MapOSMatic remote rendering API

Hartmut Holzgraefe

Version 1, September 16th 2018 (Work in progress)

Table of Contents

MapOSMatic is a web frontend for the OCitySMap renderer, which uses the Mapnik map rendering libary, Python and the CairoGraphics library to create decorated single page maps and multi page atlas-like booklets.

Originally this was a pure user centric web GUI, but recently a wish for using the infrastructure with different web frontends did come up.

So this HTTP API provides an alternative way to submit map rendering requests, and to retrieve additional data that may be needed on the way.

Requests and results are submitted as JSON documents, with the optional possibility to attach extra data files, like GPX tracks or Umap user created maps.

This documentation begins with describing the helper calls that provide lists of possible choices for different data fields, followed by the actual map rendering call and how to retrieve rendering status and results, and ends with example programs to access the API from different programming languages.

API calls

Paper Formats

Request URL

```
https://api.get-map.org/apis/paper_formats
```

Example result

```
{
    "Best fit": {
        "height": null,
        "width": null
    },
    "Din A4": {
        "height": 297,
        "width": 210
    },
    "US letter": {
        "height": 279,
        "width": 216
    }
}
```

Layouts

Request URL

```
https://api.get-map.org/apis/layouts
```

Example result

```
{
    "multi_page": {
        "description": "A multi-page layout.",
        "preview_url": "https://api.get-map.org/media/img/layout/multi_page.png"
    },
    "plain": {
        "description": "Full-page layout without index.",
        "preview_url": "https://api.get-map.org/media/img/layout/plain.png"
    },
    "single_page_index_bottom": {
        "description": "Full-page layout with the index at the bottom.",
        "preview_url": "https://api.get-
map.org/media/img/layout/single_page_index_bottom.png"
    },
    "single_page_index_side": {
        "description": "Full-page layout with the index on the side.",
        "preview_url": "https://api.get-
map.org/media/img/layout/single_page_index_side.png"
    }
}
```

Base Styles

Request URL

```
https://api.get-map.org/apis/styles
```

Example output

```
{
    "CartoOSM": {
        "annotation": "OpenStreetMap Carto standard style",
        "description": "CartoCSS OSM standard style",
        "preview_url": "https://api.get-map.org/media/img/style/CartoOSM.png"
    },
    "GermanCartoOSM": {
        "annotation": "German OSM style based on OSM Carto",
        "description": "German OSM style",
        "preview_url": "https://api.get-map.org/media/img/style/GermanCartoOSM.png"
    }
}
```


Overlay Styles

Request URL

```
https://api.get-map.org/apis/overlays
```

Example output


```
{
    "OpenRailwayMap_Overlay": {
        "annotation": "OpenRailwayMap overlay",
        "description": "OpenRailwayMap rail line overlay",
        "preview_url": "https://api.get-
map.org/media/img/style/OpenRailwayMap_Overlay.jpg"
    },
    "Scale_Bar_overlay": {
        "annotation": "",
        "description": "Map scale bar"
        "preview_url": "https://api.get-map.org/media/img/style/Scale_Bar_overlay.jpg"
    }
 }
```


Job Stati

Request URL

```
https://api.get-map.org/apis/job_stati/
```

Example result

```
{
    "0": "Submitted",
    "1": "In progress",
    "2": "Done",
    "3": "Done w/o files",
    "4": "Cancelled"
}
```

Submitting a Job

Request URL

```
POST https://api.get-map.org/apis/jobs/
```

Expects render job named parameters as a JSON collection.

Possible parameters:

osmid
: ID of an object in the osm2pgsql

bbox_top
bbox_bottom
bbox_left
bbox_right
: Render area min. bounding box, may be extended to fit paper format

title
: Text to put into the title box (default: "Untitled API Request")

language
: locale to use for rendering (default: "en_US.UTF-8")

layout
: Map layout to use (default: "plain")

style
: Map base style to use (default: "CartoOSM")

overlays
: List of overlays to put on top of base style (default: None)

paper_size

Paper size to use (default: "Din A4" 210x297mm²)

orientation

Paper orientation: "portrait" or "landscape" (default: "portrait")

track_url

URL pointing to a GPX track to download and print

umap_url

URL pointing to an Umap export file do download and print

It is also possible to upload GPX and Umap files directly using a multipart/form-data POST request. In this case you have to submit the actual API JSON as a post parameter named "json", and add file fields named "track" for a GPX track, or "umap" for an Umap JSON export.

After successfully submitting a request you can either poll the job result API every once in a while until it either returns the "200 OK" status, or a 4xx error code. Or you can redirect a web client to the human readable result page listed in the returned "interactive" job parameter.

