API DOCUMENTATION

Elements of ACLED's API

Learn more about some of the general elements of ACLED API calls

20 May 2025

On this page

ACLED's API has several endpoints that return different types of data. However, many of these endpoints have common elements that can be included in your API call to refine your data request.

ACLED's API has several endpoints that return different types of data. However, many of these endpoints have common elements that can be included in your API call to refine your data request. In this section, you can learn more about some of the general elements of ACLED API calls. Specifically, you can learn more about:

1. Response types - File extensions
2. Query types
3. Limits & pagination
4. Error messages

Response types - File extensions

A common element across all ACLED API endpoints is the read statement which is used to specify the desired file type for the returned data. As you learned in the Get started section, every API call will have a read? element after the endpoint name.

read tells the API which of the four available file types you would like to receive:

- JSON - (default)
- XML - (.xml)
- CSV - (.csv)

If you do not specify a file type, your data will be returned to you by default in JSON (.json) format. This format is great for many programming applications, but is difficult to read. If readability is a priority, you can also request the data in an easier-to-understand format, like the spreadsheet-friendly CSV or XML formats.

You can specify your desired response format by adding read?_ + {the extension of the format you want}.

For example, if you want your data returned to you in CSV format, your URL should now appear as:

https://acleddata.com/api/acled/read?_format=csv

Note: If you request the data in JSON format, you will receive some extra information with your response, such as status, count, last update, and more. You can check the expected JSON response for each endpoint in the endpoint sections of this guide.

Queries

Complex queries (|, :OR: and &)

In all of ACLED's API endpoints you can execute complex queries that allow you to request data based on multiple criteria and/or multiple query filters.

If you want to specify multiple values for a single query filter, you can do so by using the | operator to separate the desired values. For instance, if you want to request data for Argentina, Georgia, and Brazil, you would include the following query in your URL: country=Argentina|Georgia|Brazil.

If you want to filter simultaneously by multiple different query filters, you can use the & and :OR: operators to separate the filters. For example, you may want to request data from the ACLED endpoint for Argentina while also limiting the returned data to a particular disorder type like "Strategic developments" or events where there was civilian targeting. If you were to include the following query filter (disorder_type=Strategic developments&civilian_targeting=Civilian targeting) in your URL, no events would be returned because no events classified as "Strategic developments" are also classified as involving civilian targeting. Instead, you should use :OR: in place of & (i.e. disorder_type=Strategic developments:OR:civilian_targeting=Civilian targeting), meaning that you will receive data for events that are either a strategic development or involve civilian targeting (You can visit ACLED's Knowledge Base to get more information of strategic events and civilian targeting events).

Query types

The different query types

Each of ACLED's API endpoints has its own query filters (e.g. the ACLED endpoint query filters), but you can use similar code chunks to modify the query filters across all endpoints. Each query filter has a default query type that specifies how the API will filter the data based on the information you provide. A query filter may be of types such as =, >, <, LIKE, and BETWEEN.

Below you can find a table explaining each of these types:

Type Value	Type Definition
LIKE	The API looks for values like the one you requested. When using LIKE, the API adds a wildcard(*) at the beginning and end of the value you requested
%3D (=)	The API looks for an exact match of the value you requested. If you need to specify this query type using the `_where` suffix, you should write "%3D" instead of "=", as "%3D" is the URL encoding version of the equal sign (=).
>	The API looks for values greater than the value you specified
<	The API looks for values less than the value you specified
BETWEEN	The API looks for all values between two values you requested. When using BETWEEN, you must add two values separated by "|" in the query filter. E.g. "event_date=2021-01-01|2021-01-31" to get all the events in January 2021

If you would like more information, you can find an example of each query type below:

- LIKE

...&admin1=Buenos Aires&admin1_where=LIKE. This will match every record where "Buenos Aires" is part of the admin1 name. Thus, it will return events where admin1 is"Ciudad de Buenos Aires" or "Buenos Aires".

- %3D (=)

...&admin1=Buenos Aires&admin1_where=%3D. This will match every record where "Buenos Aires" is the exact name of the admin1. Thus, it will only return events where admin1 is"Buenos Aires".

- >

...&year=2021&year_where=>. This will match every record after 2021.

- <

...&year=2021&year_where=<. This will match every record before 2021.

- BETWEEN

...&year=2021|2023&year_where=BETWEEN. This will match every record from 2021 to 2023.

Note: There are some instances in which using the >, <, or BETWEEN operators work differently than expected, particularly when your query includes timestamps. If this occurs, you can instead use the following URL encoding syntax to represent each query type (Table XX).

Table 2: URL encoding syntax that can be used when querying using timestamps.

Type value	Syntax	Example
Greater than (>)	%3E	timestamp=1700000000&timestamp_where=%3E
Less than (<)	%3C	timestamp=1700000000&timestamp_where=%3C
Greater than or equal to (>=)	%3E%3D	timestamp=1700000000&timestamp_where=%3E%3D
Less than or equal to (<=)	%3C%3D	timestamp=1700000000&timestamp_where=%3C%3D
Equal to (=)	%3D	timestamp=1700000000&timestamp_where=%3D
Between	..	timestamp=1700000000..1750000000

Changing the query type

In some cases, you may want to modify a query filter to use a query type that is different from the filter's default type. This can be done in the same way across all query types and endpoints.

