geoip2 Documentation

Release 4.6.0

Gregory Oschwald

Jun 21, 2022

Contents

1 Description **1**

2 Installation **3**
2.1 Database Reader Extension 3

3 IP Geolocation Usage **5**

4 Web Service Usage **7**

5 Sync Web Service Example **9**

6 Async Web Service Example **11**

7 Web Service Client Exceptions **13**

8 Database Usage **15**

9 Database Example **17**
9.1 City Database 17
9.2 Anonymous IP Database 18
9.3 ASN Database 18
9.4 Connection-Type Database 19
9.5 Domain Database 19
9.6 Enterprise Database 19
9.7 ISP Database 20

10 Database Reader Exceptions **21**

11 Values to use for Database or Dictionary Keys **23**

12 What data is returned? **25**

13 Integration with GeoNames **27**

14 Reporting Data Problems **29**

15 Requirements **31**

16 Versioning **33**

17 Support **35**

18 Modules **37**
18.1 GeoIP2 Database Reader . . . 37
18.2 WebServices Client API . . . 38
18.2.1 SSL . . . 39
18.3 Models . . . 41
18.4 Records . . . 47
18.5 Errors . . . 55

19 Indices and tables **57**

Python Module Index **59**

Index **61**

CHAPTER 1

Description

This package provides an API for the GeoIP2 and GeoLite2 web services and databases.

CHAPTER 2

Installation

To install the `geoip2` module, type:

```
$ pip install geoip2
```

If you are not able to use pip, you may also use easy_install from the source directory:

```
$ easy_install .
```

2.1 Database Reader Extension

If you wish to use the C extension for the database reader, you must first install the libmaxminddb C API. Please see the instructions distributed with it.

CHAPTER 3

IP Geolocation Usage

IP geolocation is inherently imprecise. Locations are often near the center of the population. Any location provided by a GeoIP2 database or web service should not be used to identify a particular address or household.

CHAPTER 4

Web Service Usage

To use this API, you first construct either a `geoip2.webservice.Client` or `geoip2.webservice.AsyncClient`, passing your MaxMind `account_id` and `license_key` to the constructor. To use the GeoLite2 web service instead of the GeoIP2 web service, set the optional `host` keyword argument to `geolite.info`.

After doing this, you may call the method corresponding to request type (e.g., `city` or `country`), passing it the IP address you want to look up.

If the request succeeds, the method call will return a model class for the endpoint you called. This model in turn contains multiple record classes, each of which represents part of the data returned by the web service.

If the request fails, the client class throws an exception.

CHAPTER 5

Sync Web Service Example

```
>>> import geoip2.webservice
>>>
>>> # This creates a Client object that can be reused across requests.
>>> # Replace "42" with your account ID and "license_key" with your license
>>> # key. Set the "host" keyword argument to "geolite.info" to use the
>>> # GeoLite2 web service instead of the GeoIP2 web service.
>>> with geoip2.webservice.Client(42, 'license_key') as client:
>>>
>>>     # Replace "city" with the method corresponding to the web service
>>>     # that you are using, i.e., "country", "city", or "insights". Please
>>>     # note that Insights is not supported by the GeoLite2 web service.
>>>     response = client.city('203.0.113.0')
>>>
>>>     response.country.iso_code
'US'
>>>     response.country.name
'United States'
>>>     response.country.names['zh-CN']
u''
>>>
>>>     response.subdivisions.most_specific.name
'Minnesota'
>>>     response.subdivisions.most_specific.iso_code
'MN'
>>>
>>>     response.city.name
'Minneapolis'
>>>
>>>     response.postal.code
'55455'
>>>
>>>     response.location.latitude
44.9733
>>>     response.location.longitude
```

(continues on next page)

(continued from previous page)

```
-93.2323
>>>
>>>     response.traits.network
IPv4Network('203.0.113.0/32')
```

CHAPTER 6

Async Web Service Example

```
>>> import asyncio
>>>
>>> import geoip2.webservice
>>>
>>> async def main():
>>>     # This creates an AsyncClient object that can be reused across
>>>     # requests on the running event loop. If you are using multiple event
>>>     # loops, you must ensure the object is not used on another loop.
>>>     #
>>>     # Replace "42" with your account ID and "license_key" with your license
>>>     # key. Set the "host" keyword argument to "geolite.info" to use the
>>>     # GeoLite2 web service instead of the GeoIP2 web service.
>>>     async with geoip2.webservice.AsyncClient(42, 'license_key') as client:
>>>
>>>         # Replace "city" with the method corresponding to the web service
>>>         # that you are using, i.e., "country", "city", or "insights". Please
>>>         # note that Insights is not supported by the GeoLite2 web service.
>>>         response = await client.city('203.0.113.0')
>>>
>>>         response.country.iso_code
'US'
>>>         response.country.name
'United States'
>>>         response.country.names['zh-CN']
u''
>>>
>>>         response.subdivisions.most_specific.name
'Minnesota'
>>>         response.subdivisions.most_specific.iso_code
'MN'
>>>
>>>         response.city.name
'Minneapolis'
>>>
```

(continues on next page)

(continued from previous page)

```
>>>         response.postal.code
'55455'
>>>
>>>         response.location.latitude
44.9733
>>>         response.location.longitude
-93.2323
>>>
>>>         response.traits.network
IPv4Network('203.0.113.0/32')
>>>
>>> asyncio.run(main())
```

CHAPTER 7

Web Service Client Exceptions

For details on the possible errors returned by the web service itself, see https://dev.maxmind.com/geoip/docs/web-services?lang=en for the GeoIP2 web service docs.

If the web service returns an explicit error document, this is thrown as a `AddressNotFoundError`, `AuthenticationError`, `InvalidRequestError`, or `OutOfQueriesError` as appropriate. These all subclass `GeoIP2Error`.

If some other sort of error occurs, this is thrown as an `HTTPError`. This is thrown when some sort of unanticipated error occurs, such as the web service returning a 500 or an invalid error document. If the web service returns any status code besides 200, 4xx, or 5xx, this also becomes an `HTTPError`.

Finally, if the web service returns a 200 but the body is invalid, the client throws a `GeoIP2Error`.

CHAPTER 8

Database Usage

To use the database API, you first construct a `geoip2.database.Reader` using the path to the file as the first argument. After doing this, you may call the method corresponding to database type (e.g., `city` or `country`), passing it the IP address you want to look up.

If the lookup succeeds, the method call will return a model class for the database method you called. This model in turn contains multiple record classes, each of which represents part of the data for the record.

If the request fails, the reader class throws an exception.

