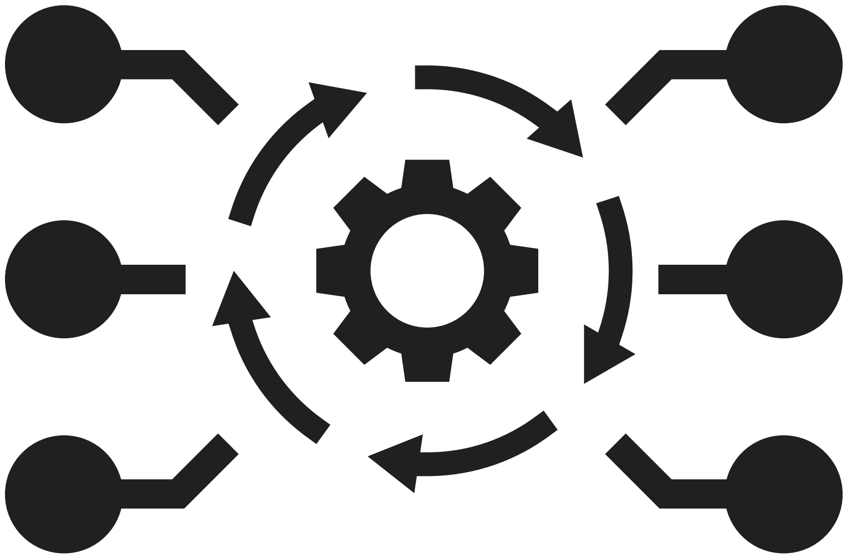
Writing REST API documentation

a manual for technical writers

Samuel Adelaar



© 2018 by Samuel Adelaar
No rights reserved.

Contents

REST APIs defined 1

Why read this manual 1
How to read this manual 1
About Google Knowledge Graph Search API 2

Get familiar with making a call 2

The endpoint 4
Authentication 4
Methods 4
Parameters 4

Making calls with curl 5

Set up curl on Windows 5
Get started with writing a request for curl 6

Understanding responses 6

Objects 7
Arrays 7

An introduction to writing documentation for a REST API 8

Overview 8
Reference 9

Going forward 15

Index 17

References 19

REST APIs defined

To get started with learning how to document REST APIs, you need to get to know them. Let's begin with the *API* of *REST API*. With an application programming interface (API), developers enable applications to interact with one another. The event that defines an API's use finds one application making a request, and another returning a response. This is referred to as making a call.

Note that Representational state transfer (REST) APIs are web-based and used for exchanging information. Like with other web-based APIs, a client uses a REST API to make a request, then a server returns a response. Further, with web-based APIs, HTTP protocol is the transport format for requests and responses. Among web-based APIS, REST is the most prevalent.

Clients use REST APIs to interact with a server's resources. Resources refer to information. With HTTP protocol as the transport format, a client uses a URL to interact with a particular resource. So you can compare the implementation of a REST API with entering a URL in a browser to go to a website.

Consider, for example, Instagram's REST API. Developers can program their applications so that they use this REST API to get photos posted to Instagram with certain tags.

Why read this manual

A developer needs documentation to enable their application to implement a REST API. This need derives from a key feature of REST APIs: that developers follow not a standard but instead guidelines for writing them. This entails that the code for REST APIs displays a wide variety. This flexibility with the development of REST APIs explains the need for documentation. As technical writers, your vocation is crafting effective, useable documentation, so developing REST API documentation is a promising line of work to pursue.

How to read this manual

For an optimal outcome, read this manual from beginning to end. After reading it in its entirety, you can also use the manual as a reference. You don't require prior knowledge of REST APIs to understand and follow it.

About Google Knowledge Graph Search API

This manual uses Google’s Knowledge Graph Search API (KG) as an example. The Knowledge Graph is a tool for improving Google searches. The carousel that may be displayed at the top of a list of Google search results represents an initial implementation of the Knowledge Graph. With the KG, developers can enable their applications to get information from the Knowledge Graph. For example, a user of an application with the KG can search for something and get a ranked list of notable instances of that thing.

Get familiar with making a call

Recall that a call refers to when a client uses a REST API to make a request, and then a server returns a response.

Build on your introduction to REST APIs in *REST APIs defined* by getting to know how to make a call with the KG.

