FeedHQ Documentation

Release 1.0

Bruno Renié

Nov 03, 2018

Contents

FeedHQ is a web-based feed build with readability and mobility in mind. It's completely open-source, meaning anyone can host FeedHQ privately and contribute to its development. A hosted service is available at https://feedhq.org. The FeedHQ source code is available on GitHub.

CHAPTER 1

Main features

- RSS/Atom support
- Readability-oriented layout on all screen sizes (mobile, tablet, desktop)
- Compatibility with the Google Reader API
- Integration with reading list services such as Wallabag, Instapaper or Pocket
- OPML export / import, Subtome integration
- Keyboard shortcuts, syntax highlighting

CHAPTER 2

Contents

2.1 The FeedHQ API

FeedHQ implements the Google Reader API. This document, while being FeedHQ's API documentation, aims to be a reference for developers seeking information about the details of this API.

The Google Reader API was never publicly released or documented but developers have reverse-engineered it and built an ecosystem of applications that use this API implementation for syncing mobile or desktop apps with Google Reader accounts.

A handful of resources are available in various places of the internet but it's tedious for developers to get an accurate and extensive idea of how the API works. This is FeedHQ's attempt to fix that.

2.1.1 Terminology

Before building things with the API, it's important to understand a couple of concepts that determine how the API works. The API is not particularly resource-oriented, not so *RESTful*, but once the concepts are understood it's rather easy to get data out of this API.

Data model

The root element is a *feed*. It's simply the URL of an RSS feed that gets polled for fetching feed items.

Feeds can optionally belong to a *label*. Google reader supported multiple labels per feed but FeedHQ only allows feeds to belong to one (or zero) label.

Feed items — or just *items* — are articles, news items or posts that are extracted and stored during feed fetching.

Streams

Streams are lists of feed items. They represent a criteria that is used to fetch a list of items, e.g.:

- the feed to which items belong to
- the label to which items belong to
- the state that items must have (starred, read)

Streams have an identifier called a *Stream ID*. This identifier can take several forms:

- For a label, the string `user/-/label/<name>` where `<name>` is the label's name
- For a feed, the string `feed/<feed url>` where `<feed url>` is the complete URL for the feed
- For a state, the string `user/-/state/com.google/<state>` where `<state>` is one of `read`, `kept-unread`, `broadcast`, `broadcast-friends`, `reading-list`, `starred`, or any other string that gets interpreted as a *tag*.
- For a combination of multiple streams, the string `splice/` followed by stream IDs separated with the pipe (`|`) character. Splice items are combined in an OR query. E.g. `splice/user/-/label/foo|user/-/state/com.google/starred` represents all items that are starred **or** in the `foo` label.

Furthermore, for states or labels, the `user/-/` prefix can also contain the user ID instead of the dash. `user/12345/label/test` is a valid stream ID, assuming the number `12345` matches with the authenticated user making the request.

Here is a summary of the filtering that is done for all states:

State	Filter
read	read items
kept-unread	unread items
broadcast, broadcast-friends	broadcast items
reading-list	all items
starred	starred items

broadcast is more or less a no-op: FeedHQ stores this attribute and lets you set it but there is no public-facing feature that uses this attribute yet.

All states that are not in this table are treated as *tags*. Items can be tagged and searching for `user/-/state/com.google/test` will look for items having the `test` tag.

Items

Items are identified by a globally unique numerical *ID*. Item IDs can take two forms:

- The short form, just the actual ID. E.g. `12345`.
- The long form, the prefix `tag:google.com,2005:reader/item/` followed by the item ID as an unsigned base 16 number and 0-padded to be always 16 characters long.

Examples:

Short form	Long form
`12309438943892`	`tag:google.com,2005:reader/item/00000b3203bc5294`
`87238913628312`	`tag:google.com,2005:reader/item/00004f57e4751898`

Here is some sample Python code that converts from and to long-form IDs.

```
import struct

def to_long_form(short_form):
    value = hex(struct.unpack("L", struct.pack("l", short_form))[0])
    if value.endswith("L"):
        value = value[:-]
    return 'tag:google.com,2005:reader/item/{0}'.format(
        value[2:].zfill(16)
    )

def to_short_form(long_form):
    value = int(long_form.split('/')[-1], 16)
    return struct.unpack("l", struct.pack("L", value))[0]
```

When the API documentation mentions passing an *item ID* as a parameter, clients are free to use the short form or the long form.

Input formats

API calls use the `GET` or `POST` HTTP methods. Some calls support both methods, some don't.

When using `GET`, parameters can be passed as querystring parameters.

When using `POST`, parameters can be passed in the request body, as form data with the `application/x-www-form-urlencoded` encoding.

