Userfeeds Documentation

Release 0.1.0

Grzegorz Kapkowski

Aug 16, 2018

Contents

CHAPTER 1

Introduction to the Userfeeds Platform

Userfeeds is envisioned as an infrastructure platform which allows developers to utilize easily content rankings for their own use or for their clients and build better ones whenever needed.

The product and core of the Userfeeds Platform which can be used for that purpose is the http *API* which allows to get data from internet transactions as well as content rankings and to access databases in which data on transactions, which make the rankings, are stored.

Internet transactions are blocks of the Javascript code which can be called snippets and placed on websites as elements serving for engaging visitors into promotional activities (such as liking and promoting) which are based on the use of crypto-currencies, mainly Ethereum and Bitcoin. Such transactions are technically called *'Claims'*. The *Claim* is a basic data entity in the Userfeeds Platform. It may represent an *endorsement*, a *like*, an *upvote* for a given URL / Text / Identifier which is signed by one of the supported signing methods. You can *sign* a *Claim* with the ECDSA and send it through the HTTP or make an Ethereum transaction with data of *Claim* which then will be treated as signed.

Rankings for transactions can be obtained through the utilization of algorithms which are actually pre-defined queries designed for searching through Ethereum. Those queries can also be seen as a set of tools supporting the interface. Examples of content rankings are: a list of products on amazon.com, songs in a playlist on spotify, news on NYTimes, and all other places which have some content (links, articles, songs) sorted in a particular way which you can access.

It is also possible to track financial information (records) regarding particular users and transactions via smart contracts which are a sort of token defined for information storing.

In order to make full use of the *API* and all algorithms it is essential that you have a wallet and an account for the Etherum crypto-currency created. Currently Ethereum, Bitcoin and IPDB are supported on the platform.

For easy set-up of both the wallet and the account, the MetaMask service can be used available at https://metamask.io/.

For more information on crypto-currencies, visit topic-related websites.

