# DISQUS API 1.1 Documentation

(VERSION: by Anton Kovaloyvok (DISQUS) – Mar 9 2010 – edited Prabir Shrestha – www.prabir.me)

## Table of Contents

## Introduction

The Disqus API enables developers to interact with the Disqus data from their programs. Our first practical goal is to offer sufficient access for someone to write a custom Disqus plugin or a third-party application.

If you have any requests, write to help@disqus.com and we will try to accommodate your request. You can also post any questions here, in the DISQUS Developers mailing list.

## Requests and Data Formats

Disqus API works over HTTP. Each method is provided at the address http://disqus.com/api/{method_name}. By default, requests are HTTP GET requests. All arguments are passed as GET parameters if the method is a GET request, and POST parameters if POST.

There are three types of Disqus objects that this API provides access to: forums, threads, and posts. A post is any comment written by a Disqus user. Each post belongs to a thread, which represents a particular topic of conversation. Each thread belongs to a forum. A forum represents a website that is using Disqus. For example, your blog might constitute a single forum, and each blog post would have its own thread.

For each request you should specify which version of the API you want to use (default is 1.0). To do it, include *api_version* parameter in each request you make to the API. **For current version use '1.1'** as a value.

### Authentication

When a third-party application wants to use the Disqus API it will need API keys. There are three different kinds of keys. First is the *user key*. Every Disqus account has a user key; it is used to perform actions associated with that account. When you are logged in, you can get yours here. Keep this key private.

The second kind of key is the forum key. Every Disqus forum has a forum key. It can be shared among trusted moderators of a forum, and is used to perform actions associated with that forum. The creator of a forum can get the forum's key through the API (see below).

The last one is a partner key. Unless you need something extraordinary, you do not need it.

## Response Formats

All responses are JSON objects with three fields:

- *succeeded*: Indicates whether the call completed successfully or encountered an error.
- *code*: 'ok' if succeeded, otherwise an short description of the error that occurred.
- *message*: The body of the response, which will depend on the method being used. In case of error, contains a longer description of the error that occurred.

Here is the example of a successful request to our API:

```
$ curl -0 -L
"http://disqus.com/api/get_forum_list?user_api_key=API_KEY_HERE&api_version=1
.1"
{
  "message": [
    {
      "created_at": "2007-07-31 17:44:00",
      "shortname": "disqus",
      "description":
      "The official Disqus forum. [...]",
      "id": "NN", "name": "DISQUS Blog and Forum"
    },
    {
      "created_at": "2008-09-10 14:37:31.744838",
      "shortname": "antonkovalyov",
      "description": "",
      "id": "NN",
      "name": "Anton Kovalyov"
    }
  ],
  "code": "ok",
  "succeeded": true
}
```

## Methods

### get_user_name — Validate API key and get username

Method: POST

Required arguments:

- *user_api_key*

This method validates API key and returns username that is associated with it.

### get_forum_api_key — Get a forum API key for a specific forum

Method: GET

Required arguments:

- user_api_key
- forum_id

Returns a string which is forum API key for a given forum.

### get_forum_list — Get a list of websites that user owns or moderates

Method: GET

Required arguments:

- *user_api_key*

### get_forum_posts — Get a list of comments on a website

Method: GET

Required arguments:

- *user_api_key*
- *forum_id*

Optional arguments:

- *category_id* — Filter entries by category
- *limit* — Number of entries that should be included in the response. Default is 25.
- *start* — Starting point for the query. Default is 0.
- *filter* — Type of entries that should be returned.
- *exclude* — Type of entries that should be excluded from the response.

Returns posts from a forum specified by id.

Both filter and exclude are multivalue arguments with coma as a divider. That makes is possible to use combined requests. For example, if you want to get all deleted spam messages, your filter argument should contain 'spam,killed' string.

### get_num_posts — Count a number of comments in articles

Method: GET

Required arguments:

- *user_api_key*
- *thread_ids* — IDs of threads, divided by coma.

This API method returns a hash of threads with their id as a key and number of posts in them as a value.

### get_categories_list — Get a list of categories on a website

Method: GET

Required arguments:

- user_api_key
- forum_id

Returns a list of categories that were created for a website (forum) provided. Returned data will contain category *id*, *title*, *forum_id* and boolean flag *is_default*.

### get_thread_list — Get a list of threads on a website

Method: GET

Required arguments:

- *user_api_key*
- *forum_id*
- *limit* — Number of entries that should be included in the response. Default is 25.
- *start* — Starting point for the query. Default is 0.

Optional arguments:

- *category_id* — Filter entries by their category.

### get_updated_threads — Get a list of threads with new comments

Method: GET

Required arguments:

- *user_api_key*
- *forum_id*
- *since* - Start date for new posts; Format: 2009-03-30T15:41, Timezone: UTC

### get_thread_posts — Get a list of comments in a thread

Method: GET

Required arguments:

- *user_api_key*
- *thread_id*

Optional arguments:

- *limit* — Number of entries that should be included in the response. Default is 25.
- *start* — Starting point for the query. Default is 0.
- *filter* — Type of entries that should be returned (new, spam or killed).
- *exclude* — Type of entries that should be excluded from the response (new, spam or killed).

Both filter and exclude are multivalue arguments with coma as a divider. That makes is possible to use combined requests. For example, if you want to get all deleted spam messages, your filter argument should contain 'spam,killed' string. Note that values are joined by AND statement so 'spam,new' will return all messages that are new **and** marked as spam. It will not return messages that are new and not spam or that are spam but not new (i.e. has already been moderated).

### thread_by_identifier — Get or create thread by identifier

Method: POST

Required arguments:

- *identifier* — unique value (per forum) for a thread that is used to keep be able to get data even if permalink is changed.
- *forum_api_key*
- *title* — if thread does not exist, the method will create it with the specified title (can be an empty string if you are sure that the thread exists)

Optional arguments:

- category_id — if thread does not exist, the method will create it in the specified category

This method tries to find a thread by its identifier and title. If there is no such thread, the method creates it. In either case, the output value is a thread object.

### get_thread_by_url — Get thread by URL

Method: GET

Required arguments:

- *url*

Optional arguments:

- *forum_api_key*
- *partner_api_key*

Finds a thread by its URL. Output value is a thread object.

### update_thread — Update thread

Method: POST

Required arguments:

- *forum_api_key*
- *thread_id*

Optional arguments:

- *title*
- *allow_comments*
- *slug*
- *url*

Updates thread, specified by id and forum API key, with values described in the optional arguments above.

### create_post — Create a new post (i.e. add a new comment)

Method: POST

Required arguments:

- *thread_id*
- *message*
- *author_name*
- *author_email*
- *forum_api_key*

Optional arguments:

- *partner_api_key*
- *created_at* — Format: 2009-03-30T15:41, Timezone: UTC
- *ip_address*
- *author_url*
- *parent_post*
- *state* — Comment's state, must be one of the following: approved, unapproved, spam, killed

Creates a comment to the thread specified by id.

### moderate_post — Delete a comment or mark it as spam (or not spam)

Method: POST

Required arguments:

- *user_api_key*
- *post_id*
- *action* — Name of action to be performed. Value can be spam, approve or kill.

Moderates post and returns modified version