CHAPTER 9

Database Example

9.1 City Database

```
>>> import geoip2.database
>>>
>>> # This creates a Reader object. You should use the same object
>>> # across multiple requests as creation of it is expensive.
>>> with geoip2.database.Reader('/path/to/GeoLite2-City.mmdb') as reader:
>>>
>>>     # Replace "city" with the method corresponding to the database
>>>     # that you are using, e.g., "country".
>>>     response = reader.city('203.0.113.0')
>>>
>>>     response.country.iso_code
'US'
>>>     response.country.name
'United States'
>>>     response.country.names['zh-CN']
u''
>>>
>>>     response.subdivisions.most_specific.name
'Minnesota'
>>>     response.subdivisions.most_specific.iso_code
'MN'
>>>
>>>     response.city.name
'Minneapolis'
>>>
>>>     response.postal.code
'55455'
>>>
>>>     response.location.latitude
44.9733
>>>     response.location.longitude
```

(continues on next page)

(continued from previous page)

```
-93.2323
>>>
>>>     response.traits.network
IPv4Network('203.0.113.0/24')
```

9.2 Anonymous IP Database

```
>>> import geoip2.database
>>>
>>> # This creates a Reader object. You should use the same object
>>> # across multiple requests as creation of it is expensive.
>>> with geoip2.database.Reader('/path/to/GeoIP2-Anonymous-IP.mmdb') as reader:
>>>
>>>     response = reader.anonymous_ip('203.0.113.0')
>>>
>>>     response.is_anonymous
True
>>>     response.is_anonymous_vpn
False
>>>     response.is_hosting_provider
False
>>>     response.is_public_proxy
False
>>>     response.is_residential_proxy
False
>>>     response.is_tor_exit_node
True
>>>     response.ip_address
'203.0.113.0'
>>>     response.network
IPv4Network('203.0.113.0/24')
```

9.3 ASN Database

```
>>> import geoip2.database
>>>
>>> # This creates a Reader object. You should use the same object
>>> # across multiple requests as creation of it is expensive.
>>> with geoip2.database.Reader('/path/to/GeoLite2-ASN.mmdb') as reader:
>>>     response = reader.asn('203.0.113.0')
>>>     response.autonomous_system_number
1221
>>>     response.autonomous_system_organization
'Telstra Pty Ltd'
>>>     response.ip_address
'203.0.113.0'
>>>     response.network
IPv4Network('203.0.113.0/24')
```

9.4 Connection-Type Database

```
>>> import geoip2.database
>>>
>>> # This creates a Reader object. You should use the same object
>>> # across multiple requests as creation of it is expensive.
>>> with geoip2.database.Reader('/path/to/GeoIP2-Connection-Type.mmdb') as reader:
>>>     response = reader.connection_type('203.0.113.0')
>>>     response.connection_type
'Corporate'
>>>     response.ip_address
'203.0.113.0'
>>>     response.network
IPv4Network('203.0.113.0/24')
```

9.5 Domain Database

```
>>> import geoip2.database
>>>
>>> # This creates a Reader object. You should use the same object
>>> # across multiple requests as creation of it is expensive.
>>> with geoip2.database.Reader('/path/to/GeoIP2-Domain.mmdb') as reader:
>>>     response = reader.domain('203.0.113.0')
>>>     response.domain
'umn.edu'
>>>     response.ip_address
'203.0.113.0'
```

9.6 Enterprise Database

```
>>> import geoip2.database
>>>
>>> # This creates a Reader object. You should use the same object
>>> # across multiple requests as creation of it is expensive.
>>> with geoip2.database.Reader('/path/to/GeoIP2-Enterprise.mmdb') as reader:
>>>
>>>     # Use the .enterprise method to do a lookup in the Enterprise database
>>>     response = reader.enterprise('203.0.113.0')
>>>
>>>     response.country.confidence
99
>>>     response.country.iso_code
'US'
>>>     response.country.name
'United States'
>>>     response.country.names['zh-CN']
u''
>>>
>>>     response.subdivisions.most_specific.name
'Minnesota'
>>>     response.subdivisions.most_specific.iso_code
```

(continues on next page)

(continued from previous page)

```
'MN'
>>>     response.subdivisions.most_specific.confidence
77
>>>
>>>     response.city.name
'Minneapolis'
>>>     response.country.confidence
11
>>>
>>>     response.postal.code
'55455'
>>>
>>>     response.location.accuracy_radius
50
>>>     response.location.latitude
44.9733
>>>     response.location.longitude
-93.2323
>>>
>>>     response.traits.network
IPv4Network('203.0.113.0/24')
```

9.7 ISP Database

```
>>> import geoip2.database
>>>
>>> # This creates a Reader object. You should use the same object
>>> # across multiple requests as creation of it is expensive.
>>> with geoip2.database.Reader('/path/to/GeoIP2-ISP.mmdb') as reader:
>>>     response = reader.isp('203.0.113.0')
>>>     response.autonomous_system_number
1221
>>>     response.autonomous_system_organization
'Telstra Pty Ltd'
>>>     response.isp
'Telstra Internet'
>>>     response.organization
'Telstra Internet'
>>>     response.ip_address
'203.0.113.0'
>>>     response.network
IPv4Network('203.0.113.0/24')
```

CHAPTER 10

Database Reader Exceptions

If the database file does not exist or is not readable, the constructor will raise a `FileNotFoundError` or a `PermissionError`. If the IP address passed to a method is invalid, a `ValueError` will be raised. If the file is invalid or there is a bug in the reader, a `maxminddb.InvalidDatabaseError` will be raised with a description of the problem. If an IP address is not in the database, a `AddressNotFoundError` will be raised.

`AddressNotFoundError` references the largest subnet where no address would be found. This can be used to efficiently enumerate entire subnets:

```
import geoip2.database
import geoip2.errors
import ipaddress

# This creates a Reader object. You should use the same object
# across multiple requests as creation of it is expensive.
with geoip2.database.Reader('/path/to/GeoLite2-ASN.mmdb') as reader:
    network = ipaddress.ip_network("192.128.0.0/15")

    ip_address = network[0]
    while ip_address in network:
        try:
            response = reader.asn(ip_address)
            response_network = response.network
        except geoip2.errors.AddressNotFoundError as e:
            response = None
            response_network = e.network
        print(f"{response_network}: {response!r}")
        ip_address = response_network[-1] + 1  # move to next subnet
```

CHAPTER 11

Values to use for Database or Dictionary Keys

We strongly discourage you from using a value from any ‘‘names‘‘ property as a key in a database or dictionaries.

These names may change between releases. Instead we recommend using one of the following:

- `geoip2.records.City` - `city.geoname_id`
- `geoip2.records.Continent` - `continent.code` or `continent.geoname_id`
- `geoip2.records.Country` and `geoip2.records.RepresentedCountry` - `country.iso_code` or `country.geoname_id`
- `geoip2.records.subdivision` - `subdivision.iso_code` or `subdivision.geoname_id`

CHAPTER 12

What data is returned?

While many of the models contain the same basic records, the attributes which can be populated vary between web service endpoints or databases. In addition, while a model may offer a particular piece of data, MaxMind does not always have every piece of data for any given IP address.

Because of these factors, it is possible for any request to return a record where some or all of the attributes are unpopulated.

The only piece of data which is always returned is the `ip_address` attribute in the `geoip2.records.Traits` record.

CHAPTER 13

Integration with GeoNames

GeoNames offers web services and downloadable databases with data on geographical features around the world, including populated places. They offer both free and paid premium data. Each feature is uniquely identified by a `geoname_id`, which is an integer.

Many of the records returned by the GeoIP web services and databases include a `geoname_id` field. This is the ID of a geographical feature (city, region, country, etc.) in the GeoNames database.

