Graph API Developer Documentation

Alpha Access

Last updated 2017-04-10 test

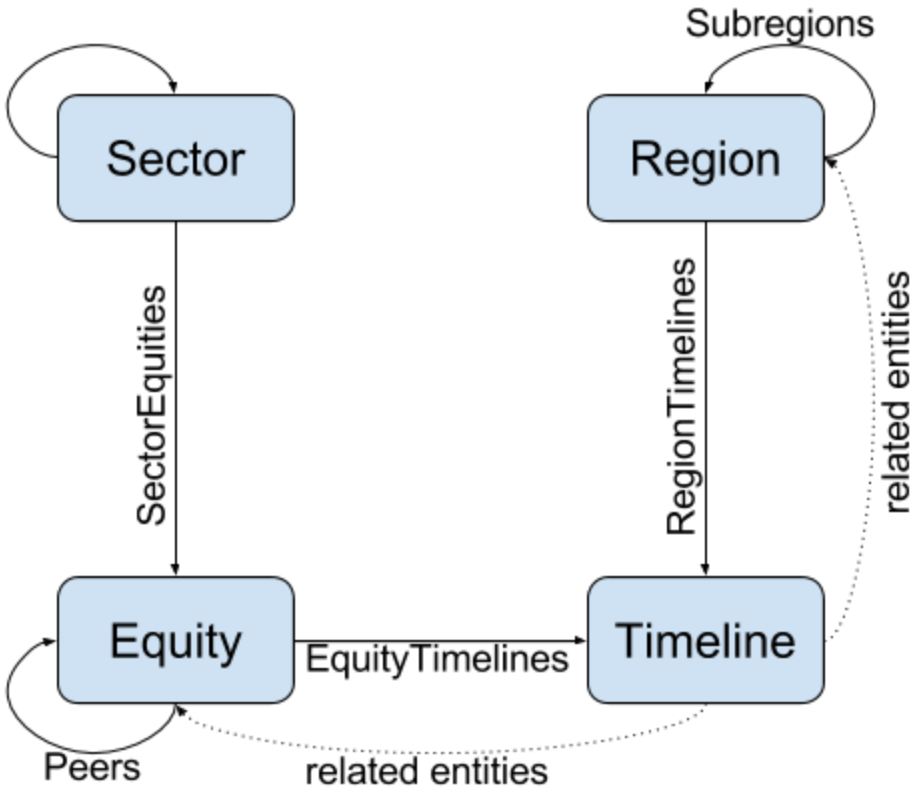
Preliminaries

The Kensho Knowledge graph contains a large number of entities and an even larger number of connections. This makes it very powerful but requires a different approach than traditional database structures. The Kensho Graph API allows users (both human and machine) to access the Kensho Knowledge Graph in a scalable way.

Overview

Central to the APIs is the concept of an entity “node” and relationship “edge”. The Kensho Graph contains large number of different types of entities (Assets, Regions, Timelines, etc). Every entity is associated with a “class”, and entities of the same class will have the same properties. Each entity is identified by a unique id and can be accessed directly. Entities of a particular class can (but may or may not) have relationships to entities of another class, as illustrated below. The graph schema is self-describing. We provide APIs for listing available classes and available relationships for each class.

Sample entity class and relationship diagram



Tables

The response from each API endpoint described below is a JSON dictionary describing a table. It contains two keys: **`data`** is a list of dictionaries describing rows of the table, whereas **`metadata`** is a list of dictionaries describing columns of the data (keys into **`data`** dictionaries), with human readable names of the columns and their data types (e.g. `Date`, `String`, etc.).

Events and Timelines

All events in the graph are organized into "timelines" such as "Alphabet Inc. Earnings Releases", "Bank of England Meeting Starts" or "Merchandise Trade: Level (UK)". Each timeline has a type that determines what additional information about events is available (e.g. period end date and actual and consensus numbers for the econometric timelines). The list of available timeline types can be retrieved from the API.

Special Cases

The graph offers special capabilities with some entity classes.

- Timeline is a regular entity (with name, ids, etc) that can be expanded into a table with multiple events, each of which has a date, a name and potentially other associated data. The events themselves cannot be accessed outside of the timeline.
- Assets can be resolved uniformly using a single API but would correspond to different classes (Equity, Currency, Index, etc) with different properties.

Accessing the graph

Service

The API endpoints can be accessed via the `GET` method on `https://www.kensho.com/external/v1/<endpoint>?<query>`, specifying parameters as name-value pairs in the query string.

