Mekong SharedWater API: Usage Guide

The API behind the Mekong SharedWater portal. It serves satellite and field data for the Lower Mekong basin: landscape change (land cover, EVI, forest loss, mining), water level (SWORD/SWOT reaches and virtual stations), and water pollution (Sentinel-2 turbidity and SPM, monitoring stations, arsenic samples).

- **Base URL (production):** `https://mekong-pollution-servir.adpc.net`
- **Base URL (local dev):** `http://localhost:8000`, or `http://localhost:3000` through the Next.js dev proxy
- **Interactive reference:** `/api/docs` (Swagger UI), `/api/redoc`, and `/api/openapi.json`. These pages need no token.
- **This guide as a PDF:** `/api/docs/api-usage.pdf`, also without a token.

All endpoints are `GET` unless noted otherwise. This guide covers what the generated OpenAPI reference doesn't: authentication, how the resources relate, and how to read the raster pixel values.

---

1. Authentication

Every `/api/*` data endpoint needs an API token, which is a signed JWT. Send it as a bearer header:

```
curl -H "Authorization: Bearer $TOKEN" \
 https://mekong-pollution-servir.adpc.net/api/vectors/basin/active
```

If a client can't set headers (for example an `<img>` tag, or a map library that only takes a URL template), pass the token as the `access_token` query parameter instead. Every endpoint accepts it:

```
https://mekong-pollution-servir.adpc.net/api/webcam/main/snapshot?access_token=<token>
```

Prefer the header wherever you can, because a token in a query string ends up in logs and browser history.

Response	Meaning
`401 Missing API token…`	No header and no `access_token` parameter.
`401 Invalid API token.`	The signature is wrong, the token is malformed, or it was signed for a different environment.
`401 API token has expired.`	The token's lifetime has passed, so ask for a new one.

Getting a token

The portal team issues tokens, each one named for a partner or service. If you maintain the backend, mint one on a host where `JWT_SECRET` matches the target environment:

```
cd backend
python scripts/issue_api_token.py --subject "partner-name"            # valid for 365 days
python scripts/issue_api_token.py --subject "partner-name" --days 30
python scripts/issue_api_token.py --subject "partner-name" --no-expiry
```

A token only works against a backend that uses the same `JWT_SECRET` it was signed with. A token signed with the local dev default is rejected in production.

Rate limits

The default limit is **300 requests per minute** (set by `API_RATE_LIMIT`). The limit applies per token subject. The portal's own frontend token (subjects starting with `sharedwater-portal`) and requests without a valid token are counted per client IP instead. A client over the limit gets `429 Too Many Requests`.

A map view can easily request dozens of tiles at once. If you pre-fetch tiles in bulk, throttle the requests or ask for a higher limit.

---

2. Conventions

Watersheds: `hybas_id` and `level`

Per-basin statistics are keyed by **HydroBASINS `HYBAS_ID`** (for example `4061040790`). The server runs at one active watershed level, 6 by default, and several endpoints accept `?level=5` or `?level=6` to override it for a single request.

To find valid IDs:

```
GET /api/vectors/basin/active          # {"level": 6, "featured_basin_id": 4061040790}
GET /api/vectors/basin/active.geojson  # the boundary polygons; each feature has
   properties.HYBAS_ID
```

Response formats

Kind	Content type	Notes
JSON / GeoJSON	`application/json`	GeoJSON is always a `FeatureCollection` in EPSG:4326.
Raster tiles	`image/png`, `image/webp`, …	Web Mercator XYZ tiles.
Vector tiles	`application/vnd.mapbox-vector-tile`	Mapbox Vector Tile (`.mvt`).
TileJSON	`application/json`	TileJSON 3.0.0, which you can pass straight to MapLibre, Mapbox, or Leaflet plugins.

An **empty tile** (outside the data extent, or no features) comes back as `204 No Content`, which is not an error.

Caching

- Raster tiles under `/api/cogs/` and `/api/tiles/raster/` are sent with `Cache-Control: public, max-age=2592000, immutable` (30 days). The server assumes a given tile URL never changes. If you need fresh pixels after a dataset is updated, add any unused query parameter such as `&v=2026-09` to bypass your cache.
- Static vector data (boundaries, station GeoJSON, vector tiles) is cached for 30 days.
- Frequently changing data (basin statistics, `/api/vectors/basin/active*`, mining sites) is revalidated on every request.
- Live data (`/api/webcam/*`, `/api/wpi/reaches.geojson`, `/api/tiles/raster/evi`) is sent `no-store`.