For instance, in the ACLED endpoint,event_date has an = type, meaning that your request will only return events that occurred on the exact date you specify in your URL. However, you may be interested in events occurring across a range of dates, meaning you need to modify the query type. You modify the query type in your URL by adding _where to the filter's name and then specifying your desired query type. For instance, imagine you want to receive data for Argentina between 2019 and 2022.

Because you are interested in the regular ACLED events dataset, you should use the ACLED endpoint. You can then use the country query filter to specify that you want to receive data only for Argentina. To filter to your desired date range, you can use the year query filter, which refers to the year in which an event took place.

Thus, your URL would look like something like this:

https://acleddata.com/api/acled/read?_format=csv&country=Argentina&year=2019|2021&year_where=BETWEEN

File size & pagination

When working with APIs, it is important to consider the size of the file that the API will return. API calls that request a very large amount of data may experience timeout errors. You can deal with very large data requests in a few ways.

Being thoughtful about the data you request (row and column choice)

First and foremost you should determine whether you need all of the data you are requesting and tailor your URL accordingly.

You can limit the amount of data you receive and reduce timeout errors by carefully considering the query filters you apply. For instance, if you are interested in all events in Sudan after 2020, but fail to include a year query filter, then you will receive all events occurring since 1997 (the beginning of ACLED data collection for Sudan). It is much easier – and you are less likely to encounter issues during download and analysis – if you filter your data during the API call rather than afterwards.

Just as you may only be interested in a small subset of all rows of ACLED data, you may only need some of the available columns. You can choose which specific columns you want by including the &fields=X|Y|Z parameter in your request URL, where X,Y,Z are the names of the columns you want to include in the response. For instance, if you are requesting data from the ACLED endpoint, and you only want to receive the event_id_cnty and event_date columns, add &fields=event_date|event_id_cnty to your URL.

https://acleddata.com/api/acled/read?_format=csv&country=Argentina&fields=event_date|event_id_cnty

Remember that being thoughtful about which rows and columns you actually need, and applying query filters and column choices appropriately, is often the best and simplest way to reduce the size of your returned file, making analyses easier and reducing timeout errors.

Adjusting the limit on the number of rows returned

If you cannot winnow down the size of your data request using query filters, then the simplest way to reduce file size is to specify the number of rows that your API call will return by including the &limit=X parameter in your request URL. The default value for ACLED's API is 5000 rows of data for dyadic calls, but you could theoretically set this parameter to any number of rows. The major drawback of using a limit to reduce file size is that you may actually want all of the events that match your query filters, rather than just a subset. For instance, if there are 3000 rows of data that match your query filter, but you include &limit=2000 in your URL, you would only receive the first 2000 rows of data and would not have access to the other 1000 rows that may be of interest.

You can avoid the issues of missing data and receive all of the data specified by your query filters in the ways outlined below.

Pagination

The suggested method for requesting large amounts of data from ACLED's API is by using pagination. Pagination is simply the process of splitting one very large API call into several smaller calls or "pages". The advantage of pagination is that it helps avoid timeout errors.

You can add pagination to your URL by including the &page=X parameter at the end of the URL, where X denotes the page number. To receive all of your data you should rerun the API call, incrementing the page number by 1 each time, until you have received all the rows you requested. For instance, if you suspect there are many events that match your query filters, you should execute an API call with a URL including &page=1. You should then repeat the call and increase the page number to 2 (i.e. &page=2), and so on, until the data request returns a number of rows less than the limit (e.g. if you request limit=5000 but you receive 4999 rows or fewer - the limit by default is 5000.).

For example, from January 2018 to December 2022, there are around 12000 events from Argentina in the ACLED event dataset. You can request these data using pagination:

— 1st Call (Returns the first 5000 rows)

https://acleddata.com/api/acled/read?_format=csv&country=Argentina&year=2018|2022&year_where=BETWEEN&page=1

— 2nd Call (Returns the second 5000 rows)

https://acleddata.com/api/acled/read?_format=csv&country=Argentina&year=2018|2022&year_where=BETWEEN&page=2

— 3rd Call (Returns fewer than 5000 rows. This indicates that you have received the final page of data matching your query filters and you do not need to continue executing API calls.)

https://acleddata.com/api/acled/read?_format=csv&country=Argentina&year=2018|2022&year_where=BETWEEN&page=3

Error messages

In the table below, you can find the error messages that you may encounter when using ACLED's API. You can also find advice on how to resolve each error. If you encounter an error that is not included in the table below, please contact ACLED access team.

Code	Message	Description
400	Sorry, unrecognized username or password.	Occurs when trying to log in with incorrect data for the /user/login API.
400	invalid_grant / The user credentials were incorrect.	Occurs when trying to log in with incorrect data for the /oauth/token API.
401	The resource owner or authorization server denied the request.	Occurs when trying to use an incorrect auth token for API requests.
403	Consent must be accepted in order to continue using the site when logged in.	Occurs when the auth user didn't apply consent.
403	Please fill in all the required fields.	Occurs when the auth user didn't fill in all the required fields in the profile.
403	Access denied	Occurs as a default message when user is not logged in, user doesn't have the administrator role, or user doesn't have the 'API' access group.
403	Invalid URL	Occurs when trying to access an incorrect URL.