Search Tweets

Premium search APIs

Introduction

There are two premium search API endpoints:

- Search Tweets: 30-day endpoint → provides Tweets posted within the last 30 days.
- Search Tweets: Full-archive endpoint → provides Tweets from as early as 2006, starting with the first Tweet posted in March 2006.

These search API endpoints share a common design and the documentation below applies to both. Note that although the premium APIs share identical 'API method/parameter' details with the enterprise APIs, they do use a different authentication method.

Below you will find important details needed when integrating with the premium search APIs:

- Methods for requesting Tweet data and counts
- Authentication
- Pagination
- API request parameters and example requests
- API response JSON payloads and example responses
- HTTP response codes

The premium APIs provide low-latency, full-fidelity, query-based access to the Tweet archive. The only difference in the two APIs is the time frame you can search, either the previous 30 days or Tweets from as early as 2006. Time frames can be specified with minute granularity. Tweet data is served in reverse chronological order, starting with the most recent Tweet that matches your query. Tweets are available from the search API approximately 30 seconds after being published.

Methods

The base URI for the premium search API is `https://api.twitter.com/1.1/tweets/search/`.

Premium search provides both *data* and *counts* endpoints. Note that the counts endpoint is not available in *Sandbox* dev environments.

Method	Description
POST /search/:product/:label	Retrieve Tweets matching the specified query.
POST /search/:product/:label/counts	Retrieve the number of Tweets matching the specified query.

Where:

- `:product` indicates the search endpoint you are making requests to, either `30day` or `fullarchive`.
- `:label` is the (case-sensitive) label associated with your search developer environment, as displayed at https://developer.twitter.com/en/account/environments.

For example, if using the 30-day endpoint and your dev environment has a label of 'dev' (short for development), the search URLs would be:

- Data endpoint providing Tweets:
  `https://api.twitter.com/1.1/tweets/search/30day/dev.json`
- Counts endpoint providing *counts* of Tweets:
  `https://api.twitter.com/1.1/tweets/search/30day/dev/counts.json`

Below there are several example requests using a simple HTTP utility called curl. These examples use URLs with `:product` and `:label` tokens. To use these examples, be sure to update the URLs with your details.

Authentication

You have two *application-only* OAuth options when authenticating with the premium search APIs:

- Authentication with a single Bearer token.

- Authentication using consumer token and secret.

If you are new to OAuth, be sure to check out our documentation.

OAuth 2 with Bearer Token

If you are completely new to Twitter search APIs and/or OAuth, Bearer Token authentication is a good place to start. Authentication is performed by passing a Bearer Token as an HTTP request header. While this application-only authentication requires you to first generate a token, after that it is easy to start making search requests with tools like curl and HTTPie. When using these types of HTTP tools you can exercise the API with a single terminal command.

Bearer Tokens are based on Twitter app consumer tokens. For a curl-based recipe for generating Bearer Tokens, see HERE. Generating bearer tokens is equally straightforward using the language of your choice. HERE is an example in Ruby.

Request examples below use Bearer Tokens.

OAuth consumer key and secret

If you have already integrated with the standard search API, with application-only consumer key and secret, your code base should be ready to authenticate with the Search Tweets: 30-day endpoint. That code base is likely using a language-specific OAuth package that abstracts away all the underlying 'handshake' details. With these packages, the authentication path typically means configuring your keys and tokens, creating some sort of HTTP object, then making requests with that object.

If you are building your first app using this form of OAuth, we recommend that you find an OAuth package for your integration language of choice and start there. For example, this Node.js simple scriptfor the Search Tweets: 30-day endpoint uses the Node 'request' package and makes a POST request. For this example, Twitter app keys and tokens are stored as environment variables.

Request/response behavior

Using the `fromDate` and `toDate` parameters, you can request any time period that the API supports. The 30-day endpoint provides Tweets from the most recent 31 days (even though referred to as the '30-day' endpoint, it makes 31 days available to enable users to make complete-month requests). The Full-archive endpoint provides Tweets back to the very first tweet (March 21, 2006).

Each Tweet *data* request contains a 'maxResults' parameter (range 10-500, with a default of 100. Sandbox environments have a maximum of 100) that specifies the maximum number of Tweets to return in the response. When the amount of data exceeds the 'maxResults' setting (and it usually will), the response will include a 'next' token and pagination will be required to receive all the data associated with your query (see the HERE section for more information).