Errors

Errors use FastAPI's standard shape, `{"detail": "..."}`. Common statuses are `400` (bad parameters, such as a malformed month or an unsupported band), `404` (unknown ID, or data not present on the server), `422` (parameter validation failed), and `503` (the PostGIS database is unavailable; only affects the reach and station-metrics endpoints).

---

3. Endpoint reference

3.1 General

Endpoint	Description
`GET /healthz`	Liveness check that returns `{"status": "ok"}`. It needs no token. It sits outside `/api/`, so it's only reachable on the backend itself, not through the public host.
`GET /api/docs/api-usage.pdf`	This guide as a PDF. It needs no token, like the Swagger and ReDoc pages it sits next to.
`GET /api/cogs`	Catalogue of every raster (Cloud Optimized GeoTIFF). See §4.
`GET /api/cogs/water-quality-colormaps`	Colour stops, units, and pixel encoding for the turbidity, SPM, and float-NTU turbidity ramps.
`GET /api/cogs/{cog_id}/info`	Bounds, zoom range, band metadata, and per-band statistics for one raster.
`GET /api/cogs/{cog_id}/tilejson.json`	TileJSON for one raster, using its default styling. Takes the optional parameters `palette`, `minzoom`, and `maxzoom`.
`GET /api/cogs/{cog_id}/point?lon=&lat=`	Decoded turbidity and SPM value at a point. Only works for water-quality rasters. See §4.3.
`GET /api/tiles/raster/...`	General-purpose tile, preview, statistics, and point routes for any raster (TiTiler). See §4.2.

3.2 Landscape Change

Watershed boundaries

Endpoint	Description
`GET /api/vectors/basin/active?level=`	The active watershed level and the basin the portal features by default.
`GET /api/vectors/basin/active.geojson?level=`	Watershed polygons for that level, with HydroBASINS attributes (`HYBAS_ID`, `SUB_AREA`, `UP_AREA`, …).
`GET /api/vectors/basin/l5.geojson`	Simplified level-5 basin polygons.
`GET /api/vectors/boundary/mekong.geojson`	Outline of the whole Lower Mekong basin.

Per-basin statistics. Each list endpoint returns the basin IDs that have data. Each `{hybas_id}` endpoint takes `?level=5|6`.

Endpoint	Response
`GET /api/landcover/basins`	`{"basins": [...]}`

Endpoint	Response
`GET /api/landcover/basins/{hybas_id}`	`{hybas_id, sub_name, area_ha, classes: [{value, label, color, area_ha, pct}]}`, the land-cover mix from SERVIR RLCMS 2023.
`GET /api/drivers-of-forest-loss/basins`	`{"basins": [...]}`
`GET /api/drivers-of-forest-loss/basins/{hybas_id}`	Same shape as land cover. `classes` lists the forest-loss drivers (WRI / Google DeepMind).
`GET /api/tree-cover/basins`	`{"basins": [hybas_id, ...]}`
`GET /api/tree-cover/basins/{hybas_id}`	`{hybas_id, sample_count, min_pct, max_pct, mean_pct, median_pct, pct_ge_10, pct_ge_30, pct_ge_50}`, canopy cover in 2000 (UMD Hansen).
`GET /api/landscape/forest-loss/basins`	`{basins: [{hybas_id, count, start, end, total_ha}]}`
`GET /api/landscape/forest-loss/basins/summary`	`{basin_count, unit, points: [{year, area_ha}]}`, the yearly loss summed over every basin.
`GET /api/landscape/forest-loss/basins/{hybas_id}`	`{hybas_id, unit, points: [{year, area_ha}]}`
`GET /api/evi/{sensor}/basins`	`{sensor, basins: [{hybas_id, count, start, end}]}`. Only `sensor=modis` is available.
`GET /api/evi/{sensor}/basins/{hybas_id}?limit=&level=`	`{hybas_id, sensor, unit, points: [{time_str, ym, evi_anom, cum_evi, n_valid}]}`, the monthly EVI anomaly and its cumulative total. `limit` keeps the most recent *n* points.

