
parselyapi Documentation

Release 0.1

Andrew Montalenti

April 24, 2010

CONTENTS

1	Parse.ly Developer Manual	3
1.1	Why use the Parse.ly API?	3
1.2	Concepts	4
1.3	Application Programming Interface (API)	8
1.4	Algorithms and Technical Background	20
1.5	Contact the Parse.ly Team	21
2	Indices and tables	23

Parse.ly is a service and platform for personalized content recommendations. Parse.ly-enhanced sites connect users with content they'll love through proprietary recommendation algorithms.

For Online Publishers: If you run a content site of any kind – online news site, magazine, or blog – and want to better engage your users with your content, Parse.ly can help. We parse and analyze your content using sophisticated natural language processing technology and analyze reader behavior and engagement with that content. We then build up rich snapshots of a user's interests – which we call profiles – that can be used to power personalized, highly-targeted content recommendations. Our system caches your content and the user data, offering a fully-hosted, plug-and-play integration. Think Google Analytics meets Amazon Product recommendations, tailored to the unique needs of online publishers.

For Techies and Mashup Authors: Want to tap into Parse.ly's powerful real-time news and blog article processing infrastructure and recommendation algorithms for your own online mashup? Have a great idea for a mobile news iPhone/Android app, or an Adobe Air desktop notification system for real-time news updates? Parse.ly's API gives you access to the best content the web has to offer – from over 120K sources – but also allows you to personalize the results based on individual user interests.

PARSE.LY DEVELOPER MANUAL

Note

You can read this document by navigating the contents listed below. However, if you prefer, we also have a [single HTML page with the entire manual](#) and a [gorgeous, printable PDF](#) you can use, instead.

1.1 Why use the Parse.ly API?

1.1.1 Problem

Reader engagement for online content sites has dropped drastically over the past years. Users still have to wade through pages of irrelevant content to find the information that resonates with them.

1.1.2 Solution

Eric Schmidt, CEO of Google, recently said, “At its best, the online version of a newspaper should learn from the information I’m giving it to automatically send me stories and photos that will interest me.”

Parse.ly processes existing feeds and user behavior in real-time. Rather than treating every visitor like a stranger, our personalized recommendations meet Eric Schmidt’s vision of the “newspaper of the future” – except Parse.ly is available today.

1.1.3 Compared to competitors

Many publisher tools focus on increasing advertising inventory (e.g. through Daylife topic pages) or “related content” (e.g. through the Evri widget). Parse.ly offers a different value proposition altogether. Parse.ly is a user-centric – not content-centric – approach, and takes its cue from systems that work in other verticals: Pandora.com for music, Amazon.com for products, Netflix.com for movies.

1.1.4 Integration approach

The Parse.ly service can currently be integrated onto existing websites in a number of ways:

1. Since the service is in early beta, you can [contact us](#) and we will guide you through most the stages of integration and provide you with an API key.
2. We offer a [whitelabel widget](#) that can be easily embedded in blogs, digital media sites, and basically any other website.

3. We offer a full *RESTful Application Programming Interface (API)*, with a number of language bindings (*Python*, *JavaScript*) as well as fully-documented request and response formats and *examples*.

1.2 Concepts

1.2.1 Basics

API Keys

Access to the Parse.ly API is provided to you when you receive an API Key from our developers. An API Key is a unique string your application uses to identify itself to Parse.ly's servers. You can [contact us](#) for an API Key.

Authentication

To comply with the emerging OAuth web standard, all requests to the Parse.ly API must use OAuth for authentication. However, because the Parse.ly API does not require any user to authorize access to personal resources, we use a form of OAuth called 2-legged OAuth, aka "Signed Fetch". More information can be found in our [OAuth Background](#) document.

Implementation

OAuth's primary reason for existence is to provide an open standard for *cross-service API access delegation*. For example, if a user needs to give Facebook.com authorization to download information about that user's favorite music from Last.fm, Facebook.com would request authorization tokens from Last.fm via the OAuth protocol. The user would then authorize Facebook's access to Last.fm via an interactive web-based screen. This prevents the user from having to share his Last.fm logon credentials with Facebook.com.

This three-step procedure is also known as 3-legged OAuth, and is often lovingly referred to as "the OAuth dance". The 2-legged version used by Parse.ly skips the interactive step. The differences are well-described in the presentation, [Wherefore Art Thou, OAuth?](#)

Also, OAuth should not be confused with OpenID, a complementary but very different standard. Whereas OpenID provides a "single sign-on" mechanism to users for numerous web-based services, OAuth is a protocol that is used to authorize access to data associated with user accounts.

HTTP, REST and JSON

The Parse.ly API is exposed as a set of HTTP resources, following the RESTful pattern of API design (see [REST](#)). Each of these resources have various operations supported, which are mapped to URLs and HTTP methods. All request and response formats use JavaScript Object Notation, aka JSON (see [JSON](#)). If you wish not to use the direct HTTP API, we also provide language bindings, described later on in this documentation.

Note

For simplicity, Parse.ly only provides JSON as an output format for its API. If other formats – like YAML, XML, Protocol Buffers, or Thrift – would make more sense for your application, [let us know](#).

1.2.2 Account Management

The purpose of integrating the Parse.ly API into your site is to offer content recommendations to *your* users, based on the interactions they have with *your* content. Parse.ly's servers stores and analyzes a cached copy of content from your site. They also store information about your site visitors that allows the system to personalize that content to your users' tastes and interests.

Therefore, we offer an abstraction both for managing the sources of content that Parse.ly will cache/analyze and the profiles of your visitors/users that will power our recommendations.

