

Metricool API

Metricool puts at your disposal an API with which you can interact to obtain all the information you need from your programs and scripts.

All data displayed on Metricool screens is accessible through our endpoints.

The base URL for all endpoints is: <https://app.metricool.com/api>

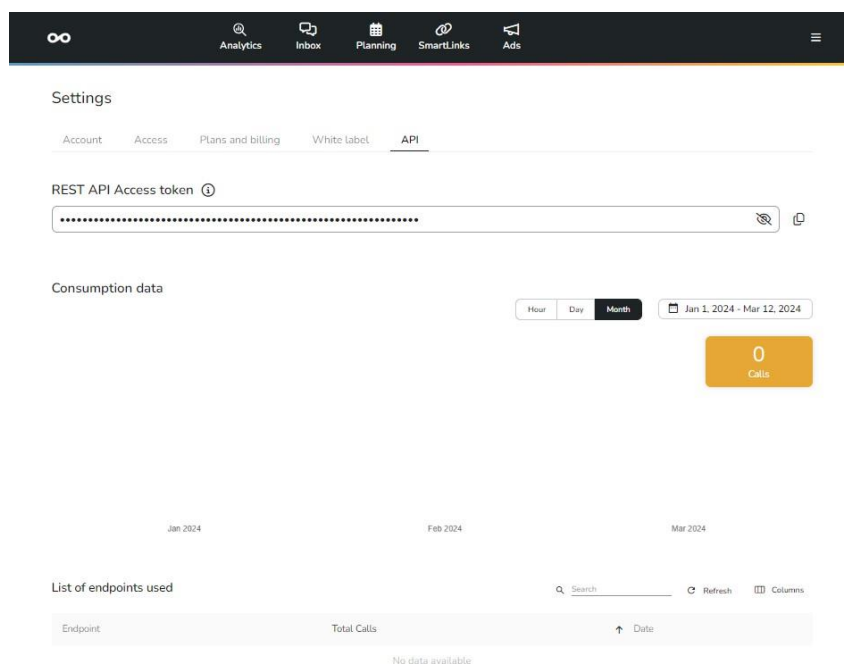
Important.

All endpoints require authentication. To identify yourself properly, you must include 3 mandatory parameters in all calls:

- **userToken:** unique authorization code for each user and can be found in the account settings.
- **userId:** the user identifier of your Metricool account.
- **blogId:** the identification number of the brand, this number can be easily found from the browser url.

You can add the parameters directly in the request (**the only parameter you have to implement in the header with name X-Mc-Auth is the *userToken***).

userToken can be found in the Account Settings menu, API section.



Example

Integration of the mandatory parameters in the request (this specific request allows you to find all the brands of a certain user):

<https://app.metricool.com/api/admin/simpleProfiles?blogId=1234567&userId=1234567>

In header you have to add as KEY X-Mc-Auth and in Value your userToken

Resources

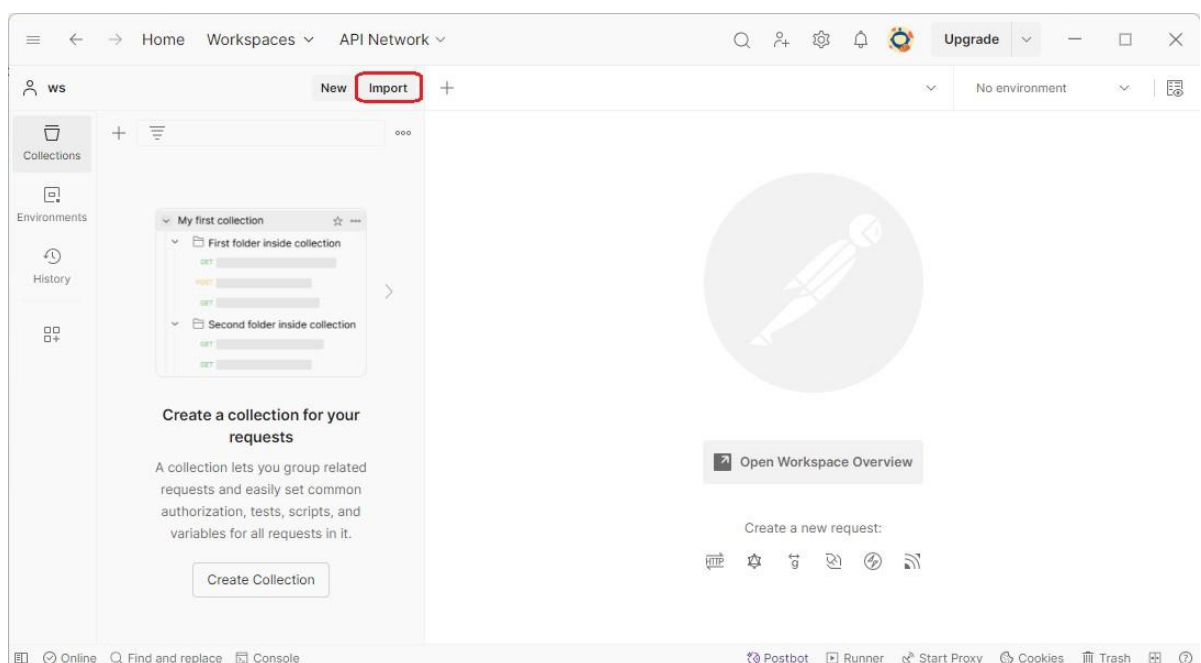
To review the description and configuration of our endpoints, you have the following files at your disposal: [swagger.json](#) or [swagger.yaml](#).

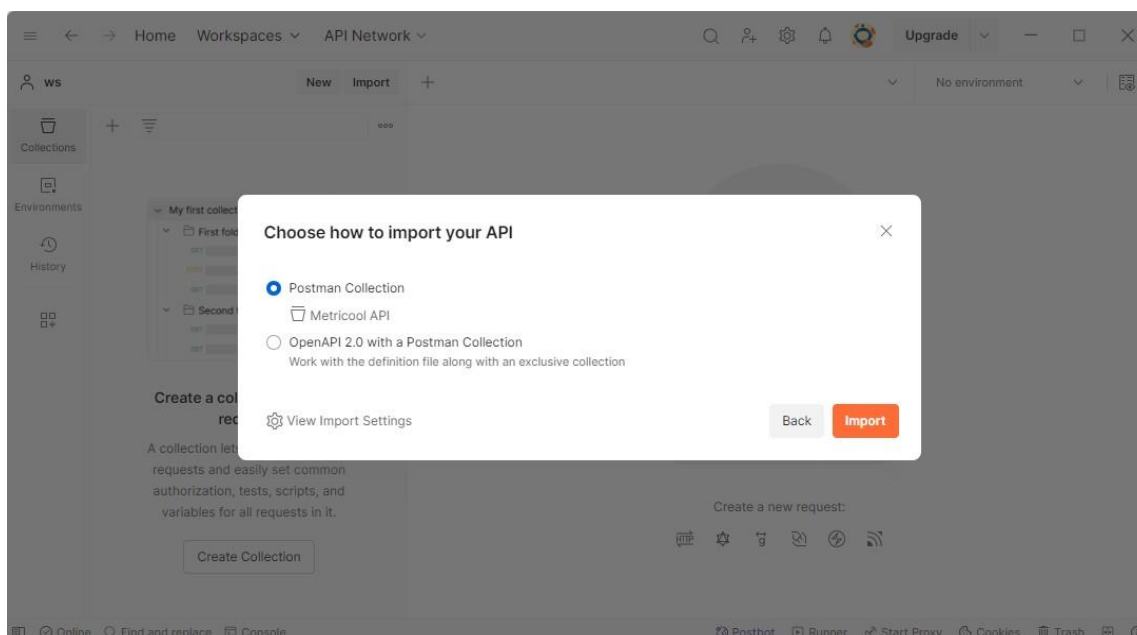
You can download any of them directly from these links. Here the files will always be updated to the latest version of our API. You should remember to download it again when you need to, to ensure you are seeing the latest version available.

Once downloaded, the file can be imported into any third-party API endpoints testing tool of your choice. For example, **Postman** or **SwaggerEditor**.