Some of the data that MaxMind provides is also sourced from GeoNames. We source things like place names, ISO codes, and other similar data from the GeoNames premium data set.

CHAPTER 14

Reporting Data Problems

If the problem you find is that an IP address is incorrectly mapped, please submit your correction to MaxMind.

If you find some other sort of mistake, like an incorrect spelling, please check the GeoNames site first. Once you've searched for a place and found it on the GeoNames map view, there are a number of links you can use to correct data ("move", "edit", "alternate names", etc.). Once the correction is part of the GeoNames data set, it will be automatically incorporated into future MaxMind releases.

If you are a paying MaxMind customer and you're not sure where to submit a correction, please contact MaxMind support for help.

CHAPTER 15

Requirements

Python 3.6 or greater is required. Older versions are not supported.

The Requests HTTP library is also required. See <http://python-requests.org> for details.

CHAPTER 16

Versioning

The GeoIP2 Python API uses Semantic Versioning.

CHAPTER 17

Support

Please report all issues with this code using the GitHub issue tracker

If you are having an issue with a MaxMind service that is not specific to the client API, please contact MaxMind support for assistance.

CHAPTER 18

Modules

18.1 GeoIP2 Database Reader

class geoip2.database.**Reader**(*fileish: Union[AnyStr, int, os.PathLike, IO], locales: Optional[List[str]] = None, mode: int = 0*)

GeoIP2 database Reader object.

Instances of this class provide a reader for the GeoIP2 database format. IP addresses can be looked up using the `country` and `city` methods.

The basic API for this class is the same for every database. First, you create a reader object, specifying a file name or file descriptor. You then call the method corresponding to the specific database, passing it the IP address you want to look up.

If the request succeeds, the method call will return a model class for the method you called. This model in turn contains multiple record classes, each of which represents part of the data returned by the database. If the database does not contain the requested information, the attributes on the record class will have a `None` value.

If the address is not in the database, an `geoip2.errors.AddressNotFoundError` exception will be thrown. If the database is corrupt or invalid, a `maxminddb.InvalidDatabaseError` will be thrown.

anonymous_ip(*ip_address: Union[str, ipaddress.IPv6Address, ipaddress.IPv4Address]*) → geoip2.models.AnonymousIP

Get the AnonymousIP object for the IP address.

Parameters **ip_address** – IPv4 or IPv6 address as a string.

Returns *geoip2.models.AnonymousIP* object

asn(*ip_address: Union[str, ipaddress.IPv6Address, ipaddress.IPv4Address]*) → geoip2.models.ASN

Get the ASN object for the IP address.

Parameters **ip_address** – IPv4 or IPv6 address as a string.

Returns *geoip2.models.ASN* object

city(*ip_address: Union[str, ipaddress.IPv6Address, ipaddress.IPv4Address]*) → geoip2.models.City

Get the City object for the IP address.

Parameters **ip_address** – IPv4 or IPv6 address as a string.

Returns *geoip2.models.City* object

close() → None
Closes the GeoIP2 database.

connection_type(*ip_address: Union[str, ipaddress.IPv6Address, ipaddress.IPv4Address]*) → geoip2.models.ConnectionType
Get the ConnectionType object for the IP address.

Parameters **ip_address** – IPv4 or IPv6 address as a string.

Returns *geoip2.models.ConnectionType* object

country(*ip_address: Union[str, ipaddress.IPv6Address, ipaddress.IPv4Address]*) → geoip2.models.Country
Get the Country object for the IP address.

Parameters **ip_address** – IPv4 or IPv6 address as a string.

Returns *geoip2.models.Country* object

domain(*ip_address: Union[str, ipaddress.IPv6Address, ipaddress.IPv4Address]*) → geoip2.models.Domain
Get the Domain object for the IP address.

Parameters **ip_address** – IPv4 or IPv6 address as a string.

Returns *geoip2.models.Domain* object

enterprise(*ip_address: Union[str, ipaddress.IPv6Address, ipaddress.IPv4Address]*) → geoip2.models.Enterprise
Get the Enterprise object for the IP address.

Parameters **ip_address** – IPv4 or IPv6 address as a string.

Returns *geoip2.models.Enterprise* object

isp(*ip_address: Union[str, ipaddress.IPv6Address, ipaddress.IPv4Address]*) → geoip2.models.ISP
Get the ISP object for the IP address.

Parameters **ip_address** – IPv4 or IPv6 address as a string.

Returns *geoip2.models.ISP* object

metadata() → maxminddb.reader.Metadata
The metadata for the open database.

Returns `maxminddb.reader.Metadata` object

18.2 WebServices Client API

This class provides a client API for all the GeoIP2 web services. The web services are Country, City Plus, and Insights. Each service returns a different set of data about an IP address, with Country returning the least data and Insights the most.

Each service is represented by a different model class, and these model classes in turn contain multiple record classes. The record classes have attributes which contain data about the IP address.

If the service does not return a particular piece of data for an IP address, the associated attribute is not populated.

The service may not return any information for an entire record, in which case all of the attributes for that record class will be empty.

18.2.1 SSL

Requests to the web service are always made with SSL.

class geoip2.webservice.**AsyncClient**(*account_id: int*, *license_key: str*, *host: str = 'geoip.maxmind.com'*, *locales: Optional[List[str]] = None*, *timeout: float = 60*, *proxy: Optional[str] = None*)

An async GeoIP2 client.

It accepts the following required arguments:

Parameters

- **account_id** – Your MaxMind account ID.
- **license_key** – Your MaxMind license key.

Go to https://www.maxmind.com/en/my_license_key to see your MaxMind account ID and license key.

The following keyword arguments are also accepted:

Parameters

- **host** – The hostname to make a request against. This defaults to "geoip.maxmind.com". To use the GeoLite2 web service instead of the GeoIP2 web service, set this to "geolite.info".
- **locales** – This is list of locale codes. This argument will be passed on to record classes to use when their name properties are called. The default value is ['en'].

  The order of the locales is significant. When a record class has multiple names (country, city, etc.), its name property will return the name in the first locale that has one.

  Note that the only locale which is always present in the GeoIP2 data is "en". If you do not include this locale, the name property may end up returning None even when the record has an English name.

  Currently, the valid locale codes are:

  - de – German
  - en – English names may still include accented characters if that is the accepted spelling in English. In other words, English does not mean ASCII.
  - es – Spanish
  - fr – French
  - ja – Japanese
  - pt-BR – Brazilian Portuguese
  - ru – Russian
  - zh-CN – Simplified Chinese.
- **timeout** – The timeout in seconds to use when waiting on the request. This sets both the connect timeout and the read timeout. The default is 60.
- **proxy** – The URL of an HTTP proxy to use. It may optionally include a basic auth username and password, e.g., `http://username:password@host:port`.

city(*ip_address: Union[str, ipaddress.IPv6Address, ipaddress.IPv4Address] = 'me'*) → geoip2.models.City

Call City Plus endpoint with the specified IP.

Parameters **ip_address** – IPv4 or IPv6 address as a string. If no address is provided, the address that the web service is called from will be used.

