Audience Scope

This guide is designed for Windows users with basic computer skills, aimed to an audience familiar with terms like file extensions (e.g., .txt), desktop shortcuts, and folder navigation. By the end, you'll confidently install, set up Notepad++, resolve troubleshoot issues, and use basic features like customize settings and create/save files, even with no prior text editor experience.

Document's scope

This document describes how to do the following:

1. Authenticate:
- Obtain and securely store API keys.

2. Call Core Endpoints:
- Fetch real-time weather data via /weather.
- Retrieve 5-day forecasts via /forecast.

3. Parse Responses:
- Extract values like temp from nested JSON objects.

4. Handle Errors:
- Handle authentication errors (HTTP 401).
- Fix "City not found" (HTTP 404).
- Debug rate-limiting errors (HTTP 429).

5. Ensure Compliance:
- Follow GDPR/data licensing rules.

This document does not describe the following:

1. Advanced Features:
- Historical weather archives (enterprise-tier endpoints).
- Custom webhook configurations.

2. Integrations:
- Connecting to databases/IDEs (e.g., PostgreSQL, VS Code).
- Third-party tools like Salesforce or HubSpot.

3. Custom Scripting:
- Automating bulk API calls with cron jobs.
- Building custom plugins or wrappers.

Outline Sections

1. Authentication and Key testing.
2. Call Core Endpoints.
3. Parse Responses.
4. Handle Errors.
5. Ensure Compliance

Section 1: Authentication and Key testing.

1.1 Obtain API Key.

Objective:
Generate and securely store an OpenWeatherMap API key to authenticate requests.

Requirements:

- Web browser.
- Valid Email account.

Steps:

1. Visit the official website of Open Weather Map API: https://openweathermap.org/
2. Sign in to your account.
    a. If you don't have an account, create one by confirming your email.
3. Select **Open** in the top menu.
    a. Select **My API keys** from the dropdown menu.
    b. Your active API keys will appear in a table.
4. Enter a unique name for your key (e.g., 'MyTravelApp') in the API Key Name field.
    a. Select **Generate**.
    b. A new key will show up.

1.2 Test New Generated Key

Objective:
Verifying working API keys and interpreting responses.

Requirements:

- Generated API key.
- Web browser

Steps:

1. Open a new browser tab.
2. Test your key with this cURL command:
```
https://api.openweathermap.org/data/2.5/weather?q=London&appid=[API_KEY]
```
3. Replace [API_KEY] with the **new generated key**.
4. Press **Enter**.

Following table explains the status codes returned during API key verification:

Code	Message	Result
200	`{"coord": {"lon": -0.13, "lat": 51.51}, "main": {"temp": 278.53}, "cod": 200}`	Working API Key
401	`{"cod": 401, "message": "Invalid API key."}`	Not working key

Section 2: Call Core Endpoints

2.1 Real-Time Weather Data (/weather)

Objective:
Fetch current weather data for any global location using OpenWeatherMap's API.

Requirements:
- Generated OpenWeatherMap API key (see Section 1).
- Web browser or API testing tool.

Parameters:
The OpenWeatherMap API allows you to fetch weather data using **city name**, **state code** (U.S. locations only), and **country code**.

The following table lists the parameters for the */weather* endpoint, including their type (required/optional) and purpose:

Parameter	Type	Description
q	required	City name, state (US only), and ISO 3166 country code, comma-separated. Example: `q=Paris,FR`
appid	required	Generated API key
mode	optional	Response format: *JSON*, *XML* and *HTML*. Default: *JSON*.
units	optional	Measurement system: *standard, metric* and *imperial*. Default parameter is: *standard*
lang	optional	Language code.

API Call examples:

1. *Basic request from Tokyo, Japan.*

```
# Get Tokyo weather (JSON)
curl
https://api.openweathermap.org/data/2.5/weather?q=Tokyo,JP&appid=[API_KEY]"
```

Response snippet:

```
  },
  "timezone": 32400,
  "id": 1850144,
  "name": "Tokyo",
  "cod": 200
}
```

2. *Request from Paris, France, in celsius units.*

```
https://api.openweathermap.org/data/2.5/weather?q=Paris,FR&units=metric&appid=[API_KEY]
```

Response snippet:

```
"main": {
   "temp": 13.47, // Celsius
}
```

3. *Request from Madrid, Spain, in metric units and Spanish.*

```
“https://api.openweathermap.org/data/2.5/weather?q=Madrid,ES&units=metric&lang=es&appid=[API_KEY]”
```

Response snippet:

```
"weather": [
   {
     "id": 803,
     "main": "Clouds",
     "description": "muy nubloso", // "Very cloudy" in Spanish
     "icon": "04n"
   }
```

Steps to test:

1. Open a **new browser tab**.
2. Paste one of the API call examples.
3. Replace **[API_KEY]** with the generated key.
4. Press **Enter**.