Suppose you want to search for *Toronto* with the KG. Let’s look at the code for making this request with curl, a command line tool for transferring data with URLs. For now, take note of the code after -*X*.

```
curl -X GET \ ‘https://kgsearch.googleapis.com/v1/
entities:search?key={API key}&query=toronto&types=-
Place&limit=1’ \
```

Within this code are four fundamental parts of a request:

- An endpoint
- Authentication
- A method
- Parameters

Now let’s look at the response that KG returns after you make the call. This response is in JSON, the most prevalent format for REST API responses.

```
{
  “@context”: {
    “@vocab”: “http://schema.org/”,
    “goog”: “http://schema.googleapis.com/”,
    “EntitySearchResult”: “goog:EntitySearchResult”,
    “detailedDescription”: “goog:detailedDescription”,
    “resultScore”: “goog:resultScore”,
    “kg”: “http://g.co/kg”
  },  “@type”: “ItemList”,
  “itemListElement”: [
    {
      “@type”: “EntitySearchResult”,
      “result”: {
        “@id”: “kg:/m/0h7h6”,
        “name”: “Toronto”,
        “@type”: [
          “Place”,
          “Thing”,
          “City”,
          “AdministrativeArea”
        ],
        “description”: “City in Ontario, Canada”,
        «image»: {
          «contentUrl»: «http://t3.gstatic.com/imag-
es?q=tbn:ANd9GcRKjp8pA6tePrZ9yTdmcL3Tuj2W2evBaOVFUbk_
MIx3w4nI5OQs»,
          “url”: “https://commons.wikimedia.org/wiki/
File:Toronto_at_Dusk_-a.jpg”
        },
        “detailedDescription”: {
          “articleBody”: “Toronto is the capital city
of the province of Ontario and the largest city in
Canada by population, with 2,731,571 residents in
2016. “,
          “url”: “https://en.wikipedia.org/wiki/Toron-
to”,
          “license”: “https://en.wikipedia.org/wiki/
Wikipedia:Text_of_Creative_Commons_Attribution-ShareA-
like_3.0_Unported_License”
        },
        “url”: “http://www.toronto.ca/”
      },
      “resultScore”: 276.141785
    }
  ]
}
```

With this code for making a call with the KG, get to know the four fundamental parts of a request:

The endpoint

Retaining only the endpoint from the call to KG, you get this:

```
https://kgsearch.googleapis.com/v1/entities:search
```

An endpoint is a unique location for the information an application can interact with using a REST API. As you can see, it is a URL. One REST API can have multiple endpoints, enabling a variety of ways for interacting with its information.

KG has only one endpoint, the one that is featured in this manual. It is called *entities.search*, and it “Searches Knowledge Graph for entities that match the constraints,” according to KG’s documentation.

Authentication

REST APIs use authentication to manage who is permitted to call them and how they are allowed to interact with them. Retaining only the authentication method for KG leaves you with this:

```
key={API key}
```

KG’s authentication method is an API key. Were you making the call yourself, you would replace *API Key* in the curly brackets with the key you solicited from Google APIs.

Methods

Retaining only the method from the call to KG, you get this:

```
GET
```

A method refers to how a request acts upon an endpoint. There is a standard set of methods, which means that developers create REST APIs so that these methods can apply to them.

KG’s endpoint permits only the GET method. With GET, a request can retrieve information from an endpoint.

The standard methods

- GET
- POST
- PUT
- PATCH
- DELETE

Parameters

Keeping only the parameters from the call to KG leaves you with this:

```
?key={API key}&query=toronto&types=Place&limit=1
```

You include parameters in a request to define how the request interacts with a resource. Developers establish a fixed set of parameters for an endpoint. The call to KG that this manual uses as an example features four parameters, but the KG documentation lists four others.

Have a look at each parameter of the call to KG:

- **Key.** You authenticate the request with this parameter.
- **Query.** This is the text that the call is requesting the Knowledge Graph to search for.
- **Types.** This limits the response to search results that belong to the value for this parameter. Types make up schemas. DITA, for instance, is a schema for XML.
- **Limit.** This limits the number of search results to the value for this parameter.

Making calls with curl

Why learn how to make a call with curl? REST API documentation routinely features requests written for curl as its request examples. Take note that examples are integral to effective documentation for REST APIs. Developers are familiar with curl, and they can adapt requests written for the tool to the programming language they use to make calls.

