Richard Holmes

https://ampersandstudio.uk/

July 2021
Query Filters

Complex Queries

Country

Commands

Read

Query Filters

Complex Queries

Region

Commands

Read

Query Filters

Complex Queries
Introduction

The following document highlights the basic steps for interacting with the Acled API. The API is RESTful in nature and is accessed via Basic HTTP(S) authentication.

API Access Detail

The full URL for accessing the API is https://api.acleddata.com/{data}/{command}, where “data” represents the type of data to be collected and “command” represents the action to be executed. See below for details regarding possible data types and usages.

API Developer Account

You must register for an API key in order to access the API. To register please visit https://developer.acleddata.com/.

Please note you should register from the device you intend to access the API from as your IP address will be linked to your account and may determine your access to the API. On completing the registration form you may be required to wait for approval or may have limited access to the API, until your account has been approved. The dashboard will display any notifications that relate to your account, that should be completed before proceeding to create an API Key. Once you are eligible to create a key you will see the following button is active, within the dashboard.

Clicking the button will generate a random API key for your account. You will only be shown the key one time and must copy and save it in a secure location. If you fail to copy the key you will need to revoke the active key and generate a new key. This does not affect your usage of the API or reset your account usage limit.
Sample API Calls and Responses

API calls may be made in any standard browser or using any programmatic language that is capable of making HTTP requests. The samples below demonstrate the URL to be called to make the request.

The following points should be noted:

- All requests will be denied without a key and email address.
- This API only uses the GET or POST request. A GET request is advised whereby all data sent will be contained within standard Query String parameter formats and URLencoded.
- All responses from the API shall be formatted as JSON unless specifically requested in either XML, CSV or TXT format.
- TXT format returns a plain text csv string.
- A limit of 500 lines of data (1000 rows for monadic) will be returned by default for ACLED and Actor data types. Larger requests may be made, however.
- All fields, specific to the data type, will be returned by default. Reduced field lists can be requested for some data types.
- ACLED data is returned in date order DESC (starting with the latest).

Response Format

By default the response is returned in JSON format but it’s possible to return the response in XML, CSV and TXT format too. In order to return another format you simply add the relevant extension to the end of the command name so the query would look like the following:

<table>
<thead>
<tr>
<th>Format</th>
<th>HTTP Request Format</th>
<th>MIME Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>JSON</td>
<td><a href="https://api.acleddata.com/%5B(data)">https://api.acleddata.com/[(data)</a>][(command)]</td>
<td>application/json</td>
</tr>
<tr>
<td>XML</td>
<td><a href="https://api.acleddata.com/%5B(data)">https://api.acleddata.com/[(data)</a>][(command)].xml</td>
<td>text/xml</td>
</tr>
<tr>
<td>CSV</td>
<td><a href="https://api.acleddata.com/%5B(data)">https://api.acleddata.com/[(data)</a>][(command)].csv</td>
<td>text/csv</td>
</tr>
<tr>
<td>TXT</td>
<td><a href="https://api.acleddata.com/%5B(data)">https://api.acleddata.com/[(data)</a>][(command)].txt</td>
<td>text/plain</td>
</tr>
</tbody>
</table>
If downloading the full ACLED dataset, do not open the CSV file with Microsoft Excel, as it contains more rows than the Excel limit. If you want to open the data in Excel, please apply filters to reduce the number of rows in the CSV file before downloading.
ACLED

This data call returns the main dataset

Commands

Read

In order to retrieve the data you must make a GET or POST request to the following URL:

https://api.acleddata.com/acled/read?key={api_key}&email={email}

Returned Data (json only)

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>status</td>
<td>int</td>
<td>A number representing the request status</td>
</tr>
<tr>
<td>success</td>
<td>boolean</td>
<td>A boolean representation on the success of the call</td>
</tr>
<tr>
<td>last_update</td>
<td>int</td>
<td>The number of hours since the last update to the data</td>
</tr>
<tr>
<td>count</td>
<td>int</td>
<td>The number of data rows returned</td>
</tr>
<tr>
<td>messages</td>
<td>array</td>
<td>An array of information messages that may require future action</td>
</tr>
<tr>
<td>data</td>
<td>array</td>
<td>The rows of data returned. For details of attributes returned in each row, see below.</td>
</tr>
<tr>
<td>filename</td>
<td>string</td>
<td>The filename that will be used for csv calls</td>
</tr>
<tr>
<td>error</td>
<td>array</td>
<td>The details of the error with a status as an integer and message as a string</td>
</tr>
</tbody>
</table>

Returned Data (json, xml, txt, csv)