Returns *geoip2.models.City* object

close()
Close underlying session

This will close the session and any associated connections.

country(*ip_address: Union[str, ipaddress.IPv6Address, ipaddress.IPv4Address] = 'me'*) → geoip2.models.Country
Call the GeoIP2 Country endpoint with the specified IP.

Parameters **ip_address** – IPv4 or IPv6 address as a string. If no address is provided, the address that the web service is called from will be used.

Returns *geoip2.models.Country* object

insights(*ip_address: Union[str, ipaddress.IPv6Address, ipaddress.IPv4Address] = 'me'*) → geoip2.models.Insights
Call the Insights endpoint with the specified IP.

Insights is only supported by the GeoIP2 web service. The GeoLite2 web service does not support it.

Parameters **ip_address** – IPv4 or IPv6 address as a string. If no address is provided, the address that the web service is called from will be used.

Returns *geoip2.models.Insights* object

class geoip2.webservice.**Client**(*account_id: int, license_key: str, host: str = 'geoip.maxmind.com', locales: Optional[List[str]] = None, timeout: float = 60, proxy: Optional[str] = None*)
A synchronous GeoIP2 client.

It accepts the following required arguments:

Parameters

- **account_id** – Your MaxMind account ID.
- **license_key** – Your MaxMind license key.

Go to https://www.maxmind.com/en/my_license_key to see your MaxMind account ID and license key.

The following keyword arguments are also accepted:

Parameters

- **host** – The hostname to make a request against. This defaults to "geoip.maxmind.com". To use the GeoLite2 web service instead of the GeoIP2 web service, set this to "geolite.info".
- **locales** – This is list of locale codes. This argument will be passed on to record classes to use when their name properties are called. The default value is ['en'].

  The order of the locales is significant. When a record class has multiple names (country, city, etc.), its name property will return the name in the first locale that has one.

  Note that the only locale which is always present in the GeoIP2 data is "en". If you do not include this locale, the name property may end up returning None even when the record has an English name.

  Currently, the valid locale codes are:

  – de – German

  – en – English names may still include accented characters if that is the accepted spelling in English. In other words, English does not mean ASCII.

  – es – Spanish

- fr – French
- ja – Japanese
- pt-BR – Brazilian Portuguese
- ru – Russian
- zh-CN – Simplified Chinese.

- **timeout** – The timeout in seconds to use when waiting on the request. This sets both the connect timeout and the read timeout. The default is 60.
- **proxy** – The URL of an HTTP proxy to use. It may optionally include a basic auth username and password, e.g., `http://username:password@host:port`.

city(*ip_address: Union[str, ipaddress.IPv6Address, ipaddress.IPv4Address] = 'me'*) → geoip2.models.City

Call City Plus endpoint with the specified IP.

Parameters **ip_address** – IPv4 or IPv6 address as a string. If no address is provided, the address that the web service is called from will be used.

Returns *geoip2.models.City* object

close()

Close underlying session

This will close the session and any associated connections.

country(*ip_address: Union[str, ipaddress.IPv6Address, ipaddress.IPv4Address] = 'me'*) → geoip2.models.Country

Call the GeoIP2 Country endpoint with the specified IP.

Parameters **ip_address** – IPv4 or IPv6 address as a string. If no address is provided, the address that the web service is called from will be used.

Returns *geoip2.models.Country* object

insights(*ip_address: Union[str, ipaddress.IPv6Address, ipaddress.IPv4Address] = 'me'*) → geoip2.models.Insights

Call the Insights endpoint with the specified IP.

Insights is only supported by the GeoIP2 web service. The GeoLite2 web service does not support it.

Parameters **ip_address** – IPv4 or IPv6 address as a string. If no address is provided, the address that the web service is called from will be used.

Returns *geoip2.models.Insights* object

18.3 Models

These classes provide models for the data returned by the GeoIP2 web service and databases.

The only difference between the City and Insights model classes is which fields in each record may be populated. See https://dev.maxmind.com/geoip/docs/web-services?lang=en for more details.

class geoip2.models.**ASN**(*raw: Dict[str, Union[str, int]]*)

Model class for the GeoLite2 ASN.

This class provides the following attribute:

autonomous_system_number

The autonomous system number associated with the IP address.

Type int

autonomous_system_organization
The organization associated with the registered autonomous system number for the IP address.

Type str

ip_address
The IP address used in the lookup.

Type str

network
The network associated with the record. In particular, this is the largest network where all of the fields besides ip_address have the same value.

Type ipaddress.IPv4Network or ipaddress.IPv6Network

class geoip2.models.**AnonymousIP**(*raw: Dict[str, bool]*)
Model class for the GeoIP2 Anonymous IP.

This class provides the following attribute:

is_anonymous
This is true if the IP address belongs to any sort of anonymous network.

Type bool

is_anonymous_vpn
This is true if the IP address is registered to an anonymous VPN provider.

If a VPN provider does not register subnets under names associated with them, we will likely only flag their IP ranges using the `is_hosting_provider` attribute.

Type bool

is_hosting_provider
This is true if the IP address belongs to a hosting or VPN provider (see description of `is_anonymous_vpn` attribute).

Type bool

is_public_proxy
This is true if the IP address belongs to a public proxy.

Type bool

is_residential_proxy
This is true if the IP address is on a suspected anonymizing network and belongs to a residential ISP.

Type bool

is_tor_exit_node
This is true if the IP address is a Tor exit node.

Type bool

ip_address
The IP address used in the lookup.

Type str

network
The network associated with the record. In particular, this is the largest network where all of the fields besides ip_address have the same value.

Type ipaddress.IPv4Network or ipaddress.IPv6Network

class geoip2.models.**City**(*raw_response: Dict[str, Any], locales: Optional[List[str]] = None*)

Model for the City Plus web service and the City database.

city

City object for the requested IP address.

Type *geoip2.records.City*

continent

Continent object for the requested IP address.

Type *geoip2.records.Continent*

country

Country object for the requested IP address. This record represents the country where MaxMind believes the IP is located.

Type *geoip2.records.Country*

location

Location object for the requested IP address.

Type *geoip2.records.Location*

maxmind

Information related to your MaxMind account.

Type *geoip2.records.MaxMind*

postal

Postal object for the requested IP address.

Type *geoip2.records.Postal*

registered_country

The registered country object for the requested IP address. This record represents the country where the ISP has registered a given IP block in and may differ from the user's country.

Type *geoip2.records.Country*

represented_country

Object for the country represented by the users of the IP address when that country is different than the country in `country`. For instance, the country represented by an overseas military base.

Type *geoip2.records.RepresentedCountry*

subdivisions

Object (tuple) representing the subdivisions of the country to which the location of the requested IP address belongs.

Type *geoip2.records.Subdivisions*

traits

Object with the traits of the requested IP address.

Type *geoip2.records.Traits*

class geoip2.models.**ConnectionType**(*raw: Dict[str, Union[str, int]]*)

Model class for the GeoIP2 Connection-Type.

This class provides the following attribute:

connection_type

The connection type may take the following values:

- Dialup
- Cable/DSL
- Corporate
- Cellular

Additional values may be added in the future.

