ipwhois Documentation

Release 1.2.0

secynic

Nov 15, 2020

Project Info

ipwhois is a Python package focused on retrieving and parsing whois data for IPv4 and IPv6 addresses.

CHAPTER 1

ipwhois

1.1 Summary

ipwhois is a Python package focused on retrieving and parsing whois data for IPv4 and IPv6 addresses.

Note: If you are experiencing latency issues, it is likely related to rate limiting. Rate limiting is based on your source IP, which may be a problem with multiple users behind the same proxy. Additionally, LACNIC implements aggressive rate limiting. Experimental bulk query support is new as of v1.0.0.

1.2 Features

- Parses a majority of whois fields in to a standard dictionary
- IPv4 and IPv6 support
- Supports RDAP queries (recommended method, see: https://tools.ietf.org/html/rfc7483)
- Proxy support for RDAP queries
- Supports legacy whois protocol queries

- Referral whois support for legacy whois protocol
- Recursive network parsing for IPs with parent/children networks listed
- National Internet Registry support for JPNIC and KRNIC
- Supports IP to ASN and ASN origin queries
- Python 2.7 and 3.4+ supported
- Useful set of utilities
- Experimental bulk query support
- BSD license
- Human readable field translations
- Full CLI for IPWhois with optional ANSI colored console output.

1.3 Links

1.3.1 Documentation

GitHub latest

https://ipwhois.readthedocs.io/en/latest

GitHub dev

https://ipwhois.readthedocs.io/en/dev

1.3.2 Examples

https://github.com/secynic/ipwhois/tree/master/ipwhois/examples

1.3.3 Github

https://github.com/secynic/ipwhois

1.3.4 Pypi

https://pypi.org/project/ipwhois

1.3.5 Changes

https://ipwhois.readthedocs.io/en/latest/CHANGES.html

1.3.6 Upgrade Notes

https://ipwhois.readthedocs.io/en/latest/UPGRADING.html

1.4 Dependencies

Python 2.7:

```
dnspython
ipaddr
```

Python 3.4+:

```
dnspython
```

1.5 Installing

Latest release from PyPi:

```
pip install --upgrade ipwhois
```

GitHub - Stable:

```
pip install -e git+https://github.com/secynic/ipwhois@master#egg=ipwhois
```

GitHub - Dev:

```
pip install -e git+https://github.com/secynic/ipwhois@dev#egg=ipwhois
```

1.6 Firewall Ports

ipwhois needs some outbound firewall ports opened from your host/server.

ASN (DNS) 53/tcp

ASN (Whois) 43/tcp

ASN (HTTP) 80/tcp

443/tcp (Pending)

RDAP (HTTP) 80/tcp

443/tcp (Pending)

NIR (HTTP) 80/tcp

443/tcp (KRNIC)

Legacy Whois 43/tcp

Get Host 43/tcp

1.7 API

1.7.1 IPWhois (main class)

ipwhois.IPWhois is the base class for wrapping RDAP and Legacy Whois lookups. Instantiate this object, then call one of the lookup functions:

RDAP (HTTP) - IPWhois.lookup_rdap() OR *Legacy Whois - IPWhois.lookup_whois()*

Input

Key	Type	Description
address	str	An IPv4 or IPv6 address as a string, integer, IPv4Address, or IPv6Address.
timeout	int	The default timeout for socket connections in seconds. Defaults to 5.
proxy_opener	object	The urllib.request.OpenerDirector request for proxy support or None.

1.7.2 RDAP (HTTP)

IPWhois.lookup_rdap() is the recommended lookup method. RDAP provides a far better data structure than legacy whois and REST lookups (previous implementation). RDAP queries allow for parsing of contact information and details for users, organizations, and groups. RDAP also provides more detailed network information.

RDAP documentation:

https://ipwhois.readthedocs.io/en/latest/RDAP.html

1.7.3 Legacy Whois

Note: Legacy Whois output is different from RDAP. See the below JSON outputs for a comparison:

Legacy Whois: https://ipwhois.readthedocs.io/en/latest/WHOIS.html#basic-usage

RDAP: https://ipwhois.readthedocs.io/en/latest/RDAP.html#basic-usage

---

Legacy Whois documentation:

https://ipwhois.readthedocs.io/en/latest/WHOIS.html

1.7.4 National Internet Registries

This library now supports NIR lookups for JPNIC and KRNIC. Previously, Whois and RDAP data for Japan and South Korea was restricted. NIR lookups scrape these national registries directly for the data restricted from regional internet registries. NIR queries are enabled by default via the inc_nir argument in the IPWhois.lookup_*() functions.

https://ipwhois.readthedocs.io/en/latest/NIR.html

1.7.5 Autonomous System Numbers

This library now supports ASN origin lookups via Whois and HTTP.

IP ASN functionality was moved to its own parser API (IPASN).

There is no CLI for these yet.

https://ipwhois.readthedocs.io/en/latest/ASN.html

1.7.6 Utilities

Utilities documentation:

https://ipwhois.readthedocs.io/en/latest/UTILS.html

1.7.7 Scripts

CLI documentation:

https://ipwhois.readthedocs.io/en/latest/CLI.html

1.7.8 Experimental Functions

Caution: Functions in experimental.py contain new functionality that has not yet been widely tested. Bulk lookup support contained here can result in significant system/network resource utilization. Additionally, abuse of this functionality may get you banned by the various services queried by this library. Use at your own discretion.

Experimental functions documentation:

https://ipwhois.readthedocs.io/en/latest/EXPERIMENTAL.html

1.8 Contributing

https://ipwhois.readthedocs.io/en/latest/CONTRIBUTING.html

1.9 IP Reputation Support

This feature is under consideration. Take a look at TekDefense's Automater:

TekDefense-Automater

1.10 Domain Support

There are no plans for domain whois support in this project.

Look at Sven Slootweg's python-whois for a library with domain support.

1.11 Special Thanks

Thank you JetBrains for the PyCharm open source support!

Thank you Chris Wells (@cdubz) for your extensive testing on the experimental functions!

Last but not least, thank you to all the issue submitters and contributors.

CHAPTER 2

Contributing

Note: If you are looking for items to contribute, start by looking at current open issues and search the source code for "TODO" items.

2.1 Issue submission

Issues are tracked on GitHub:
https://github.com/secynic/ipwhois/issues

Follow the guidelines detailed in the appropriate section below. As a general rule of thumb, provide as much information as possible when submitting issues.

2.1.1 Bug reports

- Title should be a short, descriptive summary of the bug
- Include the Python and ipwhois versions affected
- Provide a context (with code example) in the description of your issue. What are you attempting to do?
- Include the full obfuscated output. Make sure to set DEBUG logging:

```
import logging
LOG_FORMAT = ('[%(asctime)s] [%(levelname)s] [%(filename)s:%(lineno)s] '
    '[%(funcName)s()] %(message)s')
logging.basicConfig(level=logging.DEBUG, format=LOG_FORMAT)
```

- Include sources of information with links or screenshots
- Do you have a suggestion on how to fix the bug?

