stackptr Documentation

Release 0.1

Geordie Millar, Michael Farrell

January 19, 2017

Contents

Contents:

CHAPTER 1

API Documentation

The API endpoint is located at https://stackptr.com/ All REST requests **must** be made with HTTPS. All URIs in this document are relative to that path.

Access for HTTP is controlled by an API key. These can be managed in the web interface at https://stackptr.com/api/

When an API call is made, the call will return a number of objects. These objects represent the state updates that resulted from performing that API call.

Most API calls can be made via either the HTTPS or WAMP endpoints (TODO: document which ones, and make the ones that should be available usable on both)

In addition to the API calls, you can also connect via the WAMP transport and receive location updates pushed directly to your client instead of polling the `/users` endpoint repeatedly.

It is recommended that the application simply implement generic parsers for each type of message, instead of for each API call, as some endpoints return the same message types.

1.1 Swagger

There is a Swagger instance running at https://stackptr.com/swagger/ that allows you to experiment with the API.

1.2 Format specifics

API keys are passed with the GET or POST parameter `apikey`. All calls require this key unless otherwise specified.

Latitude and longitude are passed using the WGS84 CRS, and encoded as a decimal number of degrees with a precision of up to 64 bit. Southern latitudes are indicated as negative numbers, western longitudes are indicated as negative.

Altitude is reported in metres above sea level.

Heading is passed in degrees, with north being 0/360 degrees.

Speed is reported in metres per second at ground level.

1.3 Test endpoints

`POST /uid`
: Tests connectivity and authentication to the stackptr API.

Useful to see if the API key provided to your app is correct.

Returns a JSON dictionary consisting of:

Parameters

- **id** (*int*) – Your numerical uid
- **username** (*string*) – Your username
- **icon** (*string*) – The user's gravatar (in 256x256).

1.4 CSRF

GET /csrf

Returns a CSRF token. All requests made with API keys are exempt from CSRF checks, so you will not usually need this.

The Android app uses this to POST to /login, get a session and create an API key automatically for you.

The Web UI uses this before the post to /ws_token as the CSRF token obtained when the page first loaded may be expired if the websocket connection drops and reconnects a long time after the page first loaded.

This endpoint does not support CORS for obvious reasons.

1.5 Update location

POST /update

Updates your current location, using (ideally) data from your GPS receiver. Do not report a location if your location is not known.

Parameters

- **lat** (*float*) – Your current latitude.
- **lon** (*float*) – Your current longitude.
- **alt** (*float*) – Your current altitude, in metres, if known. Omit parameter if unknown.
- **hdg** (*float*) – Your current heading, in degrees, if known. Omit parameter if unknown.
- **spd** (*float*) – Your current speed, in metres/second, if known. Omit parameter if unknown.
- **ext** (*string*) – A JSON dict of additional optional information.

In response, the server will send back `OK`. Anything else indicates an error.

class Extra

Parameters

- **bat** (*string*) – Battery life between 0.0 and 1.0.
- **bst** (*string*) – Battery status: `charging`, `full` or `discharging`.
- **prov** (*string*) – Location provider name.

Arbitrary other parameters may be included - more will be defined later.

1.6 WAMP Connections

POST /ws_uid
Returns your current UID (i.e. the UID that you should authenticate to the WAMP server as).

POST /ws_token
Returns a token used in the challenge/response WAMP authentication.

1.7 User Data

GET /users | com.stackptr.api.userList
Gets a list of users on stackptr and their current locations.

The response is encoded as JSON.

This is returned as a list of *MessageItem*.

class MessageItem
Structure for storing messages sent over the wire in `/users` calls or WAMP calls.

type
The type of message being sent. This is one of the message types.

data
Types of object:

user-me A *TrackedUser* for your user.

user An array of *TrackedUser* for users that you watch.

user-pending An array of users that you want to follow but they have not accepted. (FIXME: format)

user-request An array of users that want to follow you but you have not accepted. (FIXME: format)

class TrackedUser
Structure for passing location information about tracked users in the StackPtr API.

Parameters

- **loc** (*array*) – Array containing `[latitude, longitude]` containing the current location of the user.
- **username** (*string*) – The username of the tracked user.
- **icon** (*string*) – URI of the avatar for the user (64x64).
- **lastupd** (*string*) – Time of last update, in seconds since UNIX epoch in UTC.
- **alt** (*string*) – Altitude of the user in metres above sea level.
- **extra** (*string*) – A dictionary of *Extra* information about the user.
- **hdg** (*string*) – Heading of the user.
- **id** (*string*) – User ID of the user.
- **spd** (*string*) – Speed of the user

GET /lochist | com.stackptr.api.lochist
Get the specified user's location history.