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>data_id</td>
<td>int</td>
<td>A unique id for the row of data</td>
</tr>
<tr>
<td>iso</td>
<td>int</td>
<td>A numeric code for each individual country. Referenced here - ISO Country List</td>
</tr>
<tr>
<td>event_id_cnty</td>
<td>string</td>
<td>An individual identifier by number and country acronym</td>
</tr>
<tr>
<td>event_id_no_cnty</td>
<td>string</td>
<td>An individual numeric identifier</td>
</tr>
<tr>
<td>event_date</td>
<td>date</td>
<td>The date the event occurred in the format: yyyy-mm-dd</td>
</tr>
<tr>
<td>Attribute Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>year</td>
<td>int</td>
<td>The year the event occurred.</td>
</tr>
<tr>
<td>time_precision</td>
<td>int</td>
<td>A numeric code indicating the level of certainty of the date coded for the event</td>
</tr>
<tr>
<td>event_type</td>
<td>string</td>
<td>The type of conflict event</td>
</tr>
<tr>
<td>sub_event_type</td>
<td>string</td>
<td>The type of conflict sub event</td>
</tr>
<tr>
<td>actor1</td>
<td>string</td>
<td>The named actor involved in the event</td>
</tr>
<tr>
<td>assoc_actor_1</td>
<td>string</td>
<td>The named actor allied with or identifying ACTOR1</td>
</tr>
<tr>
<td>inter1</td>
<td>int</td>
<td>A numeric code indicating the type of ACTOR1.</td>
</tr>
<tr>
<td>actor2*</td>
<td>string</td>
<td>The named actor involved in the event</td>
</tr>
<tr>
<td>assoc_actor_2*</td>
<td>string</td>
<td>The named actor allied with or identifying ACTOR2</td>
</tr>
<tr>
<td>inter2*</td>
<td>int</td>
<td>A numeric code indicating the type of ACTOR2.</td>
</tr>
<tr>
<td>interaction</td>
<td>int</td>
<td>A numeric code indicating the interaction between types of ACTOR1 and ACTOR2</td>
</tr>
<tr>
<td>region</td>
<td>string</td>
<td>The region in which the event took place</td>
</tr>
<tr>
<td>country</td>
<td>string</td>
<td>The name of the country the event occurred in</td>
</tr>
<tr>
<td>admin1</td>
<td>string</td>
<td>The largest sub-national administrative region in which the event took place</td>
</tr>
<tr>
<td>admin2</td>
<td>string</td>
<td>The second largest sub-national administrative region in which the event took place</td>
</tr>
<tr>
<td>admin3</td>
<td>string</td>
<td>The third largest sub-national administrative region in which the event took place</td>
</tr>
<tr>
<td>location</td>
<td>string</td>
<td>The location in which the event took place</td>
</tr>
<tr>
<td>latitude</td>
<td>decimal</td>
<td>The latitude of the location</td>
</tr>
<tr>
<td>longitude</td>
<td>decimal</td>
<td>The longitude of the location</td>
</tr>
<tr>
<td>geo_precision</td>
<td>int</td>
<td>A numeric code indicating the level of certainty of the location coded for the event</td>
</tr>
<tr>
<td>source</td>
<td>string</td>
<td>The source of the event report</td>
</tr>
<tr>
<td>source_scale</td>
<td>string</td>
<td>The scale of the source</td>
</tr>
<tr>
<td>notes</td>
<td>string</td>
<td>A short description of the event</td>
</tr>
<tr>
<td>fatalities</td>
<td>int</td>
<td>The number of reported fatalities which occurred during the event</td>
</tr>
<tr>
<td>timestamp</td>
<td>int / date</td>
<td>The unix timestamp this data entry was last updated</td>
</tr>
<tr>
<td>iso3</td>
<td></td>
<td>A 3 character code representation of each country</td>
</tr>
</tbody>
</table>
* These attributes will be returned as a new data row if export type is monadic.
Query Filters

Each field can be searched to filter the returned data. By default each field will search by = or LIKE based on the table below. This can be changed by sending a new query string name value pair, where the name has ‘_where’ appended to it. The following table shows the default query type that will be used by each field. **The key and email address query must be included in all requests to indicate that you have registered your use of our API.**

