API Endpoints

Once you've registered your client it's easy to start requesting data from Instagram.

All endpoints are only accessible via **https** and are located at **api.instagram.com**. For instance: you can retrieve photos with a given hashtag by accessing the following URL with your `access_token` (replace ACCESS-TOKEN with your own):

```
https://api.instagram.com/v1/tags/nofilter/media/recent?access_token=ACCESS_TOKEN
```

The Instagram API requires an `access_token` from authenticated users for each endpoint. We no longer support making requests using just the `client_id`.

Note

The URL examples throughout this documentation use ACCESS-TOKEN as a placeholder. For these examples to work, you need to substitute the value with your own `access_token`. In your application, you should have each user go through an authentication and authorization flow in order to receive a valid `access_token`.

Limits

Please refer to the limits documentation for information on rate limits that apply to API requests.

Deleting Objects

We do our best to have all our URLs be RESTful. Every endpoint (URL) may support one of four different http verbs. GET requests fetch information about an object, POST requests create objects, PUT requests update objects, and finally DELETE requests will delete objects.

Structure

The Envelope

Every response is contained by an envelope. That is, each response has a predictable set of keys with which you can expect to interact:

```
{
    "meta": {
        "code": 200
    },
    "data": {
        ...
    },
    "pagination": {
        "next_url": "...",
        "next_max_id": "13872296"
    }
}
```

META

The meta key is used to communicate extra information about the response to the developer. If all goes well, you'll only ever see a code key with value 200. However, sometimes things go wrong, and in that case you might see a response like:

```
{
    "meta": {
        "error_type": "OAuthException",
        "code": 400,
        "error_message": "..."
    }
}
```

DATA

The data key is the meat of the response. It may be a list or dictionary, but either way this is where you'll find the data you requested.

PAGINATION

Sometimes you just can't get enough. For this reason, we've provided a convenient way to access more data in any request for sequential data. Simply call the url in the next_url parameter and we'll respond with the next set of data.

```
{
    ...
    "pagination": {
        "next_url": "https://api.instagram.com/v1/tags/puppy/media/recent?
access_token=fb2e77d.47a0479900504cb3ab4a1f626d174d2d&max_id=13872296",
        "next_max_id": "13872296"
    }
}
```

On views where pagination is present, we also support the "count" parameter. Simply set this to the number of items you'd like to receive. Note that the default values should be fine for most applications - but if you decide to increase this number there is a maximum value defined on each endpoint.

JSONP

If you're writing an AJAX application, and you'd like to wrap our response with a callback, all you have to do is specify a callback parameter with any API call:

```
https://api.instagram.com/v1/tags/coffee/media/recent?access_token=ACCESS-
TOKEN&callback=callbackFunction
```

Would respond with:

```
callbackFunction({
    ...
});
```