2.1.2 Feature Requests

- Title should be a short, descriptive summary of the feature requested
- Provide use case examples
- Include sources of information with links or screenshots
- Do you have a suggestion on how to implement the feature?

2.1.3 Testing

You may have noticed that Travis CI tests are taking longer to complete. This is due to the enabling of online lookup tests (network tests in the ipwhois/tests/online directory).

When running local tests, you may include these tests by adding the –include=online flag to your nosetests command.

Example:

```
nosetests -v -w ipwhois --include=online --exclude=stress --with-coverage
 --cover-package=ipwhois
```

2.1.4 Questions

I am happy to answer any questions and provide assistance where possible. Please be clear and concise. Provide examples when possible. Check the ipwhois documentation and the issue tracker before asking a question.

Questions can be submitted as issues. Past questions can be searched by filtering the label "question".

You can also message me on IRC. I am usually idle on freenode in the Python channels

2.2 Pull Requests

2.2.1 What to include

Aside from the core code changes, it is helpful to provide the following (where applicable):

- Unit tests
- Examples
- Sphinx configuration changes in /docs
- Requirements (python2.txt, python3.txt, docs/requirements.txt)

2.2.2 GitFlow Model

This library follows the GitFlow model. As a contributor, this is simply accomplished by the following steps:

1. Create an issue (if there isn't one already)
2. Branch from dev (not master), try to name your branch to reference the issue (e.g., issue_123_feature, issue_123_bugfix).
3. Merge pull requests to dev (not master). Hotfix merges to master will only be allowed under extreme/time sensitive circumstances.

2.2.3 Guidelines

- Title should be a short, descriptive summary of the changes
- Follow PEP 8 where possible.
- Follow the Google docstring style guide for comments
- Must be compatible with Python 2.7 and 3.4+
- Break out reusable code to functions
- Make your code easy to read and comment where necessary
- Reference the GitHub issue number in the description (e.g., Issue #01)
- When running nosetests, make sure to add the following arguments:

```
--verbosity=3 --nologcapture --include=online --cover-erase
```

If you would like to exclude the aggressive online stress tests, add to the above:

```
--exclude stress
```

CHAPTER 3

License

CHAPTER 4

Changelog

4.1 1.3.0 (TBD)

- Fixed deprecated query method of dnspython (#294 - monoidic)

4.2 1.2.0 (2020-09-17)

- Removed deprecated functions: asn.IPASN._parse_fields_http, asn.IPASN._parse_fields_dns, asn.IPASN._parse_fields_whois, asn.ASNOrigin._parse_fields, asn.ASNOrigin._get_nets_radb, net.Net.lookup_asn, whois.Whois._parse_fields, whois.Whois._get_nets_arin whois.Whois._get_nets_lacnic, whois.Whois._get_nets_other, nir.NIRWhois._parse_fields, nir.NIRWhois._get_nets_jpnic nir.NIRWhois._get_nets_krnic, nir.NIRWhois._get_contact (#230)
- Removed deprecated asn_alts parameter (#230)
- Removed deprecated allow_permutations parameter (#230)
- Fixed ASNOrigin lookups (#216)
- Fixed bug in ASNOrigin lookups when multiple asn_methods provided (#216)
- Fixed bug in KRNIC queries due to a change in their service (#243)
- Fixed bug in experimental.bulk_lookup_rdap where only the last result was returned (#262 - ameidatou)
- Fixed deprecation warnings due to invalid escape sequences (#272 - tirkarthi)
- Fixed bug in root and sub-entities not getting queried/data (#247)
- Fixed NIR datetime parsing issue if only date is returned (#284)
- Added new argument root_ent_check to IPWhois.lookup_rdap and RDAP.lookup. Set this to False to revert to old functionality - missing data, but less queries (#247)
- Added support for Python 3.8 (#267)
- Fixed travis build warnings (#268)

- Pinned requirements (#274)
- Added ip_failed_total key to stats dictionary in experimental.bulk_lookup_rdap (#235)
- Added ipv4_generate_random and ipv6_generate_random to utils CLI (#236)
- Added documentation note for ASN data (#278)

4.3 1.1.0 (2019-02-01)

- Exceptions now inherit a new BaseIpwhoisException rather than Exception (#205 - Darkheir)
- Fixed list output for generate_examples.py (#196)
- Fixed bug in ASN HTTP lookup where the ARIN results were reversed, and parsing would fail on the first item (#220)
- Removed support for Python 2.6/3.3, added support for 3.7 (#221)
- Fixed deprecation warnings in core code (#203 - cstranex)
- Fixed bug in host argument for elastic_search.py example (#202)
- Set user agent in elastic_search.py example to avoid default user agent
- Updated elastic_search.py example for ES 6.6.0
- Readme update for RDAP vs Legacy Whois output (#204)
- Removed the disallow_permutations argument from ipwhois_cli.py (#226)

4.4 1.0.0 (2017-07-30)

- Deprecated asn_alts, allow_permutations in favor of new asn_methods (#158)
- Added new exception ASNOriginLookupError (#158)
- KRNIC lookups changed to HTTPS (#166)
- Added experimental functions - get_bulk_asn_whois, bulk_lookup_rdap (#134)
- Fixed bug in NIR lookups that caused addresses with multi-line contacts to error (#172 - kwheeles)
- Added IANA Reserved CIDR 198.97.38.0/24 to ipv4_is_defined (#174)
- Fixed bug in RDAP notices/remarks parsing that would omit partial entries missing one or more of title, description, links (#176)
- Added new return key asn_description via verbose ASN DNS lookup support and modified ASN whois lookups. New argument get_asn_description (#176)
- Fixed some test function naming errors
- Added new generators to utils.py: ipv4_generate_random and ipv6_generate_random (#183)
- Moved upgrade notes to new UPGRADING.rst
- Deprecated unnecessary protected class functions, changed to public in asn.py, nir.py, and whois.py (#184)
- net.Net.get_host(), utils.ipv4_is_defined(), and utils.ipv6_is_defined now return namedtuple instead of tuple.
- Changed docstrings to Google standard for better napoleon parsing (#185)
- Removed deprecated IPWhois.lookup() - This was moved to IPWhois.lookup_whois()

- Fixed 'nets'->'range' bug for legacy whois CIDR net_range values (#188)
- Fixed a bug in IPASN/Net that caused the ASN result to vary if Cymru has more than one ASN listed for an IP (#190)
- Updated ElasticSearch example for ES v5.5.1 (#138)

4.5 0.15.1 (2017-02-16)

- Fixed IPv6 parsing for ASN origin lookups and added tests (#162 - ti-mo)
- Fixed recursive role parsing at depths greater than 0 (#161 - cdubz)

4.6 0.15.0 (2017-02-02)