<table>
<thead>
<tr>
<th>Query Name</th>
<th>Type</th>
<th>Query String</th>
</tr>
</thead>
<tbody>
<tr>
<td>key</td>
<td></td>
<td>{api_key}</td>
</tr>
<tr>
<td>email</td>
<td></td>
<td>{email address associated with key}</td>
</tr>
<tr>
<td>data_id</td>
<td></td>
<td>?data_id={number}</td>
</tr>
<tr>
<td>iso</td>
<td></td>
<td>?iso={number}</td>
</tr>
<tr>
<td>event_id_cnty</td>
<td>LIKE</td>
<td>?event_id_cnty={text}</td>
</tr>
<tr>
<td>event_id_no_cnty</td>
<td>LIKE</td>
<td>?event_id_no_cnty={text}</td>
</tr>
<tr>
<td>event_date</td>
<td></td>
<td>?event_date={yyyy-mm-dd}</td>
</tr>
<tr>
<td>year</td>
<td></td>
<td>?year={yyyy}</td>
</tr>
<tr>
<td>time_precision</td>
<td></td>
<td>?time_precision={number}</td>
</tr>
<tr>
<td>event_type</td>
<td>LIKE</td>
<td>?event_type={text}</td>
</tr>
<tr>
<td>sub_event_type</td>
<td>LIKE</td>
<td>?sub_event_type={text}</td>
</tr>
<tr>
<td>actor1</td>
<td>LIKE</td>
<td>?actor1={text}</td>
</tr>
<tr>
<td>assoc_actor_1</td>
<td>LIKE</td>
<td>?assoc_actor_1={text}</td>
</tr>
<tr>
<td>inter1</td>
<td></td>
<td>?inter1={number}</td>
</tr>
<tr>
<td>actor2</td>
<td>LIKE</td>
<td>?actor2={text}</td>
</tr>
<tr>
<td>assoc_actor_2</td>
<td>LIKE</td>
<td>?assoc_actor_2={text}</td>
</tr>
<tr>
<td>inter2</td>
<td></td>
<td>?inter2={number}</td>
</tr>
<tr>
<td>interaction</td>
<td></td>
<td>?interaction={number}</td>
</tr>
<tr>
<td>region</td>
<td></td>
<td>?region={number}</td>
</tr>
<tr>
<td>country</td>
<td>LIKE</td>
<td>?country={text}</td>
</tr>
<tr>
<td>admin1</td>
<td>LIKE</td>
<td>?admin1={text}</td>
</tr>
<tr>
<td>admin2</td>
<td>LIKE</td>
<td>?admin2={text}</td>
</tr>
<tr>
<td>Query Name</td>
<td>Type</td>
<td>Query String</td>
</tr>
<tr>
<td>----------------</td>
<td>--------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>admin3</td>
<td>LIKE</td>
<td>?admin3={text}</td>
</tr>
<tr>
<td>location</td>
<td>LIKE</td>
<td>?location={text}</td>
</tr>
<tr>
<td>latitude</td>
<td>=</td>
<td>?latitude={number}</td>
</tr>
<tr>
<td>longitude</td>
<td>=</td>
<td>?longitude={number}</td>
</tr>
<tr>
<td>geo_precision</td>
<td>=</td>
<td>?geo_precision={number}</td>
</tr>
<tr>
<td>source</td>
<td>LIKE</td>
<td>?source={text}</td>
</tr>
<tr>
<td>source_scale</td>
<td>LIKE</td>
<td>?source_scale={text}</td>
</tr>
<tr>
<td>notes</td>
<td>LIKE</td>
<td>?notes={text}</td>
</tr>
<tr>
<td>fatalities</td>
<td>=</td>
<td>?fatalities={number}</td>
</tr>
<tr>
<td>timestamp</td>
<td>&gt;=</td>
<td>?timestamp={number/yyyy-mm-dd}</td>
</tr>
<tr>
<td>export_type</td>
<td>=</td>
<td>?export_type={text}</td>
</tr>
<tr>
<td>iso3</td>
<td>=</td>
<td>?iso3={text}</td>
</tr>
</tbody>
</table>

**References**

For some attributes a number is required instead of text. The following reference tables provides the numeric code to be used for certain content.

### inter 1 / inter 2

<table>
<thead>
<tr>
<th>Group</th>
<th>Numeric Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>State Forces</td>
<td>1</td>
</tr>
<tr>
<td>Rebel Forces</td>
<td>2</td>
</tr>
<tr>
<td>Militia Groups</td>
<td>3</td>
</tr>
<tr>
<td>Communal / Identity Groups</td>
<td>4</td>
</tr>
<tr>
<td>Rioters</td>
<td>5</td>
</tr>
<tr>
<td>Protesters</td>
<td>6</td>
</tr>
<tr>
<td>Civilians</td>
<td>7</td>
</tr>
<tr>
<td>Foreign / Others</td>
<td>8</td>
</tr>
</tbody>
</table>

### region

<table>
<thead>
<tr>
<th>Region</th>
<th>Numeric Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Western Africa</td>
<td>1</td>
</tr>
<tr>
<td>Middle Africa</td>
<td>2</td>
</tr>
</tbody>
</table>
The ISO country list can be viewed here - ISO Country Link
All LIKE queries will include a wildcard before and after the entered text.
Multiple queries can be searched with name/value pairs separated by &. Each field will be searched using AND so all arguments must match for data to be returned.
More complex queries can be searched to include the OR clause. See Complex Queries below.
If export_type is not included it will be dyadic. For monadic export you will need to include the export_type query.

