Functional Document for

Keen API

**API Keys**

Each of your projects will have its own set of API keys, which you can retrieve from the overview page of your project.

1. Goto <https://keen.io/project/PROJECT_ID>
2. Click the “Show API Keys” button
3. Select and copy the required API key

**Events**

Events are the individual data points collected by the Keen API. These events are stored in collections, which you can access through the Events resource. This flexible resource allows you to operate on multiple event collections for a given project with a single request. The specific actions for GET and POST requests are described in detail below.

**Resource**

<https://api.keen.io/3.0/projects/PROJECT_ID/events>

**Inspect all collections**

Return schema information for all the event collections in a given project, along with properties (and their types), and links to sub-resources

**SUPPORTED HTTP METHODS**

|  |  |  |
| --- | --- | --- |
| METHOD | AUTHENTICATION | DESCRIPTION |
| GET | [Read key](https://keen.io/docs/api/#read-key) | Return schema information for all the event collections in a given project, along with properties (and their types), and links to sub-resources. |

**OPTIONAL REQUEST PARAMETERS**

|  |  |
| --- | --- |
| PARAMETER | DESCRIPTION |
| api\_key | Optional alternative to an [Authorization header](https://keen.io/docs/api/#http-header) |

**REQUEST**

**curl** [**https://api.keen.io/3.0/projects/PROJECT\_ID/events?api\_key=READ\_KEY**](https://api.keen.io/3.0/projects/PROJECT_ID/events?api_key=READ_KEY)

**Event Collections:-**

The Event Collection resource provides a flexible interface for operating on a single event collection. This resource has different actions for GET, POST and DELETE requests, all of which are described in detail below.

**CREATING NEW EVENT COLLECTIONS**

An event collection is created the first time an event is recorded.

**RESOURCE**

<https://api.keen.io/3.0/projects/PROJECT_ID/events/COLLECTION_NAME>

**Inspect a single collection**

A GET request authenticated with a [Read key](https://keen.io/docs/api/?shell#read-key) returns schema information for a single event collection, along with properties (and their types), and links to sub-resources

**SUPPORTED HTTP METHODS**

|  |  |  |
| --- | --- | --- |
| METHOD | [AUTHENTICATION](https://keen.io/docs/api/?shell#authentication) | DESCRIPTION |
| GET | [Read key](https://keen.io/docs/api/?shell#read-key) | Return schema information for a single event collection, along with properties (and their types), and links to sub-resources |

**OPTIONAL REQUEST PARAMETERS**

|  |  |
| --- | --- |
| PARAMETER | DESCRIPTION |
| api\_key | Optional alternative to an [Authorization header](https://keen.io/docs/api/?shell#http-header) |

curl <https://api.keen.io/3.0/projects/PROJECT_ID/events/COLLECTION_NAME?api_key=READ_KEY>

**Extractions**

Creates an extraction request for full-form event data with all property values.We strongly believe you should always have full access to all of your data, and we aim to make that as simple and painless as possible.

**HTTP METHODS**

|  |  |  |
| --- | --- | --- |
| METHOD | [AUTHENTICATION](https://keen.io/docs/api/?shell#authentication) | DESCRIPTION |
| GET | [Read key](https://keen.io/docs/api/?shell#read-key) | Creates an extraction request for full-form event data with all property values. JSON objects passed as query string parameters need to be URL encoded. |
| HEAD | [Read key](https://keen.io/docs/api/?shell#read-key) | Returns the response header |
| POST | [Read key](https://keen.io/docs/api/?shell#read-key) | Creates an extraction request for full-form event data with all property values. Each parameter and value should be placed in a JSON object within the POST body |

**REQUIRED PARAMETERS**

|  |  |
| --- | --- |
| PARAMETER | DESCRIPTION |
| event\_collection | Specifies the name of the event collection to analyze. |
| timeframe | Refines the scope of events to be included in the analysis based on when the event occurred. |

OPTIONAL PARAMETERS

|  |  |
| --- | --- |
| PARAMETER | DESCRIPTION |
| api\_key | Alternative authentication method to providing an[Authorization header](https://keen.io/docs/api/?shell#http-header). |
| filters | Refines the scope of events to be included in the analysis based on event property values |
| timezone | Assigns a timezone offset to relative timeframes. |
| email | If an email address is specified, an email will be sent to it when your extraction is ready for download. If email is not specified, your extraction will be processed synchronously and your data will be returned as JSON. |
| latest | An integer containing the number of most recent events to extract. |
| property\_names | A URL-encoded array of strings containing properties you wish to extract. If this parameter is omitted, all properties will be returned |

**REQUEST**:**GET**

**$curl** <https://api.keen.io/3.0/projects/PROJECT_ID/queries/extraction?api_key=READ_KEY&event_collection=COLLECTION_NAME&timeframe=this_7_days>

# POST

# POST

$ curl https://api.keen.io/3.0/projects/PROJECT\_ID/queries/extraction \

-H "Authorization: READ\_KEY" \

-H 'Content-Type: application/json' \

-d "{

\"event\_collection\": \"COLLECTION\_NAME\",

\"timeframe\": \"this\_7\_days\"

}"

**Timeframe**

The timeframe parameter specifies a period of time over which to run an analysis. This refines the scope of events that are included in the analysis, based on when each event occurred. When omitted, analyses will default to all-time

There are two types of timeframes:

Absolute timeframes: a fixed timeframe with an explicit start and end

Relative timeframes: a rolling timeframe that is relative to “now”

**ABSOLUTE TIMEFRAMES**

Absolute timeframes are passed in with a URL-encoded JSON object containing “start” and “end” properties with ISO-8601 formatted date strings.