- Python 3.3+ dnspython3 requirement changed to dnspython (#155)
- Added ASN origin lookup support (#149)
- Moved ASN parsing from net.Net.get_asn_*() to new class asn.IPASN. The original functions now return the raw query (#157)
- net.Net.lookup_asn() is deprecated in favor of asn.IPASN.lookup() (#157)
- Added new exception ASNParseError (#157)
- Fixed rate-limiting exception handling for when HTTP errors are returned rather than JSON errors (rikonor - #144)
- Fixed rate-limit infinite recursion bug for legacy whois (rikonor - #144)
- Fixed bug in net.Net.get_http_raw() that would pass the encoded form_data on retry rather than the original argument.
- Removed nose requirements and fixed travis.yml for updated pip
- Documentation updates
- Code style tweaks
- Updated tests and version info for Python 3.6
- Added basic stress tests (#144)
- Minor tweaks to existing tests

4.7 0.14.0 (2016-08-29)

- Changed legacy whois emails output type to list (#133)
- Fixed retry count non-decrementing infinite loop in ipwhois.net.Net.get_whois() (issue #125 - krader1961)
- Added new function ipwhois.net.Net.get_http_raw() and tests (#67)
- Added National Internet Registry (JPNIC, KRNIC) support (#67). Enabled by default in IPWhois.lookup_*(). Disable by passing inc_nir=False. Optionally, lower level code can call nir.NIRWhois(). This enhancement results in extra network queries, but more detailed information for NIRs.

- Added utils CLI (ipwhois_utils_cli.py) - #121. Installed to your environments Scripts dir. This is a wrapper for utils.py.
- Documentation improvements (#123)
- kw arg readability (#115)
- Replaced usage of args with script_args in ipwhois_cli.py
- Minor optimization in whois.py and online/test_whois.py
- Added coveralls integration and re-enabled online tests with Travis CI
- Added Read the Docs support (#132)
- Added documentation (Sphinx) requirements.txt (#132)
- Fixed test imports
- Added –json argument (output in JSON format) to ipwhois_cli.py (#135)

4.8 0.13.0 (2016-04-18)

- Added events_actor parsing for RDAP results.
- Added example for caching data via Redis (#81)
- Added normalization (human-readable field information) in hr.py (#47)
- README word wrap fix (#102)
- Fixed bug in exception handling for ASN HTTP lookups.
- Fixed bug in IPWhois.lookup_rdap() that caused ASN HTTP lookup responses to be used in place of RDAP responses.
- Added new function Net.get_asn_http() and migrated code from Net.lookup_asn() + new tests.
- Fixed bug in ASN HTTP fallback lookups for DNIC (#108).
- Added new parameter extra_org_map in Net.get_asn_http(), Net.lookup_asn(), and IPWhois.lookup*() (#108).
- Fixed _RDAPCommon.summarize_notices() None check - changed len() to all().
- Added CLI (ipwhois_cli.py) - #46. Installed to your environments Scripts dir. This is a wrapper for ipwhois.py (IPWhois). Utils CLI will be in a future release (#121).
- Documentation split up and added more detail (#81).

4.9 0.12.0 (2016-03-28)

- Added headers parameter to ipwhois.Net.get_http_json() (issue #98).
- Fixed ASN HTTP lookup (fallback) Accept headers (issue #98).
- Fixed HTTP decoding, set to utf-8 (italomaia - issue #97)
- IPWhois.lookup() deprecated (issue #96), and will be removed in a future release (TBD). Use IPWhois.lookup_whois() instead.
- Added rate_limit_timeout parameter (issue #99) to Net.get_http_json(), IPWhois.lookup_rdap(), and RDAP.lookup(). New exception HTTPRateLimitError.

- Added new parameter asn_alts to Net.lookup_asn(), IPWhois.lookup_rdap() and IPWhois.lookup(). Takes a list of lookup types to attempt if the ASN dns lookup fails. Allow permutations must be enabled. Defaults to all ['whois', 'http'] (issue #93).
- Fixed socket exception handling in Net.get_http_json() for Python 2.6.
- Fixed assertIsInstance for Python 2.6 tests (issue #100). Implemented unittest._formatMessage and unittest.util.safe_repr for Python 2.6.
- Moved TestCommon to tests__init__.py to avoid duplicate code.
- Replaced remaining % with str.format (issue #95).

4.10 0.11.2 (2016-02-25)

- Added allow_permutations parameter (bool) to net.Net() and ipwhois.IPWhois() to allow alternate ASN lookups if DNS lookups fail. (FirefighterBlu3)
- Fixed ASN DNS resolver timeout/retry_count support. Retry count is used as a multiplier of timeout, to determine a limetime interval. (FirefighterBlu3)
- Fixed bug where remarks would return None if missing a title.
- Added CONTRIBUTING.rst
- Added tests

4.11 0.11.1 (2015-12-17)

- Re-added CIDR calculation for RDAP lookups.
- Improved tests - core code coverage now 100%. See '# pragma: no cover' for exclusions. A few bugs were identified in the process, detailed below.
- Moved IP zero stripping from rdap._RDAPNetwork.parse() to new helper function utils.ipv4_lstrip_zeros().
- Moved CIDR calculation from rdap._RDAPNetwork.parse() to new helper function utils.calculate_cidr().
- Fixed utils.ipv4_is_defined() if statement ordering for RFC 1918 conflict.
- Fixed utils.ipv6_is_defined() if statement ordering for Unspecified and Loopback (conflict with Reserved).
- Added is_offline parameter to whois.Whois.lookup() primarily for testing.
- Fixed bug in whois.Whois._parse_fields() that attempted to parse 'val2' of regex, which is no longer used. Also fixed the expected Exception to be IndexError.
- Fixed bug in ipwhois.IPWhois.lookup() where the argument order was mixed up, causing referral lookups to be skipped when get_referral=True.
- Fixed bug in rdap._RDAPCommon.summarize_notices() output for links.
- Fixed bug in root entity iteration exception handling in rdap.RDAP.lookup().

4.12 0.11.0 (2015-11-02)

- Support for REST lookups replaced with RDAP.

- Split code for a more structured system (net, whois, rdap, exceptions).
- Tests match the data new structure.
- Split tests for online and offline testing.
- Performance enhancements for parsing.
- Added an optional bootstrap parameter for RDAP lookups, in order to replace ASN lookups or use both. Will default to False. Afrinic is currently not supported, so I would not use this for now. ARIN acknowledged my issue for this, and will be adding support back in for Afrinic bootstrap.
- Added field_list parameter (inclusion list) for WHOIS lookups.
- Added logging.
- Added examples directory.

4.13 0.10.3 (2015-08-14)

- Fixed LACNIC lookup_rws() queries, since they switched to RDAP. This is temporary to get it working until the major library transition to RDAP and new parsed formatting is complete.

