

API Documentation

Contents

Chapter 1. Introduction.....	3
Getting Started: Obtain an API Key.....	3
Authentication.....	5
Pagination.....	5
Rate Limiting.....	6
Chapter 2. GET /Weather.....	7
How to Get Current Weather.....	7
GET Weather Response Fields.....	9
Chapter 3. POST /task.....	12
How to Post Task.....	12
POST Task Response Fileds.....	13

Chapter 1. Introduction

Welcome to the **Weather & Task API** documentation. This API provides two powerful sets of endpoints designed to help developers integrate weather data and task management capabilities into their applications with ease.

The Weather & Task API provides two sets of functionalities

1. A simple /weather endpoint for retrieving real-time weather by city. It's ideal for apps that need to display live temperature, weather descriptions, or customize content based on location.
2. A /tasks endpoints offer a full-featured task management system. Developers can:
 - Create, read, update, and delete tasks (CRUD)
 - Retrieve paginated task lists
 - Perform partial updates (PATCH)
 - Use query parameters for flexible data fetching
 - Handle rate limits gracefully using standard HTTP headers

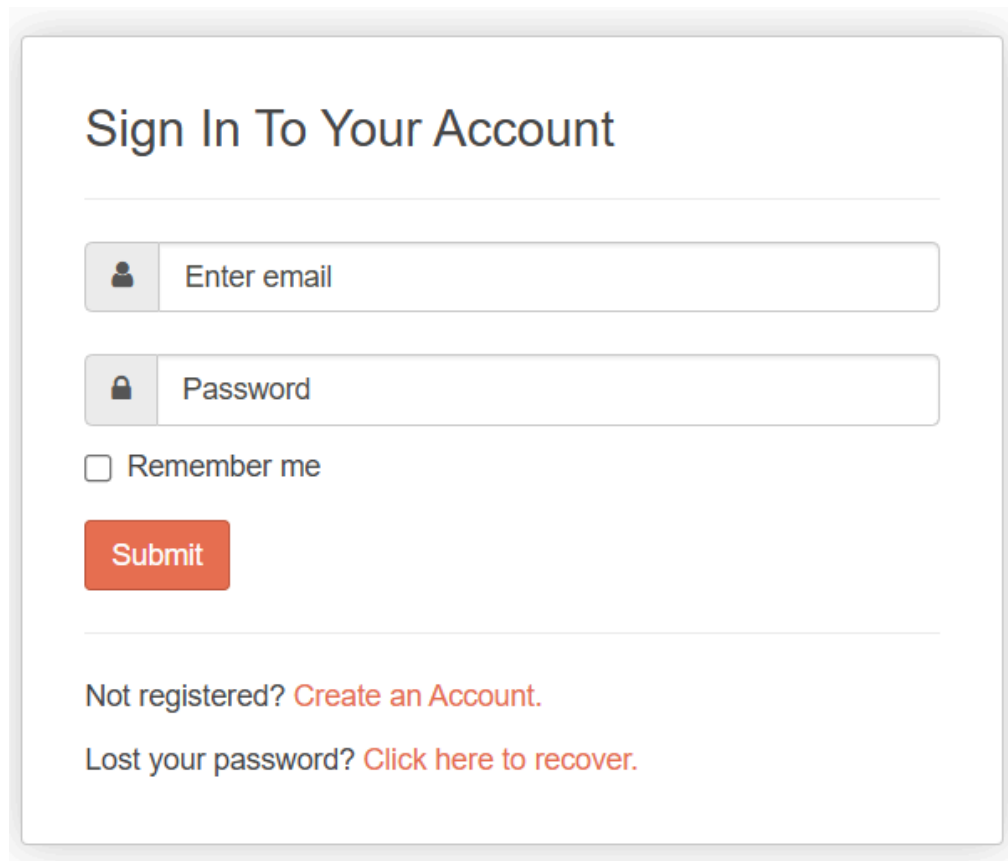
This documentation is your complete guide to integrating and using the Weather & Task API efficiently. Whether you're building a weather widget or a productivity tool, you'll find all the examples, parameters, and error handling you need to succeed.

Getting Started: Obtain an API Key

Follow these steps to get your API key from OpenWeatherMap for use with the Weather API.

1. Go to [Openweathermap](#) website.
2. Click on **Sign in to your account**.

Figure 1. Postman Login Screen



The image shows a login form titled "Sign In To Your Account". It features two input fields: "Enter email" with a user icon and "Password" with a lock icon. Below these is a checkbox labeled "Remember me" and a red "Submit" button. At the bottom, there are two links: "Not registered? Create an Account." and "Lost your password? Click here to recover."