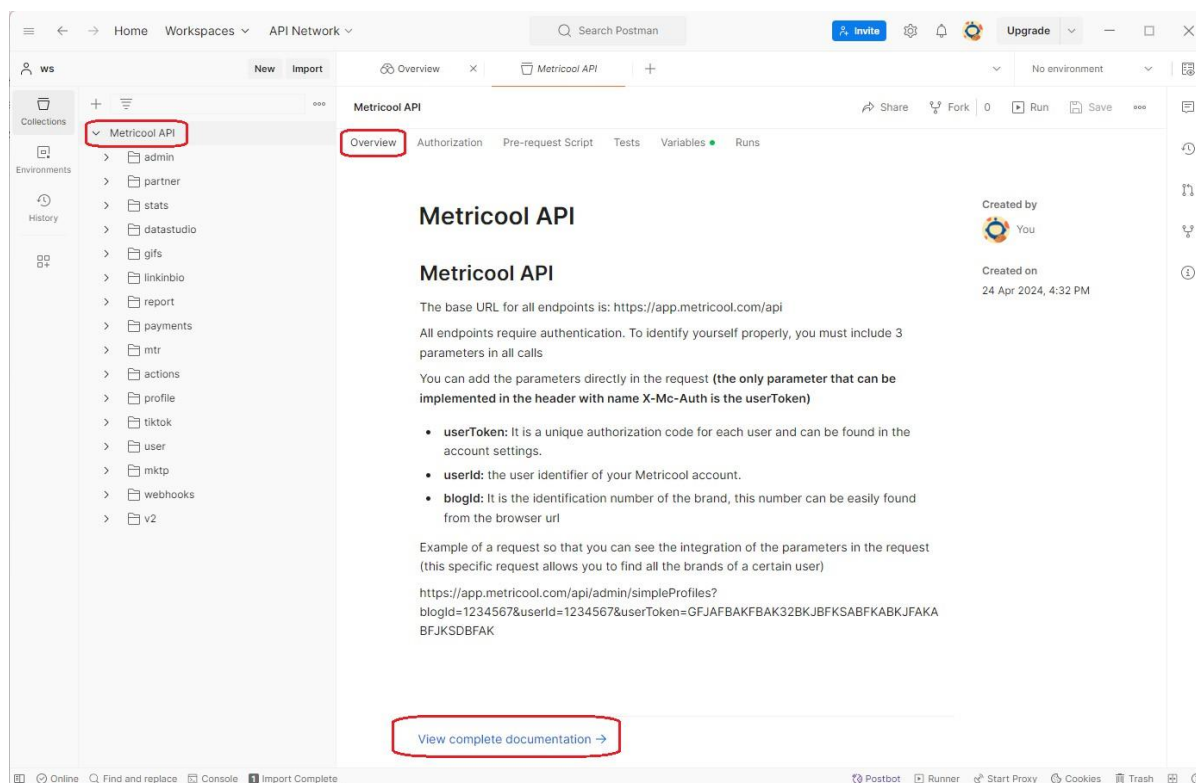
Postman

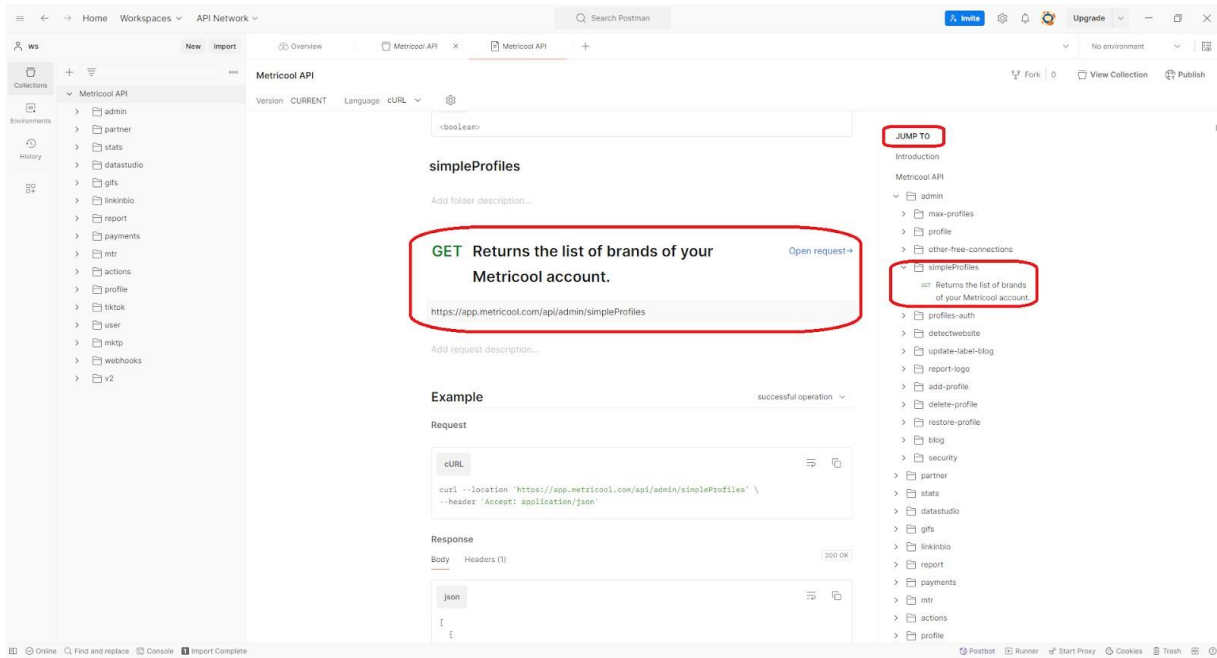
([desktop app](#)).



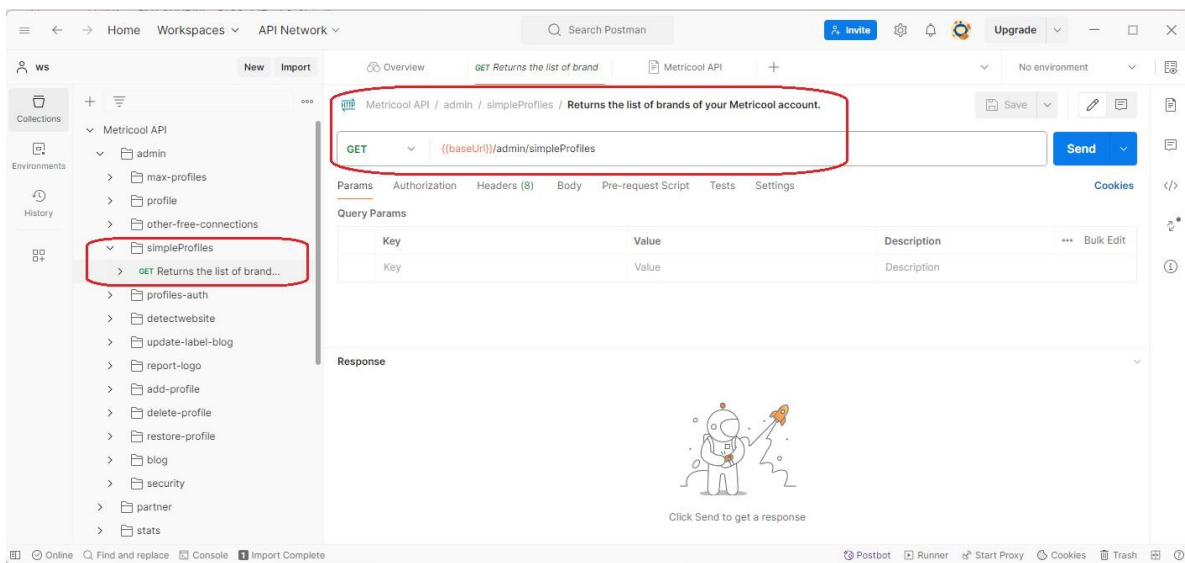


Once imported, from the **Overview** tab you can access the complete documentation.