Set up curl on Windows

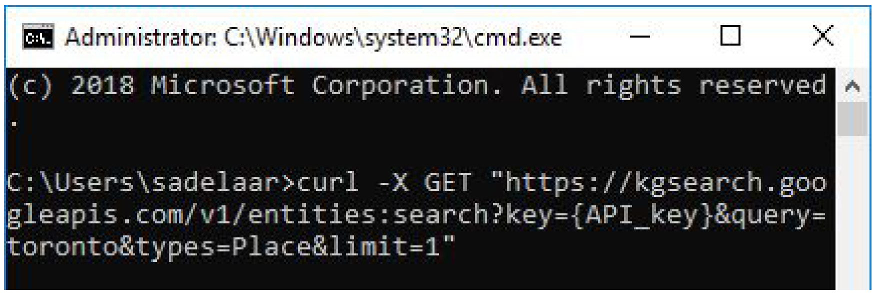
Follow these instructions to set up curl on your computer:

1. On your computer's control panel, go to System and Security, then System to find out whether Windows is 32-bit or 64-bit.
2. Search for the site Confused by Code. On the site, go to the cURL page.
3. Follow the instructions for downloading and installing curl that the page provides. Download the version appropriate for your Windows.

After you've set up curl, open it with Run: press and hold the Windows key, then click R. The run dialog opens. Enter *cmd* in Open, then click OK. Windows' command line opens.

Get started with writing a request for curl

Let's look again at the code for making the request with the KG in curl.



```
(c) 2018 Microsoft Corporation. All rights reserved
.

C:\Users\sadelaar>curl -X GET "https://kgsearch.goo
gleapis.com/v1/entities:search?key={API_key}&query=
toronto&types=Place&limit=1"
```

From the command line, you execute commands. The syntax for the command to make a call begins with the request's method. You enter *-X* and then the method in all caps to specify the method.

After the method, you enter the request's endpoint, authentication, and parameters, which you encompass in double quotation marks.

Within this code, take note of the use of symbols. A question mark follows the endpoint's URL. This symbol indicates that the subsequent code consists of the request's parameters. The ampersand distinguishes the parameters. The parameters' order is irrelevant to making the call.

Curl has commands for the variety of operations included in making calls with REST APIs.

Understanding responses

JavaScript Object Notation (JSON) is the most prevalent format for responses. You need to have a basic understanding of JSON to document REST APIs. JSON is a syntax for storing and exchanging information. Two benefits of JSON stem from the fact that it is text: first, it is effective for exchanging over the web, and second, any programming language can use it.

You use two syntactical units to write in JSON, objects and arrays. Remember that the call with KG returns a response in JSON. Let's examine parts of this response to understand these two units.

Objects

An object is a set of key-value pairs, delimited by curly brackets. This is an example of a key-value pair:

```
“url”: “https://en.wikipedia.org/wiki/Toronto”
```

The key is to the left of the colon and the value is to its right. Within an object’s curly brackets, a comma separates each key-value pair. With JSON you can write an object as a value. In the response look at the value for the key *detailedDescription*:

```
{
          “articleBody”: “Toronto is the capital city
of the province of Ontario and the largest city in
Canada by population, with 2,731,571 residents in
2016. “,
          “url”: “https://en.wikipedia.org/wiki/Toron-
to”,
          “license”: “https://en.wikipedia.org/wiki/
Wikipedia:Text_of_Creative_Commons_Attribution-ShareA-
like_3.0_Unported_License”
        }
```

Arrays

An array is a list of information, delimited by brackets. This information can take the form of standard text and integers as well as other arrays and objects. This is an example of an array from the response:

```
[
          “Place”,
          “Thing”,
          “City”,
          “AdministrativeArea”
        ]
```

Notice that it is the value for @*type*.

An introduction to writing documentation for a REST API

Now that you've secured a basic understanding of REST APIs, you're set for learning how to document one. REST API documentation typically has two main parts: a user guide and a reference. In a user guide, you may find

- an overview
- authentication and authorization requirements
- status and error code explanations
- rate limiting and threshold information.

From these conventional sections of a user guide, this manual only covers how to develop an overview.

Overview

The overview introduces the REST API. To develop a complete overview, aim to

- explain the REST API's purpose
- highlight its business benefits
- identify its audience
- list its use cases.

Consider the overview for KG. It begins by explaining the REST API's purpose: "The Knowledge Graph Search API lets you find entities in the Google Knowledge Graph. The API uses standard schema.org types and is compliant with the JSON-LD specification."