In some cases parameters can be repeated, to treat them as lists. The API simply expects parameters to be repeated. E.g. `?i=12345&i=67890&i=...`. *When the API expects a list*, it will understand that as `i = [12345, 67890]`.

Authentication

API calls are authenticated using API tokens. The API call to retrieve a token is `/accounts/ClientLogin`.

This API call accepts both `GET` parameters and `POST` data, but it is strongly recommended to use `POST`.

URL: `/accounts/ClientLogin`

Parameters:

- `Email`: the user's email
- `Passwd`: the user's account password

The response comes back as `plain/text` and contains 3 lines:

```
SID=...
LSID=...
Auth=<token>
```

Clients should store the token from the third line, following the `Auth=` marker.

API tokens expire like web sessions. Clients need to renew them every now and then. FeedHQ's expiration time for auth tokens is 7 days. When a token expires, the API returns HTTP 401 responses.

Once a token has been generated, it needs to be passed in the HTTP `Authorization` header when making API calls, with the following format:

```
Authorization: GoogleLogin auth=<token>
```

Output formats

The API supports content negotiation for most API calls. The commonly supported formats are:

- XML
- JSON

Additionally, some API calls support Atom. Some only support one output format and will disregard any content negotiation. Some other calls return plain text responses when the data to return is simple enough.

Content negotiation can be done in two ways:

- with the HTTP `Accept` header
- with the `output` querystring parameter

Here are the relevant parameters to pass to the API

Format	Accept header	`output` parameter
XML	application/xml	xml
JSON	application/json	json
Atom	text/xml	atom

The default output format — when nothing is specified by the client — is XML.

POST Token

Additionally to authentication, API calls that mutate data in the FeedHQ data store and that are made using the `POST` method need to include a *POST token*.

The POST token is a short-lived token that is used for CSRF protection. It must be included in request bodies as a `T` parameter.

Retrieving a POST token is as simple as issuing a GET request to `/reader/api/0/token`. The token is returned as a plain-text string and can be used in POST requests.

When the token is required but missing, the API will return an HTTP 400 response.

When the token is present but invalid, the API will return an HTTP 401 response with a special header, `X-Reader-Google-Bad-Token: true`. This header means that the token needs to be renewed by simply making a new request to `/reader/api/0/token` and storing the updated token.

API Clients should use their tokens as long as they are valid, and renew them only when they see the bad-token response.

FeedHQ's POST tokens are valid for 30 minutes.

2.1.2 API Reference

This page lists all API calls implemented in the FeedHQ API. It is important to read and understand the *terminology* before refering to this API reference.

user-info

Returns various details about the user.

URL	/reader/api/0/user-info
Method	GET
Supported formats	XML, JSON

unread-count

Returns all streams that have unread items, along with their unread count and the timestamp of their most recent item.

URL	/reader/api/0/unread-count
Method	GET
Supported formats	XML, JSON

Sample JSON output:


```
{
    "max": 1000,
    "unreadcounts": [
        {
            "count": 1,
            "id": "feed/http://rss.slashdot.org/Slashdot/slashdot",
            "newestItemTimestampUsec": "1405452360000000"
        },
        {
            "count": 1,
            "id": "feed/http://feeds.feedburner.com/alistapart/main",
            "newestItemTimestampUsec": "1405432727000000"
        },
        {
            "count": 2,
            "id": "user/1/label/Tech",
            "newestItemTimestampUsec": "1405432727000000"
        },
        {
            "count": 2,
            "id": "user/1/state/com.google/reading-list",
            "newestItemTimestampUsec": "1405432727000000"
        }
    ]
}
```


disable-tag

Deletes a category or a tag. Feeds that belong to the category being deleted are moved to the top-level.

URL	/reader/api/0/disable-tag
Method	POST
Supported formats	Returns “OK” in plain text
POST token required	Yes

Required POST data:

- s the category’s stream ID, **or** t, the category (label) name.

rename-tag

Renames a category.

URL	/reader/api/0/rename-tag
Method	POST
Supported formats	Returns "OK" in plain text
POST token required	Yes

Required POST data:

- s the category's stream ID, **or** t, the category (label) name.
- dest, the new label name, in its stream ID form: user/-/label/<new label>.

subscription/list

Lists all your subscriptions (feeds).

URL	/reader/api/0/subscription/list
Method	GET
Supported formats	XML, JSON

Sample JSON output:

```
{
    "subscriptions": [
        {
            "title": "A List Apart",
            "firstitemmsec": "1373999174000",
            "htmlUrl": "http://alistapart.com",
            "sortid": "B0000000",
            "id": "feed/http://feeds.feedburner.com/alistapart/main",
            "categories": [
                {
                    "id": "user/1/label/Tech",
                    "label": "Tech"
                }
            ]
        }
    ]
}
```

subscription/edit

Creates, edits or deletes a subscription (feed).