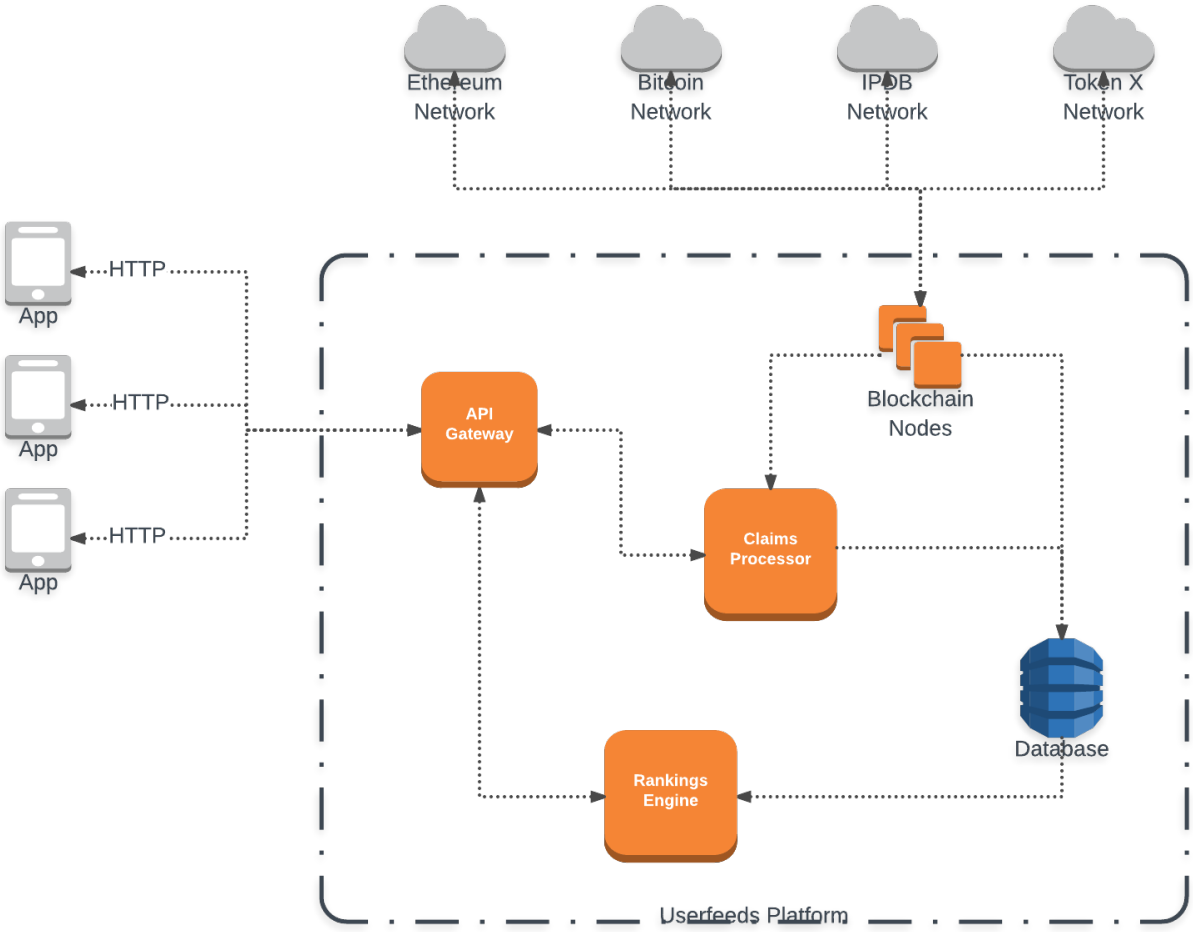
In order to become familiar technically with the Userfeeds Platform and be able to use it efficiently, go to *Quick Start - Guide for Developers* and *API Reference* to see how to use our 'read-only' APIs and how to use the Ethereum blockchain for pushing information into the Userfeeds Platform. Later on you can go to http://api.userfeeds.io/portal/, and register as a developer, and go to http://api.userfeeds.io/portal/apis/ API Catalog to request for an API Key. When the API Key is ready for you - start using all of our HTTP API's.

Our mission - as owners of the Userfeeds Platform - is to help developers in obtaining information on rankings and reliable scores (per needed perspective) as well as earning money for promotional actions on web sites by providing

them with 'ready-to-use' API's, libraries and algorithms which they can incorporate in an easy way into their software and web pages.

1.1 How does the Userfeeds Platform work?

The Userfeeds Platform is composed of a couple of parts which interact together in order to deliver rankings through *HTTP API's* to your application.



It is important to remember that any internet transaction is a claim. Such *Claim* is a basic information package signed cryptographically (with use of unique key) and connected with an account - in the json format - key - claim / key - target.

The structure of a claim can be presented in the following way:

from who ; **for whom** ; **how much / amount** ; **transaction identifier**.

The claim can also be understood as metadata for any transaction.

Claims created with help of our algorithms land on the Ethereum blockchain from where they are collected and read. *Claims* can have extensions which make them a particular type.

First it is the Userfeeds Platform which reads all the information from supported blockchains and stores it in an internal database. We store all transactions and contract calls in the Graph database and current balances in the standard SQL database. When someone sends *Claim* data to the Userfeeds Platform through the HTTP or through the

Ethereum network, it is processed and inserted into the Graph database for further use in ranking algorithms. When an application makes a HTTP request to our Ranking API endpoint, a Ranking Engine takes the requested algorithm and applies it to the current data stored in the Graph database and the SQL database and returns sorted entries for application to display to the user.