The overview then covers a few of the REST API's use cases. One of these use cases is "Getting a ranked list of the most notable entities that match certain criteria."

Reference

The reference is the core of REST API documentation. It presents in-depth information about a REST API's resources. A reference typically includes these parts:

- Resources
- Endpoints and methods
- Parameters
- Request example
- Response example

Resources

Recall that clients interact with a server's resources through endpoints. Typically, a reference for a REST API opens with a description of its resources. You can break up this description into two sequential parts:

Resource description. Briefly define the information that is returned when a server calls the resource.

Endpoint description. A set of endpoints comprise a resource. After describing the resource, for each endpoint define the information that is returned when a server makes a request to the endpoint. Take note that the endpoints that make up a resource are thematically connected.

If the resource you're describing consists of a number of endpoints, you can organize their descriptions with this category: the methods that developers can use to interact with the endpoints.

With the KG, you can only make requests to one endpoint, so its reference actually dispenses with a resource description.

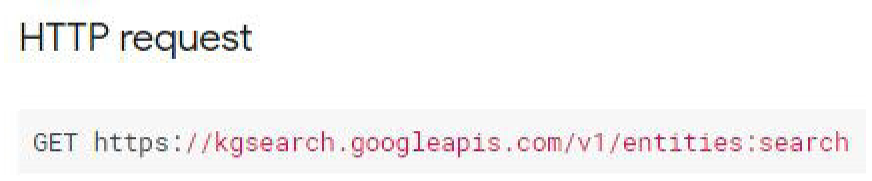
Endpoints and methods

Recall that a method refers to how a request acts upon an endpoint. So developers expect you to document these two properties of a call together. Plus methods don't require a description because they are standardized.

To develop a complete reference for methods and endpoints, aim to

- briefly describe the endpoint's information
- apply formatting to a method and endpoint that highlights the pair because for developers, the endpoint is the most significant part of a REST API
- when you refer to a particular endpoint, include the end path only. For KG's endpoint, this end path is */v1/entities:search*. Record the full URL in the overview.

Check out how the endpoint and method is documented in KG's reference:



HTTP request

```
GET https://kgsearch.googleapis.com/v1/entities:search
```

Notice that this documentation does include the full URL.