<table>
<thead>
<tr>
<th>region</th>
<th>Numeric Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eastern Africa</td>
<td>3</td>
</tr>
<tr>
<td>Southern Africa</td>
<td>4</td>
</tr>
<tr>
<td>Northern Africa</td>
<td>5</td>
</tr>
<tr>
<td>South Asia</td>
<td>7</td>
</tr>
<tr>
<td>Western Asia</td>
<td>8</td>
</tr>
<tr>
<td>Southeast Asia</td>
<td>9</td>
</tr>
<tr>
<td>Middle East</td>
<td>11</td>
</tr>
<tr>
<td>Europe</td>
<td>12</td>
</tr>
<tr>
<td>Caucasus and Central Asia</td>
<td>13</td>
</tr>
<tr>
<td>Central America</td>
<td>14</td>
</tr>
<tr>
<td>South America</td>
<td>15</td>
</tr>
<tr>
<td>Caribbean</td>
<td>16</td>
</tr>
<tr>
<td>East Asia</td>
<td>17</td>
</tr>
<tr>
<td>North America</td>
<td>18</td>
</tr>
</tbody>
</table>

Example:
?key={api_key}&email={email address}&event_id_cnty={text}&event_id_cnty_where=%3D
?key={api_key}&email={email address}&event_date={yyyy-mm-dd|yyyy-mm-dd}&event_date_where=BETWEEN
Limit Query & Pagination

By default there is a limit of 500 rows of data returned, 1000 rows if export_type = monadic. You can use the limit query name to change the default number. Setting limit as 0 will return all relevant data. It is likely returning all data will cause a timeout error and we therefore recommend using the page query instead. Each page will return 500 (1000 for monadic) rows of data.

Example:
?key={api_key}&email={email address}&limit=100 will return 100 rows of data (200 for monadic).

?key={api_key}&email={email address}&page=1 will return the first 500 rows of data (1000 for monadic)
?key={api_key}&email={email address}&page=2 will return the next 500 rows of data (1000 for monadic)

Limit Fields Returned

By default all fields will be returned for each line of data. You can use the fields query name to change the field items returned. Multiple fields can be requested by separating each one with a pipe (|).

Example:
?key={api_key}&email={email address}&fields=iso will return just the iso field.
?key={api_key}&email={email address}&fields=iso|fatalities will return the iso and fatalities data for each row.

Complex Queries

By default all fields must match for the data to be returned. In some instances more complex queries may be required to use the OR clause. This is possible by separating the fields to join, by OR, with :OR: text. The main query item will be the first item in the join, followed by the other items split with :OR:. These can be used with other queries too.

Example:
?key={api_key}&email={email address}&{fieldname}={text}:OR:{fieldname2}={text2}:OR:{fieldname3}={text3} will return data where field = text OR field2 = text2 OR field3 = text3.
?key={api_key}&email={email address}&{fieldname}={text}:OR:{fieldname2}={text2} &country={country} will return data where field = text OR field2 = text2 AND country = country.

All items wrapped in {} must be replaced with the actual fields or text required. The curly brackets must also be removed from the query.
This data call returns the deleted items from the dataset

Commands

Read
In order to retrieve the data you must make a GET or POST request to the following URL:

https://api.acleddata.com/deleted/read?key={api_key}&email={email}

Returned Data (json only)

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>status</td>
<td>int</td>
<td>A number representing the request status</td>
</tr>
<tr>
<td>success</td>
<td>boolean</td>
<td>A boolean representation on the success of the call</td>
</tr>
<tr>
<td>last_update</td>
<td>int</td>
<td>The number of hours since the last update to the data</td>
</tr>
<tr>
<td>count</td>
<td>int</td>
<td>The number of data rows returned</td>
</tr>
<tr>
<td>messages</td>
<td>array</td>
<td>An array of information messages that may require future action</td>
</tr>
<tr>
<td>data</td>
<td>array</td>
<td>The rows of data returned. For details of attributes returned in each row, see below.</td>
</tr>
<tr>
<td>filename</td>
<td>string</td>
<td>The filename that will be used for csv calls</td>
</tr>
<tr>
<td>error</td>
<td>array</td>
<td>The details of the error with a status as an integer and message as a string</td>
</tr>
</tbody>
</table>

Returned Data (json, xml, txt, csv)

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>event_id_cnty</td>
<td>string</td>
<td>An individual identifier by number and country acronym</td>
</tr>
<tr>
<td>deleted_timestamp</td>
<td>int / date</td>
<td>The unix timestamp this data entry was deleted</td>
</tr>
</tbody>
</table>
Query Filters

Each field can be searched to filter the returned data. By default each field will search by = or LIKE based on the table below. This can be changed by sending a new query string name value pair, where the name has ‘_where’ appended to it. The following table shows the default query type that will be used by each field. **The key and email address query must be included in all requests to indicate that you have registered your use of our API.**

<table>
<thead>
<tr>
<th>Query Name</th>
<th>Type</th>
<th>Query String</th>
</tr>
</thead>
<tbody>
<tr>
<td>key</td>
<td>=</td>
<td>{api_key}</td>
</tr>
<tr>
<td>email</td>
<td>=</td>
<td>{email address associated with key}</td>
</tr>
<tr>
<td>event_id_cnty</td>
<td>LIKE</td>
<td>?event_id_cnty={text}</td>
</tr>
<tr>
<td>deleted_timestamp</td>
<td>&gt;=</td>
<td>?deleted_timestamp={number/yyyy-mm-dd}</td>
</tr>
</tbody>
</table>

- All LIKE queries will include a wildcard before and after the entered text.
- Multiple queries can be searched with name/value pairs separated by &. Each field will be searched using AND so all arguments must match for data to be returned.
- More complex queries can be searched to include the OR clause. See Complex Queries below.