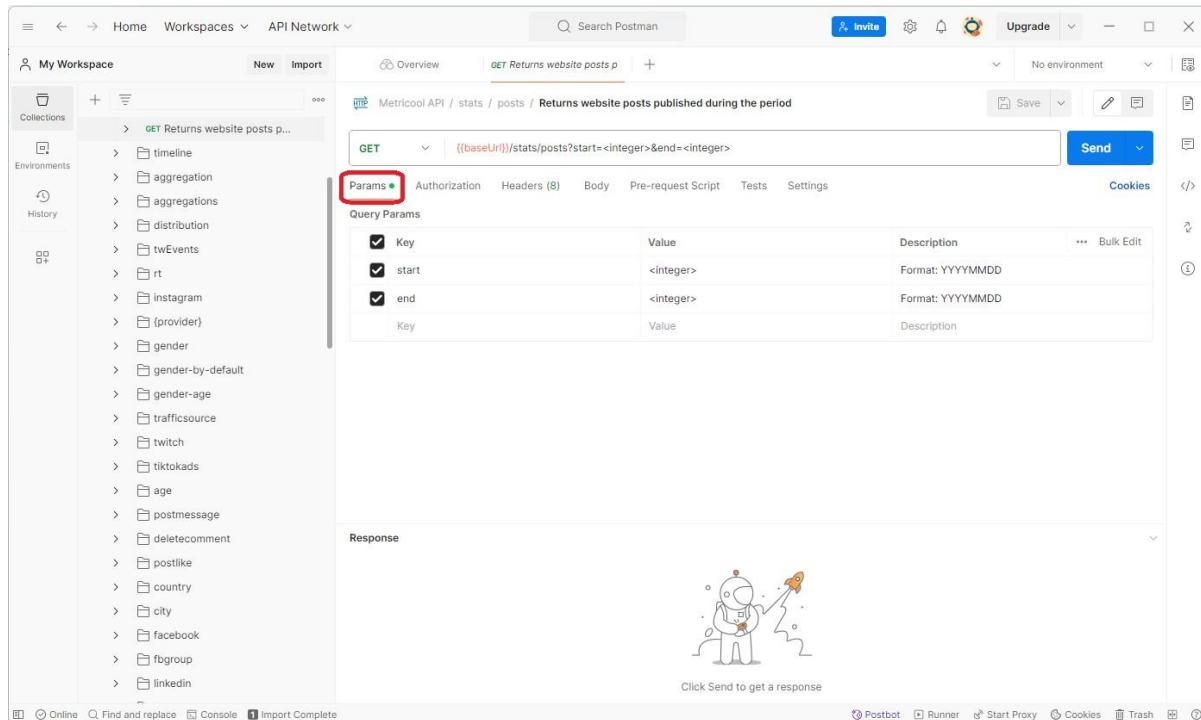




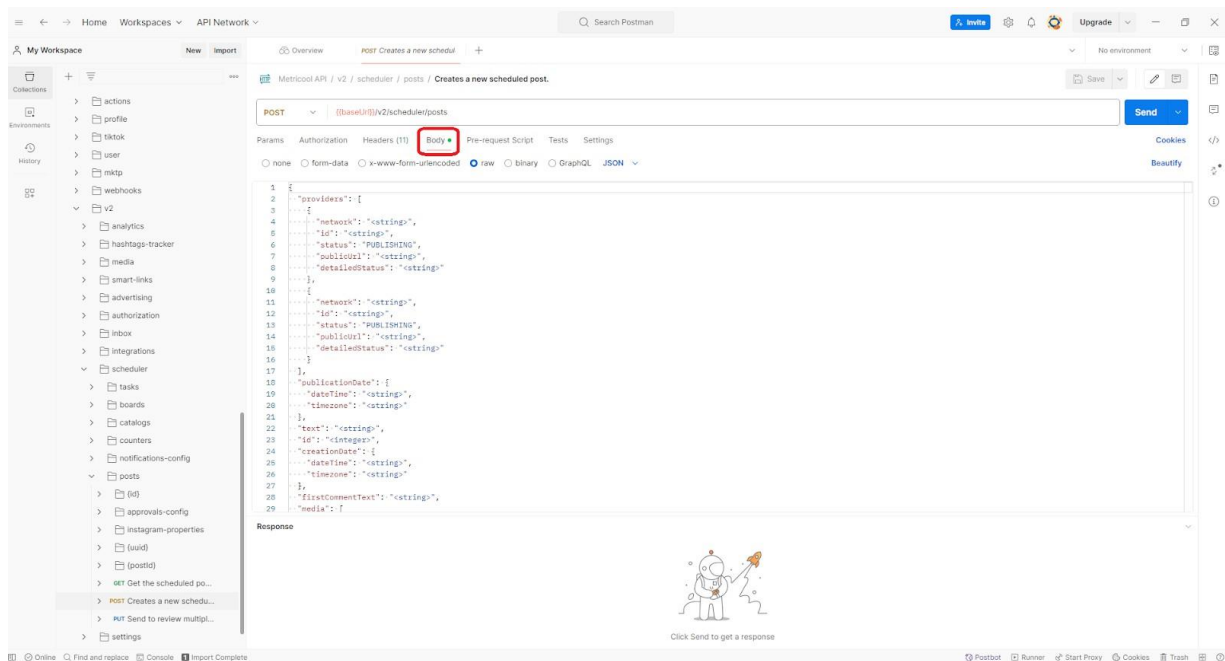
By navigating the left tree you can check all the endpoints configuration.



The **Params** tab will show you the request parameters, also known as query parameters or URL parameters.



In the **Body** tab you can find the request body parameters, the data sent along with the request in the body of the HTTP message. They carry the actual payload or data that you want to send to the server.

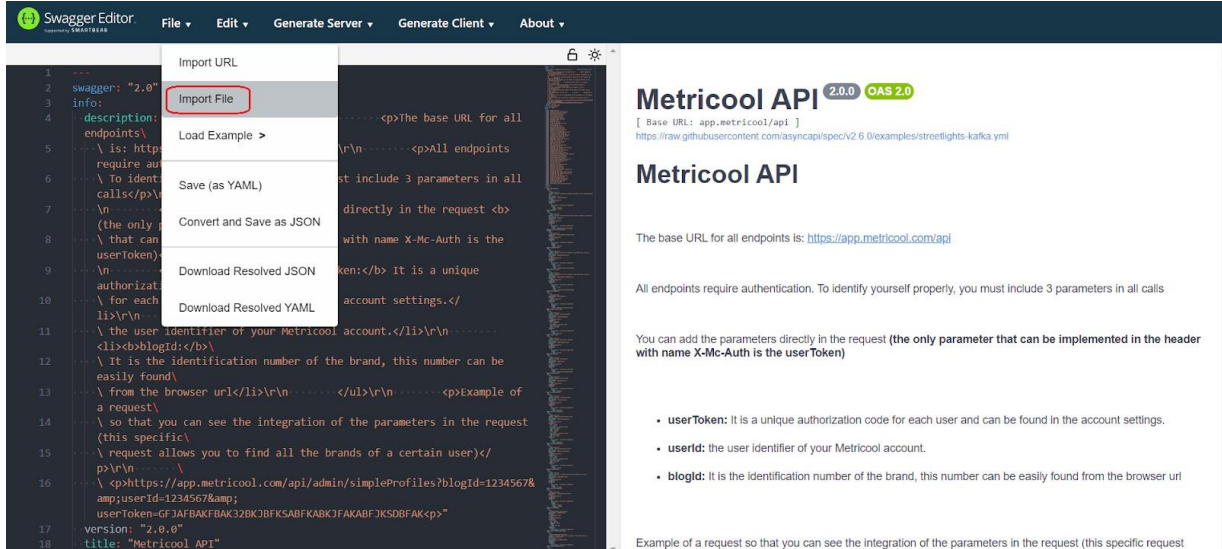


You can find more information about how to use Postman in this link:

→ [Postman Learning Center](#).

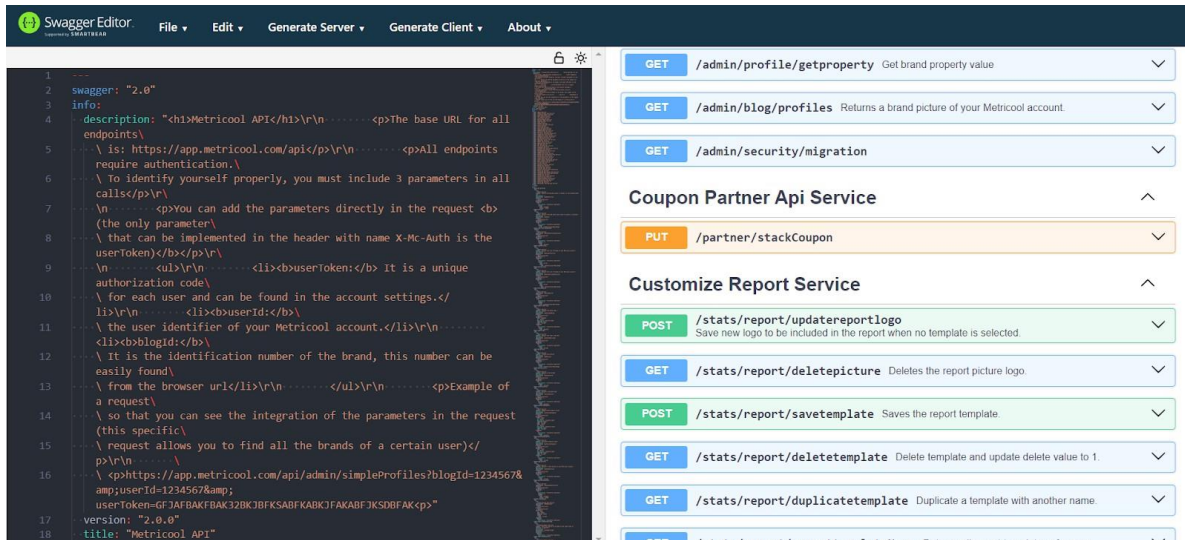
SwaggerEditor

([older web version](#) or [newer web version](#)).