1.2 Why use the Userfeeds Platform?

- All our algorithms are of Open Source type and therefore can be improved any time when an issue arises or changed per current needs
- Everyone is able to create their own custom rankings
- We use the blockchain as the source of ranking signals, for example - how many tokens are connected to given content(s), how stable were token holders endorsing given content(s), how involved they are in given token(s).
- Everyone can monetize their application / site easily by providing *sponsored* ranking on their application / site
- Everyone can deliver superior experience for their users and visitors by customizing our ranking output basing on your needs
- Introduction of promotional elements and activities to any web service is very easy with our algorithms for web components
- Everyone can make their blogs more engaging by bringing widgets for promoting

CHAPTER 2

Quick Start - Guide for Developers

2.1 How to get all links connected with Ethereum

The code snippets will return a list of news shared to Ethereum context sorted according to our *simple* algorithm.

Demo:

2.1.1 cURL

```
$ curl 'https://api.userfeeds.io/ranking/links;asset=ethereum'
```

2.1.2 In JavaScript (browser)

```
<!DOCTYPE html>
<html>
<head>
  <title>Simple ranking for Ethereum:</title>
</head>
<body>
  <script>
    fetch('https://api.userfeeds.io/ranking/links;asset=ethereum')
    .then(function(response) {
      return response.json();
    })
    .then(function(ranking) {
      var div = document.getElementById('ranking');

      for (var i = 0; i < ranking.items.length; i++) {
        var item = ranking.items[i];
        div.innerHTML += i.toString() + '.<a href="' + item.target + '">' + item.
↪target + '</a><br/>';
```

(continues on next page)

(continued from previous page)

```
        div.innerHTML += 'Score: ' + item.score + '<br/><br/>';
      }
    });
  </script>
  <h1>Simple ranking for Ethereum</h1>
  <div id="ranking"></div>
</body>
</html>
```

2.1.3 In JavaScript (node.js)

```
const https = require('https');

const options = {
  hostname: 'api.userfeeds.io',
  port: 443,
  path: '/ranking/links;asset=ethereum',
  method: 'GET'
};

console.log('Simple ranking for Ethereum:\n')

const req = https.request(options, (res) => {
  let data = "";

  res.setEncoding('utf8');
  res.on('data', (chunk) => {
    data += chunk;
  });
  res.on('end', () => {
    let ranking = JSON.parse(data);
    for (let index in ranking.items) {
      console.log(`${index}. ${ranking.items[index].target}`);
      console.log(`Score: ${ranking.items[index].score}\n`);
    }
  });
});

req.on('error', (e) => {
  console.error(e);
});

req.end();


// Simple ranking for Ethereum:
//
// 0. http://example.com/one
// Score: 123.1234324
//
// 1. http://example.com/two
// Score: 32.234542343
```

2.1.4 In python

```
import requests

RANKING_URL = "https://api.userfeeds.io/ranking/links;asset=ethereum"

response = requests.get(RANKING_URL).json()

print("Simple ranking for Ethereum:\n")

for index, item in enumerate(response['items']):
    print("{0}. {1}".format(index, item["target"]))
    print("Score: {0}".format(item['score']))
    print()


# Simple ranking for Ethereum:
#
# 0. http://example.com/one
# Score: 123.1234324
#
# 1. http://example.com/two
# Score: 32.234542343
```

2.2 How to get all messages from ERC721 token (such as CryptoKitty Captain Barbosa)

Demo:

2.2.1 cURL

```
$ curl 'https://api.userfeeds.io/ranking/feed;
↪context=ethereum:0x06012c8cf97bead5deae237070f9587f8e7a266d:134330'
{"items":[{"about":null,"abouted":[],"author":
↪"0x6be450972b30891b16c8588dcbc10c8c2aef04da","context":"ethereum:0x0...
```

2.2.2 In JavaScript (browser)

```
<!DOCTYPE html>
<html>
<head>
  <title>Simple ranking for Ethereum:</title>
</head>
<body>
  <script>
    fetch('https://api.userfeeds.io/ranking/cryptopurr_feed;
↪context=ethereum:0x06012c8cf97bead5deae237070f9587f8e7a266d:134330')
    .then(function(response) {
      return response.json();
    })
    .then(function(ranking) {
```

(continues on next page)

