e621-API-Docs

About

"e621 offers a simple API to make scripting easy. All you need is a way to GET and POST to URLs. The ability to parse JSON or XML responses is nice, but not critical. The simplicity of the API means you can write scripts using JavaScript, Perl, Python, Ruby, even shell languages like bash or tcsh."

This is a repository for a better set of documentation for the site e621 (https://e621.net). If you're here, you know what this site is for.

The official documentation is available here (https://e621.net/help/show/api), but it does not go very in depth on GET/POST endpoints and how they differ. Also some information is incorrect. This documentation is derived from the official API documentation but presented in a different and more in-depth manor.

Any general information will be available on this page. if you want to get more information on any specific endpoint, look in the `docs/` folder of this repo.

You can also find .PDF versions of these docs here (/pdf-docs/) if you need them to be portable

The Basics

The e621 API uses the two main HTTP REST methods, **GET** and **POST**. This is how you interact with the API, through REST URL endpoints. Any time you are retrieving data from the API you are using GET, any time your are giving the API information you are using POST.

e621 treats REST URLs as functions, so when you are GETting data, you pass your desired options after the base URL (separated by a `?`).

A basic example looks like this:

```
https://e621.net/post/index.json?limit=10
```

Try it in your browser (https://e621.net/post/index.json?limit=10)

Breaking down the URL

If you've dealt with REST APIs before this should seem familiar, if not here's a breakdown: The base URL will always be `https://e621.net/`, the next part of the URL `post` indicates the base endpoint we want to access. Changing this to something like `artist` would make all `artist` endpoints available to us.

The next part of the URL `index` is considered the 'action'. In this case the `index` action retrieves an index of posts.

After that, the `.json` part of the URL tells e621 that we want a JSON message returned, as opposed to XML. **Both JSON and XML are available but some endpoints are only available in one or the other**.

Lastly are your `parameters`. These change depending on the 'action' you are performing but the allow you to specify the type of data you are receiving from the API. In this case `limit=10` is our parameter. This limits the response from e621 to 10 posts being returned.

Note: Parameters for GET requests must **always** use a `?` to separate the URL from the parameters. Also, any additional parameters beyond the first must be separated with an `&`.

JSONP Support

The API does support JSONP. To use JSONP, append `&callback=mycallbackfunction` to your request. The resulting JSON will be encapsulated into a call to `mycallbackfunction`.

Example

`/blip/index.json?callback=mycallbackfunction`

would return a response like:

```
mycallbackfunction([{"user":"A User","response":null,"body":"Blip one","user_id":1,"id":1}])
```

Using cURL

If you are using cURL or something similar, you may experience failurs on your HTTP requests. The API declares this to be a problem with SSL. If this is the case, you will need to point cURL to a trusted certificate to compare the remote site's (e621.net) certificate with. You can get the certificate here (http://curl.haxx.se/ca/cacert.pem). After you have the cert downloaded, you can point cURL to it like so:

```
curl_setopt($ch, CURLOPT_CAINFO, "/server_dir/apache/cacert.pem");
```

Alternatively you can disable SSL alltogether (not advised).

User-Agent Requirements

Making any requests to the e621 API **requires** a user-agent string. Making this something helpful like your username + project can help e621 get in touch with you if issues arise. (EX: `my-username/myproject-1.0`).

NOTE: Impersonating a browser user-agent will quickly have your IP address blocked from API access by e621.

Rate limiting

e621 has a hard limit of 1 request per 500ms (2/sec). Breaking this limit will result in an HTTP 503 response.

Response Types

The type of response you get from e621 depends on the type of endpoint you requested (XML or JSON).