EVI anomaly tiles

Endpoint	Description
`GET /api/tiles/raster/evi`	Lists the EVI layers with `id`, `name`, `rescale` (the value range the colour ramp covers, for legends), and a tile URL.
`GET /api/tiles/raster/evi/{evi_id}/tilejson.json`	TileJSON for one layer.
`GET /api/tiles/raster/evi/{evi_id}/tiles/{z}/{x}/{y}.webp`	Tile. Optional parameters: `palette` (`rdylgn` default, `bwr`, `piyg`), `rescale_min`, `rescale_max`, `scale` (1-4, where 2 gives 1024 px).

`evi_id` is one of `modis_1`, `modis_5`, `modis_10` (cumulative anomaly over the last 1, 5, or 10 years), `landsat_short`, or `landsat_long` (Landsat EVI trend slopes).

Mining sites

Endpoint	Description
`GET /api/vectors/mining/sites.geojson`	Mining site points with `properties: {basin, year, year_sort}`. Supports `ETag` and `If-None-Match`, returning `304` when unchanged.
`GET /api/vectors/mining/basins/{hybas_id}`	`{hybas_id, total, years: [{year, count}]}`, the sites that fall inside that watershed polygon.

3.3 Water Level Monitoring

River network (SWORD reaches)

Endpoint	Description
`GET /api/vectors/streams/tilejson.json`	TileJSON for the river network vector tiles (layer `stream`, fields `reach_id`, `river_name`).
`GET /api/vectors/streams/tiles/{z}/{x}/{y}.mvt`	River network vector tile. `/api/tiles/vector/streams/{z}/{x}/{y}.mvt` returns the same tile.
`GET /api/vectors/streams/wse`	Water-level class for each reach in the latest month: `{column: "YYYY-MM", classes: [{code, label}], count, rows: [{reach_id, value}]}`. Codes run from `0` (Low) to `4` (High). Each reach is ranked against its own history: <=P15 Low, <P25 Transitional-low, <=P75 Medium, <P85 Transitional-high, above that High. Join `rows` to the vector tile features on `reach_id`.
`GET /api/vectors/reach/tilejson.json?year=&month=`	TileJSON for monthly reach tiles (layer `reach`). Without `year` and `month` it uses the latest available month. `year` and `month` must be given together.
`GET /api/vectors/reach/tiles/{z}/{x}/{y}.mvt?year=&month=`	Monthly reach vector tile.
`GET /api/vectors/reach/feature/{reach_id}?year=&month=`	`{month, reach_id, river_name, base_properties}`

`GET /api/vectors/reach/{YYYY-MM}/tilejson.json`
`GET /api/vectors/reach/{YYYY-MM}/tiles/{z}/{x}/{y}.mvt`
`GET /api/vectors/reach/{YYYY-MM}/feature/{reach_id}`
	The same three routes with the month in the path.

The reach endpoints read from PostGIS and return `503` when the database is unavailable.

SWOT virtual stations (SWORD nodes)

Endpoint	Description
`GET /api/vectors/nodes/points.geojson`	Virtual station points with `properties: {node_id, reach_id, river_name}`.

Endpoint	Description
`GET /api/vectors/nodes/nearest?lat=&lon=&require_timeseries=true`	The station nearest to a point: `{node_id, reach_id, river_name, lat, lon, distance_m}`. By default it skips stations that have no time series.
`GET /api/vectors/nodes/{node_id}/timeseries?limit=`	`{node_id, unit, country, hybas_id, river_name, points: [{time_str, wse, stage, node_q}]}`, the water surface elevation from each SWOT pass in time order. `limit` keeps the most recent *n* points. `country`, `hybas_id`, and `river_name` can be `null` for older stations.

3.4 Water Pollution

Endpoint	Description
`GET /api/vectors/pcd/pcd_stations.geojson`	Thai Pollution Control Department monitoring stations. Properties include the station `Code`, `Name`, `Province`, the most recent field readings (`BOD_ppm`, `pH`, `Turbidity_`, …) and the `Quality_St` class.
`GET /api/vectors/sword/sword_matched.geojson`	SWORD nodes used as satellite sampling points: `properties: {node_id}`.
`GET /api/water-quality/stations/{station_type}/latest`	The latest satellite-derived reading at every station of one type: `{station_type, stations: [{station_id, month, turbidity_ntu, spm_mgl}]}`.
`GET /api/water-quality/stations/{station_type}/{station_id}/timeseries`	`{station_type, station_id, points: [{month, turbidity_ntu, spm_mgl}]}`, monthly Sentinel-2 medians sampled at the station.

