Overview
Getting Started
API key
URL
Headless Browser
Javascript Rendering
JavaScript Execution
Wait for a fixed amount of time
Wait for selector
Wait for browser
Blocking Ads
Blocking Images and CSS
Viewport width
Viewport height
Proxies
Premium proxy
Geolocation
Stealth Proxy (beta)
Own proxy
Headers

Documentation - HTML API

Overview

Here is the list of the different parameters you can use with ScrapingBee's HTML API.
You can also discover this API using our Postman collection covering every ScrapingBee's features.

Run in Postman

	Description
name (string) (default)	
api_key (string) (required)	Your api key 3EWYVLNZ8MJ5VUJBXX3ZK1YVV8BA... Learn more
url (string) (required)	The URL of the page you want to scrape Learn more
ai_extract_rules (stringified JSON) (null)	Data extraction from description using AI Learn more
ai_query (string) (null)	The information you want to extract from the webpage using AI Learn more
ai_selector (string) (null)	CSS selector to focus the AI extraction on a specific part of the page Learn more
block_ads (boolean) (false)	Block ads on the page you want to scrape Learn more
block_resources (boolean) (true)	Block images and CSS on the page you want to scrape Learn more
cookies (string) (null)	Pass custom cookies to the webpage you want to scrape Learn more
country_code (string) (null)	Premium proxy geolocation Learn more
custom_google (boolean) (false)	Set to true to scrape Google Learn more
device (string) (desktop)	Control the device the request will be sent from Learn more
extract_rules (stringified JSON) (null)	Data extraction from CSS selectors Learn more
forward_headers (boolean) (false)	Forward particular headers to the webpage, as well as other headers generated by ScrapingBee Learn more
forward_headers_pure (boolean) (false)	Forward only particular headers to the webpage, and nothing else Learn more
js_scenario (stringified JSON) (null)	JavaScript scenario to execute Learn more
json_response (bool) (false)	Wrap response in JSON Learn more
own_proxy (string) (null)	Allows you to use ScrapingBee with your own proxy provider Learn more
premium_proxy (boolean) (false)	Use premium proxies to bypass difficult to scrape websites Learn more
render_js (boolean) (true)	Render the JavaScript on the page with a headless browser Learn more
return_page_source (boolean) (false)	Return the original HTML before the JavaScript rendering Learn more
scraping_config (string) (null)	Use a pre-saved request configuration on your reques Learn more
screenshot (boolean) (false)	Return a screenshot of the page you want to scrape Learn more
screenshot_full_page (boolean) (false)	Return a screenshot of the full page you want to scrape Learn more
screenshot_selector (string) (null)	Return a screenshot of a particular area of the page, targeted by a CSS selector Learn more
session_id (integer) (null)	Route multiple API requests through the same IP address Learn more
stealth_proxy (boolean) (false)	Use special stealth proxy pool Learn more
timeout (int) (10 000)	Timeout for your requests Learn more
transparent_status_code (boolean) (false)	Transparently return the same HTTP code of the page requested Learn more
wait (integer) (0)	Additional time in ms for JavaScript to render Learn more
wait_browser (string) (domcontentloaded)	Wait until certain browser conditions are met before returning the response Learn more
wait_for (string) (null)	CSS / XPath selector to wait for in the DOM Learn more
window_height (int) (600)	Height, in pixel, of the viewport used to render the page you want to scrape Learn more
window_width (int) (1920)	Width, in pixel, of the viewport used to render the page you want to scrape Learn more

Getting Started

ScrapingBee is meant to be the easiest scraping API available on the web.

To scrape a web page, you only need two things:

- Your API key, available here
- The **encoded** web page URL you want to scrape (learn more about URL encoding)

The following snippet is an example of a simple `GET` API call to scrape the URL defined in the query string variable `YOUR-URL`:

```
cURL
app.scrapingbee.com/api/v1/?api_key=3EWYVLNZ8MJ5VUJBXX3ZK1YVV8BA0XCMMVQU5UZ5MPPE7RY1BDQKJPGF4X9F5LS7WWAFN39H5UUHK3BGFMGwa1=YOUR-URL"
```

The API will then respond with the raw HTML content of the target URL:

```
<html>
  <head>
    ...
  </head>
  <body>
    ...
  </body>
</html>
```

Every URL that failed will be tried as many times as possible for 30 seconds.

Be aware of the maximum 30-second timeout when your code calls the API.

Headers and cookies returned by the target website are prefixed with `Spb-` (for ScraPingBee).