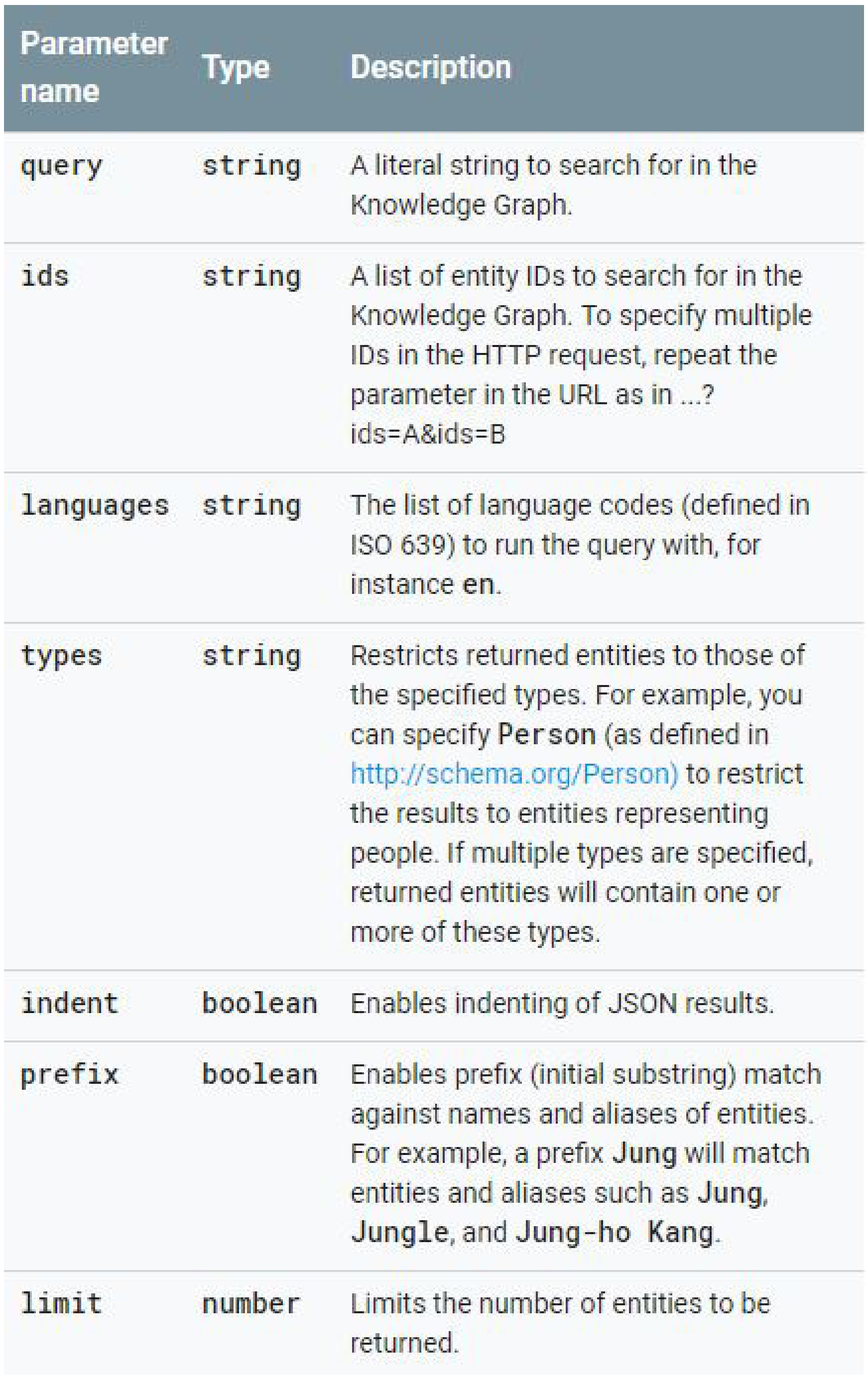
Parameters

For REST APIs, you may need to document four types of parameter: header, path, query string, and request body. Before getting in to how to document each parameter type, let's look at what to document for all parameters: data types and max and min values.

Data types. Each parameter typically takes only one type of data.

Max and min values. Record the maximum, minimum, and generally permitted values for a parameter to prevent developers from making calls that return errors.

Look at how parameters are documented in KG's reference:



Parameter name	Type	Description
`query`	`string`	A literal string to search for in the Knowledge Graph.
`ids`	`string`	A list of entity IDs to search for in the Knowledge Graph. To specify multiple IDs in the HTTP request, repeat the parameter in the URL as in ...?ids=A&ids=B
`languages`	`string`	The list of language codes (defined in ISO 639) to run the query with, for instance `en`.
`types`	`string`	Restricts returned entities to those of the specified types. For example, you can specify `Person` (as defined in http://schema.org/Person) to restrict the results to entities representing people. If multiple types are specified, returned entities will contain one or more of these types.
`indent`	`boolean`	Enables indenting of JSON results.
`prefix`	`boolean`	Enables prefix (initial substring) match against names and aliases of entities. For example, a prefix `Jung` will match entities and aliases such as `Jung`, `Jungle`, and `Jung-ho Kang`.
`limit`	`number`	Limits the number of entities to be returned.

Take note that using a table is a standard way of documenting parameters. Notice also that KG's reference does not cover max and min values.

Header parameters. You enter header parameters in a request's header. Think of the header as the header in HTML. Header parameters typically consist of authentication and authorization for requests to all of a REST API's endpoints. So you can document them in the requirements for authentication and authorization. But particular endpoints may permit including header parameters.

Path parameters. Path parameters comprise part of an endpoint's end path. REST API documentation typically delimits these parts of end paths with curly brackets to indicate that they are parameters.

Query string parameters. These are the parameters that you append to the end of an endpoint. KG's reference represents a model way to document query string parameters.

Request body parameters. With the PUT method, you can add information to an endpoint. The request body parameter refers to this information because you enter it in the request body. This information typically takes the form of a JSON object.

Request body parameters may present a challenge for documentation. The JSON object's size may be large, and its structure may be complex. Consider this method for documenting request body parameters: Present an example of a complete request body with generic keys and values. Then present a table with descriptions of these keys and values and the data types they take.