The screenshot shows the Swagger Editor interface. On the left, a code editor displays a Swagger YAML file. A context menu is open over the file, with the 'Import File' option highlighted. The right panel shows the rendered API documentation for 'Metricool API' (version 2.0.0, OAS 2.0). The documentation includes the base URL, a list of endpoints, and a list of parameters: userToken, userId, and blogId. An example request is also shown.

Once imported, on the right window you can scroll down to find documentation and the list of all the endpoints and their configuration.



The screenshot shows the Swagger Editor interface with the 'Import File' option highlighted. The right panel displays the rendered API documentation for 'Metricool API' (version 2.0.0, OAS 2.0). The documentation includes the base URL, a list of endpoints, and a list of parameters: userToken, userId, and blogId. An example request is also shown.

By deploying each of them, you will be able to see its necessary parameters, the body and the content of the expected response.

The image displays two screenshots of the Swagger Editor interface, which is used for defining and documenting RESTful APIs. The interface is split into two main panels: a left panel for the OpenAPI specification (YAML/JSON) and a right panel for the interactive API explorer.

Top Screenshot: The left panel shows the OpenAPI specification for the `/stats/values/{category}` endpoint. The right panel displays the details for the `GET /stats/posts` endpoint, which returns website posts published during a specific period. The endpoint is defined with two query parameters: `start` (integer, YYYYMMDD format) and `end` (integer, YYYYMMDD format). The response is a 200 status code with a successful operation, returning an array of objects. The response content type is set to `application/json`.

Bottom Screenshot: The left panel shows the OpenAPI specification for the `/v2/scheduler/posts` endpoint. The right panel displays the details for the `POST /v2/scheduler/posts` endpoint, which creates a new scheduled post. The endpoint is defined with a body parameter (required) containing the configuration needed to schedule a post. The response is a 200 status code with a successful operation, returning an object. The response content type is set to `application/json`.

You can find more information about SwaggerEditor in this link:

→ [Swagger Editor](#)

Endpoints in dynamic mode from Metricool page

If you want to find an endpoint for a specific data that you are viewing on one of the Metricool screens, you can follow the next process.

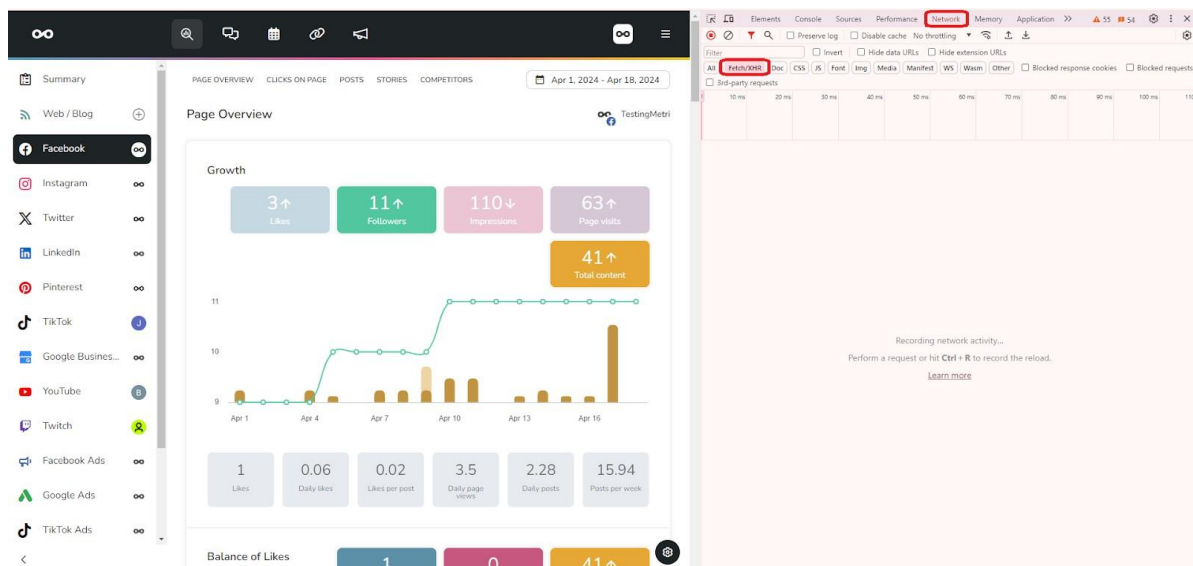
Browsing from your preferred web browser, once you are inside the Metricool platform, you just have to open the "web inspector", "debugging tool", "developers console" or whatever term that applies to the web browser you use.

For example, in *Google Chrome*, there are several ways to open up the tool:

- Three points upper right-side menu > More tools > Developers Tools.
- Ctrl + Shift + I or F12 (Windows)
- Option + ⌘ + I or fn+F12 (MacOS)
- Right click anywhere on the page > Inspect

This tool is complex and has many options, we are going to focus only on the important ones for this case.

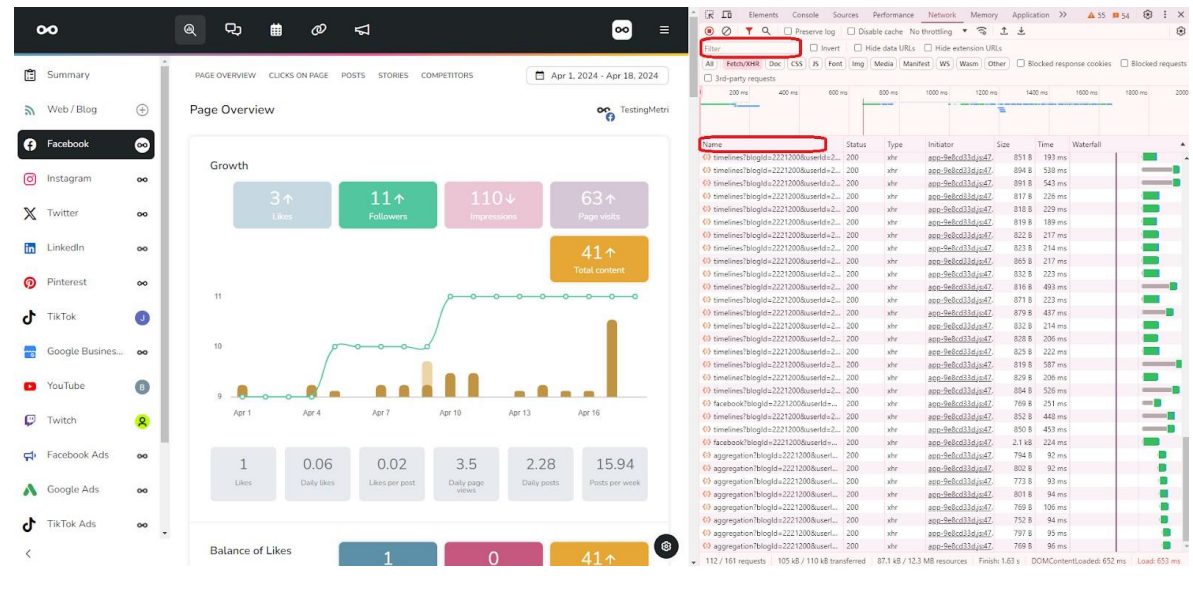
Go to *Network* > *Fetch/XHR* tab:



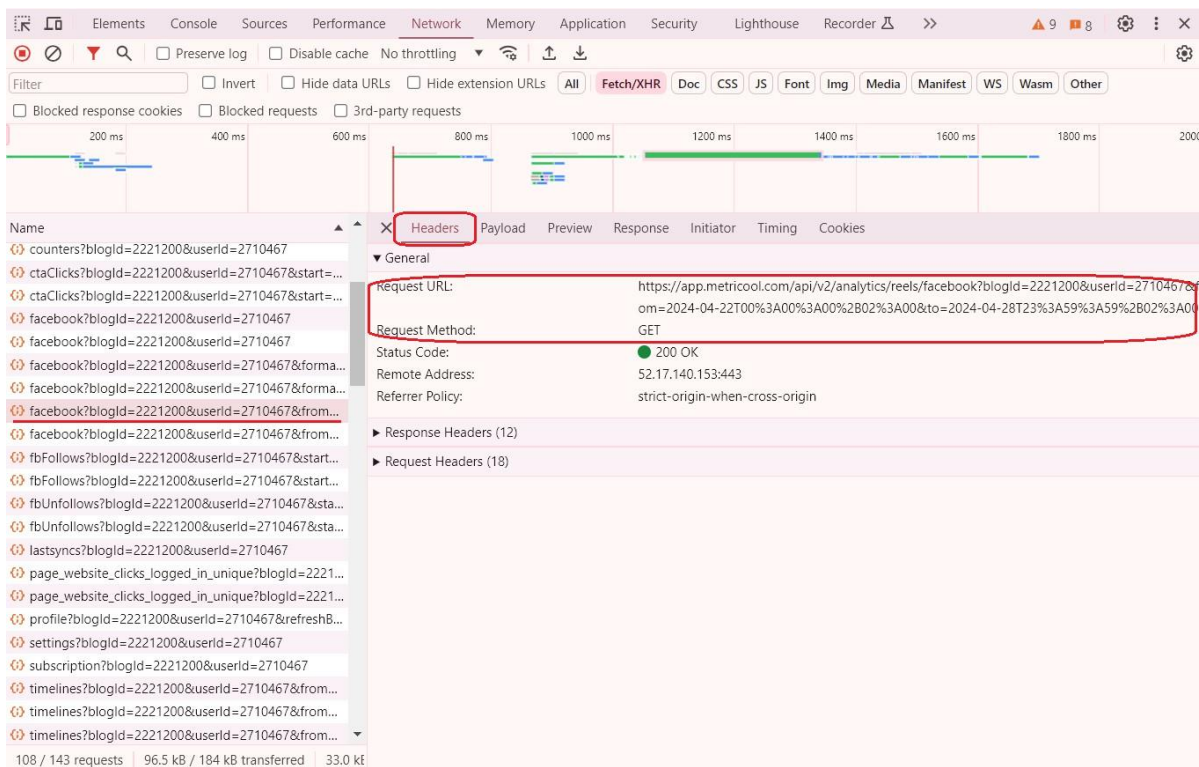
As you navigate through the Metricool platform, all the calls to endpoints that are made will appear on this screen.

