3-Step Quickstart Guide

Feel like jumping right into making API calls? Get a free weatherstack account and simply use our 3-Step Quickstart Guide.

Go to Quckstart Guide

Code Examples

Our API supports all major programming languages. Click below to browse through examples in PHP, Python, Node, jQuery and Ruby.

Code Examples

- Getting Started
  API Authentication
  HTTPS Encryption
  API Error Codes
- API Features
  Current Weather
  Historical Weather
  Historical Time-Series
  Weather Forecast
  Location Lookup
- Options
  Query Parameter
  Units Parameter
  Language Parameter
  JSONP Callbacks
- Code Examples
  PHP
  Python
  Nodejs
  jQuery
  Go
  Ruby

  weatherstack on GitHub

API Documentation

Welcome to the weatherstack API documentation. Using the instructions and interactive code examples below you will be able to start making API requests in a matter of minutes. If you have an account already and prefer to skip our detailed documentation, you can also jump to our 3-Step Quickstart Guide right away.

The weatherstack API was built to deliver accurate weather data for any application and use case, from real-time and historical weather information all the way to 14-day weather forecasts, supporting all major programming languages. Our straightforward API design will make it easy to use the API — continue reading below to get started.

Getting Started

API Authentication

The first step to using the API is to authenticate with your weatherstack account's unique API access key, which can be found in your account dashboard after registration. To authenticate with the API, simply use the base URL below and pass your API access key to the API's `access_key` parameter.

Example API Request:

```
Run API Requesthttp://api.weatherstack.com/current
    ? access_key = a4f8725fbf5d45e20e0b1c57df8f4f51
    & query = New York
```

Keep it safe: Please make sure to keep your API access key and do not expose it in any publicly available part of your application. If you ever want to reset your key, simply head over to your account dashboard to do so.

256-bit HTTPS Encryption Available on: Standard Plan and higher

Clients using the Standard Plan or higher can connect to the weatherstack using industry-standard SSL (HTTPS) by attaching an `s` to the HTTP protocol as shown in the example API request below.

Example API Request:

```
https://api.weatherstack.com/...
```

If you are currently on the Free Plan and would like to use the HTTPS API in production, please upgrade your account now. You can also learn more about available plans on our Pricing Overview.

API Error Codes

Whenever an API request fails, the weatherstack API will return an error object in lightweight JSON format. Each error object contains an error code, an error type and an info object containing details about the error that occurred. Below you will find an example error as well as a list of common API errors.

Example API Error:

```
{
    "success": false,
    "error": {
        "code": 104,
        "type": "usage_limit_reached",
        "info": "Your monthly API request volume has been reached. Please upgrade your plan."
    }
}
```

Common API Errors:

Code	Type	Info
404	404_not_found	User requested a resource which does not exist.
101	missing_access_key	User did not supply an access key.
101	invalid_access_key	User supplied an invalid access key.
102	inactive_user	User account is inactive or blocked.
103	invalid_api_function	User requested a non-existent API function.
104	usage_limit_reached	User has reached his subscription's monthly request allowance.
105	function_access_restricted	The user's current subscription does not support this API function.
105	https_access_restricted	The user's current subscription plan does not support HTTPS.
601	missing_query	An invalid (or missing) query value was specified.
602	no_results	The API request did not return any results.
603	historical_queries_not_supported_on_plan	Historical data is not supported on the current subscription plan.
604	bulk_queries_not_supported_on_plan	Bulk queries is not supported on the current subscription plan.

Code	Type	Info
605	invalid_language	An invalid language code was specified.
606	invalid_unit	An invalid unit value was specified.
607	invalid_interval	An invalid interval value was specified.
608	invalid_forecast_days	An invalid forecast days value was specified.
609	forecast_days_not_supported_on_plan	Weather forecast data is not supported on the current subscription plan.
611	invalid_historical_date	An invalid historical date was specified.
612	invalid_historical_time_frame	An invalid historical time frame was specified.
613	historical_time_frame_too_long	The specified historical time frame is too long. (Maximum: 60 days)
614	missing_historical_date	An invalid historical date was specified.
615	request_failed	API request has failed.

API Features

Current Weather Available on: All plans

To query the weatherstack API for real-time weather data in a location of your choice, simply attach your preferred location to the API's `current` endpoint as seen in the example request below. Depending on your subscription, you can also make a bulk location request by passing multiple semicolon-separated locations to the API URL.

Example API Request:

```
Run API Requesthttp://api.weatherstack.com/current
    ? access_key = a4f8725fbf5d45e20e0b1c57df8f4f51
    & query = New York
```

HTTP GET Request Parameters:

Object	Description
access_key	**[Required]** Your API access key, which can be found in your acccount dashboard.
query	**[Required]** Use this parameter to pass a single location or multiple semicolon-separated location identifiers to the API. Learn more about the Query Parameter.
units	[Optional] Use this parameter to pass one of the unit identifiers ot the API: • `m` for Metric • `s` for Scientific • `f` for Fahrenheit Learn more about the Units Parameter.
language	[Optional] Use this parameter to specify your preferred API response language using its ISO-code. (Default: unset, English) Learn more about the Language Parameter.
callback	[Optional] Use this parameter to specify a JSONP callback function name to wrap your API response in. Learn more about JSONP Callbacks.