API key

`api_key` (string) (default= **NA**) **required**

All requests are authenticated using your private API key.

To get access to your API key, create an account here and confirm your email address.

```
cURL
curl "https://app.scrapingbee.com/api/v1/?api_key=3EWY
```

URL `url` (string) (default= **NA**) **required**

This parameter is the full URL including the protocol (with `http/https`) of the page to extract data from.

You must encode your URL. For example, the `:` character is encoded to `%20`. Consult your programming language documentation for functions that encode URLs

Several examples of URL encoding can be seen below:

```
cURL
sudo apt-get install gridsite-clients
urlencode "YOUR URL"
```

Headless Browser

Javascript Rendering

`render_js` (boolean) (default= **true**)

By default, ScrapingBee fetches the URL to scrape via a headless browser that will execute the JavaScript code on the page. **This is the default behavior and costs 5 credits per request.**

This can be useful for scraping a Single Page Application built with frameworks such as React.js, Angular.js, JQuery or Vue.

To fetch the URL without using a headless browser, use the `render_js=false` parameter in the `GET` request.

The following is an example with a dummy Single Page Application (SPA).

If you use `render_js=true` (default behavior)

```
cURL
curl "https://app.scrapingbee.com/api/v1/?api_key=3EWY
```

The following result is returned:

```
<html>
  <head>
    ...
  </head>
  <body>
    <content>
    </content>
    <content>
    </content>
    <content>
    </content>
    <content>
    </content>
    <content>
    </content>
    <content>
    </content>
  </body>
</html>
```

But if you use `render_js=false` instead:

```
cURL
curl "https://app.scrapingbee.com/api/v1/?api_key=3EWY
```

This is what is returned:

```
<html>
  <head>
    ..
  </head>
  <body>
  </body>
</html>
```

JavaScript Execution

`js_scenario` (stringified JSON) (default= **{}**)

If you want to interact with pages you want to scrape before we return your the HTML you can add JavaScript scenario to your API call.

For example, if you wish to click on a button, you will need to use this scenario.

```
{
    "instructions": [
        {"click": "#buttonId"}
    ]
}
```

And so our scraper will scrape the webpage, click on the button `#buttonId` and then return you the HTML of the page.

Important: JavaScript scenario are JSON formatted, and in order to pass them to a GET request, you need to stringify them.

You can add multiple instructions to the scenario, they will get executed one by one on our end.

Below is a quick overview of all the different instruction you can use.

```
{"click": "#button_id"} # Click on an element
{"wait": 1000} # Wait for a fixed duration in ms
{"wait_for": "#slow_div"} # Wait for an element to appear
{"wait_for_and_click": "#slow_div"} # Wait for an element
{"scroll_x": 1000} # Scroll the screen in the horizontal a
{"scroll_y": 1000} # Scroll the screen in the vertical axi
{"fill": ["#input_1", "value_1"]} # Fill some input
{"evaluate": "console.log('toto')"} # Run custom JavaScrip
{"infinite_scroll": # Scroll the page until the end
    {
        "max_count": 0, # Maximum number of scroll, 0 for
        "delay": 1000, # Delay between each scroll, in m
        "end_click": {"selector": "#button_id"} # (optiona
    }
}
```

If you want to learn more about this powerful feature, you can check the full documentation here.

Wait for a fixed amount of time `wait` (integer) (default= **0**)

Some code-heavy websites need time to fully "render". To direct ScrapingBee to wait before it returns the rendered HTML, use

```
cURL
curl "https://app.scrapingbee.com/api/v1/?api_key=3EWY
```

the `wait` parameter with a value in milliseconds between 0 and 35000.

The ScrapingBee headless browsers will then wait the duration of the time set in milliseconds before returning the page's HTML.

If you need some help setting this up, do not hesitate to contact us.

Wait for selector

`wait_for` (`string`) (default= `null`).

It's sometimes necessary to wait for a particular element to appear in the DOM before ScrapingBee returns the HTML content.

Our headless browsers will wait for the CSS / Xpath selector passed in the parameter before returning the HTML.

For example, to wait for the element `<div class="loading-done">` `</div>` use `wait_for=.loading-done` in your request.

> All selectors beginning with `/` will be treated as XPath selectors. All other selectors will be treated as CSS selectors.