User Profiles

A *UserProfile* resource is associated with a single unique user. These are generally used following one or both of the following patterns:

- *Identifiable Profile*: If your site already features a user account system, you can integrate *UserProfile* with identifiable user accounts in your site. For example, if you have a user named “John Smith” with the user ID “john.smith”, you can link that account with a corresponding Parse.ly *UserProfile*, and ensure his data follows him whenever he is logged in to your site.
- *Anonymous Profile*: If you want to offer Parse.ly recommendations to anonymous visitors, you need to utilize our *JavaScript Tracker*, which identifies anonymous visitors in a way similar to systems like Google Analytics. Our tracker will automatically create *UserProfile* resources that correspond to your unique visitors, and use metrics from their browsing experience to power content recommendations, without requiring the user to log in at all.

Note: though the JavaScript Tracker is required for *Anonymous Profiles*, we also encourage you to use the tracker with *Identifiable Profiles*, since this will enhance the data Parse.ly collects about your users and allow it to make better recommendations. Only advanced API users tend to eschew the tracker altogether; for example, when a *UserProfile* can be populated with reading history/behavioral data from an existing source.

UserProfile resources can also include personal information used to enhance or otherwise link that profile with other information sources. We refer to this as *Profile Metadata*. However, the main purpose of a *UserProfile* is to:

- Capture any explicit interests (e.g., topics, people, events) the user cares about
- Capture data about the user’s interactions with content on your site

As a Parse.ly API user, you own this data. But, you derive value from it primarily by using our *Query API*, described later in this document. That is the API that allows you to recommend articles to specific users based on their reading behavior and tastes.

Interests

Interest resources specify topics that resonate with a user. These can be programmatically generated to populate the *UserProfile* with information about a user’s interests. Users with largely overlapping *Interest* sets are given similar recommendations.

Interest resources also include a **rank**, that allows you to specify which interests are more important or less important to a user. These influence results ordering in the *Query API*.

Data Sources

In order for Parse.ly to recommend content, it needs a *DataSource* resource to tell it from where to find that content. As a user of our API, you can configure *FeedURL* resources as part of your data sources, and Parse.ly will automatically monitor that feed (expected in RSS/Atom format). Updates are delivered to Parse.ly in near-real-time.

Implementation

How does Parse.ly do near-real-time delivery of content from your RSS/Atom feeds? Parse.ly embraces an emerging standard known as the [PubSubHubbub protocol](#). We integrate with our PubSubHubbub provider – currently, [Superfeedr](#) – via the [XMPP / XEP-0060](#) standard. In fact, one of our engineers has open sourced our Superfeedr XMPP wrapper for Python; check out the [sfpy project](#).

You can also organize your Data Sources by assigning *tags* to them, which can be used in the *Query API* for filtering data returned by Parse.ly.

Channels

Parse.ly's API offers a few *DataSource* instances configured out of the box; these go by the name *Channel*. Two *Channel* resources are available by default:

- **web_wide_news**: access to over 4000 mainstream news sources
- **web_wide_blogs**: access to over 120K blog sources

If your primary purpose in using our API is to increase page views of your own content, then use of our pre-configured *Channel* resources may not be of much use to you. However, if you are building a system that pulls together content from around the web to recommend to your users – for example, if you are providing a page for career advice and want to recommend articles to the user from around the web based on their career interests – then use of these channels is essential and saves you the work of setting up your own RSS/Atom feeds to monitor for high-quality content.

Note

We have been experimenting with offering some different channels to our API users aside from news and blog content. Here are some ideas we have tossed around.

- **status_updates**: pull status updates from online social networks
- **business_reviews**: reviews of businesses (e.g. restaurant, shop), written by users and editors on top online review sites
- **videos**: videos and captions/summaries taken from top user-uploaded video sites
- **images**: images and captions/summaries taken from online image providers

We have also started organizing feeds from our *web_wide_news* and *web_wide_blogs* channels into categories. Here are some of the categories we've thought about:

- **cat_technology**: technology sources
- **cat_business**: business and financial news sources
- **cat_politics**: political coverage and commentary
- **cat_entertainment**: arts and entertainment
- **cat_celebrity**: celebrity news and gossip

Feel free to [contact us](#) if any of these channels (or others that you think of) would be of particular use to you.

1.2.3 Queries

Once you have your resources set up for *DataSource*, *UserProfile*, and *Interest*, you are ready to get to the core value of our API: retrieving content recommendations via queries.

Queries are executed against a single *UserProfile*, and return personalized content recommendations based on that profile's interests and sources. Our recommendations are powered by a proprietary combination of naive Bayesian inference and collaborative filtering (see [Algorithms and Technical Background](#)). Results are **scored** based on how closely the article matches that user profile's interests and reading behavior. The higher the score, the more likely that content will resonate with that user, and thus the more likely that user is to click on that article and enjoy reading it.

The result of a Query API call is a paginated list of *Item* resources, where each *Item* corresponds roughly to an article, with fields common to RSS/Atom feeds, like **title** and **summary** and **link**. An *Item* also includes fields that expose Parse.ly's analysis results, like **score_explanation**.

You are free to display the results from our API any way you like, subject to our *Terms of Use*. We also provide a *Whitelabel JavaScript Widget* which can be used for standard integration use cases.

Our standard result set is sorted by score and recency, however, other sorting orders and search methods are available. For example, we make it possible to do **full text searching** and **dated queries**.

1.2.4 Reading Behavior

As users interact with content on your site, they are giving you valuable information about what articles, topics, and areas of interest resonate with them. Parse.ly allows you to tap into this valuable user data to connect your users with content they'll love.

However, in order for our system to work properly, we need to be able to track what a user does with your content and analyze that content for clues about the user's interests. We offer two forms of informing our system about your user's reading behavior, known as *explicit actions* and *implicit actions*.