2.2 Real-Time hourly forecast data (/forecast)

Objective:

Fetch 5-day weather forecast data for any global location using OpenWeatherMap's API.

Requirements:

- Generated OpenWeatherMap API key (see Section 1).
- Latitude and longitude coordinates (Obtain via */weather* endpoint on Section 2.1)
- Web browser or API testing tool.

Parameters:

The OpenWeatherMap API allows you to fetch forecast data using the **latitude** and **longitude** of a location. These coordinates can be retrieved using the */weather* endpoint in **section 2.1**.

The following table lists the parameters for the */forecast* endpoint, including their type (required/optional) and purpose:

Parameter	Type	Description
lat	required	Latitude of the location
lon	required	Longitude of the location.
appid	required	Generated API key
mode	optional	Response format: *JSON*, *XML* and *HTML*. Default: *JSON*.
cnt	optional	Number of forecast timestamps (1 timestamp = 3 hours). Default numbers of timestamps will be 40 (5 days).
units	optional	Measurement system: *standard, metric* and *imperial*. Default parameter is: *standard*
lang	optional	Language code.

API Call examples:

1. *Basic request.*

```
https://api.openweathermap.org/data/2.5/forecast?lat=[Latitude]&lon=[Longitude]&appid=[API_KEY]
```

2. *Basic Forecast for Tokyo, Japan.*

```
https://api.openweathermap.org/data/2.5/forecast?lat=35.6895&lon=139.6917&appid
```

```
=[API_KEY]
```

Response snippet:

```
{
  "cod": "200",
  "message": 0,
  "cnt": 2,
  "list": [
"dt": 1744513200,
      "main": {
        "temp": 285.24,
},
      "weather": [
        {
          "description": "light rain"
        }
      ]
    },
    // ... 39 more timestamps
  ]
}
```

3. *Short Forecast for Paris, France.*

```
https://api.openweathermap.org/data/2.5/forecast?lat=48.8534&lon=2.3488&cnt=2&appid=[API_KEY]
```

Response snippet:

```
{
  "cod": "200",
  "cnt": 2,
  "list": [
    {
      "dt": 1744513200,
      "main": {
        "temp": 285.13,
      },
      "weather": [
        {
          "description": "overcast clouds"
        }
      ]
    },
    {
      "dt": 1744524000,
      "main": {
        "temp": 284.39
      }
    }
  ]
}
```

Steps to test:

1. Open a new browser tab.
2. Paste one of the API call examples.
3. Replace **[Latitude]** and **[Longitude]** with coordinates from **Section 2.1**.
4. Replace **[API_KEY]** with the generated key
5. Press **Enter**.

Section 3: Parse Responses.

Objective:
Extract weather data from API responses and convert units for practical use.

Requirements:

- Generated OpenWeatherMap API key (see Section 1).
- Basic *Python*, *JavaScript* and *JSON* parsing knowledge.

Parameters:
API responses from **section 2** return nested *JSON* objects. The following table shows the location of data of **temperature**, **humidity** and **weather conditions**.

Field	Path	Description
Temperature	main.temp	Shown in Kelvin.
Conditions	weather[0].description	Summary of weather conditions.
Humidity	main.humidity	Shown in percentage.
Wind Speed	wind.speed	Shown in meters per second, in metric system.

3.1 Python Parsing Guide :

The following example summarizes the process of extracting data for Windows/macOS/Linux users with Python installed.

Setup:

1. Install the *requests* library:

```
pip install requests
```

Step-by-Step Code:

```
# Import libraries
import requests

# Step 1: Fetch data from API
API_KEY = "API_KEY"  # Replace with your key
CITY = "London"
try:
   response =
requests.get(     f"https://api.openweathermap.org/data/2.5/weather?q=[CITY]&appid
=[API_KEY]")
   data = response.json()
   # Step 2: Extract and convert Kelvin to Celsius.
   temp_kelvin = data["main"]["temp"]
   temp_celsius = round(temp_kelvin - 273.15, 2) # Formula:°C= K-273.15

   # Step 3: Extract other data
   humidity = data["main"]["humidity"]
   conditions = data["weather"][0]["description"].capitalize()

   # Step 4: Display results
   print(f"Weather in {CITY}:")
   print(f"- Temperature: {temp_celsius}°C")
   print(f"- Humidity: {humidity}%")
   print(f"- Conditions: {conditions}")

except Exception as e:
   print(f"Error: {e}")
```

Expected Output

```
Weather in London:
- Temperature: 18.50°C
- Humidity: 65%
- Conditions: Scattered clouds
```

3.2 JavaScript Parsing Guide :

The following example summarizes the process of extracting data for Node.js users with basic terminal knowledge.