Example API Response:

The successful API request from the example above now returns real-time weather data for the city of New York, including detailed information about temperature, humidity, wind, clouds, pressure, the current time, a series of location identifiers, and more.


```
{
    "request": {
        "type": "City",
        "query": "New York, United States of America",
        "language": "en",
        "unit": "m"
    },
    "location": {
```

```
        "name": "New York",
        "country": "United States of America",
        "region": "New York",
        "lat": "40.714",
        "lon": "-74.006",
        "timezone_id": "America/New_York",
        "localtime": "2019-09-07 08:14",
        "localtime_epoch": 1567844040,
        "utc_offset": "-4.0"
    },
    "current": {
        "observation_time": "12:14 PM",
        "temperature": 13,
        "weather_code": 113,
        "weather_icons": [
            "https://assets.weatherstack.com/images/wsymbols01_png_64/wsymbol_0001_sunny.png"
        ],
        "weather_descriptions": [
            "Sunny"
        ],
        "wind_speed": 0,
        "wind_degree": 349,
        "wind_dir": "N",
        "pressure": 1010,
        "precip": 0,
        "humidity": 90,
        "cloudcover": 0,
        "feelslike": 13,
        "uv_index": 4,
        "visibility": 16
    }
}
```


API Response Objects:

Response Object	Description
`request` > `type`	Returns the type of location lookup used for this request. Possible values: • City • LatLon • IP • Zipcode
`request` > `query`	Returns the exact location identifier query used for this request.
`request` > `language`	Returns the ISO-Code of the language used for this request.
`request` > `unit`	Returns the unit identifier used for this request: • `m` for Metric • `s` for Scientific • `f` for Fahrenheit
`location` > `name`	Returns the name of the location used for this request.
`location` > `country`	Returns the country name associated with the location used for this request.
`location` > `region`	Returns the region name associated with the location used for this request.
`location` > `lat`	Returns the latitude coordinate associated with the location used for this request.
`location` > `lon`	Returns the longitude coordinate associated with the location used for this request.
`location` > `timezone_id`	Returns the timezone ID associated with the location used for this request. (Example: `America/New_York`)
`location` > `localtime`	Returns the local time of the location used for this request. (Example: `2019-09-07 08:14`)
`location` > `localtime_epoch`	Returns the local time (as UNIX timestamp) of the location used for this request. (Example: `1567844040`)

Response Object	Description
`location` > `utc_offset`	Returns the UTC offset (in hours) of the timezone associated with the location used for this request. (Example: `-4.0`)
`current` > `observation_time`	Returns the UTC time for when the returned whether data was collected.
`current` > `temperature`	Returns the temperature in the selected unit. (Default: Celsius)
`current` > `weather_code`	Returns the universal weather condition code associated with the current weather condition. You can download all available weather codes using this link: Download Weather Codes (ZIP file)
`current` > `weather_icons`	Returns one or more PNG weather icons associated with the current weather condition.
`current` > `weather_descriptions`	Returns one or more weather description texts associated with the current weather condition.
`current` > `wind_speed`	Returns the wind speed in the selected unit. (Default: kilometers/hour)
`current` > `wind_degree`	Returns the wind degree.
`current` > `wind_dir`	Returns the wind direction.
`current` > `pressure`	Returns the air pressure in the selected unit. (Default: MB - millibar)
`current` > `precip`	Returns the precipitation level in the selected unit. (Default: MM - millimeters)
`current` > `humidity`	Returns the air humidity level in percentage.
`current` > `cloudcover`	Returns the cloud cover level in percentage.
`current` > `feelslike`	Returns the "Feels Like" temperature in the selected unit. (Default: Celsius)
`current` > `uv_index`	Returns the UV index associated with the current weather condition.
`current` > `visibility`	Returns the visibility level in the selected unit. (Default: kilometers)

Historical Weather Available on: Standard Plan and higher

To look up historical weather data all the way back to 2008, simply pass one date of your choice (later than July 2008) or multiple semicolon-separated dates to the weatherstack API's `historical` endpoint using the `historical_date` parameter.

Example API Request:

```
Run API Requesthttp://api.weatherstack.com/historical
    ? access_key = a4f8725fbf5d45e20e0b1c57df8f4f51
    & query = New York
    & historical_date = 2015-01-21
    & hourly = 1
```

HTTP GET Request Parameters:

Object	Description
`access_key`	**[Required]** Your API access key, which can be found in your acccount dashboard.
`query`	**[Required]** Use this parameter to pass a single location or multiple semicolon-separated location identifiers to the API. Learn more about the Query Parameter.
`historical_date`	**[Required]** Use this parameter to pass one historical date or multiple semicolon-separated dates to the API. (Example: `2015-01-21` for a single date or `2015-01-21;2015-01-22` for multiple dates)
`hourly`	[Optional] Set this parameter to `1` (on) or `0` (off) depending on whether or not you want the API to return weather data split hourly. (Default: `0` - off)
`interval`	[Optional] If hourly data is enabled, use this parameter to define the interval: • `1` hour • `3` hourly (default) • `6` hourly • `12` hourly (day/night) • `24` hourly (day average)
`units`	[Optional] Use this parameter to pass one of the unit identifiers ot the API: • `m` for Metric • `s` for Scientific • `f` for Fahrenheit

Object	Description
	Learn more about the Units Parameter.
`language`	[Optional] Use this parameter to specify your preferred API response language using its ISO-code. (Default: unset, English) Learn more about the Language Parameter.
`callback`	[Optional] Use this parameter to specify a JSONP callback function name to wrap your API response in. Learn more about JSONP Callbacks.