- **Implicit:** by installing the *JavaScript Tracker*, our system monitors what parts of your site the user is visiting, what links they are clicking on, and how long they are spending on each piece of content. We analyze this content to build up statistical text profiles for each user (see our *algorithms*). We use these metrics and data to power a model of the user's interest based on their *implicit behavior* on your site. Without the user even realizing he is using a content recommendation system, his past behavior on your site is creating virtuous circles of valid recommendations / positive browsing experiences from the Parse.ly API. Our recommendations also downplay content that would not resonate with that user and avoid boring and unengaging browsing experiences.
- **Explicit:** if you use our *Whitelabel JavaScript Widget* in its advanced form, or if you want to build a specialized reading interface powered by Parse.ly recommendations, you will prefer our explicit reading behavior model. In this model, you actually make API calls to Parse.ly to tell it what a user has done with a piece of content on your site. The actions currently available are: **Starred**, **Shared**, **Marked Read**, **Marked Unread**, **Archived**, and **Deleted**. These actions act as a form of *user training*, but are meant to be seamlessly integrated into your site.

Our users often mix the *implicit* and *explicit* models for best results, but good results can often be achieved with the *implicit* model alone. We are constantly improving each to make it work even better.

1.2.5 Real-Time Updates

Parse.ly's backend infrastructure is powered by message queues and "real-time" protocols like XMPP and PubSubHubbub. We therefore process data in "near-real-time" as that data flows into our system. An obvious use case for Parse.ly's API is in real-time applications, where updates are delivered to your client as they are made available rather than you having to poll our service for new information.

However, at this time, our API does not expose a real-time interface. Every query to the Parse.ly API gives back a JSON document which represents a "point-in-time" snapshot of the latest data we have for that user.

We are exploring different avenues for implementing a real-time interface, for example by offering an XMPP API ourselves, or by offering a PubSubHubbub hub. If the real-time use case is particularly important to you, If you are interested in this option and would like to sponsor our development of a real-time content recommendations API, please *contact us*!

1.3 Application Programming Interface (API)

1.3.1 Background

OAuth

Parse.ly uses 2-legged OAuth to authenticate API users, so you will a key and secret to make API requests. Every request you make must contain this information. We look for the following information in the *Authorization* part of the header of all of your requests:

```
{
  'oauth_consumer_key': key,
  'oauth_timestamp': current_timestamp(),
  'oauth_nonce': nonce,
  'oauth_version': 1.0,
  'oauth_signature_method': 'HMAC-SHA1',
  'oauth_signature': signature
  'xoauth_requestor_id': profile_id,
}
```

Two-Legged OAuth with Python

Not all OAuth client libraries have the same degree of support for 2-legged OAuth. Our Python client library includes a [special module](#) to implement 2-legged OAuth. Here's an example of getting our Parse.lyClient connected to our server with the OAuth secret and key:

```
client = parsely.Parse.lyClient(
    "DgB6Dcpzju2hX6xTwU",
    "CqaDTFAkneDzZTre9TgCzd34JANKZQTm",
    "api.parse.ly")
```

Meaning of xoauth_requestor_id

One of the OAuth header keys that is required on almost all requests is *xoauth_requestor_id*. This is the identifier of the *UserProfile* instance for the data being requested. Not all OAuth client libraries support this key out of the box; *let us know* if you run into issues with it.

Setting up an oauth_proxy

In order to ergonomically test our API (e.g. using your browser and using command-line tools like *curl*), you can set up an OAuth Proxy. An OAuth proxy simply receives HTTP requests and forwards them to another HTTP service, while signing every request with the proper OAuth header entries (as described in [OAuth](#)).

Luckily, Seth Fitzsimmons has written a handy little tool that does just that. You can find the tool at the [oauth_proxy github repo](#) or read [his article](#) about the tool.

Implementation

The *oauth_proxy* tool is implemented in Python as a plugin for the Twisted framework. Therefore, it requires that you have both a Python interpreter installed and a recent version of the Twisted framework, installable on most systems using *easy_install twisted*.

Once you have this tool, simply set it up with the consumer key and shared secret we provided to you via e-mail. If you don't have an API Key yet, [contact us](#) to get one.

Since we use two-legged OAuth, you only need to run the following:

```
$ cd oauth-proxy
$ twistd -n oauth_proxy
    --consumer-key <consumer key>
    --consumer-secret <consumer secret>
```

Then, you can make requests to the Parse.ly API using *curl* as follows:

```
$ curl -x localhost:8001 http://api.parse.ly/p/user/items/current
```

which should return data in JSON format.

Note

Currently, the Parse.ly API requires the parameter *oauth_requestor_id* to be included in the HTTP *Authorization* header of all requests that access *UserProfile* data. Unfortunately, *oauth_proxy* doesn't support a pass-through for this argument. We've put up an alternate version of *oauth_proxy* that allows the pass-through of this parameter here: *not yet available*

If you run into issues getting a proper response from our server, feel free to [contact us](#) for help.

RESTful APIs

Principles

These are the principles that guided our API design:

- **Aim to be RESTful:** This has a controversial definition within the API community, but we followed some of the guidelines set forth in some of the REST resources listed here.
- **Documented:** We should document our API and interchange formats in a way that makes it easy for our users to find what they're looking for and make the requests and interactions that they need to get their work done.
- **Meaningful Error States:** When there are problems finding requested data or if a service is down, we should have some well-established error states that can be handled gracefully by our clients.
- **JSON as an interchange format:** JSON is a lightweight and human-readable format. JSON's only real drawback vs. something like XML is that there is no good way to define a JSON "schema" to document the formats formally. There is a [jsonschema project](#) to do this, but it's still early days for that. So, we will just substitute a good schema for good documentation.
- **Tested:** The API should be tested, and, once it leaves "beta" state, it should be versioned and have a high degree of reliability and consistency.
- **Language Bindings Included:** RESTful API design really defines our wire format and mechanism for interacting with our server. But we should provide language bindings for various platforms to ease the ability to prototype applications atop the Parse.ly REST API.

Design Guidelines

Many of our design guidelines came from the excellent book by Leonard Richardson and Sam Ruby, *RESTful Web Services*, published by O'Reilly and the article [A Brief Intro to REST](#).

We highly recommend you read that article and [pick up a copy of the book](#) if you ever need to design a service of your own!

