Team WeFixIt Rest Documentation

Overview

The Chat REST Service (CHS) provides the interface needed to interact with a site that tracks users, conversations, and messages posted to those conversations. The site tracks many different conversations, and any user may make a post to any conversation, see other user's posts, etc.

Team WeFixIts Rest Documentation dictates the framework needed to utilize an iFixit mockup site that allows users to visualize what a potential ad would look like if displayed in the real site. The site provides the ability to add, modify, delete, and view potential advertisements that contain both text and template images.

General Points

The following design points apply across the document.

1. All resource URLs are presumed to be prefixed by a root URL giving the hostname and a slash, (e.g. http://www.softwareinventions.com/)
2. All resources accept and provide only JSON body content.
3. Some GET operations allow query parameters. These are listed directly after the GET word. All query parameters are optional unless given in bold.
4. Absent documentation to the contrary, all DELETE calls, POST, and PUT calls with a 400 HTTP response return as their body content, a list of JSON objects describing all errors that occured. Each error object within that list is of form {tag: {errorTag}, params: {params}} where errorTag is a string tag identifying the error, and params is an array of additional values needed to fill in details about the error, or is null if no values are needed. E.g. {detail: "Not Found", params: ["lastName"]} And, per REST standards, all successful (200 code) DELETE actions return empty body.
5. Resource documentation lists possible errors only when the error is not obvious from this General Points section. Relevant errors may appear in any order in the body. Missing field errors are checked first, and no further errors are reported if missing fields are found.
6. All resource-creating POST calls return the newly created resource as a URL path via the Location response header, not in the response body.
7. GET calls return one of the following, and return the earliest one that is applicable. The response body is empty in cases b-d. Get calls whose specified information is a list always return an array, even if it has just one or even zero elements.
    1. 500 for any server error, with body giving a dump of the stack upon error if available.
    2. NOT_FOUND for a URL that is not described in the REST spec.
    3. UNAUTHORIZED for missing login.
    4. FORBIDDEN for insufficient authorization despite login

5. BAD_REQUEST and a list of error strings.
6. HTTP code OK and the specified information in the body.

8. Fields of JSON content for POST and PUT calls are assumed to be strings, booleans, ints, or doubles without further documentation where obvious by their name or intent. In nonobvious cases, the docs give the type explicitly.
9. All access requires authentication via login to establish the Authenticated User (AU); no resources are public except for Prss/POST (for initial registration), and Ssns/POST (to log in). Other resources may be restricted based on admin status of the AU. The default restriction is to allow only access relevant to the AU, unless the AU is admin, in which case access to any Person's info is allowed.
10. Any database query failure constitutes a server error (status 500) with a body giving the error object returned from the database query. Ideally, no request, however badly framed, should result in such an error except as described in point 11
11. The REST interface does no general checking for forbiddenField errors, unless the spec specifically indicates it will. Absent such checking, illegal fields in PUT/POST call bodies may result in database query errors and a 500 code, as may an empty body when body content is expected.
12. All fields listed for a POST are required by default unless the description says otherwise. Required fields may not be passed as null or "". Doing so has the same outcome as if the field were entirely missing. (Note that JSON provides no way of transmitting a field with value “undefined”, so this case is moot.)
13. All times are integer values, in mS since epoch.
14. Non JSON parseable bodies result in 500 error.
15. Excess length of string fields in POST/PUT calls results in 400 error with badValue tag.

Error Codes

The possible error codes, and any parameters, are as follows.

missingField Field missing from request. Params[0] gives field name. Missing field errors may result in further errors being ignored and not reported until all required fields are present.

badValue A field has a value that violates the spec in some way not covered by more specific errors. Params[0] gives the field's name, e.g. "firstName".

notFound Entity not present in DB -- for cases where a Conversation, Person, etc. is not there.

badLogin Email/password combination invalid, for errors logging.

forbiddenRole Attempt to create a non-student person without having admin privilege to do so

dupEmail Email duplicates an existing email

noTerms Acceptance of terms is required

noOldPwd Change of password requires an old password

oldPwdMismatch Old password that was provided is incorrect.

dupTitle Conversation title duplicates an existing one

forbiddenField Field in body not allowed. Params[0] gives field name.

queryFailed Query failed (server problem)

Resources for Campaigns

The following resources allow creation, deletion, and management of Campaigns -- each a series of advertisements. Any AU may GET information on any Campaign and may POST new Campaigns.

Campaigns

GET

Any AU is acceptable, though some login is required. Return an array of 0 or more elements, with one element for each campaign in the system:

id Id of the Campaign

name Name of the Campaign (100 char max)

startDate Start date of the Campaign

endDate End date of the Campaign

POST

Create a new Campaign. Error if dates are invalid or a Campaign already exists with the specified name. Can be AU but some login is required.

Fields are

id Id of the Campaign

name Name of the Campaign (100 char max)

startDate Start date of the Campaign

endDate End date of the Campaign

Campaigns{id}

GET

Returns single campaign object with matching ID. Returns error detail not found if id is not found. Any AU is acceptable though some login is required.

PUT

Update all fields of the campaign. Fields as for Campaign's POST, not including id. Not providing a field will return error with matching field and "This field is required". Any AU is acceptable though some login is required.

PATCH

Update a single field of the campaign. Can include any field from Campaigns post, not including id. Unlike PUT, not including a field will not return an error. Any AU is acceptable though some login is required.

DELETE

Delete the Campaign, including all associated advertisements. Any AU is acceptable though some login is required.

Resources for Advertisements

The following resources allow creation, deletion, and management of Advertisements; each of which contains an id, header text, image, secondary text, and button clicked link. Any AU may GET information on any Advertisement and may POST new Advertisements.

Advertisements

GET

Any AU is acceptable, though some login is required. Returns a json of 0 or more elements, with one element for each advertisement in the system:

id Id of the Advertisement

name Name of the Advertisement (400 char max)

image Link to image stored in /media

second_text Secondary name of the add

button_rendered_link Link that will display when the user click on the button

POST

Create a new Advertisement. Error if image is not in media or Advertisement exists with given id or name. Can be any AU but some login is required.

Fields are

id Id of the Advertisement

name Name of the Advertisement (400 char max)

image Link to image stored in /media

second_text Secondary name of the add

button_rendered_link Link that will display when the user click on the button

Advertisements{id}

GET

Returns single Advertisement object with matching ID. Returns error detail not found if id is not found. Any AU is acceptable though some login is required.

PUT

Update all fields of the Advertisement. Fields as for Advertisement's POST, not including id. Not providing a field will return error with matching field and "This field is required". Any AU is acceptable though some login is required.

PATCH

Update a single field of the Advertisement. Can include any field from Advertisement's post, not including id. Unlike PUT, not including a field will not return an error. Any AU is acceptable though some login is required.

DELETE

Delete the Advertisement, including all associated data. Any AU is acceptable though some login is required.