4.14 0.10.2 (2015-05-19)

- Fixed APNIC parsing for updated field.
- Fixed datetime parsing and validation when Zulu (Z) is appended.
- Added RIPE parsing for created and updated fields (whois and RWS).
- Removed unnecessary parentheses in IPWhois class declaration.
- Some documentation and comment tweaking to work with Sphinx.
- Minor PEP 8 tweaks.

4.15 0.10.1 (2015-02-09)

- Fixed setup.py bug.

4.16 0.10.0 (2015-02-09)

- Added .csv support for country code source. You can no longer download country code information from iso.org.
- Added support for IPv4Address or IPv6Address as the address arg in IPWhois.
- Fixed file open encoding bug. Moved from open to io.open.
- Fixed parameter in IPWhois ip defined checks.
- Fixed TestIPWhois.test_ip_invalid() assertions.

CHAPTER 5

Changelog (Archive)

5.1 0.9.1 (2014-10-14)

- Added ignore_referral_errors parameter to lookup().
- Fixed ipaddress import conflicts with alternate ipaddress module.
- Tuned import exception in ipwhois.utils.
- Fixed retry handling in get_whois().
- Fixed CIDR regex parsing bug where some nets were excluded from the results.

5.2 0.9.0 (2014-07-27)

- Fixed order on REST email fields
- Fixed setup error for initial install when dependencies don't exist.
- Added RWhois support.
- Added server and port parameters to IPWhois.get_whois().
- Added unique_addresses() to ipwhois.utils and unit tests.
- Added some unit tests to test_lookup().
- Replaced dict.copy() with copy.deepcopy(dict).
- Fixed bug in abuse emails parsing.
- Added handle and range values to returned nets dictionary.

5.3 0.8.2 (2014-05-12)

- Fixed multi-line field parsing (Issue #36).
- Added unique_everseen() to ipwhois.utils to fix multi-line field order.
- Re-added support for RIPE RWS now that their API is fixed.

5.4 0.8.1 (2014-03-05)

- Fixed encoding error in IPWhois.get_whois().

5.5 0.8.0 (2014-02-18)

- Added ASNRegistryError to handle unknown ASN registry return values.
- Added ASN registry lookup third tier fallback to ARIN.
- Fixed variable naming to avoid shadows built-in confusion.
- Fixed some type errors: Expected type 'str', got 'dict[str, dict]' instead.
- Fixed RIPE RWS links, since they changed their API.
- Temporarily removed RIPE RWS functionality until they fix their API.
- Removed RADB fallback, since RIPE removed it.

5.6 0.7.0 (2014-01-14)

- Added Python 2.6+ support.
- The country field in net dicts is now forced uppercase.

5.7 0.6.0 (2014-01-13)

- Added APNIC RWS support for IPWhois.lookup_rws().
- Fixed issue in IPWhois.lookup_rws() for radb-grs fallback.

5.8 0.5.2 (2013-12-07)

- Fixed special character issue in countries XML file (Issue #23).

5.9 0.5.1 (2013-12-03)

- Moved regex string literal declarations to NIC_WHOIS dict.
- Moved RWS parsing to own private functions.
- Moved base_net dict to global BASE_NET.
- More granular exception handling in lookup functions.
- Fixed email parsing for ARIN and RIPE RWS.
- Changed some 'if key in dict' statements to try/except for slight performance increase in lookup functions.
- Removed generic exception handling (returned blank dict) on get_countries().
- More PEP 8 reformatting.
- Minor docstring modifications.
- Added some unit tests to test_lookup() and test_lookup_rws().

5.10 0.5.0 (2013-11-20)

- Reformatting for PEP 8 compliance.
- Added LACNIC RWS (Beta v2) support for IPWhois.lookup_rws().

5.11 0.4.0 (2013-10-17)

- Added support for network registered and updated time stamps (keys: created, updated). Value in ISO 8601 format.
- Added value assertion to test_utils.py.
- Fixed IPWhois.lookup() handling of processed values. If processing throws an exception, discard the value and not the net dictionary.

5.12 0.3.0 (2013-09-30)

- Fixed get_countries() to work with frozen executables.
- Added dnspython3 rdtypes import to fix issue with frozen executables.
- Moved iso_3166-1_list_en.xml to /data.
- Added retry_count to IPWhois.lookup() and IPWhois.lookup_rws().

5.13 0.2.1 (2013-09-27)

- Fixed LACNIC CIDR validation on IPWhois.lookup().
- Fixed bug in IPWhois.get_whois() for query rate limiting. This was discovered via testing multiprocessing with 8+ processes running asynchronously.

5.14 0.2.0 (2013-09-23)

- Added support for emails (keys: abuse_emails, tech_emails, misc_emails).
- Changed regex to use group naming for more complex searching.
- Added some missing exception handling in lookup_rws().

5.15 0.1.9 (2013-09-18)

- Added exceptions to import in __init__.py.
- Added IPWhois.__repr__().
- Moved exceptions to get_*() functions.
- Added exception HostLookupError.
- Various optimizations.
- Added some unit tests.

5.16 0.1.8 (2013-09-17)

- Removed set_proxy() in favor of having the user provide their own urllib.request.OpenerDirector instance as a parameter to IPWhois().
- Restructured package in favor of modularity. get_countries() is now located in ipwhois.utils.
- Added exception WhoisLookupError for IPWhois.lookup() and IPWhois.lookup_rws().

5.17 0.1.7 (2013-09-16)

- Fixed bug in set_proxy().
- Removed ARIN top level network entries from return dictionary of IPWhois.lookup_rws().
- Fixed bug in ARIN RWS parsing when only one network.

5.18 0.1.6 (2013-09-16)

- Added IPWhois.get_host() to resolve hostname information.
- Added address and postal_code fields to parsed results.
- Normalized single/double quote use.

5.19 0.1.5 (2013-09-13)

- Added set_proxy() function for proxy support in Whois-RWS queries.
- Added IPWhois.lookup_rws() function for Whois-RWS queries.

5.20 0.1.4 (2013-09-12)

- Added validity checks for the asn_registry value due to a bug in the Team Cymru ASN lookup over night.
- Added timeout argument to IPWhois(). This is the default timeout in seconds for socket connections.
- Fixed decoding issue in IPWhois.get_whois().

5.21 0.1.3 (2013-09-11)

- Added exception handling with query retry support for socket errors, timeouts, connection resets.
- Moved ASN queries to their own functions (IPWhois.get_asn_dns() and IPWhois.get_asn_whois())
- Moved whois query to its own function (IPWhois.get_whois())
- Country codes are now forced as upper case in the return dictionary.

5.22 0.1.2 (2013-09-10)

- Fixed file path for get_countries().
- Fixed variable names that conflicted with builtins.
- Added content to README.
- Moved CHANGES.txt to CHANGES.rst and added to setup.py.
- Download URL now points to GitHub master tarball.

5.23 0.1.1 (2013-09-09)

- Fixed README issue.

5.24 0.1.0 (2013-09-06)

- Initial release.