Please note that if you use `wait` and `wait_for` , our system will first execute `wait_for` and then `wait` . And, after `wait` , `js_scenario` is executed. If you want to control the order of `wait` and `wait_for` , you can use it in `js_scenario` . In `js_scenario` , the execution is based on the order in which you specify the instructions (learn more)

cURL

```
curl "https://app.scrapingbee.com/api/v1/?api_key=3IEVY
```

Wait for browser

`wait_browser` (`string`) (default= `domcontentloaded`).

This advanced parameter tells the browser to wait until certain network condition are met.

It can take 4 different values:

- `domcontentloaded` (default): Wait until the DOM is loaded
- `load` : Wait until the page is fully loaded
- `networkidle0` : Wait until there are no more than 0 network connections for at least 500 ms
- `networkidle2` : Wait until there are no more than 2 network connections for at least 500 ms

For example, to wait until the page is fully loaded before getting the results, you can use `wait_browser=load` .

cURL

```
curl "https://app.scrapingbee.com/api/v1/?api_key=3IEVY
```

Blocking Ads

`block_ads` (`boolean`) (default= `false`).

By default, ScrapingBee does not block ads. To avoid scraping them (e.g. to speed up your request), use `block_ads=true`

This parameter is unnecessary if JavaScript rendering is disabled.

cURL

```
curl "https://app.scrapingbee.com/api/v1/?api_key=3IEVY
```

Blocking Images and CSS

`block_resources` (`boolean`) (default= `true`).

By default, and to speed up requests, ScrapingBee blocks all images and CSS in the scraped page, but to scrape them, use `block_resources=false`

This parameter is unnecessary if JavaScript rendering is disabled.

cURL

```
curl "https://app.scrapingbee.com/api/v1/?api_key=3IEVY
```

Viewport width

`window_width` (`int`) (default= `1920`).

If you need to change the dimension of the browser's viewport (window) when scraping the target page you can use the `window_width` and `window_height` parameters.

Only useful when using `render_js=True` .

cURL

```
curl "https://app.scrapingbee.com/api/v1/?api_key=3IEVY
```

Viewport height

`window_height` (`int`) (default= `1080`).

If you need to change the dimension of the browser's viewport (window) when scraping the target page you can use the `window_width` and `window_height` parameters.

Only useful when using `render_js=True` .

cURL

```
curl "https://app.scrapingbee.com/api/v1/?api_key=3IEVY
```

Proxies

Premium proxy

`premium_proxy` (`boolean`) (default= `false`) For some hard-to-scrape websites, you may need to use premium proxies (or residential proxies). These proxies are rarely blocked and we recommend trying premium proxies when you receive error codes or difficult to scrape websites, like search engines, social networks, or hard to scrape E-commerce websites.

To scrape these sites, you need to add the parameter `premium_proxy=true` .

Each request with this parameter will count as **25 API credits with Javascript enabled.** If used without JavaScript rendering it will cost **10 credits**

cURL

```
curl "https://app.scrapingbee.com/api/v1/?api_key=3IEVY
```

Geolocation

`country_code` (`string`) (default= `us`).

In addition to premium proxies, you can also choose the proxy country from the following list of countries using the parameter `country_code=COUNTRY_CODE` .

To use premium proxies from Germany for example you need to set both `premium_proxy=true` and `country_code=de` parameters in your API call.*

You can also use stealth proxies from different countries, by setting `stealth_proxy=true` instead of `premium_proxy=True` .

The following is the list of the most popular supported country codes using (ISO 3166-1 format).

The **whole list of supported country codes** can be found here.

country_code	Country Name
br	Brazil
in	India
mx	Mexico
...	...
ru	Russia
us	UnitedStates
gb	UnitedKingdom

cURL

```
curl "https://app.scrapingbee.com/api/v1/?api_key=3IEVY
```

Stealth Proxy (beta)

`stealth_proxy` (`boolean`) (default= `false`).

Sometimes, even using `premium_proxy=True` is not enough. For hard websites to scrape, we have developed a new pool of proxies that should be enough to scrape even the hardest to scrape websites.

To use this pool, simply add `stealth_proxy=True` to your API calls. These proxies also support geolocation, so to use an IP address from Germany for example, you need to set both `stealth_proxy=true` and `country_code=de` parameters in your API call.*

Things to keep in mind if you use this option:

- this option currently only works when JavaScript rendering is enabled.
- **Each successful API call using this option will cost 75 credits** (Even if you use custom_google)

Some features are currently not supported with this option:

- The `infinite_scroll` instruction of the JavaScript scenario is not supported with this option.
- Custom headers and cookies
- The `timeout` parameter
- `evaluate_results` data when using `json_response`

Here is an example if you want to use this option.

cURL

```
curl "https://app.scrapingbee.com/api/v1/?api_key=3IEVY
```

Own proxy `own_proxy` (`string`) (default= `null`).

