# Introducing REST APIs

## A Quick Guide for Technical Communicators

TMB
TMATHBIRD.com

# Contents

# Introduction

**REST API** uses HTTP requests to GET, PUT, POST and DELETE data across web sites and web applications. Web based API is code that allows two applications to communicate.

**REST** ( *Representational state transfer)* is the web architectural style that administers client and server behaviour.

**API** (*Application program interface*) is a general protocol set deployed over software applications allowing them to interact.

Designed for web applications only, REST manages HTTP requests and responses.

**High-level Example**: a **GET** request to /userguides/ returns a list of user guides on a web knowledge base; a **POST** request to /userguides/A234 creates a user guide with the ID A234 using the body data; a **PUT** request to /user-guides/A234 updates the user guide A234 with the body data.

**Guide Objectives**

1. Designed for technical communicators, this quick reference guide acts as a starting point when documenting REST APIs.
2. The guide summarises REST APIs, the key elements required for documentation and API reference, and example API documentation.

**Important**

1. This guide uses hypothetical examples to illustrate key areas and approaches. It only provides an overview and it is advised to follow up this outline with actual API examples and references.

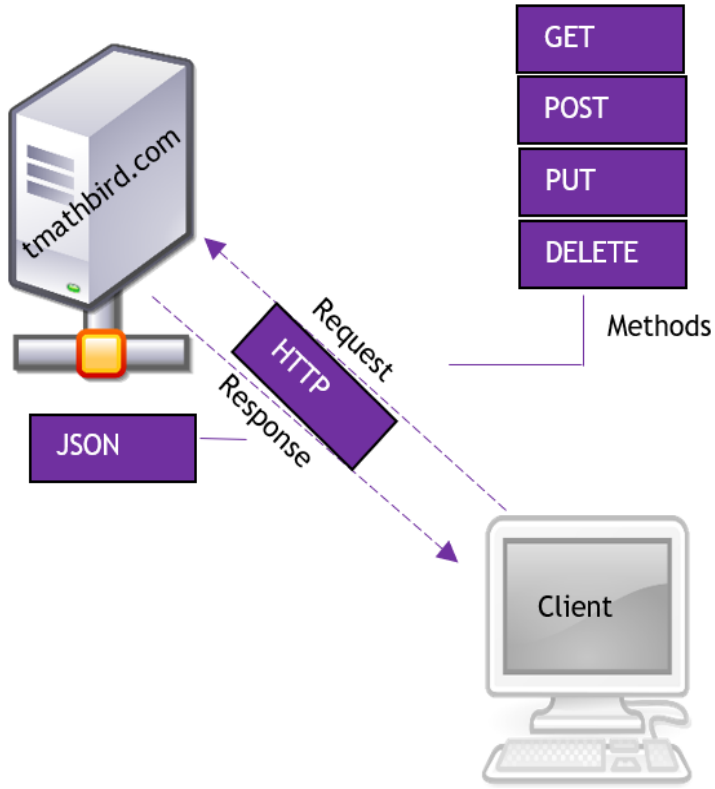
# REST API Model

As stated in the introduction, REST APIs use the HTTP protocol, typically referred to as web services.

Web services are web-based applications providing resources between the client who initiates the request and the API server that delivers the response. Using a common HTTP protocol enables requests and responses to be both language and platform independent. This is often referred to as interoperability.

When documenting REST APIs, it is important to note that individual programming languages submit requests and parse responses using language-specific functions. This is not part of the REST API.

As previously noted, REST is not a protocol but an architectural style, which may not strictly adhere to the authorised REST structure, hence why they are often referred to as RESTful APIs. One advantage of this is that there are no message format restrictions. Requests and responses can use XML, JSON, RSS, CSV, etc. However, because of its simplicity, speed and flexibility, JSON (JavaScript Object Notation) is often the format of choice.




REST accesses resources through URLs (Uniform Resource Locators).

Each URL is accompanied by a METHOD which determines how to interact with the resource.

Typical methods include GET(read), POST(create) and DELETE (remove).

And an endpoint which includes query parameters.

# Documenting APIs

One of the main tasks for technical communicators is to document API endpoints.