CHAPTER 6

Upgrade Notes

Version upgrade notes, warnings, and critical changes will be displayed here. This does not supplement the changelog, but serves to provide information on any changes that may affect user experience when upgrading to a new release.

This page is new as of version 1.0.0. Any information on older versions is likely missing or incomplete.

6.1 v1.2.0

- Removed deprecated functions: asn.IPASN._parse_fields_http, asn.IPASN._parse_fields_dns, asn.IPASN._parse_fields_whois, asn.ASNOrigin._parse_fields, asn.ASNOrigin._get_nets_radb, net.Net.lookup_asn, whois.Whois._parse_fields, whois.Whois._get_nets_arin whois.Whois._get_nets_lacnic, whois.Whois._get_nets_other, nir.NIRWhois._parse_fields, nir.NIRWhois._get_nets_jpnic nir.NIRWhois._get_nets_krnic, nir.NIRWhois._get_contact
- Removed deprecated asn_alts parameter
- Removed deprecated allow_permutations parameter
- Added new argument root_ent_check to IPWhois.lookup_rdap and RDAP.lookup. Set this to False to revert to old functionality - missing data, but less queries. If you leave this set to default of True, you will notice more queries and potentially more rate-limiting.
- Added support for Python 3.8
- Pinned requirements

6.2 v1.1.0

- Exceptions now inherit a new BaseIpwhoisException rather than Exception
- Removed support for Python 2.6/3.3, added support for 3.7
- Removed the disallow_permutations argument from ipwhois_cli.py. Use ans_methods instead.

- Fixed deprecation warnings in core code

6.3 v1.0.0

- Removed deprecated IPWhois.lookup() - This was moved to IPWhois.lookup_whois()
- HTTPS (port 443) requirement added for KRNIC lookups.
- Experimental bulk functions added: experimental.get_bulk_asn_whois and experimental.bulk_lookup_rdap.
- Added new return key asn_description to net.Net.get_asn_whois, experimental.get_bulk_asn_whois, and hr.py. New argument get_asn_description.
- The IPWhois argument allow_permutations and the lookup argument asn_alts have been deprecated in favor of new argument asn_methods.
- Deprecated unnecessary protected class functions, changed to public in asn.py, nir.py, and whois.py (#184): asn.IPASN._parse_fields_dns, asn.IPASN._parse_fields_whois, asn.IPASN._parse_fields_http, asn.ASNOrigin._parse_fields, asn.ASNOrigin._get_nets_radb, nir.NIRWhois._parse_fields, nir.NIRWhois._get_nets_jpnic, nir.NIRWhois._get_nets_krnic, nir.NIRWhois._get_contact, whois.Whois._parse_fields, whois.Whois._get_nets_arin, whois.Whois._get_nets_lacnic, whois.Whois._get_nets_other
- New IP generators added: utils.ipv4_generate_random and utils.ipv6_generate_random
- net.Net.get_host(), utils.ipv4_is_defined(), and utils.ipv6_is_defined now return namedtuple instead of tuple.
- net.Net.get_asn_dns now returns a list rather than a str

6.4 v0.14.0

- NIR (National Internet Registry) lookups are enabled by default. This is currently only performed for JPNIC and KRNIC addresses. To disable, set inc_nir=False in your IPWhois.lookup_*() query.
- The 'nets' -> 'emails' key in IPWhois.lookup_whois() was changed from a '\n' separated string to a list.

6.5 v0.11.0

- The new RDAP return format was introduced and split off from the legacy whois return format. Using RDAP lookup (IPWhois.lookup_rdap()) is now the recommended method to maximize indexable values. RDAP return data is different in nearly every way from the legacy whois data. For information on raw RDAP responses, please see the RFC: https://tools.ietf.org/html/rfc7483

CHAPTER 7

RDAP (HTTP) Lookups

IPWhois.lookup_rdap() is now the recommended lookup method. RDAP provides a far better data structure than legacy whois and REST lookups (previous implementation). RDAP queries allow for parsing of contact information and details for users, organizations, and groups. RDAP also provides more detailed network information.

7.1 Input

Arguments supported by IPWhois.lookup_rdap().

Key	Type	Description
inc_raw	bool	Whether to include the raw whois results in the returned dictionary. Defaults to False.
retry_count	int	The number of times to retry in case socket errors, timeouts, connection resets, etc. are encountered. Defaults to 3.
depth	int	How many levels deep to run queries when additional referenced objects are found. Defaults to 0.
excluded_entities	list	Entity handles to not perform lookups. Defaults to None.
bootstrap	bool	If True, performs lookups via ARIN bootstrap rather than lookups based on ASN data. ASN lookups are not performed and no output for any of the asn* fields is provided. Defaults to False.
rate_limit_timeout	int	The number of seconds to wait before retrying when a rate limit notice is returned via rdap+json. Defaults to 120.
extra_org_map	dict	Dictionary mapping org handles to RIRs. This is for limited cases where ARIN REST (ASN fallback HTTP lookup) does not show an RIR as the org handle e.g., DNIC (which is now built in ORG_MAP) e.g., {'DNIC': 'arin'}. Valid RIR values are (note the case-sensitive - this is meant to match the REST result): 'ARIN', 'RIPE', 'apnic', 'lacnic', 'afrinic' Defaults to None.
inc_nir	bool	Whether to retrieve NIR (National Internet Registry) information, if registry is JPNIC (Japan) or KRNIC (Korea). If True, extra network requests will be required. If False, the information returned for JP or KR IPs is severely restricted. Defaults to True.
nir_field_list	list	If provided and inc_nir, a list of fields to parse: ['name', 'handle', 'country', 'address', 'postal_code', 'nameservers', 'created', 'updated', 'contacts'] If None, defaults to all.
asn_methods	list	ASN lookup types to attempt, in order. If None, defaults to all ['dns', 'whois', 'http']
get_asn_description	bool	Whether to run an additional query when pulling ASN information via dns, in order to get the ASN description. Defaults to True.
root_entity_check	bool	If True, will perform additional RDAP HTTP queries for missing entity data at the root level. Defaults to True.

7.2 Output

7.2.1 Results Dictionary

The output dictionary from IPWhois.lookup_rdap(). Contains many nested lists and dictionaries, detailed below this section.

Key	Type	Description
query	str	The IP address
asn	str	Globally unique identifier used for routing information exchange with Autonomous Systems.
asn_cidr	str	Network routing block assigned to an ASN.
asn_country_code	str	ASN assigned country code in ISO 3166-1 format.
asn_date	str	ASN allocation date in ISO 8601 format.
asn_registry	str	ASN assigned regional internet registry.
asn_description	str	The ASN description
network	dict	The assigned network for an IP address. May be a parent or child network. See *Network Dictionary*.
entities	list	list of object names referenced by an RIR network. Map these to the objects dict keys.
objects	dict	The objects (entities) referenced by an RIR network or by other entities (depending on depth parameter). Keys are the object names with values as *Objects Dictionary*.
raw	dict	The raw results dictionary (JSON) if inc_raw is True.
nir	dict	The National Internet Registry results if inc_nir is True. See NIR result

Network Dictionary

The dictionary mapped to the network key in the objects list within *Results Dictionary*.

