Contents

Mumblr REST API Documentation

Mumblr is a micro-blogging service that allows it's users to publish short messages that can be seen by anybody.

Resource endpoints

This is a list of all endpoints exposed by the application. All endpoints need to be made to a domain and prefixed by an API version. For this version of the API these requests will need to be sent to:

```
http://api.mumblr.rossmasters.com/1/
```

So, for example, a request to get information about the user `rossmasters`:

```
http://api.mumblr.rossmasters.com/1/user/rossmasters
```

Each version of the API will increment the version number used in the URI for all requests (a changelog will be available to describe which endpoints have actually changed and the details).

Endpoints marked with an asterisk (*) will require authentication to use. This is further explained in the session resource documentation.

Users

GET /user/:id Show information about the user with the username (id) given.

POST /user/:id * Update user information for a user (which you will need to be authenticated as).

POST /user Create a new user.

DELETE /user/:id * Delete a user (which you will need to be authenticated as).

Session (Authentication)

GET /session Get session information for a session token.

POST /session Create a new session by authenticating as a user.

DELETE /session * End a session by deleting it's token.

Posts

POST /post * Post a new message as a user.

GET /post/:id Get an individual post.

DELETE /post/:id * Delete a post made by a user (who you will need to be authenticated as).

Stream

GET /stream Get the 30 most recently made posts by all users.

Generic errors

A few errors that can be emitted by all requests include:

- `No resource found for your request` - attempted to access a URL that doesn't have a resource attached to it. Check your URL. (404)
- `No use for (verb) on (resource)` - attempting to use a verb (e.g. POST) on a resource that has no implementation for this case. (404)

Sessions

The **session** resource handles the authentication of a user, which is required for some requests.

Authentication method

To authenticate with the API, you will need a pair of user credentials (username and password).

1. Send a POST request to `/session` with the username and password (see below).
2. You will receive a message to confirm it's success, a 128 character token and the session information.
3. Store this token in your application, you will need to pass it to future requests by sending a HTTP header `X-Auth-Token: abcd...12f` with requests that require it.
4. You can use this token as long as the session is valid (which will be until it's expiry date, included in the response from the POST to `/session`.

Sessions last approximately one week before the user must re-authenticate.

Endpoints

POST /session

This method allows you to authenticate as a user and make requests specific to their account.

Parameters:

- `username` - the user's username. (required)
- `password` - the user's password. (required)

Ensure you do not have a token set in X-Auth-Token before making this request.

Responses:

- `message: "Authenticated as (username)"` - successfully authenticated, followed with session information. Remember to keep the token from this response! (200)
- `error: "Credentials were invalid"` - no match was found for the credentials supplied. (401)
- `error: "username and password must be supplied"` - missing a parameter. (406)

GET /session

GET /session/:token

This method will give you information about your session token for this user.

Parameters:

- `token` - a session token, as provided by `POST /session`. (required, or passed as a `X-Auth-Token` header)

If a token was not in the URL, but was available in X-Auth-Token that will be used instead.

Responses:

- `status: "Session valid"` - session has not expired (accompanied by user info, creation and expiration time). (200)
- `status: "Session expired"` - session has expired - authenticate the user again. (200)
- `error: "Invalid session token"` - no session was found with that token. (404)

DELETE /session

DELETE /session/:token

This method allows you to mark the session as closed, so that the token cannot be used for further authenticated requests.

Parameters:

- `token` - a session token, as provided by `POST /session`. (required, or passed as a `X-Auth-Token` header)

If a token was not in the URL, but was available in X-Auth-Token that will be used instead.

Responses:

- `status: "Session ended"` - session closed successfully. (200)
- `status: "Session not ended, probably already closed."` - session not closed, however this is usually due to it already being deleted internally. (200)

Posts

The **post** resource allows creation, viewing and deletion of messages.

Endpoints

GET /post/:id

Retrieves a single message.

Parameters:

- `id` - the post ID number. (required)

Responses:

- Post information (200)
- `error: "Post not found"` - no post with that ID. (404)

POST /post

Allows you to post a message, when you are authenticated as a user.

Parameters:

- `message` - the message to post. (required)

Responses:

- `message: "Posted successfully"` (200)
- `error: "Missing message parameter"` - you are missing the message parameter. (406)
- `error: "You are not authorised to access this entity"` - you have not passed an authentication token. (401)

DELETE /post/:id

Deletes a post the authenticated user has made.

Parameters:

- `id` - the post ID number. (required)

Responses:

- `message: "Post deleted"` (200)
- `message: "Failed to delete post"` - an internal error
- `error: "Post not found"` - no post with that ID. (404)
- `error: "You are not authorised to access this entity"` - you have not passed an authentication token. (401)

Users

The **user** resource allows developers to create, view, edit and delete users.

Endpoints

GET /user/:username

This method allows you to get any user's information by their username.

Parameters:

- `username` - the username of the user

Responses:

- User information as JSON (200)
- `error: "No user found with that username"` - no user was found.

POST /user

This method allows you to register new users.

Parameters:

- `name` - their username, must be lowercase and comprised of characters in the set a-z, _, 0–9. (required)
- `password` - their password. (required)

Responses:

- `message: "Created user"` - the user has been created. (200)
- `error: "You cannot be authenticated and create another user"` - ensure you are not passing `X-Auth-Token` to the request. (401)
- `error: "Supply a username and password"` - you are missing one of the two required fields. (406)

POST /user/:username

This method allows you to update a user's details. You must first authenticate as the user.

Parameters:

- `name` - their username, must be lowercase and comprised of characters in the set a-z, _, 0–9. (optional)
- `password` - their password. (optional)

Responses:

- `message: "Updated user"` - user was updated. (200)
- `error: "You must be authenticated as the user you wish to edit"` - you need to be authenticated as this user. (401)

DELETE /user/:username

This method allows you to delete the user account of the authenticated user.

Parameters:

- `name` - their username.

Responses:

- 'message: "User (username) deleted" - user deleted. (200)
- 'message: "Failed to delete user (username)" - user was not deleted - an internal error.
- `error: "You are not authorised to access this entity"` - you must be authenticated as this user. (401)
- `error: "User not found"` - no user with that username was found. (404)

Stream

The **stream** resource holds lists of posts in chronological order.

Endpoints

GET /stream

Retrieves the 30 most recent messages by all users.

Responses:

- JSON-encoded array of posts (possibly an empty array).