Return codes

202

Accepted when successfully submitted, "Location:" header will contain result URL

400

Bad request on request validation errors

Example requests

```
Content-type: application/json

{
    "title": "API Test",
    "bbox_bottom": 52.00,
    "bbox_left": 8.52,
    "bbox_right": 8.50,
    "bbox_top": 52.02
}
```

```
Content-Type: multipart/form-data; boundary=--XXXXXXXX

--XXXXXXXX
Content-Disposition: form-data; name="json"

{
    "title": "GPX Test",
    "paper_size": "Din A1"
}
--XXXXXXXX
Content-Disposition: form-data; name="track"; filename="test.gpx"
Content-Type: application/gpx+xml

<?xml version="1.0" encoding="UTF-8"?>

<gpx version="1.1">
  <metadata>
    <name>GPX test</name>
  </metadata>
  <trk>
    <trkseg>
      <trkpt lat="52.00" lon="8.50" />
      <trkpt lat="52.02" lon="8.52" />
    </trkseg>
  </trk>
</gpx>
--XXXXXXXX--
```

Example results

```
HTTP/1.1 202 Accepted
[...]
Location: /api/jobs/36
Content-Type: text/json

{
    "bbox_bottom": 52.01,
    "bbox_left": 8.52,
    "bbox_right": 8.50,
    "bbox_top": 52.02,
    "id": 36,
    "interactive": "https://api.get-map.org/maps/214"
    "language": "en_US.UTF-8",
    "layout": "plain",
    "paper_height_mm": 297,
    "paper_width_mm": 210,
    "status": 0,
    "styles": "CartoOSM"
    "title": "API Test",
}
```

```
HTTP/1.1 400 Bad Request
[...]
Content-Type: text/json

{
    "error": {
        "non_field_errors": [
            "No bounding box area or OsmID given"
        ]
    }
}
```

Checking Job results

Request Url

```
https://api.get-map.org/apis/jobs/
```

Return codes

200

OK when request has been processed successfully

202

Accepted when rendering is still in progress, "Location:" header will contain same URL again

410

Gone when a request had been processed successfuly in the past, but result files have since been purged to free file system space

400

Bad request when a request failed to render

Example result

```
HTTP/1.1 200 OK
[...]
Content-type: application/json

{
    "bbox_bottom": 52.01,
    "bbox_left": 8.53,
    "bbox_right": 8.49,
    "bbox_top": 52.02,
    "files":
    {
        "8bit.png": "https://api.get-map.org/results//000215_2018-09-12_23-14_GPX-
test.8bit.png",
        "jpg": "https://api.get-map.org/results//000215_2018-09-12_23-14_GPX-
test.jpg",
        "pdf": "https://api.get-map.org/results//000215_2018-09-12_23-14_GPX-
test.pdf",
        "png": "https://api.get-map.org/results//000215_2018-09-12_23-14_GPX-
test.png",
        "svgz": "https://api.get-map.org/results//000215_2018-09-12_23-14_GPX-
test.svgz"
    }
    "id": 215,
    "language": "en_US.UTF-8",
    "layout": "plain",
    "paper_height_mm": 594,
    "paper_width_mm": 841,
    "queue_size": 0,
    "status": 2,
    "style": "CartoOSM",
    "title": "GPX test"
}
```

Example code

PHP

All examples in this script use the `PEAR:HTTP_Request2` class for sending HTTP requests and receiving server replies.

Retrieve parameter choices

Here we are using the `/paper_formats`, `/layouts`, `styles` and `overlays` API calls to retrieve possible choices for all of these parameters, and their textual description where available.

parameters.php

```
<?php
require_once 'HTTP/Request2.php';

define('BASE_URL', 'https://api.get-map.org/apis/v1/');

function api_call($url)
{
  $request = new HTTP_Request2(BASE_URL . $url);

  $request->setMethod(HTTP_Request2::METHOD_GET);

  $response = $request->send();
  $status = $response->getStatus();

  if ($status < 200 || $status > 299) {
      echo "invalid HTTP response to '$url': $status\n";
      echo $response->getBody();
      exit(3);
  }

  return json_decode($response->getBody());
}

echo "PAPER FORMATS:\n";
$paper_formats = api_call('paper_formats');
foreach ($paper_formats AS $name => $format)
  printf("%-10s: %4dx%4d mm²\n", $name, $format->width, $format->height);

echo "\nLAYOUTS\n";
$layouts = api_call('layouts');
foreach ($layouts as $name => $layout)
  printf("%-25s: %s\n", $name, $layout->description);

echo "\nSTYLES\n";
$styles = api_call('styles');
foreach ($styles as $name => $style)
  printf("%-25s: %s\n", $name, $style->description);

echo "\nOVERLAYS\n";
$overlays = api_call('overlays');
foreach ($overlays as $name => $overlay)
  printf("%-25s: %s\n", $name, $overlay->description);
```

Submitting a simple request

Now we are submitting a simple request, only setting the map title, and the bounding box of the area to display. We do not check for rendering to complete, but redirect the web client to the interactive result URL given as `interactive` in the returned JSON data.