For example, say your query matches 6,000 Tweets over the past 30 days (if you do not include date parameters in your request, the API will default to the full 30-day period). The API will respond with the first 'page' of results with either the first 'maxResults' of Tweets or all Tweets from the first 30 days if there are less than that for that time period. That response will contain a 'next' token and you'll make another call with that 'next' token added to the request. To retrieve all of the 6,000 Tweets, approximately 12 requests will be necessary.

Pagination

When making both data and count requests it is likely that there is more data than can be returned in a single response. When that is the case the response will include a 'next' token. The 'next' token is provided as a root-level JSON attribute. Whenever a 'next' token is provided, there is additional data to retrieve so you will need to keep making API requests.

Note: The 'next' token behavior differs slightly for data and counts requests, and both are described below with example responses provided in the API Reference section.

Data pagination

Requests for data will likely generate more data than can be returned in a single response. Each data request includes a parameter that sets the maximum number of Tweets to return per request. The 'maxResults' parameter defaults to 100 and can be set to a range of 10-500 (or a maximum of 100 with Sandbox environments). If your query matches more Tweets than the 'maxResults' parameter used in the request, the response will include a 'next' token (as a root-level JSON attribute). This 'next' token can be used in a subsequent request to retrieve the next portion of the matching Tweets for that query (i.e. the next 'page'). Next tokens will continue to be provided until you have reached the last "page" of results for that query when no 'next' token is provided.

To request the next 'page' of data, you must make the exact same query as the original, including `query`, `toDate`, and `fromDate`, if used, and also include a 'next' request parameter set to the value from the previous response. This can be utilized with either a GET or POST request. However, the 'next' parameter must be URL encoded in the case of a GET request. You can continue to pass in the 'next' element from your previous query until you have received all Tweets from the time period covered by your query. When you receive a response that does not include a 'next' element, it means that you have reached the last page and no additional data are available for the specified query and time range.

Counts pagination

The 'counts' endpoint provides Tweet volumes associated with a query on either a daily, hourly, or per-minute basis. The 'counts' API endpoint will return a timestamped array of counts for a maximum of a 31-day payload of counts.

For higher volume queries, there is the potential that the generation of counts will take long enough to potentially trigger a response timeout. When this occurs you will receive less than 31 days of counts but will be provided a 'next' token in order to continue making requests for the entire payload of counts.

As with the data 'next' tokens, you must make the exact same query as the original and also include a 'next' request parameter set to the value from the previous response.

Additional notes

- When using a `fromDate` or `toDate` in a search request, you will only get results from within your time range. When you reach the last group of results within your time range, you will not receive a 'next' token.
- The 'next' element can be used with any `maxResults` value between 10-500 (with a default value of 100. Sandbox dev environments have a maximum of 100). The `maxResults` parameter determines how many Tweets are returned in each response, but does not prevent you from eventually getting all results.
- The 'next' token does not expire. Multiple requests using the same 'next' query will receive the same results, regardless of when the request is made. Be sure to always update the 'next' token as you paginate through the results.
- When paging through results using the 'next' parameter, you may encounter duplicates at the edges of the query. Your application should be tolerant of these.

Data endpoint

POST /search/:product

Endpoint pattern:

- 30-day: /search/30day/:label.json
- Full-archive: /search/fullarchive/:label.json

This endpoint returns data for the specified query and time period. If a time period is not specified the time parameters will default to the last 30 days. Note: This functionality can also be accomplished using a GET request, instead of a POST, by encoding the parameters described below into the URL.

Data request parameters

Parameters Parameters	Description Description	Required Required	Sample Value Sample Value
`query`	The equivalent of one premium rule/filter, with up to 1,024 characters (256 with Sandbox dev environments). This parameter should include ALL portions of the rule/filter, including all operators, and portions of the rule should not be separated into other parameters of the query. **Items to Note:** • Supported Operators are listed HERE.	Yes	(snow OR cold OR blizzard) weather
`tag`	Tags can be used to segregate rules and their matching data into different logical groups. If a rule tag is provided, the rule tag is included in the 'matching_rules' attribute. It is recommended to assign rule-specific UUIDs to rule tags and maintain desired mappings	No	8HYG54ZGTU

Parameters	Description	Required	Sample Value
	on the client side.		
`fromDate`	The oldest UTC timestamp (from most recent 30 days) from which the Tweets will be provided. Timestamp is in minute granularity and is inclusive (i.e. 12:00 includes the 00 minute). *Specified:* Using only the fromDate with no toDate parameter will deliver results for the query going back in time from now( ) until the fromDate. *Not Specified:* If a fromDate is not specified, the API will deliver all of the results for 30 days prior to now( ) or the toDate (if specified). If neither the fromDate or toDate parameter is used, the API will deliver all results for the most recent 30 days, starting at the time of the request, going	No	201512220000

backwards.