Key	Type	Description
cidr	str	Network routing block an IP address belongs to.
country	str	Country code registered with the RIR in ISO 3166-1 format.
end_address	str	The last IP address in a network block.
events	list	List of event dictionaries. See *Events Dictionary*.
handle	str	Unique identifier for a registered object.
ip_version	str	IP protocol version (v4 or v6) of an IP address.
links	list	HTTP/HTTPS links provided for an RIR object.
name	str	The identifier assigned to the network registration for an IP address.
notices	list	List of notice dictionaries. See *Notices Dictionary*.
parent_handle	str	Unique identifier for the parent network of a registered network.
remarks	list	List of remark (notice) dictionaries. See *Notices Dictionary*.
start_address	str	The first IP address in a network block.
status	list	List indicating the state of a registered object.
type	str	The RIR classification of a registered network.

Objects Dictionary

The dictionary mapped to the object (entity) key in the objects list within *Results Dictionary*.

Key	Type	Description
contact	dict	Contact information registered with an RIR object. See *Objects Contact Dictionary*.
entities	list	List of object names referenced by an RIR object. Map these to other objects dictionary keys.
events	list	List of event dictionaries. See *Events Dictionary*.
events_actor	list	List of event (no actor) dictionaries. See *Events Dictionary*.
handle	str	Unique identifier for a registered object.
links	list	List of HTTP/HTTPS links provided for an RIR object.
notices	list	List of notice dictionaries. See *Notices Dictionary*.
remarks	list	List of remark (notice) dictionaries. See *Notices Dictionary*.
roles	list	List of roles assigned to a registered object.
status	list	List indicating the state of a registered object.

Objects Contact Dictionary

The contact information dictionary registered to an RIR object. This is the contact key contained in *Objects Dictionary*.

Key	Type	Description
address	list	List of contact postal address dictionaries. Contains key type and value.
email	list	List of contact email address dictionaries. Contains key type and value.
kind	str	The contact information kind (individual, group, org).
name	str	The contact name.
phone	list	List of contact phone number dictionaries. Contains key type and value.
role	str	The contact's role.
title	str	The contact's position or job title.

Events Dictionary

Common to lists in *Network Dictionary* and *Objects Dictionary*. Contained in events and events_actor (no actor).

Key	Type	Description
action	str	The reason for an event.
timestamp	str	The date an event occured in ISO 8601 format.
actor	str	The identifier for an event initiator (if any).

Notices Dictionary

Common to lists in *Network Dictionary* and *Objects Dictionary*. Contained in notices and remarks.

Key	Type	Description
title	str	The title/header for a notice.
description	str	The description/body of a notice.
links	list	list of HTTP/HTTPS links provided for a notice.

7.3 Usage Examples

7.3.1 Basic usage

```
>>>> from ipwhois import IPWhois
>>>> from pprint import pprint

>>>> obj = IPWhois('74.125.225.229')
>>>> results = obj.lookup_rdap(depth=1)
>>>> pprint(results)

{
"asn": "15169",
"asn_cidr": "74.125.225.0/24",
"asn_country_code": "US",
"asn_date": "2007-03-13",
"asn_description": "GOOGLE - Google Inc., US",
"asn_registry": "arin",
"entities": [
    "GOGL"
],
"network": {
    "cidr": "74.125.0.0/16",
    "country": None,
    "end_address": "74.125.255.255",
    "events": [
        {
            "action": "last changed",
            "actor": None,
            "timestamp": "2012-02-24T09:44:34-05:00"
        },
        {
            "action": "registration",
```

(continues on next page)

(continued from previous page)


```
            "actor": None,
            "timestamp": "2007-03-13T12:09:54-04:00"
        }
    ],
    "handle": "NET-74-125-0-0-1",
    "ip_version": "v4",
    "links": [
        "https://rdap.arin.net/registry/ip/074.125.000.000",
        "https://whois.arin.net/rest/net/NET-74-125-0-0-1"
    ],
    "name": "GOOGLE",
    "notices": [
        {
            "description": "By using the ARIN RDAP/Whois service, you are agreeing to␣
↪the RDAP/Whois Terms of Use",
            "links": [
                "https://www.arin.net/whois_tou.html"
            ],
            "title": "Terms of Service"
        }
    ],
    "parent_handle": "NET-74-0-0-0-0",
    "raw": None,
    "remarks": None,
    "start_address": "74.125.0.0",
    "status": None,
    "type": None
},
"nir": None,
"objects": {
    "ABUSE5250-ARIN": {
        "contact": {
            "address": [
                {
                    "type": None,
                    "value": "1600 Amphitheatre Parkway\nMountain␣
↪View\nCA\n94043\nUNITED STATES"
                }
            ],
            "email": [
                {
                    "type": None,
                    "value": "network-abuse@google.com"
                }
            ],
            "kind": "group",
            "name": "Abuse",
            "phone": [
                {
                    "type": [
                        "work",
                        "voice"
                    ],
                    "value": "+1-650-253-0000"
                }
            ],
            "role": None,
```


(continues on next page)

(continued from previous page)


```
            "title": None
        },
        "entities": None,
        "events": [
            {
                "action": "last changed",
                "actor": None,
                "timestamp": "2016-11-08T14:12:52-05:00"
            },
            {
                "action": "registration",
                "actor": None,
                "timestamp": "2015-11-06T15:36:35-05:00"
            }
        ],
        "events_actor": None,
        "handle": "ABUSE5250-ARIN",
        "links": [
            "https://rdap.arin.net/registry/entity/ABUSE5250-ARIN",
            "https://whois.arin.net/rest/poc/ABUSE5250-ARIN"
        ],
        "notices": [
            {
                "description": "By using the ARIN RDAP/Whois service, you are␣
↪agreeing to the RDAP/Whois Terms of Use",
                "links": [
                    "https://www.arin.net/whois_tou.html"
                ],
                "title": "Terms of Service"
            }
        ],
        "raw": None,
        "remarks": [
            {
                "description": "Please note that the recommended way to file abuse␣
↪complaints are located in the following links.\r\n\r\nTo report abuse and illegal␣
↪activity: https://www.google.com/intl/en_US/goodtoknow/online-safety/reporting-
↪abuse/ \r\n\r\nFor legal requests: http://support.google.com/legal \r\n\r\nRegards,
↪\r\nThe Google Team",
                "links": None,
                "title": "Registration Comments"
            }
        ],
        "roles": [
            "abuse"
        ],
        "status": [
            "validated"
        ]
    },
    "GOGL": {
        "contact": {
            "address": [
                {
                    "type": None,
                    "value": "1600 Amphitheatre Parkway\nMountain␣
↪View\nCA\n94043\nUNITED STATES"
```


(continues on next page)