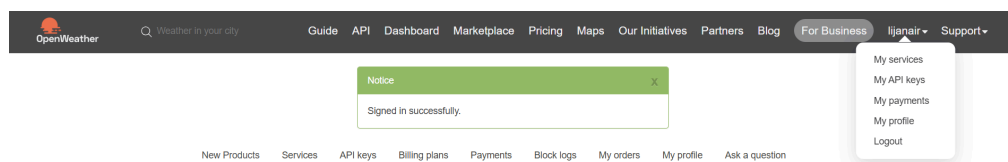


Note:

If you don't have an account, click **Create an Account** and register.

3. Once signed in, click on your profile name in the top menu.
4. From the dropdown, select **My API keys**.

Figure 2. My API Key



5. You'll see your existing API keys listed. Do one of the following:

- a. Copy an existing key, or
- b. Click **Generate** to create a new API key. Copy the new API Key.

**Important:**

These API keys are secret and should not be shared with anyone. Treat them like passwords.

6. Optionally, give your API key a name for easy identification.

Authentication

Authentication is the process of verifying the identity of a client before granting access to the API.

This API uses an API key to authenticate requests. Including the correct API key in your request headers is required to access protected endpoints. It ensures that only authorized users can use the service and helps track usage per user.

Refer to the [Getting Started \(on page 3\)](#) section to learn how to obtain your API key.

Pagination

Pagination refers to dividing a large set of results into smaller, manageable chunks or "pages." When fetching data like task lists or weather logs, pagination prevents overloading the client and server with too much data at once. It also helps improve performance and allows users to navigate through data efficiently using `page` and `limit` parameters.

For endpoints returning large datasets (e.g., `/tasks`):

- Use query parameters:
 - `page`: Page number (default: 1)
 - `limit`: Items per page (default: 10, max: 100)

Example

```
GET /tasks?page=2&limit=20
```

Rate Limiting

The API enforces rate limiting. It restricts the number of API requests a client can make within a specified time period. To maintain server performance and ensure fair access for all users, this API limits the number of requests based on the user's subscription level. If the limit is exceeded, the API returns a `429 Too Many Requests` error. This helps protect the API from abuse and overload.

1. Limit: 60 requests per minute.

2. Headers:

- `X-RateLimit-Limit`: Max requests allowed
- `X-RateLimit-Remaining`: Requests left
- `Retry-After`: Time to wait before retrying after hitting the limit

3. On exceeding limit:

```
HTTP 429 Too Many Requests
```

Chapter 2. GET /Weather

The `GET /Weather` endpoint allows users to retrieve current weather data for any location on Earth. It aggregates real-time data from global and local weather models, satellites, radars, and an extensive network of weather stations. The data is available in multiple formats, including JSON, XML, and HTML.

This endpoint is designed to provide detailed weather information such as temperature, humidity, wind speed and direction, atmospheric pressure, cloud coverage, and visibility.

Usecases

- Displaying weather information in travel and tourism apps
- Powering smart home devices to react to local weather conditions
- Building weather widgets or dashboards for websites or mobile apps

How to Get Current Weather

You must have a valid API key. Refer to the [Getting Started t \(on page 3\)](#)opic for more details.

Ensure your application can make HTTP GET requests.

The `/weather` endpoint provides real-time weather data using various query parameters such as city name or coordinates. Responses are returned in JSON by default.

1. Send a **GET** request to the following endpoint:

```
https://api.openweathermap.org/data/2.5/weather
```

2. Include the following parameters:

Table 1. Parameters

Name	Parameter Type (in)	Example
zip	query	65051
units	query	Imperial

3. Include your API key using the `appid` query parameter.

```
appid=your_api_key
```

4. (Optional) Add other query parameters to customize the response:

- `units` – Metric, imperial, or standard
- `mode` – Response format (json, xml, html)

5. Execute the request and inspect the response.

```
GET https://api.openweathermap.org/data/2.5/weather?zip=95050&units=imperial
```

```
curl --location --globoff
```

```
'https://api.openweathermap.org/data/2.5/weather?zip=95050&units=imperial&appid=*****'
```

If the request is successful, a `200 OK` response is returned with the current weather data.