If you want to use our infrastructure with your own proxy, you can use the `own_proxy` parameter. This will allows you to use all the ScrapingBee features, with your own proxies.

The proxy information syntax is: `<protocol>://username:<password>@<host>:<port>`.

Things to keep in mind:

- protocol is optional
- if no `port` is specified, we will use `1080` (`cURL` default)

Here is an example if you want to use `https://johndoe:password@my_proxy.com:123` proxy.

cURL

```
curl "https://app.scrapingbee.com/api/v1/?api_key=3IEVY
```

Headers

Header Forwarding

`forward_headers` (`boolean`) (default= `false`).

You might need to forward specific headers to the website that you want to scrape.

In order to forward headers, you must set `forward_headers` to `true` and then pass your custom headers.

You must then prefix the headers to forward to the website with `"Spb-"` (for ScrapingBee).

This prefix will be trimmed by ScrapingBee and headers will be forwarded to the target web page.

Example :

If you want to send the header `Accept-Language: En-US` , add the header: `Spb-Accept-Language: En-US` and the parameter `forward_headers=true` to the request sent to the ScrapingBee API.

Note :

If you are using the ScrapingBee Python or Node library, no need to prefix headers with `"Spb-"`

or to use `forward_headers=True` .

Please note that in order to make your request look like real ones, ScrapingBee adds several headers to all API requests. Use `forward_headers_pure=True` to avoid this behavior.

cURL

```
curl "https://app.scrapingbee.com/api/v1/?api_key=3IEVY
  -H "Spb-Accept-Language:En-US"
```

Here we are scraping `httpbin.org/headers?json` , a page that demonstrates and displays the headers it received.

The following is the response from the above code. Note the `Accept-Language` header in the response.

```
{
  "headers": {
    "Accept": "text/html,application/xhtml+xml,applica
    "Accept-Encoding": "gzip, deflate",
    "Accept-Language": "fr-FR,fr;q=0.9,en-US;q=0.8,en;
    "Host": "httpbin.org",
    "Accept-Language": "En-US" # <-- Your header
    "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64
  }
}
```

Pure Header Forwarding

`forward_headers_pure` (`boolean`) (default= `false`).

If you want to forward specific headers to the website that you want to scrape, and don't need ScrapingBee to add any headers to your request, you should use `forward_headers_pure=True` .

You must then prefix the headers to forward to the website with `"Spb-"` (for ScrapingBee).

This prefix will be trimmed by ScrapingBee and headers will be forwarded to the target web page.

Example :

If you want to only send the header `Accept-Language: En-US` , add the header: `Spb-Accept-Language: En-US` and the parameter `forward_headers_pure=true` to the request sent to the ScrapingBee API.

Note :

If you are using the ScrapingBee Python or Node library, no need to prefix headers with `"Spb-"`

or to use `forward_headers=True` .

cURL

```
curl "https://app.scrapingbee.com/api/v1/?api_key=3IEVY
  -H "Spb-Accept-Language:En-US"
```

Here we are scraping `httpbin.org/headers?json` , a page that demonstrates and displays the headers it received.

The following is the response from the above code. Note the `Accept-Language` header in the response.

```
{
  "headers": {
    "Accept-Encoding": "gzip, deflate", # Technical he
    "Host": "httpbin.org", # Technical header sent al
    "Accept-Language": "En-US" # <-- Your header, and
  }
}
```

This parameter is only useful when using `render_js=False` .

Response format

By default, the API will transparently return you the resource you want to scrape.

But you can do way more.

Downloading Picture and Files

The API will transparently download images, PDF or anything that is not HTML.

We recommend downloading files with `render_js=false`.

There is a **2 MB** limit per request.

Data extraction with AI (BETA)

If you want to extract specific information from a webpage using AI, you can use the `ai_query` and `ai_selector` parameters.
The `ai_query` parameter allows you to specify the information you want to extract, while the optional `ai_selector` parameter lets you focus the AI extraction on a specific part of the page.

AI Query `ai_query` (`string`, {default= `""`})

The `ai_query` parameter allows you to specify the information you want to extract from the webpage using natural language. For example:

```
ai_query="price of the product"
```

This instructs the AI to find and extract the price of the product from the page content.

