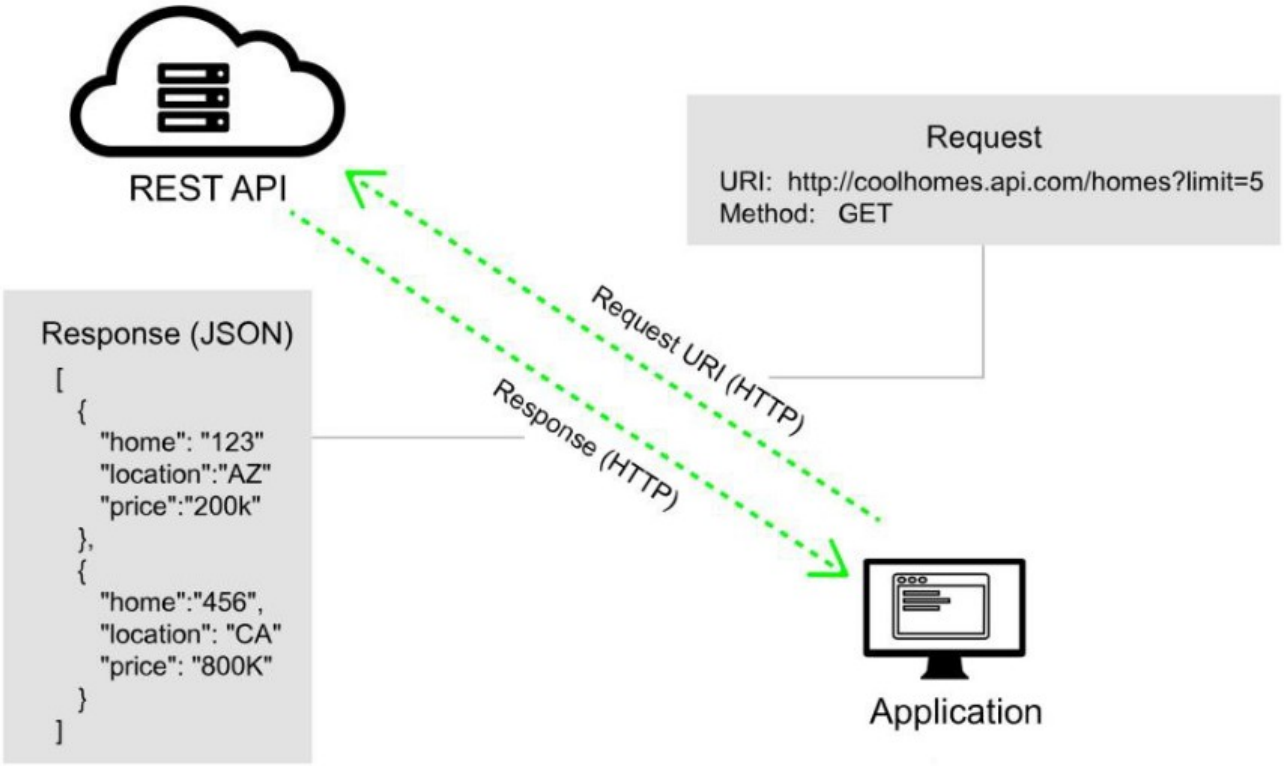


REST&JSON

The REpresentational State Transfer (REST) style and the JavaScript Object Notation (JSON) data interchange format

REST

- A style of software architecture for distributed systems
- Platform-independent
  - you don't care if the server is Unix, the client is a Mac, or anything else
- Language-independent
  - C# can talk to Java, etc…
- Standards-based
  - runs on top of HTTP
- Can easily be used in the presence of firewalls

What is a Resource?

- A resource can be anything that has identity
  - a document or image
  - a service, e.g., "today's weather in New York"
  - a collection of other resources
  - non-networked objects (e.g., people)

- The resource is the **conceptual mapping** to an entity or set of entities, not necessarily the entity that corresponds to that mapping at any particular point in time!