(continued from previous page)


```
                }
            ],
            "email": None,
            "kind": "org",
            "name": "Google Inc.",
            "phone": None,
            "role": None,
            "title": None
        },
        "entities": [
            "ABUSE5250-ARIN",
            "ZG39-ARIN"
        ],
        "events": [
            {
                "action": "last changed",
                "actor": None,
                "timestamp": "2017-01-28T08:32:29-05:00"
            },
            {
                "action": "registration",
                "actor": None,
                "timestamp": "2000-03-30T00:00:00-05:00"
            }
        ],
        "events_actor": None,
        "handle": "GOGL",
        "links": [
            "https://rdap.arin.net/registry/entity/GOGL",
            "https://whois.arin.net/rest/org/GOGL"
        ],
        "notices": None,
        "raw": None,
        "remarks": None,
        "roles": [
            "registrant"
        ],
        "status": None
    },
    "ZG39-ARIN": {
        "contact": {
            "address": [
                {
                    "type": None,
                    "value": "1600 Amphitheatre Parkway\nMountain␣
↪View\nCA\n94043\nUNITED STATES"
                }
            ],
            "email": [
                {
                    "type": None,
                    "value": "arin-contact@google.com"
                }
            ],
            "kind": "group",
            "name": "Google Inc",
            "phone": [
```


(continues on next page)

(continued from previous page)


```
                {
                    "type": [
                        "work",
                        "voice"
                    ],
                    "value": "+1-650-253-0000"
                }
            ],
            "role": None,
            "title": None
        },
        "entities": None,
        "events": [
            {
                "action": "last changed",
                "actor": None,
                "timestamp": "2017-03-13T07:08:09-04:00"
            },
            {
                "action": "registration",
                "actor": None,
                "timestamp": "2000-11-30T13:54:08-05:00"
            }
        ],
        "events_actor": None,
        "handle": "ZG39-ARIN",
        "links": [
            "https://rdap.arin.net/registry/entity/ZG39-ARIN",
            "https://whois.arin.net/rest/poc/ZG39-ARIN"
        ],
        "notices": [
            {
                "description": "By using the ARIN RDAP/Whois service, you are␣
↪agreeing to the RDAP/Whois Terms of Use",
                "links": [
                    "https://www.arin.net/whois_tou.html"
                ],
                "title": "Terms of Service"
            }
        ],
        "raw": None,
        "remarks": None,
        "roles": [
            "administrative",
            "technical"
        ],
        "status": [
            "validated"
        ]
    }
},
"query": "74.125.225.229",
"raw": None
}
```

7.3.2 Use a proxy

```
>>>> from urllib import request
>>>> from ipwhois import IPWhois
>>>> handler = request.ProxyHandler({
        'http': 'http://192.168.0.1:80/',
        'https': 'https://192.168.0.1:443/'
    })
>>>> opener = request.build_opener(handler)
>>>> obj = IPWhois('74.125.225.229', proxy_opener = opener)
```

7.3.3 Optimizing queries for your network

Multiple factors will slow your queries down. Several *Input* arguments assist in optimizing query performance:

bootstrap

False: ASN lookups are performed to determine the correct RIR to query RDAP. This adds minor overhead for single queries.

True: Use ARIN bootstrap (redirection), significantly reducing overall time for bulk queries, but at the sacrifice of not having asn* field data in the results.

depth

This value equates to the number of entity levels deep to search for sub-entity information. Found entities each result in a query to the RIR. The larger this value, the longer a single IP query will take. More queries will cause RIR rate limiting to trigger more often for bulk IP queries (only seen with LACNIC).

retry_count

This is the number of times to retry a query in the case of failure. If a rate limit error (HTTPRateLimitError) is raised, the lookup will wait for rate_limit_timeout seconds before retrying. A combination of adjusting retry_count and rate_limit_timeout is needed to optimize bulk queries.

rate_limit_timeout

When a HTTPRateLimitError is raised, and retry_count > 0, this is the amount of seconds to sleep before retrying the query. Using the default value, or setting this too high, will have a large impact on bulk IP queries. I recommend setting this very low for bulk queries, or disable completely by setting retry_count=0.

Note that setting this result too low may cause a larger number of IP lookups to fail.

root_ent_check

When root level entities (depth=0) are missing vcard data, additional entity specific HTTP lookups are performed. In the past, you would expect depth=0 to mean a single lookup per IP. This was a bug and has been fixed as of v1.2.0. Set this to False to revert back to the old method, although you will be missing entity specific data.

CHAPTER 8

Legacy Whois Lookups

IPWhois.lookup() is deprecated as of v0.12.0 and will be removed. Legacy whois lookups were moved to IPWhois.lookup_whois().

Parsing is currently limited to the keys in the output *Results Dictionary*. This is assuming that those fields are present (for both whois and rwhois).

Some IPs have parent networks listed. The parser attempts to recognize this, and break the networks into individual dictionaries. If a single network has multiple CIDRs, they will be separated by ‘, ‘.

Sometimes, you will see whois information with multiple consecutive same name fields, e.g., Description: some text\nDescription: more text. The parser will recognize this and the returned result will have the values separated by ‘\n’.

8.1 Input

Arguments supported by IPWhois.lookup_whois().

Key	Type	Description
inc_raw	bool	Whether to include the raw whois results in the returned dictionary. Defaults to False.
retry_count	int	The number of times to retry in case socket errors, timeouts, connection resets, etc. are encountered. Defaults to 3.
get_referral	bool	Whether to retrieve referral whois information, if available. Defaults to False.
extra_blacklist	list	Blacklisted whois servers in addition to the global BLACKLIST. Defaults to None.
ignore_referral_errors	bool	Whether to ignore and continue when an exception is encountered on referral whois lookups. Defaults to False.
field_list	list	If provided, a list of fields to parse: ['name', 'handle', 'description', 'country', 'state', 'city', 'address', 'postal_code', 'emails', 'created', 'updated']. If None, defaults to all.
extra_org_map	dict	Dictionary mapping org handles to RIRs. This is for limited cases where ARIN REST (ASN fallback HTTP lookup) does not show an RIR as the org handle e.g., DNIC (which is now built in ORG_MAP) e.g., {'DNIC': 'arin'} Valid RIR values are (note the case-sensitive - this is meant to match the REST result): 'ARIN', 'RIPE', 'apnic', 'lacnic', 'afrinic' Defaults to None.
inc_nir	bool	Whether to retrieve NIR (National Internet Registry) information, if registry is JPNIC (Japan) or KRNIC (Korea). If True, extra network requests will be required. If False, the information returned for JP or KR IPs is severely restricted. Defaults to True.
nir_field_list	list	If provided and inc_nir, a list of fields to parse: ['name', 'handle', 'country', 'address', 'postal_code', 'nameservers', 'created', 'updated', 'contacts'] If None, defaults to all.
asn_methods	list	ASN lookup types to attempt, in order. If None, defaults to all ['dns', 'whois', 'http'].
get_asn_description	bool	Whether to run an additional query when pulling ASN information via dns, in order to get the ASN description. Defaults to True.

