Sample data grabbed from <https://developer.okta.com/blog/2021/02/03/api-key-best-practices-and-examples>

class WeatherModel : ObservableObject {

private let apiKey: String

private let apiUri: String

@Published var weatherData: WeatherData

init(location: String) {

apiKey = "a1b2c33d4e5f6g7h8i9jakblc"

apiUri = "http://api.openweathermap.org/data/2.5/weather?q=" + location + "&appid=" + apiKey

updateWeather()

}

}

*<?xml version="1.0" encoding="UTF-8"?>*

*<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">*

<plist version="1.0">

<dict>

<key>API\_KEY</key>

<string>a1b2c33d4e5f6g7h8i9jakblc</string>

</dict>

</plist>

class WeatherModel:ObservableObject {

private let apiKey: String

private let apiUri: String

@Published var weatherData: WeatherData

init(location: String) {

guard let filePath = Bundle.main.path(forResource: "Open-Weather-Map-Info", ofType: "plist") else {

fatalError("Couldn't find file 'Open-Weather-Map-Info.plist'.")

}

let plist = NSDictionary(contentsOfFile: filePath)

guard let key = plist?.object(forKey: "API\_KEY") as? String else {

fatalError("Couldn't find key 'API\_KEY' in 'Open-Weather-Map-Info.plist'.")

}

apiKey = key

apiUri = "http://api.openweathermap.org/data/2.5/weather?q=" + location + "&appid=" + apiKey

updateWeather()

}

Sample from <https://swagger.io/docs/specification/authentication/api-keys/>

## API Keys

Some APIs use API keys for authorization. An API key is a token that a client provides when making API calls. The key can be sent in the query string:

1. GET /something?api\_key=abcdef12345

or as a request header:

1. GET /something HTTP/1.1
2. X-API-Key: abcdef12345

or as a [cookie](https://swagger.io/docs/specification/authentication/cookie-authentication/):

1. GET /something HTTP/1.1
2. Cookie: X-API-KEY=abcdef12345

API keys are supposed to be a secret that only the client and server know. Like [Basic authentication](https://swagger.io/docs/specification/authentication/basic-authentication/), API key-based authentication is only considered secure if used together with other security mechanisms such as HTTPS/SSL.

### Describing API Keys

In OpenAPI 3.0, API keys are described as follows:

1. openapi: 3.0.0
2. ...
3. # 1) Define the key name and location
4. components:
5. securitySchemes:
6. ApiKeyAuth: # arbitrary name for the security scheme
7. type: apiKey
8. in: header # can be "header", "query" or "cookie"
9. name: X-API-KEY # name of the header, query parameter or cookie
10. # 2) Apply the API key globally to all operations
11. security:
12. - ApiKeyAuth: [] # use the same name as under securitySchemes

This example defines an API key named X-API-Key sent as a request header X-API-Key: <key>. The key name *ApiKeyAuth* is an arbitrary name for the security scheme (not to be confused with the API key name, which is specified by the name key). The name *ApiKeyAuth* is used again in the security section to apply this security scheme to the API. **Note:** The securitySchemes section alone is not enough; you must also use security for the API key to have effect. security can also be set on the operation level instead of globally. This is useful if just a subset of the operations need the API key:

1. paths:
2. /something:
3. get:
4. # Operation-specific security:
5. security:
6. - ApiKeyAuth: []
7. responses:
8. '200':
9. description: OK (successfully authenticated)

Note that it is possible to support multiple authorization types in an API. See [Using Multiple Authentication Types](https://swagger.io/docs/specification/authentication/#multiple).

### Multiple API Keys

Some APIs use a pair of security keys, say, API Key and App ID. To specify that the keys are used together (as in logical AND), list them in the same array item in the security array:

1. components:
2. securitySchemes:
3. apiKey:
4. type: apiKey
5. in: header
6. name: X-API-KEY
7. appId:
8. type: apiKey
9. in: header
10. name: X-APP-ID
11. security:
12. - apiKey: []
13. appId: [] # <-- no leading dash (-)

Note the difference from:

1. security:
2. - apiKey: []
3. - appId: []

which means either key can be used (as in logical OR). For more examples, see [Using Multiple Authentication Types](https://swagger.io/docs/specification/authentication/#multiple).

### 401 Response

You can define the 401 “Unauthorized” response returned for requests with missing or invalid API key. This response includes the WWW-Authenticate header, which you may want to mention. As with other common responses, the 401 response can be defined in the global components/responses section and referenced elsewhere via $ref.