Example:

```
{
  "coord": {
    "lon": -121.953,
    "lat": 37.3492
  },
  "weather": [
    {
      "id": 804,
      "main": "Clouds",
      "description": "overcast clouds",
      "icon": "04n"
    }
  ],
  "base": "stations",
  "main": {
    "temp": 53.46,
    "feels_like": 52.36,
    "temp_min": 50.41,
    "temp_max": 54.28,
    "pressure": 1018,
    "humidity": 82,
    "sea_level": 1018,
    "grnd_level": 996
  },
  "visibility": 10000,
  "wind": {
    "speed": 3.44,
```



```
    "deg": 0
  },
  "clouds": {
    "all": 100
  },
  "dt": 1745147846,
  "sys": {
    "type": 1,
    "id": 5845,
    "country": "US",
    "sunrise": 1745155545,
    "sunset": 1745203627
  },
  "timezone": -25200,
  "id": 0,
  "name": "Santa Clara",
  "cod": 200
}
```

Handle possible errors:

- 401 Unauthorized – Invalid or missing API key
- 404 Not Found – Invalid location parameters
- 429 Too Many Requests – Rate limit exceeded

GET Weather Response Fields

Field	Description
coord.lon	Longitude of the location
coord.lat	Latitude of the location
weather.id	Weather condition ID
weather.main	Group of weather parameters (Rain, Snow, Clouds, etc.)
weather.description	Detailed description of the weather condition
weather.icon	Weather icon ID

Field	Description
base	Internal parameter
main.temp	Temperature. Default: Kelvin. Metric: Celsius. Imperial: Fahrenheit
main.feels_like	Perceived temperature. Same units as <code>main.temp</code>
main.pressure	Atmospheric pressure (hPa)
main.humidity	Humidity in percentage (%)
main.temp_min	Minimum observed temperature. Same units as <code>main.temp</code>
main.temp_max	Maximum observed temperature. Same units as <code>main.temp</code>
main.sea_level	Atmospheric pressure at sea level (hPa)
main.grnd_level	Atmospheric pressure at ground level (hPa)
visibility	Visibility in meters. Max value: 10,000
wind.speed	Wind speed. Default/Metric: m/s, Imperial: mph
wind.deg	Wind direction in degrees (meteorological)
wind.gust	Wind gust. Same units as <code>wind.speed</code>
clouds.all	Cloudiness in percentage (%)
rain.1h	Precipitation in the last hour (mm/h), if available
snow.1h	Snowfall in the last hour (mm/h), if available
dt	Time of data calculation (UNIX UTC)
sys.type	Internal parameter
sys.id	Internal parameter
sys.country	Country code (e.g., GB, JP)
sys.sunrise	Sunrise time (UNIX UTC)
sys.sunset	Sunset time (UNIX UTC)
timezone	Time shift from UTC in seconds

Field	Description
id	City ID
name	City name
cod	Internal parameter (status code)

Chapter 3. POST /task

The `POST /tasks` endpoint allows clients to create new tasks in the task management system. This endpoint is publicly accessible and does not require authentication.

A task represents a unit of work, including attributes such as title, description, due date, status, and priority.

Developers can use this endpoint to add tasks dynamically to a user's list or queue.

Usecases

- User opens a to-do application.
- They enter task details such as the title, due date, and priority.
- The application sends a `POST` request to `/tasks` with the task data.
- The server responds with 201 Created status and the created task details

How to Post Task

The `POST /tasks` endpoint allows you to create a new task by providing task details in the request body.

1. Send a `POST` request to `/tasks` endpoint.
Use JSON format in the request body.
2. Include the following fields in your request body:
 - `title` – The title of the task (required)
 - `description` – Details about the task (optional)
 - `due_date` – Due date in ISO 8601 format (optional)
 - `priority` – An integer from 1 (low) to 5 (high)

```
{
  "title": "Finish API Documentation",
  "description": "Complete the POST /tasks section in the Task API doc.",
  "due_date": "2025-04-25T18:00:00Z",
  "priority": 3
}
```

3. Check the response status:

◦ 201 Created – Task created successfully

```
{
  "id": "a1b2c3d4e5",
  "title": "Finish API Documentation",
  "description": "Complete the POST /tasks section in the Task API doc.",
  "due_date": "2025-04-25T18:00:00Z",
  "priority": 3,
  "status": "pending",
  "created_at": "2025-04-21T14:32:10Z"
}
```

◦ 400 Bad Request – Missing or invalid fields

A successful response returns the complete task object with an assigned ID.

POST Task Response Fields

Field	Type	Description
id	string	Unique identifier of the task
title	string	Title of the task
description	string	Additional details about the task
due_date	string (ISO 8601)	Deadline for the task
priority	integer	Priority level from 1 (low) to 5 (high)
status	string	Current status (e.g., pending, completed)
created_at	string (ISO 8601)	Timestamp when the task was created