Type str

ip_address
The IP address used in the lookup.

Type str

network
The network associated with the record. In particular, this is the largest network where all of the fields besides ip_address have the same value.

Type ipaddress.IPv4Network or ipaddress.IPv6Network

class geoip2.models.**Country**(*raw_response: Dict[str, Any], locales: Optional[List[str]] = None*)
Model for the Country web service and Country database.

This class provides the following attributes:

continent
Continent object for the requested IP address.

Type *geoip2.records.Continent*

country
Country object for the requested IP address. This record represents the country where MaxMind believes the IP is located.

Type *geoip2.records.Country*

maxmind
Information related to your MaxMind account.

Type *geoip2.records.MaxMind*

registered_country
The registered country object for the requested IP address. This record represents the country where the ISP has registered a given IP block in and may differ from the user's country.

Type *geoip2.records.Country*

represented_country
Object for the country represented by the users of the IP address when that country is different than the country in `country`. For instance, the country represented by an overseas military base.

Type *geoip2.records.RepresentedCountry*

traits
Object with the traits of the requested IP address.

Type *geoip2.records.Traits*

class geoip2.models.**Domain**(*raw: Dict[str, Union[str, int]]*)
Model class for the GeoIP2 Domain.

This class provides the following attribute:

domain
The domain associated with the IP address.

Type str

ip_address
The IP address used in the lookup.

Type str

network
The network associated with the record. In particular, this is the largest network where all of the fields besides ip_address have the same value.

Type ipaddress.IPv4Network or ipaddress.IPv6Network

class geoip2.models.**Enterprise**(*raw_response: Dict[str, Any], locales: Optional[List[str]] = None*)
Model for the GeoIP2 Enterprise database.

city
City object for the requested IP address.

Type *geoip2.records.City*

continent
Continent object for the requested IP address.

Type *geoip2.records.Continent*

country
Country object for the requested IP address. This record represents the country where MaxMind believes the IP is located.

Type *geoip2.records.Country*

location
Location object for the requested IP address.

maxmind
Information related to your MaxMind account.

Type *geoip2.records.MaxMind*

registered_country
The registered country object for the requested IP address. This record represents the country where the ISP has registered a given IP block in and may differ from the user's country.

Type *geoip2.records.Country*

represented_country
Object for the country represented by the users of the IP address when that country is different than the country in `country`. For instance, the country represented by an overseas military base.

Type *geoip2.records.RepresentedCountry*

subdivisions
Object (tuple) representing the subdivisions of the country to which the location of the requested IP address belongs.

Type *geoip2.records.Subdivisions*

traits
Object with the traits of the requested IP address.

Type *geoip2.records.Traits*

class geoip2.models.**ISP**(*raw: Dict[str, Union[str, int]]*)

Model class for the GeoIP2 ISP.

This class provides the following attribute:

autonomous_system_number
The autonomous system number associated with the IP address.

Type int

autonomous_system_organization
The organization associated with the registered autonomous system number for the IP address.

Type str

isp
The name of the ISP associated with the IP address.

Type str

organization
The name of the organization associated with the IP address.

Type str

ip_address
The IP address used in the lookup.

Type str

network
The network associated with the record. In particular, this is the largest network where all of the fields besides ip_address have the same value.

Type ipaddress.IPv4Network or ipaddress.IPv6Network

class geoip2.models.**Insights**(*raw_response: Dict[str, Any], locales: Optional[List[str]] = None*)

Model for the GeoIP2 Insights web service.

city
City object for the requested IP address.

Type *geoip2.records.City*

continent
Continent object for the requested IP address.

Type *geoip2.records.Continent*

country
Country object for the requested IP address. This record represents the country where MaxMind believes the IP is located.

Type *geoip2.records.Country*

location
Location object for the requested IP address.

maxmind
Information related to your MaxMind account.

Type *geoip2.records.MaxMind*

registered_country
The registered country object for the requested IP address. This record represents the country where the ISP has registered a given IP block in and may differ from the user's country.

Type *geoip2.records.Country*

represented_country
Object for the country represented by the users of the IP address when that country is different than the country in `country`. For instance, the country represented by an overseas military base.

Type *geoip2.records.RepresentedCountry*

subdivisions
Object (tuple) representing the subdivisions of the country to which the location of the requested IP address belongs.

Type *geoip2.records.Subdivisions*

traits
Object with the traits of the requested IP address.

Type *geoip2.records.Traits*

class geoip2.models.**SimpleModel**(*raw: Dict[str, Union[bool, str, int]]*)
Provides basic methods for non-location models

network
The network for the record

18.4 Records

class geoip2.records.**City**(*locales: Optional[List[str]] = None*, *confidence: Optional[int] = None*, *geoname_id: Optional[int] = None*, *names: Optional[Dict[str, str]] = None*, ***_*)
Contains data for the city record associated with an IP address.

This class contains the city-level data associated with an IP address.

This record is returned by `city`, `enterprise`, and `insights`.

Attributes:

confidence
A value from 0-100 indicating MaxMind's confidence that the city is correct. This attribute is only available from the Insights end point and the Enterprise database.

Type int

geoname_id
The GeoName ID for the city.

Type int

name
The name of the city based on the locales list passed to the constructor.

Type str

names
A dictionary where the keys are locale codes and the values are names.

Type dict

class geoip2.records.**Continent**(*locales: Optional[List[str]] = None*, *code: Optional[str] = None*, *geoname_id: Optional[int] = None*, *names: Optional[Dict[str, str]] = None*, ***_*)
Contains data for the continent record associated with an IP address.

This class contains the continent-level data associated with an IP address.

Attributes:

code
A two character continent code like "NA" (North America) or "OC" (Oceania).

Type str

geoname_id
The GeoName ID for the continent.

Type int

name
Returns the name of the continent based on the locales list passed to the constructor.

Type str

names
A dictionary where the keys are locale codes and the values are names.

Type dict

class geoip2.records.**Country**(*locales: Optional[List[str]] = None*, *confidence: Optional[int] = None*, *geoname_id: Optional[int] = None*, *is_in_european_union: bool = False*, *iso_code: Optional[str] = None*, *names: Optional[Dict[str, str]] = None*, ***_*)
Contains data for the country record associated with an IP address.

This class contains the country-level data associated with an IP address.

Attributes:

confidence
A value from 0-100 indicating MaxMind's confidence that the country is correct. This attribute is only available from the Insights end point and the Enterprise database.

Type int

geoname_id
The GeoName ID for the country.

Type int

is_in_european_union
This is true if the country is a member state of the European Union.

Type bool

iso_code
The two-character ISO 3166-1 alpha code for the country.

Type str

name
The name of the country based on the locales list passed to the constructor.

Type str

names
A dictionary where the keys are locale codes and the values are names.

Type dict

class geoip2.records.**Location**(*average_income: Optional[int] = None*, *accuracy_radius: Optional[int] = None*, *latitude: Optional[float] = None*, *longitude: Optional[float] = None*, *metro_code: Optional[int] = None*, *population_density: Optional[int] = None*, *time_zone: Optional[str] = None*, **_)

Contains data for the location record associated with an IP address.