> Cost: The AI extraction parameters (`ai_query` and `ai_extract_rules`) incur an additional 5 credits cost on top of the regular API cost.
> To speed up the process of your request we encourage you to use a relevant `ai_selector` value.

AI Selector (Optional) `ai_selector` (`string`, {default= `""`})

The `ai_selector` parameter is optional and allows you to specify a CSS selector to focus the AI extraction on a specific part of the page.
This can help improve accuracy and reduce processing time. For example:

```
ai_selector="#product-details"
```

This tells the AI to only consider the content within the element with the ID "product-details" when extracting the information specified in the `ai_query`.

Using the `ai_selector` can help speed up the request by limiting the amount of content the AI needs to process.

Using both parameters together can provide more precise and efficient data extraction:

```
ai_query="price of the product"
ai_selector="#product-details"
```

cURL

```
curl "https://app.scrapingbee.com/api/v1/?api_key=3BNYYLKZ8MJ5VUJB3X3ZK17VY08AWV2MMVQUSUZWPPE7RY18OQCJPGF4X9F5L57WW4FR
```

AI data extraction

`ai_extract_rules` (`stringified json`, {default= `""`})

cURL

```
curl "https://app.scrapingbee.com/api/v1/?api_key=3BNY
```

If you want to extract data from pages and don't want to parse the HTML on your side, you can add AI extraction rules to your API call.

The simplest way to use JSON rules is to use the following format

```
{"key_name" : "what you want to extract"}
```

If you wish to extract the title and a summary of some of our blog posts, you will just need to use those rules.

```
{
    "title" : "title of the blog post",
    "summary" : "a short summary of the blog post",
}
```

And this will be the JSON response

```
{
    "title" : "How to web ",
    "summary" : "We help you get better at web-scraping: d
}
```

Important: extraction rules are JSON formatted, and in order to pass them to a GET request, you need to stringify them.

We've just described the easiest and quickest way to use this feature. You use more advanced option as described below:

```
{
    "name": {
        "description": "the product name",
        "type": "string"
    },
    "categories": {
        "description": "all product categories",
        "type": "list"
    },
    "price": {
        "description": "the product price in dollars",
        "type": "number"
    },
    "in_stock": {
        "description": "whether the product is currently a
        "type": "boolean"
    },
    "shipping_info": {
        "description": "shipping details including deliver
        "type": "item",
        "output": {
            "delivery_time": "estimated delivery in days",
            "shipping_cost": "shipping cost in dollars"
        }
    },
    "size": {
        "description": "product size",
        "type": "list",
        "enum": ["XS", "S", "M", "L", "XL"]
    },
}
```

Do not hesitate to check out the full documentation here.

> Cost: The AI extraction parameters (`ai_query` and `ai_extract_rules`) incur an additional 5 credits cost on top of the regular API cost. To speed up the process of your request we encourage you to use a relevant `ai_selector` value.

CSS / XPATH data extraction

`extract_rules` (`stringified JSON`, {default= `""`})

cURL

```
curl "https://app.scrapingbee.com/api/v1/?api_key=3BNY
```

If you want to extract data from pages and don't want to parse the HTML on your side, you can add extraction rules to your API call.

The simplest way to use JSON rules is to use the following format

```
{"key_name" : "css_or_xpath_selector"}
```

If you wish to extract the title, subtitle and intro of our blog, you will just need to use those rules.

```
{
    "title" : "h1",
    "subtitle" : "#subtitle",
}
```

And this will be the JSON response

```
{
    "title" : "The ScrapingBee Blog",
    "subtitle" : "We help you get better at web-scraping:
}
```

Important: extraction rules are JSON formatted, and in order to pass them to a GET request, you need to stringify them.

We've just described the easiest and quickest way to use this feature.
If you want to read more about it, check out our full guide.

Screenshot

`screenshot` (`boolean`, {default= `false`})

cURL

```
curl "https://app.scrapingbee.com/api/v1/?api_key=3BNY
```

If you want to get a screenshot of the page you want to scrape, use the `screenshot=True` parameter.

Screenshots are only available when using `render_js=True`.

When using `screenshot=True`, `block_resources` will automatically be set to `False` so the browser will load images and CSS before taking the screenshot.

> If you need both the screenshot of the page and the HTML content of it, use `screenshot=True` and `json_response=True` (learn more about json_response)
>
> If you need to wait for a particular amount of time, a DOM element, or a browser event, use our `wait`, `wait_for` and `wait_browser` parameters (learn more)

Screenshot a particular CSS selector

`screenshot_selector` (`string`, {default= `""`})

By default, the screenshot will only capture the portion of the page that is visible in the browser's viewport.