Main Principles

- Resource: source of specific information
- Mapping: Resources ⇔ URIs
- Client and server exchange representations of the resource
  - the same resource may have different representations
  - e.g., XML, JSON, HTML, RDF, …
- Operations on the Resource is done by means of HTTP methods
  - GET, POST, PUT, DELETE
    - Mapping to CRUD: create, read, update, delete

Main Types of Resources

- **Collection** resource
  - Represents a set (or list) of items
  - Format: /resource
  - e.g., http://api.uniupo.it/students
    http://api.uniupo.it/courses
- **Element**(Item) resource
  - Represents a single item, and its properties
  - Format: /resource/identifier
  - e.g., http://api.uniupo.it/students/s123456
    http://api.uniupo.it/courses/S1729

Best Practice

- Nouns (not verbs)
- Plural nouns
- Concrete names (not abstract)
  - /courses, not /items

Actions use HTTP Methods

- GET
  - Retrieve the representation of the resource (in the HTTP response body)
  - Collection: the list of items
  - Element: the properties of the element
- POST
  - Create a new resource (data in the HTTP request body)
  - Use a URI for a Collection
- PUT
  - Update an existing element (data in the HTTP request body)
  - Mainly for elements' properties
- DELETE

Actions on Resources: Example

RESOURCE	GET	POST	PUT	DELETE
/dogs	List dogs	Create a new dog	Bulk update dogs(avoid)	Delete all dogs(avoid)
/dogs/1234	Show info about thedog with id 1234	ERROR	If exists, update the info about dog #1234	Delete the dog #1234

Relationships

- A given Element may have a (1:1 or 1:N) relationship with other Element(s)
- Represent with: /resource/identifier/resource
- e.g.,

  http://api.uniupo.it/students/s123456/courses
  http://api.uniupo.it/courses/S1729/students

Representations

- Returned in GET, sent in PUT/POST
- Different formats are possible
- Mainly: XML, JSON
  - But also: SVG, JPEG, TXT, …
  - In POST: URL-encoding
- Format may be specified in
  - Request headers
    - Accept: application/json
  - URI extension
    - http://api.uniupo.it/students/s123456.json
  - Request parameter
    - http://api.uniupo.it/students/s123456?format=json

Real Life: GitHub API

GitHub Docs

English

Search topics, products...

All products

REST API

OVERVIEW

- Resources in the REST API
- Media types
- Other authentication methods
- Troubleshooting
- API previews
- Libraries
- OpenAPI description
- Endpoints available for GitHub Apps

REFERENCE

GUIDES

REST API / Overview / Resources in the REST API

Article version: GitHub.com

Resources in the REST API

Learn how to navigate the resources provided by the GitHub API.

This describes the resources that make up the official GitHub REST API. If you have any problems or requests, please contact GitHub Support or GitHub Premium Support.

Current version

By default, all requests to `https://api.github.com` receive the **v3** version of the REST API. We encourage you to explicitly request this version via the `Accept` header.

```
Accept: application/vnd.github.v3+json
```

For information about GitHub's GraphQL API, see the v4 documentation. For information about migrating to GraphQL, see "Migrating from REST."

Schema

All API access is over HTTPS, and accessed from `https://api.github.com`. All data is sent and received as JSON.

```
$ curl -I https://api.github.com/users/octocat/orgs

> HTTP/1.1 200 OK
> Server: nginx
> Date: Fri, 12 Oct 2012 23:33:14 GMT
> Content-Type: application/json; charset=utf-8
> Status: 200 OK
```

In this article

- Current version
- Schema
- Authentication
- Parameters
- Root endpoint
- GraphQL global node IDs
- Client errors
- HTTP redirects
- HTTP verbs
- Hypermedia
- Pagination
- Rate limiting
- User agent required
- Conditional requests
- Cross origin resource sharing
- JSON-P callbacks
- Timezones

https://docs.github.com/en/rest/overview/resources-in-the-rest-api

Real Life: Twitter API

Developer | Use cases | Solutions | Products | Docs | Community | Updates | Support | Apply | Sign in

Documentation

Search the docs

- Getting started
- Fundamentals
- Tools and libraries
- Tutorials
- API reference Index

- Twitter API
- Twitter Ads API
- Twitter for Websites
- Twitter Developer Labs

API reference index

The API reference index is a central list of all endpoints included on the Twitter Developer Platform across our different APIs.

Jump to...

- Twitter API v2
- Twitter API - Enterprise
- Twitter API - Premium v1.1
- Twitter API - Standard v1.1
- Twitter Ads API
- Labs
- Platform-wide

Twitter API v2: Early Access

https://developer.twitter.com/en/docs/api-reference-index

Real Life: Google Calendar API

Google Calendar API

Search | English | D

Home | Guides | Reference | Samples | Support

REST Reference
- Resource Summary
- Acl
- CalendarList
- Calendars
- Channels
- Colors
- Events
- Freebusy
- Settings

Client Library Reference
- Browser
- Go
- Java
- .NET
- Node.js
- PHP
- Python
- Ruby

Home > Products > Calendar API > Reference

Rate and review

Send feedback

API Reference

This API reference is organized by resource type. Each resource type has one or more data representations and one or more methods.

