# Basics of an API

This guide serves as a basic introduction to the syntax and structure of making an API call. Where possible, MAST APIs adhere to the following syntax and structure.

**On this page...**



APIs, or Application Programming Interfaces, are a collection of services (sometimes referred to as routes or endpoints) aimed at providing programmatic access to some data, feature, or functionality. It is common for APIs to be designed around a particular product, idea, or function, often grouping related services together. Each service is designed as an HTTP Request/Response system, i.e. "you submit a request for something, or to do something, and receive a response". A simple playground for trying out sample HTTP requests/responses is HttpBin. An alternative playground with curl and python examples is ReqBin.

Once an API is served by a host, all services become available as accessible HTTP URLs. To make an HTTP request to a service, one provides the URL name, and any allowed optional input parameters. Additional information, e.g. authorization information, or intended request format, can also be attached in the header of your request. HTTP responses typically, but not always, will return JSON objects containing the result of the API service call.

# Types of HTTP Requests

The two most common types of requests are GET and POST requests, but other HTTP Methods exist. They both involve sending a request to a URL but deal with input parameters slightly differently. Generally, the syntax for a URL is the following **scheme://host_domain/path**, e.g. https://www.google.com or https://mast.stsci.edu/portal/Mashup/Clients/Mast/Portal.html.

# GET

GET requests are simple requests to retrieve some data. Input parameters are often referred to as **query parameters**, and can be appended to the end of a URL by a **?**, with additional parameters added with an **&**, e.g. the syntax is **url?parameter=value&parameter=value**. Alternatively, when using a library to submit the request, they can be supplied separately from the URL. See https://httpbin.org/get?x=5&y=hello or HttpBin Get for an example.

## Example API call with GET

Example GET HTTP requests with a variety of libraries.

**web browser** - can only make GET requests

Paste into browser url bar: https://httpbin.org/get?x=5&y=hello

**curl** - common library for the shell command-line

```
# parameters in url
curl -X GET "https://httpbin.org/get?x=5&y=hello"

# parameters supplied
curl -X GET "https://httpbin.org/get" -G -d 'x=5' -d 'y=hello'
```

**httpie -** Python library that provides an alternative command-line interface to curl

```
# parameters in url
http get "https://httpbin.org/get?x=5&y=hello"

# parameters supplied
http get "https://httpbin.org/get" x==5 y==hello
```

**Python requests library**

```
# parameters in url
import requests
r = requests.get("https://httpbin.org/get?x=5&y=hello")

# parameters supplied
r = requests.get("https://httpbin.org/get", params={"x": 5, "y": "hello"})
```

## POST

POST requests are requests to retrieve or change some data. While similar to GET requests, they differ in a few ways. One, the input parameters are not appended to the end of the URL as in GET requests, and two, they cannot be submitted by the url bar of a front-end web browser. Input parameters are instead passed as form, JSON, or dictionary parameters in the request itself, and are often referred to as **data, form, or json parameters**. See HttpBin Post for an example.

### Example API call with POST

Example POST HTTP requests with a variety of libraries.

**curl** - common library for the shell command-line

```
# as a form
curl -X POST "https://httpbin.org/post" -d 'x=5' -d 'y=hello'

# as a JSON
curl -X POST "https://httpbin.org/post" -d 'x=5' -d 'y=hello' -H 'Content-Type: application/json'
```

**httpie -** Python library that provides an alternative command-line interface to curl

```
# as a form
http post "https://httpbin.org/post" x=5 y=hello -f

# as a JSON
http post "https://httpbin.org/post" x=5 y=hello
```

**Python requests library**

```
# as a form
import requests
r = requests.post("https://httpbin.org/post", data={"x": 5, "y": "hello"})

# as a JSON
r = requests.post("https://httpbin.org/post", json={"x":5, "y":"hello"})
```

# HTTP Responses

After you make a request, you generally receive an HTTP response.  Like requests, HTTP responses have headers, providing contextual information, and a body content with output response data. The success or failure of an HTTP response is indicated with a status code. A successful response usually has a status code of **200**. Other status codes indicate the type of problem that occurred. For example, a status code of **404** means the requested URL does not exist. See HTTP Status Codes for a complete list of status codes and their categories. A good API will try to account for various types of problems, and return a proper failure status code that informs you of what went wrong. A status code of **500** means something has gone wrong on the server that has not been accounted for, or is an uncaught error.

HTTP response data can be in a wide variety of formats. These formats are called Mime-Types and are usually specified in the Content-Type field of the response header, which indicates the type of content found in the response. For APIs, a common response data format is JSON, which has a Content-Type of "**application/json**". Other example formats might be plain text, html, or binary data such as images or files. See Common Mime-Types for a list of common response data formats.