`station_type` is `pcd` or `sword`. `station_id` is the PCD station's `Code` (for example `CI03`), or the SWORD `node_id`. These readings come from PostGIS, so the endpoints return `503` when the database is unavailable. The rasters behind them are the monthly `s2_mekong_median_*` COGs described in §4.

3.5 Use Case (Kok basin storymap)

Endpoint	Description
`GET /api/webcam/locations`	`{cameras: [{id, name, nameTh, lat, lon}]}`
`GET /api/webcam/{camera_id}/snapshot`	The latest JPEG frame, refreshed every few minutes. To use it as an `<img src>`, pass the token with `?access_token=`.
`GET /api/arsenic/field-data`	`{samples: [{id, lat, lon, type, sampledAt, concentration, units, sourceLink, riskTier}], summary: {total, locations, low, medium, high, aboveLimit, minimum, maximum, mean, median, latestObservationDate, observationDates}}`. `riskTier` is `low` (<= 0.010), `medium` (<= 0.030), or `high`, measured in the sample's `units`.
`GET /api/citizen-science/water-quality/stations`	Kok basin stations: `[{id, type: "pcd"|"sword", geometry}]`.
`GET /api/citizen-science/water-quality/stations/{station_id}/timeseries?band=turbidity|spm`	Ten-day ("dekadal") statistics: `[{Year, Month, Dekad, mean, median, max}]`. Periods with no data are included with `null` values, so the series has no gaps.
`GET /api/wpi/reaches.geojson`	Reach lines with the latest turbidity-based Water Pollution Index: `properties: {reachId, river, status, turbidityNtu, turbidityIndex, satDate}`. Reaches with no current reading are still included, with `status: null`. The data is regenerated daily.

---

4. Raster data (COGs)

4.1 The catalogue and COG IDs

`GET /api/cogs` returns every raster on the server:

```
[
 {
   "id": "s2_mekong_median_2024_05",
   "name": "S2 Mekong Median 2024 05",
   "tilejson": "/api/cogs/s2_mekong_median_2024_05/tilejson.json",
   "tiles": ["https://…/api/tiles/raster/WebMercatorQuad/{z}/{x}/{y}.png?cog_id=s2_mekong_m
       edian_2024_05&bidx=1&colormap_name=turbidity&nodata=65535"]
 }
]
```

A raster's `id` is its filename without the extension, in lower case, with any character outside `a-z 0-9 _ -` replaced by `-`. Always get the current list from this endpoint, because datasets are added over time without code changes. The main families are:

COG id pattern	Dataset	Pixel values
`s2_mekong_median_YYYY_MM`	Sentinel-2 monthly median across the Mekong. Band 1 is turbidity, band 2 is SPM.	uint16, log10 encoded (see below)

COG id pattern	Dataset	Pixel values
`s2_mekong_median_YYYY_MM_validcount`	Number of cloud-free scenes behind each pixel of that month. Only some months have one.	uint8, 0 = no data
`s2_kok_median_tur_YYYY_MM_d1` … `_d3`	Kok basin ten-day turbidity (D1 covers days 1-10, D2 days 11-20, D3 the rest of the month).	float32, in **NTU**, NoData -9999, capped at 1000
`lmb_30day_turbidity_*`, `mekong_s2_tur_*`	Basin-wide turbidity mosaics, turbidity only and with no SPM band. `lmb_30day_turbidity_10m_mosaic` is a rolling composite of the last 30 days, so it carries no date in its ID and is replaced in place as new imagery arrives.	float32, in **NTU**, NoData -9999, capped at 1000
`s2_kok_median_YYYY_MM_d1` … `_d3`	Older Kok ten-day turbidity and SPM files.	uint16, log10 encoded
`rlcms_2023_mekong_v3-001`	SERVIR RLCMS 2023 land cover	class codes 1-22, NoData -9999
`drivers_of_forest_loss_2001_2025_lowermekong`	Drivers of forest loss	class codes 1-7, 0 = no loss
`umd_treecover2000`	Tree canopy cover in 2000	percent, 0-100
`umd_forest_lossyear`	Year of forest loss	1-24 (= 2001-2024), 0 = no loss
`grwl_mekongbasin_landsat`	GRWL surface water extent	255 = water
`modis_evi_trend`, `landsat_evi_trend`	Greening/browning trend (Sen's slope)	uint16 `clip(slope, -500, 500) + 500`, so 500 = no change, 1001 = NoData
`modis_evi_trend_class`, `landsat_evi_trend_class`	Trend class	1 significant browning … 5 significant greening