Example API Response:

In addition to the requested historical weather data, a successful historical weather API call will also return the current weather in the location used for the request, as well as information about the API request and location.


```
{
    "request": {
        "type": "City",
        "query": "New York, United States of America",
        "language": "en",
        "unit": "m"
    },
    "location": {
        "name": "New York",
        "country": "United States of America",
        "region": "New York",
        "lat": "40.714",
        "lon": "-74.006",
        "timezone_id": "America/New_York",
        "localtime": "2019-09-07 10:05",
        "localtime_epoch": 1567850700,
        "utc_offset": "-4.0"
    },
    "current": {
        "observation_time": "02:05 PM",
        "temperature": 15,
        "weather_code": 113,
        "weather_icons": [
            "https://assets.weatherstack.com/images/wsymbols01_png_64/wsymbol_0001_sunny.png"
        ],
        "weather_descriptions": [
            "Sunny"
        ],
        "wind_speed": 0,
        "wind_degree": 0,
        "wind_dir": "N",
        "pressure": 1011,
        "precip": 0,
        "humidity": 78,
        "cloudcover": 0,
        "feelslike": 15,
        "uv_index": 5,
        "visibility": 16
    },
    "historical": {
        "2008-07-01": {
            "date": "2008-07-01",
            "date_epoch": 1214870400,
            "astro": {
                "sunrise": "05:29 AM",
                "sunset": "08:31 PM",
                "moonrise": "03:24 AM",
                "moonset": "07:37 PM",
                "moon_phase": "Waning Crescent",
                "moon_illumination": 4
            },
```

```
            "mintemp": 0,
            "maxtemp": 0,
            "avgtemp": 19,
            "totalsnow": 0,
            "sunhour": 14.5,
            "uv_index": 4,
            "hourly": [
                {
                    "time": "0",
                    "temperature": 27,
                    "wind_speed": 7,
                    "wind_degree": 201,
                    "wind_dir": "SSW",
                    "weather_code": 113,
                    "weather_icons": [
                        "https://assets.weatherstack.com/images/wsymbols01_png_64/wsymbol_0001_sunny.png"
                    ],
                    "weather_descriptions": [
                        "Sunny"
                    ],
                    "precip": 1.8,
                    "humidity": 80,
                    "visibility": 9,
                    "pressure": 1011,
                    "cloudcover": 15,
                    "heatindex": 25,
                    "dewpoint": 20,
                    "windchill": 24,
                    "windgust": 11,
                    "feelslike": 25,
                    "chanceofrain": 0,
                    "chanceofremdry": 0,
                    "chanceofwindy": 0,
                    "chanceofovercast": 0,
                    "chanceofsunshine": 0,
                    "chanceoffrost": 0,
                    "chanceofhightemp": 0,
                    "chanceoffog": 0,
                    "chanceofsnow": 0,
                    "chanceofthunder": 0,
                    "uv_index": 6
                },
                {   "time": "300", ...   },
                {   "time": "600", ...   },
                // 6 more items
            ]
        }
    }
}
```


Please note: The response objects `request`, `location` and `current` will not be explained below as they are already mentioned in the Current Weather Endpoint section above.

API Response Objects:

Response Object	Description
`historical` > `...` > `date`	Returns the requested historical date.
`historical` > `...` > `date_epoch`	Returns the requested historical date as UNIX timestamp.
`historical` > `...` > `astro`	Returns a total of 6 sub response objects containing astronomic weather details, listed and explained in detail below.
`astro` > `sunrise`	Returns the local sunrise time in the format `hh:mm am/pm`.

