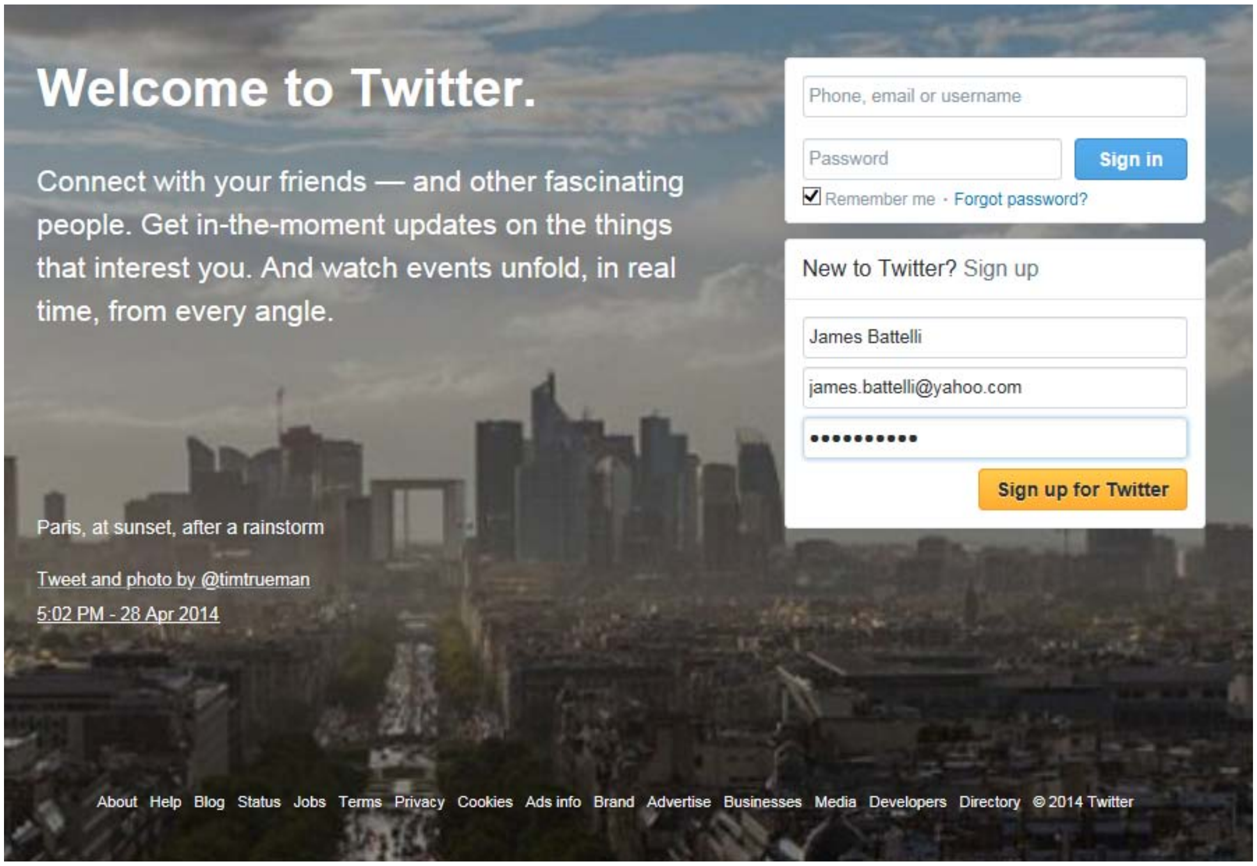
Twitter API

Understanding Twitter OAuth

Step 1: Create a Twitter Account



Welcome to Twitter.

Connect with your friends — and other fascinating people. Get in-the-moment updates on the things that interest you. And watch events unfold, in real time, from every angle.

Paris, at sunset, after a rainstorm

Tweet and photo by @timtrueman

5:02 PM - 28 Apr 2014

Phone, email or username

Password

Sign in

Remember me · Forgot password?

New to Twitter? Sign up

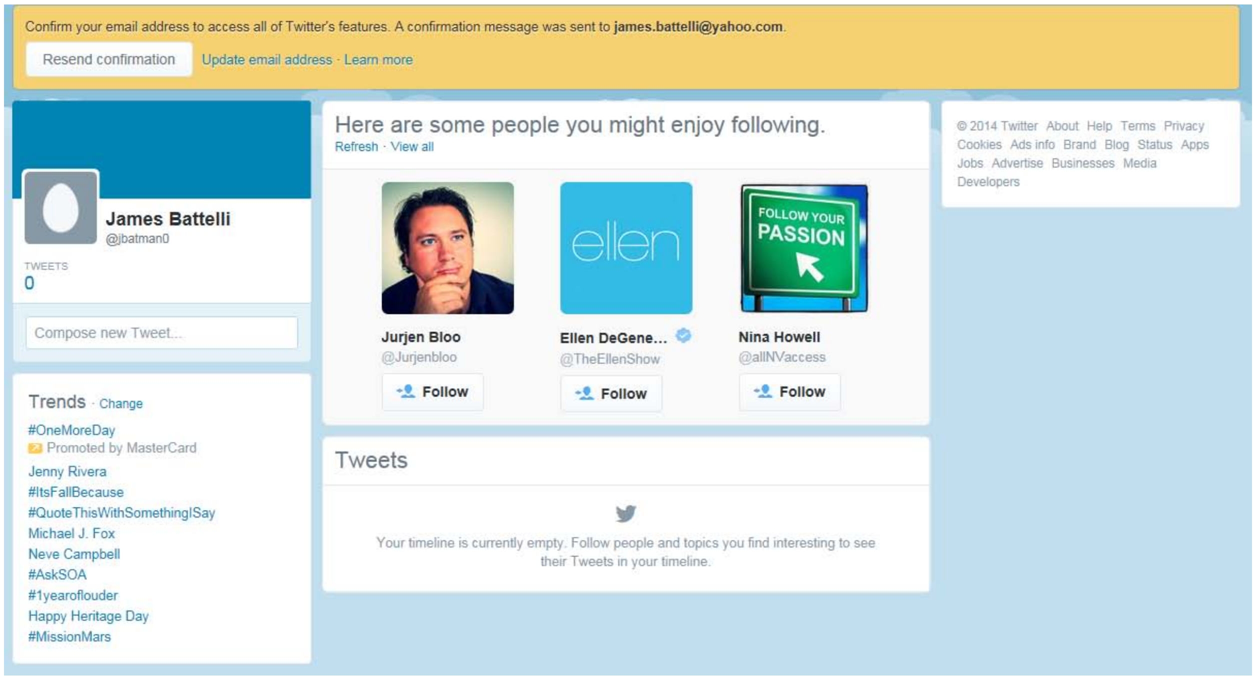
James Battelli

james.battelli@yahoo.com

•••••••••••

Sign up for Twitter

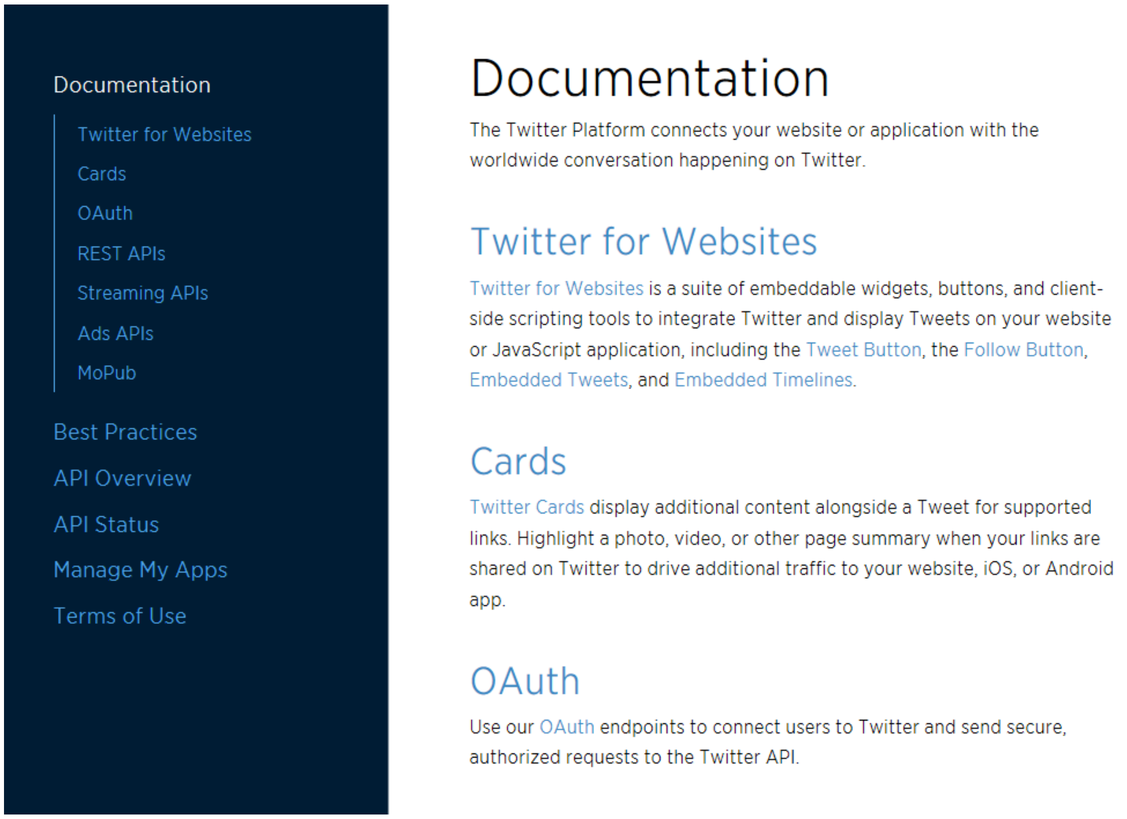
About Help Blog Status Jobs Terms Privacy Cookies Ads info Brand Advertise Businesses Media Developers Directory © 2014 Twitter



Confirm your email address to access all of Twitter's features. A confirmation message was sent to **james.battelli@yahoo.com**.

Your timeline is currently empty. Follow people and topics you find interesting to see their Tweets in your timeline.

Step 2: Once you have created your Twitter account learn about OAuth



Documentation

The Twitter Platform connects your website or application with the worldwide conversation happening on Twitter.

Twitter for Websites

Twitter for Websites is a suite of embeddable widgets, buttons, and client-side scripting tools to integrate Twitter and display Tweets on your website or JavaScript application, including the Tweet Button, the Follow Button, Embedded Tweets, and Embedded Timelines.

Cards

Twitter Cards display additional content alongside a Tweet for supported links. Highlight a photo, video, or other page summary when your links are shared on Twitter to drive additional traffic to your website, iOS, or Android app.

OAuth

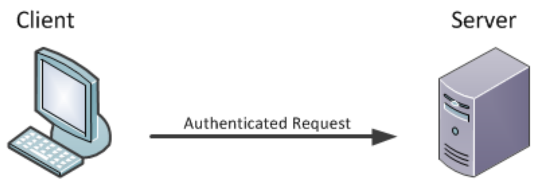
Use our OAuth endpoints to connect users to Twitter and send secure, authorized requests to the Twitter API.