To change the default query type you can add an additional name/value pair using the query name and appending ‘_where’ to the query name. The query value could be LIKE or %3D for ‘=’. Additional types of ‘<', ‘>’ and ‘BETWEEN may also be used, representing less than, greater than and between. The between requires the query name value to separate the 2 values with a |.

**Example:**
?key={api_key}&email={email address}&event_id_cnty={text}&event_id_cnty_where=%3D
?key={api_key}&email={email address}&deleted_timestamp={yyyy-mm-dd|yyyy-mm-dd}
&deleted_timestamp_where=BETWEEN

**Limit Query & Pagination**

By default there is a limit of 500 rows of data returned. You can use the limit query name to change the default number. Setting limit as 0 will return all relevant data. It is likely
returning all data will cause a timeout error and we therefore recommend using the page query instead. Each page will return 500 (1000 for monadic) rows of data.

**Example:**
?key={api_key}&email={email address}&limit=100 will return 100 rows of data.
?key={api_key}&email={email address}&page=1 will return the first 500 rows of data
?key={api_key}&email={email address}&page=2 will return the next 500 rows of data

**Limit Fields Returned**
By default all fields will be returned for each line of data. You can use the fields query name to change the field items returned. Multiple fields can be requested by separating each one with a pipe (|).

**Example:**
?key={api_key}&email={email address}&fields=event_id_cnty will return just the event_id_cnty field.

**Complex Queries**
By default all fields must match for the data to be returned. In some instances more complex queries may be required to use the OR clause. This is possible by separating the fields to join, by OR, with :OR: text. The main query item will be the first item in the join, followed by the other items split with :OR:. These can be used with other queries too.

**Example:**
?key={api_key}&email={email address}&{fieldname}={text}:OR:{fieldname2}={text2}:OR:{fieldname3}={text3} will return data where field = text OR field2 = text2 OR field3 = text3.

All items wrapped in {} must be replaced with the actual fields or text required. The curly brackets must also be removed from the query.
Actor

This data call returns the actors

Commands

Read

In order to retrieve the data you must make a GET or POST request to the following URL:

https://api.acleddata.com/actor/read?key={api_key}&email={email address}

Returned Data (json only)

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>status</td>
<td>int</td>
<td>A number representing the request status</td>
</tr>
<tr>
<td>success</td>
<td>boolean</td>
<td>A boolean representation on the success of the call</td>
</tr>
<tr>
<td>last_update</td>
<td>int</td>
<td>The number of hours since the last update to the data</td>
</tr>
<tr>
<td>count</td>
<td>int</td>
<td>The number of data rows returned</td>
</tr>
<tr>
<td>messages</td>
<td>array</td>
<td>An array of information messages that may require future action</td>
</tr>
<tr>
<td>data</td>
<td>array</td>
<td>The rows of data returned. For details of attributes returned in each row, see below.</td>
</tr>
<tr>
<td>filename</td>
<td>string</td>
<td>The filename that will be used for csv calls</td>
</tr>
<tr>
<td>error</td>
<td>array</td>
<td>The details of the error with a status as an integer and message as a string</td>
</tr>
</tbody>
</table>

Returned Data (json, xml, txt, csv)

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>actor_name</td>
<td>string</td>
<td>The name of the actor</td>
</tr>
<tr>
<td>first_event_date</td>
<td>date</td>
<td>The date the first event for this actor occurred in the format: yyyy-mm-dd</td>
</tr>
<tr>
<td>last_event_date</td>
<td>date</td>
<td>The date the last event for this actor occurred in the format: yyyy-mm-dd</td>
</tr>
<tr>
<td>event_count</td>
<td>int</td>
<td>The number of events that have occurred with this actor</td>
</tr>
</tbody>
</table>
Query Filters

Each field can be searched to filter the returned data. By default each field will search by = or LIKE based on the table below. This can be changed by sending a new query string name value pair, where the name has '_where' appended to it. The following table shows the default query type that will be used by each field. The key and email address query must be included in all requests to indicate that you have registered your use of our API.

