Twitter Events API Endpoint - beta

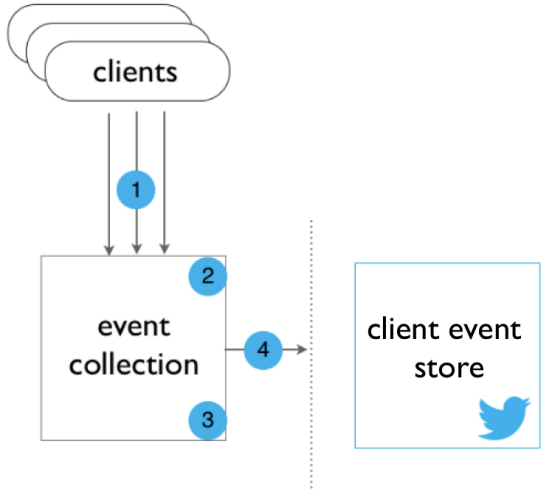
Developer Guide

November 2014

Background

Impressions and Engagements are two key aspects of Twitter syndication. These metrics help measure the true reach of user content. To support the capture and aggregation of these metrics, Twitter has implemented a logging endpoint (Event API) which provides key Partners a mechanism to capture and propagate this data back to Twitter.

High-level flow



1) Capture client impressions and engagements
2) Gather and persist events in a centralized store
3) Serialize events into JSON object containing the collection of events
4) POST JSON to Twitter's partner_event endpoint

API Details

The Event API is a **REST JSON** API for posting bulk **impression and engagement events** to Twitter

Endpoint

https://api.twitter.com/1.1/jot/partner_event

POST { "events": [event1,...eventN] } to https://api.twitter.com/1.1/jot/partner_event

Format

The POST body is a **JSON** object containing an **array** of partner events in the "events" element. Note: This means even singular partner events need to be wrapped in an array.

Authorization

The API is authenticated with application-only oauth. (two-legged oauth)

As an example, a Bearer token can be generated through the following curl:
curl -X POST "https://api.twitter.com/oauth2/token" -d "grant_type=client_credentials" -u consumerKey:consumerSecret

The bearer token is then used in the Authorization header:
curl -X POST -H "Authorization: Bearer AAAAAACJASKSKCJXLCAHXHXXAASS" ...

Example API Call

Once you have your Bearer Token, make a POST with **application/x-www-form-urlencoded** as the Content-Type:

```
curl -d '{ "events": [{            \
   "id": "1343-3451-3513-9DDD",    \
   "entity_id":"240298423",        \
   "entity_type":"tweet",          \
   "event_type":"favorite",        \
   "application_type":"web", \
   "timestamp": 138481503900       \
}]}' https://api.twitter.com/1.1/jot/partner_event
```

Responses

If every event in the payload passes validation, simply an empty **200 OK** is returned.

However, if **any** event fails validation, or the request cannot be parsed, a **400 Bad Request** is returned with a detailed response of which events failed and why. For example:

```
{
  "field_errors": {
    "1343-3451-3513-9DDD": [
      {
        "name": "timestamp",
        "msg": "may not be null"
      },
      {
        "name": "entity_type",
        "msg": "invalid value"
      }
    ]
  }
}
```

If **any** event is missing an **id**, then a **400 Bad Request** is returned with Missing Id(s) in Request as we can no longer group the errors by events meaningfully.

Errors

A best effort will be made to service all valid requests, but if the server cannot for any reason, a **500 Server Error** is returned. We encourage retries with exponential backoff, if possible. Due to the usage of **id** for each individual event, the endpoint can be considered **idempotent**.

Partner Event Metadata

Field	Description	Example	Required?
id	unique identifier of event provided by partner (in order to guarantee uniqueness across partners, we require the partner to prefix the ID with the corresponding twitter AppID.)	1343-3451-3513-9DDD	Y
entity_id	The Twitter-assigned identifier of the Twitter object (e.g. id_str or user_id)) being displayed or clicked. Use “0” if no identifier is available, e.g. for a hashtag.	8964936277	Y
entity_type	Twitter Entity. Valid types include: • tweet • user • hashtag • trend • trending_list_tv • trending_object_tv • main_menu • music_chart_140 • music_chart_last24 • music_chart_emerging	tweet	Y
event_type	Recorded action. Valid actions include: • view • retweet • favorite • click • click_url_in_tweet • reply • follow • navigate_up • navigate_down	view	Y

	• navigate_left • navigate_right • exit • click_play • follow		
device_id	Unique device identifier, eg: Android ID, Android AdID or Apple IDFA.	2b6f0cc904d137be2e1730235f5664094b831186	Y
user_agent	User Agent string	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_3) AppleWebKit/535.19 (KHTML, like Gecko) Chrome/18.0.1025.151 Safari/535.19	
application_type	There are five valid application types: • web • native • tv_oem • mobile_oem • mobile_app	web	Y
application_location	location of the viewed tweet.	top	
property_id	Partner application name and property the application is running on: • [application name].[property]		Y
mcc_mnc	Operator / carrier Mobile Country Code and Mobile Network Code, format: [mcc].[mnc]	404.02	
twitter_user_id	Logged in viewer viewing this tweet		