If you need to screenshot a particular area of the page, you can use `screenshot_selector=<CSS_selector>` where `<CSS_selector>` is the CSS selector of the area you want to capture.

Screenshot full page

`screenshot_full_page` (`boolean`, {default= `false`})

By default, the screenshot will only capture the portion of the page that is visible in the browser's viewport.

If you need a screenshot of the full page from the target website use `screenshot_full_page=True`

If you need to change the size of the browser's viewport before taking a screenshot you can do it using `window_width` and `window_height` parameters. (learn more).

JSON Response `json_response` (`boolean`, {default= `false`})

If you are planning to integrate ScrapingBee with third-party tools that only accept JSON response, or want to intercept the response of some XHR / Ajax requests, you can send your API call with `json_response=True`.

cURL

```
curl "https://app.scrapingbee.com/api/v1/?api_key=3BNYYLKZ8MJ5VUJB3X3ZK17VY08AWV2MMVQUSUZWPPE7RY18OQCJPGF4X9F5L57WW4FR
```

The following is the received response when using this parameter:

```
{
  # Headers sent by the server
  "headers": {
    "Date": "Fri, 16 Apr 2021 15:03:54 GMT",
    ...
    "Access-Control-Allow-Credentials": "true"
  },
  # Credit cost of your request
  "cost": 1,
  # Initial status code of the server
  "initial-status-code": 200,
  # Resolved URL (following redirection)
  "resolved-url": "https://httpbin.org/",
  # Type of the response "html" or "json" or "b64_bytes" for file, image, pdf...
  "type": "html",
  # Content of the answer. Content will be base 64 encoded if is a file, image, pdf...
  "body": "<html>... </body>"
  # Time at which the screenshot of the page, if screenshot=true is used
  "screenshot": "b4918...aaf",
  # Cookies sent back by the server
  'cookies': [
    {
        "name": "cookie_name",
        "value": "cookie_value",
        "domain": "test.com",
        ...
    },
    ...
  ],
  # Results of the JS scenario "evaluate" instructions
  "evaluate_results": [...]
  # Content and source of iframes in the page
  "iframes": [
    {
      "content": "<html>... </body>",
      "src": "https://site.com/iframe"
    },
    ...
  ],
  # XHR / Ajax requests sent by the browser
```

```
  "xhr": [
    {
      # URL
      "url": "https://",
      # Status code of the server
      "status_code": 200,
      # Method of the request
      "method": "POST",
      # Headers of the XHR / Ajax request
      "headers": {
        "pragma": "no-cache",
        ...
      },
      # Response of the XHR / Ajax request
      "body": "2d.x"
    },
    ...
  ],
  # js_scenario detailed report ( only useful if using render_js=True and js_scenario...)
  "js_scenario_report": {
    "task_executed": 1,
    "task_failure": 0,
    "task_success": 1,
    "tasks": [
      {
          "duration": 3.042,
          "params": 3000,
          "success": true,
          "task": "wait"
      }
    ],
    "total_duration": 3.042
  },
  # Metada / Schema data
  "metadata": {
    "microdata": ...,
    "json-ld": ...,
  }
}
```

If the requested content is json, then the answers will look like this:

```
{
  # Headers sent by the server
  "headers": {
    "Date": "Fri, 16 Apr 2021 15:13:02 GMT",
    ...
    "Access-Control-Allow-Credentials": "true"
  },
  # Credit cost of your request
  "cost": 1,
  # Initial status code of the server
  "initial-status-code": 200,
  # Resolved URL following redirections
  "resolved-url": "https://httpbin.org/anything?json",
  # Type of the response "html" or "json"
  "type": "json",
  # Content of the answer
  "body": {
    "args": {
      ....
    }
  }
  # Results of the JS scenario "evaluate" instructions
  "evaluate_results": [...]
  # XHR / Ajax requests sent by the browser
  "xhr": [
    ...
  ]
  # js_scenario detailed report ( only useful if using render_js=True and js_scenario...)
  "js_scenario_report": {
    ...
  },
  # Metada / Schema data
  "metadata": {
    "microdata": ...,
    "json-ld": ...,
  }
}
```

Return Page Source

`return_page_source` (`boolean`) (default= `false`)

To have HTML returned by the server and unaltered by the browser (before the JavaScript execution), use `return_page_source=true`

This parameter is unnecessary if JavaScript rendering is disabled.

cURL

```
curl "https://app.scrapingbee.com/api/v1/?api_key=3DVY
```

Scraping Configuration

`scraping_config` (`string`) (default= `no`)

To use a pre-saved scraping configuration, use `scraping_config=[Scraping Configuration Name]`

This parameter allows you to apply a settings configuration for any website, without having to type the settings each time you send a request. And you can find more information about it here: Preconfigured Requests

cURL

```
curl "https://app.scrapingbee.com/api/v1/?api_key=3DVY
```

HTML to PDF conversion

Coming soon.