Request example

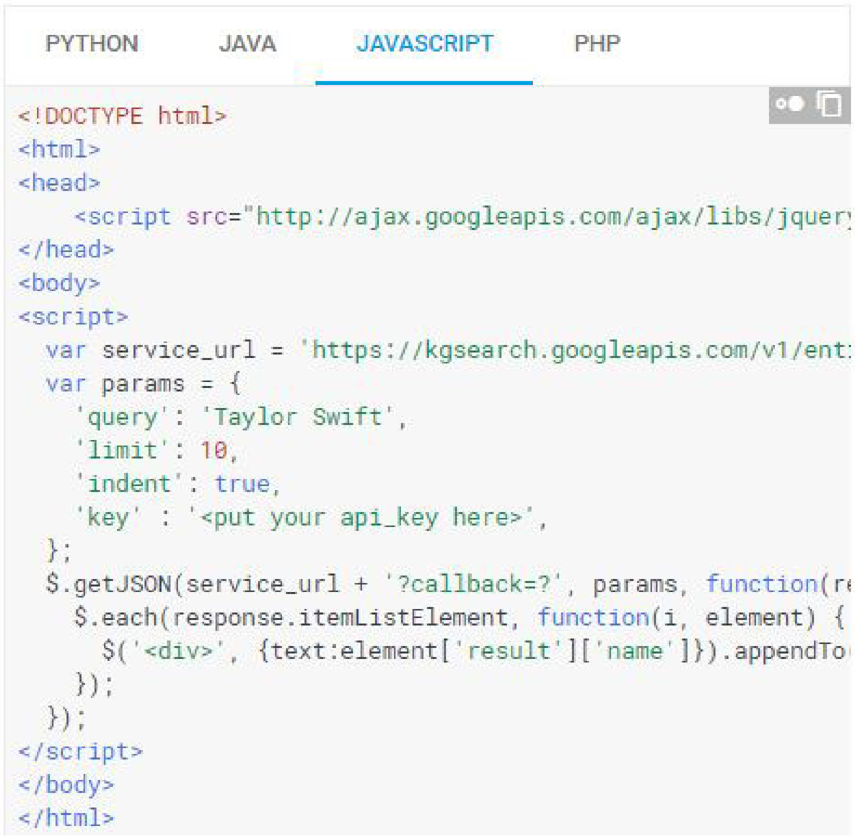
Developers tend to best grasp a REST API when its reference features a request example. To develop a complete request example for a reference, aim to

- write the request for curl
- formulate a request that seeks to cover the scope for possible calls, but you're not required to include all parameters in it
- present more than one example if a developer can define a number of parameters for a request. With several examples, you can also highlight likely combinations of parameters.

Take note that KG's documentation includes a sample request in its overview.

Though this manual does not cover how to develop them, consider two further aspects of documenting a request example:

Requests in a variety of programming languages. Instead of including only a sample request for curl, some REST API references present examples in languages such as Java, JavaScript, and Python. KG's documentation features this way of covering request examples:



```
<!DOCTYPE html>
<html>
<head>
    <script src="http://ajax.googleapis.com/ajax/libs/jquery
</head>
<body>
<script>
  var service_url = 'https://kgsearch.googleapis.com/v1/ent
  var params = {
    'query': 'Taylor Swift',
    'limit': 10,
    'indent': true,
    'key' : '<put your api_key here>',
  };
  $.getJSON(service_url + '?callback=?', params, function(re
    $.each(response.itemListElement, function(i, element) {
      $('<div>', {text:element['result']['name']}).appendTo
    });
  });
</script>
</body>
</html>
```

API explorers. REST API documentation often contains a way to test a resource, referred to as an API explorer. Readers can input parameters for a request, and the documentation returns a response. Implementing a REST API is a beneficial way for developers to grasp and evaluate it.

