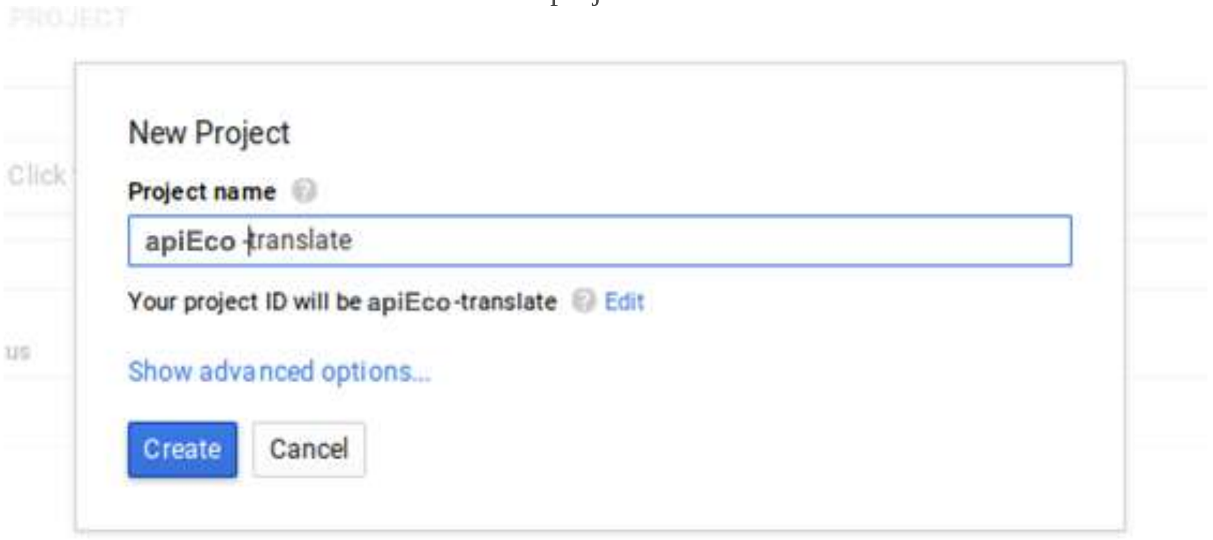
Google Translate Package

This package allows you to translates a string of text from one language to another.

How to get `apiKey`:

1. Go to the projects page
2. Select or create a Cloud Platform Console project.



New Project

Project name

apiEco translate

Your project ID will be apiEco-translate Edit

3. Enable billing for your project.
4. Click ***Continue*** to enable the Translate API and any related services.
5. On the ***Credentials*** page, get an ***API key*** (select Browser key when prompted). *Note: If you have an existing API key, you can use that key.*
6. Go to the translate api overview and press the "Enable" button.

GoogleTranslate.translate

Translate `string` from `sourceLanguage` to `targetLanguage`

Field	Type	Description
`apiKey`	string	Required: The api key obtained from Google Cloud.
`string`	string	Required: The string to translate.
`targetLanguage`	string	Required: The ISO 639-1 language code to translate the input to. All supported languages
`sourceLanguage`	string	The ISO 639-1 language code the source input is written in. All supported languages

Request example

```
{
    "apiKey": "XXXXXXX",
    "string": "Hello",
    "sourceLanguage": "en",
    "targetLanguage": "es"
}
```

Response example

```
{
    "callback": "success",
    "contextWrites": {
        "#": {
            "to": "Hola"
        }
    }
}
```

GoogleTranslate.translateAutomatic

Translate `string` from `sourceLanguage` to target language automatically

Field	Type	Description
`apiKey`	string	Required: The api key obtained from Google Cloud.
`string`	string	Required: The string to translate.
`targetLanguage`	string	Required: The ISO 639-1 language code to translate the input to. All supported languages

Request example

```
{
    "apiKey": "XXXXXXX",
    "string": "Hello",
    "targetLanguage": "es"
}
```

Response example

```
{
    "callback": "success",
    "contextWrites": {
        "#": {
            "to": "Hola"
        }
    }
}
```

GoogleTranslate.detectLanguage

Detect language of `string`

Field	Type	Description
`apiKey`	string	Required: The api key obtained from Google Cloud.
`string`	string	Required: The string to translate.

Request example

```
{
    "apiKey": "XXXXXXX",
    "string": "Hello",
}
```

Response example

```
{
    "callback": "success",
    "contextWrites": {
        "#": {
            "to": "en"
        }
    }
}
```

MuslimSalat

Introduction

Prayer times made available through api, so everyone can benefit the easy access for prayer times with less effort to develop custom applications for verity of software's / application's.

JSON

JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate.

All API data is served in json format.

JSONP

JSONP is a communication technique used in Javascript. It provides a method to request data from a server in a different domain, something prohibited by typical web browsers because of the same origin policy.

To use jsonp technique, just use callback or jsoncallback parameter on the url and it will call the javascript callback function for execution.

Example

```
jQuery(function($) {

        $.getJSON('https://api.apieco.ir/muslimsalat/london/daily.json?key=API_K
EY&jsoncallback=?', function (times)

        {

                $('.prayerTimesExample')

                .append('Today in '+times.title)

                .append(' Fajr: '+times.items[0].fajr)

                .append(' Dhuhr: '+times.items[0].dhuhr)

                .append(' Asr: '+times.items[0].asr)

                .append(' Maghrib: '+times.items[0].maghrib)

                .append(' Isha: '+times.items[0].isha)

                .append(' by api.apieco.ir');

        });

});
```

Output:

```
Today in Fajr: 3:54 am Dhuhr: 1:00 pm Asr: 4:48 pm Maghrib: 8:04 pm Isha: 9:57 pm by api.apieco.ir
```

Get My API Key

To get started with api request, you will need api key. Get your API KEY now.

Auto Location

This will fetch user prayer times with automatically selecting the location for user.

https://api.apieco.ir/muslimsalat/**daily**.json?key=api_key

You can get time in the following timeline.

- daily
- weekly
- monthly
- yearly

Location Based

This will fetch the prayer time based on the location you request for with the given parameters.

Prayer Times

Today prayer times for london:

https://api.apieco.ir/muslimsalat/**london**.json?key=api_key

Weekly prayer times for london starting from today:

https://api.apieco.ir/muslimsalat/**london**/**weekly**.json?key=api_key

Next date weekly prayer times for london.

https://api.apieco.ir/muslimsalat/**london**/**weekly**/**12-01-2013**.json?key=api_key

with daylight saying

https://api.apieco.ir/muslimsalat/**london**/**weekly**/**12-01-2013**/**true**.json?key=api_key

and with Muslim World League calculation method.

https://api.apieco.ir/muslimsalat/**london**/**weekly**/**12-01-2013**/**true**/**5**.json?key=api_key

URL Parameters

All following parameters are optional, you can set as you wish. If you wish to set two or more parameters, then you have to set url in the order the parameter is given below.

if you want to set location and date. You should do following
https://api.apieco.ir/muslimsalat/location/date.json?key=api_key

Parameters Properties and starting with the order for url

Name	**Values**	**Default value**	**Description**
Location	City, country, state...	Auto select for the user	Name of the location where user is at or his state name or his country name or with his latitude and longitude .
Times	Daily, weekly, monthly or yearly	Monthly	Limit the prayer times by the value.

Name	**Values**	**Default value**	**Description**
Date	12-02-2012	Today Date	Get the prayer times for the given date, please make sure the date is further head or current date. **Heads up!** Previous dates will be deprecated from the api.
Daylight saving	True or false	Auto select	Daylight saving for the user, if true then hours are incremented by 1+
Method	1 = Egyptian General Authority of Survey 2 = University Of Islamic Sciences, Karachi (Shafi) 3 = University Of Islamic Sciences, Karachi (Hanafi) 4 = Islamic Circle of North America 5 = Muslim World League 6 = Umm Al-Qura 7 = Fixed Isha	Auto select based on the country where user is from.	Method to use for calculation of the timing. If method is provided invalid based on the country, it will give incorrect timing.

Qibla Compass Image

Qibla compass images are based on given direction from the api response.

https://api.apieco.ir/muslimsalat/qibla_compass/200/118.82.png

URL Parameters

Parameters Properties and starting with the order for url

Name	**Values**	**Default value**	**Description**
Size	50 to 600 (optional)	300	Size of the image you want to show
Direction	0 to 360 degree		Direction degree from json response

stormglass

Introduction

The Storm Glass Marine Weather API allows you to fetch forecasts for any coordinate in a simple, programmatic way using conventional HTTP requests. When a request is successful, a response will be sent back in the form of a JSON object.

Authentication

Storm Glass uses API keys to allow access to the API. You can register for an API key in the dashboard.

Storm Glass expects the API key to be included in all API requests to the server in a header that looks like the following:

Authorization: example-api-key

You must replace `example-api-key` with your personal API key.

Date and time

At Storm Glass we work with dates and times in UTC timezone.

In query parameters

When sending a request to our API, timestamps in the following formats are accepted:

UNIX Timestamp

For example: 1542967200

URL Encoded ISO Formatted Timestamp

For example: 2018-11-23T10%3A00%3A00%2B00%3A00

In response

Date and time returned from our API will be expressed in ISO format:

ISO Formatted Timestamp

For example: 2018-11-23T10:00:00+00:00

Sources

Storm Glass provides data from several weather institutes around the world. Each source contains its' own set of attributes and the data is updated individually on different intervals.

Below you can find two tables overviewing the available data sources and what attributes to expect from each source.

Available sources

Name	Abbr	Resolution	Update Frequency	Forecast Span	Area	Description
NOAA Wavewatch 3	noaa	0.25°x1.0°	Every 6 hours	183 hours	Global - Except for smaller seas such as the Baltic Sea and the Mediterranean Sea	The National Oceanic and Atmospheric Administration wave model forecast
NOAA GFS	noaa	0.5°x0.5°	Daily	240 hours	Global	The National Oceanic and Atmospheric Administration global forecast system
Météo-France	meteo	0.5°x1.0°	Daily	120 hours	Global	French National Meteorological service,

Name	Abbr	Resolution	Update Frequency	Forecast Span	Area	Description
						Météo-France
Deutscher Wetterdienst	dwd	0.25°x0.25°	Every 12 hours	92 hours	The Atlantic coast of Ireland, UK, France, Spain and Portugal. The North Sea, Mediterranean Sea and Baltic Sea.	Germany's National Meteorological Service, the Deutscher Wetterdienst
UK MetOffice	meto	0.5°x1.0°	Daily	240 hours	Global currents and water temperature. Wave data for North Sea and North Atlantic	United Kingdom's national weather service, The UK MetOffice
FCOO	fcoo	0.5°x0.5°	Every 12 hours	48 hours	Baltic Sea including Gulf of Botnia and Gulf of Finland.	Danish Defence Centre for Operational Oceanography
FMI	fmi	0.25°x0.25°	Every 4 hours	55 hours	Baltic Sea including Gulf of Botnia and Gulf of Finland.	The Finnish Meteorological Institution

Name	Abbr	Resolution	Update Frequency	Forecast Span	Area	Description
YR	yr	0.05°x0.05°	Every 4 hours	73 hours	The Norwegian coast.	Norwegian Meteorological Institute and NRK
SMHI	smhi	0.5°x0.5°	Every 12 hours	240 hours	Baltic Sea including Gulf of Botnia and Gulf of Finland.	Swedish Meteorological and Hydrological Institute
Storm Glass	sg	0.25°x0.25°	Every 4 hours	168 hours	Global	Handpicked local forecast in each geographical area

We are continuously working on adding more sources and attributes to our service. Please contact support@stormglass.io if you have any specific requests.