Terminology

Client, Server, and Resource Owner

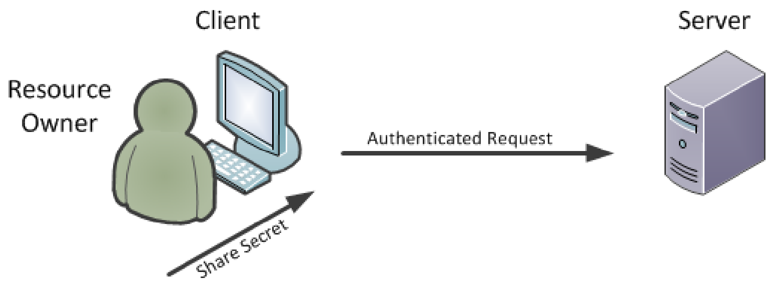
OAuth defines three roles: client, server, and resource owner (nicknamed the OAuth Love Triangle by Leah Culver). These three roles are present in any OAuth transaction; in some cases the client is also the resource owner. The original version of the specification used a different set of terms for these roles: **consumer** (client), **service provider** (server), and **user** (resource owner).

In the traditional client-server authentication model, the client uses **its** credentials to access **its** resources hosted on the server. As far as the server is concerned, the shared secret used by the client belongs to the client. The server doesn't really care where it came from or if the client is acting on behalf of some other entity. As long as the shared secret matches the server's expectation, the request is processed.



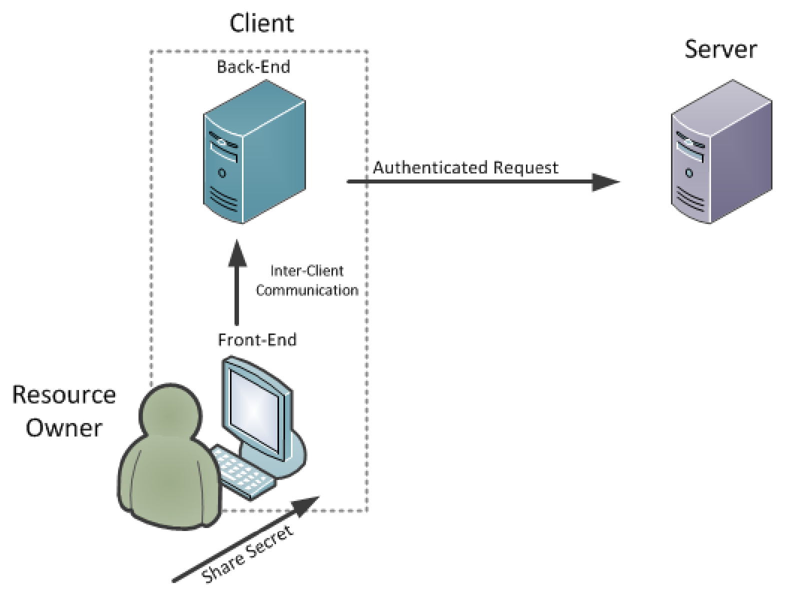
There are many times when the client is acting on behalf of another entity. That entity can be another machine or person. When such a third actor is involved, typically a user interacting with the client, the client is acting on the user's behalf. In these cases, the client is not accessing its own resource but those of the user – the resource owner.

Instead of using the client's credentials, the client is using the resource owner's credentials to make requests – pretending to be the resource owner. User credentials typically include a username or screen-name and a password, but resource owners are not limited to users, they can be any entity controlling the server resources.



The model gets a bit more detailed when the client is a web-based application. In that case, the client is split between a front-end component, usually running within a web browser on the resource owner's desktop, and a back-end component, running on the client's server.

The resource owner is interacting with one part of the client application while the server is receiving requests from another part. However, no matter what internal architecture the client uses, it is still acting as a single entity and on behalf of the resource owner.



Protected Resources

A protected resource is a resource stored on (or provided by) the server which requires authentication in order to access it. Protected resources are owned or controlled by the resource owner. Anyone requesting access to a protected resource must be authorized to do so by the resource owner (enforced by the server).

A protected resource can be data (photos, documents, contacts), services (posting blog item, transferring funds), or any resource requiring access restrictions. While OAuth can be used with other transport protocols, it is only defined for HTTP(S) resources.

2-Legged, 3-Legged, n-Legged

The number of legs used to describe an OAuth request typically refers to the number of parties involved. In the simple OAuth flow: a client, a server, and a resource owner, the flow is described as 3-legged. When the client is also the resource owner (that is, acting on behalf of itself), it is described as 2-legged. Additional legs usually mean different things to different people, but in general mean that access is shared by the client with other clients (re-delegation).

Credentials and Tokens

OAuth uses three kinds of credentials: client credentials, temporary credentials, and token credentials. The original version of the specification used a different set of terms for these credentials: **consumer key and secret** (client credentials), **request token and secret** (temporary credentials), and **access token and secret** (token credentials). The specification still uses a parameter name '*oauth_consumer_key*' for backwards compatibility.

The client credentials are used to authenticate the client. This allows the server to collect information about the clients using its services, offer some clients special treatment such as throttling-free access, or provide the resource owner with more information about the clients seeking to access its protected resources. In some cases, the client credentials cannot be trusted and can only be used for informational purposes only, such as in desktop application clients.

Token credentials are used in place of the resource owner's username and password. Instead of having the resource owner share its credentials with the client, it authorizes the server to issue a special class of credentials to the client which represent the access grant given to the client by the resource owner. The client uses the token credentials to access the protected resource without having to know the resource owner's password.

Token credentials include a token identifier, usually (but not always) a random string of letters and numbers that is unique, hard to guess, and paired with a secret to protect the token from being used by unauthorized parties. Token credentials are usually limited in scope and duration, and can be revoked at any time by the resource owner without affecting other token credentials issued to other clients.

The OAuth authorization process also uses a set of temporary credentials which are used to identify the authorization request. In order to accommodate different kind of clients (web-based, desktop, mobile, etc.), the temporary credentials offer additional flexibility and security.

In OAuth 1.0, the secret half of each set of credentials is defined as a symmetric shared secret. This means that both the client and server must have access to the same secret string. However, OAuth supports an RSA-based authentication method which uses an asymmetric client secret. The different credentials are explained in more detailed later on.

Specification Structure

The OAuth specification consists of two parts. The first part defines a redirection-based browser process for end-users to authorize client access to their resources. This is done by having the users authenticate directly with the server, instructing the server to provision tokens to the client for use with the authentication method.

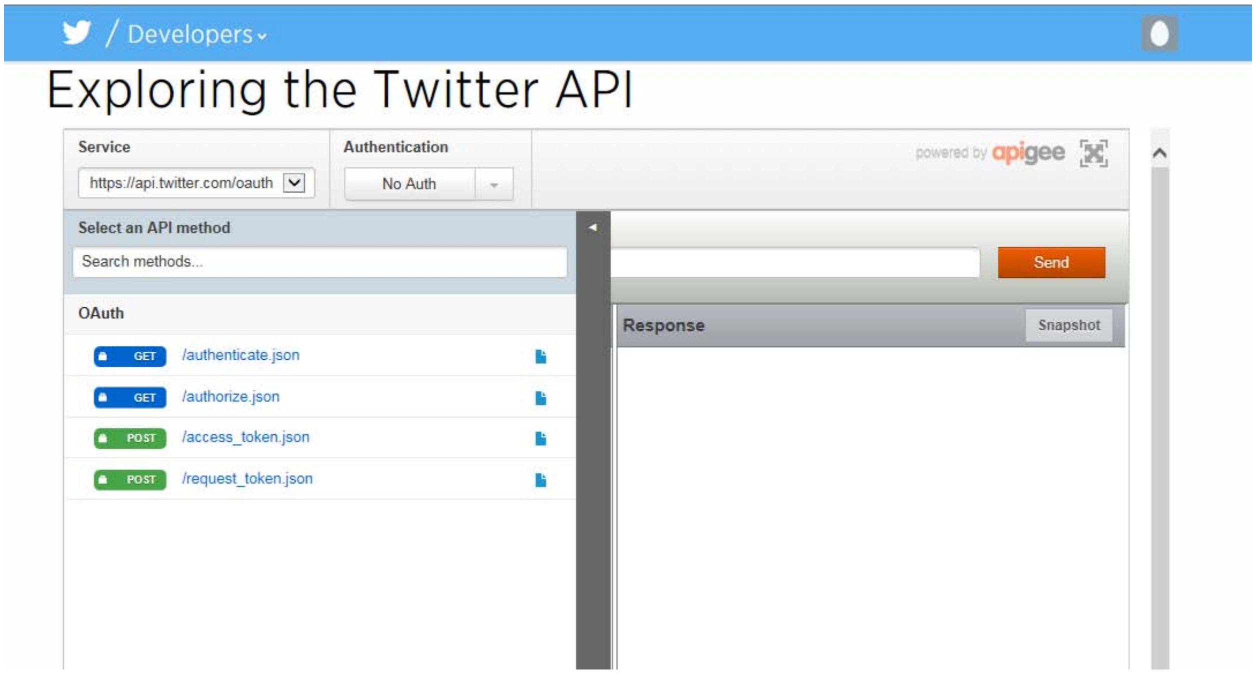
The second part defines a method for making authenticated HTTP requests using two sets of credentials, one identifying the client making the request, and the other identifying the resource owner on whose behalf the request is being made.

The following is an outline of the OAuth 1.0 protocol specification:

1. **Introduction** – the introduction provides a quick overview of the specification and its objectives. The terminology sub-section defines the terms used and their relation to the HTTP specification (this guide provides a more detailed description). The example sub-section provides a full walk-through, describing a typical use case of a user sharing photos on one site and looking to print them on another.
2. **Redirection-Based Authorization** – the authorization flow is what most people think of when they talk about OAuth. It is the process in which the user is redirected to the server to provide access. This section describes the three steps used to request and grant access: Obtaining Temporary Credentials, Requesting Resource Owner Authorization, and Obtaining Token Credentials.
3. **Authenticated Requests** – In order for the client to obtain a set of token credentials and use it to access protected resources, the client must make authenticated HTTP requests. This section describes how the client makes such requests, how the server verifies them, and the various steps and cryptographic options available. Most of this section is dedicated to the construction of the signature base string, the normalized version of the request used for signing.
4. **Security Considerations** – this often-overlooked section is a must-read for any developer writing client or server code. It provides a comprehensive (but never complete) list of issues which can greatly impact the security properties of any given implementation. Developers should read this list before starting any OAuth related project, and read it at least once more when they are done to review.

Another section worth mentioning is **Appendix A** which lists the differences from the community edition. While the RFC edition is the same as the OAuth Core 1.0 Revision A specification, some of its clarifications and errata require code changes on both the client and server sides. Developers working on existing implementations should pay close attentions to the variations described.

Step 3: Explore the Twitter API



Exploring the Twitter API

Service: https://api.twitter.com/oauth

Authentication: No Auth

Select an API method

Search methods...