Water-quality pixel encoding (log10)

Pixels in `s2_mekong_median_*` and the older Kok files map to real values with:

```
real = 10 ** ((pixel / 65534) * 6.0 - 2.0)      # covers 0.01 to 10,000
```

Turbidity (NTU) and SPM (g/m³) use the same scale. Pixel `65535` is NoData and `65534` means the value was clipped at the top of the range. Treat both as having no value. `GET /api/cogs/water-quality-colormaps` returns these constants along with the colour stops.

The float32 rasters (`s2_kok_median_tur_*`, `lmb_30day_turbidity_*`, `mekong_s2_tur_*`) are **not** encoded: their pixel value is the NTU reading itself, and -9999 is NoData.

4.2 Tiles and other raster operations

Raster tiles are rendered by TiTiler under `/api/tiles/raster`. In place of TiTiler's usual `url` parameter, each request identifies its raster with **`cog_id`**. The main routes are:

Endpoint	Description
`GET /api/tiles/raster/WebMercatorQuad/{z}/{x}/{y}.{png|webp|jpg|tif|npy}`	XYZ tile. `@2x` or `?scale=2` returns a 512 px tile at double resolution.
`GET /api/tiles/raster/WebMercatorQuad/tilejson.json`	TileJSON built from the same query parameters.
`GET /api/tiles/raster/WebMercatorQuad/map.html`	Quick map viewer for checking a layer in a browser.
`GET /api/tiles/raster/info?cog_id=`	Raster metadata.
`GET|POST /api/tiles/raster/statistics`	Band statistics and histograms. The `POST` form takes a GeoJSON feature and computes the statistics inside it.
`GET /api/tiles/raster/point/{lon},{lat}`	Raw pixel values at a point.
`GET /api/tiles/raster/preview.{format}`, `/bbox/{minx},{miny},{maxx},{maxy}.{format}`, `POST /feature.{format}`	Static images of the whole raster, a bounding box, or a GeoJSON area.

Common query parameters are `bidx` (band index), `nodata`, `rescale=min,max`, `colormap_name`, `algorithm`, and `expression`. The full list is in `/api/docs`.

Named colormaps that match the portal's styling:

`colormap_name`	Use with	Extra parameters required
`turbidity`	`s2_mekong_median_*`, older Kok files	`bidx=1&nodata=65535`
`spm`	same	`bidx=2&nodata=65535`
`water-quality-confidence`	`*_validcount`	`bidx=1`. Greys out pixels with fewer than 2 cloud-free scenes.
`kok-turbidity`	every float-NTU raster: `s2_kok_median_tur_*`, `lmb_30day_turbidity_*`, `mekong_s2_tur_*`	`algorithm=kok_turbidity_index`, which is **required** because the data is floating point
`landcover`	`rlcms_2023_mekong_v3-001`	`nodata=-9999`
`drivers`	`drivers_of_forest_loss_*`	
`green` / `red`	`umd_treecover2000` / `umd_forest_lossyear`	
`water`	`grwl_mekongbasin_landsat`	
`greening-browning-trend`	`*_evi_trend`	
`greening-browning-trend-class`	`*_evi_trend_class`	

The standard rio-tiler colormaps (`viridis`, `magma`, …) are also available.