Parameters	Description	Required	Sample Value
`toDate`	The latest, most recent UTC timestamp to which the Tweets will be provided. Timestamp is in minute granularity and is not inclusive (i.e. 11:59 does not include the 59th minute of the hour). *Specified:* Using only the toDate with no fromDate parameter will deliver the most recent 30 days of data prior to the toDate. *Not Specified:* If a toDate is not specified, the API will deliver all of the results from now( ) for the query going back in time to the fromDate. If neither the fromDate or toDate parameter is used, the API will deliver all results for the entire 30-day index, starting at the time of the request, going backwards.	No	201712220000

	Description	Required	Sample Value
`maxResults`	The maximum number of search results to be returned by a request. A number between 10 and the system limit (currently 500, 100 for Sandbox environments). By default, a request response will return 100 results.	No	500
`next`	This parameter is used to get the next 'page' of results. The value used with the parameter is pulled directly from the response provided by the API, and should not be modified.	No	NTcxODIyMDMyODMwMjU1MTA0

Additional details

Available Timeframe	Last 31 days
Query Format	The equivalent of one PowerTrack rule, with up to 1,024 characters (256 with Sandbox dev environment). **Items to Note:** • Supported Operators are listed HERE.
Rate Limit	Request rate limits at both minute and second granularity. The per minute rate limit is 60 requests per minute (30 with Sandbox environment). Requests are also limited to 10 per second. Requests are aggregated across both the data and counts endpoints. Monthly request limits are also applied. Sandbox environments are limited to 250 requests per month, and paid access can range between 500 and 10,000 requests.
Compliance	All data delivered via the search APIs is compliant at the time of delivery.
Realtime Availability	Data is available in the index within 30 seconds of generation on the Twitter Platform

Example data requests and responses

Example POST request

- Request parameters in a POST request are sent via a JSON-formatted body, as shown below.
- All portions of the PowerTrack rule being queried for (e.g. keywords, other operators like bounding_box:) should be placed in the 'query' parameter.

- Do not split portions of the rule out as separate parameters in the query URL.

Here is an example POST (using cURL) command for making an initial data request:

```
curl -X POST "https://api.twitter.com/1.1/tweets/search/:product/:label.json"
```

Note that the rule used contains an exact phrase with quotes, so those quotes must be escaped (with a '' character) in order for the JSON to be valid.

If the API data response includes a 'next' token, below is a subsequent request that consists of the original request, with the 'next' parameter set to the provided token:

```
curl -X POST "https://api.twitter.com/1.1/tweets/search/:product/:label.json"
"next":"NTcxODIyMDMyODMwMjU1MTA0"}' -H "Authorization: Bearer TOKEN"
```

Example GET request

- Request parameters in a GET request are encoded into the URL, using standard URL encoding.
- All portions of the PowerTrack rule being queried for (e.g. keywords, other operators like bounding_box:) should be placed in the 'query' parameter.
- Do not split portions of the rule out as separate parameters in the query URL.
- These examples use a dev environment label of 'prod' and 'TOKEN' represents a Bearer token.

Here is an example GET (using cURL) command for making an initial data request:

```
curl  "https://api.twitter.com/1.1/tweets/search/:product/:label.json?query=T
```

Example data responses

Below is an example response to a data query. This example assumes that there were more than ‘maxResults’ Tweets available so a 'next' token is provided for subsequent requests. If 'maxResults' or fewer Tweets are associated with your query, no 'next' token would be included in the response.
The value of the 'next' element will change with each query and should be treated as an opaque string. The 'next' element will look like the following in the response body:

```
    "results":
      [
           {--Tweet 1--},
           {--Tweet 2--},
           ...
           {--Tweet 500--}
      ],
    "next":"NTcxODIyMDMyODMwMjU1MTA0",
    "requestParameters":
      {
        "maxResults":500,
        "fromDate":"201711010000",
        "toDate":"201711030000"
      }
}
```

The response to a subsequent request might look like the following (note the new Tweets and different 'next' value):

```
      "results":
      [
           {--Tweet 501--},
           {--Tweet 502--},
           ...
           {--Tweet 1000--}
      ],
      "next":"R2hCDbpBFR6eLXGwiRF1cQ",
      "requestParameters":
      {
        "maxResults":500,
        "fromDate":"201101010000",
        "toDate":"201201010000"
      }
}
```

You can continue to pass in the 'next' element from your previous query until you have received all Tweets from the time period covered by your query. When you receive a response that does not include a 'next' element, it means that you have reached the last page and no additional data is available in your time range.