OAuth

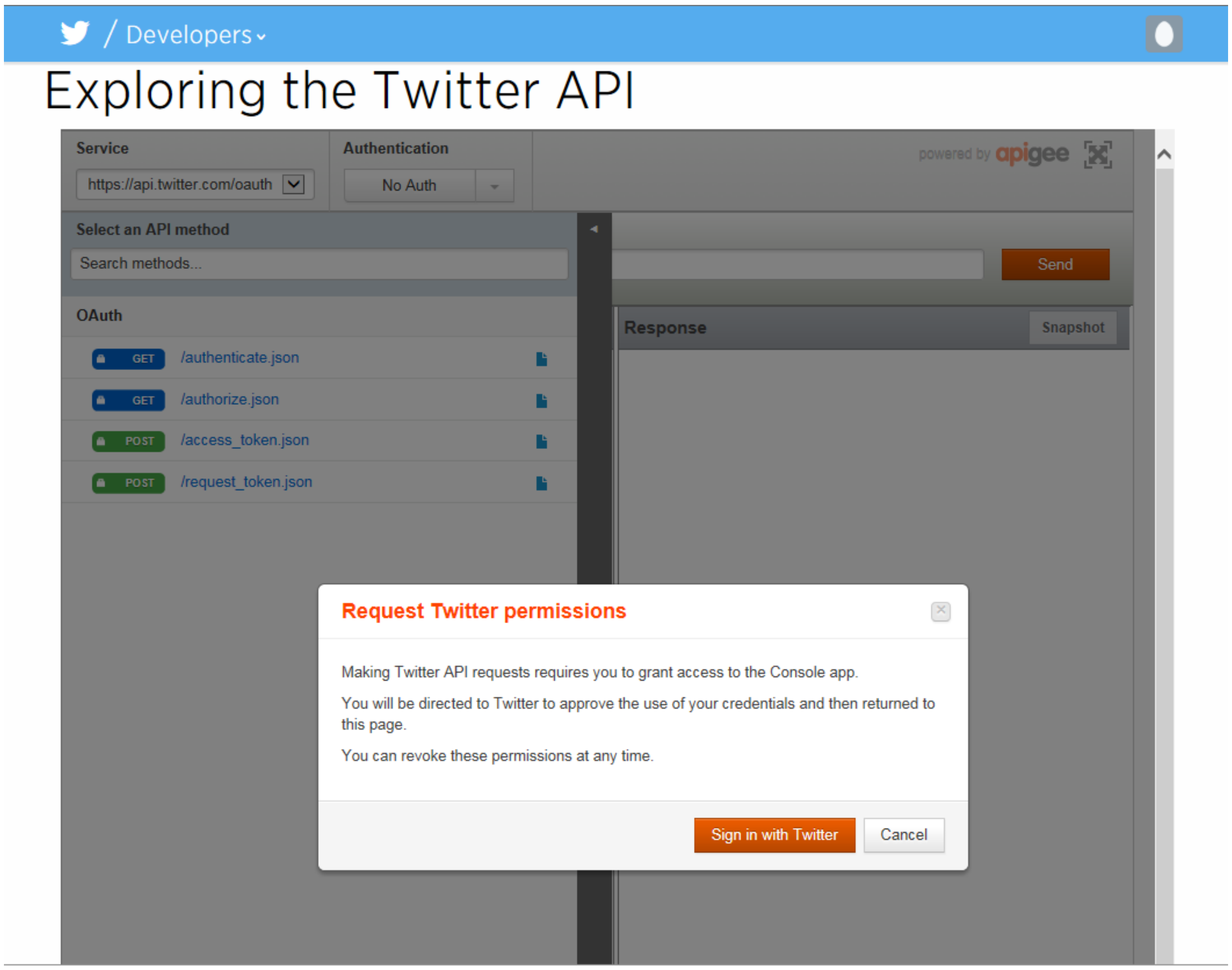
GET /authenticate.json

GET /authorize.json

POST /access_token.json

POST /request_token.json

Response

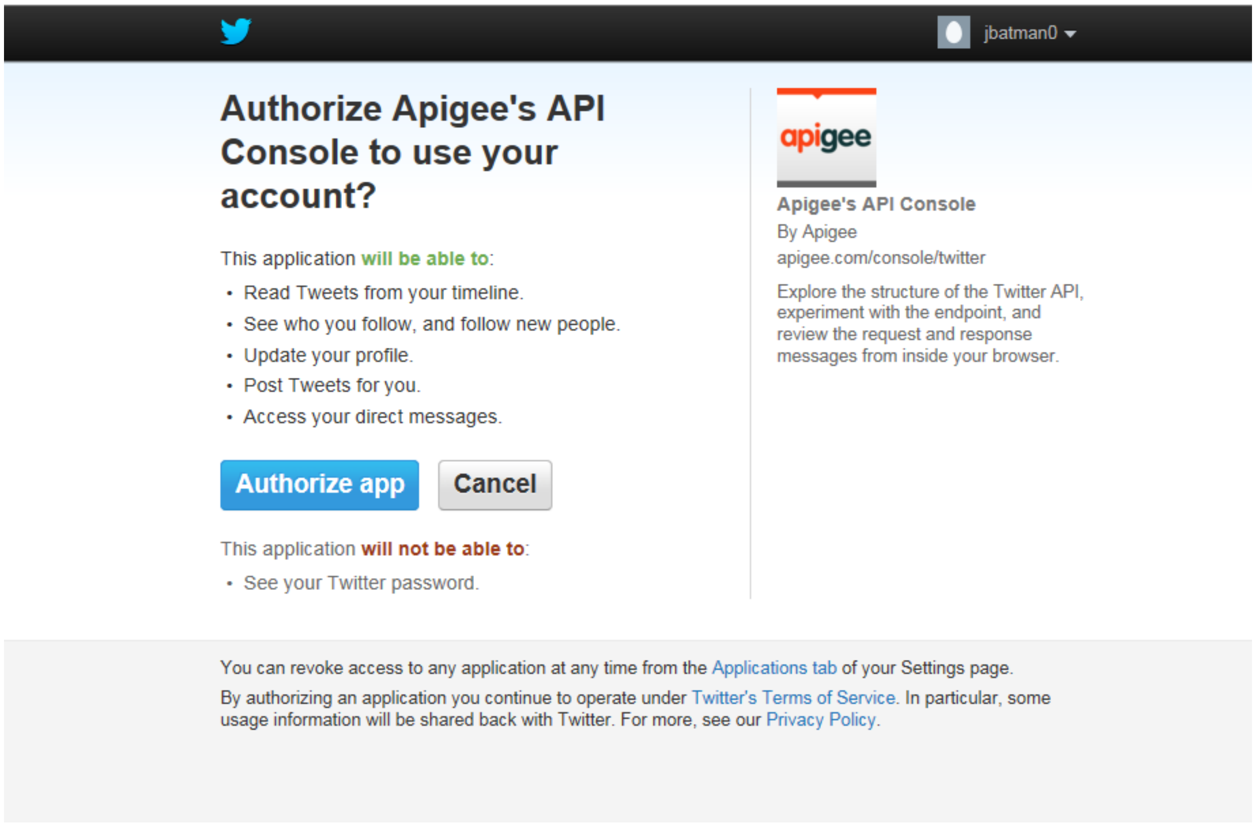


Request Twitter permissions

Making Twitter API requests requires you to grant access to the Console app.

You will be directed to Twitter to approve the use of your credentials and then returned to this page.

You can revoke these permissions at any time.



Authorize Apigee's API Console to use your account?

This application **will be able to**:

- Read Tweets from your timeline.
- See who you follow, and follow new people.
- Update your profile.
- Post Tweets for you.
- Access your direct messages.

This application **will not be able to**:

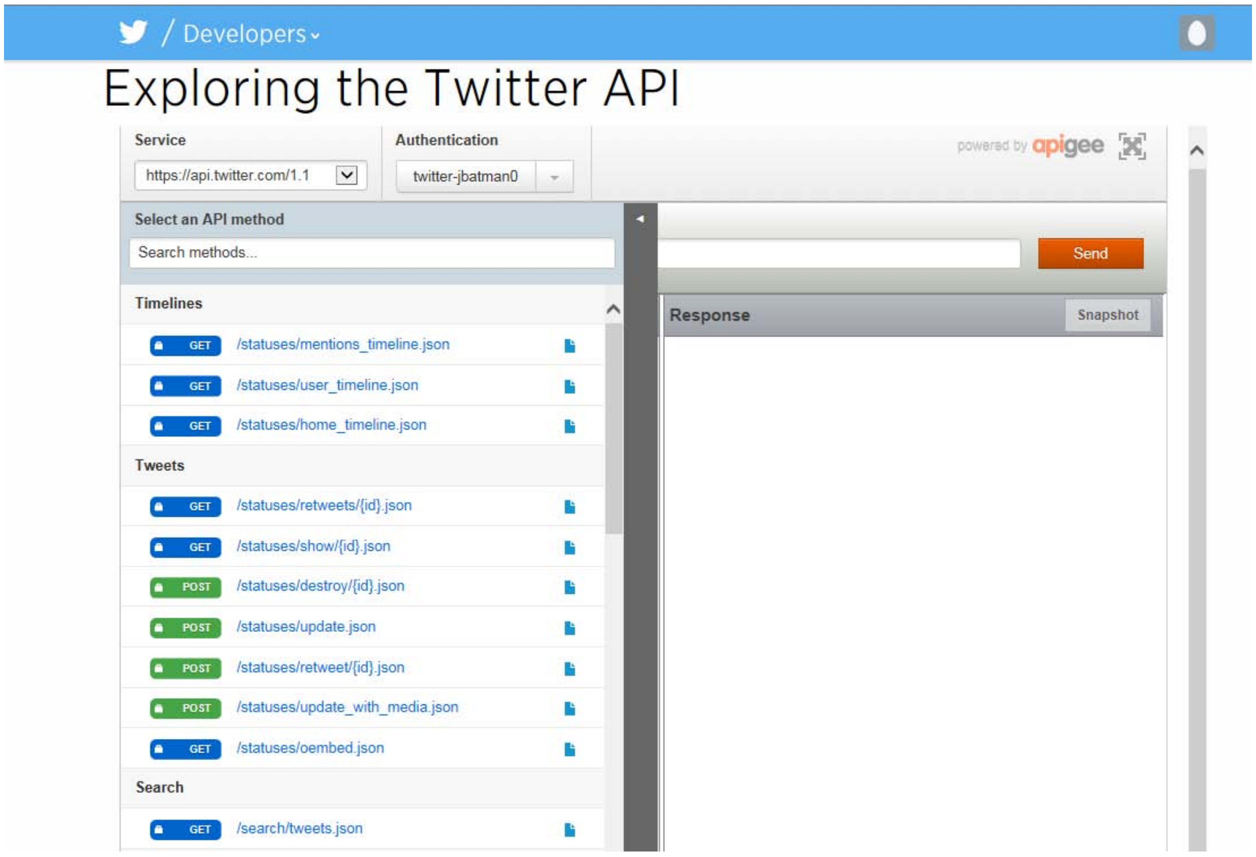
- See your Twitter password.

Apigee's API Console
By Apigee
apigee.com/console/twitter

Explore the structure of the Twitter API, experiment with the endpoint, and review the request and response messages from inside your browser.

You can revoke access to any application at any time from the Applications tab of your Settings page.

By authorizing an application you continue to operate under Twitter's Terms of Service. In particular, some usage information will be shared back with Twitter. For more, see our Privacy Policy.