Give every “thing” an ID From “A Brief Intro to REST”:

Use URIs to identify everything that merits being identifiable, specifically, all of the “high-level” resources that your application provides, whether they represent individual items, collections of items, virtual and physical objects, or computation results.

Link things together This doesn’t just mean that you actually have links (which is a corollary of “everything has an ID”), but that when you return an object that references another object, you use the URL (link) to connect the two. “Use links to refer to identifiable things (resources) **wherever** possible.”

Use standard methods What REST does is provide a “metamodel” for all the “things” in your web application. You could think of this as a widely-used base interface called “Resource”. If you were to model this in Java. you’d have e.g.

```
interface Resource {
    Resource (URI u);
    Response get ();
    Response post (Request r);
    Response put (Request r);
    Response delete ();
}
```

Though things which implement the *Resource* interface can have a richer set of operations and behaviors, we should have reasonable actions that correspond to the above 4 operations. If we can map everything into those 4 (which sometimes correspond to the four actions in the CRUD model), all the better.

Communicate statelessly This quote summarizes this best:

“First of all, it’s important to stress that although REST includes the idea of statelessness, this does not mean that an application that exposes its functionality cannot have state – in fact, this would render the whole approach pretty useless in most scenarios. REST mandates that state be either turned into resource state, or kept on the client.”

The way the Parse.ly team interprets this guideline is to “prefer stateless communication, otherwise, idempotence, otherwise, *documentation!*”. Obviously, you can’t make things stateless, but making them idempotent is probably a good idea, and if you can’t at least make it idempotent, then we better have documented it!

Guarantee Idempotence Idempotence should be expected by our clients for many of our HTTP methods:

“GET is idempotent – if you issue a GET request and don’t get a result, you might not know whether your request never reached its destination or the response got lost on its way back to you. The idempotence guarantee means you can simply issue the request again. Idempotence is also guaranteed for PUT (which basically means “update this resource with this data, or create it at this URI if it’s not there already”) and for DELETE (which you can simply try again and again until you get a result – deleting something that’s not there is not a problem). POST, which usually means “create a new resource”, can also be used to invoke arbitrary processing and thus is neither safe nor idempotent.”

Multiple “Endpoints” We assign unique URLs to user profiles, articles, sources, and everything else in our system. This follows the practice described here:

“In a RESTful approach, an application might add a few million customer URIs to the Web; if it’s designed the same way applications have been designed in CORBA times, its contribution usually is a single “endpoint” – comparable to a very small door that provides entry to a universe of resource only for those who have the key.”

Bibliography

This API was designed taking the following documents into consideration:

- A Brief Intro to REST
- Atlassian REST API Design Guidelines
- REST Anti-Patterns
- REST for the rest of us
- How I Explained REST to my Wife

JavaScript Object Notation (JSON)

What is JSON?

From <http://json.org>:

JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. It is based on a subset of the JavaScript Programming Language ...

Parse.ly JSON Usage

The Parse.ly API uses some standard JSON message formats for all of its request and response documents. We also follow some conventions that make our documents easier to produce and consume. These are described in more detail below:

Metadata Structure for JSON In order to allow JSON responses to contain both a data payload and a bit of meta-data about that payload, we use the following structure for all JSON generated from our API:

```
{
  "meta": { ... }
  "data": { ... }
}
```

The “meta” property contains all metadata, for example related to caching, paging, performance indicators, or auxiliary/warning messages from the server. For example, here is a sample JSON response for a user’s details that includes pagination details in meta:

```
{
  "meta": {
    "paging": {
      "totalItems": 59642,

```

```
    "pageNumber": 1,
    "startIndex": 0,
    "limit": 125
  },
  "type": "list<item>"
},
"data":
  "articles": [
    {
      "title": "Parse.ly releases its Publisher Platform and API",
      "score": 10.0,
      "signature_hash": "cbe668d4b7e2545cf4890ce1cf9bb47b",
      "updated": "2010-01-24 19:01:57",
      "unique_id": "LNqp" //, ...
    } //, ...
  ]
}
```

This way, direct properties from this payload could be accessed like this:

```
var response = getItemsResponse();
var articles = response.data.articles;
var meta = response.meta;

assert meta.type == "list<item>";

for (var i = 0; i < articles.length; i++) {
  var article = articles[i];
  // do something with article
}
```

Paging Metadata The purpose of paging metadata is to make it possible to return paginated results. Think of a Google search results screen. What important information do you need for pagination to work properly, with minimal coding on the client-side?

Using the “item query” example from above, we had a *list<item>* we could page through. The current page is specified as *paging.pageNumber*. A frontend might use the information as follows:

```
var response = getItemsResponse()
var meta = response.meta;
var articles = response.data.articles;
assert meta.type == "list<item>";

var resultsTemplate = Ext.XTemplate(
  "<div id='resultsNavigator'"
  "Pg {pageNumber}: ",
  "Showing {limit} articles out of ",
  "{totalItems} articles total"
  "</div>")

resultsTemplate.overwrite(meta.paging, Ext.get("results"));
```

Note

This example happens to use the *Ext.XTemplate* and *Ext.get* functions from the ExtJS library, but this could be done with any JavaScript library. This is just an example.

Date Representation We need to ensure that all dates that are represented on the wire meet the following criteria:

- Serialized into UTC timezone, rather than server local time.
- Deserialized from UTC and then timezone-adjusted based on user preferences.
- Represent up to millisecond precision.
- Can easily be parsed and processed into *datetime* objects on the Python side, into *Date* objects on the Javascript side, and with other *DateTime* abstractions on other platforms.

To this end, we output all our dates using the [ISO8601](#) date format, with no time zone information included. We don't include time zone information because all of our datetimes are assumed to be in Coordinated Universal Time (UTC).

1.3.2 API Specification

API Reference

The root URL for all Parse.ly API requests is:

```
http://api.parse.ly/p
```

If you try to open <http://api.parse.ly> in a browser, you're just going to hit a landing page. Each of our REST resources (as described from a high level in our [Concepts](#) document) is exposed as a URL below this root.