Resource types

Acl

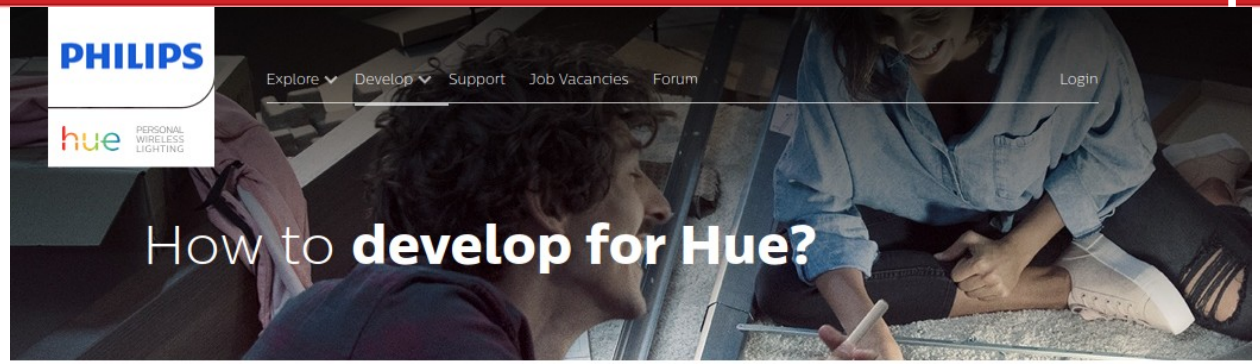
For Acl Resource details, see the resource representation page.

Method	HTTP request	Description
URIs relative to https://www.googleapis.com/calendar/v3, unless otherwise noted		
delete	DELETE /calendars/*calendarId*/acl/*ruleId*	Deletes an access control rule.
get	GET /calendars/*calendarId*/acl/*ruleId*	Returns an access control rule.
insert	POST /calendars/*calendarId*/acl	Creates an access control rule.
list	GET /calendars/*calendarId*/acl	Returns the rules in the access control list for the calendar.

Table of contents
- Resource types
- Acl
- CalendarList
- Calendars
- Channels
- Colors
- Events
- Freebusy
- Settings

https://developers.google.com/calendar/v3/reference

Real Life: Philips Hue API



Develop | Get Started | Application Design Guidance | Hue API | Hue Entertainment | Tools and SDKs

Get Started
Core Concepts
Important Notes

Get Started

The fastest way to learn how to build apps which control the hue system is to use the simple test web app built into every bridge. This lets you directly input commands and send them to the lights. You can look at the source HTML and JavaScript code for directions on how to do something different.

Follow 3 Easy Steps

Step 1

First make sure your bridge is connected to your network and is functioning properly. Test that the smartphone app can control the lights on the same network.

Step 2

Then you need to discover the IP address of the bridge on your network. You can do this in a few ways.

On this page:

- Follow 3 Easy Steps
- So let's get started...
- Turning a light on and off

https://developers.meethue.com/develop/get-started-2/
Login required

Complex resource search

- Use ?parameter=value for more advanced resource filtering (or search)
  - E.g.,
  - https://api.twitter.com/1.1/statuses/user_timeline.json **?screen_name=twitterapi&count=2**

Errors

- When errors or exceptions are encountered, use meaningful HTTP Status Codes
  - The Response Body may contain additional information (e.g., informational error messages)

```
{
    "developerMessage" : "Verbose, plain language description of
     the problem for the developer with hints about how to fix it.",
    "userMessage" : "Pass this message on to the app user if
     needed.",
    "errorCode" : 12345,
     "more info" : "http://dev.teachdogrest.com/errors/12345"
}
```

Authentication

- Twitter Streaming API

Authorization: Oauth version 1 and 2
oauth_consumer_key="xvz1evFS4wEEPTGEFPHBog", ...

- Amazon Web Services API

Authorization: AWS based on keyed-HMAC (Hash Message Authentication Code)
AKIAIOSFODNN7EXAMPLE:frJIUNo//yllqDzg=

- Google API

Authorization: Oauth version 2 from the Google API Console

Guidelines(1/2)

URL Design	
Plural nouns for collections	/dogs
ID for entity	/dogs/1234
Associations/	owners/5678/dogs
HTTP Methods	POST GET PUT DELETE
Bias toward concrete names	/dogs (not animals)
Multiple formats in URL	/dogs.json /dogs.xml
Paginate with limit and offset	?limit=10&offset=0
Query params	?color=red&state=running
Partial selection	?fields=name,state
Use medial capitalization	"createdAt": 1320296464myObject.createdAt;
Use verbs for non-resource requests	/convert?from=EUR&to=CNY&amount=100
Search	/search?q=happy%2Blabrador

Guidelines(2/2)

Versioning	
Include version in URL	/v1/dogs
Keep one previous version long enough for developers to migrate	/v1/dogs /v2/dogs

Errors	
Status Codes	200 201 304 400 401 403 404 500
Verbose messages	{"msg":"verbose, plain language hints"}

JSON –What is it?