Exploring the Twitter API

Service: https://api.twitter.com/1.1
Authentication: twitter-jbatman0

Timelines

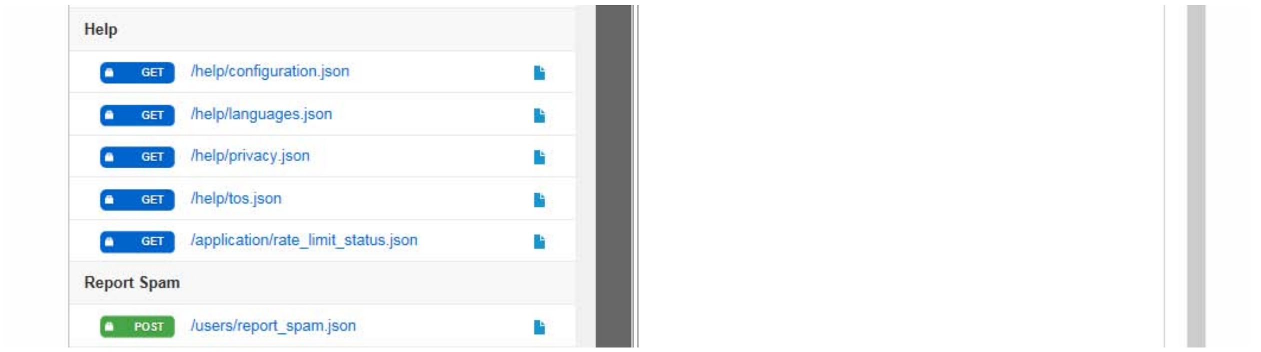
- GET /statuses/mentions_timeline.json
- GET /statuses/user_timeline.json
- GET /statuses/home_timeline.json

Tweets

- GET /statuses/retweets/{id}.json
- GET /statuses/show/{id}.json
- POST /statuses/destroy/{id}.json
- POST /statuses/update.json
- POST /statuses/retweet/{id}.json
- POST /statuses/update_with_media.json
- GET /statuses/oembed.json

Search

- GET /search/tweets.json

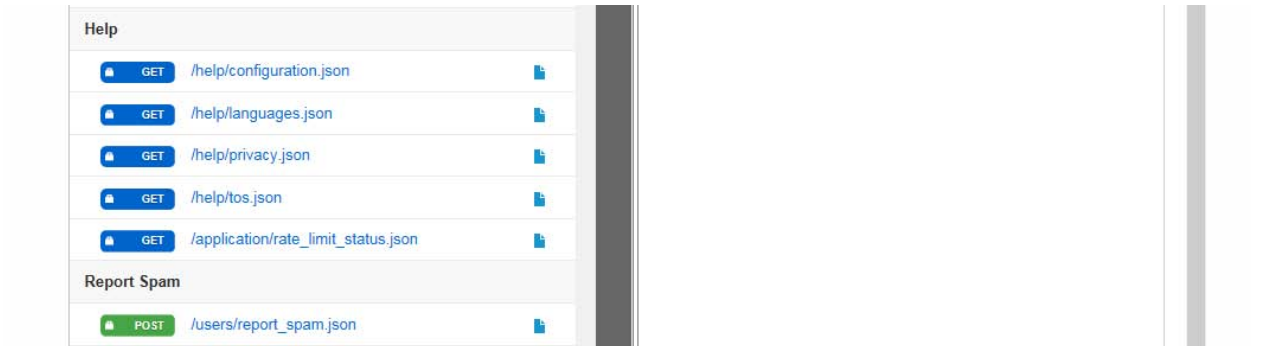


Help

- GET /help/configuration.json
- GET /help/languages.json
- GET /help/privacy.json
- GET /help/tos.json
- GET /application/rate_limit_status.json

Report Spam

- POST /users/report_spam.json

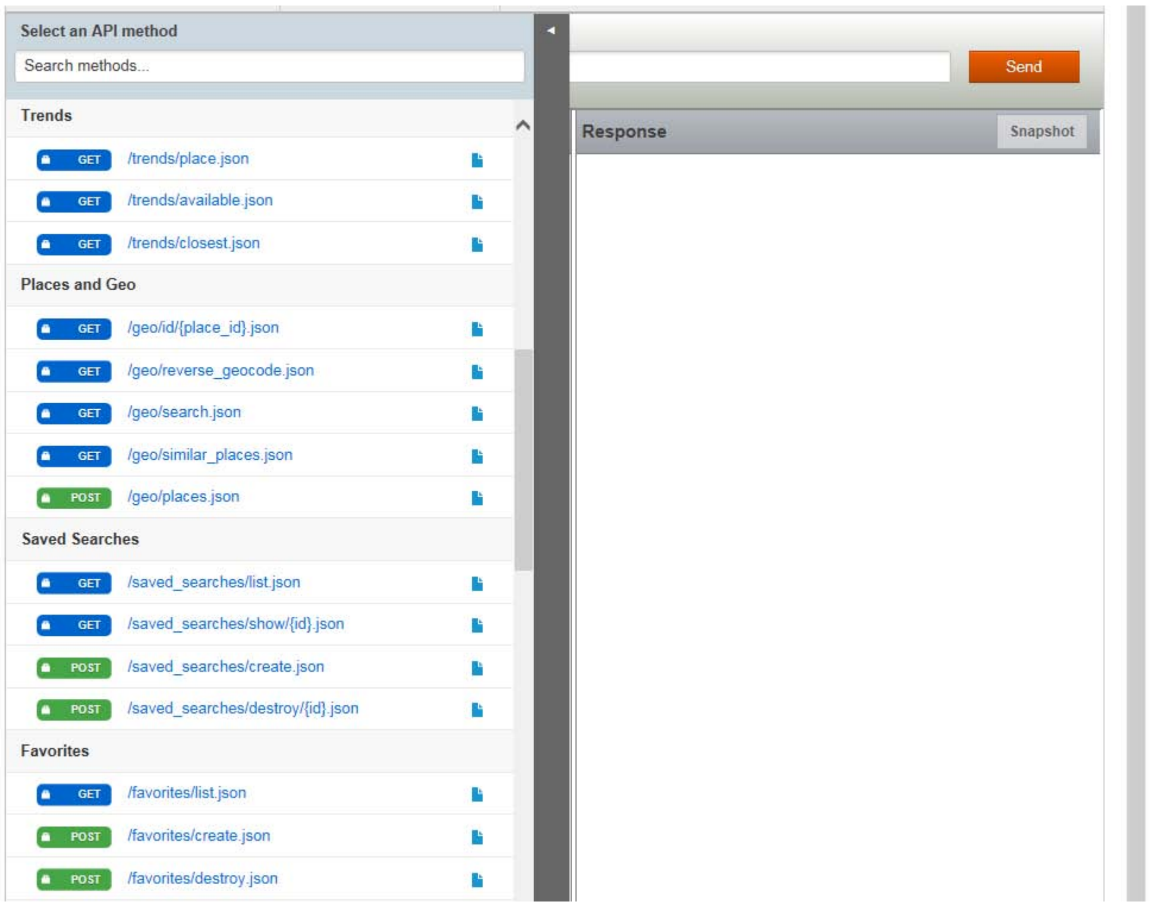


Help

- GET /help/configuration.json
- GET /help/languages.json
- GET /help/privacy.json
- GET /help/tos.json
- GET /application/rate_limit_status.json

Report Spam

- POST /users/report_spam.json



Trends

- GET /trends/place.json
- GET /trends/available.json
- GET /trends/closest.json

Places and Geo

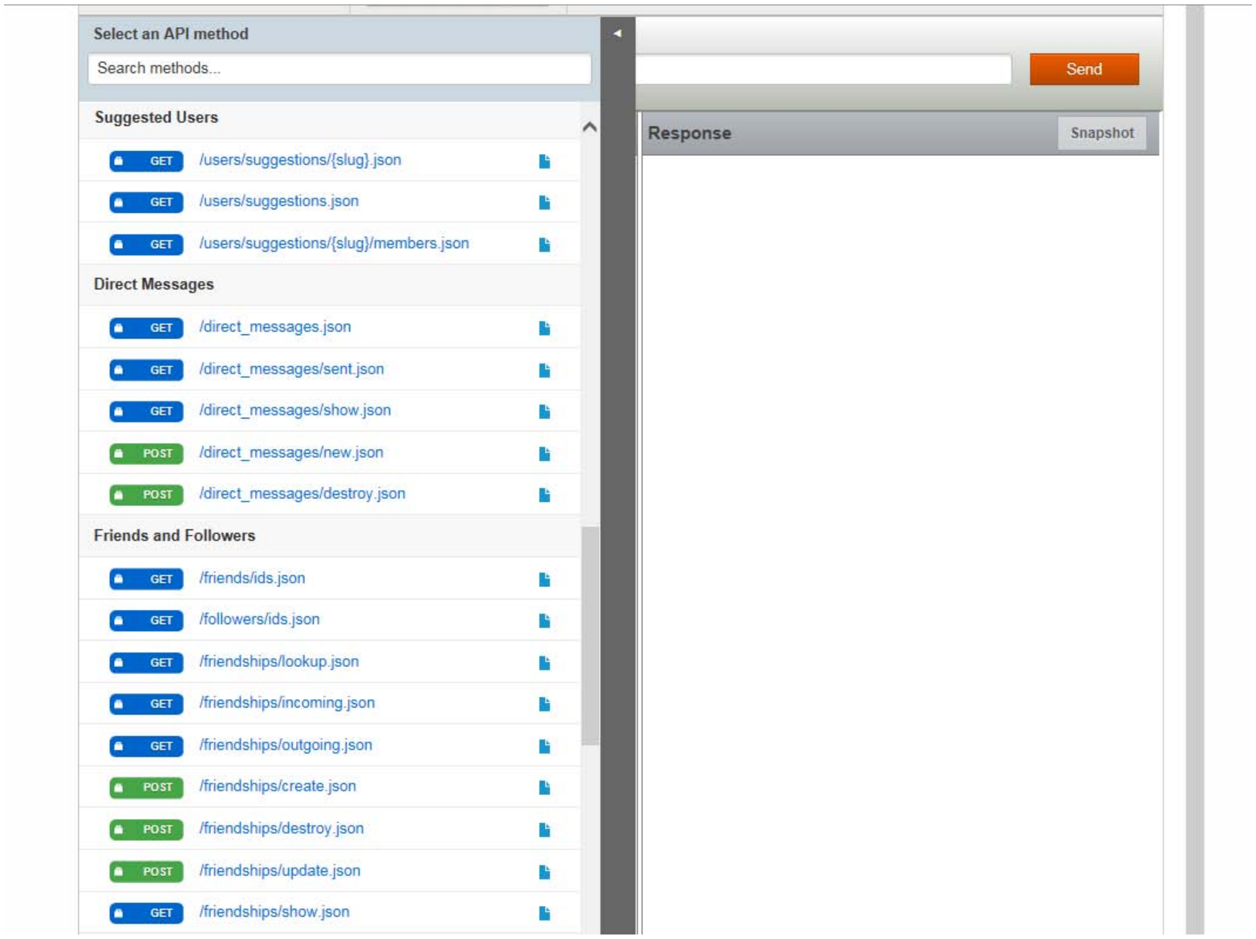
- GET /geo/id/{place_id}.json
- GET /geo/reverse_geocode.json
- GET /geo/search.json
- GET /geo/similar_places.json
- POST /geo/places.json

Saved Searches

- GET /saved_searches/list.json
- GET /saved_searches/show/{id}.json
- POST /saved_searches/create.json
- POST /saved_searches/destroy/{id}.json

Favorites

- GET /favorites/list.json
- POST /favorites/create.json
- POST /favorites/destroy.json


Select an API method

Search methods...