Exposed URLs

To keep things nice and organized, we have identified all the URLs exposed by our API in terms of a lightweight grammar.

Note

This grammar uses a subset of Perl, including string literal (“...”), string concatenation (.), and regex literal syntax (*/.../*). The named capturing groups use the PCRE syntax of *(?P<param>...)*, where ... is a regular expression and *param* is the name of the capturing group. These named capture groups are distinct parameters passed along to our API.

The following are all the URLs exposed by the API.

```
# item query
"/user/items/" . /(?P<page_type>current|archived|deleted)/
"/user/items/matching/" . /(?P<query>[\w\d\+]+)/
"/user/items/dated/" . /(?P<from_date>[\d\w\-\:]+)/ . /(?P<to_date>[\d\w\-\:]*)/
"/user/items/status/" . /(?P<status_type>read|starred|archived|deleted)/
"/user/items/" . /(?P<signature_hash>[A-Za-z0-9]+)/ . "/status"

# channel query
"/channels/list/"

# data source management
"/user/source/create/"
"/user/source/list/"
"/user/source/update/" . /(?P<source_id>\d+)/
"/user/source/delete/" . /(?P<source_id>\d+)/

# interest management
"/user/interest/create/"
```

```
"/user/interest/list/"
"/user/interest/update/" . /(?P<interest_id>\d+)/
"/user/interest/delete/" . /(?P<interest_id>\d+)/

# profile management
"/user/profile/create/"
"/user/profile/list/"
"/user/profile/update/" . /(?P<profile_id>\d+)/
"/user/profile/delete/" . /(?P<profile_id>\d+)/
```

Testing and Poking Around

In order to test the Parse.ly API, the trickiest thing is to make sure that all of your HTTP requests are signed using the OAuth protocol. This is described in full detail in our *OAuth* background document; there is also a section there on setting up a proxy using the *oauth_proxy* tool. The examples of testing below assume that an *oauth_proxy* is already set up on your local development machine.

Testing with curl A free and easy-to-use client-side library for HTTP requests and URL transfers is provided in most systems in the form of the *curl* executable (aka cURL).

The main argument that you need in order to make curl work with *oauth_proxy* is the *-x* or *-proxy* argument. In the standard *oauth_proxy* configuration described in *OAuth*, the proxy host is *localhost:8001*.

A basic request for all user profiles available ...

Testing with Firefox

Testing with RESTClient

UserProfile Operations

List Output all UserProfile resources associated with your API key:

```
>>>> GET("/user/profile/list")
out> [
  {
    'interests': [{u'aliases': u'',
                  u'id': 7,
                  u'qualifiers': u'',
                  u'rank': 2,
                  u'term': u'!apple'}]
    'user_profile': {u'api_user_id': 1,
                    u'email': u'nnw345y@not.com',
                    u'id': 2,
                    u'name': u'123nnn33',
                    u'user_id': 3}
  }
]
```

Create Create a new UserProfile resource and return its auto-generated identifier:

```
>>>> POST({
    "user_profile":{"email": "nnw345y@not.com", "name": "123nnn33"},
    "interests": [{"term": "internet", "alias": "ewewe", "rank":6}
  })
out> 1234
```

Update Update a UserProfile, given a profile identifier. PUT data should be valid JSON. Example:

```
>>>> PUT("/user/profile/update/1234",
  {
    "user_profile":{"email": "nnw345y@not.com", "name": "123nnn33"},
    "interests": [{"term": "internet", "alias": "ewewe", "rank":6}
  })
out> 1234
```

Delete Delete a UserProfile, given a profile identifier. Example:

```
>>>> DELETE /user/profile/delete/<profile_id>
out> GONE
```

API Detailed Examples

1.3.3 Language Bindings

Python Binding (parselyclient)

JavaScript Binding (parsely-js)

1.3.4 Integrations

Whitelabel JavaScript Widget

JavaScript Tracker

What does the Tracker do?

Inserting the tracker on your site

Optimizing tracker placement

<http://mrcoles.com/blog/how-tracking-scripts-affect-page-loads/>

1.3.5 Licensing Information

Premium Licensing

Legal Summary

Parse.ly API Terms of Use

Thank you for using the Parse.ly, Inc. (“Parse.ly”) application programming interfaces (“Parse.ly APIs”). By using Parse.ly APIs, you agree to these terms of use (the “Terms of Use”) and the Parse.ly Terms of Service (“TOS”). The TOS can be found at: <http://blog.cogtree.com/terms-of-service/>

BY ACCEPTING THESE TERMS OF USE, OR BY USING OR ACCESSING ANY PORTION OF THE PARSE.LY APIs, YOU IRREVOCABLY AGREE TO THESE TERMS OF USE, AND YOU REPRESENT AND WARRANT THAT YOU HAVE ALL AUTHORITY NECESSARY TO BIND YOURSELF (AND, IF YOU ARE EMPLOYED BY OR OTHERWISE REPRESENT ANY CORPORATION OR OTHER LEGAL ENTITY THAT WISHES TO USE THE PARSE.LY APIs, THAT ENTITY) TO THESE TERMS OF USE. IF YOU DO NOT AGREE TO THESE TERMS OF USE, YOU MAY NOT USE THE PARSE.LY APIS.

“You” means the individual person accessing or using the Parse.ly APIs on his or her own behalf; or, if the APIs or Documentation are accessed or used on behalf of an organization, corporation or other legal entity, “you” means such legal entity.

Parse.ly reserves the right to update and change, from time to time, these Terms of Use and all documents incorporated by reference. You can always find the most recent version of these Terms of Use at <http://parse.ly/api/termsfuse.html>. Parse.ly may change these Terms of Use by posting a new version without notice to you. Use of the Parse.ly APIs after such change constitutes acceptance of such changes.