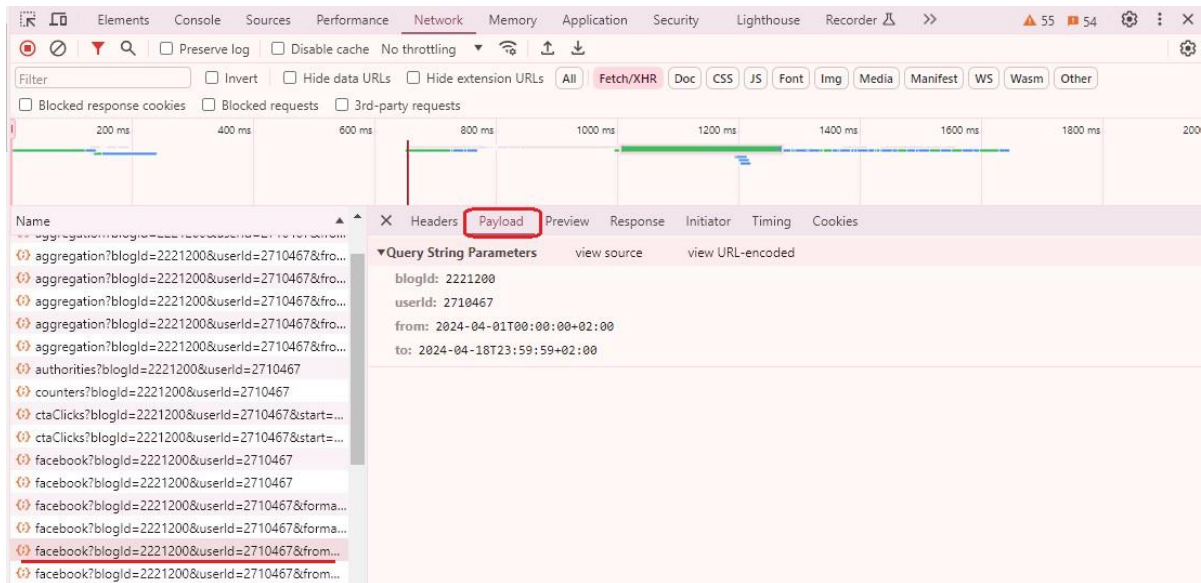
Note.

You can sort the endpoint calls by name by clicking on the 'name' column, or filter them by texting something in the appropriate field.