Proxy Mode

ScrapingBee also offers a proxy front-end to the API. This can make integration with third-party tools easier. The Proxy mode only changes the way you access ScrapingBee. The ScrapingBee API will then handle requests just like any standard request.

Request cost, return code and default parameters will be the same as a standard no-proxy request.

We recommend disabling Javascript rendering in proxy mode, which is enabled by default. The following credentials and configurations are used to access the proxy mode:

- HTTP address: `proxy.scrapingbee.com:8886`
- HTTPS address: `proxy.scrapingbee.com:8887`
- Socks5 address: `socks.scrapingbee.com:8888`
- Username: `3DVYVLNZBMJSVUJ83XZK1TVV8AANV2MMVQUSUZWPPE7RYI8OQKJPGF4X9FSL5YWW4FH39HSUUHKB8GFM`
- Password: `PARAMETERS`

Important : Replace `PARAMETERS` with our supported API parameters. If you don't know what to use, you can begin by using `render_js=False` .

As an alternative, you can use URLs like the following:

```
{
    "url_http": "http://3DVYVLNZBMJSVUJ83XZK1TVV8AANV2MMVQUSUZWPPE7RYI8OQKJPGF4X9FSL5YWW4FH39HSUUHKB8GFM:proxy.scrapin
    "url_https": "https://3DVYVLNZBMJSVUJ83XZK1TVV8AANV2MMVQUSUZWPPE7RYI8OQKJPGF4X9FSL5YWW4FH39HSUUHKB8GFM:proxy.scrap
    "url_socks5": "socks://3DVYVLNZBMJSVUJ83XZK1TVV8AANV2MMVQUSUZWPPE7RYI8OQKJPGF4X9FSL5YWW4FH39HSUUHKB8GFM:proxy.scr
}
```

Important: If you try to scrape Google with this mode, each requests will cost 20 credits.

To read more on how to integrate the proxy-mode with your favorite tools and language, check out the our dedicated-page.

Session

`session_id` (`integer`) (default= `no`)

All API requests using the same `session_id` will be routed through the same IP address for a duration of **5 minutes**.

We advice that you use a random integer between `0` and `10,000,000` everytime you wish to generate a new `session_id` .

cURL

```
curl "https://app.scrapingbee.com/api/v1/?api_key=3DVY
```

Timeout (in ms)

`timeout` (`int`) (default= `140 000`)

The maximum number of ms, comprise between `1000` and `140000` . ScrapingBee will wait before returning you a results, use `timeout=45000` in your API call to modify it.

Changing it could have a negative impact on your success rate.

Important: There will be a 0.5 second margin of error between the `timeout` used in your API call and the actual maximum duration of this API call.

cURL

```
curl "https://app.scrapingbee.com/api/v1/?api_key=3DVY
```

Custom Cookies

`cookies` (`string`) (default= `no`)

You can pass custom cookies to the target webpages

To do this you must pass a cookie string in the `cookies` parameter.

If you want to set multiple cookies, separate them with `;` .

ScrapingBee currently supports cookies with the following attributes:

- name (required)
- value (required)
- domain (optional)
- path (optional)
- expires (optional)

You need to separate each attribute with `;` and each cookie with `;` .

Cookie syntax is as follows:

```
name=value,other_attributes=other_attributes_value;
```

Example

```
cookie_name_1=cookie_value1,domain=scrapingbee.com;cookie_
```

Example with `cookie_name_1=cookie_value1;cookie_name_2=cookie_value_2` :

cURL

```
curl "https://app.scrapingbee.com/api/v1/?api_key=3DVY
```

In the above example, ScrapingBee is set to scrape `httpbin.org/cookies?json` , a page that displays the cookies it received.

The following will be returned by the above request.

```
{
  "cookies": {
    "cookie_name_1": "cookie_value1",
    "cookie_name_2": "cookie_value_2"
  }
}
```

Device

`device` (`string`) (default= `desktop`)

Choose the kind of device that will send the request to the server. Only two choices are available, `desktop` , the default, and `mobile` .