Send

Response

Snapshot

Suggested Users

GET /users/suggestions/{slug}.json

GET /users/suggestions.json

GET /users/suggestions/{slug}/members.json

Direct Messages

GET /direct_messages.json

GET /direct_messages/sent.json

GET /direct_messages/show.json

POST /direct_messages/new.json

POST /direct_messages/destroy.json

Friends and Followers

GET /friends/ids.json

GET /followers/ids.json

GET /friendships/lookup.json

GET /friendships/incoming.json

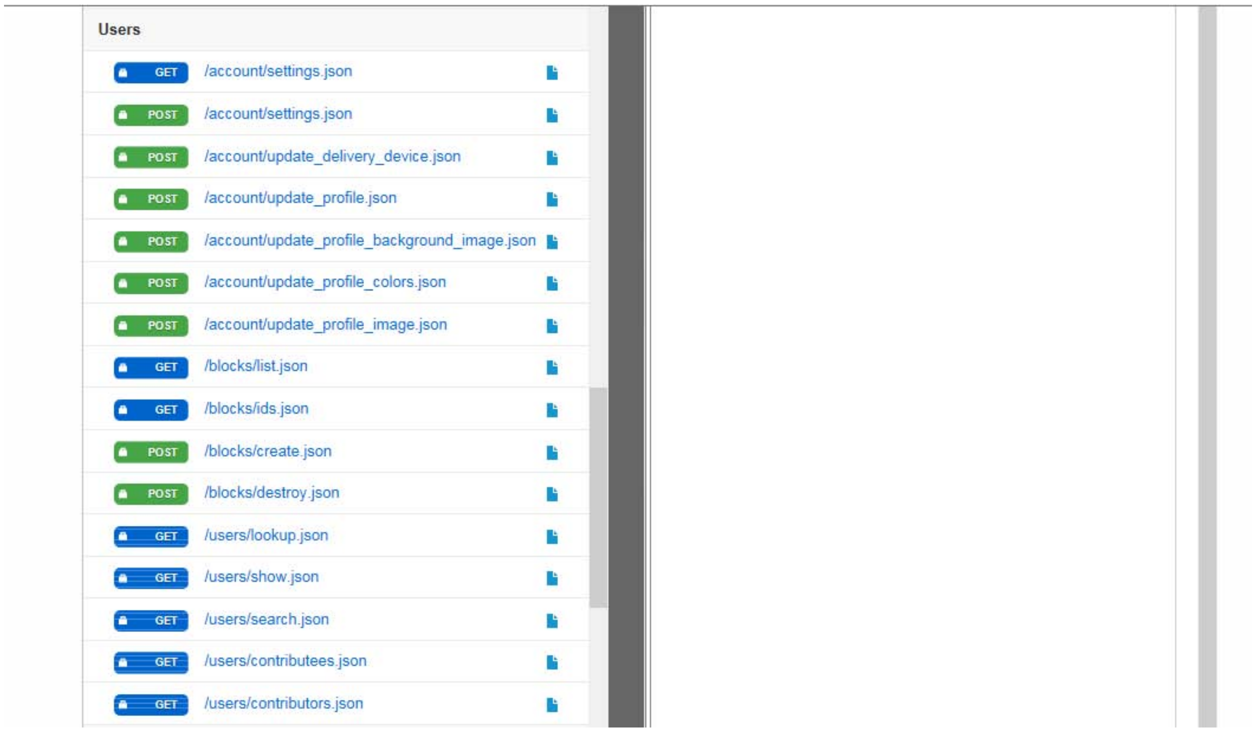
GET /friendships/outgoing.json

POST /friendships/create.json

POST /friendships/destroy.json

POST /friendships/update.json

GET /friendships/show.json


Users

GET /account/settings.json

POST /account/settings.json

POST /account/update_delivery_device.json

POST /account/update_profile.json

POST /account/update_profile_background_image.json

POST /account/update_profile_colors.json

POST /account/update_profile_image.json

GET /blocks/list.json

GET /blocks/ids.json

POST /blocks/create.json

POST /blocks/destroy.json

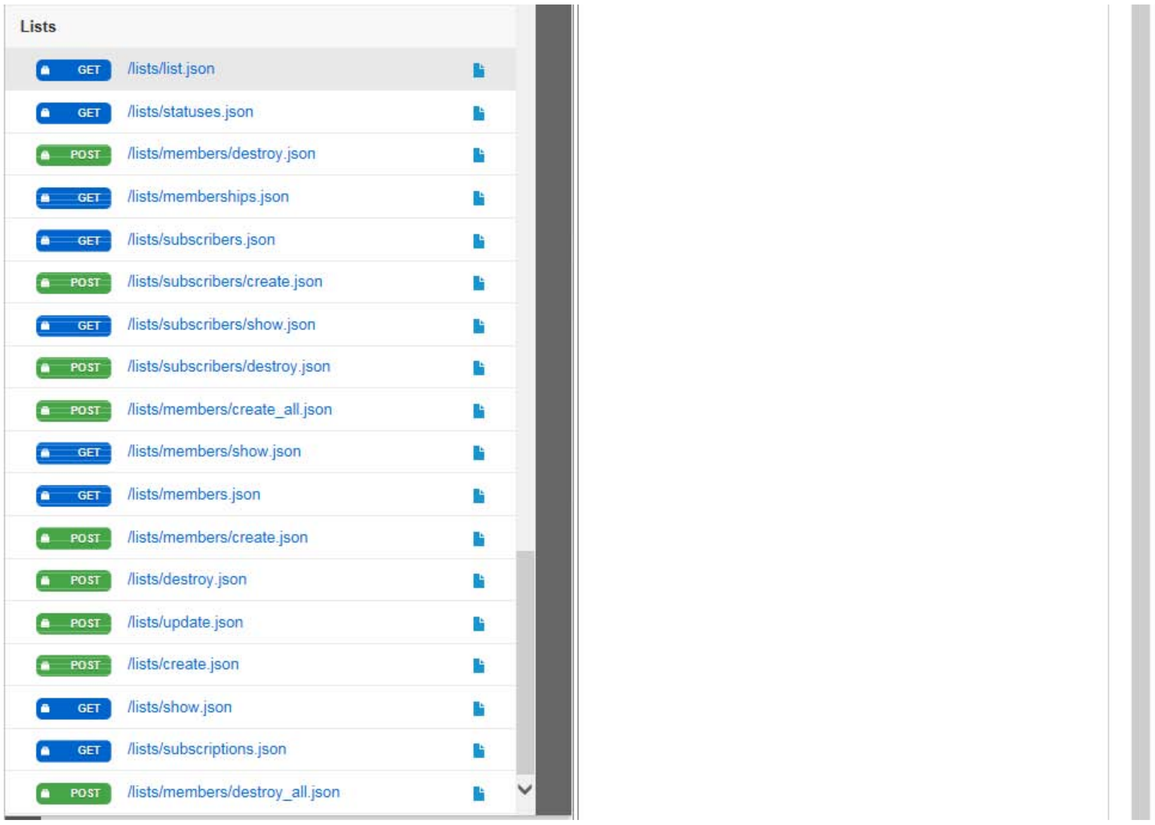
GET /users/lookup.json

GET /users/show.json

GET /users/search.json

GET /users/contributees.json

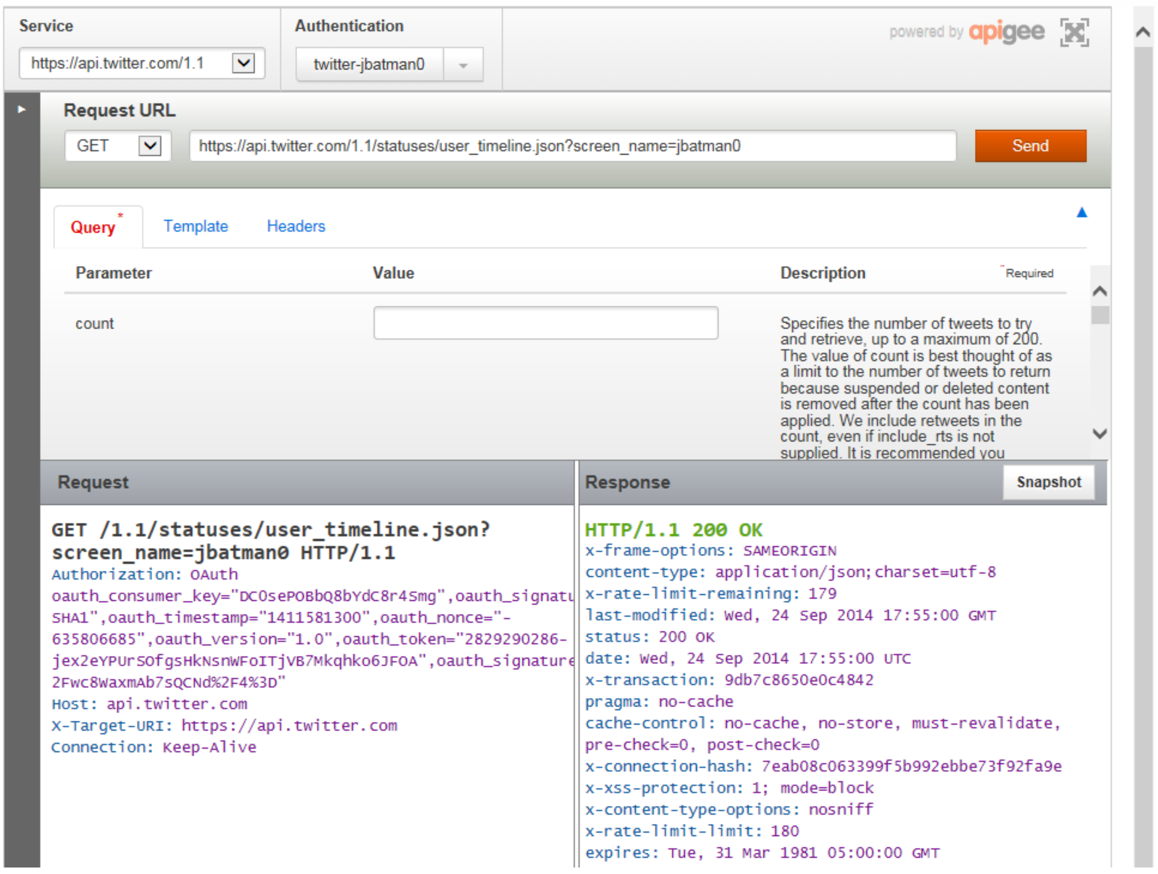
GET /users/contributors.json



Lists

GET /lists/list.json
GET /lists/statuses.json
POST /lists/members/destroy.json
GET /lists/memberships.json
GET /lists/subscribers.json
POST /lists/subscribers/create.json
GET /lists/subscribers/show.json
POST /lists/subscribers/destroy.json
POST /lists/members/create_all.json
GET /lists/members/show.json
GET /lists/members.json
POST /lists/members/create.json
POST /lists/destroy.json
POST /lists/update.json
POST /lists/create.json
GET /lists/show.json
GET /lists/subscriptions.json
POST /lists/members/destroy_all.json

