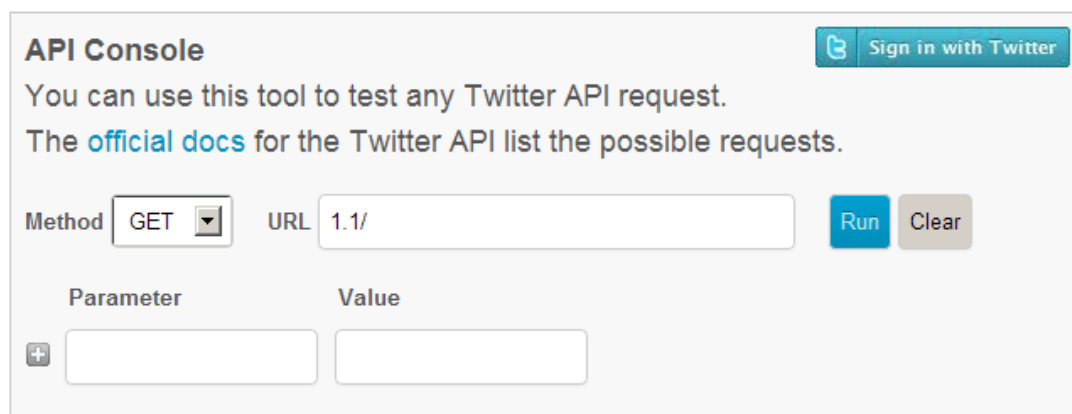


140dev API Console Users Guide



API Console [Sign in with Twitter](#)

You can use this tool to test any Twitter API request.
The [official docs](#) for the Twitter API list the possible requests.

Method: URL:

Parameter	Value
<input type="text"/>	<input type="text"/>

Thanks for using the API Console tool at 140dev.com (<http://140dev.com/twitter-api-console/>). I think you'll find it useful in many ways. Before I had this tool I used to debug Twitter API issues by writing little PHP scripts that called the Twitter API and printed the results. The whole cycle of modifying these scripts for different parameter values and running them with an SSH client was a pain. Even worse was trying to read the response data. I used to copy these responses from the SSH client, and paste them into a text editor to help me figure out the path through one of these response arrays.

So I built this API console. Once I had it for a few days, I realized that I couldn't do API programming without it. If I needed it, others must also. And now it is available for everyone. Enjoy.

Adam Green
<http://140dev.com>
140dev@gmail.com
[@140dev](#)

Requests are made for your Twitter account

The first thing to understand about the API Console is that it only makes requests on behalf of a logged in Twitter user. The requests pass through one of the 140dev servers, but they use the OAuth tokens of someone who has authorized the 140dev API development tools application. So your first step in using the API Console is to click the **Sign in with Twitter button** and authorize this app.

Using the dev.twitter.com docs

The user interface of the API Console is meant to map into the [Twitter API docs](#). It's a straight forward process to take the page for any API request and translate that into inputs for this tool. Let's try it with the [/users/show](#) request, which returns the information for any Twitter account.

The screenshot shows the documentation for the `GET users/show` endpoint. It includes a 'View' button, a 'What links here' button, and a note that it was updated on Mon, 2012-09-10 08:10. The API version is 1.1. The description states that it returns a variety of information about the user specified by the required `user_id` or `screen_name` parameter. The resource URL is `http://api.twitter.com/1.1/users/show.json`. The parameters section lists `user_id` and `screen_name` as required parameters. The resource information table shows that the method is GET, the response format is json, and the resource family is users.

Resource Information	
Rate Limited?	Yes
Requests per rate limit window	180
Authentication	Requires user context
Response Formats	json
HTTP Methods	GET
Resource family	users
Response Object	Users
API Version	v1.1

There are three basic elements of an API request that you need to gather from the documentation: URL, method, and parameters. You only need a portion of the URL for use with the API console. In this case, the URL is shown as: `http://api.twitter.com/1.1/users/show.json`

The part that you need is: `1.1/users/show`. The `1.1` is the version of the API, and the remainder is the name of the API request. The ending of `.json` shown in the docs specifies the format of the response, but since JSON is the default, it isn't needed when you make the request.