<table>
<thead>
<tr>
<th>Query Name</th>
<th>Type</th>
<th>Query String</th>
</tr>
</thead>
<tbody>
<tr>
<td>api</td>
<td>=</td>
<td>{api_key}</td>
</tr>
<tr>
<td>email</td>
<td>=</td>
<td>{email address associated with key}</td>
</tr>
<tr>
<td>actor_name</td>
<td>LIKE</td>
<td>?actor_name={text}</td>
</tr>
<tr>
<td>first_event_date</td>
<td>=</td>
<td>?first_event_date={yyyy-mm-dd}</td>
</tr>
<tr>
<td>last_event_date</td>
<td>=</td>
<td>?last_event_date={yyyy-mm-dd}</td>
</tr>
<tr>
<td>event_count</td>
<td>&gt;=</td>
<td>?event_count={number}</td>
</tr>
</tbody>
</table>

- All LIKE queries will include a wildcard before and after the entered text.
- Multiple queries can be searched with name/value pairs separated by &. Each field will be searched using AND so all arguments must match for data to be returned.
- More complex queries can be searched to include the OR clause. See Complex Queries below.

To change the default query type you can add an additional name/value pair using the query name and appending '_where' to the query name. The query value could be LIKE or %3D for '='. Additional types of '<', '>' and 'BETWEEN may also be used, representing less than, greater than and between. The between requires the query name value to separate the 2 values with a |.

**Example:**
?key={api_key}&email={email address}&actor_name={text}&actor_name_where=%3D
?key={api_key}&email={email address}&last_event_date={yyyy-mm-dd|yyyy-mm-dd}
&last_event_date_where=BETWEEN

Limit Query & Pagination
By default there is a limit of 500 rows of data returned. You can use the limit query name to change the default number. Setting limit as 0 will return all relevant data. It is likely returning all data will cause a timeout error and we therefore recommend using the page query instead. Each page will return 500 rows of data.

**Example:**

?key={api_key}&email={email address}&limit=100 will return 100 rows of data.

?key={api_key}&email={email address}&page=1 will return the first 500 rows of data

?key={api_key}&email={email address}&page=2 will return the next 500 rows of data

**Complex Queries**

By default all fields must match for the data to be returned. In some instances more complex queries may be required to use the OR clause. This is possible by separating the fields to join, by OR, with :OR: text. The main query item will be the first item in the join, followed by the other items split with :OR:. These can be used with other queries too.

**Example:**

?key={api_key}&email={email address}&field={text}:OR:field2={text2}:OR:field3={text3} will return data where field = text OR field2 = text2 OR field3 = text3.

?key={api_key}&email={email address}&field={text}:OR:field2={text2} &event_count={number} will return data where field = text OR field2 = text2 AND event_count = number.
Actor Type

This data call returns the actor types

Commands

Read

In order to retrieve the data you must make a GET or POST request to the following URL:

https://api.acleddata.com/actortype/read?key={api_key}&email={email address}

Returned Data (json only)

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>status</td>
<td>int</td>
<td>A number representing the request status</td>
</tr>
<tr>
<td>success</td>
<td>boolean</td>
<td>A boolean representation on the success of the call</td>
</tr>
<tr>
<td>last_update</td>
<td>int</td>
<td>The number of hours since the last update to the data</td>
</tr>
<tr>
<td>count</td>
<td>int</td>
<td>The number of data rows returned</td>
</tr>
<tr>
<td>messages</td>
<td>array</td>
<td>An array of information messages that may require future action</td>
</tr>
<tr>
<td>data</td>
<td>array</td>
<td>The rows of data returned. For details of attributes returned in each row, see below.</td>
</tr>
<tr>
<td>filename</td>
<td>string</td>
<td>The filename that will be used for csv calls</td>
</tr>
<tr>
<td>error</td>
<td>array</td>
<td>The details of the error with a status as an integer and message as a string</td>
</tr>
</tbody>
</table>

Returned Data (json, xml, txt, csv)

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>actor_type_id</td>
<td>int</td>
<td>The id of the actor type</td>
</tr>
<tr>
<td>actor_type_name</td>
<td>string</td>
<td>The name of the actor type</td>
</tr>
<tr>
<td>first_event_date</td>
<td>date</td>
<td>The date the first event for this actor type occurred in the format: yyyy-mm-dd</td>
</tr>
<tr>
<td>last_event_date</td>
<td>date</td>
<td>The date the last event for this actor type occurred in the format: yyyy-mm-dd</td>
</tr>
</tbody>
</table>
Query Filters

Each field can be searched to filter the returned data. By default each field will search by = or LIKE based on the table below. This can be changed by sending a new query string name value pair, where the name has '_where' appended to it. The following table shows the default query type that will be used by each field. **The key and email address query must be included in all requests to indicate that you have registered your use of our API.**