Information varies across organisations. However, developers generally provide endpoint details via a scrum call, a document, email or internal wiki page.

The information provided might be incomplete, outdated, too technical, or contain information for internal reference only.

The technical communicator’s role is to take this information and transform it into an accurate, complete and up-to-date API reference suitable for its target audience.

To achieve this, the recommended best practice when producing effective API documentation is to structure it into the following key sections.

| Name | Description |
|---|---|
| Resource Description | Refers to the resource information the API returns. |
| Endpoints and methods | Endpoints show how to access the resource, whereas methods show the resource's permitted interactions (GET, POST, DELETE) etc. |
| Parameters | The options that can be passed with the endpoint (e.g., response format) |
| Request example | Provides an example request including configured parameters. |
| Response example and schema | Provides an example response with the schema showing all possible elements. |

# Example endpoint

## Userguides

Contains information on available knowledge base user guides, which includes subject, author, and date published, and average user rating.

### Endpoints

GET **userguides**/{guideId}

Gets the guide information for the specified user guide

### Parameters

***Path parameters***

| Path parameter | Description |
|---|---|
| {guideId} | The value for the required user guide. Valid guideId values are available on request |

***Query String Parameters***

| String Param | Req/ Opt | Description | Type |
|---|---|---|---|
| ver | Optional | Returns details for selected version | Int |

### Sample Request

```
curl -I -X GET "https://tmathbird.com/userguides?guideId=A234
```

### Sample Response


```
{
"userguides": [
{
"guideId": "A234",
"subject": "Documenting API endpoints",
"author": "tmbcommunications",
"datepublished": 20200226,
"rating": 9
}
]
}
```

# Glossary

## A

**API**
Programming Interfaces enable software to interact with other software through exposed functionality.

**API Key**
An authorization code passed in to an API request via a header or parameter to identify the requester.

**Authentication**
Identifying the user of the API. Common techniques for authentication include API Keys and OAuth.

## C

**Cache**
A collection of responses that are reused by the client to improve performance.

**Client**
The client is the initiating party that sends an API request. Often times there will be many clients consuming the same API.

**cURL**
Command Line Interface to HTTP. Extremely popular for testing APIs and the building block for many client libraries.

## D

**Delete**
The HTTP method for deleting resources with a RESTful API.

## E

**Endpoint**
The URI that goes after the base URL and points towards the requested API functionality.

## G

**GET**
The HTTP method for retrieving resources from a RESTful API.

## H

**HTTP**
Hypertext Transfer Protocol is how websites and APIs communicate over the internet.

**HTTPS**
Hypertext Transfer Protocol Secure is how websites and APIs communicate securely over the internet.

## I

**Idempotent**
When the side-effects of multiple requests are the same as a single request. GET, PUT, and DELETE are idempotent methods.

## J

**JSON**
Javascript Object Notation is a data format commonly used for APIs requests parameters and response body.

## L

**Link**
A fully-qualified HTTP address for a particular resource (e.g., "http://my.api.com/v1/resources/resource-name"). RESTful APIs by definition should provide links from a resource to all related resources and collections which provide subsequent actions using the resource. This allows for an API to be traversed organically and for an application developer to ensure his API Client is not in (as much) jeopardy if a resource's location changed.

## M

**Mashup**
Combining multiple APIs to create a new web application.

## O

**OAuth**
Open standard authorization framework. Grants access on behalf of an end-user without directly sharing credentials.

## P

**Parameter**
A parameter is an argument sent to the API which helps define the request and expected response.

**POST**
The HTTP method for creating resources with a RESTful API.

**Protocol**
A defined way of transferring data between peers.

**PUT**
The HTTP method for updating resources with a RESTful API.

## R

**REST**
Representational state transfer is an architectural pattern for interacting with resources via HTTP methods.

## S

**SSL**
: A cryptographic protocol that secures traffic on the internet.

## U

**URI**
: Unique Resource Identifier

## V

**Versioning**
: Assinging a unique identifier to keep track of the state of the API. If changes are made to the API, the version should change.

## W

**Web Service**
: Web Service is used to describe an API that is accessable over the internet through HTTP.

## X

**XML**
: Extensible markup language is a format that is used to describe documents and data.