user_id	Hashed Partner's userid (Partner+ID) to maintain uniqueness		
hashtag	For hashtag impression: String of the hashtag that is tracked.	#TheBachelor	
trend	For trend impression: String of the trend that is tracked	#Sharknado2	Y
context_referrer	The URL of the referrer to the current page	http://url-that-took-you-to-url-that-showed-tweet.com	
url	The URL of the page on which the tweet is displayed	http://url-that-showed-the-tweet.com	
timestamp	The time at which the display occurred, in milliseconds since the Unix epoch GMT	1384815039000 *for the time: Mon, 18 Nov 2013 22:50:39 GMT*	Y
language_code	Language of the user encoded as 2 letter ISO 639-1 format.	en	
(location) country_code	Country location of the user encoded as 2 letter ISO 3166-1 format.	us	
(location) ip	The IP address of the user requesting the page containing the tweets	100.100.100.1	
(location) lat	The geographic latitude of the requesting user	38.056332	
(location) lon	The geographic longitude of the requesting user	-122.655837	
search_query	The query that led to this tweet being displayed	cool music	
(media) id	Unique identifier for the displayed	285989016-	

	media content	0F3AAEEDA9351A79C04ED7B0004C37FF	
(media) id_type	Type of media content: • tv • music • movie • telecast	music	
(media) subtype	Media classifier which is either: • live • recorded	recorded	
(media) provider	Name of service providing the media	rdio	
(media) channel	Textual or numeric description of the media channel providing the content	34	
(media) date	Date of the provided media in milliseconds since the Unix epoch GMT	1182815039000	
(media) album	Name of the album	until the quiet	
(media) artist	Artist name	flying lotus	

Example Event (required fields only)


```
{
    "id": "1343-3451-3513-9DDD",
    "entity_id": "1234",
    "entity_type": "tweet",
    "event_type": "click",
    "application_type: "web",
    "timestamp": 1384815039000
}
```


Example Event (sample)

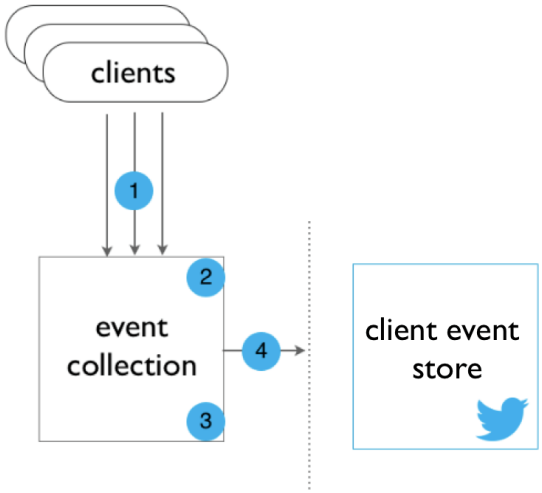

```
{
    "id" : "an-event",
    "timestamp" : 1384815039000,
    "entity_id" : "1234",
    "entity_type" : "tweet",
    "event_type" : "favorite",
    "device_id" : "12908-351-353-313-351-3",
    "user_agent" : "Iphone 5.0",
    "application_type" : "native",
    "application_location" : "top/banner",
    "property_id" : "bing",
    "twitter_user_id" : "34143413515",
    "context_referrer" : "homepage/hot-news",
    "url" : "http://example.com/news-article-here",
    "language_code" : "fr",
    "search_query" : "cool articles",
    "location" : {
        "country_code" : "fr",
        "ip" : "12.31.33.135",
        "lat" : "1.133141",
        "lon" : "2.35135"
    },
    "media" : {
```


```
        "id" : "12312",
        "id_type" : "album",
        "subtype" : "cd",
        "provider" : "amazon",
        "channel" : "music",
        "date" : 1182815039000,
        "album" : "Make it Big",
        "artist" : "Wham!"
    }
}
```


Integration Patterns

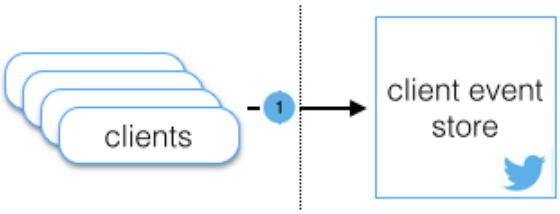
The Event API has been modeled to support two integration patterns: 1) Server to Server and 2) Client to Server.

Server to Server is the recommended integration design. While this increases latency in terms of event collection, it provides greater scalability and simplicity from an integration standpoint. Per the earlier visualization, the pattern is quite straightforward:



1) clients capture impression events and push these to a centralized store
2) the centralized store collects the inbound events into collections
3) the event collections are serialized into JSON
4) JSON events are pushed to the Twitter Event API endpoint.

Client to Server is another integration design. This pattern supports disparate clients which have no centralized store, eg: mobile applications. This pattern, while lower latency, scales less efficiently as events can't be bundled across client instances. While this is not an immediate performance concern, it is a consideration for the integration design. One important note, for each client, it is recommended to cache or batch events vs. publishing in real-time, lowering the number of HTTP calls and associated bandwidth.



1) clients capture impression events, serializes to JSON and pushes to Twitter Event API endpoint

Legal Requirements

When using the Partner Event API to send events back to Twitter, please ensure following legal requirements are met.

Logged Out

Verify with your legal counsel that your Terms of Service, End User License Agreement or Privacy Policy includes language that specifically calls out that you may share usage information to 3rd party companies.

Logged In

For logged in experiences the user will already have agreed to the Twitter Terms of Service during signup and therefore not further action is necessary.