- “JSON (JavaScript Object Notation) is a lightweight data interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate”

  -JSON.org

- Important:
  - JSON is a subset of JavaScript

JSON Logical Structure

- JSON is built on two structures:
  - A **collection** of name/value pairs. In various languages, this is realized as an **object**, record, struct, dictionary, hash table, keyed list, or associative array.

    { ... }
  - An **ordered list** of values. In most languages, this is realized as an **array**, vector, list, or sequence.

    [...]

JSON
What does it look like?

```
{
    "firstName": "John",
    "lastName": "Smith",
    "address": {
        "streetAddress": "21 2nd Street",
        "city": "New York",
        "state": "NY",
        "postalCode": 10021
    },
    "phoneNumbers": [
        "212 555-1234",
        "646 555-4567"
    ]
}
```

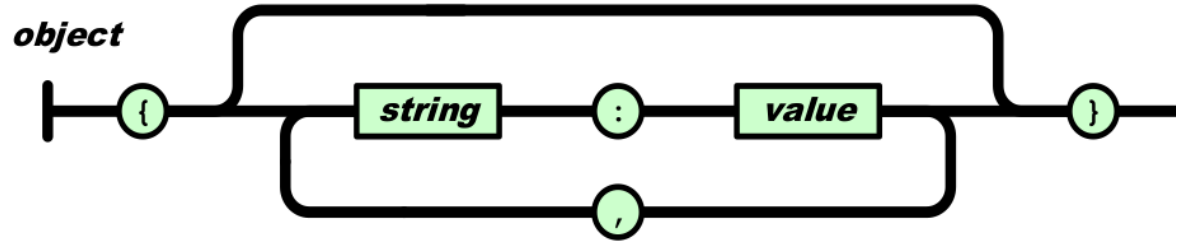