The method for a request will be either **GET** or **POST**, based on whether you are GETting information or POSTing changes.

The parameters vary for each API request, and are entered as a pair of parameter name and value. The docs are a little confusing on this subject. For example, both **user_id** and **screen_name** are listed as required, while the description explains that only one of the them is required. Some requests don't need any parameters.

You can fill in these elements of the API request and click **Run** to execute the request. The console will display the header and response portions of the results.

API Console

You can use this tool to test any Twitter API request.
The [official docs](#) for the Twitter API list the possible requests.

@140dev 
Sign Out [Tweet](#) [Print](#)

Method URL

Parameter	Value
<input type="text" value="screen_name"/>	<input type="text" value="justinbieber"/>
<input type="text"/>	<input type="text"/>

API request for PHP and tmhOAuth [?](#)

```
$connection->request('GET',  
    $connection->url('1.1/users/show'),  
    array( 'screen_name' => 'justinbieber'));
```

HTTP Code 200

Headers

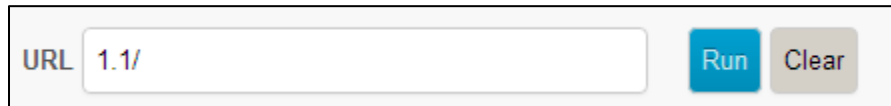
```
Array  
(  
    [HTTP/1.1 200 OK] =>  
    [X-Access-Level] => read-write  
    [Content-Type] => application/json;charset=utf-8  
    [Last-Modified] => Sun, 20 Jan 2013 18:02:33 GMT
```

Response

```
Array  
(  
    [id] => 27260086  
    [id_str] => 27260086  
    [name] => Justin Bieber  
    [screen_name] => justinbieber
```

Defaulting to version 1.1

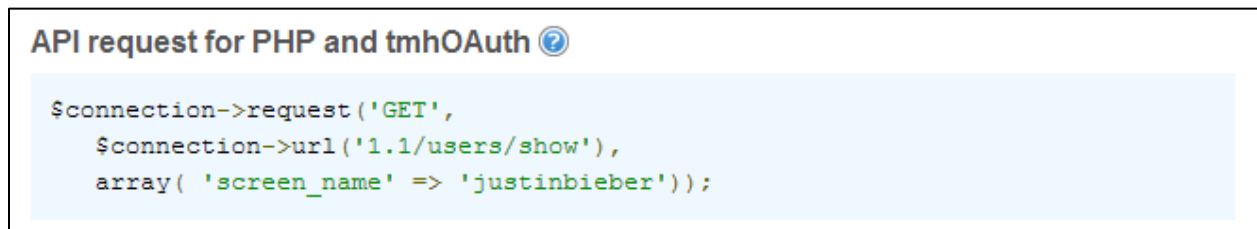
One thing you'll notice when you start using the console is that the URL already contains the version number of **1.1/**. You can change this to a version 1.0 request by modifying the URL field to start with **1/** instead.



The image shows a text input field with the label 'URL' and the value '1.1/'. To the right of the input field are two buttons: a blue 'Run' button and a grey 'Clear' button.

API request for tmhOAuth

You can use the API Console to test requests and then use them with any programming language. I do all my programming in PHP and use the [tmhOAuth](#) library to connect to the API. As a convenience I have the API Console display an API request using the current inputs formatted for tmhOAuth. You copy and paste this directly into your PHP code. If you are unfamiliar with OAuth programming with this library, you can learn how it works with this [free ebook](#).



The image shows a code block titled 'API request for PHP and tmhOAuth' with a help icon. The code is as follows:

```
$connection->request('GET',  
    $connection->url('1.1/users/show'),  
    array( 'screen_name' => 'justinbieber'));
```

Headers

Most people only look at the HTTP code returned in the header of an API response, but there is more useful data there. One thing to keep in mind is that some of these header fields are related to the console app itself, and not specifically your logged in Twitter account. One of these is the access level.