Counts endpoint

/search/:label/counts

Endpoint pattern:

- 30-day: /search/30day/:label/counts.json
- Full-archive: /search/fullarchive/:label/counts.json

This endpoint returns counts (data volumes) data for the specified query. If a time period is not specified the time parameters will default to the last 30 days. Data volumes are returned as a timestamped array on either daily, hourly (default), or by the minute.

Counts are only an estimate. It shouldn't be expected that the count your receive will be the exact number of activities returned. However, you can and should expect that counts will always return a higher value than the number of activities returned via the data endpoints.

Note: This functionality can also be accomplished using a GET request, instead of a POST, by encoding the parameters described below into the URL.

Counts request parameters

Parameters	Description	Required	Sample Value
`query`	The equivalent of one PowerTrack rule, with up to 1,024 characters (256 with Sandbox dev environments). This parameter should include ALL portions of the PowerTrack rule, including all operators, and portions of the rule should not be separated into other	Yes	(snow OR cold OR blizzard) weather

Parameters	parameters of the query. **Description** **Items to Note:** • Supported Operators are listed HERE.	Required	Sample Value
`fromDate`	The oldest UTC timestamp from which the Tweets will be provided. Timestamp is in minute granularity and is inclusive (i.e. 12:00 includes the 00 minute). *Specified:* Using only the fromDate with no toDate parameter, the API will deliver counts (data volumes) data for the query going back in time from now until the fromDate. If the fromDate is older than 31	No	201207220000

Parameters	Description	Required	Sample Value
	the fromDate is older than 31 days from now( ), you will receive a 'next' token to page through your request. *Not Specified:* If a fromDate is not specified, the API will deliver counts (data volumes) for 30 days prior to now( ) or the toDate (if specified). If neither the fromDate or toDate parameter is used, the API will deliver counts (data volumes) for the most recent 30 days, starting at the time of the request, going backwards.		
`toDate`	The latest, most recent UTC timestamp to which the Tweets will be provided. Timestamp is in minute granularity and is not inclusive (i.e. 11:59 does not include the 59th minute of the hour). *Specified:* Using only the toDate with no fromDate parameter will deliver the most recent counts (data volumes) for 30 days prior to the toDate.	No	201208220000

Parameters	Description	Required	Sample Value
	Description: If a toDate is not specified, the API will deliver counts (data volumes) for the query going back in time to the fromDate. If the fromDate is more than 31 days from now( ), you will receive a 'next' token to page through your request. If neither the fromDate or toDate parameter is used, the API will deliver counts (data volumes) for the most recent 30 days, starting at the time of the request, going backwards.		
`bucket`	The unit of time for which count data will be provided. Count data can be returned for every day, hour or minute in the requested timeframe. By default, hourly counts will be provided. Options: "day", "hour", "minute"	No	minute
`next`	This parameter is used to get the next 'page' of results. The value used with the parameter is pulled directly from the response provided by the API, and should not be modified.	No	NTcxODIyMDMyODMwMjU1MTA0

Additional details

Available timeframe	30-day: last 31 days
Query format	The equivalent of one PowerTrack rule, with up to 1,024 characters (256 with Sandbox dev environments).

Items to Note:

- Not all PowerTrack operators are supported. Supported Operators are listed HERE.

Rate limit	Request rate limits at both minute and second granularity. The per minute rate limit is 60 requests per minute (30 with Sandbox environment). Requests are also limited to 10 per second. Requests are aggregated across both the data and counts endpoints. Monthly request limits are also applied. Sandbox environments are limited to 250 requests per month, and paid access can range between 500 and 10,000 requests.
Count precision	The counts delivered through this endpoint reflect the number of Tweets that occurred and do not reflect any later compliance events (deletions, scrub geos). Some Tweets counted may not be available via data endpoint due to user compliance actions.

Example counts requests and responses

Example POST request

- Request parameters in a POST request are sent via a JSON-formatted body, as shown below.
- All portions of the PowerTrack rule being queried for (e.g. keywords, other operators like bounding_box:) should be placed in the 'query' parameter.
- Do not split portions of the rule out as separate parameters in the query URL.