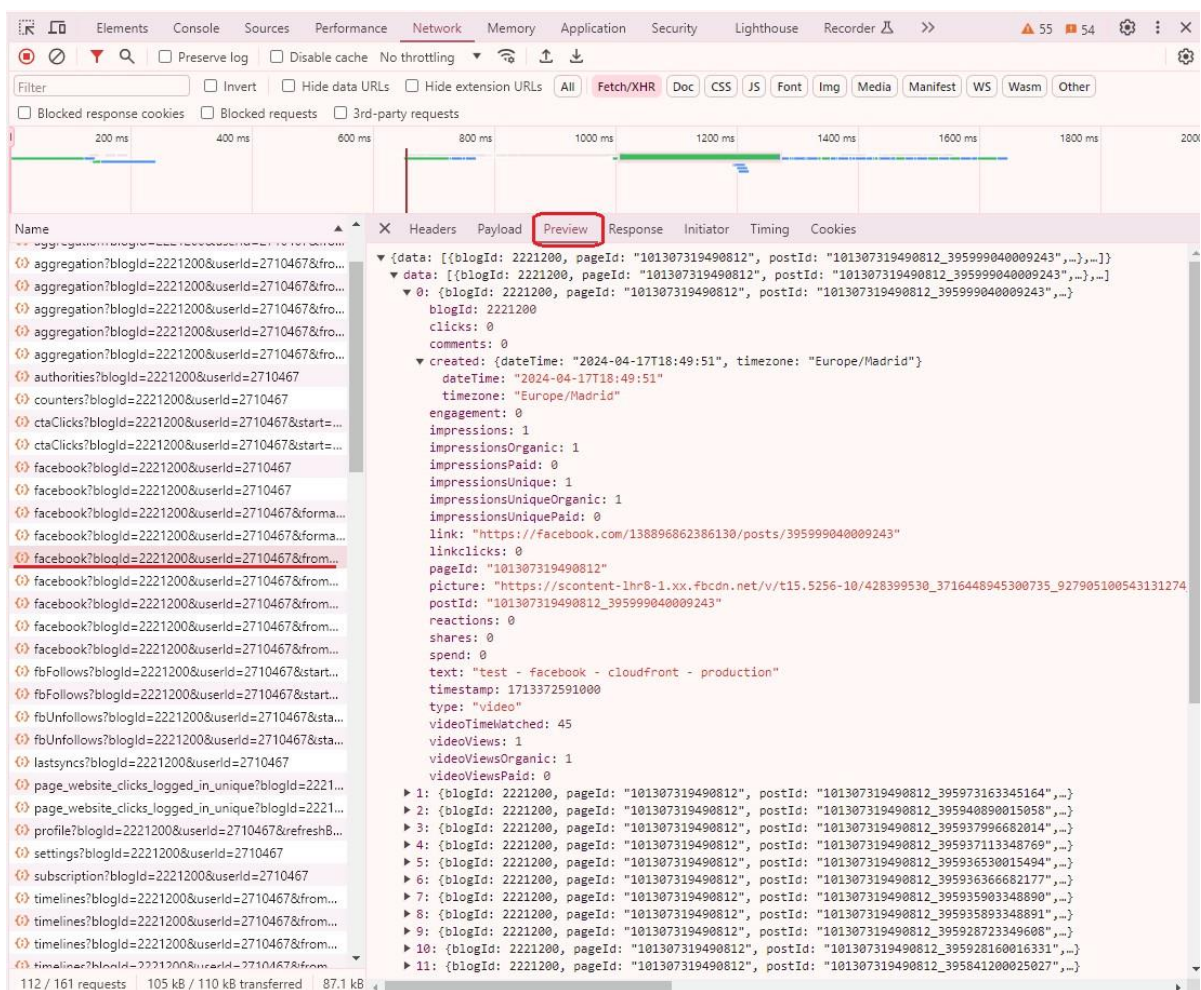


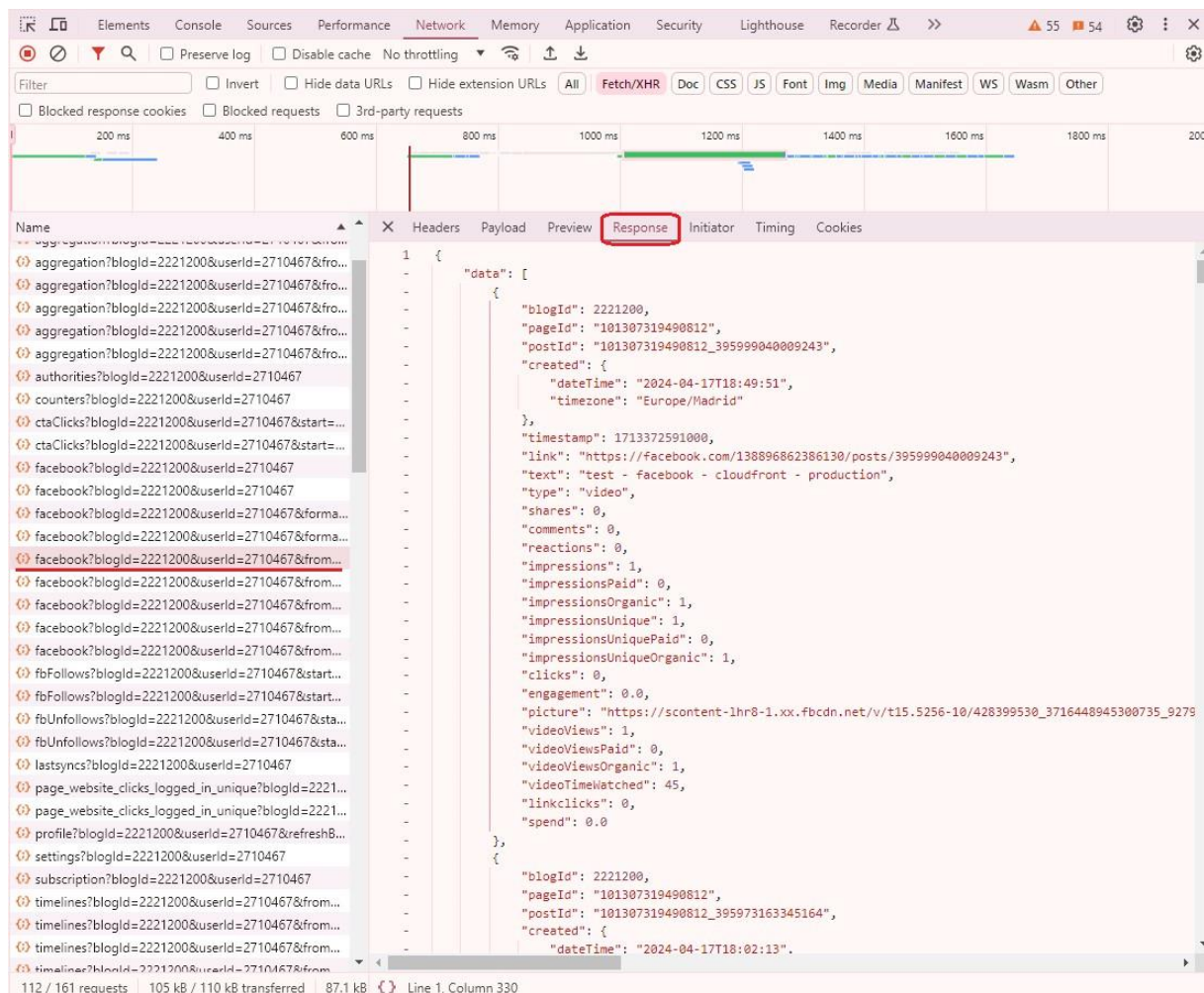
By clicking on each of the listed calls you will be able to see the details: the complete request url and method (*Headers* tab) and the parameters necessary for its configuration (*Payload* tab)





The response with the data delivered by the call to the endpoint can be found in the *Preview* or *Response* tabs.





Examples

Let's see some examples.

Note.

Remember that on each call you must include the authentication parameters (*userId*, *blogId*). In header you have to add as KEY X-Mc-Auth and in Value your userToken

Create a new scheduled post.

There are several endpoints to manage posts.

Posts Api Service

| | | |
|--------|--|---|
| GET | /v2/scheduler/posts/{id} | Get a scheduled post by id. |
| PUT | /v2/scheduler/posts/{id} | Update an existing scheduled post. |
| DELETE | /v2/scheduler/posts/{id} | Delete the selected post by id. If the post belongs to a thread and is the parent post, this post and all those belonging to the thread, will be deleted. |
| PATCH | /v2/scheduler/posts/{id} | Edit and update a scheduled post. If the post belongs to a thread and is the parent post, this post and all those belonging to the thread, will be updated. |
| GET | /v2/scheduler/posts | Get the scheduled posts between two dates. |
| POST | /v2/scheduler/posts | Creates a new scheduled post. |
| PUT | /v2/scheduler/posts | Send to review multiple posts of a brand in bulk. |
| PUT | /v2/scheduler/posts/{id}/approvals | Approve or reject a scheduled post by id and uuid. |
| GET | /v2/scheduler/posts/approvals-config | Get the brand configuration needed to send a post to review. |
| GET | /v2/scheduler/posts/instagram-properties | |

We are going to review in a little more detail the one used to create publications in the planner.

It's the **`/v2/scheduler/posts`** **POST** endpoint and you can find and copy the complete request body parameters in the `.yaml` or `.json` files. Required parameters are marked with a red *****.

POST `/v2/scheduler/posts` Creates a new scheduled post.

Parameters

body object (body)

Configuration needed to schedule a post.

```
{
  "publicationDate": {
    "dateTime": "string",
    "timezone": "string"
  },
  "creationDate": {
    "dateTime": "string",
    "timezone": "string"
  },
  "text": "string",
  "firstCommentText": "string",
  "providers": [
    {
      "network": "string",
      "id": "string",
      "status": "PUBLISHING",
      "publication": "string",
      "detailedStatus": "string"
    }
  ],
  "media": [
    "string"
  ],
  "autoPublish": true,
  "saveExternalMediaFiles": true,
  "mediaAltText": "string"
}
```

Parameter content type:

body object (body)

Configuration needed to schedule a post.

```
ScheduledPost {
  id > [...]
  publicationDate* > [...]
  creationDate > [...]
  text* string
  firstCommentText > [...]
  providers* > [ProviderStatus {
    network string
    id string
    status Enum: { PUBLISHED, PUBLISHING, PENDING, ERROR, DRAFT }
    publicUrl string
    detailedStatus string
  }]
  media > [...]
```

Parameters

| Name | Description |
|--------------------|--|
| body object (body) | Configuration needed to schedule a post. |

Example Value | Model

```
ScheduledPost {
  id > [...]
  publicationDate* > [...]
  creationDate > [...]
  text* > [...]
  firstCommentText > [...]
  providers* > [...]
  media > [...]
  autoPublish > [...]
  saveExternalMediaFiles > [...]
  mediaAltText > [...]
  shortener > [...]
  draft > [...]
  location > [...]
  videoCoverMilliseconds > [...]
  videoThumbnailUrl > [...]
  parentId > [...]
  twitterData > [...]
  facebookData > [...]
  smartLinkData > [...]
  instagramData > [...]
  pinterestData > [...]
  youtubeData > [...]
  tiktokData > [...]
  linkedinData > [...]
  autolistData > [...]
  brandName > [...]
  targetBrandId > [...]
  gmbData > [...]
  descendants > [...]
  notes > [...]
  hasNotReadNotes > [...]
  uuid > [...]
  creatorUserMail > [...]
  creatorUserId > [...]
  postApprovalData > [...]
}
```

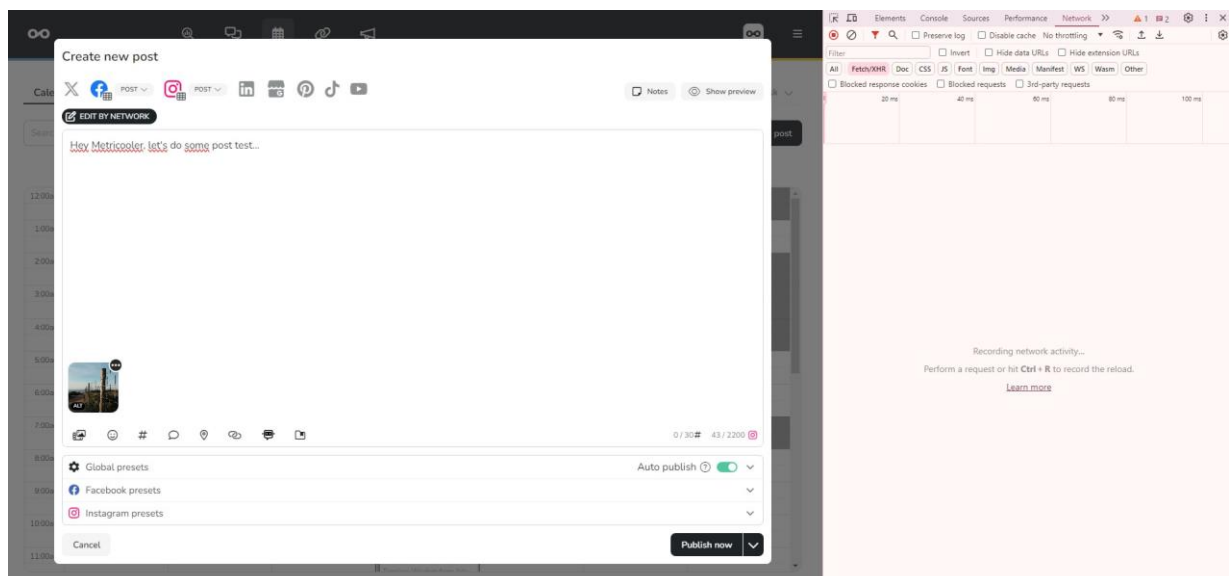
Here you have an example of a request body parameters:

```
{
  "publicationDate": {
    "dateTime": "2024-02-05T13:24:00",
    "timezone": "Europe/Madrid"
  },
  "creationDate": {
    "dateTime": "2023-11-24T16:22:00",
    "timezone": "Europe/Madrid"
  },
  "text": "Hello! This is a scheduled post test.",
  "firstCommentText": "",
  "providers": [
    {
      "network": "facebook"
```