Response Object	Description
astro > sunset	Returns the local sunset time in the format hh:mm am/pm.
astro > moonrise	Returns the local moonrise time in the format hh:mm am/pm.
astro > moonset	Returns the local moonset time in the format hh:mm am/pm.
astro > moon_phase	Returns the local moon phase. Possible values: • New Moon • Waxing Crescent • First Quarter • Waxing Gibbous • Full Moon • Waning Gibbous • Last Quarter • Waning Crescent
astro > moon_illumination	Returns the moon illumination level as percentage.
historical > ... > mintemp	Returns the minimum temperature of the day in the selected unit. (Default: Celsius)
historical > ... > maxtemp	Returns the maximum temperature of the day in the selected unit. (Default: Celsius)
historical > ... > avgtemp	Returns the average temperature of the day in the selected unit. (Default: Celsius)
historical > ... > totalsnow	Returns the snow fall amount in the selected unit. (Default: Centimeters - cm)
historical > ... > sunhour	Returns the number of sun hours.
historical > ... > uv_index	Returns the UV index associated with the current weather condition.
historical > ... > hourly	Returns a series of sub response objects containing hourly weather data, listed and explained in detail below.
hourly > time	Returns the time as a number in 24h format: • 0 = 12:00 AM • 100 = 1:00 AM • 200 = 2:00 AM • ... • 1200 = 12:00 PM • 1300 = 1:00 PM • etc.
hourly > temperature	Returns the temperature in the selected unit. (Default: Celsius)
hourly > wind_speed	Returns the wind speed in the selected unit. (Default: kilometers/hour)
hourly > wind_degree	Returns the wind degree.
hourly > wind_dir	Returns the wind direction.
hourly > weather_code	Returns the universal weather condition code associated with the current weather condition. You can download all available weather codes using this link: Download Weather Codes (ZIP file)
hourly > weather_icons	Returns one or more PNG weather icons associated with the current weather condition.
hourly > weather_descriptions	Returns one or more weather description texts associated with the current weather condition.
hourly > precip	Returns the precipitation level in the selected unit. (Default: MM - millimeters)
hourly > humidity	Returns the air humidity level in percentage.
hourly > visibility	Returns the visibility level in the selected unit. (Default: kilometers)
hourly > pressure	Returns the air pressure in the selected unit. (Default: MB - millibar)
hourly > cloudcover	Returns the cloud cover level in percentage.
hourly > heatindex	Returns the heat index temperature in the selected unit. (Default: Celsius)
hourly > dewpoint	Returns the dew point temperature in the selected unit. (Default: Celsius)
hourly > windchill	Returns the wind chill temperature in the selected unit. (Default: Celsius)
hourly > windgust	Returns the wind gust speed in the selected unit. (Default: kilometers/hour)
hourly > feelslike	Returns the "Feels Like" temperature in the selected unit. (Default: Celsius)

Response Object	Description
`hourly` > `chanceofrain`	Retuns the chance of rain (precipitation) in percentage.
`hourly` > `chanceofremdry`	Retuns the chance of remaining dry in percentage.
`hourly` > `chanceofwindy`	Retuns the chance of being windy in percentage.
`hourly` > `chanceofovercast`	Retuns the chance of being overcast in percentage.
`hourly` > `chanceofsunshine`	Retuns the chance of sunshine in percentage.
`hourly` > `chanceoffrost`	Retuns the chance of frost in percentage.
`hourly` > `chanceofhightemp`	Retuns the chance of high temperatures in percentage.
`hourly` > `chanceoffog`	Retuns the chance of fog in percentage.
`hourly` > `chanceofsnow`	Retuns the chance of snow in percentage.
`hourly` > `chanceofthunder`	Retuns the chance of thunder in percentage.
`hourly` > `uv_index`	Returns the UV index associated with the current weather condition.

Historical Time-Series Available on: Standard Plan and higher

In addition to looking up historical weather data for specific dates, the API is also capable of processing historical time-series results if the parameters `historical_date_start` and `historical_date_end` are set to valid dates.

Example API Request:

```
Run API Requesthttp://api.weatherstack.com/historical
    ? access_key = a4f8725fbf5d45e20e0b1c57df8f4f51
    & query = New York
    & historical_date_start = 2015-01-21
    & historical_date_end = 2015-01-25
```

HTTP GET Request Parameters:

Object	Description
`access_key`	**[Required]** Your API access key, which can be found in your acccount dashboard.
`query`	**[Required]** Use this parameter to pass a single location or multiple semicolon-separated location identifiers to the API. Learn more about the Query Parameter.
`historical_date_start`	**[Required]** Use this parameter to pass a start date for the current historical time-series request.
`historical_date_end`	**[Required]** Use this parameter to pass an end date for the current historical time-series request.
`hourly`	[Optional] Set this parameter to `1` (on) or `0` (off) depending on whether or not you want the API to return weather data split hourly. (Default: `0` - off)
`interval`	[Optional] If hourly data is enabled, use this parameter to define the interval: • `1` hour • `3` hourly (default) • `6` hourly • `12` hourly (day/night) • `24` hourly (day average)
`units`	[Optional] Use this parameter to pass one of the unit identifiers ot the API: • `m` for Metric • `s` for Scientific • `f` for Fahrenheit Learn more about the Units Parameter.
`language`	[Optional] Use this parameter to specify your preferred API response language using its ISO-code. (Default: unset, English) Learn more about the Language Parameter.
`callback`	[Optional] Use this parameter to specify a JSONP callback function name to wrap your API response in. Learn more about JSONP Callbacks.

API Response:

For more information about the API response, please refer to the API response in the Historical Weather section above.

Please note: The historical time-series can accept a maximum timeframe of 60 days.

Note: Each day and location included in your request will count towards your monthly allowed API request volume.

Weather Forecast Available on: Professional Plan and higher

The weatherstack is capable of returning weather forecast data for up to 14 days into the future. To get weather forecasts, simply use the API's `forecast` and define your preferred number of forecast days using the `forecast_days` parameter.