Attributes per source

Attribute	noaa	meteo	dwd	meto	fcoo	fmi	yr	smhi	sg
airTemperature	✓		✓					✓	✓
airPressure	✓		✓				✓	✓	✓
cloudCover	✓		✓					✓	✓
currentDirection				✓					✓
currentSpeed				✓					✓

Attribute	noaa	meteo	dwd	meto	fcoo	fmi	yr	smhi	sg
gust	✓		✓					✓	✓
humidity	✓		✓					✓	✓
iceCover	✓								✓
precipitation	✓		✓					✓	✓
seaLevel									✓
snowDepth	✓								✓
swellDirection	✓	✓	✓	✓					✓
swellHeight	✓	✓	✓	✓					✓
swellPeriod	✓	✓	✓	✓					✓
visiblity	✓							✓	✓
waterTemperature	✓			✓					✓
waveDirection	✓	✓		✓	✓	✓	✓		✓
waveHeight	✓	✓	✓	✓	✓	✓	✓		✓
wavePeriod	✓	✓	✓	✓	✓	✓			✓
windWaveDirection	✓	✓	✓	✓					✓

Attribute	noaa	meteo	dwd	meto	fcoo	fmi	yr	smhi	sg
windWaveHeight	✓	✓	✓	✓					✓
windWavePeriod	✓	✓	✓	✓					✓
windDirection	✓		✓				✓	✓	✓
windSpeed	✓		✓				✓	✓	✓

Point Request

```python
import arrow
import requests

# Get first hour of today
start = arrow.now().floor('day')

# Get last hour of today
end = arrow.now().ceil('day')

response = requests.get(
  'https://api.apieco.ir/stormglass/point',
  params={
    'lat': 58.7984,
    'lng': 17.8081,
    'params': ','.join(['waveHeight', 'airTemperature']),
    'start': start.to('UTC').timestamp,  # Convert to UTC timestamp
    'end': end.to('UTC').timestamp  # Convert to UTC timestamp
  },
  headers={
    'Authorization': 'example-api-key'
  }
)

# Do something with response data.
json_data = response.json()
```

The above requests returns JSON structured like this:

```json
{
  "hours": [
    {
      "time": "2018-01-19T17:00:00+00:00",
      "airTemperature": [
        {
          "source": "smhi",
          "value": "-2.6"
        }
      ],
      "waveHeight": [
        {
          "source": "noaa",
          "value": 2.1
        },
        {
          "source": "meteo",
          "value": 2.3
        }
      ]
      ...
    }
  ],
  "meta": {
    "dailyQuota": 5,
    "lat": 58.7984,
    "lng": 17.8081,
    "requestCount": 2
  }
}
```

Point Requests are used to retrieve data for a single coordinate.

To get data for land and lakes – simply send in a coordinate on land or on a lake.

HTTP Request

GET https://api.apieco.ir/stormglass/point

Query Parameters

Parameter	Required	Default	Description
lat	✓	n/a	Latitude of the desired coordinate
lng	✓	n/a	Longitude of the desired coordinate
params		all (see list below)	Comma separeted list of params included in response, Eg `swellHeight,waveHeight`
start		Today at 00.00	Timestamp in UTC for first forecast hour - UNIX format or URL encoded ISO format.

Parameter	Required	Default	Description
end		all	Timestamp in UTC for last forecast hour - UNIX format or URL encoded ISO format.
source		all	Specify a single source. Eg `noaa` or `dwd`

Default params returned: `airTemperature, airPressure, humidity, cloudCover, currentDirection, currentSpeed, precipitation, visiblity, swellDirection, swellHeight, swellPeriod, waterTemperature, waveDirection, waveHeight, wavePeriod, windWaveDirection, windWaveHeight, windWavePeriod, seaLevel, windDirection, windSpeed, gust,`

Response format

The response will be sent back in the form of a JSON object. The resource root contains two objects, the *hours* and the *meta* keys.

Meta

The meta key contains information about the API request. Such as requested latitude and longitude, your daily quota and how many requests you've made so far today.

Hours

The hours key contains the actual weather data on an hourly basis. One item in the hours array contains:

key	value
time	Timestamp in UTC
airTemperature	Air temperature in degrees celsius
airPressure	Air pressure in hPa
cloudCover	Total cloud coverage in percent
currentDirection	Direction of current. 0° indicates current coming from north

key	value
currentSpeed	Speed of current in meters per second
gust	Wind gust in meters per second
humidity	Relative humidity in percent
iceCover	Proportion, 0-1
precipitation	Mean precipitation in kg/m²
seaLevel	Height of sea level in MLLW in meters (tide)
snowDepth	Depth of snow in meters
swellDirection	Direction of swell waves. 0° indicates swell coming from north
swellHeight	Height of swell waves in meters
swellPeriod	Period of swell waves in seconds
visiblity	Horizontal visibility in km
waterTemperature	Water temperature in degrees celsius
waveDirection	Direction of combined wind and swell waves. 0° indicates waves coming from north
waveHeight	Height of combined wind and swell waves in meters
wavePeriod	Period of combined wind and swell waves in seconds
windWaveDirection	Direction of wind waves. 0° indicates waves coming from north

key	value
windWaveHeight	Height of wind waves in meters
windWavePeriod	Period of wind waves in seconds
windDirection	Direction of wind. 0° indicates wind coming from north
windSpeed	Speed of wind in meters per second

Each parameter (eg. swellHeight) is a list that contains an object for each available source. The object consists of *source* and *value*.

You must replace `example-api-key` with your personal API key.

Area Request

Area requests are deducted from the daily quota included in your monthly subscription and are calculated as 0.5 x (the amount of points within the area). As an example a request for an area containing 100 points will equal to 50 single point requests.

```
import requests

response = requests.get(
  'https://api.apieco.ir/stormglass/area',
  params={
      'box': '60,20:58,17',
      'params': 'waveHeight,airTemperature'
  },
  headers={
    'Authorization': 'example-api-key'
  }
)

# Do something with response data.
json_data = response.json()
```

The above requests returns JSON structured like this:

```
{
  "grid": {
      "58": {
          "17": [
            {
                "time": "2018-09-27T17:00:00+00:00",
                "airTemperature": -2.6,
                "waveHeight": 2.3,
            },
            ...
          ],
          "18": [ ... ]
      },
      "59": { ... },
      ...
  },
  "meta": {
    "dailyQuota": 150,
    "cost": 10,
    "date: "2018-09-27",
    "box": [
        [60, 20],
        [58, 17]
    ],
    "requestCount": 12
  }
}
```

Retrieve forecast for an area described by either a box or a polygon.

Since the response for this request will have a larger payload than the request for a single point, there will only be one value for each attribute. This will be the value from the "most local" weather source that we have available.

HTTP Request

GET https://api.apieco.ir/stormglass/area

Query Parameters

Parameter	Required	Default	Description
box	*	n/a	Top right and bottom left coordinate of box on format: `lat,lng:lat,lng`
polygon	*	n/a	List of coordinates describing area. Will be connected in order left to right: `lat,lng:lat,lng:lat,lng`
date	✓	Today	Forecast date on format: `YYYY-MM-DD`
params		all	Comma separeted list of params to include in response. Eg `swellHeight,waveHeight`

Parameter	Required	Default	Description
step_size		1	Lenght between coordinates in grid. Valid values are `1`, `0.5` and `0.25`

* Either polygon or box must be defined

Response format

The response will be sent back in the form of a JSON object. The resource root contains two objects, the *grid* and the *meta* keys.

Meta

The meta key contains information about the API request. Such as requested latitude and longitude, your daily quota and how many requests you've made so far today.

Grid

The grid is an object consisting of latitudes with nested longitudes. Each longitude contains weather data on hourly basis. One hour can contain the following attributes:

key	value
time	Timestamp in UTC
airTemperature	Air temperature in degrees celsius
airPressure	Air pressure in hPa
cloudCover	Total cloud coverage in percent
currentDirection	Direction of current.
currentSpeed	Speed of current in meters per second.
gust	Wind gust in m/s
humidity	Relative humidity in percent

key	value
iceCover	Proportion, 0-1
precipitation	Mean precipitation in kg/m²
seaLevel	Height of sea level in MLLW (tides).
snowDepth	Depth of snow in meters
swellDirection	Direction of swell waves. 0° indicates swell coming from north
swellHeight	Height of swell waves in meters
swellPeriod	Period of swell waves in seconds
visiblity	Horizontal visibility in km
waterTemperature	Water temperature in degrees celsius
waveDirection	Direction of combined wind and swell waves. 0° indicates waves coming from north
waveHeight	Height of combined wind and swell waves
wavePeriod	Period of combined wind and swell waves
windWaveDirection	Direction of wind waves. 0° indicates waves coming from north
windWaveHeight	Height of wind waves
windWavePeriod	Period of wind waves
windDirection	Direction of wind. 0° indicates wind coming from north

key	value
windSpeed	Speed of wind in meters per second

You must replace `example-api-key` with your personal API key.

Astronomy Request

```
import arrow
import requests

# Get first hour of today
start = arrow.now().floor('day')

# Set how many days we want astronomy data for
numberOfDays = 7

response = requests.get(
  'https://api.apieco.ir/stormglass/astronomy',
  params={
    'lat': 58.7984,
    'lng': 17.8081,
    'start': start.to('UTC').timestamp,  # Convert to UTC timestamp
    'numberOfDays': numberOfDays
  },
  headers={
    'Authorization': 'example-api-key'
  }
)

# Do something with response data.
json_data = response.json()
```

The above requests returns JSON structured like this:

```
{
    "days": [
        {
            "astronomicalDawn": "2018-11-22T04:29:13+00:00",
            "astronomicalDusk": "2018-11-22T16:43:25+00:00",
            "civilDawn": "2018-11-22T06:07:58+00:00",
            "civilDusk": "2018-11-22T15:04:39+00:00",
            "moonFraction": 0.9773405348657047,
            "moonPhase": {
                "closest": {
                    "text": "Full moon",
                    "time": "2018-11-23T10:05:00+00:00",
                    "value": 0.5
                },
                "current": {
                    "text": "Waxing gibbous",
                    "time": "2018-11-22T00:00:00+00:00",
                    "value": 0.45190179144442527
                }
            },
            "moonrise": "2018-11-22T13:58:41.948883+00:00",
            "moonset": "2018-11-22T05:04:59.690726+00:00",
            "nauticalDawn": "2018-11-22T05:17:04+00:00",
            "nauticalDusk": "2018-11-22T15:55:34+00:00",
            "sunrise": "2018-11-22T06:56:32+00:00",
            "sunset": "2018-11-22T14:16:06+00:00",
            "time": "2018-11-22T00:00:00+00:00"
        },
        ...
    ],
    "meta": {
        "cost": 1,
        "dailyQuota": 50,
        "lat": 58.7984,
        "lng": 17.8081,
        "requestCount": 1,
        "start": "2018-11-22T00:00:00+00:00"
    }
}
```

Retrieve sunrise, sunset, moonrise, moonset and moon phase for a single coordinate.