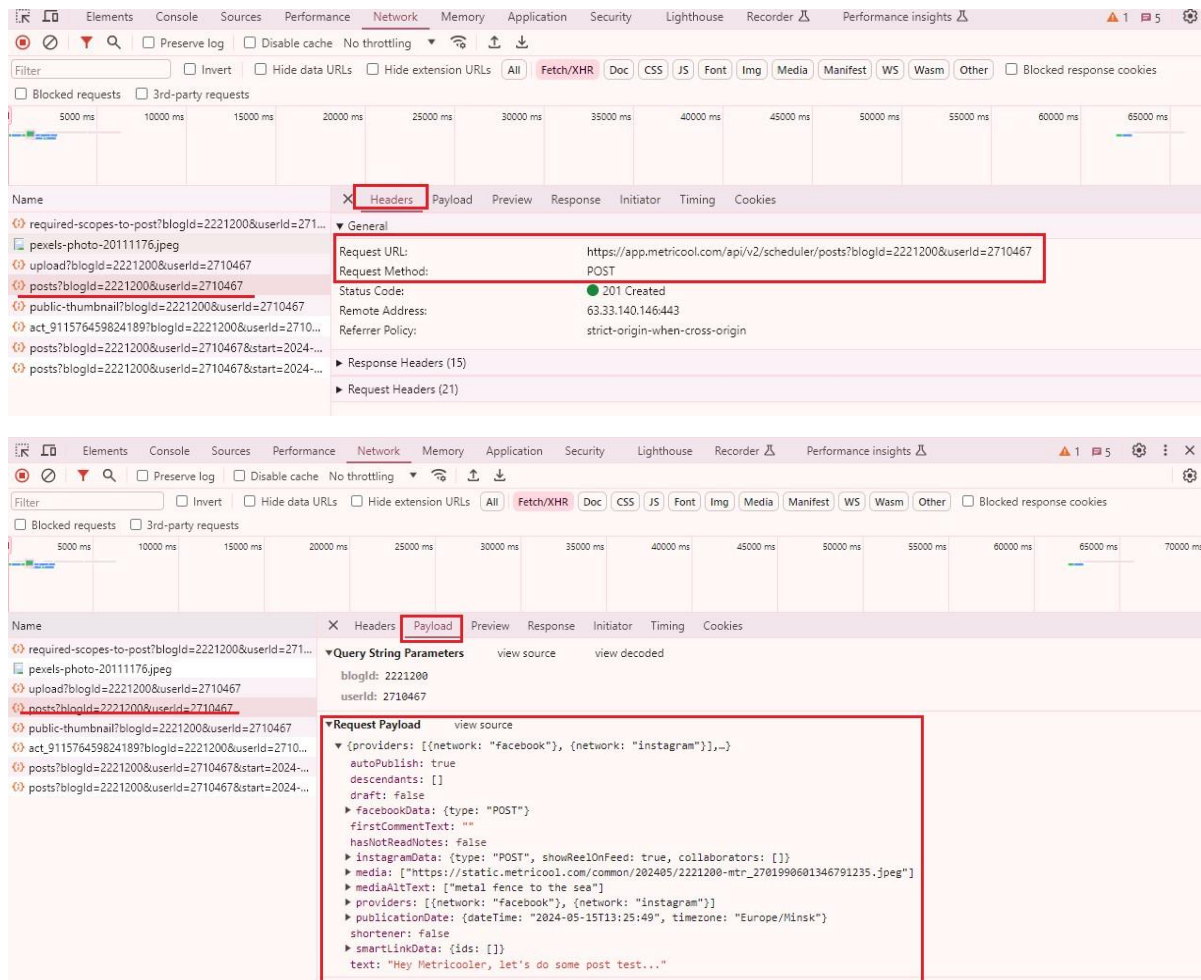
```
}  
],  
"autoPublish": true,  
"saveExternalMediaFiles": false,  
"shortener": false,  
"draft": false,  
"facebookData": {  
  "type": "POST"  
},  
"instagramData": {  
  "autoPublish": true  
},  
"hasNotReadNotes": false,  
"creatorUserMail": "me.myself@metricool.com",  
"creatorUserId": 1234567  
}  
}
```

Since the parameters of the request body are numerous and depend on the type of publication, social network and other series of circumstances, it is usually best to perform a test by making a publication, of the specific type you want, from the Metricool planner itself, and **through the browser's inspector**, see the specific call, with the necessary parameters, and then be able to program each desired type of specific publication via API.

The process will be as follows: with the web browser inspector open in our Metricool account, we go to Planning > Calendar and open the option to create a new publication. We fill in all the necessary data, in this example we are going to create a post for Facebook and Instagram, and then click on "publish now".



With this, we can now see the complete call that has been made and review the necessary parameters for the request body.



There is another particularity when you want to create a post that contains some type of media (video, photo...). An upload-preview must be done to ensure that we have access to the URL of the media that you want to publish, and if so, a copy of the image/video is created on our servers, so that we do not lose access. Afterwards, you can create the post, using the URL of the element that we have saved on our servers.

Note.

You have to make sure that the original media URL you are using is from a public site and/or that you have all the permissions to access it (for example, in the case of Drive, have the "anyone with the link can see it" option enabled).

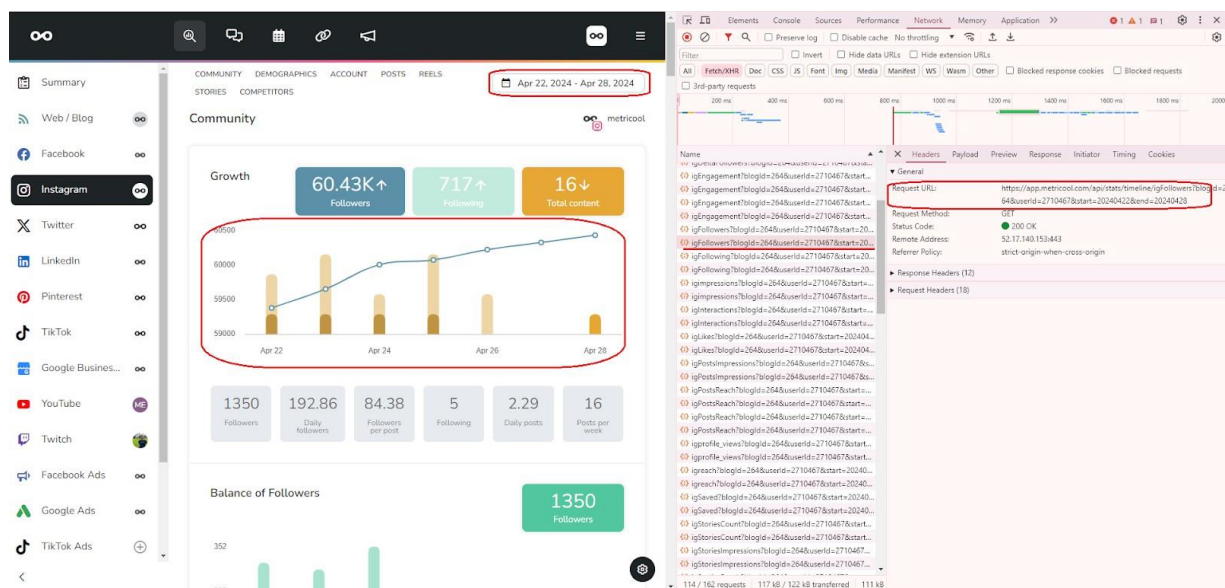
So, before making the call to **`https://app.metricool.com/api/v2/scheduler/posts?...`** , you would have to make a call to **`https://app.metricool.com/api/actions/normalize/image/url?url=<url of your media>...`** that will return the URL of the copy of your image/video on our servers, and that URL will be the one you have to put in the **`media`** parameter of the call to `v2/scheduler/posts`.

Note.

By following the same procedure of "execute the action in your Metricool account" and reviewing the calls made in the web browser inspector, you can easily obtain all the usage information for all available endpoints.