In the event of any inconsistency between these Terms of Use and the TOS, these Terms of Use control.

1. Licensed Uses and Restrictions

Parse.ly APIs are owned by Parse.ly and are licensed to you on a worldwide (except as limited below), non-exclusive, non-sublicenseable basis on the terms and conditions set forth herein. These Terms of Use define legal use of the Parse.ly APIs, all updates, revisions, substitutions, and any copies of the Parse.ly APIs made by or for you. All rights not expressly granted to you are reserved by Parse.ly.

1. Subject to the restrictions set forth in these Terms of Use, you may use the Parse.ly APIs and any updates provided by Parse.ly (in its sole discretion) solely to interface with Parse.ly’s proprietary content recommendation service (the “Parse.ly Service”). Your license to the Parse.ly APIs under these Terms of Use continues until it is terminated by either party. You may terminate the license by discontinuing use of all or any of the Parse.ly APIs. Parse.ly may terminate the license at any time for any reason. Parse.ly may make changes, upgrades, or discontinue all or any portion of any Parse.ly API at any time for any reason. These Terms of Use terminate automatically if (i) you violate any term of these Terms of Use, (ii) Parse.ly publicly posts a written notice of termination on Parse.ly’s Web site, (iii) Parse.ly sends a written notice of termination to you, or (iv) Parse.ly ceases providing access to the Parse.ly APIs to you. Upon the expiration or any termination of these Terms of Service, the license granted to you will terminate and you, at your expense, will promptly return all copies of the API and all Confidential Information in your possession to Parse.ly. The provisions of Sections 2, 3, 7, 8, 10, and 12 shall survive termination or expiration of this Agreement for any reason.
2. Your application may make automated calls or other data requests to or through the Parse.ly network (“Calls”). Parse.ly may at any time, and over any given period of time, limit the number of Calls

you may send to the Parse.ly network, or prohibit any application created by you from sending Calls to the Parse.ly network, as Parse.ly deems appropriate in its sole discretion.

3. If your product or service uses or is based upon the Parse.ly APIs, then you will comply with any Parse.ly attribution policy as in effect from time to time.
4. You will use the APIs in compliance with all applicable laws, statutes, regulations, ordinances or other rules promulgated by governing authorities having jurisdiction over the parties,
5. You shall use instructions provided in the Parse.ly APIs to place application identification information (“API Key”) into any application or service you develop that incorporates or makes any use of the Parse.ly APIs. You can sign up for an API Key at <http://parse.ly/p3>. You must provide accurate identification, contact, and other information required as part of the registration process. You will NOT create any script or other automated tool that attempts to create multiple API Keys. You may not allow any third party to use your API Key for their own benefit.
6. You shall NOT:
 1. use the Parse.ly APIs in any manner or for any purpose that violates any law or regulation, promotes illegal activities, violates third party rights or terms of service, violates any right of any person, including but not limited to intellectual property rights, rights of privacy, or rights of personality, or in any manner inconsistent with the Parse.ly TOS or these Terms of Use;
 2. modify, adapt, alter, translate the API;
 3. use the Parse.ly APIs to engage in spamming or other advertising or marketing activities that violate any applicable laws, regulations or generally-accepted advertising industry guidelines;
 4. sell, lease, share, transfer, or sublicense the Parse.ly APIs, or access or access codes thereto, or derive income from the use or provision of the Parse.ly APIs, whether for direct commercial or monetary gain or otherwise, without Parse.ly’s prior, express, written permission;
 5. use the Parse.ly APIs in a manner that adversely effects Parse.ly and/or the Parse.ly Service or exceeds: (a) reasonable request volume, as set by Parse.ly from time to time, (b) constitutes excessive or abusive usage, or (c) otherwise fails to comply or is inconsistent with any part of the Parse.ly API documentation, as determined by Parse.ly in its sole discretion;
 6. otherwise exercise rights to the API except as expressly allowed by these Terms of Service;
 7. reverse engineer or attempt to reconstruct, identify or discover any underlying ideas, underlying user interface techniques or algorithms related to the Parse.ly Service;
 8. remove, obscure or alter any Parse.ly’s copyright notices, trademarks or other proprietary rights notices affixed to or contained within the API; or
 9. use the Parse.ly APIs in a product or service that competes with products or services offered by Parse.ly, without Parse.ly’s prior, express, written permission.

Parse.ly reserves the right to immediately suspend access to the API if you breach any terms of these Terms.

2. Ownership and Relationship of Parties

The Parse.ly APIs may be protected by copyrights, trademarks, service marks, international treaties, and/or other proprietary rights and laws of the U.S. and other countries. Parse.ly’s rights apply to the Parse.ly APIs and all output and executables of the Parse.ly APIs, excluding any software components developed by you which do not themselves incorporate the Parse.ly APIs or any output or executables of the Parse.ly APIs. You agree to abide by all applicable proprietary rights laws and other laws, as well as any additional copyright notices or restrictions contained in these Terms of Use and in the TOS. Parse.ly owns all rights, title, and interest in and to the Parse.ly APIs. These Terms of Use grant you no right, title, or interest in any intellectual property owned or licensed by Parse.ly, including (but not limited to) the Parse.ly APIs and Parse.ly trademarks.

3. Confidentiality.

You acknowledge that the API is a commercially valuable asset which constitutes a trade secret of Parse.ly and that it contains Confidential Information proprietary to Parse.ly. For purposes of these Terms of Service, “Confidential Information” means the API, all information provided by Parse.ly about the API, and all information provided by Parse.ly that is clearly marked or identified as confidential or that a reasonable person would understand that such information is confidential. You shall not disclose Confidential Information to any third party or use Confidential Information for any purpose other than as expressly permitted by these terms of Service. You agree that you, including your employees, officers, and agents, if any, shall treat all Confidential Information with the same degree of care as it accords to your own confidential information but which, in no event, shall be less than reasonable care and shall not disclose, sell, transfer, publish, copy, display or otherwise make the API available, or any part thereof in any form, to any person or entity not a party to these Terms of Service. You shall take all reasonable precautions to ensure fulfillment of this confidentiality and nondisclosure obligations and agrees to immediately notify Parse.ly of any unauthorized use, copying and/or disclosure of the API, as well as, take such actions as are necessary to cease and prevent any further unauthorized use, copying and/or disclosure.