This class contains the location data associated with an IP address.

This record is returned by `city`, `enterprise`, and `insights`.

Attributes:

average_income
The average income in US dollars associated with the requested IP address. This attribute is only available from the Insights end point.

Type int

accuracy_radius
The approximate accuracy radius in kilometers around the latitude and longitude for the IP address. This is the radius where we have a 67% confidence that the device using the IP address resides within the circle centered at the latitude and longitude with the provided radius.

Type int

latitude
The approximate latitude of the location associated with the IP address. This value is not precise and should not be used to identify a particular address or household.

Type float

longitude
The approximate longitude of the location associated with the IP address. This value is not precise and should not be used to identify a particular address or household.

Type float

metro_code
The metro code of the location if the location is in the US. MaxMind returns the same metro codes as the Google AdWords API.

Type int

population_density

The estimated population per square kilometer associated with the IP address. This attribute is only available from the Insights end point.

Type int

time_zone
The time zone associated with location, as specified by the IANA Time Zone Database, e.g., "America/New_York".

Type str

class geoip2.records.**MaxMind**(*queries_remaining: Optional[int] = None*, **_)

Contains data related to your MaxMind account.

Attributes:

queries_remaining
The number of remaining queries you have for the end point you are calling.

Type int

class geoip2.records.**PlaceRecord**(*locales: Optional[List[str]] = None*, *names: Optional[Dict[str, str]] = None*)

All records with `names` subclass *PlaceRecord*.

name

Dict with locale codes as keys and localized name as value.

class geoip2.records.**Postal**(*code: Optional[str] = None*, *confidence: Optional[int] = None*, **_)

Contains data for the postal record associated with an IP address.

This class contains the postal data associated with an IP address.

This attribute is returned by `city`, `enterprise`, and `insights`.

Attributes:

code

The postal code of the location. Postal codes are not available for all countries. In some countries, this will only contain part of the postal code.

Type str

confidence

A value from 0-100 indicating MaxMind's confidence that the postal code is correct. This attribute is only available from the Insights end point and the Enterprise database.

Type int

class geoip2.records.**Record**

All records are subclasses of the abstract class `Record`.

class geoip2.records.**RepresentedCountry**(*locales: Optional[List[str]] = None*, *confidence: Optional[int] = None*, *geoname_id: Optional[int] = None*, *is_in_european_union: bool = False*, *iso_code: Optional[str] = None*, *names: Optional[Dict[str, str]] = None*, *type: Optional[str] = None*, **_)

Contains data for the represented country associated with an IP address.

This class contains the country-level data associated with an IP address for the IP's represented country. The represented country is the country represented by something like a military base.

Attributes:

confidence

A value from 0-100 indicating MaxMind's confidence that the country is correct. This attribute is only available from the Insights end point and the Enterprise database.

Type int

geoname_id

The GeoName ID for the country.

Type int

is_in_european_union

This is true if the country is a member state of the European Union.

Type bool

iso_code

The two-character ISO 3166-1 alpha code for the country.

Type str

name
The name of the country based on the locales list passed to the constructor.

Type str

names
A dictionary where the keys are locale codes and the values are names.

Type dict

type
A string indicating the type of entity that is representing the country. Currently we only return `military` but this could expand to include other types in the future.

Type str

class geoip2.records.**Subdivision**(*locales: Optional[List[str]] = None*, *confidence: Optional[int] = None*, *geoname_id: Optional[int] = None*, *iso_code: Optional[str] = None*, *names: Optional[Dict[str, str]] = None*, ***_*)
Contains data for the subdivisions associated with an IP address.

This class contains the subdivision data associated with an IP address.

This attribute is returned by `city`, `enterprise`, and `insights`.

Attributes:

confidence
This is a value from 0-100 indicating MaxMind's confidence that the subdivision is correct. This attribute is only available from the Insights end point and the Enterprise database.

Type int

geoname_id
This is a GeoName ID for the subdivision.

Type int

iso_code
This is a string up to three characters long contain the subdivision portion of the ISO 3166-2 code.

Type str

name
The name of the subdivision based on the locales list passed to the constructor.

Type str

names
A dictionary where the keys are locale codes and the values are names

Type dict

class geoip2.records.**Subdivisions**(*locales: Optional[List[str]], *subdivisions*)
A tuple-like collection of subdivisions associated with an IP address.

This class contains the subdivisions of the country associated with the IP address from largest to smallest.

For instance, the response for Oxford in the United Kingdom would have England as the first element and Oxfordshire as the second element.

This attribute is returned by `city`, `enterprise`, and `insights`.

most_specific
The most specific (smallest) subdivision available.

If there are no *Subdivision* objects for the response, this returns an empty *Subdivision*.

Type *Subdivision*

class geoip2.records.**Traits**(*autonomous_system_number: Optional[int] = None*, *autonomous_system_organization: Optional[str] = None*, *connection_type: Optional[str] = None*, *domain: Optional[str] = None*, *is_anonymous: bool = False*, *is_anonymous_proxy: bool = False*, *is_anonymous_vpn: bool = False*, *is_hosting_provider: bool = False*, *is_legitimate_proxy: bool = False*, *is_public_proxy: bool = False*, *is_residential_proxy: bool = False*, *is_satellite_provider: bool = False*, *is_tor_exit_node: bool = False*, *isp: Optional[str] = None*, *ip_address: Optional[str] = None*, *network: Optional[str] = None*, *organization: Optional[str] = None*, *prefix_len: Optional[int] = None*, *static_ip_score: Optional[float] = None*, *user_count: Optional[int] = None*, *user_type: Optional[str] = None*, *mobile_country_code: Optional[str] = None*, *mobile_network_code: Optional[str] = None*, **_)

Contains data for the traits record associated with an IP address.

This class contains the traits data associated with an IP address.

This class has the following attributes:

autonomous_system_number
The autonomous system number associated with the IP address. This attribute is only available from the City Plus and Insights web services and the Enterprise database.

Type int

autonomous_system_organization
The organization associated with the registered autonomous system number for the IP address. This attribute is only available from the City Plus and Insights web service end points and the Enterprise database.

Type str

connection_type
The connection type may take the following values:

- Dialup
- Cable/DSL
- Corporate
- Cellular

Additional values may be added in the future.

This attribute is only available in the Enterprise database.

Type str

domain
The second level domain associated with the IP address. This will be something like "example.com" or "example.co.uk", not "foo.example.com". This attribute is only available from the City Plus and Insights web service end points and the Enterprise database.

Type str

ip_address
The IP address that the data in the model is for. If you performed a "me" lookup against the web service, this will be the externally routable IP address for the system the code is running on. If the system is behind a NAT, this may differ from the IP address locally assigned to it.

Type str

is_anonymous
This is true if the IP address belongs to any sort of anonymous network. This attribute is only available from Insights.

Type bool

is_anonymous_proxy
This is true if the IP is an anonymous proxy.

Type bool

Deprecated since version 2.2.0: Use our our GeoIP2 Anonymous IP database instead.

is_anonymous_vpn
This is true if the IP address is registered to an anonymous VPN provider.