HTTP Request

GET https://api.apieco.ir/stormglass/astronomy

Query Parameters

Parameter	Required	Default	Description
lat	✓	n/a	Latitude of the desired coordinate
lng	✓	n/a	Longitude of the desired coordinate
start		Today at 00.00	Timestamp in UTC for first forecast hour - UNIX format or URL encoded ISO format.
numberOfDays		1	For how many days ahead to receive data. max 31 days

Default params returned: `sunrise, sunset, moonrise, moonset, moonFraction, moonPhase, time`

Response format

The response will be sent back in the form of a JSON object. The resource root contains two objects, the *days* and the *meta* keys.

Meta

The meta key contains information about the API request. Such as requested latitude and longitude, your daily quota and how many requests you've made so far today.

Days

The days key contains the actual data on a daily basis. One item in the days array contains:

key	**value**
time	Timestamp in UTC indicating the day for the data
sunrise	Timestamp for sunrise in UTC. Will return null if no sunsrise occurs on the given day
sunset	Timestamp for sunset in UTC. Will return null if no sunset occurs on the given day
moonrise	Timestamp for moonrise in UTC. Will return null if no moonrise occurs on the given day
moonset	Timestamp for moonset in UTC. Will return null if no moonset occurs on the given day
moonFraction	A float number between 0 and 1 indicating how much of the moon is illuminated
moonPhase	Objects describing the current and the closest moon phase
astronomicalDawn	Timestamp in UTC. Will return null if no dawn occurs on the given day

key	value
astronomicalDusk	Timestamp in UTC. Will return null if no dusk occurs on the given day
civilDawn	Timestamp for sunset in UTC. Will return null if no dawn occurs on the given day
civilDusk	Timestamp for sunset in UTC. Will return null if no dusk occurs on the given day
nauticalDawn	Timestamp for sunset in UTC. Will return null if no dawn occurs on the given day
nauticalDusk	Timestamp for sunset in UTC. Will return null if no dusk occurs on the given day

A moon phase is described by an object with the structure according to the table below. **current** describes the current moon phase and **closest** gives you the timestamp for the closest phase being one of New moon, First quarter, Full moon or Third quarter.

key	value
time	Timestamp in UTC showing what time the moon phase object describes
text	A string describing the moon phase. The possible values are:`New moon`, `Waxing crescent`, `First quarter`, `Waxing gibbous`, `Full moon`, `Vaning gibbous`, `Third quarter`, `Vaning crescent`
value	A float value for the phase of the given time.

The **value** parameter gives you a float value for the given time where 0.0 or 1.0 equals **New moon**, 0.25 equals **First Quarter**, 0.5 equals **Full moon**and 0.75 equals **Third quarter**.

Astronomical Dawn occurs when the sun reaches 18° below the horizon, Nautical at 12° and Civil at 6°. The same degrees apply for the Dusk definitions.

Errors

The Storm Glass API uses the following response error codes:

Error Code	Meaning
400	Bad Request -- Your request is invalid.
401	Unauthorized -- Your API key is invalid.
429	Too Many Requests -- You've reached your daily limit.
500	Internal Server Error -- We had a problem with our server. Try again later.
503	Service Unavailable -- We're temporarily offline for maintenance. Please try again later.

clearip

Making your first request

Making a request for geolocation data with Clearip is pretty simple. Try running this in the command line:

```
curl
https://api.apieco.ir/clearip/ip/54.144.220.0/json?apikey=API_KEY
```

That should return a JSON object that maybe looks something like this:

```
{
"country": "United States",
"continent": "Americas",
"country_flag": "US",
"CountryCode": "US",
"City": "Ashburn",
"Region": "Virginia",
"lat": 39.0481,
"lng": -77.4728,
"tz": "America/New_York",
"isp": "Amazon",
"is_anonymous_proxy": false,
"is_satellite_provider": false,
"currency": [
"USD",
"USN",
"USS"
],
"country_details": {
"name": {
"common": "United States",
"official": "United States of America",
"Native": {
"eng": {
"common": "United States",
"official": "United States of America"
}
}
},
"EuMember": false,
"LandLocked": false,
"Nationality": "",
"tld": [
".us"
],
"Languages": {
"eng": "English"
},
"Translations": {
"FRA": {
"common": "États-Unis",
"official": "Les états-unis d'Amérique"
},
"HRV": {
"common": "Sjedinjene Američke Države",
"official": "Sjedinjene Države Amerike"
},
"ITA": {
"common": "Stati Uniti d'America",
"official": "Stati Uniti d'America"
},
"JPN": {
"common": "アメリカ合衆国",
"official": "アメリカ合衆国"
},
"NLD": {
"common": "Verenigde Staten",
"official": "Verenigde Staten van Amerika"
},
"RUS": {
"common": "Соединённые Штаты Америки",
"official": "Соединенные Штаты Америки"
},
"SPA": {
"common": "Estados Unidos",
"official": "Estados Unidos de América"
}
},
"currency": [
"USD",
"USN",
"USS"
],
"Borders": [
"CAN",
"MEX"
],
"cca2": "US",
"cca3": "USA",
"CIOC": "USA",
"CCN3": "840",
"callingCode": [
"1"
],
"InternationalPrefix": "011",
"region": "Americas",
"subregion": "Northern America",
"Continent": "North America",
"capital": "Washington D.C.",
"Area": 9372610,
"longitude": "97 00 W",
"latitude": "38 00 N",
"MinLongitude": -179.23108,
"MinLatitude": 17.831509,
"MaxLongitude": -66.885414,
"MaxLatitude": 71.441055,
"Latitude": 39.443256,
"Longitude": -98.95734
}
}
```

Clear and simple.

---

cURL

```
curl -X GET /

'https://api.apieco.ir/clearip/ip/197.164.124.28/json?apikey=API_KEY'
```

PHP

With HttpRequest

```
$request = new HttpRequest();

$request->setUrl( 'https://api.apieco.ir/clearip/ip/197.164.124.28/json');

$request->setMethod(HTTP_METH_GET);

$request->setQueryData( array( 'apikey' => 'API_KEY' ));

try { $response = $request->send(); echo $response->getBody();
} catch (HttpException $ex) { echo $ex; }
```

With cURL

```
        $curl = curl_init();
        curl_setopt_array($curl, array(
          CURLOPT_URL =>
"https://api.apieco.ir/clearip/ip/197.164.124.28/json?apikey=API_KEY",
          CURLOPT_RETURNTRANSFER => true,
          CURLOPT_ENCODING => "",
          CURLOPT_MAXREDIRS => 10,
          CURLOPT_TIMEOUT => 30,
          CURLOPT_HTTP_VERSION => CURL_HTTP_VERSION_1_1,
          CURLOPT_CUSTOMREQUEST => "GET",
          CURLOPT_HTTPHEADER => array(
        "cache-control: no-cache",
        "postman-token: 8c102565-006a-9496-c414-65a7b7a00e20"
          ),
        ));

        $response = curl_exec($curl);
        $err = curl_error($curl);

        curl_close($curl);

        if ($err) {
          echo "cURL Error #:" . $err;
        } else {
          echo $response;
        }

        }
```

Python

Python http.client (Python3)

```
            import http.client

            conn = http.client.HTTPSConnection("api.clearip.io")

            headers = {
            'cache-control': "no-cache",
            'postman-token': "af79289d-56ce-8e00-16e2-43a279b931e0"
            }

            conn.request("GET", "/ip/197.164.124.28/json?apikey=API_KEY",
            headers=headers)

            res = conn.getresponse()
            data = res.read()
            print(data.decode("utf-8"))
```

With Request

```
          import requests

          url = "https://api.apieco.ir/clearip/ip/197.164.124.28/json"
```

```
        querystring = {"apikey":"API_KEY"}

        headers = {
        'cache-control': "no-cache",
        'postman-token': "95b5b551-7870-b00f-b554-d4f41631bf3b"
        }

        response = requests.request("GET", url, headers=headers,
params=querystring)

        print(response.text)
```

Nodejs

native

```
          var http = require("https");
          var options = {
            "method": "GET",
            "hostname": "api.clearip.io",
            "port": null,
            "path": "/ip/197.164.124.28/json?apikey=API_KEY",
            "headers": {
          "cache-control": "no-cache",
          "postman-token": "c6d01235-1513-3e3c-e79f-80f3d0b20a01"
            }
          };

          var req = http.request(options, function (res) {
            var chunks = [];

            res.on("data", function (chunk) {
          chunks.push(chunk);
            });

            res.on("end", function () {
          var body = Buffer.concat(chunks);
          console.log(body.toString());
            });
          });
          req.end();
```

With Request

```
        var request = require("request");

        var options = { method: 'GET',
          url: 'https://api.apieco.ir/clearip/ip/197.164.124.28/json',
          qs: { apikey: 'API_KEY' },
          headers:
          { 'postman-token': '0010f3d4-76e5-d044-31f3-42ecc4260371',
        'cache-control': 'no-cache' } };

        request(options, function (error, response, body) {
          if (error) throw new Error(error);

          console.log(body);
        });
```

Ruby

With NET:Http

```
        require 'uri'
        require 'net/http'

        url =
URI("https://api.apieco.ir/clearip/ip/197.164.124.28/json?apikey=API_KEY")

        http = Net::HTTP.new(url.host, url.port)
        http.use_ssl = true
        http.verify_mode = OpenSSL::SSL::VERIFY_NONE

        request = Net::HTTP::Get.new(url)
        request["cache-control"] = 'no-cache'
        request["postman-token"] = 'a4f01260-f493-6218-a952-245020768e29'

        response = http.request(request)
        puts response.read_body
```

Go

```
          package main

          import (
            "fmt"
            "net/http"
            "io/ioutil"
          )

          func main() {

            url :=
"https://api.apieco.ir/clearip/ip/197.164.124.28/json?apikey=API_KEY"

            req, _ := http.NewRequest("GET", url, nil)
```

```
		req.Header.Add("cache-control", "no-cache")
		req.Header.Add("postman-token", "b3f5610d-291a-6a18-9b5e-
27d2d0e6cb5b")

		res, _ := http.DefaultClient.Do(req)

		defer res.Body.Close()
		body, _ := ioutil.ReadAll(res.Body)

		fmt.Println(res)
		fmt.Println(string(body))

	}
```

Java

OK HTTP

```
        OkHttpClient client = new OkHttpClient();
        Request request = new Request.Builder()
          .url("https://api.apieco.ir/clearip/ip/197.164.124.28/json?apikey=API_KEY")
          .get()
          .addHeader("cache-control", "no-cache")
          .addHeader("postman-token", "95fee9d2-30eb-5877-ac5b-235255c43f51")
          .build();

        Response response = client.newCall(request).execute();
```

Unirest

```
        HttpResponse response =
Unirest.get("https://api.apieco.ir/clearip/ip/197.164.124.28/json?apikey=API_KEY")
          .header("cache-control", "no-cache")
          .header("postman-token", "4f24365f-81ff-7469-0333-c475afb454bf")
          .asString();
```

Swift

NSURL

```
        import Foundation

        let headers = [
          "cache-control": "no-cache",
          "postman-token": "efd01ea4-cd6e-ad57-366a-94130554d360"
        ]
        let request = NSMutableURLRequest(url: NSURL(string: 
"https://api.apieco.ir/clearip/ip/197.164.124.28/json?apikey=API_KEY")! as URL,
        cachePolicy: .useProtocolCachePolicy,
        timeoutInterval: 10.0)
        request.httpMethod = "GET"
        request.allHTTPHeaderFields = headers

        let session = URLSession.shared
        let dataTask = session.dataTask(with: request as URLRequest, 
completionHandler: { (data, response, error) -> Void in
          if (error != nil) {
        print(error)
          } else {
        let httpResponse = response as? HTTPURLResponse
        print(httpResponse)
          }
        })

        dataTask.resume()
```

addrandomuser

Introduction

In August 2013, we set out with a goal to create a free and easy to use service to generate random user data for application testing.