Example API Request:

```
Run API Requesthttp://api.weatherstack.com/forecast
    ? access_key = a4f8725fbf5d45e20e0b1c57df8f4f51
    & query = New York
    & forecast_days = 1
    & hourly = 1
```

HTTP GET Request Parameters:

Object	Description
`access_key`	**[Required]** Your API access key, which can be found in your acccount dashboard.
`query`	**[Required]** Use this parameter to pass a single location or multiple semicolon-separated location identifiers to the API. Learn more about the Query Parameter.
`forecast_days`	[Optional] Use this parameter to specify the number of days for which the API returns forecast data. (Default: 7 or 14 days, depending on your subscription)
`hourly`	[Optional] Set this parameter to `1` (on) or `0` (off) depending on whether or not you want the API to return weather data split hourly. (Default: `0` - off)
`interval`	[Optional] If hourly data is enabled, use this parameter to define the interval: • `1` hour • `3` hourly (default) • `6` hourly • `12` hourly (day/night) • `24` hourly (day average)
`units`	[Optional] Use this parameter to pass one of the unit identifiers ot the API: • `m` for Metric • `s` for Scientific • `f` for Fahrenheit Learn more about the Units Parameter.
`language`	[Optional] Use this parameter to specify your preferred API response language using its ISO-code. (Default: unset, English) Learn more about the Language Parameter.
`callback`	[Optional] Use this parameter to specify a JSONP callback function name to wrap your API response in. Learn more about JSONP Callbacks.

Note: Each day and location included in your request will count towards your monthly allowed API request volume.

API Response:


```
{
    "request": {
        "type": "City",
        "query": "New York, United States of America",
        "language": "en",
        "unit": "m"
    },
    "location": {
```

```
        "name": "New York",
        "country": "United States of America",
        "region": "New York",
        "lat": "40.714",
        "lon": "-74.006",
        "timezone_id": "America/New_York",
        "localtime": "2019-09-07 11:38",
        "localtime_epoch": 1567856280,
        "utc_offset": "-4.0"
    },
    "current": {
        "observation_time": "03:38 PM",
        "temperature": 18,
        "weather_code": 113,
        "weather_icons": [
            "https://assets.weatherstack.com/images/wsymbols01_png_64/wsymbol_0001_sunny.png"
        ],
        "weather_descriptions": [
            "Sunny"
        ],
        "wind_speed": 0,
        "wind_degree": 345,
        "wind_dir": "NNW",
        "pressure": 1011,
        "precip": 0,
        "humidity": 58,
        "cloudcover": 0,
        "feelslike": 18,
        "uv_index": 5,
        "visibility": 16
    },
    "forecast": {
        "2019-09-07": {
            "date": "2019-09-07",
            "date_epoch": 1567814400,
            "astro": {
                "sunrise": "06:28 AM",
                "sunset": "07:19 PM",
                "moonrise": "03:33 PM",
                "moonset": "12:17 AM",
                "moon_phase": "First Quarter",
                "moon_illumination": 54
            },
            "mintemp": 17,
            "maxtemp": 25,
            "avgtemp": 21,
            "totalsnow": 0,
            "sunhour": 10.3,
            "uv_index": 5,
            "hourly": [
                {
                    "time": "0",
                    "temperature": 18,
                    "wind_speed": 28,
                    "wind_degree": 15,
                    "wind_dir": "NNE",
                    "weather_code": 122,
                    "weather_icons": [
                        "https://assets.weatherstack.com/images/wsymbols01_png_64/wsymbol_0004_black_low_cloud.png"
                    ],
                    "weather_descriptions": [
                        "Overcast"
                    ],
                    "precip": 0,
                    "humidity": 68,
```

```
                    "visibility": 10,
                    "pressure": 1008,
                    "cloudcover": 75,
                    "heatindex": 18,
                    "dewpoint": 12,
                    "windchill": 18,
                    "windgust": 35,
                    "feelslike": 18,
                    "chanceofrain": 0,
                    "chanceofremdry": 87,
                    "chanceofwindy": 0,
                    "chanceofovercast": 90,
                    "chanceofsunshine": 15,
                    "chanceoffrost": 0,
                    "chanceofhightemp": 0,
                    "chanceoffog": 0,
                    "chanceofsnow": 0,
                    "chanceofthunder": 0,
                    "uv_index": 0
                },
                {   "time": "300", ...   },
                {   "time": "600", ...   },
                // 6 more items
            ]
        }
    }
}
```


Please note: The response objects `request`, `location` and `current` will not be explained below as they are already mentioned in the Current Weather Endpoint section.