If a VPN provider does not register subnets under names associated with them, we will likely only flag their IP ranges using the `is_hosting_provider` attribute.

This attribute is only available from Insights.

Type bool

is_hosting_provider
This is true if the IP address belongs to a hosting or VPN provider (see description of `is_anonymous_vpn` attribute). This attribute is only available from Insights.

Type bool

is_legitimate_proxy
This attribute is true if MaxMind believes this IP address to be a legitimate proxy, such as an internal VPN used by a corporation. This attribute is only available in the Enterprise database.

Type bool

is_public_proxy
This is true if the IP address belongs to a public proxy. This attribute is only available from Insights.

Type bool

is_residential_proxy
This is true if the IP address is on a suspected anonymizing network and belongs to a residential ISP. This attribute is only available from Insights.

Type bool

is_satellite_provider
This is true if the IP address is from a satellite provider that provides service to multiple countries.

Type bool

Deprecated since version 2.2.0: Due to the increased coverage by mobile carriers, very few satellite providers now serve multiple countries. As a result, the output does not provide sufficiently relevant data for us to maintain it.

is_tor_exit_node
This is true if the IP address is a Tor exit node. This attribute is only available from Insights.

Type bool

isp
The name of the ISP associated with the IP address. This attribute is only available from the City Plus and Insights web services and the Enterprise database.

Type str

network
The network associated with the record. In particular, this is the largest network where all of the fields besides ip_address have the same value.

Type ipaddress.IPv4Network or ipaddress.IPv6Network

organization
The name of the organization associated with the IP address. This attribute is only available from the City Plus and Insights web services and the Enterprise database.

Type str

static_ip_score
An indicator of how static or dynamic an IP address is. The value ranges from 0 to 99.99 with higher values meaning a greater static association. For example, many IP addresses with a user_type of cellular have a lifetime under one. Static Cable/DSL IPs typically have a lifetime above thirty.

This indicator can be useful for deciding whether an IP address represents the same user over time. This attribute is only available from Insights.

Type float

user_count
The estimated number of users sharing the IP/network during the past 24 hours. For IPv4, the count is for the individual IP. For IPv6, the count is for the /64 network. This attribute is only available from Insights.

Type int

user_type
The user type associated with the IP address. This can be one of the following values:

- business
- cafe
- cellular
- college
- consumer_privacy_network
- content_delivery_network
- dialup
- government
- hosting
- library
- military
- residential
- router
- school

- search_engine_spider
- traveler

This attribute is only available from the Insights end point and the Enterprise database.

Type str

network
The network for the record

18.5 Errors

exception geoip2.errors.**AddressNotFoundError**(*message: str*, *ip_address: Optional[str] = None*, *prefix_len: Optional[int] = None*)
The address you were looking up was not found.

ip_address
The IP address used in the lookup. This is only available for database lookups.

Type str

network
The network associated with the error. In particular, this is the largest network where no address would be found. This is only available for database lookups.

Type ipaddress.IPv4Network or ipaddress.IPv6Network

network
The network for the error

exception geoip2.errors.**AuthenticationError**
There was a problem authenticating the request.

exception geoip2.errors.**GeoIP2Error**
There was a generic error in GeoIP2.

This class represents a generic error. It extends `RuntimeError` and does not add any additional attributes.

exception geoip2.errors.**HTTPError**(*message: str*, *http_status: Optional[int] = None*, *uri: Optional[str] = None*, *decoded_content: Optional[str] = None*)
There was an error when making your HTTP request.

This class represents an HTTP transport error. It extends *GeoIP2Error* and adds attributes of its own.

Variables

- **http_status** – The HTTP status code returned
- **uri** – The URI queried
- **decoded_content** – The decoded response content

exception geoip2.errors.**InvalidRequestError**
The request was invalid.

exception geoip2.errors.**OutOfQueriesError**
Your account is out of funds for the service queried.

exception geoip2.errors.**PermissionRequiredError**
Your account does not have permission to access this service.

CHAPTER 19

Indices and tables

- genindex
- modindex
- search

copyright

(c) 2013-2022 by MaxMind, Inc.

license Apache License, Version 2.0

Python Module Index

g

geoip2, 37
geoip2.database, 37
geoip2.errors, 55
geoip2.models, 41
geoip2.records, 47
geoip2.webservice, 38

Index

A

accuracy_radius (*geoip2.records.Location attribute*), 49
AddressNotFoundError, 55
anonymous_ip() (*geoip2.database.Reader method*), 37
AnonymousIP (*class in geoip2.models*), 42
ASN (*class in geoip2.models*), 41
asn() (*geoip2.database.Reader method*), 37
AsyncClient (*class in geoip2.webservice*), 39
AuthenticationError, 55
autonomous_system_number (*geoip2.models.ASN attribute*), 41
autonomous_system_number (*geoip2.models.ISP attribute*), 46
autonomous_system_number (*geoip2.records.Traits attribute*), 52
autonomous_system_organization (*geoip2.models.ASN attribute*), 42
autonomous_system_organization (*geoip2.models.ISP attribute*), 46
autonomous_system_organization (*geoip2.records.Traits attribute*), 52
average_income (*geoip2.records.Location attribute*), 49

C

City (*class in geoip2.models*), 43
City (*class in geoip2.records*), 47
city (*geoip2.models.City attribute*), 43
city (*geoip2.models.Enterprise attribute*), 45
city (*geoip2.models.Insights attribute*), 46
city() (*geoip2.database.Reader method*), 37
city() (*geoip2.webservice.AsyncClient method*), 39
city() (*geoip2.webservice.Client method*), 41
Client (*class in geoip2.webservice*), 40
close() (*geoip2.database.Reader method*), 38
close() (*geoip2.webservice.AsyncClient method*), 40
close() (*geoip2.webservice.Client method*), 41
code (*geoip2.records.Continent attribute*), 48
code (*geoip2.records.Postal attribute*), 50
confidence (*geoip2.records.City attribute*), 47
confidence (*geoip2.records.Country attribute*), 48
confidence (*geoip2.records.Postal attribute*), 50
confidence (*geoip2.records.RepresentedCountry attribute*), 50
confidence (*geoip2.records.Subdivision attribute*), 51
connection_type (*geoip2.models.ConnectionType attribute*), 43
connection_type (*geoip2.records.Traits attribute*), 52
connection_type() (*geoip2.database.Reader method*), 38
ConnectionType (*class in geoip2.models*), 43
Continent (*class in geoip2.records*), 47
continent (*geoip2.models.City attribute*), 43
continent (*geoip2.models.Country attribute*), 44
continent (*geoip2.models.Enterprise attribute*), 45
continent (*geoip2.models.Insights attribute*), 46
Country (*class in geoip2.models*), 44
Country (*class in geoip2.records*), 48
country (*geoip2.models.City attribute*), 43
country (*geoip2.models.Country attribute*), 44
country (*geoip2.models.Enterprise attribute*), 45
country (*geoip2.models.Insights attribute*), 46
country() (*geoip2.database.Reader method*), 38
country() (*geoip2.webservice.AsyncClient method*), 40
country() (*geoip2.webservice.Client method*), 41