How to use

You can use AJAX to call the Random User Generator API and will receive a randomly generated user in return. If you are using jQuery, you can use the $.ajax() function in the code snippet below to get started.

```
$.ajax({
  url: 'https://api.apieco.ir/randomuser/api/',
  dataType: 'json',
  success: function(data) {
    console.log(data);
  }
});
```

Results

The API will provide you with a formatted object of your choice that you can parse and apply to your application.

JSON is the default format. You can request a different format using the format parameter.

```
{
  "results": [
    {
      "gender": "male",
      "name": {
        "title": "mr",
        "first": "rolf",
        "last": "hegdal"
      },
      "location": {
        "street": "ljan terrasse 346",
        "city": "vear",
        "state": "rogaland",
        "postcode": "3095",
        "coordinates": {
          "latitude": "54.8646",
          "longitude": "-97.3136"
        },
        "timezone": {
          "offset": "-10:00",
          "description": "Hawaii"
        }
      },
      "email": "rolf.hegdal@example.com",
      "login": {
        "uuid": "c4168eac-84b8-46ea-b735-c9da9bfb97fd",
        "username": "bluefrog786",
        "password": "ingrid",
        "salt": "GtRFz4NE",
        "md5": "5c581c5748fc8c35bd7f16eac9efbb55",
        "sha1": "c3feb8887abed9ec1561b9aa2c9f58de21d1d3d9",
        "sha256": "684c478a98b43f1ef1703b35b8bbf61b27dbc93d52acd515e141e97e04447712"
      },
      "dob": {
        "date": "1975-11-12T06:34:44Z",
        "age": 42
      },
      "registered": {
        "date": "2015-11-04T22:09:36Z",
        "age": 2
      },
      "phone": "66976498",
      "cell": "40652479",
      "id": {
        "name": "FN",
        "value": "12117533881"
      },
      "picture": {
        "large": "https://api.apieco.ir/randomuser/api/portraits/men/65.jpg",
        "medium": "https://api.apieco.ir/randomuser/api/portraits/med/men/65.jpg",
        "thumbnail": "https://api.apieco.ir/randomuser/api/portraits/thumb/men/65.jpg"
      },
      "nat": "NO"
    }
  ],
  "info": {
    "seed": "2da87e9305069f1d",
    "results": 1,
    "page": 1,
    "version": "1.2"
  }
}
```

API Errors

If our API service is offline or if we are experiencing server issues, we'll return a simple JSON object with an error.

```
{
  error: "Uh oh, something has gone wrong. Please tweet us @randomapi
about the issue. Thank you."
}
```

Requesting Multiple Users

Random User Generator allows you to fetch up to 5,000 generated users in one request using the **results**parameter.

```
https://api.apieco.ir/randomuser/api/?results=5000
```

Specifying a gender

You can specify whether you would like to have only male or only female users generated by adding the **gender** parameter to your request. Valid values for the gender parameter are "male" or "female", or you may leave the parameter blank. Any other value will cause the service to return both male and female users.

```
https://api.apieco.ir/randomuser/api/?gender=female
```

Passwords

By default, passwords are chosen randomly from a list of ~10k top used passwords. Starting with version 1.1, you can have more control over how passwords are generated using the password option.

```
https://api.apieco.ir/randomuser/api/?password=upper,lower,1-16
```

The example above would generate a password consisting of uppercase and lowercase characters ranging between 1 to 16 characters long.

You can specify options for the passwords using this format:

```
https://api.apieco.ir/randomuser/api/?password=CHARSETS,MIN_LENGTH-
MAX_LENGTH
OR
https://api.apieco.ir/randomuser/api/?password=CHARSETS,MAX_LENGTH
```

You can mix and match the charsets below for the CHARSETS option above:

```
special     !"#$%&'()*+,- ./:;<=>?@[\]^_`{|}~
upper       ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

```
lower       abcdefghijklmnopqrstuvwxyz
number      0123456789
```

MIN_LENGTH and MAX_LENGTH are the min/max length of the passwords that you want to generate.
By default, passwords will be between 8 - 64 characters long.

Here are some more examples of password option combinations:

```
// Special chars exactly 32 characters long
https://api.apieco.ir/randomuser/api/?password=special,32

// Uppercase chars between 1 to 8 characters long
https://api.apieco.ir/randomuser/api/?password=upper,1-8

// Special, uppercase, lowercase, and numeric chars between the default 8
to 64 characters long
https://api.apieco.ir/randomuser/api/?password=special,upper,lower,number
```

Seeds

Seeds allow you to always generate the same set of users. For example, the seed "foobar" will always return results for Becky Sims (for version 1.0). Seeds can be any string or sequence of characters.

```
https://api.apieco.ir/randomuser/api/?seed=foobar
```

Formats

We currently offer the following data formats:

- JSON (default)
- PrettyJSON or pretty
- CSV
- YAML
- XML

Just specify the format you would like returned by using the **format** parameter.

```
https://api.apieco.ir/randomuser/api/?format=csv
```

Using previous versions

When we release a new version the API, it could possibly break your application. By accessing the API via:

```
https://api.apieco.ir/randomuser/api/
```

The result that is returned automatically uses the latest version of the API.
If you want to access a specific version of the API that won't be affected by updates, do this:

```
https://api.apieco.ir/randomuser/api/1.2/
```

Nationalities

You can request a different nationality of a randomuser.

Pictures won't be affected by this, but data such as location, cell/home phone, id, etc. will be more appropriate.

Currently, randomuser offers these nationalities:

- v1.0: AU, BR, CA, CH, DE, DK, ES, FI, FR, GB, IE, IR, NL, NZ, TR, US
- v1.1: AU, BR, CA, CH, DE, DK, ES, FI, FR, GB, IE, IR, NL, NZ, TR, US
- v1.2: AU, BR, CA, CH, DE, DK, ES, FI, FR, GB, IE, IR, NO, NL, NZ, TR, US

You can specify a nationality like so:

```
https://api.apieco.ir/randomuser/api/?nat=gb
```

Randomuser will return random nats by default. You can have some control with the nats that you'd like to have generated by specifying a comma seperated list:

```
https://api.apieco.ir/randomuser/api/?nat=us,dk,fr,gb
```

Pagination

You can request multiple pages of a seed with the **page** parameter.
Make sure that you use the same seed and page number (1 based index) in order to get back the same results

```
https://api.apieco.ir/randomuser/api/?page=3&results=10&seed=abc
```

Including/Excluding fields

Sometimes, maybe you want some random names and not extraneous data such as location, phone, etc.
Using the **inc** and **exc** parameters, you can specify which fields to include or exclude respectively.

By specifying only the fields you want, the generator can save time by skipping CPU intensive fields like "login" for example.

These parameters accept the following values in a comma delimited list

- gender
- name
- location
- email
- login
- registered
- dob
- phone
- cell
- id
- picture
- nat

If you only wanted the names,genders,and nats of users:

```
https://api.apieco.ir/randomuser/api/?inc=gender,name,nat
```

If you want everything except for login data:

```
https://api.apieco.ir/randomuser/api/?exc=login
```

Misc

Some extra parameters that you can add to a request.

- **dl** - Download the results with the appropriate format extension.

```
https://api.apieco.ir/randomuser/api/?results=25&nat=gb,us,es&for
mat=csv&dl
```

- **noinfo** - If you only want the data, and don't care for seed, results, page, and version data.

```
https://api.apieco.ir/randomuser/api/?results=5&inc=name,gender,n
at&noinfo
```

- **callback** - If you want the payload in JSONP, supply a callback using the callback parameter. Only available with JSON formats.

```
https://api.apieco.ir/randomuser/api/?results=5&callback=randomus
erdata
```

If you find a mistake with an API or would like to contribute to our database, feel free to visit our Github Repo. We'd really appreciate it :)

Restcountries

REST COUNTRIES

Get information about countries via a RESTful API

Current version: 2.0.5

DONATE!

The restcountries project has been acquired by apilayer, one of the leading providers of API microservices. We will keep supporting restcountries and providing it as a free solution for developers. We will finance this project fully and have turned off the donations feature.

USERS

RESTCountries has over 1200 users, including:
TTÜ
Spotify International Pricing Index
Gorillaz
Wanderlust
Xero
FxPro
SKROSS
onefinestay
Much Better Adventures

STAY UP-TO-DATE

Follow RESTCountries on Twitter
Or subscribe to the mailing list

API ENDPOINTS

Below are described the REST endpoints available that you can use to search for countries

ALL

https://api.apieco.ir/restcountries/rest/v2/all

NAME

Search by country name. It can be the native name or partial name
https://api.apieco.ir/restcountries/rest/v2/name/{name}
https:// api.apieco.ir/restcountries
/rest/v2/name/eesti
https://api.apieco.ir/restcountries/rest/v2/name/united

FULL NAME

Search by country full name
https://api.apieco.ir/restcountries /rest/v2/name/{name}?fullText=true
https://api.apieco.ir/restcountries/rest/v2/name/aruba?fullText=true

CODE

Search by ISO 3166-1 2-letter or 3-letter country code
https://api.apieco.ir/restcountries /rest/v2/alpha/{code}
https://api.apieco.ir/restcountries/rest/v2/alpha/co
https://api.apieco.ir/restcountries/rest/v2/alpha/col

LIST OF CODES

Search by list of ISO 3166-1 2-letter or 3-letter country codes
https://api.apieco.ir/restcountries /rest/v2/alpha?codes={code};{code};{code}
https://api.apieco.ir/restcountries/rest/v2/alpha?codes=col;no;ee

CURRENCY

Search by ISO 4217 currency code

https://api.apieco.ir/restcountries/rest/v2/currency/{currency}
https://api.apieco.ir/restcountries/rest/v2/currency/cop

LANGUAGE

Search by ISO 639-1 language code.

https://api.apieco.ir/restcountries/rest/v2/lang/{et}
https://api.apieco.ir/restcountries/rest/v2/lang/es

CAPITAL CITY

Search by capital city

https://api.apieco.ir/restcountries/rest/v2/capital/{capital}
https://api.apieco.ir/restcountries/rest/v2/capital/tallinn

CALLING CODE

Search by calling code

https://api.apieco.ir/restcountries/rest/v2/callingcode/{callingcode}
https://api.apieco.ir/restcountries/rest/v2/callingcode/372

REGION

Search by region: Africa, Americas, Asia, Europe, Oceania

https://api.apieco.ir/restcountries/rest/v2/region/{region}
https://api.apieco.ir/restcountries/rest/v2/region/europe

REGIONAL BLOC

Search by regional bloc:

- EU (European Union)
- EFTA (European Free Trade Association)
- CARICOM (Caribbean Community)
- PA (Pacific Alliance)
- AU (African Union)
- USAN (Union of South American Nations)
- EEU (Eurasian Economic Union)
- AL (Arab League)
- ASEAN (Association of Southeast Asian Nations)
- CAIS (Central American Integration System)
- CEFTA (Central European Free Trade Agreement)
- NAFTA (North American Free Trade Agreement)
- SAARC (South Asian Association for Regional Cooperation)

https://api.apieco.ir/restcountries/rest/v2/regionalbloc/{regionalbloc}
https://api.apieco.ir/restcountries/rest/v2/regionalbloc/eu

RESPONSE EXAMPLE

https://restcountries.eu/rest/v2/alpha/col