Here is an example POST (using cURL) command for making an initial counts request:

```
curl -X POST "https://api.twitter.com/1.1/tweets/search/:product/:label/count
```

If the API counts response includes a 'next' token due to very high data count volumes, below is a subsequent request that consists of the original request, with the 'next' parameter set to the provided token:

```
curl -X POST "https://api.twitter.com/1.1/tweets/search/:product/:label/count
"next":"YUcxO87yMDMyODMwMjU1MTA0"}' -H "Authorization: Bearer TOKEN"
```

Example GET request

- Request parameters in a GET request are encoded into the URL, using standard URL encoding.
- All portions of the PowerTrack rule being queried for (e.g. keywords, other operators like bounding_box:) should be placed in the 'query' parameter.
- Do not split portions of the rule out as separate parameters in the query URL.

Here is an example GET (using cURL) command for making an initial counts request:

```
curl -u<username> "https://api.twitter.com/1.1/tweets/search/:product/:label/
```

Example counts responses

Below is an example response to a counts (data volume) query.

```
  "results": [
    { "timePeriod": "201701010000", "count": 32 },
    { "timePeriod": "201701020000", "count": 45 },
    { "timePeriod": "201701030000", "count": 57 },
    { "timePeriod": "201701040000", "count": 123 },
    { "timePeriod": "201701050000", "count": 134 },
    { "timePeriod": "201701060000", "count": 120 },
    { "timePeriod": "201701070000", "count": 43 },
    { "timePeriod": "201701080000", "count": 65 },
    { "timePeriod": "201701090000", "count": 85 },
    { "timePeriod": "201701100000", "count": 32 },
    { "timePeriod": "201701110000", "count": 23 },
    { "timePeriod": "201701120000", "count": 85 },
    { "timePeriod": "201701130000", "count": 32 },
    { "timePeriod": "201701140000", "count": 95 },
    { "timePeriod": "201701150000", "count": 109 },
    { "timePeriod": "201701160000", "count": 34 },
    { "timePeriod": "201701170000", "count": 74 },
    { "timePeriod": "201701180000", "count": 24 },
    { "timePeriod": "201701190000", "count": 90 },
    { "timePeriod": "201701200000", "count": 85 },
    { "timePeriod": "201701210000", "count": 93 },
    { "timePeriod": "201701220000", "count": 48 },
    { "timePeriod": "201701230000", "count": 37 },
    { "timePeriod": "201701240000", "count": 54 },
    { "timePeriod": "201701250000", "count": 52 },
    { "timePeriod": "201701260000", "count": 84 },
    { "timePeriod": "201701270000", "count": 120 },
    { "timePeriod": "201701280000", "count": 34 },
    { "timePeriod": "201701290000", "count": 83 },
    { "timePeriod": "201701300000", "count": 23 },
    { "timePeriod": "201701310000", "count": 12 }
   ],
  "totalCount":2027,
  "requestParameters":
    {
      "bucket":"day",
      "fromDate":"201701010000",
      "toDate":"201702010000"
    }
}
```


Note that Tweet count requests for very high volume queries may result in a 'next' token. You can continue to pass in the 'next' element from your previous query until you have received all counts from the query time period. When you receive a response that does not include a 'next' element, it means that you have reached the last page and no additional counts are available in your time range.

HTTP response codes

Status	Text	Description
200	OK	The request was successful. The JSON response will be similar to the following:
400	Bad Request	Generally, this response occurs due to the presence of invalid JSON in the request, or where the request failed to send any JSON payload.
401	Unauthorized	HTTP authentication failed due to invalid credentials.
403	Forbidden	You are not authorized to use a specific resource (e.g. an operator). For example, you will receive this error message if you try to use the `is:reply` operator with the sandbox version of the Search API, as this operator is only available to use with the paid premium and enterprise search API endpoints.
404	Not Found	The resource was not found at the URL to which the request was sent, likely because an incorrect URL was used.
422	Unprocessable Entity	This is returned due to invalid parameters in the query -- e.g. invalid PowerTrack rules.
429	Unknown Code	Your app has exceeded the limit on connection requests. The corresponding JSON message will look similar to the following:
500	Internal Server Error	There was an error on the server side. Retry your request using an exponential backoff pattern.
502	Proxy Error	There was an error on the server side. Retry your request using an exponential backoff pattern.
503	Service Unavailable	