You acknowledge that any use or disclosure of the API by you in violation of or in any manner inconsistent with these Terms of Service would cause substantial and irreparable injury to Parse.ly, and that Parse.ly’s remedies at law will not be adequate. Accordingly, you agree that Parse.ly shall be entitled to injunctive relief with respect to any breach, or threatened breach of this Section, and that such right shall be in addition to, and not in limitation of, any other rights or remedies to which Parse.ly is or may be entitled at law or equity.

4. Support.

Parse.ly may elect to provide you with support or modifications for the Parse.ly APIs (collectively, “Support”), in its sole discretion, and may terminate such Support at any time without notice to you. Parse.ly may change, suspend, or discontinue any aspect of the Parse.ly APIs at any time, including the availability of any Parse.ly APIs. Parse.ly may also impose limits on certain features and services or restrict your access to parts or all of the Parse.ly APIs or the Parse.ly website without notice or liability.

5. Fees and Payments.

If more than a single individual user will be using or accessing the interface to the Parse.ly Services, you will pay Parse.ly a monthly license fee per user, as set forth on the order form accepted by Parse.ly. Parse.ly reserves the right to charge additional or different fees for future use of, support for, or access to the Parse.ly APIs or the Parse.ly Services in Parse.ly’s sole discretion provided that such charges will be disclosed in advance.

6. Copyright Complaints; Repeat Infringer Policy.

You agree to take whatever actions are necessary or are requested by Parse.ly to enable Parse.ly to comply with Parse.ly’s copyright policy and the take-down and other provisions of the Digital Millennium Copyright Act (“DMCA”) or other applicable laws and regulations with respect to your Parse.ly API. In addition, you acknowledge that in accordance with the DMCA and other applicable law, Parse.ly has adopted a policy of terminating, in appropriate circumstances and at its sole discretion, users and developers who are deemed to be repeat infringers, and that you agree that you will, if requested by Parse.ly, take reasonable steps to terminate access to your Parse.ly API for any user who Parse.ly identifies to you as a repeat infringer.

7. Disclaimer of Any Warranty.

SOME OF THE PARSE.LY APIS ARE EXPERIMENTAL AND HAVE NOT BEEN TESTED IN ANY MANNER. PARSE.LY DOES NOT REPRESENT OR WARRANT THAT ANY PARSE.LY APIS ARE FREE OF INACCURACIES, ERRORS, BUGS, OR INTERRUPTIONS, OR ARE RELIABLE, ACCURATE, COMPLETE, OR OTHERWISE VALID.

THE PARSE.LY APIS ARE PROVIDED “AS IS” WITH NO WARRANTY, EXPRESS OR IMPLIED, OF ANY KIND AND PARSE.LY EXPRESSLY DISCLAIMS ANY AND ALL WARRANTIES AND CONDITIONS, INCLUDING, BUT NOT LIMITED TO, ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AVAILABILITY, SECURITY, TITLE AND/OR NON-INFRINGEMENT. PARSE.LY DOES NOT WARRANT THAT THE API WILL OPERATE WITHOUT INTERRUPTION OR ERROR. NO WARRANTY IS MADE BY PARSE.LY ON THE BASIS OF TRADE USAGE, COURSE OF DEALING OR COURSE OF TRADE. PARSE.LY DOES NOT WARRANT THAT THE PARSE.LY PLATFORM OR PARSE.LY SERVICES WILL MEET YOUR REQUIREMENTS OR THAT THE OPERATION OF THE PARSE.LY PLATFORM WILL BE UNINTERRUPTED OR ERROR-FREE, OR THAT ALL ERRORS WILL BE CORRECTED.

YOUR USE OF PARSE.LY APIS IS AT YOUR OWN DISCRETION AND RISK, AND YOU WILL BE SOLELY RESPONSIBLE FOR ANY DAMAGE THAT RESULTS FROM THE USE OF ANY PARSE.LY APIS INCLUDING, BUT NOT LIMITED TO, ANY DAMAGE TO YOUR COMPUTER SYSTEM OR LOSS OF DATA.

8. Limitation of Liability.

PARSE.LY SHALL NOT, UNDER ANY CIRCUMSTANCES, BE LIABLE TO YOU FOR ANY INDIRECT, INCIDENTAL, CONSEQUENTIAL, SPECIAL OR EXEMPLARY DAMAGES ARISING OUT OF OR IN CONNECTION WITH USE OF THE PARSE.LY APIS, WHETHER BASED ON BREACH OF CONTRACT, BREACH OF WARRANTY, TORT (INCLUDING NEGLIGENCE, PRODUCT LIABILITY OR OTHERWISE), OR ANY OTHER PECUNIARY LOSS, WHETHER OR NOT PARSE.LY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. IN NO EVENT SHALL PARSE.LY’S TOTAL AGGREGATE LIABILITY ARISING FROM OR RELATED TO THIS AGREEMENT, WHETHER IN CONTRACT OR IN TORT OR UNDER ANY OTHER LEGAL THEORY (INCLUDING STRICT LIABILITY AND NEGLIGENCE) EXCEED THE LESSER OF (A) THE TOTAL AMOUNT PAID TO PARSE.LY BY YOU DURING THE SIX (6) MONTH PERIOD IMMEDIATELY PRIOR TO THE EVENT GIVING RISE TO SUCH LIABILITY OR (B) ONE THOUSAND DOLLARS (\$1,000).