```
[{
    "name": "Colombia",
    "topLevelDomain": [".co"],
    "alpha2Code": "CO",
    "alpha3Code": "COL",
    "callingCodes": ["57"],
    "capital": "Bogotá",
    "altSpellings": ["CO", "Republic of Colombia", "República de Colombia"],
    "region": "Americas",
    "subregion": "South America",
    "population": 48759958,
    "latlng": [4.0, -72.0],
    "demonym": "Colombian",
    "area": 1141748.0,
    "gini": 55.9,
    "timezones": ["UTC-05:00"],
    "borders": ["BRA", "ECU", "PAN", "PER", "VEN"],
    "nativeName": "Colombia",
    "numericCode": "170",
    "currencies": [{
        "code": "COP",
        "name": "Colombian peso",
        "symbol": "$"
    }],
    "languages": [{
        "iso639_1": "es",
        "iso639_2": "spa",
        "name": "Spanish",
        "nativeName": "Español"
    }],
    "translations": {
        "de": "Kolumbien",
        "es": "Colombia",
        "fr": "Colombie",
        "ja": "コロンビア",
        "it": "Colombia",
        "br": "Colômbia",
        "pt": "Colômbia"
    },
    "flag": "https://api.apieco.ir/restcountries /data/col.svg",
    "regionalBlocs": [{
        "acronym": "PA",
        "name": "Pacific Alliance",
        "otherAcronyms": [],
        "otherNames": ["Alianza del Pacífico"]
    }, {
        "acronym": "USAN",
        "name": "Union of South American Nations",
        "otherAcronyms": ["UNASUR", "UNASUL", "UZAN"],
        "otherNames": ["Unión de Naciones Suramericanas", "União de Nações Sul-Americanas", "Unie van Zuid-Amerikaanse Naties", "South
American Union"]
    }],
    "cioc": "COL"
}]
```

FILTER RESPONSE

You can filter the output of your request to include only the specified fields.

```
https://api.apieco.ir/restcountries /rest/v2/{service}?fields={field};{field};{field}
https://api.apieco.ir/restcountries/rest/v2/all?fields=name;capital;currencies
```

SOURCES

- @mledoze
- List of countries
- Languages
- Currencies
- Area

SIMILAR PROJECTS

- Countries of the world
- REST Countries Node.js
- REST Countries Ruby
- REST Countries Go
- REST Countries Python
- world-currencies

LICENSE

Mozilla Public License MPL 2.0

dbtimezone

References

List Time Zone

`http://api.apieco.ir/timezonedb/v2.1/list-time-zone`

List out all available time zones supported by TimeZoneDB.

Reference: List Time Zone

`http://api.apieco.ir/timezonedb/v2.1/list-time-zone`

List out all available time zones supported by TimeZoneDB.

Other End Points

- Get Time Zone
- Convert Time Zone

Parameters

Parameter	Description
key	REQUIRED Your unique API key you get after register your account.
format	OPTIONAL The response format from API. It can be either **xml** or **json**. **DEFAULT:** xml
callback	OPTIONAL Use for JavaScript JSON callback.
fields	OPTIONAL

Parameter	Description
	Customize the field to display in response. Use commas ("," without spaces) to separate the field names. **FIELDS:** countryCode, countryName, zoneName, gmtOffset, timestamp **DEFAULT:** all OPTIONAL
country	A valid ISO 3166 country code. Only time zones of provided country will list out. OPTIONAL
zone	The name of a time zone. Use asterisk (*) for wildcard search.

Responses

Field	Description
status	Status of the API query. Either **OK** or **FAILED**.
message	Error message. Empty if no error.
countryCode	Country code of the time zone.
countryName	Country name of the time zone.
zoneName	The time zone name.
gmtOffset	The time offset in seconds based on UTC time.
timestamp	Current local time in Unix time. Minus the value with **gmtOffset** to get UTC time.

Usage Examples

Get all time zones supported by TimeZoneDB

XML

Query

```
http://api.apieco.ir/timezonedb/v2.1/list-time-
zone?key=YOUR_API_KEY&format=xml
```

Response

```
<?xml version="1.0" encoding="UTF-8"?>
<result>
    <status>OK</status>
    <message/>
    <zones>
        <zone>
            <countryCode>AD</countryCode>
            <countryName>Andorra</countryName>
            <zoneName>Europe/Andorra</zoneName>
            <gmtOffset>7200</gmtOffset>
            <timestamp>1464453737</timestamp>
        </zone>
        <zone>
            <countryCode>AE</countryCode>
            <countryName>United Arab Emirates</countryName>
            <zoneName>Asia/Dubai</zoneName>
            <gmtOffset>14400</gmtOffset>
            <timestamp>1464460937</timestamp>
        </zone>
        <zone>
            <countryCode>AF</countryCode>
            <countryName>Afghanistan</countryName>
            <zoneName>Asia/Kabul</zoneName>
            <gmtOffset>16200</gmtOffset>
            <timestamp>1464462737</timestamp>
        </zone>
        .
        .
        .
    </zones>
</result>
```

JSON

Query

```
http://api.apieco.ir/timezonedb/v2.1/list-time-
zone?key=YOUR_API_KEY&format=json
```

Response

```
{
    "status":"OK",
    "message":"",
    "zones":[
        {
            "countryCode":"AD",
            "countryName":"Andorra",
            "zoneName":"Europe\/Andorra",
            "gmtOffset":7200,
            "timestamp":1464453737
        },
        {
            "countryCode":"AE",
            "countryName":"United Arab Emirates",
            "zoneName":"Asia\/Dubai",
            "gmtOffset":14400,
            "timestamp":1464460937
        },
        {"
            countryCode":"AF",
            "countryName":"Afghanistan",
            "zoneName":"Asia\/Kabul",
            "gmtOffset":16200,
            "timestamp":1464462737
        }
        .
        .
        .
    ]
}
```

Get all time zones in New Zealand

XML

Query

```
http://api.apieco.ir/timezonedb/v2.1/list-time-
zone?key=YOUR_API_KEY&format=xml&country=NZ
```

Response

```
<?xml version="1.0" encoding="UTF-8"?>
<result>
    <status>OK</status>
    <message/>
    <zones>
        <zone>
            <countryCode>NZ</countryCode>
            <countryName>New Zealand</countryName>
            <zoneName>Pacific/Auckland</zoneName>
            <gmtOffset>43200</gmtOffset>
            <timestamp>1464537416</timestamp>
        </zone>
        <zone>
            <countryCode>NZ</countryCode>
            <countryName>New Zealand</countryName>
            <zoneName>Pacific/Chatham</zoneName>
            <gmtOffset>45900</gmtOffset>
            <timestamp>1464540116</timestamp>
        </zone>
    </zones>
</result>
```

JSON

Query

```
http://api.apieco.ir/timezonedb/v2.1/list-time-zone
?key=YOUR_API_KEY&format=json&country=NZ
```

Response

```
{
    "status":"OK",
    "message":"",
    "zones":[
        {
            "countryCode":"NZ",
            "countryName":"New Zealand",
            "zoneName":"Pacific\/Auckland",
            "gmtOffset":43200,
            "timestamp":1464537416
        },
        {
            "countryCode":"NZ",
            "countryName":"New Zealand",
            "zoneName":"Pacific\/Chatham",
            "gmtOffset":45900,
            "timestamp":1464540116
        }
    ]
}
```

Get all time zones in United States where zone name contains "New"

XML

Query

```
http://api.apieco.ir/timezonedb/v2.1/list-time-zone?key=YOUR_API_KEY&format
=xml&country=US&zone=*New*
=NZ
```

Response

```
<?xml version="1.0" encoding="UTF-8"?>
<result>
    <status>OK</status>
    <message/>
    <zones>
        <zone>
            <countryCode>US</countryCode>
            <countryName>United States</countryName>
            <zoneName>America/New_York</zoneName>
            <gmtOffset>-14400</gmtOffset>
            <timestamp>1464480694</timestamp>
        </zone>
        <zone>
            <countryCode>US</countryCode>
            <countryName>United States</countryName>
            <zoneName>America/North_Dakota/New_Salem</zoneName>
            <gmtOffset>-18000</gmtOffset>
            <timestamp>1464477094</timestamp>
        </zone>
    </zones>
</result>
```

JSON

Query

```
http://api.apieco.ir/timezonedb/v2.1/list-time-
zone?key=YOUR_API_KEY&format=json&country=US&zone=*New*
```

Response

```
{
    "status":"OK",
    "message":"",
    "zones":[
        {
            "countryCode":"US",
            "countryName":"United States",
            "zoneName":"America\/New_York",
            "gmtOffset":-14400,
            "timestamp":1464480694
        },
        {
            "countryCode":"US",
            "countryName":"United States",
            "zoneName":"America\/North_Dakota\/New_Salem",
            "gmtOffset":-18000,
            "timestamp":1464477094
        }
    ]
}
```

Customize which field to show in result

XML

Quer

```
http://api.apieco.ir/timezonedb/v2.1/list-time-
zone?key=YOUR_API_KEY&format=xml&zone=Asia/Tokyo&fields=zoneName,gmtOffset
```

Response

```
<?xml version="1.0" encoding="UTF-8"?>
<result>
    <status>OK</status>
    <message/>
    <zones>
        <zone>
            <zoneName>Asia/Tokyo</zoneName>
            <gmtOffset>32400</gmtOffset>
        </zone>
    </zones>
</result>
```

JSON

Query

```
http://api.apieco.ir/timezonedb/v2.1/list-time-
zone?key=YOUR_API_KEY&format=json&zone=Asia/Tokyo&fields=zoneName,gmtOffset
```

Response