Any API calls that are modifying data (POST and some special GETs) will return a response like this:

```
{success: false, reason: "duplicate"}
```

For XML the response looks like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<response success="false" reason="duplicate"/>
```

e621 also uses standard HTTP responses as well as some custom responses, listed below.

Status Code	Description
200 OK	Request was successful
403 Forbidden	Access denied. May indicate that your request lacks a User-Agent header (see Notice #2 above).
404 Not Found	Not found
420 Invalid Record	Record could not be saved
421 User Throttled	User is throttled, try again later
422 Locked	The resource is locked and cannot be modified
423 Already Exists	Resource already exists
424 Invalid Parameters	The given parameters were invalid
500 Internal Server Error	Some unknown error occurred on the server
502 Bad Gateway	A gateway server received an invalid response from the e621 servers
503 Service Unavailable	Server cannot currently handle the request or you have exceeded the request rate limit. Try again later or decrease your rate of requests.
520 Unknown Error	Unexpected server response which violates protocol
522 Origin Connection Time-out	CloudFlare's attempt to connect to the e621 servers timed out
524 Origin Connection Time-out	A connection was established between CloudFlare and the e621 servers, but it timed out before an HTTP response was received
525 SSL Handshake Failed	The SSL handshake between CloudFlare and the e621 servers failed

Logging In

Some API actions (Mostly POSTs) will require you to have an e621 account and log in through the API to complete actions as yourself.

You must supply these parameters in your `form headers` even if you are sending a GET request. If you don't know what they are, check out POSTing-to-e621 (/docs/POSTing-to-e621.md) in the `docs/` folder.

The required `form headers` parameters look like this for logging in:

```
{
    // ...
    form: {
        login: "YOUR_E621_USER_NAME",
        password_hash: "YOUR_API_KEY"
    }
}
```

The example here is for Node.js's `request` library but this should be pretty easy to implement in your favorite language.

Every time you are making a GET/POST call to the API you must supply your information within the `form header`..

NOTE: To enable API access, you must go to your account settings and generate an API key. After you have enabled API access **and generated an API key at least once**, you can use the below URL to get your `password_hash`.

```
https://e621.net/user/login.json?name=USERNAME_HERE&password=PASSWORD_HERE
```

If you do not have an API key generated, you will receive a failed login, even if the details are correct. Also, **be aware of the security risks involved in sending your API key through an unencrypted channel**. Typically it's better to generate the API-key on the site and store it in a file somewhere to be used in your project.

Available Base Endpoints

Posts

Refer to Posts (/docs/Posts.md) in the `docs/` folder.

Tags

Refer to Tags (/docs/Tags.md) in the `docs/` folder.

Aliases

Refer to Aliases (/docs/Aliases.md) in the `docs/` folder.

Implications

Refer to Implications (/docs/Implications.md) in the `docs/` folder.

Artists

Refer to Artists (/docs/Artists.md) in the `docs/` folder.

Comments

Refer to Comments (/docs/Comments.md) in the `docs/` folder.

Blips

Refer to Blips (/docs/Blips.md) in the `docs/` folder.

Wiki

Refer to Wiki (/docs/Wiki.md) in the `docs/` folder.

Notes

Refer to Notes (/docs/Notes.md) in the `docs/` folder.

Users

Refer to Users (/docs/Users.md) in the `docs/` folder.

User Records

Refer to User-Records (/docs/User-Records.md) in the `docs/` folder.

Dmail

Refer to Dmail (/docs/Dmail.md) in the `docs/` folder.

Forum

Refer to Forum (/docs/Forum.md) in the `docs/` folder.

Pools

Refer to Pools (/docs/Pools.md) in the `docs/` folder.

Sets

Refer to Sets (/docs/Sets.md) in the `docs/` folder.

Favorites

Refer to Favorites (/docs/Favorites.md) in the `docs/` folder.

Tag History

Refer to Tag-History (/docs/Tag-History.md) in the `docs/` folder.

Flag History

Refer to Flag-History (/docs/Flag-History.md) in the `docs/` folder.

Tickets

Refer to Tickets (/docs/Tickets.md) in the `docs/` folder.

Errors?

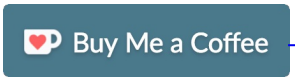
If you find any errors in this documentation please let me know! I try my best but I am only human.

Contributing

If you would like to contribute to this project feel free to make some changes and open a pull request!

Donating

Like what I do?



(https://ko-fi.com/J3J1CYJ2)

Please consider donating even a little. These projects do take time and effort to maintain and aren't exactly mass-marketable

License