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>event_count</td>
<td>int</td>
<td>The number of events that have occurred with this actor type</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Query Name</th>
<th>Type</th>
<th>Query String</th>
</tr>
</thead>
<tbody>
<tr>
<td>api</td>
<td>=</td>
<td>{api_key}</td>
</tr>
<tr>
<td>email</td>
<td>=</td>
<td>{email address associated with key}</td>
</tr>
<tr>
<td>actor_type_id</td>
<td>=</td>
<td>?actor_type_id={number}</td>
</tr>
<tr>
<td>actor_type_name</td>
<td>LIKE</td>
<td>?actor_name={text}</td>
</tr>
<tr>
<td>first_event_date</td>
<td>=</td>
<td>?first_event_date={yyyy-mm-dd}</td>
</tr>
<tr>
<td>last_event_date</td>
<td>=</td>
<td>?last_event_date={yyyy-mm-dd}</td>
</tr>
<tr>
<td>event_count</td>
<td>&gt;=</td>
<td>?event_count={number}</td>
</tr>
</tbody>
</table>

- All LIKE queries will include a wildcard before and after the entered text.
- Multiple queries can be searched with name/value pairs separated by &. Each field will be searched using AND so all arguments must match for data to be returned.
- More complex queries can be searched to include the OR clause. See Complex Queries below.

To change the default query type you can add an additional name/value pair using the query name and appending '_where' to the query name. The query value could be LIKE or %3D for '='. Additional types of '<', '>' and 'BETWEEN may also be used, representing less than, greater than and between. The between requires the query name value to separate the 2 values with a |.

**Example:**
**Complex Queries**

By default all fields must match for the data to be returned. In some instances more complex queries may be required to use the OR clause. This is possible by separating the fields to join, by OR, with :OR: text. The main query item will be the first item in the join, followed by the other items split with :OR:. These can be used with other queries too.

**Example:**

?key={api_key}&email={email address}&field={text}:OR:field2={text2}:OR:field3={text3} will return data where field = text OR field2 = text2 OR field3 = text3.

?key={api_key}&email={email address}&field={text}:OR:field2={text2} &event_count={number} will return data where field = text OR field2 = text2 AND event_count = number.
Country

This data call returns the countries

Commands

Read

In order to retrieve the data you must make a GET or POST request to the following URL:

https://api.acleddata.com/country/read?key={api_key}&email={email address}

Returned Data (json only)

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>status</td>
<td>int</td>
<td>A number representing the request status</td>
</tr>
<tr>
<td>success</td>
<td>boolean</td>
<td>A boolean representation on the success of the call</td>
</tr>
<tr>
<td>last_update</td>
<td>int</td>
<td>The number of hours since the last update to the data</td>
</tr>
<tr>
<td>count</td>
<td>int</td>
<td>The number of data rows returned</td>
</tr>
<tr>
<td>messages</td>
<td>array</td>
<td>An array of information messages that may require future action</td>
</tr>
<tr>
<td>data</td>
<td>array</td>
<td>The rows of data returned. For details of attributes returned in each row, see below.</td>
</tr>
<tr>
<td>filename</td>
<td>string</td>
<td>The filename that will be used for csv calls</td>
</tr>
<tr>
<td>error</td>
<td>array</td>
<td>The details of the error with a status as an integer and message as a string</td>
</tr>
</tbody>
</table>

Returned Data (json, xml, txt, csv)

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>country</td>
<td>string</td>
<td>The name of the country</td>
</tr>
<tr>
<td>iso</td>
<td>int</td>
<td>The iso number of the country</td>
</tr>
<tr>
<td>iso3</td>
<td>string</td>
<td>The iso3 representation of the country</td>
</tr>
<tr>
<td>first_event_date</td>
<td>date</td>
<td>The date the first event for this actor type occurred in the format: yyyy-mm-dd</td>
</tr>
<tr>
<td>last_event_date</td>
<td>date</td>
<td>The date the last event for this actor type occurred in the format: yyyy-mm-dd</td>
</tr>
</tbody>
</table>
Query Filters

Each field can be searched to filter the returned data. By default each field will search by = or LIKE based on the table below. This can be changed by sending a new query string name value pair, where the name has ‘_where’ appended to it. The following table shows the default query type that will be used by each field. **The key and email address query must be included in all requests to indicate that you have registered your use of our API.**

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>event_count</td>
<td>int</td>
<td>The number of events that have occurred with this actor type</td>
</tr>
</tbody>
</table>

- All LIKE queries will include a wildcard before and after the entered text.
- Multiple queries can be searched with name/value pairs separated by &. Each field will be searched using AND so all arguments must match for data to be returned.
- More complex queries can be searched to include the OR clause. See Complex Queries below.

To change the default query type you can add an additional name/value pair using the query name and appending ‘_where’ to the query name. The query value could be LIKE or %3D for ‘='. Additional types of ‘<’, ‘>’ and ‘BETWEEN may also be used, representing less
than, greater than and between. The between requires the query name value to separate
the 2 values with a |.

Example:
?key={api_key}&email={email address}&country={text}&country_where=%3D
?key={api_key}&email={email address}&last_event_date={yyyy-mm-dd|yyyy-mm-dd}
&last_event_date_where=BETWEEN

Complex Queries
By default all fields must match for the data to be returned. In some instances more
complex queries may be required to use the OR clause. This is possible by separating the
fields to join, by OR, with :OR: text. The main query item will be the first item in the join,
followed by the other items split with :OR: . These can be used with other queries too.