Set device desktop

cURL

```
curl "https://app.scrapingbee.com/api/v1/?api_key=3DVY
```

Set device mobile

cURL

```
curl "https://app.scrapingbee.com/api/v1/?api_key=3DVY
```

Google

`custom_google` (`boolean`) (default= `false`)

If you need to scrape webpage on Google main domain (google.com) or subdomains (news.google.com, scholar.google.com, etc ...) you'll need to use `custom_google=True` .

Important: each request using `custom_google=True` will cost you **20 credits.**

cURL

```
curl "https://app.scrapingbee.com/api/v1/?api_key=3DVY
```

POST / PUT

To send a POST / PUT request, send the POST / PUT request to the main endpoint with your `api_key` and `url` parameter.

Data will be forwarded transparently to the target web page.

Headers and cookies will also be returned.

Below is an example using `httpbin.org` , a service mirroring HTTP requests sent to the site.

cURL

```
curl -X "POST" "https://app.scrapingbee.com/api/v1/url
    -H 'Content-Type: application/x-www-form-urlencod
    --data-urlencode "KEY_1=VALUE_1"
```

Credit cost for your requests

Each ScrapingBee plan provides a certain amount of API credits per month.

It costs 1 to 75 credits to make requests to the ScrapingBee API. The credit cost depends on the parameters used with your API calls.

Here is a breakdown of ScrapingBee API credit costs:

Feature used	API credit cost
Rotating Proxy without JavaScript rendering	1
Rotating Proxy with JavaScript rendering (default)	5
Premium Proxy without JavaScript rendering	10
Premium Proxy with JavaScript rendering	25
Stealth Proxy without JavaScript rendering	(coming soon)
Stealth Proxy with JavaScript rendering	75

If you use AI features (`ai_query` `ai_extract_rules`), it will cost an additional 5 credits.

For example:

- If you make an API call with `render_js=True` and `ai_query="price of the product"` , it will cost you 5+5 = 10 API credits.
- If you make an API call with `render_js=True` , `premium_proxy=True` and `ai_query="price of the product"` , it will cost you 25+5 = 30 API

credits.

Usage endpoint

To programmatically monitor credit consumption and concurrency usage use `/usage` endpoint.

Calls to this endpoint will **not** increase concurrency, but you can only call it **6** times per minute.

Please note that the results are available in real-time.

cURL

```
curl "https://app.scrapingbee.com/api/v1/usage?api_key
```

Results.

```
{
    "max_api_credit": 20000000,
    "used_api_credit": 3704332,
    "max_concurrency": 200,
    "current_concurrency": 1,
    "renewal_subscription_date": "2022-04-18T10:05:58.
}
```

Status Code

Response Status Code

The following list of HTTP codes are returned by ScrapingBee.

Code	Billed?	Meaning	Solution
200	**Yes**	Successful Call	
400	No	Bad request	Incorrect parameters or parameters type. See the message in the response body.
401	No	No more credit available	Please upgrade your plan or contact sales.
404	**Yes**	Requested URL not found	Provide a valid URL.
410	**Yes**	Requested URL gone	Provide a valid URL.
413	No	File too large	Request a smaller file.
429	No	Too many concurrent requests.	Please upgrade your plan or contact sales.
500	No	Misc error	Please retry, and see the message in the response body.

Transparent HTTP status code

`transparent_status_code` (`boolean`) (default= `false`)

By default, ScrapingBee will return an HTTP 500 whenever the requested URL returns something other than: a 200-299 or a 404.

To get same body and status code as the requested URL in any circumstances, use the `transparent_status_code=true` parameter.

When this parameter is set to true:

- Every requests you make will be considered successful and will cost some amount of credits.
- ScrapingBee will not retry the request multiple time if the request returns a 500 error.

cURL

```
curl "https://app.scrapingbee.com/api/v1/?api_key=3EN7
```

Response Headers

The following is the list of additional HTTP headers returned by ScrapingBee.

Name	Meaning
Spb-cost	Request cost in credits.
Spb-initial-status-code	The initial status code returned by the scraped page. Useful when the page redirects.
Spb-resolved-url	The resolved URL of the scraped page. Useful when the page redirects.

Ready to get started?

Get access to 1,000 free API credits, no credit card required!

Try ScrapingBee for Free

Copyright © 2025

Made in France