```
{
    "status":"OK",
    "message":"",
    "zones":[
        {
            "zoneName":"Asia\/Tokyo",
            "gmtOffset":32400
        }
    ]
}
```

Reference

Get Time Zone

`http://api.apieco.ir/timezonedb/v2.1/get-time-zone`

Get local time of a city by its name, time zone, latitude & longtiude, or IP address.

Other End Points

- List Time Zone
- Convert Time Zone

Parameters

Parameter	Description
key	REQUIRED Your unique API key you get after register your account.
format	OPTIONAL The response format from API. It can be either **xml** or **json**. **DEFAULT:** xml
callback	OPTIONAL Use for JavaScript JSON callback.
fields	OPTIONAL Customize the field to display in response. Use commas (",") without spaces) to separate the field names. **FIELDS:** countryCode, countryName, regionName, cityName, zoneName, abbreviation, gmtOffset, dst, zoneStart, zoneEnd, nextAbbreviation, timestamp, formatted **DEFAULT:** all
by	REQUIRED The method of lookup. **zone** - Lookup local time by using a time zone name.

Parameter	Description
	position - Lookup local time by using latitude & longitude of a city. • PREMIUM **city** - Lookup time zone by searching city name. • PREMIUM **ip** - Lookup time zone based on visitor IP address.
zone	REQUIRED A time zone abbreviation or time zone name. Required if lookup by **zone** method.
lat	REQUIRED Latitude of a city. Required if lookup by **position** method.
lng	REQUIRED Longitude of a city. Required if lookup by **position** method.
country	REQUIRED A valid ISO 3166 country code. Required if lookup by **city** method.
region	OPTIONAL A valid region code of United States. Optional when lookup by **city** method to limit the search result.
city	REQUIRED The name of a city. Use asterisk (*) for wildcard search. Required if lookup by **city** method.
page	OPTIONAL Navigate to other page when result is more than 25 records.
time	OPTIONAL Unix time in UTC. **DEFAULT:** Current UTC time.

Responses

Field	Description
status	Status of the API query. Either **OK** or **FAILED**.
message	Error message. Empty if no error.
countryCode	Country code of the time zone.
countryName	Country name of the time zone.
zoneName	The time zone name.
abbreviation	Abbreviation of the time zone.
gmtOffset	The time offset in seconds based on UTC time.
dst	Whether Daylight Saving Time (DST) is used. Either **0** (No) or **1** (Yes).
zoneStart	The Unix time in UTC when current time zone start.
zoneEnd	The Unix time in UTC when current time zone end.
timestamp	Current local time in Unix time. Minus the value with **gmtOffset** to get UTC time.
formatted	Formatted timestamp in **Y-m-d h:i:s** format. E.g.: 2019-04-16 12:56:42
totalPage	The total page of result when exceed 25 records.
currentPage	Current page when navigating.

Usage Examples

Get current local time in Chicago, USA

XML

Query

```
http://api.apieco.ir/timezonedb/v2.1/get-time-
zone?key=YOUR_API_KEY&format=xml&by=zone&zone=America/Chicago
```

Response

```
<?xml version="1.0" encoding="UTF-8"?>

<result>

    <status>OK</status>

    <message/>

    <countryCode>US</countryCode>

    <countryName>United States</countryName>

    <zoneName>America/Chicago</zoneName>

    <abbreviation>CST</abbreviation>

    <gmtOffset>-21600</gmtOffset>

    <dst>0</dst>

    <zoneStart>1446361200</zoneStart>

    <zoneEnd>1457856000</zoneEnd>

    <nextAbbreviation>CDT</nextAbbreviation>

    <timestamp>1454446991</timestamp>

    <formatted>2016-02-02 21:03:11</formatted>

</result>
```

JSON

Query

```
http://api.apieco.ir/timezonedb/v2.1/get-time-
zone?key=YOUR_API_KEY&format=json&by=zone&zone=America/Chicago
```

Response

```
{
    "status":"OK",
    "message":"",
    "countryCode":"US",
    "countryName":"United States",
    "zoneName":"America\/Chicago",
    "abbreviation":"CST",
    "gmtOffset":-21600,
    "dst":"0",
    "zoneStart":1446361200,
    "zoneEnd":1457856000,
    "nextAbbreviation":"CDT",
    "timestamp":1454446991,
    "formatted":"2016-02-02 21:03:11"
}
```

Get current local time in Taipei, Taiwan when UTC is 24 December, 2015 11:50:55 PM

XML

Query

```
http://api.apieco.ir/timezonedb/v2.1/get-time-
zone?key=YOUR_API_KEY&format=xml&by=zone&zone=Asia/Taipei&time=1451001055
```

Response

```
<?xml version="1.0" encoding="UTF-8"?>
<result>
    <status>OK</status>
    <message/>
    <countryCode>TW</countryCode>
    <countryName>Taiwan</countryName>
    <zoneName>Asia/Taipei</zoneName>
    <abbreviation>CST</abbreviation>
    <gmtOffset>28800</gmtOffset>
    <dst>0</dst>
    <zoneStart>307551600</zoneStart>
    <zoneEnd>0</zoneEnd>
    <nextAbbreviation/>
    <timestamp>1451029855</timestamp>
    <formatted>2015-12-25 07:50:55</formatted>
</result>
```

JSON

Query

```
http://api.apieco.ir/timezonedb/v2.1/get-time-
zone?key=YOUR_API_KEY&format=json&by=zone&zone=Asia/Taipei&time=1451001055
```

Response

```
{
    "status":"OK",
    "message":"",
    "countryCode":"TW",
    "countryName":"Taiwan",
    "zoneName":"Asia\/Taipei",
    "abbreviation":"CST",
    "gmtOffset":28800,
    "dst":"0",
    "zoneStart":307551600,
    "zoneEnd":0,
    "nextAbbreviation":"",
    "timestamp":1451029855,
    "formatted":"2015-12-25 07:50:55"
}
```

Get local time zone for City of Buffalo in US

XML

Query

```
http://api.apieco.ir/timezonedb/v2.1/get-time-
zone?key=YOUR_API_KEY&format=xml&by=city&city=City+of+Buffalo&country=US
```

Response

```
<?xml version="1.0" encoding="UTF-8"?>
<result>
    <status>OK</status>
    <message/>
    <totalPage>1</totalPage>
    <currentPage>1</currentPage>
    <zones>
        <zone>
            <countryCode>US</countryCode>
            <countryName>United States</countryName>
            <regionName>Iowa</regionName>
            <cityName>City of Buffalo</cityName>
            <zoneName>America/Chicago</zoneName>
            <abbreviation>CST</abbreviation>
            <gmtOffset>-21600</gmtOffset>
            <dst>0</dst>
            <zoneStart>1320562800</zoneStart>
            <timestamp>1321195745</timestamp>
            <formatted>2011-11-13 14:49:05</formatted>
            <zoneEnd>1331452800</zoneEnd>
            <nextAbbreviation>CDT</nextAbbreviation>
        </zone>
        <zone>
            <countryCode>US</countryCode>
            <countryName>United States</countryName>
            <regionName>Kansas</regionName>
            <cityName>City of Buffalo</cityName>
            <zoneName>America/Chicago</zoneName>
            <abbreviation>CST</abbreviation>
            <gmtOffset>-21600</gmtOffset>
            <dst>0</dst>
            <zoneStart>1320562800</zoneStart>
            <timestamp>1321195745</timestamp>
            <formatted>2011-11-13 14:49:05</formatted>
            <zoneEnd>1331452800</zoneEnd>
            <nextAbbreviation>CDT</nextAbbreviation>
        </zone>
        <zone>
            <countryCode>US</countryCode>
            <countryName>United States</countryName>
            <regionName>Missouri</regionName>
            <cityName>City of Buffalo</cityName>
            <zoneName>America/Chicago</zoneName>
            <abbreviation>CST</abbreviation>
            <gmtOffset>-21600</gmtOffset>
            <dst>0</dst>
            <zoneStart>1320562800</zoneStart>
            <timestamp>1321195745</timestamp>
            <formatted>2011-11-13 14:49:05</formatted>
            <zoneEnd>1331452800</zoneEnd>
            <nextAbbreviation>CDT</nextAbbreviation>
        </zone>
        <zone>
```

```
            <zoneName>America/Chicago</zoneName>
            <abbreviation>CST</abbreviation>
            <gmtOffset>-21600</gmtOffset>
            <dst>0</dst>
            <zoneStart>1320562800</zoneStart>
            <timestamp>1321195745</timestamp>
            <formatted>2011-11-13 14:49:05</formatted>
            <zoneEnd>1331452800</zoneEnd>
            <nextAbbreviation>CDT</nextAbbreviation>
        </zone>
        <zone>
            <countryCode>US</countryCode>
            <countryName>United States</countryName>
            <regionName>North Dakota</regionName>
            <cityName>City of Buffalo</cityName>
            <zoneName>America/Chicago</zoneName>
            <abbreviation>CST</abbreviation>
            <gmtOffset>-21600</gmtOffset>
            <dst>0</dst>
            <zoneStart>1320562800</zoneStart>
            <timestamp>1321195745</timestamp>
            <formatted>2011-11-13 14:49:05</formatted>
            <zoneEnd>1331452800</zoneEnd>
            <nextAbbreviation>CDT</nextAbbreviation>
        </zone>
        <zone>
            <countryCode>US</countryCode>
            <countryName>United States</countryName>
            <regionName>New York</regionName>
            <cityName>City of Buffalo</cityName>
            <zoneName>America/New_York</zoneName>
            <abbreviation>EST</abbreviation>
            <gmtOffset>-18000</gmtOffset>
            <dst>0</dst>
            <zoneStart>1320559200</zoneStart>
            <timestamp>1321199345</timestamp>
            <formatted>2011-11-13 15:49:05</formatted>
            <zoneEnd>1331449200</zoneEnd>
            <nextAbbreviation>EDT</nextAbbreviation>
        </zone>
        <zone>
            <countryCode>US</countryCode>
            <countryName>United States</countryName>
            <regionName>Texas</regionName>
            <cityName>City of Buffalo</cityName>
            <zoneName>America/Chicago</zoneName>
            <abbreviation>CST</abbreviation>
            <gmtOffset>-21600</gmtOffset>
            <dst>0</dst>
            <zoneStart>1320562800</zoneStart>
            <timestamp>1321195745</timestamp>
            <formatted>2011-11-13 14:49:05</formatted>
            <zoneEnd>1331452800</zoneEnd>
            <nextAbbreviation>CDT</nextAbbreviation>
        </zone>
        <zone>
            <countryCode>US</countryCode>
            <countryName>United States</countryName>
            <regionName>Wyoming</regionName>
            <cityName>City of Buffalo</cityName>
            <zoneName>America/Denver</zoneName>
            <abbreviation>MST</abbreviation>
            <gmtOffset>-25200</gmtOffset>
            <dst>0</dst>
            <zoneStart>1320566400</zoneStart>
            <timestamp>1321192145</timestamp>
            <formatted>2011-11-13 13:49:05</formatted>
            <countryCode>US</countryCode>
            <countryName>United States</countryName>
            <regionName>Wyoming</regionName>
            <cityName>City of Buffalo</cityName>
            <zoneName>America/Denver</zoneName>
            <abbreviation>MST</abbreviation>
            <gmtOffset>-25200</gmtOffset>
            <dst>0</dst>
            <zoneStart>1320566400</zoneStart>
            <timestamp>1321192145</timestamp>
            <formatted>2011-11-13 13:49:05</formatted>
            <zoneEnd>1331456400</zoneEnd>
            <nextAbbreviation>MDT</nextAbbreviation>
        </zone>
    </zones>
</result>
```

JSON

Query

```
http://api.apieco.ir/timezonedb/v2.1/get-time-
zone?key=YOUR_API_KEY&format=json&by=city&city=City+of+Buffalo&country=US
```

Response