Example:
?key={api_key}&email={email address}&field={text}:OR:field2={text2}:OR:field3={text3} will
return data where field = text OR field2 = text2 OR field3 = text3.

?key={api_key}&email={email address}&field={text}:OR:field2={text2}
&event_count={number} will return data where field = text OR field2 = text2 AND
event_count = number.
Region

This data call returns the regions

Commands

Read
In order to retrieve the data you must make a GET or POST request to the following URL:

https://api.acleddata.com/region/read?key={api_key}&email={email address}

Returned Data (json only)

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>status</td>
<td>int</td>
<td>A number representing the request status</td>
</tr>
<tr>
<td>success</td>
<td>boolean</td>
<td>A boolean representation on the success of the call</td>
</tr>
<tr>
<td>last_update</td>
<td>int</td>
<td>The number of hours since the last update to the data</td>
</tr>
<tr>
<td>count</td>
<td>int</td>
<td>The number of data rows returned</td>
</tr>
<tr>
<td>messages</td>
<td>array</td>
<td>An array of information messages that may require future action</td>
</tr>
<tr>
<td>data</td>
<td>array</td>
<td>The rows of data returned. For details of attributes returned in each row, see below.</td>
</tr>
<tr>
<td>filename</td>
<td>string</td>
<td>The filename that will be used for csv calls</td>
</tr>
<tr>
<td>error</td>
<td>array</td>
<td>The details of the error with a status as an integer and message as a string</td>
</tr>
</tbody>
</table>

Returned Data (json, xml, txt, csv)

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>region</td>
<td>int</td>
<td>The id of the region</td>
</tr>
<tr>
<td>region_name</td>
<td>string</td>
<td>The name of the region</td>
</tr>
<tr>
<td>first_event_date</td>
<td>date</td>
<td>The date the first event for this actor type occurred in the format: yyyy-mm-dd</td>
</tr>
<tr>
<td>last_event_date</td>
<td>date</td>
<td>The date the last event for this actor type occurred in the format: yyyy-mm-dd</td>
</tr>
</tbody>
</table>
Query Filters

Each field can be searched to filter the returned data. By default each field will search by = or LIKE based on the table below. This can be changed by sending a new query string name value pair, where the name has ‘_where’ appended to it. The following table shows the default query type that will be used by each field. **The key and email address query must be included in all requests to indicate that you have registered your use of our API.**

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>event_count</td>
<td>int</td>
<td>The number of events that have occurred with this actor type</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Query Name</th>
<th>Type</th>
<th>Query String</th>
</tr>
</thead>
<tbody>
<tr>
<td>api</td>
<td>=</td>
<td>{api_key}</td>
</tr>
<tr>
<td>email</td>
<td>=</td>
<td>{email address associated with key}</td>
</tr>
<tr>
<td>region</td>
<td>=</td>
<td>?region={number}</td>
</tr>
<tr>
<td>region_name</td>
<td>LIKE</td>
<td>?region_name={text}</td>
</tr>
<tr>
<td>first_event_date</td>
<td>=</td>
<td>?first_event_date={yyyy-mm-dd}</td>
</tr>
<tr>
<td>last_event_date</td>
<td>=</td>
<td>?last_event_date={yyyy-mm-dd}</td>
</tr>
<tr>
<td>event_count</td>
<td>&gt;=</td>
<td>?event_count={number}</td>
</tr>
</tbody>
</table>

⚠️ All LIKE queries will include a wildcard before and after the entered text.
⚠️ Multiple queries can be searched with name/value pairs separated by &. Each field will be searched using AND so all arguments must match for data to be returned.
⚠️ More complex queries can be searched to include the OR clause. See Complex Queries below.

To change the default query type you can add an additional name/value pair using the query name and appending ‘_where’ to the query name. The query value could be LIKE or %3D for ‘=’. Additional types of ‘<’, ‘>’ and ‘BETWEEN’ may also be used, representing less than, greater than and between. The between requires the query name value to separate the 2 values with a |.
Example:
?key={api_key}&email={email address}&region_name={text}&region_name_where=%3D
?key={api_key}&email={email address}&last_event_date={yyyy-mm-dd|yyyy-mm-dd}
&last_event_date_where=BETWEEN

Complex Queries
By default all fields must match for the data to be returned. In some instances more complex queries may be required to use the OR clause. This is possible by separating the fields to join, by OR, with :OR: text. The main query item will be the first item in the join, followed by the other items split with :OR:. These can be used with other queries too.

Example:
?key={api_key}&email={email address}&field={text}:OR:field2={text2}:OR:field3={text3} will return data where field = text OR field2 = text2 OR field3 = text3.

?key={api_key}&email={email address}&field={text}:OR:field2={text2}
&event_count={number} will return data where field = text OR field2 = text2 AND event_count = number.