URL	/reader/api/0/subscription/edit
Method	POST
Supported formats	Returns "OK" in plain text
POST token required	Yes

POST data for each action:

- Creation:
  - `ac`: the string `subscribe`
  - `s`: the stream ID to create (`feed/<feed url>`).
  - `t`: the name for this subscription.
  - (optional) `a`: the stream ID of a category. If the category doesn't exist, it will be created.
- Edition:
  - `ac`: the string `edit`
  - `s`: the stream ID to edit (`feed/<feed url>`).
  - `r` or `a`: the stream ID of a category. `r` moves the feed out of the category, `a` adds the feed to the category.
  - `t` a new title for the feed.
- Deletion:
  - `ac`: the string `unsubscribe`
  - `s`: the stream ID to delete (`feed/<feed url>`).

subscription/quickadd

Adds a new subscription (feed), given only the feed's URL.

URL	`/reader/api/0/subscription/quickadd`
Method	`POST`
Supported formats	XML, JSON
POST token required	Yes

POST data:

- `quickadd`: the URL of the feed, as a stream ID or just a standard URL.

Sample JSON output:

```
{
    "numResults": 1,
    "query": "http://feeds.feedburner.com/alistapart/main",
    "streamId": "feed/http://feeds.feedburner.com/alistapart/main",
}
```

subscription/export

Returns the list of subscriptions in OPML (XML) format.

URL	`/reader/api/0/subscription/export`
Method	`GET`
Supported formats	XML (OPML)

subscription/import

Imports all subscriptions from an OPML file.

URL	`/reader/api/0/subscription/import`
Method	`POST`
Supported formats	Returns "OK: <count>" in plain text

Instead of form data, this API call expects the contents of the OPML file to be provided directly in the request body.

subscribed

Returns whether the user is subscribed to a given feed.

URL	`/reader/api/0/subscribed`
Method	`GET`
Supported formats	Returns "true" or "false" in plain text

Querystring parameters:

- `s`: the stream ID of the feed to check.

stream/contents

Returns paginated, detailed items for a given stream.

URL	`/reader/api/0/stream/contents/<stream ID>`
Method	`GET`
Supported formats	XML, JSON, Atom

The stream ID is part of the URL. Additionally, the following querystring parameters are supported:

- `r`: sort criteria. Items are sorted by date (descending by default), `r=o` inverts the order.
- `n`: the number of items per page. Default: 20.
- `c`: the *continuation* string (see below).
- `xt`: a stream ID to exclude from the list.
- `it`: a steam ID to include in the list.
- `ot`: an epoch timestamp. Items older than this timestamp are filtered out.
- `nt`: an epoch timestamp. Items newer than this timestamp are filtered out.

Continuation is used for pagination. When FeedHQ returns a page, it contains a `continuation` key that can be passed as a `c` parameter to fetch the next page.

Sample JSON output:

```
{
    "direction": "ltr",
    "author": "brutasse",
    "title": "brutasse's reading list on FeedHQ",
    "updated": 1405538866,
```

(continues on next page)

(continued from previous page)

```
    "continuation": "page2",
    "id": "user/1/state/com.google/reading-list"
    "self": [{
        "href": "https://feedhq.org/reader/api/0/stream/contents/user/-/state/com.
↪google/reading-list?output=json"
    }],
    "items": []
}
```

`items` contains the list of feed items. Each item has the following structure:

```
{
    "origin": {
    },
    "updated": 1405538866,
    "id": "tag:google.com,2005:reader/item/0000000009067698",
    "categories": [
        "user/1/state/com.google/reading-list",
        "user/1/label/Tech"
    ],
    "author": "Somebody",
    "alternate": [{
        "href": "http://example.com/href.html",
        "type": "text/html"
    }]
    "timestampUsec": "1405538280000000",
    "content": {
        "direction": "ltr",
        "content": "actual content",
    },
    "crawlTimeMsec": "1405538280000",
    "published": 1405538280,
    "title": "Example item test title"
}
```

You'll notice that epoch timestamps are integers but when dates are expressed in miliseconds (Msec) or microseconds (Usec) they are returned as strings.

stream/items/ids

Returns item IDs for a given stream ID.

URL	`/reader/api/0/stream/items/ids`
Method	`GET`
Supported formats	XML, JSON

Querystring parameters:

- `s`: the stream ID.
- `n` the number of item IDs per page to return.
- (optional) `includeAllDirectStreamIds`: set it to `true` to include stream IDs in items.
- (optional) `c`: the continuation string when requesting a page.
- (optional) `xt`, `it`, `nt` and `ot` are supported like in the *stream/contents* API call.

stream/items/count

Returns the number of items in a given stream.