```
{
    "status":"OK",
    "message":"",
    "zones":[
        {
            "countryCode":"US",
            "countryName":"United States",
            "regionName":"Iowa",
            "cityName":"City of Buffalo",
            "zoneName":"America\/Chicago",
            "abbreviation":"CST",
            "gmtOffset":-21600,
            "dst":"0",
            "zoneStart":1320562800,
            "timestamp":1321195745,
            "formatted":"2011-11-13 14:49:05",
            "zoneEnd":1331452800,
            "nextAbbreviation":"CDT"
        },
        {
            "countryCode":"US",
            "countryName":"United States",
            "regionName":"Kansas",
            "cityName":"City of Buffalo",
            "zoneName":"America\/Chicago",
            "abbreviation":"CST",
            "gmtOffset":-21600,
            "dst":"0",
            "zoneStart":1320562800,
            "abbreviation":"CST",
             "gmtOffset":-21600,
             "dst":"0",
             "zoneStart":1320562800,
             "timestamp":1321195745,
             "formatted":"2011-11-13 14:49:05",
             "zoneEnd":1331452800,
             "nextAbbreviation":"CDT"
         },
         {
             "countryCode":"US",
             "countryName":"United States",
             "regionName":"North Dakota",
             "cityName":"City of Buffalo",
             "zoneName":"America\/Chicago",
             "abbreviation":"CST",
             "gmtOffset":-21600,
             "dst":"0",
             "zoneStart":1320562800,
             "timestamp":1321195745,
             "formatted":"2011-11-13 14:49:05",
             "zoneEnd":1331452800,
             "nextAbbreviation":"CDT"
         },
         {
             "countryCode":"US",
             "countryName":"United States",
             "regionName":"New York",
             "cityName":"City of Buffalo",
             "zoneName":"America\/New_York",
             "abbreviation":"EST",
             "gmtOffset":-18000,
             "dst":"0",
             "zoneStart":1320559200,
             "timestamp":1321199345,
             "formatted":"2011-11-13 15:49:05",
             "zoneEnd":1331449200,
             "nextAbbreviation":"EDT"
         },
         {
             "countryCode":"US",
             "countryName":"United States",
             "regionName":"Texas",
             "cityName":"City of Buffalo",
             "zoneName":"America\/Chicago",
             "abbreviation":"CST",
             "gmtOffset":-21600,
             "dst":"0",
             "zoneStart":1320562800,
             "timestamp":1321195745,
             "formatted":"2011-11-13 14:49:05",
             "zoneEnd":1331452800,
             "nextAbbreviation":"CDT"
         },
         {
             "countryCode":"US",
             "countryName":"United States",
             "regionName":"Wyoming",
             "cityName":"City of Buffalo",
             "zoneName":"America\/Denver",
             "abbreviation":"MST",
             "gmtOffset":-25200,
             "dst":"0",
             "zoneStart":1320566400,
             "timestamp":1321192145,
             "formatted":"2011-11-13 13:49:05",
             "zoneEnd":1331456400,
             "nextAbbreviation":"MDT"
```

```
                "abbreviation":"MST",

                "gmtOffset":-25200,
                "dst":"0",
                "zoneStart":1320566400,
                "timestamp":1321192145,
                "formatted":"2011-11-13 13:49:05",
                "zoneEnd":1331456400,
                "nextAbbreviation":"MDT"
           }
      ]
}
```

Get time zone by IP address with custom fields

XML

Query

```
http://api.apieco.ir/timezonedb/v2.1/get-time-
zone?key=YOUR_API_KEY&format=xml&by=ip&ip=66.249.64.135&fields

=countryCode,cityName,gmtOffset,dst
```

Response

```
<?xml version="1.0" encoding="UTF-8"?>

<result>

    <status>OK</status>

    <message/>

    <countryCode>US</countryCode>

    <cityName>Mountain View</cityName>

    <gmtOffset>-25200</gmtOffset>

    <dst>1</dst>

</result>
```

JSON

Query

```
http://api.apieco.ir/timezonedb/v2.1/get-time-
zone?key=YOUR_API_KEY&format=json&by=ip&ip=66.249.64.135&fields
=countryCode,cityName,gmtOffset,dst
```

Response

```
{
    "status":"OK",
    "message":"",
    "countryCode":"US",
    "cityName":"Mountain View",
    "gmtOffset":-25200,
    "dst":"1"
}
```

Reference:

Convert Time Zone

```
http://api.apieco.ir/timezonedb/v2.1/convert-time-zone
```

Convert timestamp between two different time zone.

Other End Points

- List Time Zone
- Get Time Zone

Parameters

Parameter	Description
key	REQUIRED Your unique API key you get after register your account.
format	OPTIONAL The response format from API. It can be either **xml** or **json**. **DEFAULT:** xml
callback	OPTIONAL Use for JavaScript JSON callback.
fields	OPTIONAL Customize the field to display in response. Use commas ("," without spaces) to separate the field names. **FIELDS:** fromZoneName, fromAbbreviation, fromTimestamp, toZoneName, toAbbreviation, toTimestamp, toFormatted, offset **DEFAULT:** all
from	REQUIRED A valid abbreviation or name of time zone to convert from.
to	REQUIRED

Parameter	Description
	A valid abbreviation or name of time zone to convert to.
	OPTIONAL
	Local Unix time of the **from** time zone.
time	**DEFAULT:** Current UTC time.

Responses

Field	Description
status	Status of the API query. Either **OK** or **FAILED**.
message	Error message. Empty if no error.
fromZoneName	The time zone name of the origin city.
fromAbbreviation	Time zone abbreviation of the origin city.
fromTimestamp	Time of the origin city in Unix time.
toZoneName	The time zone name of the destination city.
toAbbreviation	Time zone abbreviation of the destination city.
toTimestamp	Time of the destination city in Unix time.
offset	Different in seconds between origin time zone and destination time zone.

Usage Examples

What is the time in Sydney, Australia when 01 June, 2016 03:00PM in Los Angeles, USA?

XML

Query

```
http://api.apieco.ir/timezonedb/v2.1/convert-time-zone?key=YOUR_API_KEY&format=xml&from=America/Los_Angeles&to
=Australia/Sydney&time=1464793200
```

Response

```
<?xml version="1.0" encoding="UTF-8"?>
<result>
    <status>OK</status>
    <message/>
    <fromZoneName>America/Los_Angeles</fromZoneName>
    <fromAbbreviation>PDT</fromAbbreviation>
    <fromTimestamp>1464793200</fromTimestamp>
    <toZoneName>Australia/Sydney</toZoneName>
    <toAbbreviation>AEST</toAbbreviation>
    <toTimestamp>1464854400</toTimestamp>
    <offset>61200</offset>
</result>
```

JSON

Query

```
http:// api.apieco.ir/timezonedb /v2.1/convert-time-
zone?key=YOUR_API_KEY&format=json&from=America/Los_Angeles&to

=Australia/Sydney&time=1464793200
```

Response

```
{
    "status":"OK",
    "message":"",
    "fromZoneName":"America\/Los_Angeles",
    "fromAbbreviation":"PDT",
    "fromTimestamp":1464793200,
    "toZoneName":"Australia\/Sydney",
    "toAbbreviation":"AEST",
    "toTimestamp":1464854400,
    "offset":61200
}
```

Geopluging

JSON Geolocation Web Service

The JSON Geolocation web service API [1] allows you to directly program your back-end JSON scripts to deliver dynamic geo-localized pages using the JSON variables provided by geoPlugin. To access this service, add the following url to a remote include call

```
http://api.apieco.ir/geoplugin/json.gp?ip=xx.xx.xx.xx
```

Of course, substitute the xx's with your visitor's IP number. If you are calling the JSON service via jQuery, you don't need to include the IP since the IP will be automatically resolved to your visitor.

Example output of a JSON query for your IP address (195.181.168.180) is:

```
{



  "geoplugin_request":"195.181.168.180",



  "geoplugin_status":200,

  "geoplugin_delay":"1ms",



  "geoplugin_credit":"Some of the returned data includes GeoLite data created by
MaxMind, available from http:\/\/www.maxmind.com<\/a>.",

  "geoplugin_city":"New York",

  "geoplugin_region":"New York",

  "geoplugin_regionCode":"NY",

  "geoplugin_regionName":"New York",

  "geoplugin_areaCode":"",

  "geoplugin_dmaCode":"501",

  "geoplugin_countryCode":"US",

  "geoplugin_countryName":"United States",

  "geoplugin_inEU":0,
  "geoplugin_euVATrate":false,
  "geoplugin_continentCode":"NA",

  "geoplugin_continentName":"North America",

  "geoplugin_latitude":"40.7185",

  "geoplugin_longitude":"-74.0025",

  "geoplugin_locationAccuracyRadius":"200",

  "geoplugin_timezone":"America\/New_York",

  "geoplugin_currencyCode":"USD",

  "geoplugin_currencySymbol":"$",

  "geoplugin_currencySymbol_UTF8":"$",

  "geoplugin_currencyConverter":1

}
```

JSON Geolocation Currency Converter

The variable "geoplugin_currencyConverter" is the conversion rate for the currency converter base currency.
Like all calls to any of geoPlugin's web services, the default base_currency is USD ($US).
Thus, if your base currency is NOT $US, then you must add the variable base_currency=XXX to the call to geoplugin.net eg

```
http://api.apieco.ir/geoplugin/json.gp?base_currency=EUR
```

Now geoplugin_currencyConverter will output the exchange rate of one Euro for your visitor.

The base_currency value must be a valid ISO 4217 3-letter code.

AJAX and Error: Invalid label

If you are using jQuery for example to do AJAX calls to the JSON webservice, you will probably be seeing the Javascript error ***Error:*** *Invalid label*

This is due to data requests from a server (geoPlugin) in a different domain (your web server).
To eliminate this problem, JSONP or "JSON with padding" is required.
To return the JSON results as JSONP, tag **jsoncallback=?** onto the url when making the jQuery Ajax call to any JSON webservice eg

```
$.getJSON("http:// api.apieco.ir/geoplugin /json.gp?jsoncallback=?",
function (data) {
        for (var i in data) {
                document.write('data["i"] = ' + i + '<br/>');
        }
);
```

An AJAX self-contained currency converter

Here is a working, extremely simple example of how to use the JSON service. Here, we create a live AJAX currency converter, with little more than a few lines of HTML.