Exploring the Twitter API



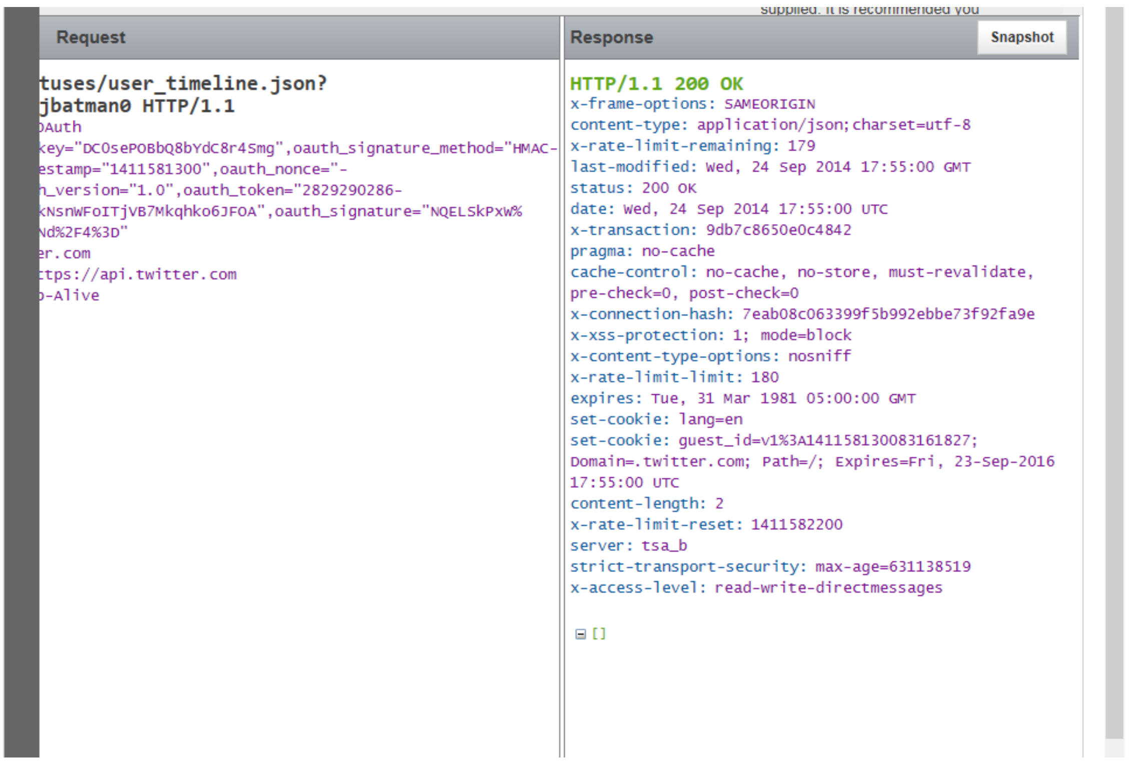
https://api.twitter.com/1.1/statuses/user_timeline.json?screen_name=jbatman0

count

Specifies the number of tweets to try and retrieve, up to a maximum of 200. The value of count is best thought of as a limit to the number of tweets to return because suspended or deleted content is removed after the count has been applied. We include retweets in the count, even if include_rts is not supplied. It is recommended you

```
GET /1.1/statuses/user_timeline.json?
screen_name=jbatman0 HTTP/1.1
Authorization: OAuth
oauth_consumer_key="DCOsePOBbQ8bYdC8r4Smg",oauth_signatu
SHA1",oauth_timestamp="1411581300",oauth_nonce="-
635806685",oauth_version="1.0",oauth_token="2829290286-
jex2eYPUrSOfgsHkNsnWFoITjVB7Mkqhko6JFOA",oauth_signature
2Fwc8WaxmAb7sQCNd%2F4%3D"
Host: api.twitter.com
X-Target-URI: https://api.twitter.com
Connection: Keep-Alive
```

```
HTTP/1.1 200 OK
x-frame-options: SAMEORIGIN
content-type: application/json;charset=utf-8
x-rate-limit-remaining: 179
last-modified: Wed, 24 Sep 2014 17:55:00 GMT
status: 200 OK
date: Wed, 24 Sep 2014 17:55:00 UTC
x-transaction: 9db7c8650e0c4842
pragma: no-cache
cache-control: no-cache, no-store, must-revalidate,
pre-check=0, post-check=0
x-connection-hash: 7eab08c063399f5b992ebbe73f92fa9e
x-xss-protection: 1; mode=block
x-content-type-options: nosniff
x-rate-limit-limit: 180
expires: Tue, 31 Mar 1981 05:00:00 GMT
```



supplied. It is recommended you

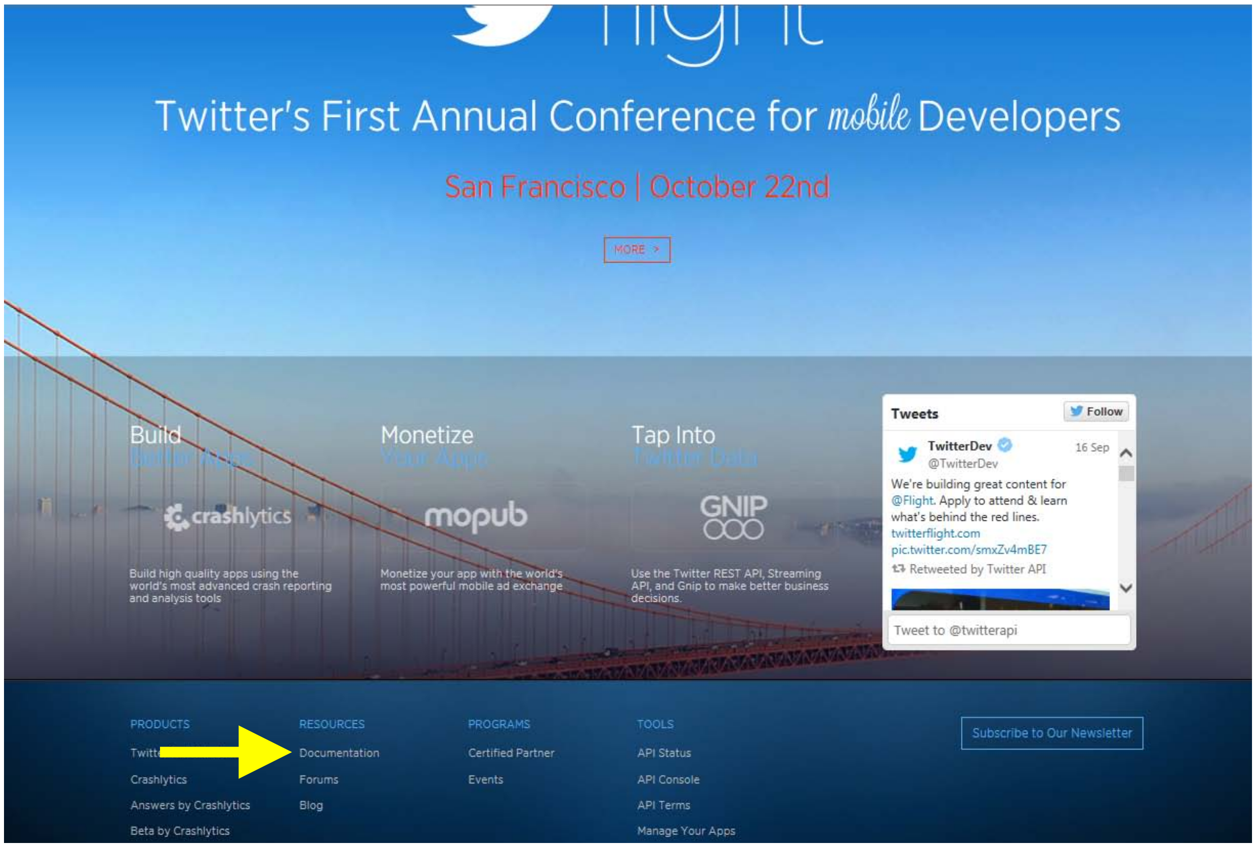
```
tuses/user_timeline.json?
jbatman0 HTTP/1.1
DAuth
key="DCOsePOBbQ8bYdC8r4Smg",oauth_signature_method="HMAC-
estamp="1411581300",oauth_nonce="-
h_version="1.0",oauth_token="2829290286-
kNsnWFoITjVB7Mkqhko6JFOA",oauth_signature="NQELSkPxW%
Nd%2F4%3D"
er.com
ttps://api.twitter.com
p-Alive
```

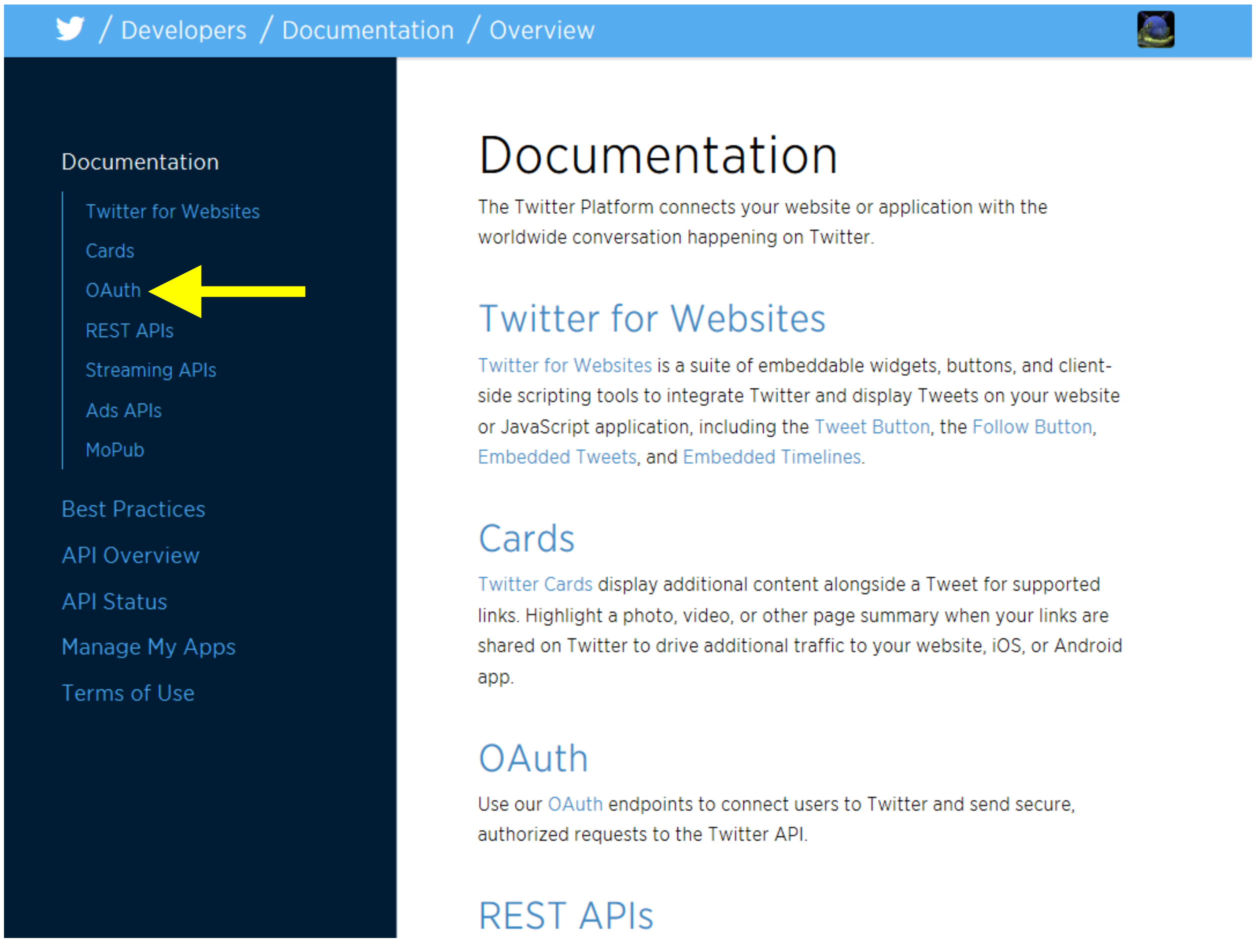
```
HTTP/1.1 200 OK
x-frame-options: SAMEORIGIN
content-type: application/json;charset=utf-8
x-rate-limit-remaining: 179
last-modified: Wed, 24 Sep 2014 17:55:00 GMT
status: 200 OK
date: Wed, 24 Sep 2014 17:55:00 UTC
x-transaction: 9db7c8650e0c4842
pragma: no-cache
cache-control: no-cache, no-store, must-revalidate,
pre-check=0, post-check=0
x-connection-hash: 7eab08c063399f5b992ebbe73f92fa9e
x-xss-protection: 1; mode=block
x-content-type-options: nosniff
x-rate-limit-limit: 180
expires: Tue, 31 Mar 1981 05:00:00 GMT
set-cookie: lang=en
set-cookie: guest_id=v1%3A141158130083161827;
Domain=.twitter.com; Path=/; Expires=Fri, 23-Sep-2016
17:55:00 UTC
content-length: 2
x-rate-limit-reset: 1411582200
server: tsa_b
strict-transport-security: max-age=631138519
x-access-level: read-write-directmessages

⊟ []
```

Step 4: Obtain a Twitter Access Token for Your Application... The token you select will depend on what you need to access from Twitter and how you will use the information.