URL	`/reader/api/0/stream/items/count`
Method	`GET`
Supported formats	Returns the count in plain text

Querystring parameters:

- `s`: the stream ID.
- (optional) `a`: set it to `true` to also get the date of the latest item in the stream.

Sample output, without `a`:

```
20174
```

Sample output, with `a`:

```
20174#July 16, 2014
```

stream/items/contents

Returns the details about requested feed items.

URL	`/reader/api/0/stream/items/contents`
Method	`GET`, `POST`
Supported formats	XML, JSON, Atom

Items are requested via the `i` querystring parameter or post parameter. It can be repeated as many times as needed. When requesting a large number of items, it is recommended to use POST to avoid hitting URI length limits.

tag/list

Returns the list of special tags and labels.

URL	`/reader/api/0/tag/list`
Method	`GET`
Supported formats	XML, JSON

Sample JSON output:

```
{
    "tags": [
        {
            "id": "user/1/state/com.google/starred",
            "sortid": "A0000001"

        },
        {
            "id": "user/1/states/com.google/broadcast",
            "sortid": "A0000002"
```

(continues on next page)

(continued from previous page)

```
        },
        {
            "id": "user/1/label/Tech",
            "sortid": "A0000003"
        },
    ]
}
```

edit-tag

Adds or remove tags from items. This API call is used to mark items as read or unread or star / unstar items.

URL	`/reader/api/0/edit-tag`
Method	`POST`
Supported formats	Returns "OK" in plain text
POST token required	Yes

POST parameters:

- `i`: ID of the item to edit. Can be repeated to edit multiple items at once.
- `a`: tag to add to the items. Can be repeated to add multiple tags at once.
- `r`: tag to remove from the items. Can be repeated to remove multiple tags at once.

Possible tags are:

- `user/-/state/com.google/kept-unread`
- `user/-/state/com.google/starred`
- `user/-/state/com.google/broadcast`
- `user/-/state/com.google/read`

For example, to mark an item as read and star it at the same time:

```
i=12345&a=user/-/state/com.google/starred&a=user/-/state/com.google/read
```

mark-all-as-read

Marks all items in a stream as read.

URL	`/reader/api/0/mark-all-as-read`
Method	`POST`
Supported formats	Returns "OK" in plain text
POST token required	Yes

POST parameters:

- `s` the stream ID to act on.
- (optional) `ts`: an epoch timestamp **in microseconds**. When provided, only items *older* than this timestamp are marked as read.

preference/list

URL	/reader/api/0/preference/list
Method	GET
Supported formats	XML, JSON

Returns a static response:

```
{
    "prefs": [{
        "id": "lhn-prefs",
        "value": "{\"subscriptions\":{\"ssa\":\"true\"}}"
    }]
}
```

Yes, `value` is JSON-encoded JSON. `ssa=true` tells clients that subscriptions are sorted alphabetically. FeedHQ doesn't support custom sorting.

preference/stream/list

URL	/reader/api/0/preference/stream/list
Method	GET
Supported formats	XML, JSON

Returns a static response:

```
{
    "streamprefs": { }
}
```

friend/list

URL	/reader/api/0/friend/list
Method	GET
Supported formats	XML, JSON

Returns a single friend, the authenticated user:

```
{
    "friends": [{
        "p": "",
        "contactId": "-1",
        "flags": 1,
        "stream": "user/1/state/com.google/broadcast",
        "hasSharedItemsOnProfile": false,
        "profileIds": [
            "1"
        ],
        "userIds": [
            "1"
```

(continues on next page)

(continued from previous page)

```
        ],
        "givenName": "brutasse",
        "displayName": "brutasse",
        "n": ""
    }]
}
```

Undocumented / not implemented

The following API calls are known to exist in the Google Reader API but haven't been implemented in the FeedHQ API:

- /related/list
- /stream/details
- /item/edit
- /item/delete
- /item/likers
- /friend/groups
- /friend/acl
- /friend/edit
- /friend/feeds
- /people/search
- /people/suggested
- /people/profile
- /comment/edit
- /conversation/edit
- /shorten-url
- /preference/set
- /preference/stream/set
- /search/items/ids
- /recommendation/edit
- /recommendation/list
- /list-user-bundle
- /edit-bundle
- /get-bundle
- /delete-bundle
- /bundles
- /list-friends-bunle
- /list-featured-bundle

CHAPTER 3

Indices and tables

- genindex
- modindex
- search