Examples, as the portal requests them:

```
# Monthly turbidity, May 2024
/api/tiles/raster/WebMercatorQuad/{z}/{x}/{y}.webp?cog_id=s2_mekong_median_2024_05&bidx=1&colorm
   ap_name=turbidity&nodata=65535&scale=2

# Kok ten-day turbidity, second ten days of August 2025
/api/tiles/raster/WebMercatorQuad/{z}/{x}/{y}.webp?cog_id=s2_kok_median_tur_2025_08_d2&algorithm
   =kok_turbidity_index&colormap_name=kok-turbidity&scale=2

# Rolling 30-day basin-wide turbidity mosaic (same algorithm and ramp)
/api/tiles/raster/WebMercatorQuad/{z}/{x}/{y}.webp?cog_id=lmb_30day_turbidity_10m_mosaic&algorit
   hm=kok_turbidity_index&colormap_name=kok-turbidity&scale=2

# Land cover
/api/tiles/raster/WebMercatorQuad/{z}/{x}/{y}.webp?cog_id=rlcms_2023_mekong_v3-001&colormap_name
   =landcover&nodata=-9999&scale=2
```

`GET /api/cogs/{cog_id}/tiles/{z}/{x}/{y}.png?palette=&band=&scale=` is an older tile route that still works for the log10 water-quality, tree-cover, forest-loss, and water-extent rasters. It rejects the float-NTU turbidity rasters with `400`. New clients should use the TiTiler routes above.

4.3 Reading water-quality values at a point

`GET /api/cogs/{cog_id}/point?lon=&lat=` applies the same decoding the tiles use, so the numbers match the colours on the map:

```
{
 "cog_id": "s2_mekong_median_2024_05",
 "lon": 104.25, "lat": 15.52,
 "encoding": "log10",
 "turbidity": {"pixel": 38120, "value": 30.9, "unit": "NTU"},
 "spm":       {"pixel": 37400, "value": 26.6, "unit": "g/m³"},
 "valid_count": 3,
 "valid_count_available": true,
 "min_confident_valid_count": 2
}
```

- `value` is `null` when the pixel is NoData or clipped, or when the point falls outside the raster.
- `valid_count` is `null` when the month has no valid-count raster, or when that raster has no reading at this point. If `valid_count` is below `min_confident_valid_count`, treat the reading as low confidence.
- For the float-NTU rasters (`s2_kok_median_tur_*`, `lmb_30day_turbidity_*`, `mekong_s2_tur_*`), `encoding` is `"linear"`, `pixel` and `value` are both the NTU reading, and `spm` is `null`.
- Any raster that isn't a water-quality raster returns `400`. For those, use `/api/tiles/raster/point/{lon},{lat}?cog_id=` to get raw values.

---

5. Using the API from a map client

MapLibre GL JS. MapLibre fetches tiles from a web worker, where your page's own `fetch` settings don't apply. Add the token to requests with `transformRequest`:

```
const API = "https://mekong-pollution-servir.adpc.net";

const map = new maplibregl.Map({
 container: "map",
 style: "https://demotiles.maplibre.org/style.json",
 transformRequest: (url) =>
   url.startsWith(`${API}/api/`)
     ? { url, headers: { Authorization: `Bearer ${TOKEN}` } }
     : { url },
});

map.on("load", () => {
```

```
 map.addSource("turbidity", {
   type: "raster",
   tileSize: 512,
   tiles: [
     `${API}/api/tiles/raster/WebMercatorQuad/{z}/{x}/{y}.webp` +
      "?cog_id=s2_mekong_median_2024_05&bidx=1&colormap_name=turbidity&nodata=65535&scale=2",
   ],
 });
 map.addLayer({ id: "turbidity", type: "raster", source: "turbidity" });

 map.addSource("rivers", { type: "vector", url: `${API}/api/vectors/streams/tilejson.json` });
 map.addLayer({ id: "rivers", type: "line", source: "rivers", "source-layer": "stream" });
});
```

Leaflet, QGIS, and other clients that can't set headers. Add `&access_token=<token>` to the tile URL template.

Python.

```
import requests

API = "https://mekong-pollution-servir.adpc.net"
s = requests.Session()
s.headers["Authorization"] = f"Bearer {TOKEN}"

featured = s.get(f"{API}/api/vectors/basin/active").json()["featured_basin_id"]
loss = s.get(f"{API}/api/landscape/forest-loss/basins/{featured}").json()
for p in loss["points"]:
   print(p["year"], p["area_ha"])
```