Creating a signature

This page explains how to generate an OAuth 1.0a HMAC-SHA1 signature for an HTTP request. This signature will be suitable for passing to the Twitter API as part of an authorized request, as described in authorizing a request.

The request used to demonstrate signing is a POST to https://api.twitter.com/1.1/statuses/update.json. The raw request looks like this:

```
POST /1.1/statuses/update.json?include_entities=true HTTP/1.1
Accept: */*
Connection: close
User-Agent: OAuth gem v0.4.4
Content-Type: application/x-www-form-urlencoded
Content-Length: 76
Host: api.twitter.com

status=Hello%20Ladies%20%2b%20Gentlemen%2c%20a%20signed%20OAuth%20request%21
```

Collecting the request method and URL

To produce a signature, start by determining the HTTP method and URL of the request. These two are known when creating the request, so they are easy to obtain.

The request method will almost always be GET or POST for Twitter API requests.

HTTP Method	POST

The base URL is the URL to which the request is directed, minus any query string or hash parameters. It is important to use the correct protocol here, so make sure that the “https://” portion of the URL matches the actual request sent to the API.

Base URL	https://api.twitter.com/1.1/statuses/update.json

Collecting parameters

Next, gather all of the parameters included in the request. There are two such locations for these additional parameters - the URL (as part of the query string) and the request body. The sample request includes a single parameter in both locations:

```
POST /1.1/statuses/update.json?include_entities=true HTTP/1.1
Accept: */*
Connection: close
User-Agent: OAuth gem v0.4.4
Content-Type: application/x-www-form-urlencoded
Content-Length: 76
Host: api.twitter.com

status=Hello%20Ladies%20%2b%20Gentlemen%2c%20a%20signed%20OAuth%20request%21
```

An HTTP request has parameters that are URL encoded, but you should collect the raw values. In addition to the request parameters, every oauth_* parameter needs to be included in the signature, so collect those too. Here are the parameters from authorizing a request:

status	Hello Ladies + Gentlemen, a signed OAuth request!
include_entities	true
oauth_consumer_key	xvz1evFS4wEEPTGEFPHBog
oauth_nonce	kYjzVBB8Y0ZFabxSWbWovY3uYSQ2pTgmZeNu2VS4cg
oauth_signature_method	HMAC-SHA1
oauth_timestamp	1318622958
oauth_token	370773112-GmHxMAgYyLbNEtIKZeRNFsMKPR9EyMZeS9weJAEb
oauth_version	1.0

These values need to be encoded into a single string, which will be used later on. The process to build the string is very specific:

1. Percent encode every key and value that will be signed.
2. Sort the list of parameters alphabetically [1] by encoded key [2].
3. For each key/value pair:
4. Append the encoded key to the output string.
5. Append the '=' character to the output string.
6. Append the encoded value to the output string.
7. If there are more key/value pairs remaining, append a '&' character to the output string.

[1] The OAuth spec says to sort lexicographically, which is the default alphabetical sort for many libraries.

[2] In the case of two parameters with the same encoded key, the OAuth spec says to continue sorting based on value. However, Twitter does not accept duplicate keys in API requests

Parameter string

The following *parameter string* will be produced by repeating these steps with the parameters collected above:

```
include_entities=true&oauth_consumer_key=xvz1evFS4wEEPTGEFPHBog&oauth_nonce=kYjzVBB8Y0ZFabxSWbWovY3uYSQ2pTgmZeNu2VS
4cg&oauth_signature_method=HMAC-SHA1&oauth_timestamp=1318622958&oauth_token=370773112-
GmHxMAgYyLbNEtIKZeRNFsMKPR9EyMZeS9weJAEb&oauth_version=1.0&status=Hello%20Ladies%20%2B%20Gentlemen%2C%20a%20signed%
20OAuth%20request%21
```

Creating the signature base string

The three values collected so far must be joined to make a single string, from which the signature will be generated. This is called the **signature base string** by the OAuth specification.

To encode the HTTP method, base URL, and parameter string into a single string:

1. Convert the HTTP Method to uppercase and set the output string equal to this value.
2. Append the '&' character to the output string.
3. Percent encode the URL and append it to the output string.
4. Append the '&' character to the output string.
5. Percent encode the parameter string and append it to the output string.

This will produce the following *signature base string*:

```
POST&https%3A%2F%2Fapi.twitter.com%2F1.1%2Fstatuses%2Fupdate.json&include_entities%3Dtrue%26oauth_consumer_key%3Dxv
z1evFS4wEEPTGEFPHBog%26oauth_nonce%3DkYjzVBB8Y0ZFabxSWbWovY3uYSQ2pTgmZeNu2VS4cg%26oauth_signature_method%3DHMAC-
SHA1%26oauth_timestamp%3D1318622958%26oauth_token%3D370773112-
GmHxMAgYyLbNEtIKZeRNFsMKPR9EyMZeS9weJAEb%26oauth_version%3D1.0%26status%3DHello%2520Ladies%2520%252B%2520Gentlemen%
252C%2520a%2520signed%2520OAuth%2520request%2521
```

Make sure to percent encode the parameter string. The signature base string should contain exactly 2 ampersand '&' characters. The percent '%' characters in the parameter string should be encoded as %25 in the signature base string.

Getting a signing key

The last pieces of data to collect are secrets which identify the Twitter app making the request, and the user the request is on behalf of. It is very important to note that these values are incredibly sensitive and should never be shared with anyone.

The value which identifies your app to Twitter is called the **consumer secret** and can be found in the developer portal by viewing the app details page. This will be the same for every request your Twitter app sends.

Consumer secret	kAcSOqF21Fu85e7zjz7ZN2U4ZRhfV3WpwPAoE3Z7kBw

The value which identifies the account your application is acting on behalf of is called the **OAuth token secret**. This value can be obtained in several ways, all of which are described in obtaining access tokens.

OAuth token secret	LswwdoUaIvS8ltyTt5jkRh4J50vUPVVHtR2YPi5kE

Once again, it is very important to keep these values private to your application. If you feel that your values have been compromised, regenerate your tokens (the tokens on this page have been marked as invalid for real requests).

Both of these values need to be combined to form a **signing key** which will be used to generate the signature. The signing key is simply the percent encoded consumer secret, followed by an ampersand character '&', followed by the percent encoded token secret:

Note that there are some flows, such as when obtaining a request token, where the token secret is not yet known. In this case, the signing key should consist of the percent encoded **consumer secret** followed by an ampersand character '&'.

Signing key	kAcSOqF21Fu85e7zjz7ZN2U4ZRhfV3WpwPAoE3Z7kBw&LswwdoUaIvS8ltyTt5jkRh4J50vUPVVHtR2YPi5kE

Calculating the signature

Finally, the signature is calculated by passing the signature base string and signing key to the HMAC-SHA1 hashing algorithm. The details of the algorithm are explained as hash_hmac function.

The output of the HMAC signing function is a binary string. This needs to be base64 encoded to produce the signature string. For example, the output given the base string and signing key given on this page
is 84 2B 52 99 88 7E 88 7602 12 A0 56 AC 4E C2 EE 16 26 B5 49. That value, when converted to base64, is the OAuth signature for this request:

OAuth signature	hCtSmYh+iHYCEqBWrE7C7hYmtUk=

Next steps

- See how to authorize a request
- Learn about percent encoding parameters