Bulk Data Pull via the Account Engagement API

This document outlines the approach and best practices for pulling large amounts of data out of Account Engagement via the API. When you need large amounts of visitor activity data, use the Export API. If you need a different object, see Query the Account Engagement API for details on how to retrieve the information using the query endpoint.

The Export API is asynchronous and efficiently scales data retrieval based on the available resources. There are more server resources available for the Export API than the query endpoint, which provides faster results without hitting concurrency and API limits to process multiple queries.

Before getting started, review the documentation for the Export API.

Because export is asynchronous, you can poll the Read endpoint of the Export API to see if the export is done processing. The time an export takes depends on how much data is being exported and the resources available for processing. Because export times can vary, we recommend that you start by setting a polling interval between 30 seconds and a few minutes.

If you run an integration regularly, set your polling intervals to the average of the time taken for a few runs of your integration. For example, if your integration runs daily and retrieves all visitor activity data for the previous day, start with a polling frequency of 1 minute. Note the export durations for a few days, then average those times to find a polling interval. In the example, over a few days you notice that your export finishes with an average duration of 15 minutes. Polling every minute is too often and polls 15 times. If you set the polling interval to 5 minutes, it polls three or four times.

If you run integrations sporadically or infrequently, try increasing the polling interval over time. For example, if you track the number of times the integration has polled, you can calculate a gradually increasing polling interval using this function:

Where `n` is the index of the poll, starting with 0

Using this function to calculate polling interval, you poll a minute the first time, 2 minutes for the second and continue to increase at larger intervals each time. This process allows the integration to check frequently soon after an export is created to catch the quick running exports but also limit the number of polls for exports that are long-running.

Remember that the Read endpoint of the Export API counts toward API limits, so these tips let you adjust how much of that limit is used by your integration.

Because the results of the export API are returned only after the query has executed, limiting the date range of the export returns results faster. A way limit date range is to break large date ranges into several smaller ranges.

For example, it's OK to create an export to retrieve a year's worth of visitor activity within a single export by specifying a year duration, like the following.

The issue is that all 12 months of data must be queried in order for the results to be returned. There are cases where your integration can use a smaller data set up front and be refined later as more data from the export becomes available. For example, an integration can work off the last month of data while waiting on a three-month range and then increasing to a twelve-month range.

In this example, you could create three different exports with the three different ranges:

Last month

Three months prior

Eight months prior

Instead of waiting for all twelve months of data to be exported at once, the same data is returned in smaller increments, so you can see the newest data sooner.

Pulling data from a query endpoint in the Account Engagement API is a straightforward, two-step process. First, log in with your Account Engagement user email, password, and user key to acquire an API key. Then, use the query endpoint for each data set to pull the data.

Before getting started, review the object documentation and determine which data set you are going to pull. For a data set to be eligible for a bulk data pull, it must have a query operation that supports at least one of the following three sets of search filter criteria parameters:

`created_before` and `created_after`

`updated_before` and `updated_after`

`id_greater_than` and `id_less_than`

Issue an HTTP GET to the appropriate query endpoint to pull the data back. The query response contains up to 200 rows from the data set you have requested.

For bulk data pulls, we strongly recommended that you use the `bulk` output format, which is highly optimized for performance. The `bulk` output format disables functionality that displays additional data in the API response such as nested object relationships (for example, campaign name for activities). Additionally, the bulk output format omits information about the total number of rows that matched your query, which enables the API to return a response more quickly.

For bulk data pulls, we strongly discourage use of the `offset` parameter. Instead, iterate through the data set using one of the supported search filter criteria parameters.

As you process the query results in your code, store a high water mark value like the largest `id` or most recent `created_at` or `updated_at` value. Then, on your next query, send that high water mark value as the `id_greater_than`, `created_after`, or `updated_after` search filter parameter value.

Example Request & Response (XML)

Example Request & Response (JSON)

Because using the offset parameter is discouraged, let's see how you can use a high water mark value to iterate through multiple pages of results instead. In this example, you query for all visitor activities that occurred yesterday. The `created_after` and `created_before` search filter criteria parameters are perfect for this use case.

To keep things simple, assume there were a total of 5 activities created yesterday (at time of writing, 2016-03-29) at times 02:00, 06:00, 10:00, 14:00, and 18:00. Also assume the API page size limit is 2 rather than 200.

To iterate over the results in the suggested approach, your API integration process would proceed as follows:

Log in to the Account Engagement API

Pull all visitor activity created after yesterday at midnight (`2016-03-29T00:00:00`) and before tonight at midnight (`2016-03-30T00:00:00`):

Example Request & Response (JSON)

Note: The example was edited for brevity.

The API has returned two results - the activities from 02:00 and 06:00. Since you are paging by `created_at`, set your high water mark value to the `created_at` value of the last record you pull back, `2016-03-30T06:00:00`. You have now retrieved all activities up to that time.

Pull all visitor activity created after your high water mark (`2016-03-29T06:00:00`) and before tonight at midnight (`2016-03-30T00:00:00`):

Example Request & Response (JSON)

Note: The example was edited for brevity.

The API returns two more results back - the activities from 10:00 and 14:00. Since you are paging by `created_at`, update your high water mark value to the `created_at` value of the last record you pull back, `2021-03-29T14:00:00`. You have now retrieved all activities up to that time.

Pull all visitor activity created after your high water mark (`2021-03-29T14:00:00`) and before tonight at midnight (`2021-03-30T00:00:00`):

Example Request & Response (JSON)

Note: The example was edited for brevity.

The API returns one result back, the activity from 18:00. As the max page size is 2 and the API returned one record, you know you are at the end of the list of records and do not need to query any further. If the API had returned 2 records you would repeat the query process again with a new high water mark. You know it's safe to stop when the number of records returned by the API is less than the max page size.