Step 5: Goto https://dev.twitter.com then select Documentation


Twitter's First Annual Conference for mobile Developers
San Francisco | October 22nd
MORE >
Build
Monetize
Tap Into
crashlytics
mopub
GNIP
Build high quality apps using the world's most advanced crash reporting and analysis tools
Monetize your app with the world's most powerful mobile ad exchange
Use the Twitter REST API, Streaming API, and Gnip to make better business decisions.
Tweets
Follow
TwitterDev
@TwitterDev
16 Sep
We're building great content for @Flight. Apply to attend & learn what's behind the red lines. twitterflight.com pic.twitter.com/smxZv4mBE7
Retweeted by Twitter API
Tweet to @twitterapi
PRODUCTS
Crashlytics
Answers by Crashlytics
Beta by Crashlytics
RESOURCES
Documentation
Forums
Blog
PROGRAMS
Certified Partner
Events
TOOLS
API Status
API Console
API Terms
Manage Your Apps
Subscribe to Our Newsletter



Documentation

The Twitter Platform connects your website or application with the worldwide conversation happening on Twitter.

Twitter for Websites

Twitter for Websites is a suite of embeddable widgets, buttons, and client-side scripting tools to integrate Twitter and display Tweets on your website or JavaScript application, including the Tweet Button, the Follow Button, Embedded Tweets, and Embedded Timelines.

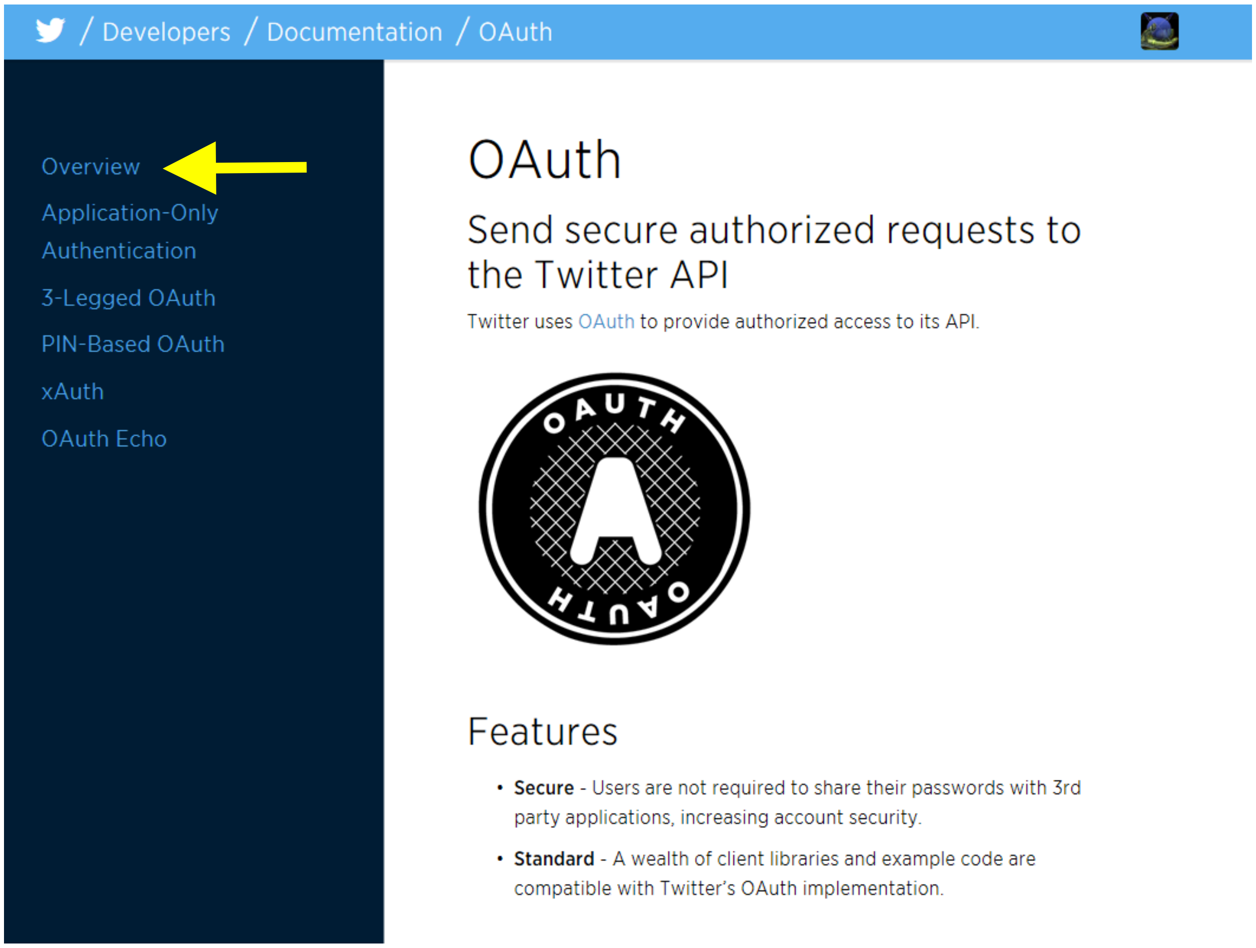
Cards

Twitter Cards display additional content alongside a Tweet for supported links. Highlight a photo, video, or other page summary when your links are shared on Twitter to drive additional traffic to your website, iOS, or Android app.

OAuth

Use our OAuth endpoints to connect users to Twitter and send secure, authorized requests to the Twitter API.

REST APIs


/ Developers / Documentation / OAuth
Overview
Application-Only Authentication
3-Legged OAuth
PIN-Based OAuth
xAuth
OAuth Echo

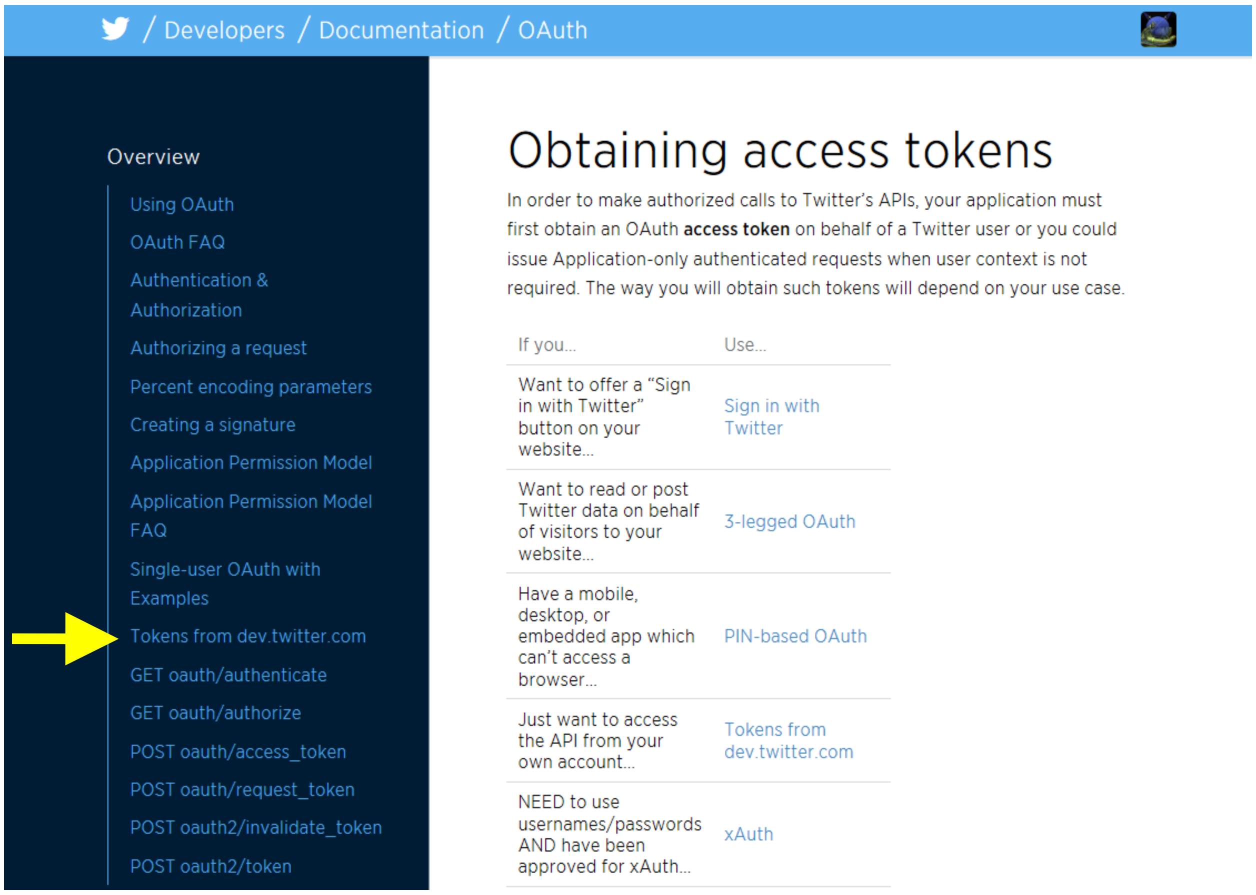
OAuth

Send secure authorized requests to the Twitter API

Twitter uses OAuth to provide authorized access to its API.

Features

- **Secure** - Users are not required to share their passwords with 3rd party applications, increasing account security.
- **Standard** - A wealth of client libraries and example code are compatible with Twitter's OAuth implementation.



/ Developers / Documentation / OAuth

Overview

Using OAuth
OAuth FAQ
Authentication & Authorization
Authorizing a request
Percent encoding parameters
Creating a signature
Application Permission Model
Application Permission Model FAQ
Single-user OAuth with Examples
Tokens from dev.twitter.com
GET oauth/authenticate
GET oauth/authorize
POST oauth/access_token
POST oauth/request_token
POST oauth2/invalidate_token
POST oauth2/token

Obtaining access tokens

In order to make authorized calls to Twitter's APIs, your application must first obtain an OAuth **access token** on behalf of a Twitter user or you could issue Application-only authenticated requests when user context is not required. The way you will obtain such tokens will depend on your use case.