Name/Value Pairs

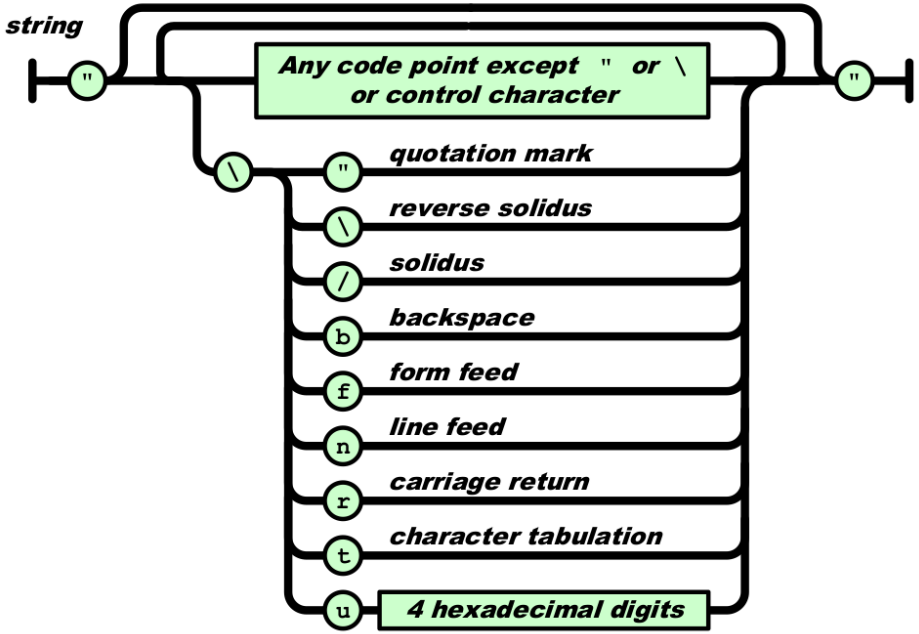
Child properties

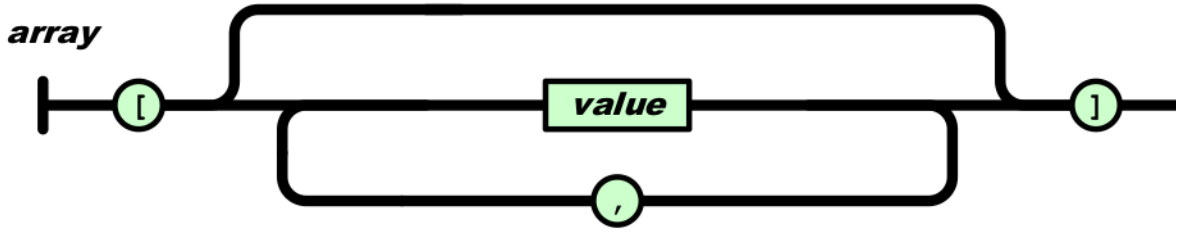
String Array

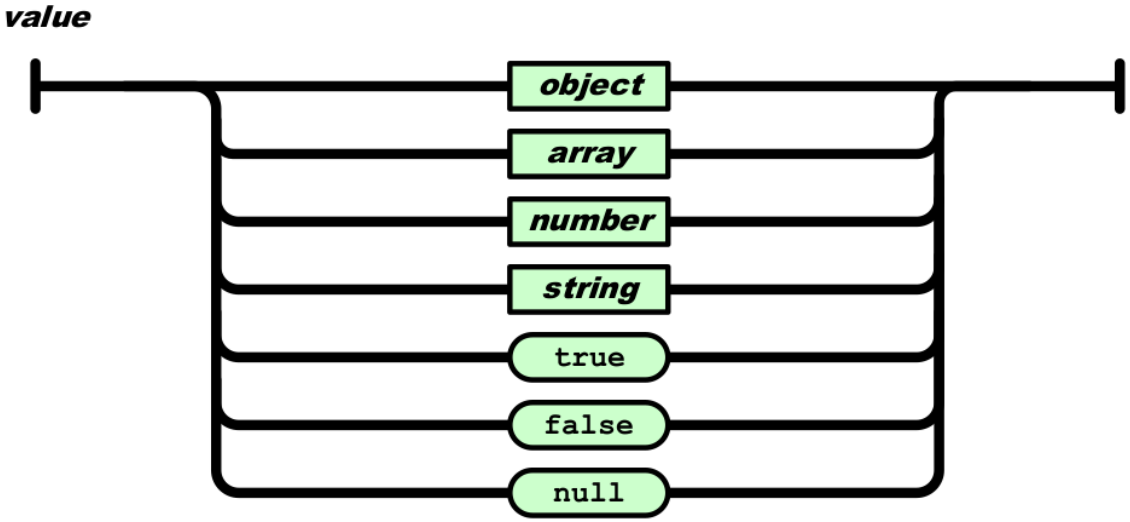
Number data type

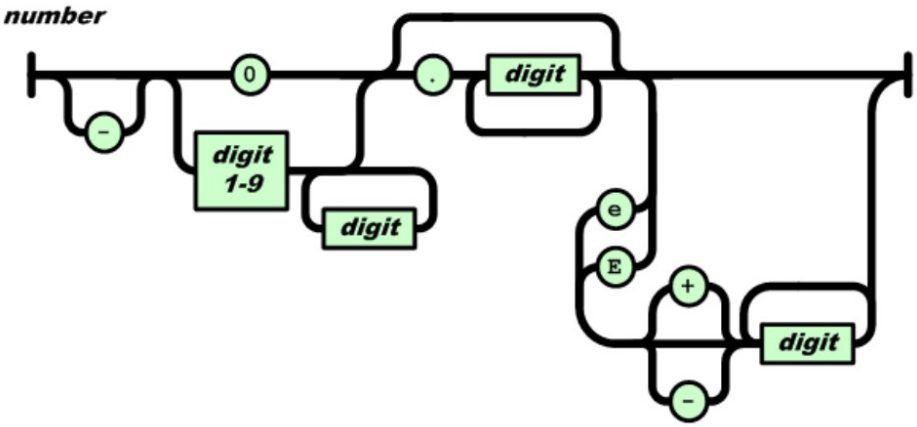
JSON Data structures











References (1/2)

- REST
  - http://en.wikipedia.org/wiki/Representational_state_transfer
  - R. Fielding, Architectural Styles and the Design of Network-based Software Architectures,

    https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm
  - https://pages.apigee.com/ebook-web-api-design-registration.html
  - http://www.slideshare.net/apigee/api-design-3rd-edition
  - https://cloud.google.com/apis/design/

References (2/2)

- JSON
  - http://json.org
  - ECMA-404 The JSON Data Interchange Standard

    https://www.ecma-international.org/publications-and-standards/standards/ecma-404/

License