API Response Objects:

Response Object	Description
`forecast` > `...` > `date`	Returns the requested forecast date.
`forecast` > `...` > `date_epoch`	Returns the requested forecast date as UNIX timestamp.
`forecast` > `...` > `astro`	Returns a total of 6 sub response objects containing astronomic weather details, listed and explained in detail below.
`astro` > `sunrise`	Returns the local sunrise time in the format `hh:mm am/pm`.
`astro` > `sunset`	Returns the local sunset time in the format `hh:mm am/pm`.
`astro` > `moonrise`	Returns the local moonrise time in the format `hh:mm am/pm`.
`astro` > `moonset`	Returns the local moonset time in the format `hh:mm am/pm`.
`astro` > `moon_phase`	Returns the local moon phase. Possible values: • New Moon • Waxing Crescent • First Quarter • Waxing Gibbous • Full Moon • Waning Gibbous • Last Quarter • Waning Crescent
`astro` > `moon_illumination`	Returns the moon illumination level as percentage.
`forecast` > `...` > `mintemp`	Returns the minimum temperature of the day in the selected unit. (Default: Celsius)
`forecast` > `...` > `maxtemp`	Returns the maximum temperature of the day in the selected unit. (Default: Celsius)
`forecast` > `...` > `avgtemp`	Returns the average temperature of the day in the selected unit. (Default: Celsius)
`forecast` > `...` > `totalsnow`	Returns the snow fall amount in the selected unit. (Default: Centimeters - cm)
`forecast` > `...` > `sunhour`	Returns the number of sun hours.

Response Object	Description
forecast > ... > uv_index	Returns the UV index associated with the current weather condition.
forecast > ... > hourly	Returns a series of sub response objects containing hourly weather data, listed and explained in detail below.
hourly > time	Returns the time as a number in 24h format: • 0 = 12:00 AM • 100 = 1:00 AM • 200 = 2:00 AM • ... • 1200 = 12:00 PM • 1300 = 1:00 PM • etc.
hourly > temperature	Returns the temperature in the selected unit. (Default: Celsius)
hourly > wind_speed	Returns the wind speed in the selected unit. (Default: kilometers/hour)
hourly > wind_degree	Returns the wind degree.
hourly > wind_dir	Returns the wind direction.
hourly > weather_code	Returns the universal weather condition code associated with the current weather condition. You can download all available weather codes using this link: Download Weather Codes (ZIP file)
hourly > weather_icons	Returns one or more PNG weather icons associated with the current weather condition.
hourly > weather_descriptions	Returns one or more weather description texts associated with the current weather condition.
hourly > precip	Returns the precipitation level in the selected unit. (Default: MM - millimeters)
hourly > humidity	Returns the air humidity level in percentage.
hourly > visibility	Returns the visibility level in the selected unit. (Default: kilometers)
hourly > pressure	Returns the air pressure in the selected unit. (Default: MB - millibar)
hourly > cloudcover	Returns the cloud cover level in percentage.
hourly > heatindex	Returns the heat index temperature in the selected unit. (Default: Celsius)
hourly > dewpoint	Returns the dew point temperature in the selected unit. (Default: Celsius)
hourly > windchill	Returns the wind chill temperature in the selected unit. (Default: Celsius)
hourly > windgust	Returns the wind gust speed in the selected unit. (Default: kilometers/hour)
hourly > feelslike	Returns the "Feels Like" temperature in the selected unit. (Default: Celsius)
hourly > chanceofrain	Retuns the chance of rain (precipitation) in percentage.
hourly > chanceofremdry	Retuns the chance of remaining dry in percentage.
hourly > chanceofwindy	Retuns the chance of being windy in percentage.
hourly > chanceofovercast	Retuns the chance of being overcast in percentage.
hourly > chanceofsunshine	Retuns the chance of sunshine in percentage.
hourly > chanceoffrost	Retuns the chance of frost in percentage.
hourly > chanceofhightemp	Retuns the chance of high temperatures in percentage.
hourly > chanceoffog	Retuns the chance of fog in percentage.
hourly > chanceofsnow	Retuns the chance of snow in percentage.
hourly > chanceofthunder	Retuns the chance of thunder in percentage.
hourly > uv_index	Returns the UV index associated with the current weather condition.

Location Lookup/Autocomplete Available on: Standard Plan and higher

The weatherstack API's location `autocomplete` endpoint can be used to pinpoint one or more specific locations and their identifying response objects with the aim of later passing them to a weather data endpoint. In our example below, we are looking for London, United Kingdom.

Example API Request:

```
Run API Requesthttp://api.weatherstack.com/autocomplete
    ? access_key = a4f8725fbf5d45e20e0b1c57df8f4f51
    & query = london
```

HTTP GET Request Parameters:

Object	Description
`access_key`	**[Required]** Your API access key, which can be found in your acccount dashboard.
`query`	**[Required]** Use this parameter to pass your location search/autocomplete query to the API in free-text.
`callback`	[Optional] Use this parameter to specify a JSONP callback function name to wrap your API response in. Learn more about JSONP Callbacks.