9. Essential Basis; Exclusions and Limitations.

The parties acknowledge and agree that the disclaimers, exclusions and limitations of liability set forth in these Terms of Service form an essential basis of these terms of Service, and that, absent any of such disclaimers, exclusions or limitations of liability, the terms of these Terms of Service, including, without limitation, the economic terms, would be substantially different.

SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF CERTAIN WARRANTIES OR THE LIMITATION OR EXCLUSION OF LIABILITY FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES. ACCORDINGLY, SOME OF THE ABOVE LIMITATIONS OF SECTIONS 7 AND 8 MAY NOT APPLY TO YOU.

10. Release and Waiver.

To the maximum extent permitted by applicable law, you hereby release and waive all claims against Parse.ly, and its subsidiaries, affiliates, officers, agents, licensors, co-branders or other partners, and employees from any and all liability for claims, damages (actual and/or consequential), costs and expenses (including litigation costs and attorneys’ fees) of every kind and nature, arising from or in any way related to your use of Parse.ly APIs. If you are a California resident, you waive your rights under California Civil Code § 1542, which states, “A general release does not extend

to claims which the creditor does not know or suspect to exist in his favor at the time of executing the release, which if known by him must have materially affected his settlement with the debtor.” You understand that any fact relating to any matter covered by this release may be found to be other than now believed to be true and you accept and assume the risk of such possible differences in fact. In addition, you expressly waive and relinquish any and all rights and benefits which you may have under any other state or federal statute or common law principle of similar effect, to the fullest extent permitted by law.

11. Release, Hold Harmless and Indemnity.

You irrevocably and unconditionally release and covenant not to use or pursue any claim against Parse.ly and its subsidiaries, affiliates, officers, agents, licensors, co-branders or other partners, and employees, for any and all damages, liabilities, causes of action, judgments, and claims: (i) pertaining to your Parse.ly API, or (ii) which otherwise may arise in connection with your use of, reliance on, or reference to the Parse.ly platform, documentation, or Parse.ly API.

To the maximum extent permitted by applicable law, you agree to hold harmless and indemnify Parse.ly and its subsidiaries, affiliates, officers, agents, licensors, co-branders or other partners, and employees from and against any third party claim arising from or in any way related to your use of Parse.ly APIs or any violation of these Terms of Use or TOS, including any liability or expense arising from all claims, losses, damages (actual and/or consequential), suits, judgments, litigation costs and attorneys’ fees, of every kind and nature. Parse.ly shall use good faith efforts to provide you with written notice of such claim, suit or action.

12. General Terms.

Notwithstanding any provision hereof, for all purposes of the Terms of Use, you and Parse.ly shall be and act independently and not as partner, joint venturer, agent, employee or employer of the other. You shall not have any authority to assume or create any obligation for or on behalf of Parse.ly, express or implied, and you shall not attempt to bind Parse.ly to any contract. If any provision of the Terms of Use is found by a court of competent jurisdiction to be invalid, the parties nevertheless agree that the court should endeavor to give effect to the parties’ intentions as reflected in the provision and the other provisions of such documents remain in full force and effect. The Terms of Use and the relationship between you and Parse.ly shall be governed by the laws of the State of Delaware without regard to its conflict of law provisions. You and Parse.ly agree to submit to the personal jurisdiction of the courts located within the State of Delaware. Parse.ly’s failure to exercise or enforce any right or provision of the Terms of Use shall not constitute a waiver of such right or provision. The section headings and subheadings contained in this agreement are included for convenience only, and shall not limit or otherwise affect the terms of the Terms of Use. Any construction or interpretation to be made of the Terms of Use shall not be construed against the drafter. The Terms of Use constitute the entire agreement between Parse.ly and you with respect to the subject matter hereof. You may not assign any of your rights or obligations under these Terms of Service without the prior written consent of Parse.ly. Any attempted assignment or transfer in violation of the foregoing will be void. No failure of either party to enforce any of its rights under these Terms of Service will act as a waiver of those rights.

Last updated: [January] 2010 / 428478 v2/RE

1.4 Algorithms and Technical Background

Parse.ly recommends articles to users based on several factors: statistical text profiles of the user’s past reading behavior, user taste clustering, and optional user preferences. These will be discussed in turn:

Text profiles: Every action a user performs on an article (such as reading, sharing or ignoring it) is used to train a document classifier based on Naive Bayesian inference. When new articles enter the system, P3 can boost their importance or filter them away – on a per-user basis – by leveraging the user’s existing “resonance profile”. The net effect is that the user sees more personally relevant articles and less that have been known not to elicit a good response.

User taste clustering: Users across the entire Parse.ly Platform are continually clustered into taste profiles, allowing us to perform Amazon/Netflix-style recommendations with their actions. In this way, P3 not only leverages your individual history, but also the collective intelligence of users with similar tastes across all Parse.ly partner sites.

Optional user preferences: This is a set of weighted interests, which are similar to topics/keywords. The interests are normalized and matched against articles to provide the user with an empowering experience, similar to “taste preferences” in systems like Netflix. We recognize that only a small subset of users can be bothered to customize their experience and provide input to our system, so user preferences are entirely optional.

Finally, Parse.ly collects the data that powers our recommender using sophisticated real-time feed processing infrastructure. We plug a publisher’s feed into a PubSubHubub hub, and apply a number of content cleaning and massaging steps in a distributed architecture hosted on our cluster of nodes at Rackspace Cloud. This ensures a clean, high-fidelity reproduction of your content in P3, and ensures content recommendations are available in our widget and API within seconds of publication.

1.5 Contact the Parse.ly Team

We’re happy to help you use the Parse.ly API and provide you with an API key so you can get started.

1.5.1 Get an API Key

You simply need to [fill out this short form](#) in order to have us send you an API Key.

1.5.2 Contact Us Directly

To get this, you can contact us directly, either via e-mail (info@getparsely.com) or [@parse_ly](https://twitter.com/parse_ly) on Twitter.

INDICES AND TABLES

- *Index*
- *Module Index*
- *Search Page*