This is the same result page as when you submit a map request manually via the standard web user interface, so this approach is suitable for interactive web applications:

simple-request.php

```
<?php
require_once 'HTTP/Request2.php';

define('BASE_URL', 'https://api.get-map.org/apis/v1/');

$data = ['title'       => 'PHP Test',
         'bbox_bottom' => 52.00,
         'bbox_top'    => 52.02,
         'bbox_left'   => 8.50,
         'bbox_right'  => 8.52
        ];

$request = new HTTP_Request2(BASE_URL . "jobs");

$request->setMethod(HTTP_Request2::METHOD_POST)
        ->setHeader('Content-type: application/json; charset=utf-8')
        ->setBody(json_encode($data));

$response = $request->send();
$status   = $response->getStatus();

if ($status < 200 || $status > 299) {
    header("Content-type: text/plain");
    print_r($response->getBody());
    exit;
}

$reply = json_decode($response->getBody());

header("Location: " . $reply->interactive);
```

Attaching a file

Now, to make things more interesting, we are going to attach a GPX file to our request. As the page title and the necessariy bounding box to fit all track data can be extracted from the file, if it is a valid GPX file, we do not need to pass any other data at all.

file-request.php

```
<?php
require_once 'HTTP/Request2.php';

define('BASE_URL', 'https://api.get-map.org/apis/v1/');
define('GPX_FILE', 'A1.gpx');

$data = [];

$request = new HTTP_Request2(BASE_URL . "jobs");

$request->setMethod(HTTP_Request2::METHOD_POST)
        ->addPostParameter('job', json_encode($data))
        ->addUpload('track', GPX_FILE, basename(GPX_FILE), 'application/gpx+xml');

$response = $request->send();
$status   = $response->getStatus();

echo "$status: ".$request->getBody()."\n";

if ($status < 200 || $status > 299) {
    header("Content-type: text/plain");
    print_r($response->getBody());
    exit;
}

$reply = json_decode($response->getBody());

echo "$status: ".$reply->interactive."\n";
```

Wait for rendering to complete

To have your application check for the rendering status yourself you need to poll the `/jobs` status page, adding the job `id` returned by the API. Please wait at least 15 seconds between status poll requests.

As long as the returned HTTP status is "202 Accepted" the request is either currently being processed, or still waiting in the job queue.

When the job has completed successfully "200 OK" will be returned, and the `files` object in the returned JSON result will contain a hash of produced file formats, and under what URL each of them can be retrieved.

In the example we retrieve the PDF version, and start the users preferred PDF viewer to present the result.

wait-for-result.php

```
<?php
```

```php
require_once 'HTTP/Request2.php';

define('BASE_URL', 'https://api.get-map.org/apis/v1/');

function api_GET($url)
{
  $request = new HTTP_Request2(BASE_URL . $url);

  $request->setMethod(HTTP_Request2::METHOD_GET);

  $response = $request->send();
  $status = $response->getStatus();

  if ($status < 200 || $status > 299) {
      echo "invalid HTTP response to '$url': $status\n";
      echo $response->getBody();
      exit(3);
  }

  return array($status, json_decode($response->getBody()));
}


$data = ['title'       => 'PHP Test',
         'bbox_bottom' => 52.00,
         'bbox_top'    => 52.02,
         'bbox_left'   => 8.50,
         'bbox_right'  => 8.52
        ];

$request = new HTTP_Request2(BASE_URL . "jobs");

$request->setMethod(HTTP_Request2::METHOD_POST)
        ->setHeader('Content-type: application/json; charset=utf-8')
        ->setBody(json_encode($data));

$response = $request->send();
$status   = $response->getStatus();

if ($status < 200 || $status > 299) {
    header("Content-type: text/plain");
    print_r($response->getBody());
    exit;
}

$reply = json_decode($response->getBody());

echo $reply->interactive."\n";

$job_url = "/jobs/" .  $reply->id;
```

```
$done = 0;

while (!$done) {
    echo "waiting ... ";
    sleep(15);
    list($status, $reply) = api_GET($job_url);

    switch ($status) {
    case 200:
        $done = 1;
        break;
    case 202:
        switch($reply->status) {
        case 0:
            echo "still in queue\n";
            break;
        case 1:
            echo "now being rendered\n";
            break;
        default:
            echo "unexpected status: ".$reply->status."\n";
            break;
        }
        break;
    default:
        die("unexpected HTTP status: $status");
        break;
    }
}

$pdf = basename($reply->files->pdf);
copy($reply->files->pdf, $pdf);

system("xdg-open $pdf");
rm($pdf);
```

Python

To be done

Javascript

To be done, maybe. Due to the "same origin" restrictions this is low on my list ...