D

Domain (*class in geoip2.models*), 44
domain (*geoip2.models.Domain attribute*), 44
domain (*geoip2.records.Traits attribute*), 52
domain() (*geoip2.database.Reader method*), 38

E

Enterprise (*class in geoip2.models*), 45

enterprise() (*geoip2.database.Reader method*), 38

G

geoip2 (*module*), 37
geoip2.database (*module*), 37
geoip2.errors (*module*), 55
geoip2.models (*module*), 41
geoip2.records (*module*), 47
geoip2.webservice (*module*), 38
GeoIP2Error, 55
geoname_id (*geoip2.records.City attribute*), 47
geoname_id (*geoip2.records.Continent attribute*), 48
geoname_id (*geoip2.records.Country attribute*), 48
geoname_id (*geoip2.records.RepresentedCountry attribute*), 50
geoname_id (*geoip2.records.Subdivision attribute*), 51

H

HTTPError, 55

I

Insights (*class in geoip2.models*), 46
insights() (*geoip2.webservice.AsyncClient method*), 40
insights() (*geoip2.webservice.Client method*), 41
InvalidRequestError, 55
ip_address (*geoip2.errors.AddressNotFoundError attribute*), 55
ip_address (*geoip2.models.AnonymousIP attribute*), 42
ip_address (*geoip2.models.ASN attribute*), 42
ip_address (*geoip2.models.ConnectionType attribute*), 44
ip_address (*geoip2.models.Domain attribute*), 45
ip_address (*geoip2.models.ISP attribute*), 46
ip_address (*geoip2.records.Traits attribute*), 52
is_anonymous (*geoip2.models.AnonymousIP attribute*), 42
is_anonymous (*geoip2.records.Traits attribute*), 53
is_anonymous_proxy (*geoip2.records.Traits attribute*), 53
is_anonymous_vpn (*geoip2.models.AnonymousIP attribute*), 42
is_anonymous_vpn (*geoip2.records.Traits attribute*), 53
is_hosting_provider (*geoip2.models.AnonymousIP attribute*), 42
is_hosting_provider (*geoip2.records.Traits attribute*), 53
is_in_european_union (*geoip2.records.Country attribute*), 48
is_in_european_union (*geoip2.records.RepresentedCountry attribute*), 50
is_legitimate_proxy (*geoip2.records.Traits attribute*), 53
is_public_proxy (*geoip2.models.AnonymousIP attribute*), 42
is_public_proxy (*geoip2.records.Traits attribute*), 53
is_residential_proxy (*geoip2.models.AnonymousIP attribute*), 42
is_residential_proxy (*geoip2.records.Traits attribute*), 53
is_satellite_provider (*geoip2.records.Traits attribute*), 53
is_tor_exit_node (*geoip2.models.AnonymousIP attribute*), 42
is_tor_exit_node (*geoip2.records.Traits attribute*), 53
iso_code (*geoip2.records.Country attribute*), 48
iso_code (*geoip2.records.RepresentedCountry attribute*), 50
iso_code (*geoip2.records.Subdivision attribute*), 51
ISP (*class in geoip2.models*), 46
isp (*geoip2.models.ISP attribute*), 46
isp (*geoip2.records.Traits attribute*), 54
isp() (*geoip2.database.Reader method*), 38

L

latitude (*geoip2.records.Location attribute*), 49
Location (*class in geoip2.records*), 48
location (*geoip2.models.City attribute*), 43
location (*geoip2.models.Enterprise attribute*), 45
location (*geoip2.models.Insights attribute*), 46
longitude (*geoip2.records.Location attribute*), 49

M

MaxMind (*class in geoip2.records*), 49
maxmind (*geoip2.models.City attribute*), 43
maxmind (*geoip2.models.Country attribute*), 44
maxmind (*geoip2.models.Enterprise attribute*), 45
maxmind (*geoip2.models.Insights attribute*), 46
metadata() (*geoip2.database.Reader method*), 38
metro_code (*geoip2.records.Location attribute*), 49
most_specific (*geoip2.records.Subdivisions attribute*), 51

N

name (*geoip2.records.City attribute*), 47
name (*geoip2.records.Continent attribute*), 48
name (*geoip2.records.Country attribute*), 48
name (*geoip2.records.PlaceRecord attribute*), 50
name (*geoip2.records.RepresentedCountry attribute*), 51

name (*geoip2.records.Subdivision attribute*), 51
names (*geoip2.records.City attribute*), 47
names (*geoip2.records.Continent attribute*), 48
names (*geoip2.records.Country attribute*), 48
names (*geoip2.records.RepresentedCountry attribute*), 51
names (*geoip2.records.Subdivision attribute*), 51
network (*geoip2.errors.AddressNotFoundError attribute*), 55
network (*geoip2.models.AnonymousIP attribute*), 42
network (*geoip2.models.ASN attribute*), 42
network (*geoip2.models.ConnectionType attribute*), 44
network (*geoip2.models.Domain attribute*), 45
network (*geoip2.models.ISP attribute*), 46
network (*geoip2.models.SimpleModel attribute*), 47
network (*geoip2.records.Traits attribute*), 54, 55

O

organization (*geoip2.models.ISP attribute*), 46
organization (*geoip2.records.Traits attribute*), 54
OutOfQueriesError, 55

P

PermissionRequiredError, 55
PlaceRecord (*class in geoip2.records*), 50
population_density (*geoip2.records.Location attribute*), 49
Postal (*class in geoip2.records*), 50
postal (*geoip2.models.City attribute*), 43

Q

queries_remaining (*geoip2.records.MaxMind attribute*), 49

R

Reader (*class in geoip2.database*), 37
Record (*class in geoip2.records*), 50
registered_country (*geoip2.models.City attribute*), 43
registered_country (*geoip2.models.Country attribute*), 44
registered_country (*geoip2.models.Enterprise attribute*), 45
registered_country (*geoip2.models.Insights attribute*), 46
represented_country (*geoip2.models.City attribute*), 43
represented_country (*geoip2.models.Country attribute*), 44
represented_country (*geoip2.models.Enterprise attribute*), 45
represented_country (*geoip2.models.Insights attribute*), 47
RepresentedCountry (*class in geoip2.records*), 50

S

SimpleModel (*class in geoip2.models*), 47
static_ip_score (*geoip2.records.Traits attribute*), 54
Subdivision (*class in geoip2.records*), 51
Subdivisions (*class in geoip2.records*), 51
subdivisions (*geoip2.models.City attribute*), 43
subdivisions (*geoip2.models.Enterprise attribute*), 45
subdivisions (*geoip2.models.Insights attribute*), 47

T

time_zone (*geoip2.records.Location attribute*), 49
Traits (*class in geoip2.records*), 52
traits (*geoip2.models.City attribute*), 43
traits (*geoip2.models.Country attribute*), 44
traits (*geoip2.models.Enterprise attribute*), 45
traits (*geoip2.models.Insights attribute*), 47
type (*geoip2.records.RepresentedCountry attribute*), 51

U

user_count (*geoip2.records.Traits attribute*), 54
user_type (*geoip2.records.Traits attribute*), 54