1. paths:
2. /something:
3. get:
4. ...
5. responses:
6. ...
7. '401':
8. $ref: "#/components/responses/UnauthorizedError"
9. post:
10. ...
11. responses:
12. ...
13. '401':
14. $ref: "#/components/responses/UnauthorizedError"
15. components:
16. responses:
17. UnauthorizedError:
18. description: API key is missing or invalid
19. headers:
20. WWW\_Authenticate:
21. schema:
22. type: string

Sample from <https://cloud.google.com/docs/authentication/api-keys>

## Creating an API key

To create an API key:

1. Navigate to the [APIs & Services→Credentials](https://console.cloud.google.com/apis/credentials) panel in Cloud Console.
2. Select **Create credentials**, then select **API key** from the dropdown menu.

The **API key created** dialog box displays your newly created key.

An API key is a long string containing upper and lower case letters, numbers, and dashes, such as a4db08b7-5729-4ba9-8c08-f2df493465a1.

You should copy your key and keep it secure. Unless you are using a testing key that you intend to delete later, add [application and API key restrictions](https://cloud.google.com/docs/authentication/api-keys#api_key_restrictions).

## Using an API key

Pass the API key into a REST API call as a query parameter with the following format. Replace API\_KEY with your API key,

key=API\_KEY

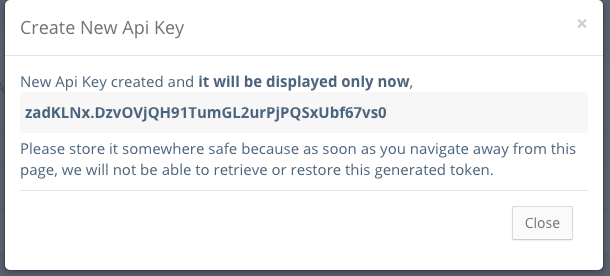
For example, to pass an API key for a Cloud Natural Language API request for [documents.analyzeEntities](https://cloud.google.com/natural-language/docs/reference/rest/v1/documents/analyzeEntities):

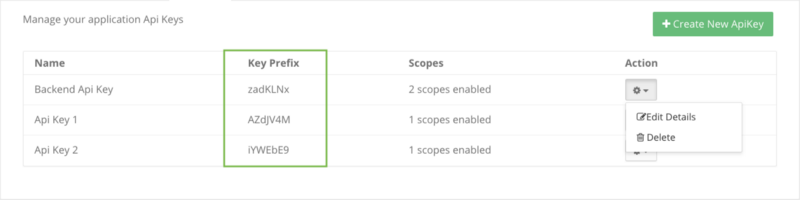
POST https://language.googleapis.com/v1/documents:analyzeEntities?key=API\_KEY

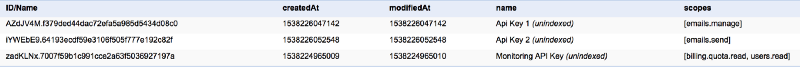
Sample from: <https://www.freecodecamp.org/news/best-practices-for-building-api-keys-97c26eabfea9/>

### API Key Generation

Since the API key itself is an identity by which to identify the application or the user, it needs to be unique, random and non-guessable. API keys that are generated must also use Alphanumeric and special characters. An example of such an API key is zaCELgL.0imfnc8mVLWwsAawjYr4Rx-Af50DDqtlx.







Sample from: <https://api.data.gov/docs/api-key/>

# **API Key Usage**

After [signing up](https://api.data.gov/signup), you'll be given your own, unique API key. This 40 character string is your API key. The key:

* Uniquely identifies you.
* Gives you access to all APIs from agencies participating in api.data.gov's service.
* Should be kept private and should not be shared.

## Ways to Pass Your API Key

Your API key may be passed to the service in a few different ways. Pick which ever method is easiest for you.

### HTTP Header

Pass the API key into the X-Api-Key header:

curl -H 'X-Api-Key: DEMO\_KEY' 'https://developer.nrel.gov/api/alt-fuel-stations/v1.json?limit=1'

### GET Query Parameter

In some cases, the API may support passing the API key into the api\_key GET query string parameter:

curl 'https://developer.nrel.gov/api/alt-fuel-stations/v1.json?limit=1&api\_key=YOUR\_KEY\_HERE'

https://developer.nrel.gov/api/alt-fuel-stations/v1.json?limit=1&api\_key=YOUR\_KEY\_HERE

Note: The GET query parameter may be used for non-GET requests (such as POST and PUT).

### HTTP Basic Auth Username

As an alternative, pass the API key as the username (with an empty password) using HTTP basic authentication:

curl 'https://YOUR\_KEY\_HERE@developer.nrel.gov/api/alt-fuel-stations/v1.json?limit=1'

[*Help Improve this Content*](https://github.com/18F/api.data.gov/blob/main/source/docs/api-key.html.md)