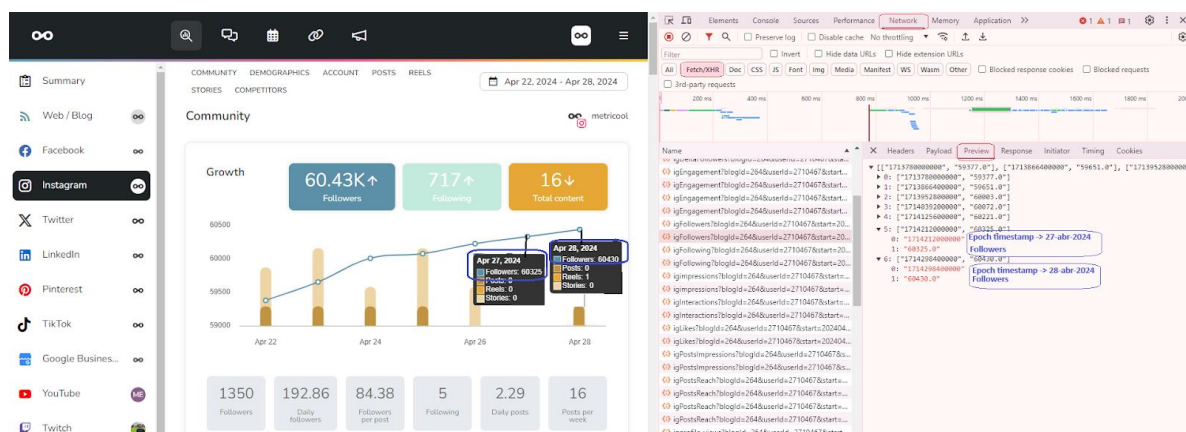
Recover Instagram followers timeline.

Let's try to get the timeline series for the Instagram followers count. Once the date range is settled, we have to look for the proper endpoint in the 'network' tab.



In this case, the endpoint

<https://app.metricool.com/api/stats/timeline/igFollowers?start=<...>&end=<...>> will give us the data.



/stats/timelining{metric} supports a great variety of possible metrics, you can check all of them in the `.yaml` or `.json` files:

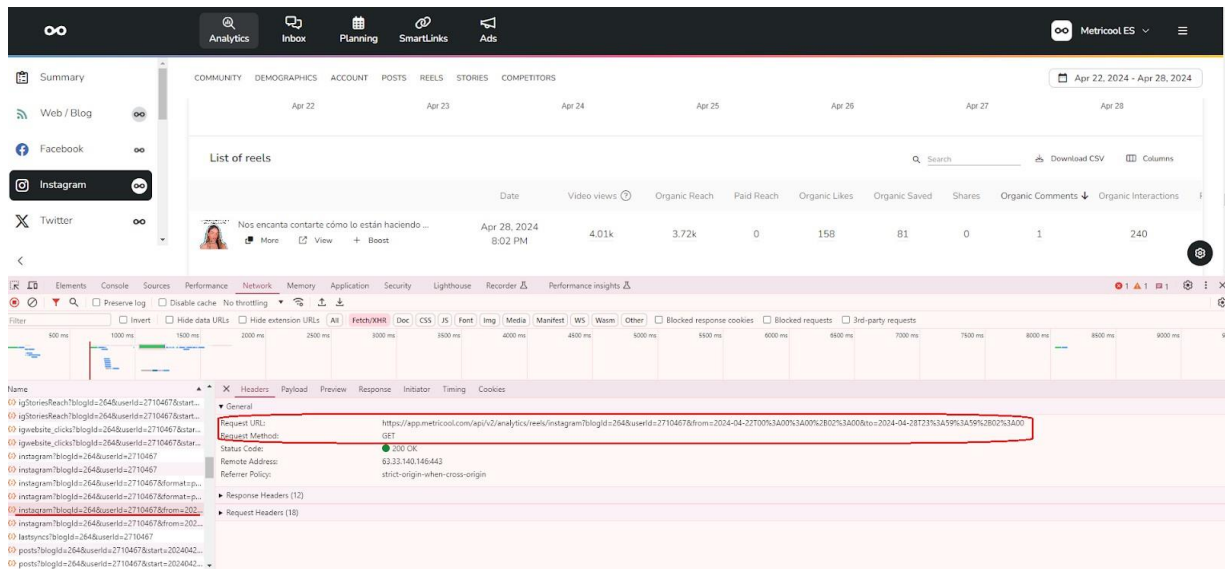
GET /stats/timeline/{metric}

Returns time serie values for a concrete metric during a period of time

Parameters

| Name | Description |
|--|---|
| metric ★ required string <i>(path)</i> | Supported values: For Website: visitors, PageViews, SessionsCount For Blog: DailyComments, DailyPosts For Twitter: twitterFollowers, twTweets, 2ndLevelFollowers, LastDayActiveFollowers, LastMonthActiveFollowers, twFavorites, twFriends, twListed, twMentions, twRetweets, follows, unfollows For Facebook pages: facebookLikes, fbPosts, dailyImpressions, dailyImpressionsUnique, dailyReactions, dailyClicks, dailyShares, dailyComments, pageImpressions, pageViews, fbFollows, fbUnfollows, ctaClicks, callPhoneClicks, getDirectionsClicks, For Facebook groups: fb_group_members, fb_group_members_req, fbGroupsPosts, fbGroupsEngagement, fbGroupsInteractions, fbGroupsReactions, fbGroupsComments For Instagram: igFollowers, igFollowing, igDeltaFollowers, igPosts, igLikes, igComments, igEngagement, igSaved, igInteractions, igimpressions, igreach, igprofile_views, igwebsite_clicks, igPostsImpressions, igPostsReach, igStoriesImpressions, igStoriesReach, igStoriesCount For LinkedIn: inFollowers, inPaidFollowers, inCompanyImpressions, inPosts, inCliks, inPostsLikes, inComments For Facebook Ads: clicks, total_action_value, cpc, cpm, cpp, ctr, impressions, reach, spend, unique_clicks, unique_ctr For Youtube: ytviewerPercentage, yttotalComments, yttotalViews, yttotalVideos, yttotalSubscribers, yttotalLikes, yttotalDislikes, yttotalFavorites, ytsubscribersGained, ytsubscribersLost, ytvideos, ytestimatedRevenue, |

Check Instagram reels analytics.



Here we use the endpoint

<https://app.metricool.com/api/v2/analytics/reels/instagram?from=<...>&to=<...>>

The screenshot displays the Metricool dashboard interface and the browser's developer tools. The dashboard shows a summary of social media performance for Instagram, with a table of reels. The browser's developer tools show the Network tab with a list of requests, and the Preview tab showing the JSON response for a specific Instagram reel.

Summary

COMMUNITY DEMOGRAPHICS ACCOUNT POSTS REELS STORIES COMPETITORS

Apr 22 Apr 23 Apr 24 Apr 25 Apr 26 Apr 27 Apr 28

List of reels

| Date | Video views | Organic Reach | Paid Reach | Organic Likes | Organic Saved | Shares | Organic Comments | Organic Interactions | Paid Interactions | Organic Engagement | Paid Post Clicks |
|----------------------|-------------|---------------|------------|---------------|---------------|--------|------------------|----------------------|-------------------|--------------------|------------------|
| Apr 28, 2024 8:02 PM | 4.01k | 3.72k | 0 | 158 | 81 | 0 | 1 | 240 | 0 | 6.45 | 0 |

Network

Filter: 500 ms 1000 ms 1500 ms 2000 ms 2500 ms 3000 ms 3500 ms 4000 ms 4500 ms 5000 ms 5500 ms 6000 ms 6500 ms 7000 ms 7500 ms 8000 ms 8500 ms 9000 ms 950 ms

Headers

Payload

Preview

```
{data: [{"reelId": "3356343132478323806_1816304772", "userId": "Metricool ES m", "businessId": "17992073405629737", "comments": 1, "content": "Nos encanta contarte c\u00f3mo lo est\u00e1n haciendo marcas para tener \u00e9xito en redes sociales. Hoy hablamos de @arenarobjabrand\n\nLa CEO de #arenaroja siempre sale en sus engagement: 6.451612903225806", "FILTER": "imageId=1: \"https://scontent-lhr6-2.cdninstagram.com/v/t51.29350-15/440108245_463296926119845_8935824946908414930_n.jpg?_nc_cat=100&ccb=1-7&_nc_sid=18de74&_nc_ohc=s1ow5zeW60s impressions: 0", "impressionsTotal": 0, "likes": 158, "reach": 3720, "reelId": "3356343132478323806_1816304772", "saved": 81, "shares": 0, "type": "REELS_VIDEO", "url": "https://www.instagram.com/reel/C6UIEvbMype/", "userId": "Metricool ES m", "videoViews": 4007, "videoViewsTotal": 4007}]}
```