A query will be inclusive of events starting at the exact same time as the start time and exclusive of events starting with the exact same time as the end time. In other words, to run a query on an exact 24 hour window, you can use a timeframe that starts at midnight one day and ends at midnight the next day.

REQUEST

$ curl https://api.keen.io/3.0/projects/PROJECT\_ID/queries/count \

-H "Authorization: READ\_KEY" \

-H 'Content-Type: application/json' \

-d "{

\"event\_collection\": \"COLLECTION\_NAME\",

\"timeframe\": {

\"start\": \"2012-08-13T19:00:00.000Z\",

\"end\": \"2013-09-20T19:00:00.000Z\"

}

}"

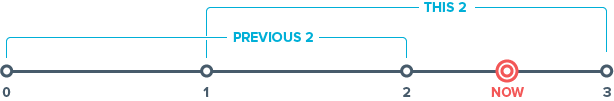
RELATIVE TIMEFRAMES

Relative timeframes are passed in with a patterned string sequence:

{rel}\_{n}\_{units}

|  |  |
| --- | --- |
| PATTERN | DESCRIPTION |
| {rel} | “this” or “previous” – Use “this” when you want to include events happening right up until now. Use “previous” when you only want to get results for complete chunks of time (e.g. the full hour, day, or week). |
| {n} | Any whole number greater than 0 (zero) |
| {units} | “minutes”, “hours”, “days”, “weeks”, “months”, or “years”. |

This chart illustrates the difference between “this” and “previous”:



Notice that this\_2\_days will include all of the current day and all of the previous day, whereas previous\_2\_days includes the previous two fully completed days and none of the current day.

Below are the supported relative timeframes for “this”:

|  |  |
| --- | --- |
| TIMEFRAME | DESCRIPTION |
| this\_minute | Creates a timeframe starting from the beginning of the current minute until now. |
| this\_hour | Creates a timeframe starting from the beginning of the current hour until now. |
| this\_day | Creates a timeframe starting from the beginning of the current day until now. |
| this\_week | Creates a timeframe starting from the beginning of the current week until now. |
| this\_month | Creates a timeframe starting from the beginning of the current month until now. |
| this\_year | Creates a timeframe starting from the beginning of the current year until now. |
| this\_n\_minutes | All of the current minute and the previous completed n-1 minutes. |
| this\_n\_hours | All of the current hour and the previous completed n-1 hours |
| this\_n\_days | All of the current day and the previous completed n-1 days. |
| this\_n\_weeks | All of the current week and the previous completed n-1 weeks. |
| this\_n\_months | All the current month and previous completed n-1 months. |
| this\_n\_years | All the current year and previous completed n-1 years. |

Below are the supported relative timeframes for “previous”:

|  |  |
| --- | --- |
| TIMEFRAME | DESCRIPTION |
| previous\_n\_minutes | Gives a start of n-minutes before the most recent complete minute and an end at the most recent complete minute. (For example: If right now it is 7:15:30pm and I specify “previous\_3\_minutes”, the timeframe would stretch from 7:12pm until 7:15pm.) |
| previous\_n\_hours | Gives a start of n-hours before the most recent complete hour and an end at the most recent complete hour. (For example: If right now it is 7:15pm and I specify “previous\_7\_hours”, the timeframe would stretch from noon until 7:00pm.) |
| previous\_n\_days | Gives a starting point of n-days before the most recent complete day and an end at the most recent complete day. (For example: If right now it is Friday at 9:00am and I specify a timeframe of “previous\_3\_days”, the timeframe would stretch from Tuesday morning at 12:00am until Thursday night at midnight.) |
| previous\_n\_weeks | Gives a start of n-weeks before the most recent complete week and an end at the most recent complete week. (For example: If right now it is Monday, and I specify a timeframe of “previous\_2\_weeks”, the timeframe would stretch from three Sunday mornings ago at 12:00am until the most recent Sunday at 12:00am (yesterday morning).) |
| previous\_n\_months | Gives a start of n-months before the most recent completed month and an end at the most recent completed month. (For example: If right now is the 5th of the month, and I specify a timeframe of “previous\_2\_months”, the timeframe would stretch from the start of two months ago until the end of last month.) |
| previous\_n\_years | Gives a start of n-years before the most recent completed year and an end at the most recent completed year. (For example: If right now is the June 5th, and I specify a timeframe of “previous\_2\_years”, the timeframe would stretch from the start of two years ago until the end of last year.) |

$ curl https://api.keen.io/3.0/projects/PROJECT\_ID/queries/count \

-H "Authorization: READ\_KEY" \

-H 'Content-Type: application/json' \

-d "{

\"event\_collection\": \"COLLECTION\_NAME\",

\"timeframe\": \"this\_7\_days\"

}"

**Interval**

The interval parameter groups results into sub-timeframes spanning a specified length of time.

This type of analysis can help answer questions such as:

How many signups have occurred daily, over the past 21 days?

How much has revenue grown per week since launching a new product?

SUPPORTED INTERVALS

* minutely
* hourly
* daily
* weekly
* monthly
* yearly

CUSTOM INTERVALS:

In addition to the above intervals, the following pattern can be used to create highly specific custom intervals: every\_{n}\_{units}, where {n} can be any whole integer greater than 0 (zero), and {units} can be minutes, hours, days, weeks, months, or years. Here are a few examples:

* every\_30\_minutes
* every\_8\_hours
* every\_3\_days
* every\_2\_weeks
* every\_6\_months
* every\_3\_years

REQUEST

$ curl https://api.keen.io/3.0/projects/PROJECT\_ID/queries/count \

-H "Authorization: READ\_KEY" \

-H 'Content-Type: application/json' \

-d '{

"event\_collection": "user logins",

"timeframe": "previous\_3\_days",

"interval": "daily"

}'