If you...	Use...
Want to offer a "Sign in with Twitter" button on your website...	Sign in with Twitter
Want to read or post Twitter data on behalf of visitors to your website...	3-legged OAuth
Have a mobile, desktop, or embedded app which can't access a browser...	PIN-based OAuth
Just want to access the API from your own account...	Tokens from dev.twitter.com
NEED to use usernames/passwords AND have been approved for xAuth...	xAuth

Overview

- Using OAuth
- OAuth FAQ
- Authentication & Authorization
- Authorizing a request
- Percent encoding parameters
- Creating a signature
- Application Permission Model
- Application Permission Model FAQ
- Single-user OAuth with Examples
- Tokens from dev.twitter.com
- GET oauth/authenticate
- GET oauth/authorize
- POST oauth/access_token
- POST oauth/request_token
- POST oauth2/invalidate_token
- POST oauth2/token

Application-Only Authentication

3-Legged OAuth

Tokens from dev.twitter.com

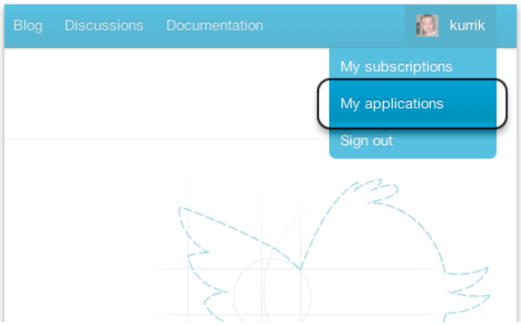
Overview

The dev.twitter.com application control panel offers the ability to generate an OAuth access token for the owner of the application. This is useful if:

- Your application only needs to make requests on behalf of a single user (for example, establishing a connection to the Streaming API).
- You wish to test API functionality from a single account before worrying about the 3-Legged OAuth flow.

Generating a token

Start by visiting “My applications” page by navigating to apps.twitter.com, or hovering over your profile image in the top right hand corner of the site and selecting “My applications”:



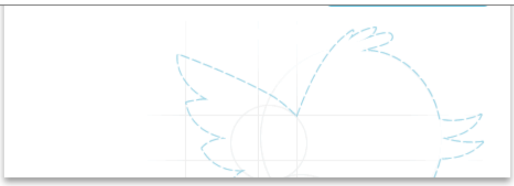
Application-Only Authentication

3-Legged OAuth

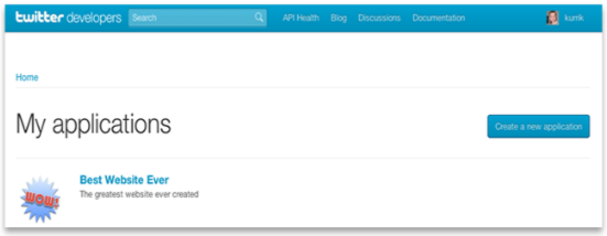
PIN-Based OAuth

xAuth

OAuth Echo



This page contains a list of the applications you have created, along with a button to create a new application. Select the application you wish to generate a token for, and click on its title:

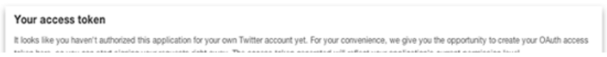


My applications

Best Website Ever

The greatest website ever created

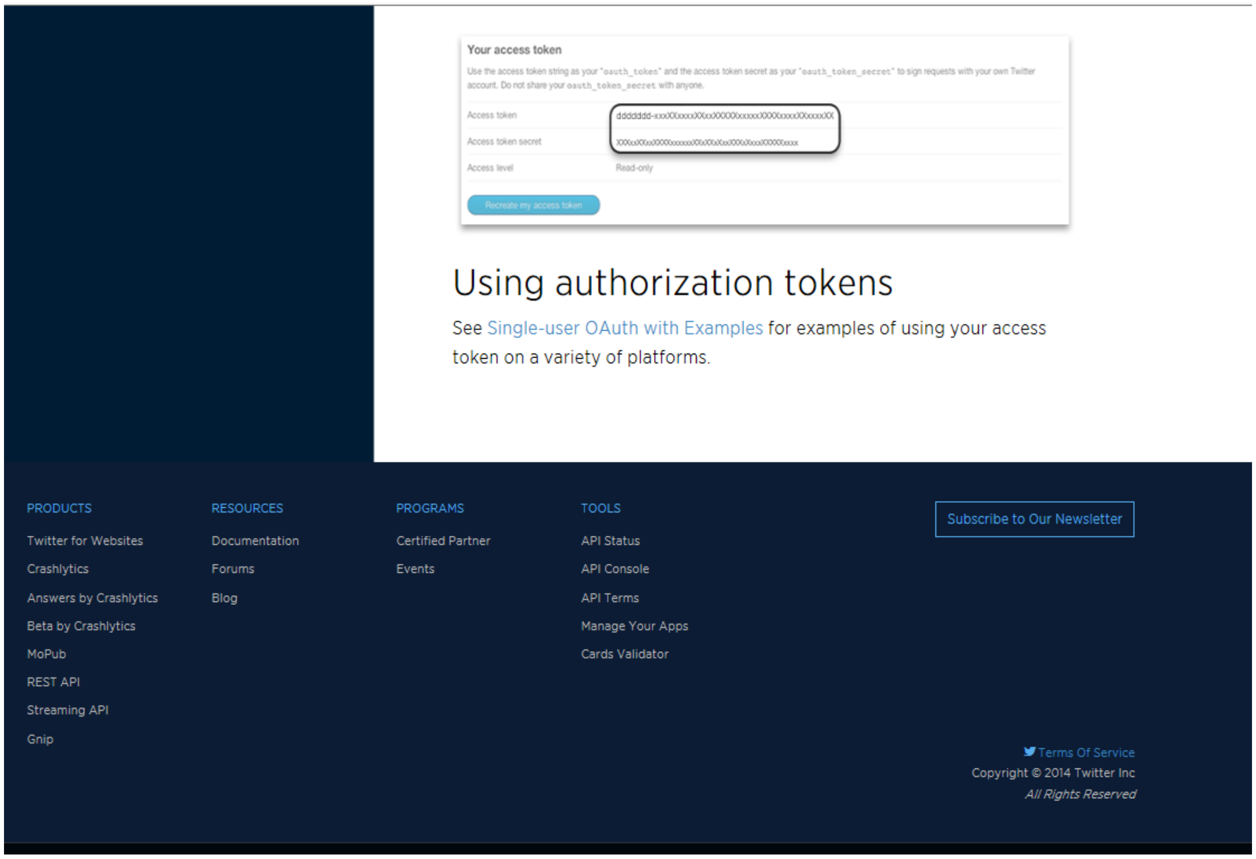
At the bottom of the next page, you will see a section labeled “**your access token**”:



Your access token

It looks like you haven't authorized this application for your own Twitter account yet. For your convenience, we give you the opportunity to create your OAuth access

Click on the “**Create my access token**” button, and an authorized access token and secret will be generated for your account and the current application. These values may be used to authorize requests to the Twitter API.



Your access token

Use the access token string as your "`oauth_token`" and the access token secret as your "`oauth_token_secret`" to sign requests with your own Twitter account. Do not share your `oauth_token_secret` with anyone.

Access token	dddddd-xxxXXxxxxXXxxXXXXXxxxxxXXXXxxxxXXxxxxXX
Access token secret	XXXxxXXxxXXXXxxxxxxXXxXXxXxxXXXxXxxxXXXXXxxxx
Access level	Read-only

Using authorization tokens

See Single-user OAuth with Examples for examples of using your access token on a variety of platforms.

Step 4: Obtain a Twitter Access Token for Your Application... The token you select will depend on what you need to access from Twitter and how you will use the information.

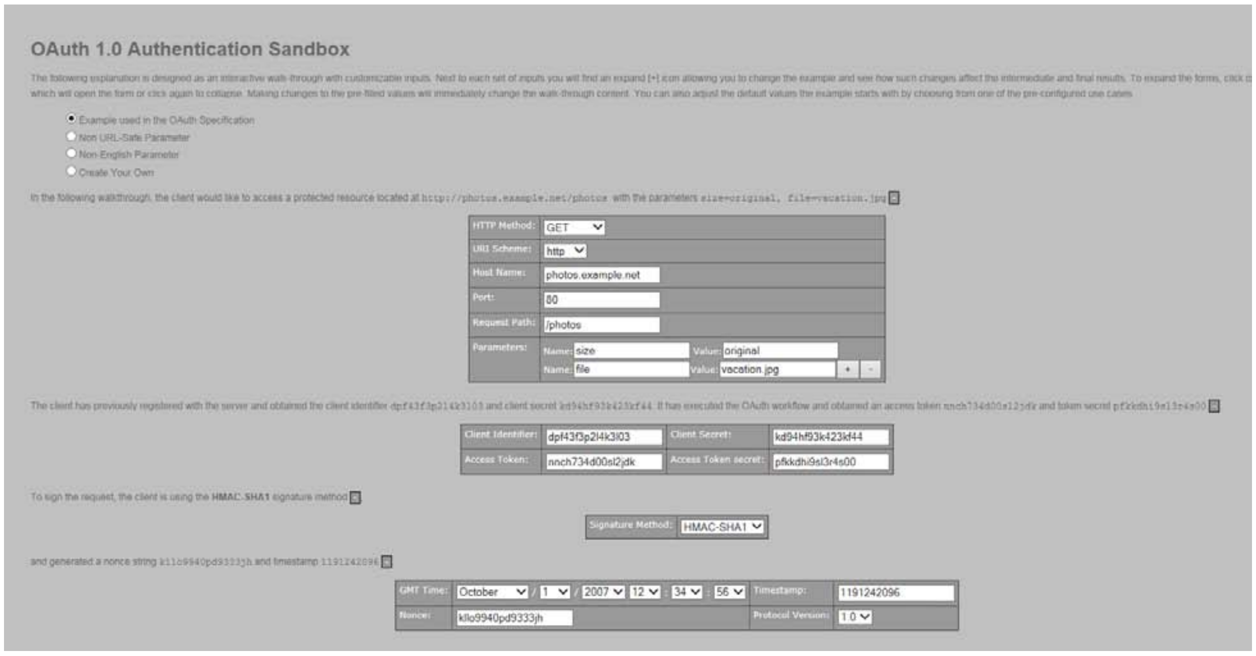
The best way to see this is to visit OAuth Obtaining Access Tokens

After you obtain your token you can now use it in your application to make calls using the Twitter API

Additional Info

Here is another link to a twitter token generator that shows how the process works

http://nouncer.com/oauth/authentication.html



OAuth 1.0 Authentication Sandbox

The following explanation is designed as an interactive walk-through with customizable inputs. Next to each set of inputs you will find an expand [+] icon allowing you to change the example and see how such changes affect the intermediate and final results. To expand the forms, click o which will open the form or click again to collapse. Making changes to the pre-filled values will immediately change the walk-through content. You can also adjust the default values the example starts with by choosing from one of the pre-configured use cases.

- [✓ Example used in the OAuth Specification]
- Non URL-Safe Parameter
- Non-English Parameter
- Create Your Own

In the following walkthrough, the client would like to access a protected resource located at `http://photos.example.net/photos` with the parameters `size=original, file=vacation.jpg`

HTTP Method:	GET
URI Scheme:	http
Host Name:	photos.example.net
Port:	80
Request Path:	/photos
Parameters:	Name: size Value: original Name: file Value: vacation.jpg

The client has previously registered with the server and obtained the client identifier `dpf43f3p2l4k3l03` and client secret `kd94hf93k423kf44`. It has executed the OAuth workflow and obtained an access token `nnch734d00sl2jdk` and token secret `pfkkdhi9sl3r4s00`

Client Identifier:	dpf43f3p2l4k3l03	Client Secret:	kd94hf93k423kf44
Access Token:	nnch734d00sl2jdk	Access Token secret:	pfkkdhi9sl3r4s00

To sign the request, the client is using the **HMAC-SHA1** signature method

Signature Method: HMAC-SHA1

and generated a nonce string `kllo9940pd9333jh` and timestamp `1191242096`

GMT Time:	October / 1 / 2007 12 : 34 : 56	Timestamp:	1191242096
Nonce:	kllo9940pd9333jh	Protocol Version:	1.0