The three rate limit fields, on the other hand, report on your logged in Twitter account in connection with this app. Each app you authorize has a different set of rate limits available for your account.

```
Headers
Array
(
    [HTTP/1.1 200 OK] =>
    [X-Access-Level] => read-write
    [Content-Type] => application/json;charset=utf-8
    [Last-Modified] => Sun, 20 Jan 2013 15:55:22 GMT
    [Expires] => Tue, 31 Mar 1981 05:00:00 GMT
    [Pragma] => no-cache
    [Cache-Control] => no-cache, no-store, must-revalidate, pre-check=0, post-check=0
    [Set-Cookie] => lang=en
    [X-Transaction] => 451a94dalfaf91de
    [X-Frame-Options] => SAMEORIGIN
    [Status] => 200 OK
    [Date] => Sun, 20 Jan 2013 15:55:22 GMT
    [Content-Encoding] => gzip
    [Content-Length] => 958
    [X-Rate-Limit-Limit] => 180
    [X-Rate-Limit-Remaining] => 179
    [X-Rate-Limit-Reset] => 1358698222
    [Server] => tfe
    [] =>
```

Response

The response data is the real value of the API Console. It serves several purposes. As I said at the start, this is invaluable for finding the proper path through the result data structure to a particular value.

```
Response
Array
(
    [id] => 27260086
    [id_str] => 27260086
    [name] => Justin Bieber
    [screen_name] => justinbieber
    [location] => All Around The World
    [description] => #BELIEVE is on ITUNES and in STORES WORLDWIDE! - SO MUCH LOVE FOR THE FANS...you are always there for me and I will always be there for you. MUCH LOVE. thanks
    [url] => http://www.youtube.com/justinbieber
    [entities] => Array
        (
            [url] => Array
                (
                    [urls] => Array
                        (
                            [0] => Array
                                (
                                    [url] => http://www.youtube.com/justinbieber
                                    [expanded_url] =>
                                    [indices] => Array
```

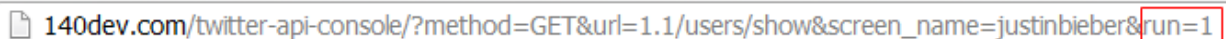
Sharing console URLs

This is my favorite part of the tool. When you run the API Console to execute a request, the arguments are included in the URL of the results page. You can copy this URL and share it with others. When someone else clicks the resulting link in an email or a tweet, the inputs are filled in and the request is automatically run. Even better, the request is run with their own logged in Twitter account. So you don't have to worry about revealing your OAuth tokens when sharing API code. To make this sharing easier, the page has a Tweet button, which opens a tweet box with the current page's URL.

I'm hoping that people will use this feature to document and report on their API work. This should be especially useful when reporting an API problem on the Twitter dev mailing lists. Submitting a URL that will reproduce your request along with a complete header and response will be a much simpler process than having to copy and paste your own script code and the various response strings into an email that reports a problem.

Run parameter

When you run an API request, the page is loaded with a URL that includes a run argument. This tells the page to execute the request.



140dev.com/twitter-api-console/?method=GET&url=1.1/users/show&screen_name=justinbieber&run=1

If you share this URL, anyone clicking it will also cause the request to execute. You can remove this argument before sharing the URL. This means that the page will load with the input fields filled in, and the user can modify these values before clicking the **Run** button.

Free to make mistakes

The input fields for the API Console have no error checking. You can leave the URL or parameter values blank, or use request URLs and parameter names that are invalid. This was done deliberately so you can test different error conditions. The error codes returned by the API tend to be inconsistent at times. It can be useful to see exactly what does get returned with different types of missing or incorrect values. Twitter also changes the response messages for errors frequently, so being able to trigger an error and then see the response can be useful in debugging and writing error handling code.

Useful API calls

The API Console is a convenient way to do quick status checks on your account. You can see what your rate limits are for all possible requests with this request:

http://140dev.com/twitter-api-console/?method=GET&url=1.1/application/rate_limit_status&run=1

You can verify that your account is still valid with this one:

http://140dev.com/twitter-api-console/?method=GET&url=1.1/account/verify_credentials&run=1

No DM access

The only API requests you can't do with the API Console is anything related to the Direct Message system. Twitter is very sensitive about abuse of DMs, making the idea of a public web page that allows DMing a little risky. I decided to avoid any problems in this area by not registering the underlying application for DM access.