Authentication

API endpoints use token authentication. Each HTTPS API request will need to include the authorization header. For example:

```
Authorization: Token APrettyLongHexValueSecret
```

Limits

API endpoints are configurable and scalable. During the initial release, each user is limited to

- 60 requests / minute

Downloading Results as CSV

A CSV of the query output can be downloaded by appending `output=csv` to the query

APIs

API Endpoint	Parameters	Output
list_entity_classes	none	A table with one row per supported entity class. Each row contains the name of the class and its capabilities (e.g. whether all entities in the class can be listed).
list_timeline_types	none	A table with one row per supported timeline type
list_entities_of_class	class_name (string)	A table with all entities of the requested class. Available only for classes that support listing, as specified in list_entity_classes output.
get_class_relationships	class_name (string)	A table with two columns: name of the relationship and the name of the related entity class.
search_entities	class_name (string) search_string (string)	A table with all entities of the requested class that match the search string. Columns depend on the entity class.
get_entity	entity_id (string)	A table with a single row describing the entity identified by *entity_id*. Columns depend on the entity class.
get_related_entities	entity_id (string) relationship (string)	A table with all entities related to the one identified by *entity_id* by *relationship.*
get_timeline	timeline_id (string)	A table containing all events in the timeline with that ID.
get_calendar	start_date (string, formatted as YYYY-MM-DD) end_date (string, formatted as YYYY-MM-DD)	A table containing all events that have happened (or are scheduled to happen) between *start_date* and

		end_date (not including *end_date*). Each event is annotated with the timeline it belongs to. *Dates are specified in UTC. Date ranges of at most 10 days long are supported.*
get_ongoing_episodes	start_date (string, formatted as YYYY-MM-DD) end_date (string, formatted as YYYY-MM-DD)	A table containing all episodes (happenings that may stretch into multiple days) that have been/will be ongoing some time between start_date and end_date (not including end_date). *Dates are specified in UTC. Date ranges of at most 10 days long are supported.*
translate_asset_id	id_type (string, one of TICKER, RIC, CUSIP, KENSHO) asset_id (string)	Given an identification string for an asset (e.g. ticker, CUSIP, etc) return all known identifiers for this particular asset, including the Kensho one that should be used when interacting with the rest of the graph API.

Data Types

As noted above, each API returns metadata about data elements it returns. In particular, each field has 'units' associated with it. This indicates the type of the data contained in that field. This section documents different types output by the API.

Unit	**Usage**	**Encoded as**
Date	date value	ISO-8601 encoded string, YYYY-MM-DD format
DateTime	datetime value	ISO-8601 encoded string, YYYY-MM-DDTHH:mm:ssZ format. Note: datetimes may come with a mask indicating to what precision the time is known to be accurate.
String	a single string (name, URL, etc)	N/A
StringSet	a list of strings (e.g. aliases)	N/A
EntitySet	list of other entities	a list of dictionaries, one for each entity, with keys `'entity_type'`, `'entity_id'`, `'entity_name'`
Boolean		
Number		

Examples

List all available entity classes

```
curl -H 'Authorization: Token <YOUR TOKEN HERE>'
https://www.kensho.com/external/v1/list_entity_classes
```

List all relationships the class “Equity” can have

```
curl -H 'Authorization: Token <YOUR TOKEN HERE>'
https://www.kensho.com/external/v1/get_class_relationships?class_name
=Equity
```

Get Events Calendar for April 1st, 2017 (in UTC)

```
curl -H 'Authorization: Token <YOUR TOKEN HERE>
'https://www.kensho.com/external/v1/get_calendar?start_date=2017-04-0
1&end_date=2017-04-02'
```

Find all timelines related to Apple

This is a two-step process. First get the graph identifier for Apple security:

```
curl -H 'Authorization: Token <YOUR TOKEN HERE>'
'https://www.kensho.com/external/v1/translate_asset_id?id_type=TICKER
&asset_id=AAPL'
```

Then use the identifier corresponding to the entry with type KENSHO from the result (in this case `f9d09c10-c6de-4e8f-9624-996774888bcc`) to retrieve related timelines.

```
curl -H 'Authorization: Token <YOUR TOKEN HERE>'
'https://www.kensho.com/external/v1/get_related_entities?entity_id=f9
d09c10-c6de-4e8f-9624-996774888bcc&relationship=EquityTimelines'
```

Get earnings timeline for Apple

Using data retrieved in the previous example (timeline with the type `Earnings releases for an asset`)

```
curl -H 'Authorization: Token <YOUR TOKEN HERE>'
'https://www.kensho.com/external/v1/get_timeline?timeline_id=ec5a179d
-ae08-4912-8bca-6a348baa2213'
```

Getting Support

Email your questions to api-support@kensho.com.

Topical/Conceptual Relationships (Experimental)

The common usecase for the APIs described above is retrieving direct relationships between companies, timelines and events. But often asset performance is affected by events in the same "space" not directly linked to it.

1. Events for an explicit entity, e.g., Microsoft
2. Events for peers at corporate identity level (e.g., based on GICS)
3. Events for peers defined by Smart Connections (functional, context related)

For example Microsoft and Alphabet compete in mobile and cloud, so any event in those spaces could potentially impact both of them. A significant Azure (MSFT) disruption might affect GOOGL's performance. On the other hand until Alphabet enters video gaming space, new XBox releases might not be of interest to Alphabet investors.

To provide more context around equities and events we introduce *Concepts.* These can be product areas (e.g. smartphones or video games), company classifications (e.g. "value" vs "growth"), or company relationships (e.g. supplier or customer) to give a few examples. These concepts mediate the fine-grained relationships between companies and events or pairs of companies.

For the first example below, Alphabet, Microsoft and Azure disruption are linked in the graph to the "Cloud computing" concept, allowing the user to discover the relationships between an event (Azure outage) to GOOGL via the concept.

A second example listed below is around a product release such as the impact that the Amazon Fire Phone has on either the Amazon parent company or, in the Concept of smartphones, its relationship to competing phone companies.