Parameters **uid** (*int*) – The user you want to get the location history of. If not specified, will fetch your own.

Returns a message of type `lochist`, with the data containing `id` and `lochist`. `lochist` in this is an array of dictionaries containing `lat` and `lng` objects. This array is ordered from least recent to most recent.

Returns "Permission Denied" if you fetch a user that is not in your user list.

Your application should fetch this only once upon first load, and then append to this list itself instead of repeatedly fetching this endpoint.

1.8 User Management

POST /adduser | com.stackptr.api.addUser
Request permission to see a user's location. Grants them permission to see yours.

Parameters **user** (*string*) – Username or email address of user to add

POST /acceptuser | com.stackptr.api.acceptUser
Accept another user's add request

Parameters **uid** (*int*) – User ID to accept

POST /deluser | com.stackptr.api.delUser
Delete a user from your contact list and you from theirs

Parameters **uid** (*int*) – User ID to delete

1.9 Group Data

GET /grouplist | com.stackptr.api.groupList
Get the list of groups the user is in

class Group
Structure for grouplist responses

name
Name of the group

id
ID of the group

description
Description of the group

status
0 = open to join via group discovery 1 = require owner approval to join

members
List of members in the group, containing username, icon, id, role

Role: 1 = member 2 = administrator

GET /groupdiscover | com.stackptr.api.groupDiscover
Gets a list of groups that are open for public discovery and that you are not already in.

POST /creategroup | com.stackptr.api.createGroup
Create a new group.

Parameters

- **name** (*string*) – Name for group
- **description** (*string*) – Description for group
- **status** (*string*) – 0 if others can discover group, 1 for private group.

POST /joingroup | com.stackptr.api.joinGroup

Parameters **gid** (*string*) – ID of group to join

POST /leavegroup | com.stackptr.api.leaveGroup

Parameters **gid** (*string*) – ID of group to leave. You can't leave a group that you are the sole admin of.

POST /deletegroup | com.stackptr.api.deleteGroup

Parameters **gid** (*string*) – ID of group to delete. You must be an admin.

POST /updategroup | com.stackptr.api.updateGroup

Parameters

- **name** (*string*) – Name for group
- **description** (*string*) – Description for group
- **status** (*string*) – 0 if others can discover group, 1 for private group.
- **gid** (*string*) – Group ID

POST /groupusermod | com.stackptr.api.groupUserMod

Parameters

- **gid** (*string*) – ID of group
- **uid** (*string*) – ID of user
- **user** (*string*) – Alternatively specify user by username or email.
- **role** (*string*) – New role for user. 0 to delete user, 1 to demote to regular user, 2 to promote to admin.

POST /groupdata | com.stackptr.api.groupData

Gets a dict of the data (placemarks etc) for a group. The key for the dict is the object's ID (unique across all groups) and the value is a *GroupData* item.

Parameters **gid** (*int*) – The group ID you want data for.

class GroupData

Structure representing an object in a group like a placemark, line or polygon.

name

Name of the item.

owner

Username of the owner / creator of the object.

json

GeoJSON representing the object as it is to be drawn on the map.

POST /addfeature | com.stackptr.api.addFeature

Adds a new item to the group.

Parameters

- **name** (*string*) – Name for object
- **group** (*string*) – Group id to add feature to
- **gjson** (*string*) – GeoJSON representation of the object

POST /delfeature | com.stackptr.api.deleteFeature
Deletes an item in the group.

Parameters **fid** (*int*) – ID of object to delete

POST /editfeature | com.stackptr.api.editFeature
Edits the geometry of an item in the group.

Parameters

- **fid** (*int*) – ID of object to rename
- **gjson** (*string*) – new geoJSON of object

POST /renamefeature | com.stackptr.api.renameFeature
Renames an item in the group.

Parameters

- **fid** (*int*) – ID of object to rename
- **name** (*string*) – New name for object

| com.stackptr.api.setSharedToGroup
Start or stop sharing to a group.

Parameters

- **gid** (*string*) – ID of group
- **share** (*string*) – 1 to start sharing to group, 0 to stop sharing

| com.stackptr.api.sharedGroupLocs
Get the locations of group members sharing to the group.

Parameters **gid** (*string*) – ID of group

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

E

G

M

T