# Example Responses

Examples of response data from the browser or from specific libraries. Most of the examples shown here highlight a JSON response format. See https://httpbin.org/json or HttpBin JSON for an example.  The below example shows the response using the Python requests library. Performing the same request in **httpie** or **curl**, or pasting the request in your browser, will display a similar response.

**Python requests library**

When you send a request in Python, you get a Response object back. The raw body content is found in the **r.content** attribute. If the response is in JSON format, you can conveniently convert the content to JSON with the **r.json()** method. The following code block shows an example response for the example POST request: **r=requests.post("https://httpbin.org/json")**

**example python JSON response**

```
# send the request and get a response
r = requests.post("https://httpbin.org/json")
r
<Response [200]>

# get the JSON response data
r.json()
{'slideshow': {'author': 'Yours Truly',
  'date': 'date of publication',
  'slides': [{'title': 'Wake up to WonderWidgets!', 'type': 'all'},
   {'items': ['Why <em>WonderWidgets</em> are great',
     'Who <em>buys</em> WonderWidgets'],
    'title': 'Overview',
    'type': 'all'}],
  'title': 'Sample Slide Show'}}
```

Here is an example response of content that is html instead of JSON, to highlight extracting the raw body content instead, from: **r = requests.post("https://httpbin.org/html")**. See https://httpbin.org/html or HttpBin Html. This example has a Content-Type of **text/html.**

**example python html response**

```
# send the request and get a response
r = requests.post("https://httpbin.org/html")
r
<Response [200]>

# get the HTML content
r.content
b"<!DOCTYPE html>\n<html>\n  <head>\n  </head>\n  <body>\n      <h1>Herman Melville - Moby-Dick<
/h1>\n\n      <div>\n        <p>\n          Availing himself of the mild, summer-cool weather that now
reigned in these latitudes, and in preparation for the peculiarly active pursuits shortly to be anticipated,
Perth, the begrimed, blistered old blacksmith, had not removed his portable forge to the hold again, after
concluding his contributory work for Ahab's leg, but still retained it on deck, fast lashed to ringbolts by
the foremast; being now almost incessantly invoked by the headsmen, and harpooneers, and bowsmen to do some
little job for them; altering, or repairing, or new shaping their various weapons and boat furniture. Often
he would be surrounded by an eager circle, all waiting to be served; holding boat-spades, pike-heads,
harpoons, and lances, and jealously watching his every sooty movement, as he toiled. Nevertheless, this old
man's was a patient hammer wielded by a patient arm. No murmur, no impatience, no petulance did come from
him. Silent, slow, and solemn; bowing over still further his chronically broken back, he toiled away, as if
toil were life itself, and the heavy beating of his hammer the heavy beating of his heart. And so it was.
\xe2\x80\x94Most miserable! A peculiar walk in this old man, a certain slight but painful appearing yawing
in his gait, had at an early period of the voyage excited the curiosity of the mariners. And to the
importunity of their persisted questionings he had finally given in; and so it came to pass that every one
now knew the shameful story of his wretched fate. Belated, and not innocently, one bitter winter's midnight,
on the road running between two country towns, the blacksmith half-stupidly felt the deadly numbness
stealing over him, and sought refuge in a leaning, dilapidated barn. The issue was, the loss of the
extremities of both feet. Out of this revelation, part by part, at last came out the four acts of the
gladness, and the one long, and as yet uncatastrophied fifth act of the grief of his life's drama. He was an
old man, who, at the age of nearly sixty, had postponedly encountered that thing in sorrow's technicals
called ruin. He had been an artisan of famed excellence, and with plenty to do; owned a house and garden;
embraced a youthful, daughter-like, loving wife, and three blithe, ruddy children; every Sunday went to a
cheerful-looking church, planted in a grove. But one night, under cover of darkness, and further concealed
in a most cunning disguisement, a desperate burglar slid into his happy home, and robbed them all of
everything. And darker yet to tell, the blacksmith himself did ignorantly conduct this burglar into his
family's heart. It was the Bottle Conjuror! Upon the opening of that fatal cork, forth flew the fiend, and
shrivelled up his home. Now, for prudent, most wise, and economic reasons, the blacksmith's shop was in the
basement of his dwelling, but with a separate entrance to it; so that always had the young and loving
healthy wife listened with no unhappy nervousness, but with vigorous pleasure, to the stout ringing of her
young-armed old husband's hammer; whose reverberations, muffled by passing through the floors and walls,
came up to her, not unsweetly, in her nursery; and so, to stout Labor's iron lullaby, the blacksmith's
infants were rocked to slumber. Oh, woe on woe! Oh, Death, why canst thou not sometimes be timely? Hadst
thou taken this old blacksmith to thyself ere his full ruin came upon him, then had the young widow had a
delicious grief, and her orphans a truly venerable, legendary sire to dream of in their after years; and all
of them a care-killing competency.\n        </p>\n      </div>\n  </body>\n</html>"
```

# For Further Reading...

- MAST Data Holdings
- What is an API?, from RedHat