Contents

[HTTP Methods 1](#_Toc118416072)

[HTTP METHOD BEST PRACTICES 1](#_Toc118416073)

[Common HTTP Methods 1](#_Toc118416074)

[GET method: 1](#_Toc118416075)

[POST method: 1](#_Toc118416076)

[PUT method: 2](#_Toc118416077)

[DELETE method: 2](#_Toc118416078)

[PATCH method: 2](#_Toc118416079)

[HTTP response status codes 3](#_Toc118416080)

[Common status code 3](#_Toc118416081)

## HTTP Methods

REST is built around HTTP methods. Clients rely on them, such as browsers, applications that use Facebook or Google APIs, or any other client.

## HTTP METHOD BEST PRACTICES

HTTP method best practices are:

* Using HTTP Methods properly. Commonly used http methods are get (), post () , put (), delete ().
* Knowing about common http status code. It can help knowing if the application is working properly or not.
* Avoid using GET method for sensitive datas.

## Common HTTP Methods

HTTP defines a set of request methods to indicate the desired action to be performed for a given resource. Although they can also be nouns, these request methods are sometimes referred to as HTTP verbs. Each of them implements a different semantic, but some common features are shared by a group of them: e.g., a request method can be safe, idempotent, or cacheable.

### GET method:

Read resource operations.

**Return status code:**

**200 OK** if the response contains data.

**204 No Content** if the response contains no data.

### POST method:

Create resource operations

Read resource if URL/ query string exceeds maximum allowed characters (2,048/256).

Read resource if query parameters contain sensitive information.

**Return status code:**

**201 Created** for successful create operation.

**200 OK** for successful read operation if the response contains data.

**204 No Content** for successful read operation if the response contains no data.

### PUT method:

Update resource operations.

**Return status code:**

**200 OK** if the response contains data.

**204 No Content** if the response contains no data.

### DELETE method:

Delete resource operations.

**Return status code:**

**204 No Content** for successful delete operation.

### PATCH method:

Partial update resource operations.

**Return status code:**

**200 OK** for successful partial update operation.

## HTTP response status codes

HTTP response status codes indicate whether a specific HTTP request has been successfully completed. Responses are grouped in five classes:

1. Informational responses (100 – 199)

2. Successful responses (200 – 299)

3. Redirection messages (300 – 399)

4. Client error responses (400 – 499)

5. Server error responses (500 – 599)

## Common status code

**200 OK**

The request succeeded. The result meaning of "success" depends on the HTTP method:

**201 Created**

The request succeeded, and a new resource was created as a result. This is typically the response sent after POST requests, or some PUT requests.

**202 Accepted**

The request has been received but not yet acted upon. It is noncommittal, since there is no way in HTTP to later send an asynchronous response indicating the outcome of the request. It is intended for cases where another process or server handles the request, or for batch processing.

**203 Non-Authoritative Information**

This response code means the returned metadata is not exactly the same as is available from the origin server, but is collected from a local or a third-party copy. This is mostly used for mirrors or backups of another resource. Except for that specific case, the 200 OK response is preferred to this status.

**204 No Content**

There is no content to send for this request, but the headers may be useful. The user agent may update its cached headers for this resource with the new ones.

**205 Reset Content**

Tells the user agent to reset the document which sent this request.

**206 Partial Content**

This response code is used when the Range header is sent from the client to request only part of a resource.

**400 Bad Request**

The server cannot or will not process the request due to something that is perceived to be a client error (e.g., malformed request syntax, invalid request message framing, or deceptive request routing).

**401 Unauthorized**

Although the HTTP standard specifies "unauthorized", semantically this response means "unauthenticated". That is, the client must authenticate itself to get the requested response.

**402 Payment Required Experimental**

This response code is reserved for future use. The initial aim for creating this code was using it for digital payment systems, however this status code is used very rarely and no standard convention exists.

**403 Forbidden**

The client does not have access rights to the content; that is, it is unauthorized, so the server is refusing to give the requested resource. Unlike 401 Unauthorized, the client's identity is known to the server.

**404 Not Found**

The server cannot find the requested resource. In the browser, this means the URL is not recognized. In an API, this can also mean that the endpoint is valid but the resource itself does not exist. Servers may also send this response instead of 403 Forbidden to hide the existence of a resource from an unauthorized client. This response code is probably the most well-known due to its frequent occurrence on the web.

**405 Method Not Allowed**

The request method is known by the server but is not supported by the target resource. For example, an API may not allow calling DELETE to remove a resource.

**500 Internal Server Error**

The server has encountered a situation it does not know how to handle.

**501 Not Implemented**

The request method is not supported by the server and cannot be handled. The only methods that servers are required to support (and therefore that must not return this code) are GET and HEAD.

**502 Bad Gateway**

This error response means that the server, while working as a gateway to get a response needed to handle the request, got an invalid response.

**503 Service Unavailable**

The server is not ready to handle the request. Common causes are a server that is down for maintenance or that is overloaded. Note that together with this response, a user-friendly page explaining the problem should be sent. This response should be used for temporary conditions and the Retry-After HTTP header should, if possible, contain the estimated time before the recovery of the service. The webmaster must also take care about the caching-related headers that are sent along with this response, as these temporary condition responses should usually not be cached.

There’s no strict rule that specifies which method. should be used for which operation. It’s better to follow some certain rules widely used by the industry. Be flexible, it’s not strict requirement.

There is no hard and fast rule dictating which method should be used for which operation. It is preferable to adhere to some industry-wide guidelines. Be adaptable; there are no hard and fast rules.