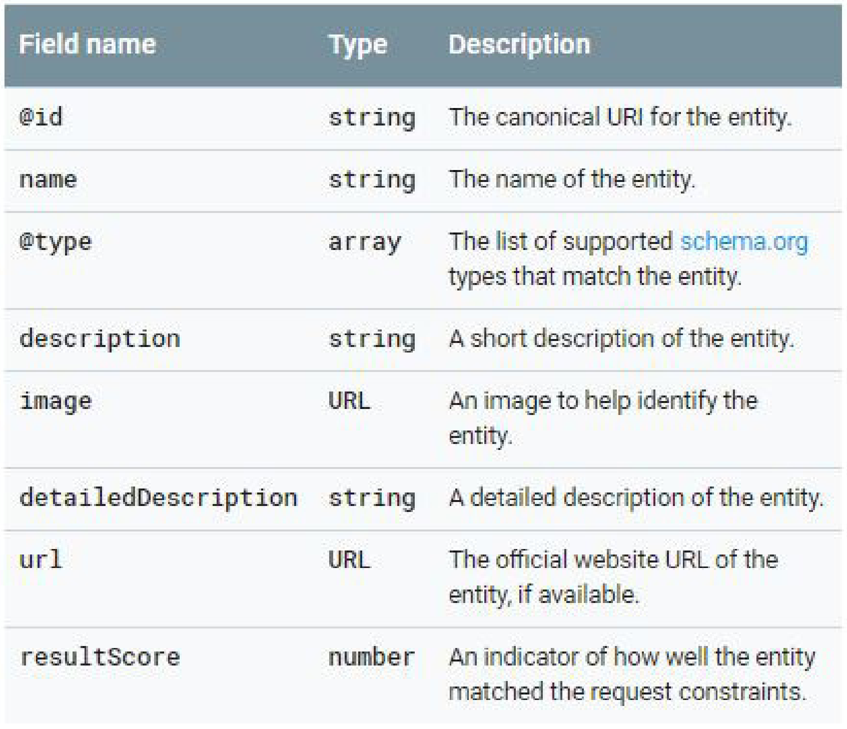
Response example

Following the rationale for the request example, include a response example so that developers can get an impression of a resource's information and how it is structured and identified. In a REST API reference, a response example consists of two parts: the example itself and a response schema. The response schema comprehensively presents all the objects and arrays a response can contain.

To develop a complete response example, aim to

- present the response that the REST API returns from the request example
- for the response schema, describe each key and note its data type.

Look at how KG's documentation presents a response schema:



Field name	Type	Description
`@id`	`string`	The canonical URI for the entity.
`name`	`string`	The name of the entity.
`@type`	`array`	The list of supported schema.org types that match the entity.
`description`	`string`	A short description of the entity.
`image`	`URL`	An image to help identify the entity.
`detailedDescription`	`string`	A detailed description of the entity.
`url`	`URL`	The official website URL of the entity, if available.
`resultScore`	`number`	An indicator of how well the entity matched the request constraints.

A table is a typical method for presenting a response schema. If you return to *Get familiar with making a call,* you'll notice that this table doesn't document every key for a response from KG. The keys not included in this table belong to the JSON-LD specification. You can compare this specification with the role DITA plays for XML documentation. KG's documentation prompts its reader to consult the specification for descriptions of the excluded keys.

When using a table to document a response method, consider this way of presenting key names:

Use slashes. You can use slashes to represent how the keys that make up objects and arrays are nested within one another. A table following this method presents the key from the KG response *articleBody* as *result/detailedDescription/articleBody*.

Going forward

Check out these resources to advance your instruction in how to document REST APIs:

Documenting APIs: A guide for technical writers. This comprehensive course on writing REST API documentation covers everything from the core concepts of requests and responses to methods for publishing documentation. Plus it puts you in the shoes of a developer for an experiential understanding of using REST APIs. You can find it at http://idratherbewriting.com/learnapidoc.

Intercom, September 2014. If you're a member of the Society for Technical Communication, you can download issues of their magazine *Intercom*. This issue is devoted to API documentation. It includes articles that introduce API documentation, suggest properties of good documentation, and offer advice for how to secure employment documenting APIs and be effective at the job. You can find the issue at stc.org.

The Programmable Web. This is a well-known outlet that covers the business of web-based APIs and compiles a directory of available APIs. You can visit it at programmableweb.com

Index

A

array
  definition 7
authentication
  definition 4

C

call. **See also** curl
  introduction to how to make a 2–5
curl
  how to make a call with 5–6

E

endpoint
  definition 4
  how to document 10

J

JSON
  definition 6–7

M

method
  definition 4
  how to document 10

O

object
  definition 7

P

parameter
  definition 4
  how to document 10–12

R

reference
  how to document 9–14
  parts of 9
request
  how to document an example of 12–13
  introduction to how to make. **See** call

resource
  definition 1
  how to document 9
response
  format of. **See** JSON
  how to document an example of 13–14
  introduction to a 2–3
REST API
  definition 1
  introduction to how to document 8–14
  to learn more about 15

U

user guide
  conventional sections of 8
  how to write an overview for 8

References

Johnson, Tom. "Documenting APIs: A guide for technical writers." *I'd rather be writing, 2017*, http://idratherbewriting.com/learnapidoc/. Accessed 27 May 2018.

Intercom: the magazine of The Society of Technical Communication, volume 61, issue 8, September 2014.