ([read more about authentication and authorization](#)).

Scope

```
https://www.googleapis.com/auth/youtube.force-ssl
```

```
https://www.googleapis.com/auth/youtubepartner
```

Parameters

The following table lists the parameters that this query supports. All of the parameters listed are query parameters.

Parameters	
Required parameters	
part	string The part parameter specifies the caption resource parts that the API response will include.  The list below contains the part names that you can include in the parameter value and the <a href="#">quota cost</a> for each part: <ul style="list-style-type: none"><li>• id: 0</li><li>• snippet: 1</li></ul>
videoId	string The videoId parameter specifies the YouTube video ID of the video for which the API should return caption tracks.
Optional parameters	
id	string The id parameter specifies a comma-separated list of IDs that identify the caption resources that should be retrieved. Each ID must identify a caption track associated with the specified video.
onBehalfOfContentOwner	string This parameter can only be used in a properly <a href="#">authorized request</a> . Note: This parameter is intended exclusively for YouTube content partners.  The onBehalfOfContentOwner parameter indicates that the request's authorization credentials identify a YouTube CMS user who is acting on behalf of the content owner specified in the parameter value. This parameter is intended for YouTube content partners that own and manage many different YouTube channels. It allows content owners to authenticate once

## Tip 4: Experimentation is Power

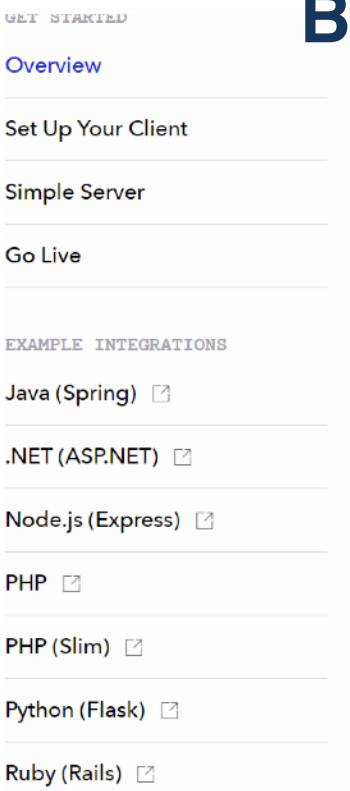
---

The difference between documentation and good documentation is a little more effort with these Nice-to-haves.

Allow developers to experiment with your API

1. Getting Started Guides
2. Interactive Docs and Console
3. SDKs
4. Tutorials

# Examples



## Braintree

### Get Started

Braintree provides complementary client and server SDKs to complete your integration:

- The client SDKs enable you to collect payment method (e.g. credit card, PayPal) details
- The server SDKs manage all requests to the Braintree gateway

Before we get started, there are two key concepts to introduce: the client token and the payment method nonce.

#### Client token

A client token is a signed data blob that includes configuration and authorization information required by the [Braintree client SDK](#). These should not be reused; a new client token should be generated for each request that's sent to Braintree. For security, we will revoke client tokens if they are reused excessively within a short time period.

# Examples

## API console

Microsoft Graph Explore lets you try various end points and see what the response packet is

Using demo tenant [SIGN IN](#)

V1.0 ▾ GET ▾ https://graph.microsoft.com/v1.0/me/ X GO ↻

REQUEST HEADER REQUEST BODY

RESPONSE

```
odata-version: 4.0
content-encoding: gzip
vary: Accept-Encoding
x-ms-ags-diagnostic: {"ServerInfo":{"DataCenter":"West US","Slice":"SliceB","ScaleUnit":"000","Host":"AGSFE_IN_4","ADSiteName":"WST"}}
duration: 131.6833
client-request-id: 98f1f330-6421-4e92-bfde-ebbd18a6e8b4
content-type: application/json;odata.metadata=minimal;odata.streaming=true;IEEE754Compatible=false;charset=utf-8
access-control-allow-origin: https://graph.microsoft.io
access-control-expose-headers: Cache-Control, Content-Type, Content-Encoding, Vary, request-id, client-request-id, x-ms-ags-diagnostic, Access-Control-Allow-Origin,
cache-control: private
request-id: 98f1f330-6421-4e92-bfde-ebbd18a6e8b4
Status Code: 200
{
    "@odata.context": "https://graph.microsoft.com/v1.0/$metadata#users/$entity",
    "id": "b63d5fb9-4f43-44c4-8f9d-fd0727842876",
    "businessPhones": [
        "4255550100"
    ],
    "displayName": "Alex Darrow",
    "givenName": "Alex",
    "jobTitle": null,
    "mail": "alex@a830edad9050849NDA1.onmicrosoft.com",
    "mobilePhone": null,
    "officeLocation": null,
    "preferredLanguage": "en-US",
    "surname": "Darrow",
    "userPrincipalName": "alex@a830edad9050849NDA1.onmicrosoft.com"
}
```

# Summary

---

- Good API Developer Experience can help drive business and tech goals
- Three step framework for understanding API DX
  - Understand audience
  - Understand journey
  - Map journey to audience
- Tips for Good API Documentation
  - List the Fundamentals
  - Write for Humans
  - Explain your Request-Response Cycles
  - Empower with Experimentation

# Questions?