```
<script src="http://www.google.com/jsapi"></script>
<script                      src="http://api.apieco.ir/geoplugin/javascript.gp"
type="text/javascript"></script>
<script                                     src="http://api.apieco.ir/geoplugin
/ajax_currency_converter.gp"></script>
Convert
<input type='text' id='gp_amount' size='4' />
<select id="gp_from"></select>
to
<select id="gp_to"></select>
<p><input type='button' onClick='gp_convertIt()' value = 'Convert It' /></p>
<script>gp_currencySymbols()</script>

<div id="gp_converted"></div>
```

which will output:

Convert to

World Clock API

Alpha Version .1

REST Services that will return current date/time in JSON for any registered time zone.

Eastern Standard Time

{"$id":"1","currentDateTime":"2019-04-17T02:42-04:00","utcOffset":"-04:00:00","isDayLightSavingsTime":true,"dayOfTheWeek":"Wednesday","timeZoneName":"Eastern Standard Time","currentFileTime":131999425243874734,"ordinalDate":"2019-107","serviceResponse":null}

Coordinated Universal Time

{"$id":"1","currentDateTime":"2019-04-17T06:42Z","utcOffset":"00:00:00","isDayLightSavingsTime":false,"dayOfTheWeek":"Wednesday","timeZoneName":"UTC","currentFileTime":131999569650543902,"ordinalDate":"2019-107","serviceResponse":null}

Central European Standard Time

mycallback({"$id":"1","currentDateTime":"2019-04-17T08:42+02:00","utcOffset":"02:00:00","isDayLightSavingsTime":true,"dayOfTheWeek":"Wednesday","timeZoneName":"Central Europe Standard Time","currentFileTime":131999641798049644,"ordinalDate":"2019-107","serviceResponse":null});

Crime Data

Crime Data API Overview

Crime data is great for safe neighbourhoods.

Source on Github

View source here: https://github.com/jgentes/crimedata

Proxy for viewing crime data from crimemapping.com based on latitude and longitude coordinates
Access the server at https://citizenrequests.herokuapp.com (may disappear at any time)

API instructions can be found here: https://www.mashape.com/jgentes/crime-data

Code Snippet
cURL

```
curl --get --include 'https://jgentes-Crime-Data-v1.p.apieco.com/crime?startdate=9%2F19%2F2015&enddate=9%2F25%2F2015&lat=37.757815&long=-122.5076392' \
  -H 'X-apieco-Host: jgentes-Crime-Data-v1.p.apieco.com' \
  -H 'X-apieco-Key: SIGN-UP-FOR-KEY'
```

Such Flight API Documentation

Flight Search API. Searches in Airblue. Other flights coming soon PIA, Shaheen Air.

Code Snippet

cURL

```
curl --get --include 'https://siddiq-such-flight-v1.p.apieco.com/search?return-
date=2015-04-07&to=LHE&depart-date=2015-03-31&from=DXB' \
  -H 'X-apieco-Host: siddiq-such-flight-v1.p.apieco.com' \
  -H 'X-apieco-Key: SIGN-UP-FOR-KEY'
```

DuckDuckGo Zero-click Info

DuckDuckGo Zero-click Info includes topic summaries, categories, disambiguation, official sites, !bang redirects, definitions and more. You can use this API for many things, e.g. define people, places, things, words and concepts; provides direct links to other services (via !bang syntax); list related topics; and gives official sites when available.

Code Snippet

cURL

```
curl --get --include 'https://duckduckgo-duckduckgo-zero-click-
info.p.apieco.com/?no_redirect=1&no_html=1&callback=process_duckduckgo&skip_disamb
ig=1&q=DuckDuckGo&format=json' \
  -H 'X-apieco-Host: duckduckgo-duckduckgo-zero-click-info.p.apieco.com' \
  -H 'X-apieco-Key: SIGN-UP-FOR-KEY'
```

ChicMic Test Login

These are the testing APIs for ChicMic Users to test the login functionality

Code Snippet

cURL

```
curl -X POST --include 'https://skumar-chicmic-chicmic-test-login-
v1.p.apieco.com/login' \
  -H 'X-apieco-Host: skumar-chicmic-chicmic-test-login-v1.p.apieco.com' \
  -H 'X-apieco-Key: SIGN-UP-FOR-KEY' \
  -H 'Content-Type: application/json' \
  --data-binary '{"deviceId":"device_1","deviceType":1}'
```

Football betting tips

Passionate about sports betting? Tipsxpert.com provides your daily free betting tips and a collection of the best free bets !

Code Snippet

cURL

```
curl --get --include 'https://scommetix-football-betting-tips-v1.p.apieco.com/betting-tips' \
  -H 'X-apieco-Host: scommetix-football-betting-tips-v1.p.apieco.com' \
  -H 'X-apieco-Key: SIGN-UP-FOR-KEY'
```

skicams

Italian Ski Webcam Database / API

- more at: http://skicams.it/api (italian)

Code Snippet
cURL

```
curl --get --include 'https://makevoid-skicams.p.apieco.com/cams.json' \
  -H 'X-apieco-Host: makevoid-skicams.p.apieco.com' \
  -H 'X-apieco-Key: SIGN-UP-FOR-KEY'
```

DND Checker - India

Check DND status of any valid mobile number before sending a SMS/Text Message to India using an up to date database of NDNC, NCCP, TRAI, DNC or NCPR.

Code Snippet

cURL

```
curl -X POST --include 'https://dnd.p.apieco.com/' \
  -H 'X-apieco-Host: dnd.p.apieco.com' \
  -H 'X-apieco-Key: SIGN-UP-FOR-KEY' \
  -H 'Content-Type: application/json' \
  --data-binary '{"mobile":"9999999999"}'
```

Movie Database (IMDB Alternative)

Access movie and TV information similar to that of IMDB. Get Title, Year, Metascore Rating, IMDB rating, Release date, Runtime, Genre, Directors, Writers, Actors, Plot, Awards, Posters & tons of other data for each title.

RESTful web service to access information, pictures, and more from the movie database.
Get Title, Year, Metascore Rating, IMDB rating, Release date, Runtime, Genre, Directors, Writers, Actors, Plot, Awards, Posters, IMDB ID, Type, DVD, Boxoffice, Production company, website and response data.

Movie Database API

Looking for a API to provide data on your favorite movies? Check out the IMDb Alternative API to help enrich your application. The API gets data from sources like IMDb and returns results in JSON or XML format. You can even search by IMDb ID to get similar results from the IMDb database. Check out the API endpoints to query different types of data including:

- lists of movie titles
- movie posters
- TV show episodes
- and much more!

How do you integrate the IMDb API into your application?

1. Sign up for a apieco account
2. Subscribe to a pricing plan (There's a free tier that offers 1000 requests/day).
3. Return to the API documentation/endpoints page and select a programming language of your choice from the dropdown. The available choices include: node.js, PHP, Python, Ruby, Objective-C, Java (Android), C# (.NET), and cURL.
4. Copy the code snippet and integrate it into your website, software or mobile application.

How do you apply for an API key?

When you sign up for a free apieco user account, you will automatically be given an API key.

Code Snippet
cURL

```
curl --get --include 'https://movie-database-imdb-
alternative.p.apieco.com/?page=1&r=json&s=Avengers+Endgame' \
  -H 'X-apieco-Host: movie-database-imdb-alternative.p.apieco.com' \
  -H 'X-apieco-Key: SIGN-UP-FOR-KEY'
```

Email Validator

Checks for fake DNS as well as uses regex functions to check the email for the right length and accepted characters. For example it will mark as valid an email like 'john@gmail.com' but it will recognize as a fake 'john@gmaill.com' (because of the not existing domain)

Code Snippet

cURL

```
curl --get --include 'https://pozzad-email-validator.p.apieco.com/emailvalidator/validateEmail/john%40gmail.com' \
  -H 'X-apieco-Host: pozzad-email-validator.p.apieco.com' \
  -H 'X-apieco-Key: SIGN-UP-FOR-KEY'
```

IMG4Me - Text to Image Service

IMG4Me is a free service to convert your text into image. You can use this service to prevent crawlers and robots from copying your email address, articles, or website contents. Meanwhile, you can handle encoding errors as well if your text is in non-English characters.

Code Snippet

cURL

```
curl --get --include
'https://img4me.p.apieco.com/?fcolor=000000&bcolor=FFFFFF&font=trebuchet&size=12&type=png&text=Test+Me' \
  -H 'X-apieco-Host: img4me.p.apieco.com' \
  -H 'X-apieco-Key: SIGN-UP-FOR-KEY'
```

Pinterest

This is an unofficial Pinterest API. Pinterest is a pinboard-style photo sharing website that allows users to create and manage theme-based image collections such as events, interests, hobbies, and more. Users can browse other pinboards for inspiration, 're-pin' images to their own pinboards, or 'like' photos. (Credits to …

Code Snippet

cURL

```
curl --get --include 'https://ismaelc-pinterest.p.apieco.com/jwmoz/pins?page=2' \
  -H 'X-apieco-Host: ismaelc-pinterest.p.apieco.com' \
  -H 'X-apieco-Key: SIGN-UP-FOR-KEY'
```

Youtube To Mp3 Download

Convert & Download single video or multiple Youtube videos in Mp3 format using our youtube to mp3 API. Multiple Embedding features are supported like html link, javascript, iframe. Its free also provide affiliate programs.

Code Snippet

cURL

```
curl --get --include 'https://coolguruji-youtube-to-mp3-download-
v1.p.apieco.com/?id=lF-jPBnZ098' \
  -H 'X-apieco-Host: coolguruji-youtube-to-mp3-download-v1.p.apieco.com' \
  -H 'X-apieco-Key: SIGN-UP-FOR-KEY'
```

YandexStatic Package

Yandex is a Russian multinational technology company specializing in Internet-related services and products. Yandex operates the largest search engine in Russia with about 65% market share in that country.Yandex operates the largest search engine in Russia with about 65% market share in that country.The Static API generates a map image based on the parameter values passed to the service.

- Domain: yandex.com

How to get credentials:

1. Navigate to Developers Console.
2. Create API app.

Custom datatypes:

Dataty pe	Description	Example
Datepicker	String which includes date and time	`2016-05-28 00:00:00`
Map	String which includes latitude and longitude coma separated	`50.37, 26.56`
List	Simple array	`["123", "sample"]`
Select	String with predefined values	`sample`
Array	Array of objects	`[{"Second name":"123","Age":"12","Photo":"sdf","Draft":"sdfsdf"},{"name":"adi","Second name":"bla","Age":"4","Photo":"asfserwe","Draft":"sdfsdf"}]`

YandexStatic.getStaticMap

The Static API generates a map image in accordance with the values of the parameters passed to the service.

Field	Type	Description
mapType	List	The list of layers that determine the type of map.
mapCenter	Map	Longitude and latitude of the center of the map in degrees.
viewportRange	String	The length of the map display area by longitude and latitude (in degrees).Can not be used with zoom parameter.
zoom	Number	The zoom level of the map (0-17).Can not be used with viewportRange parameter.
size	String	The width and height of the requested map image (in pixels), see Map size. The default value is 650x450.
scale	String	The coefficient of magnification of objects on the map. Can take a fractional value from 1.0 to 4.0.
markersDefinitions	List	Contains descriptions of one or more labels that you want to display on the map. Example `32.810152,39.889847,pm2rdl1`. For more information see documentation.
geoFiguresDefinitions	List	Contains a set of descriptions of geometric shapes (polygons and polygons) that you want to display on the map.Example `pl=c:ec473fFF,f:00FF00A0,w:5,37.51,55.83,37.67,55.82,37.66,55.74,37.49,55.70,37.51,55.83`. For more information see documentation.
lang	String	API allows you to display maps, localized in different languages, taking into account the specifics of individual countries.

Field	Type	Description
showTraffic	List	Show traffic.

Trumail Package

Prevent bounced emails and low-quality users with FREE professional grade email verification.Trumail is a fast and accurate email address verification API written entirely in Go. It was built with the intention of providing software engineers and businesses with a simple and easy to use solution to bounced emails.

- Domain: trumail.io

Custom datatypes:

Datatype	Description	Example
Datepicker	String which includes date and time	`2016-05-28 00:00:00`
Map	String which includes latitude and longitude coma separated	`50.37, 26.56`
List	Simple array	`["123", "sample"]`
Select	String with predefined values	`sample`
Array	Array of objects	`[{"Second name":"123","Age":"12","Photo":"sdf","Draft":"sdfsdf"},{"name":"adi","Second name":"bla","Age":"4","Photo":"asfserwe","Draft":"sdfsdf"}]`

Trumail.verifyEmail

The speed of each verification is dependent on the speed of the mail server.Slow servers will always result in slow verifications.

Field	Type	Description
email	String	Email for verification.