Pendulum Intelligence

506 2nd Ave, Ste. 3100 Seattle, WA 98104
https://www.pendulumfn.com/

API Reference Documentation

For

Release Date: 10-2-2024
Document Reference Number: v1.A

Section 1. API Documentation

Explore Feed

[POST] https://api.pendulumintel.com/explore/feeds

NOTE: The URL above is an api only endpoint and will just show a 4xx error if one tries to directly access the url with the Authentication Header(s).

NOTE: No other public-facing API interfaces are available in the application besides the one shared as part of this reference document.

Query Params

- q | string

```
Defines a collection of filters pertaining to the feed being explored
Include a filter expression in the query string of the URI, using the following
syntax:
* filter=field:operator:value
* Refer to the top level documentation for each entity to identify field names
and operators which may be used in the filter expression.
* Operators are delimited with colons
* If any reserved characters following the question mark (?) in the URI are
encoded, than all reserved characters must be encoded. For example, the colon
character (:) would be encoded as %3A and the equals character (=) would be
encoded as %3D.
* The operators are:
      * eq-equals
      * gt-greater than
      * gte-greater than or equal
      * has-contains the specified string
      * lt-less than
      * lte-less than or equal
      * in-any of [list] (for searching tags)
* Field names, operator names, and values are case sensitive.
* Values should be URL-encoded
* To filter on multiple fields, combine expressions using a comma, as in this
example:
      * q=start_date:gte:2024-04-09,end_date:lte:2024-05-12,risk_score:gte:0
* For date values, use [ISO 8601](https://en.wikipedia.org/wiki/ISO_8601)
format (for example, 2024-06-04).
```

- sort | string

```
The value of this parameter is a comma-separated list of sort keys.
```

```
Sort directions can optionally be appended to each sort key, separated by the
‘:’ character. The supported sort directions are either ‘asc’ for ascending or
‘desc’ for descending.
The caller may (but is not required to) specify a sort direction for each key.
If a sort direction is not specified for a key, then a default is set by the
server.
* Sample Sort
       * sort=upload_date:desc
* Supported sort Fields:
       * upload_date
       * risk_score
       * impression_count
```

- per_page | int
  `Number of results to return per page.Defaults to 25`

- page | int
  `Page number to query results for. Defaults to 1`

Request Body

```
{
   "conditions": [
       {
           "keywords_expression": "\"cars\""
       }
   ]
}
```

The Payload above represents the JSON body that the POST request expects.
`keyword_expression` may be a boolean expression with AND/OR Operators added to it.

Pendulum uses the following operators and it is important that they are all UPPERCASE when included.

AND: search for the presence of all terms in your search.
OR: search for the presence of either term.
NOT: remove items that contain a specific term.

Some search examples:
Hans Niemann and Magnus Carlsen recent scandal:

(((("Hans Demon" OR "Hans Niemann" OR ((Hans OR Niemann OR Neiman OR Nieman) AND Chess)) OR ("Magnus Carlsen" OR ((Magnus OR Carlsen OR Carlson OR Carlton) AND Chess))) AND (Cheat* OR Scandal OR Withdr* OR Drama OR Device OR Vibr*))

Response

200 - OK

```
{
   "total_count": # Total number of snippets matching a search,
   "items": [
        {
            "snippet_text": #Text of the Snippet,
            "snippet_offset": # Optional - Present if the snippet belongs to a
transcript,
            "snippet_posted_datetime": # Upload date of the post associated with the
snippet,
            "total_views": # Total number of views (if available) of the post
            associated with the snippet,
            "post_link": # Link to the associated post,
            "platform_name": # Social Media platform the post belongs to,
            "post_title": # Title of the post, if available,
            "post_description": # Description of the post, if available,
            "creator_name": # Screen Name of the content creator,
            "creator_link": # Link to the profile of content creator,
            "creator_total_followers": # Total number of followers of the content
creator
        },
}
```

400 - Bad Request

- Implies that either the auth token was invalid or the request is malformed.

Code Sample

The code sample below is an illustration of how the api may be called and then paginated. It the the following building blocks/steps:

1) Read the CLIENT_ID, CLIENT_SECRET, API_KEY as an environment variable.
2) Call the Auth endpoint using the CLIENT_ID & CLIENT_SECRET to fetch an id token.
3) Use the Id Token to make the actual call to the pendulum explore api
4) Specify the date range and number of results per page to return using the query params
5) Specify the search term using the request body
6) Make the API call and fetch results
7) If the total number of results is more than the number of results per page, then continue to query the next page of results until all the results are collected.

```
import os
import json
import requests
from requests.auth import HTTPBasicAuth

client_id = os.environ.get("CLIENT_ID")
client_secret = os.environ.get("CLIENT_SECRET")
token_url = 'https://auth.pendulumintel.com/oauth2/token'
scope = "api.pendulumintel.com/explore.api.read"
api_key = os.environ.get("API_KEY")
ap_endpoint = "https://api.pendulumintel.com/explore/feeds"


def get_access_token():
   """Fetch the OAuth Token"""

   # Obtain access token
   response = requests.post(
       token_url,
       headers={'Content-Type': 'application/x-www-form-urlencoded'},
       auth=HTTPBasicAuth(client_id, client_secret),
       data={
           'grant_type': 'client_credentials',
           'scope': scope
           }
   )
```

```
    if response.status_code == 200:
        return response.json().get('access_token')
    else:
        print('Failed to obtain access token:', response.status_code, response.text)
        return None


def query_pendulum():
    """Make the API Request"""

    # Define the initial query parameters
    query_params =
"start_date:gte:2024-07-01,end_date:lte:2024-08-31,risk_score:gte:0,risk_score:lte:100
"
    sort_param = "upload_date:desc"
    per_page = 10
    page = 1

    # Set the authorization token (replace 'your_token' with your actual token)
    auth_token = get_access_token()

    # Set the headers, including the Authorization header
    headers = {
        "Authorization": auth_token,
        "x-api-key": api_key,
        "Content-Type": "application/json"
    }

    # Initialize an empty list to store all results
    all_results = []

    while True:
        # Define the data for the current request
        params = {
            "q": query_params,
```

```
            "sort": sort_param,
            "per_page": per_page,
            "page": page
        }
        condition_payload = {
            "conditions": [
                {"keywords_expression": "elections AND 2024 AND Washington"}
            ]
        }

        # Make the POST request
        response = requests.post(ap_endpoint, headers=headers, params=params,
data=json.dumps(condition_payload))

        # Check if the request was successful
        if response.status_code == 200:
            response_data = response.json()

            # Add the current page's results to the list
            all_results.extend(response_data.get('items', []))

            # Calculate total pages based on total_count and per_page
            total_count = response_data.get('total_count', 0)
            total_pages = (total_count // per_page) + (1 if total_count % per_page > 0
else 0)

            print(f"Processing page {page} of {total_pages}")

            # Check if we've processed all pages
            if page >= total_pages:
                break

            # Increment the page number for the next iteration
            page += 1
        else:
```

```
        print(f"Request failed with status code {response.status_code}")
        break

# After the loop, all results are stored in all_results
print(f"Total results retrieved: {len(all_results)}")
```