

**Open Weather One Call**  
**Copyright 2020 J. Hershey**  
**v1.2.0**

Thank you for your interest in the Open Weather One call library. This library streamlines the ability to gather information from the OpenWeatherMap website. There are two items you will need to get started:

**OPENWEATHERMAP API KEY**  
(optional for geolocation via WiFi)  
**GOOGLE DEVELOPER API KEY**

You can get those at [www.openweathermap.org](http://www.openweathermap.org) and Google's developer website. Once you have those using the library is as simple as providing the calling function with the following items:

**OpenWeather Key, Google Key, Latitude, Longitude, Unit type, City ID, Excludes**

Calling the weather forecast you only need call:

**parseWeather(OpenWeather Key, Google Key, Latitude, longitude, Unit type, City ID, Excludes)**

OpenWeather Key is self-explanatory

Google Key is self-explanatory

Latitude and Longitude (if known, otherwise NULL) If you are using a GPS simply use those coordinates for Latitude and Longitude.

Unit type is a boolean (true/false/NULL) (METRIC, IMPERIAL, KELVIN)

CITY ID is now active in this release and can be used if known, otherwise NULL.

If an invalid coordinate is sent (out of range of worldwide measurements) geolocation will take over and gather your current location based on WiFi triangulation. Forcing an out of bounds Latitude or Longitude ensures geolocation will be used.

Sending an invalid Google Key (NULL) will ensure coordinates are used.

If none of those are valid a return error code of -1 will take place.

Exclude options are CURRENT, DAILY, HOURLY, MINUTELY, ALERTS

And they are entered in the API CALL as EXCL\_C+EXCL\_D+EXCL\_H,\_EXCL\_M+EXCL\_A  
You only need use the items you wish to exclude, this will save memory used in the parsing program by allocating only what is required for the incoming information.

As an example of what type of call to make:

```
parseWeather(OpenWeatherKey,NULL,Latitude,Longitude,true,NULL,EXCL_A+EXCL_M)
```

will use YOUR gps or other manually controlled coordinates to gather weather for THAT location and returns info in METRIC units, the data for Minutely and Alerts is also excluded from the generated JSON file at OpenWeatherMap.

So if you are in St. Louis, Missouri and you know the coordinates for Los Angeles, California, use those coordinates to get the weather in LA.

`parseWeather(OpenWeatherKey,NULL,NULL,NULL,true,city_id,NULL)` will use a CITY ID to gather weather for THAT location, or within 20 miles and returns info in METRIC units, no excludes will be sent so ALL information available will be returned.

So if you are in St. Louis, Missouri and you know the City ID for Los Angeles, California (5368361), use that to get the weather in L.A.

`parseWeather(OpenWeatherKey,GoogleKey,NULL,NULL,false,NULL,EXCL_C)` will use your current location based on WiFi triangulation and returns information in IMPERIAL units and exclude CURRENT weather information (but will send back hourly, minutely, daily, and alerts).

Including a Google Key in ANY call will return your current location weather regardless if you have sent latitude and longitude of another location, so be sure you are only using one or the other (Either a Google Key or City ID, or Latitude and Longitude) based on your needs. ALL three do NOT need to be used.

Many items are returned once a correct call is made and you will have access to the following items for CURRENT conditions:

1. temperature
2. apparentTemperature
3. pressure
4. humidity
5. dewPoint
6. uvIndex (midday number)
7. cloudCover
8. visibility

9. windSpeed
10. windBearing
11. windGust
12. Icon

Access these by using `current.variable` (example `current.windSpeed`)

For FUTURE forecasts you will have access to a 7 day forecast of the following items:

1. sunriseTime
2. sunsetTime
3. temperatureHigh
4. temperatureMin
5. temperatureMax
6. temperatureLow
7. apparentTemperatureHigh
8. apparentTemperatureLow
9. pressure
10. humidity
11. dewPoint
12. windSpeed
13. windBearing
14. precipType
15. icon
16. cloudCover
17. precipIntensity
18. uvIndex

Each day is accessible by using `forecast[x].variable` where x is 0-6 (Starting with TODAY-- >0).

MINUTELY supplies 60 minutes of info

1. Time
2. Precipitation

Use `minute.dt` or `minute.precipitation`

HOURLY supplies 48 hours of information with the following variables in form hour.variable

dt (date/time)  
temp  
feels\_like  
pressure  
humidity  
dew\_point  
clouds  
visibility  
wind\_speed  
wind\_deg  
id  
main  
description  
icon  
pop

As you can see, it couldn't be easier to get weather information for any day, any location up to 7 days into the future.

Please **examine the examples** available to see just how to send the required pieces of information to the library. As always, if you have any questions, the creator is available on GitHub. Thanks for your interest in the library!