API Response:

A successful API request will return one or multiple results that match your search query. In our example, the first array object (London, United Kingdom) contains the correct result. Now that we have our identifying response objects of the target location we were looking for, we can make sure that the correct location is used by other API endpoints by using one of the available location identifiers (ideally: `lat` and `lon`) for upcoming queries.


```
{
    "request": {
        "query": "london",
        "results": 2
    },
    "results": [
        {
            "name": "London",
            "country": "United Kingdom",
            "region": "City of London, Greater London",
            "lon": "-0.106",
            "lat": "51.517",
            "timezone_id": "Europe/London",
            "utc_offset": "1.0"
        },
        {
            "name": "London",
            "country": "Canada",
            "region": "Ontario",
            "lon": "-81.250",
            "lat": "42.983",
            "timezone_id": "America/Toronto",
            "utc_offset": "-4.0"
        }
    ]
}
```


API Response Objects:

Response Object	Description
`request` > `query`	Returns the exact query sent to the API.
`request` > `results`	Returns the number of results found as an integer.
`results` > `name`	Returns the name of the resulting city.
`results` > `country`	Returns the associated country.
`results` > `region`	Returns the name of the resulting region/state/district.
`results` > `lon`	Returns the longitude coordinates of the resulting location.
`results` > `lat`	Returns the latitude coordinates of the resulting location.
`results` > `timezone_id`	Returns the timezone ID associated with the resulting location. (Example: `America/New_York`)
`results` > `utc_offset`	Returns the UTC offset (in hours) of the timezone associated with the resulting location. (Example: `-4.0`)

General Options

Query Parameter Available on: All plans

The API's `query` parameter can be used in various ways to pass a single location or multiple locations to the API when making requests to one of the API's weather data endpoints. Using the example of the API's `current` endpoint, all available options will be outlined below:

Single Location:

The most common use case for the `query` parameter is to pass a single location to the API when making a weather data request. Find an example below:

```
Run API Requesthttp://api.weatherstack.com/current
    ? access_key = a4f8725fbf5d45e20e0b1c57df8f4f51
    & query = London, United Kingdom
```

Multiple Locations: (Professional Plan and higher)

To make use of the API's bulk query capability, you can also pass multiple semicolon-separated locations to the API:

```
Run API Requesthttp://api.weatherstack.com/current
    ? access_key = a4f8725fbf5d45e20e0b1c57df8f4f51
    & query = London;Singapur;Shanghai
```

Supported Location Identifiers:

Aside from simply passing the name of a city, there are multiple other ways of passing a location to the API:

Definition	Example	Description
Location Name	`query = New York`	The standard way of passing a location name to the API.
UK/Canada/US ZIP Code	`query = 99501`	Pass a UK/Canada/US ZIP code to the API and auto-detect the associated location.
Coordinates (Lat/Lon)	`query = 40.7831,-73.9712`	Pass latitude and longitude coordinates to the API and auto-detect the associated location.
IP Address	`query = 153.65.8.20`	Pass an IP address to the API and auto-detect the associated location.
IP Address (Auto-Fetch)	`query = fetch:ip`	Pass `fetch:ip` to the API in order to auto-detect the requester IP address and location.

Note: Each location passed to this parameter as part of a bulk query will count towards your monthly request volume.

Units Parameter Available on: All plans

By default, the API will return all results in metric units. Aside from metric units, other common unit formats are supported as well. You can use the `units` parameter to switch between the different unit formats Metric, Scientific and Fahrenheit.

`m` for Metric:

Parameter	Units
`units = m`	temperature: Celsius
`units = m`	Wind Speed/Visibility: Kilometers/Hour
`units = m`	Pressure: MB - Millibar
`units = m`	Precip: MM - Millimeters
`units = m`	Total Snow: CM - Centimeters

`s` for Scientific:

Parameter	Units
`units = s`	temperature: Kelvin
`units = s`	Wind Speed/Visibility: Kilometers/Hour
`units = s`	Pressure: MB - Millibar

Parameter	Units
`units = s`	Precip: MM - Millimeters
`units = s`	Total Snow: CM - Centimeters

`f` for Fahrenheit:

Parameter	Units
`units = f`	temperature: Fahrenheit
`units = f`	Wind Speed/Visibility: Miles/Hour
`units = f`	Pressure: MB - Millibar
`units = f`	Precip: IN - Inches
`units = f`	Total Snow: IN - Inches

Language Parameter Available on: Professional Plan and higher

The API is capable of delivering results in a total of 40 world languages. To change the default value (English) to another language, simply attach the `language` parameter to your API URL and set it to the 2-letter ISO Code of your preferred language.

Supported Languages:

Parameter (ISO Code)	Language
`language = ar`	Arabic
`language = bn`	Bengali
`language = bg`	Bulgarian
`language = zh`	Chinese Simplified
`language = zh_tw`	Chinese Traditional
`language = cs`	Czech
`language = da`	Danish
`language = nl`	Dutch
`language = fi`	Finnish
`language = fr`	French
`language = de`	German
`language = el`	Greek
`language = hi`	Hindi
`language = hu`	Hungarian
`language = it`	Italian
`language = ja`	Japanese
`language = jv`	Javanese
`language = ko`	Korean
`language = zh_cmn`	Mandarin
`language = mr`	Marathi
`language = pl`	Polish
`language = pt`	Portuguese
`language = pa`	Punjabi
`language = ro`	Romanian
`language = ru`	Russian
`language = sr`	Serbian
`language = si`	Sinhalese
`language = sk`	Slovak
`language = es`	Spanish
`language = sv`	Swedish
`language = ta`	Tamil
`language = te`	Telugu
`language = tr`	Turkish

Parameter (ISO Code)	Language
`language = uk`	Ukrainian
`language = ur`	Urdu
`language = vi`	Vietnamese
`language = zh_wuu`	Wu (Shanghainese)
`language = zh_hsn`	Xiang
`language = zh_yue`	Yue (Cantonese)
`language = zu`	Zulu

JSONP Callbacks Available on: All plans

The API supports JSONP Callbacks. To make use of this feature, simply append the API's `callback` parameter to your API request URL and set it to your preferred function name. The API will then return your API results set wrapped inside the tags of the function you specified.