Setup:

1. Create a project folder.
2. Open **PowerShell** on project folder, initialize *npm* and install *node-fetch*:

```
npm init -y
npm install node-fetch
```

Step-by-Step Code:

```
// Import libraries
import fetch from 'node-fetch';
```

```
// Step 1: Fetch data from API
const API_KEY = "API_KEY";  // Replace with your key
const CITY = "London";


const fetchWeather = async () => {
  try {
    const response = await
fetch(    `https://api.openweathermap.org/data/2.5/weather?q=$[CITY]&appid=$[API_
KEY]`
    );
    const data = await response.json();

    // Step 2: Extract and convert temperature
    const tempCelsius = (data.main.temp - 273.15).toFixed(2);

    // Step 3: Extract other data
    const humidity = data.main.humidity;
    const conditions = data.weather[0].description.toUpperCase();

    // Step 4: Display results
    console.log(`Weather in ${CITY}:`);
    console.log(`- Temperature: ${tempCelsius}°C`);
    console.log(`- Humidity: ${humidity}%`);
    console.log(`- Conditions: ${conditions}`);

  } catch (error) {
    console.error("Error:", error.message);
  }
};

// Run the script
fetchWeather();
```

Expected Output

```
Weather in London:
- Temperature: 18.50°C
- Humidity: 65%
```

- Conditions: Scattered clouds

Section 4: Handle Errors.

Objective:
Diagnose and resolve common API errors.

Requirements:
- Basic understanding of *HTTP* status codes.
- Access to API key **[Section 1]**

4.1 Authentication Errors (HTTP 401)

Causes:

- Invalid or missing API key.
- Key not activated.

Solutions:

1. *Regenerate Key*

Follow **[Section 1]** steps to create a new key.

4.2 City Not Found (HTTP 404)

Causes:

- Misspelled city name.
- Incorrect format.

Solutions:

1. *Incorrect format*

Always add **country code** to get the correct city.

2. *Using coordinates:*

If city names fail, use latitude and longitude from */weather*:

```
https://api.openweathermap.org/data/2.5/weather?lat=48.85&lon=2.35&appid=[API_KEY]
```

4.3 Rate-Limiting Errors (HTTP 429)

Causes:

- Exceeding **free tier** limits (60 calls/minute).
- *Burst* requests from multiple users.

Solutions:

1. *Add Delay Between Calls*

Wait 1 second between calls, the following is a *Python* example:

```python
import time
```

```python
def make_call(url):
    """Safely handles rate limits with 1.2s buffer"""
    time.sleep(1.2)  # 60 calls/min = 1 call/sec + 20% buffer
    return requests.get(url)
```

2. *Cache Responses*

The following is a JavaScript caching example:

```javascript
const cache = {};
async function getCachedWeather(city) {
   if (cache[city]) return cache[city];
   const data = await fetchWeather(city);
   cache[city] = data;
   return data;
}
```

3. *Upgrade Tier*

Consider paid plans for higher limits.

Section 5: Ensure Compliance

Objective:
Adhere to **OpenWeatherMap's** licensing terms and **GDPR** requirements.

Requirements:

- Access to **OpenWeatherMap** pricing page

5.1 GDPR & Data Licensing

Key requirements:

1. **OpenWeather`s** weather platform is fully GDPR-compliant.
2. Attribution is mandatory for free tier and optional for paid tiers.
3. You may process data internally and not resell raw API responses.

Attribution implementation:

1. *Web example:*

```
<footer>
 <small>
  Weather data by
  <a href="https://openweathermap.org/" target="_blank">OpenWeatherMap</a>
 </small>
</footer>
```

2. *Mobile (React Native) example:*

```
<Text style={styles.footer}>
 Powered by <Text style={styles.link} onPress={() =>
Linking.openURL('https://openweathermap.org')}>OpenWeatherMap</Text>
</Text>
```