(continued from previous page)

```
      var div = document.getElementById('ranking');

      for (var i = 0; i < ranking.items.length; i++) {
        var item = ranking.items[i];
        div.innerHTML += i.toString() + '. ' + item.target.id + '<br/>';
      }
    });
  </script>
  <h1>Simple ranking for Ethereum</h1>
  <div id="ranking"></div>
</body>
</html>
```

2.2.3 In JavaScript (node.js)

```
const https = require('https');

const options = {
  hostname: 'api.userfeeds.io',
  port: 443,
  path: '/ranking/cryptopurr_feed;
↪context=ethereum:0x06012c8cf97bead5deae237070f9587f8e7a266d:134330',
  method: 'GET'
};

console.log('Simple ranking for Ethereum:\n')

const req = https.request(options, (res) => {
  let data = "";

  res.setEncoding('utf8');
  res.on('data', (chunk) => {
    data += chunk;
  });
  res.on('end', () => {
    let ranking = JSON.parse(data);
    for (let index in ranking.items) {
      console.log(`${index}. ${ranking.items[index].target.id}`);
    }
  });
});

req.on('error', (e) => {
  console.error(e);
});

req.end();


// Simple ranking for Ethereum:
//
// 0. http://example.com/one
// Score: 123.1234324
//
// 1. http://example.com/two
```

(continues on next page)

(continued from previous page)

```
// Score: 32.234542343
```

2.2.4 In python

```
import requests

RANKING_URL = "https://api.userfeeds.io/ranking/cryptopurr_feed;
↪context=ethereum:0x06012c8cf97bead5deae237070f9587f8e7a266d:134330"

response = requests.get(RANKING_URL).json()

print("Simple ranking for Ethereum:\n")

for index, item in enumerate(response['items']):
    print("{0}. {1}".format(index, item["target"]["id"]))
    print()
```

2.3 How to get all bots (ERC721 tokens) owned by given address

How to get all CryptoBots owned by given address

2.3.1 cURL

```
$ curl 'https://api.userfeeds.io/ranking/tokens;
↪identity=0x6be450972b30891b16c8588dcbc10c8c2aef04da;
↪asset=ethereum:0xf7a6e15dfd5cdd9ef12711bd757a9b6021abf643'
{"items":[{"sequence":5289388,"token":"4085"}]}
```

CHAPTER 3

Web Components

3.1 Button

…

CHAPTER 4

API Reference

The API root is available at https://api.userfeeds.io/

4.1 Retrieving Data

read-only

The available algorithms are described in the *Algorithms* section.

```
Schema:

$ curl https://api.userfeeds.io/ranking/algorithm1Name;param1=value1;param2=value2.../
↪algorithm2Name...

An example:

$ curl https://api.userfeeds.io/ranking/links;asset=ethereum
```

`GET /ranking/algorithm1Name;param1=value1;param2=value2.../algorithm2Name...`

An example request:

An example response:

OR

```
Schema:

$ curl -X POST -v -d '{"flow":[{"algorithm":"algorithm1name","params":{"param1":
↪"value1",...},...}]}' -H 'Content-Type: application/json' 'https://api.userfeeds.io/
↪ranking/'

An example:

$ curl -X POST -v -d '{"flow":[{"algorithm":"links","params":{"asset":"ethereum"}}]}'␣
↪-H 'Content-Type: application/json' 'https://api.userfeeds.io/ranking/'
```

(continues on next page)

(continued from previous page)

POST /ranking/

An example request:

An example response:

CHAPTER 5

Algorithms

The Userfeeds Platform allows running custom algorithms on the top of data gathered from all supported sources such as Ethereum blockchain (mainnet, ropsten, rinkeby, kovan)

Algorithms can be referenced by their identifier which depends on how the author has decided to share them.

Note: The support is currently limited to built-in algorithms created by Userfeeds only. We will open custom algorithms for external developers in the near future.

5.1 Available built-in algorithms

CHAPTER 6

Indexes and tables

- genindex
- modindex
- search

HTTP Routing Table

/ranking