Example API Request:

```
Run API Requesthttp://api.weatherstack.com/current
    ? access_key = a4f8725fbf5d45e20e0b1c57df8f4f51
    & query = New York
    & callback = FUNCTION_NAME
```

Example API Response:

```
CALLBACK_FUNCTION ({
    {
    "request": {
        "type": "City",
        "query": "New York, United States of America",
        "language": "en",
        [...]
    }
})
```

Please note: The API also supports Access-Control (CORS) headers.

Code Examples

Find below a series of straightforward code examples in different programming languages, all requesting and printing the latest available weather.

Code Example - PHP

```php
$location = 'New York';

$queryString = http_build_query([
  'access_key' => 'a4f8725fbf5d45e20e0b1c57df8f4f51',
  'query' => $location,
]);

$ch = curl_init(sprintf('%s?%s', 'https://api.weatherstack.com/current', $queryString));
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);

$json = curl_exec($ch);
curl_close($ch);

$api_result = json_decode($json, true);
```

```
echo "Current temperature in $location is {$api_result['current']['temperature']}℃", PHP_EOL;
```

Code Example - Python

```python
# coding: utf-8
import requests

params = {
  'access_key': 'a4f8725fbf5d45e20e0b1c57df8f4f51',
  'query': 'New York'
}

api_result = requests.get('https://api.weatherstack.com/current', params)

api_response = api_result.json()

print(u'Current temperature in %s is %d℃' % (api_response['location']['name'], api_response['current']['temperature']))
```

Code Example - Nodejs

```nodejs
const axios = require('axios');
const params = {
  access_key: 'a4f8725fbf5d45e20e0b1c57df8f4f51',
  query: 'New York'
}

axios.get('https://api.weatherstack.com/current', {params})
  .then(response => {
    const apiResponse = response.data;
    console.log(`Current temperature in ${apiResponse.location.name} is ${apiResponse.current.temperature}℃`);
  }).catch(error => {
    console.log(error);
  });
```

Code Example - jQuery

```jquery
$.ajax({
  url: 'https://api.weatherstack.com/current',
  data: {
    access_key: 'a4f8725fbf5d45e20e0b1c57df8f4f51',
    query: 'New York'
  },
  dataType: 'json',
  success: function(apiResponse) {
    console.log(`Current temperature in ${apiResponse.location.name} is ${apiResponse.current.temperature}℃`);
  }
});
```

Code Example - Go

```go
package main

import (
  "encoding/json"
  "fmt"
```

```
  "net/http"
)

type Location struct {
  Name string `json:"name"`
}

type Weather struct {
  Temperature int `json:"temperature"`
}

type Response struct {
        Location Location `json:"location"`
        Current Weather `json:"current"`
}

func main() {
  httpClient := http.Client{}

  req, err := http.NewRequest("GET", "https://api.weatherstack.com/current", nil)
  if err != nil {
    panic(err)
  }

  q := req.URL.Query()
  q.Add("access_key", "a4f8725fbf5d45e20e0b1c57df8f4f51")
  q.Add("query", "New York")
  req.URL.RawQuery = q.Encode()

  res, err := httpClient.Do(req)
  if err != nil {
    panic(err)
  }
  defer res.Body.Close()

  var apiResponse Response
  json.NewDecoder(res.Body).Decode(&apiResponse)

  fmt.Println(fmt.Sprintf("Current temperature in %s is %d℃", apiResponse.Location.Name, apiResponse.Current.Temperature))
}
```

Code Example - Ruby

```
require 'net/http'
require 'json'

params = {
  :access_key => "a4f8725fbf5d45e20e0b1c57df8f4f51",
  :query => "New York"
}
uri = URI('https://api.weatherstack.com/current')
uri.query = URI.encode_www_form(params)
json = Net::HTTP.get(uri)
api_response = JSON.parse(json)

puts "Current temperature in #{api_response['location']['name']} is #{api_response['current']['temperature']}"
```

Any technical questions left? Reach out to us, our team is happy to help. Contact Us

weatherstack

- PRODUCT
- Pricing
- List Your API
- Why choose us?
- Marketplace

- DOCS & HELP
- Documentation
- Blog
- FAQs
- Press
- API Glossary

- ACCOUNT
- Log In
- Forgot Password
- FREE Signup

- GET IN TOUCH
- Contact
- Get a quote

Imprint / Legal | Privacy | Terms | Cookie Preference | Sitemap