# Application Programming Interfaces (APIs)

> Prerequisite: Computer Networks

Today, the way most humans are familiar with interacting with computer-based information systems is through visually- and spatially-oriented interfaces known as **Graphical User Interfaces (GUI)**. GUI interactions include clicking, dragging, tapping, and other gestures.

But many systems additionally or alternatively allow programmatic usage through textually-oriented interfaces known as **Application Programming Interfaces (APIs)**. APIs provide the instructions and mechanisms for a human or computer to programmatically interact with the system.

## Web Services

**Web Services** are APIs which facilitate the transmission of a system's data across the Internet. Web services provide one or more servers which accept HTTP requests at specified URLs and return HTTP responses containing textual information in a machine-readable format.

Some notable example web services and providers include:

- New York Times APIs
- Google APIs
- Twitter APIs
- Facebook Social Graph API
- Instagram API
- Foursquare API
- GitHub API
- Yelp API
- Flickr API
- US Federal Elections Commission API
- Alpha Vantage (Stock Market) API

### Requests

#### Authentication

Many web services require developers to first register to obtain valid credentials in the form of an **API Key** (i.e. a secret token string) and subsequently authenticate by providing the key alongside each API request. This allows the service provider to understand who is issuing each request, and can help prevent or mitigate abuse of the service.

#### Request Parameters

Many APIs allow you to specify URL parameters along with your HTTP request. These URL parameters are appended to the end of the API's base URL, starting with a single question mark (`?`) to denote the rest of the URL contains parameters. Then each parameter follows a convention where the name of the parameter is followed by an equal sign (`=`), which is followed by the desired parameter value. If there are multiple parameters, subsequent parameters after the first are separated by the ampersand character (`&`).

Example request URL with parameters, where `https://www.alphavantage.co/query` is the base URL and `function`, `symbol`, `outputsize`, and `apikey` are the parameter names:

```
https://www.alphavantage.co/query?
function=TIME_SERIES_DAILY&symbol=MSFT&outputsize=compact&apikey=demo
```

> Note: you might have to register and specify your own API key if you are seeing a message like "The demo API key is for demo purposes only."

### Responses

The most common format for API response data is JSON, but some APIs alternatively or additionally provide response data in XML or CSV format.

Example CSV:

```
city,name,league
New York,Mets,Major
New York,Yankees,Major
Boston,Red Sox,Major
Washington,Nationals,Major
New Haven,Ravens,Minor
```

Example JSON:

```
[
  {"city": "New York", "name": "Yankees", "league":"Major"},
  {"city": "New York", "name": "Mets", "league":"Major"},
  {"city": "Boston", "name": "Red Sox", "league":"Major"},
  {"city": "Washington", "name": "Nationals", "league":"Major"},
  {"city": "New Haven", "name": "Ravens", "league":"Minor"}
]
```

Example XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<teams>
  <team>
    <city>New York</city>
    <league>Major</league>
    <name>Yankees</name>
  </team>
  <team>
    <city>New York</city>
    <league>Major</league>
    <name>Mets</name>
  </team>
  <team>
    <city>Boston</city>
    <league>Major</league>
    <name>Red Sox</name>
  </team>
  <team>
    <city>Washington</city>
    <league>Major</league>
    <name>Nationals</name>
  </team>
  <team>
    <city>New Haven</city>
    <league>Minor</league>
    <name>Ravens</name>
  </team>
</teams>
```

### Representational State Transfer (REST)

Most of today's web services follow an architectural pattern called "REST", which involves performing one or more operations (e.g. "create", "read", "update", "destroy") on one or more resources (usually records in a database).

> One of the key characteristics of a RESTful Web service is the explicit use of HTTP methods in a way that follows the protocol as defined by RFC 2616. HTTP GET, for instance, is defined as a data-producing method that's intended to be used by a client application to retrieve a resource, to fetch data from a Web server, or to execute a query with the expectation that the Web server will look for and respond with a set of matching resources.
>
> REST asks developers to use HTTP methods explicitly and in a way that's consistent with the protocol definition. This basic REST design principle establishes a one-to-one mapping between create, read, update, and delete (CRUD) operations and HTTP methods. According to this mapping:
>
> - To create a resource on the server, use POST.
> - To retrieve a resource, use GET.
> - To change the state of a resource or to update it, use PUT.

> - To remove or delete a resource, use DELETE.
>
> ... - IBM website

Each web service URL, sometimes also referred to as an **API Endpoint**, corresponds to a given operation and resource. For example, take these URL naming conventions for an example "photos" resource:

| HTTP Verb | Path | Operation | Used for |
|---|---|---|---|
| GET | /photos | List | display a list of all photos |
| POST | /photos | Create | create a new photo |
| GET | /photos/:id | Show | display a specific photo |
| PATCH/PUT | /photos/:id | Update | update a specific photo |
| DELETE | /photos/:id | Destroy | delete a specific photo |

> FYI: The `:id` variable represents a given resource's unique identifier.