8.2 Output

8.2.1 Results Dictionary

The output dictionary from IPWhois.lookup_whois().

Key	Type	Description
query	str	The IP address input
asn	str	Globally unique identifier used for routing information exchange with Autonomous Systems.
asn_cidr	str	Network routing block assigned to an ASN.
asn_country_code	str	ASN assigned country code in ISO 3166-1 format.
asn_date	str	ASN allocation date in ISO 8601 format.
asn_registry	str	ASN assigned regional internet registry.
asn_description	str	The ASN description
nets	list	List of network dictionaries. See *Network Dictionary*.
raw	str	Raw whois results if inc_raw is True.
referral	dict	Referral whois information if get_referral is True and the server isn't blacklisted. See *Referral Dictionary*.
raw_referral	str	Raw referral whois results if the inc_raw parameter is True.
nir	dict	The National Internet Registry results if inc_nir is True. See NIR result

Network Dictionary

The dictionary mapped to the nets key in the *Results Dictionary*.

Key	Type	Description
cidr	str	Network routing block an IP address belongs to.
range	str	Network range an IP address belongs to.
name	str	The identifier assigned to the network registration for an IP address.
handle	str	Unique identifier for a registered network.
description	str	Description for a registered network.
country	str	Country code registered with the RIR in ISO 3166-1 format.
state	str	State for a registered network (if applicable).
city	str	City for a registered network (if applicable).
address	str	The mailing address for a registered network.
postal_code	str	The postal code for a registered network.
emails	list	The email addresses listed for a registered network.
created	str	Network registration date in ISO 8601 format.
updated	str	Network registration updated date in ISO 8601 format.

Referral Dictionary

The dictionary mapped to the referral key in the *Results Dictionary*.

Key	Type	Description
cidr	str	Network routing block an IP address belongs to.
range	str	Network range an IP address belongs to.
name	str	The identifier assigned to the network registration for an IP address.
description	str	Description for a registered network.
country	str	Country code registered in ISO 3166-1 format.
state	str	State for a registered network (if applicable).
city	str	City for a registered network (if applicable).
address	str	The mailing address for a registered network.
postal_code	str	The postal code for a registered network.
emails	list	The email addresses listed for a registered network.
created	str	Network registration date in ISO 8601 format.
updated	str	Network registration updated date in ISO 8601 format.

8.3 Usage Examples

8.3.1 Basic usage

```
>>>> from ipwhois import IPWhois
>>>> from pprint import pprint

>>>> obj = IPWhois('74.125.225.229')
>>>> results = obj.lookup_whois()
>>>> pprint(results)

{
"asn": "15169",
"asn_cidr": "74.125.225.0/24",
"asn_country_code": "US",
"asn_date": "2007-03-13",
```

(continues on next page)

(continued from previous page)

```
"asn_description": "GOOGLE - Google Inc., US",
"asn_registry": "arin",
"nets": [
    {
        "address": "1600 Amphitheatre Parkway",
        "cidr": "74.125.0.0/16",
        "city": "Mountain View",
        "country": "US",
        "created": "2007-03-13",
        "description": "Google Inc.",
        "emails": [
            "network-abuse@google.com",
            "arin-contact@google.com"
        ],
        "handle": "NET-74-125-0-0-1",
        "name": "GOOGLE",
        "postal_code": "94043",
        "range": "74.125.0.0 - 74.125.255.255",
        "state": "CA",
        "updated": "2012-02-24"
    }
],
"nir": None,
"query": "74.125.225.229",
"raw": None,
"raw_referral": None,
"referral": None
}
```

8.3.2 Multiple networks listed and referral whois

```
>>> from ipwhois import IPWhois
>>> from pprint import pprint

>>> obj = IPWhois('38.113.198.252')
>>> results = obj.lookup_whois(get_referral=True)
>>> pprint(results)

{
"asn": "174",
"asn_cidr": "38.0.0.0/8",
"asn_country_code": "US",
"asn_date": "",
"asn_description": "COGENT-174 - Cogent Communications, US",
"asn_registry": "arin",
"nets": [
    {
        "address": "2450 N Street NW",
        "cidr": "38.0.0.0/8",
        "city": "Washington",
        "country": "US",
        "created": "1991-04-16",
        "description": "PSINet, Inc.",
        "emails": [
            "ipalloc@cogentco.com",
```

(continues on next page)

(continued from previous page)


```
            "abuse@cogentco.com",
            "noc@cogentco.com"
        ],
        "handle": "NET-38-0-0-0-1",
        "name": "COGENT-A",
        "postal_code": "20037",
        "range": "38.0.0.0 - 38.255.255.255",
        "state": "DC",
        "updated": "2011-05-20"
    },
    {
        "address": "2450 N Street NW",
        "cidr": "38.112.0.0/13",
        "city": "Washington",
        "country": "US",
        "created": "2003-08-20",
        "description": "PSINet, Inc.",
        "emails": [
            "ipalloc@cogentco.com",
            "abuse@cogentco.com",
            "noc@cogentco.com"
        ],
        "handle": "NET-38-112-0-0-1",
        "name": "COGENT-NB-0002",
        "postal_code": "20037",
        "range": None,
        "state": "DC",
        "updated": "2004-03-11"
    }
],
"nir": None,
"query": "38.113.198.252",
"raw": None,
"raw_referral": None,
"referral": {
    "address": "2450 N Street NW",
    "city": "Washington",
    "country": "US",
    "description": "Cogent communications - IPENG",
    "name": "NET4-2671C60017",
    "postal_code": "20037",
    "state": "DC",
    "updated": "2007-09-18 22:02:09"
}
}
```

CHAPTER 9

NIR (National Internet Registry)

IPWhois.nir provides functionality for national registries which restrict information on regional registries. Currently, JPNIC (Japan) and KRNIC (South Korea) are supported.

9.1 Input (IPWhois Wrapper)

NIR is included by default (inc_nir=True) in the wrapper functions: IPWhois.lookup(), IPWhois.lookup_rdap(). For use with the wrappers, see the following input documentation links:

RDAP documentation:

https://ipwhois.readthedocs.io/en/latest/RDAP.html#input

Legacy Whois documentation:

https://ipwhois.readthedocs.io/en/latest/WHOIS.html#input

9.2 Input (Direct)

If you prefer to use NIRWhois(net).lookup() directly, here are the input arguments for that function call:

Key	Type	Description
nir	str	The NIR to query ('jpnic' or 'krnic').
inc_raw	bool	Whether to include the raw whois results in the returned dictionary. Defaults to False.
retry_count	int	The number of times to retry in case socket errors, timeouts, connection resets, etc. are encountered. Defaults to 3.
response	str	Optional response object, this bypasses the NIR lookup.
field_list	list	If provided, a list of fields to parse: ['name', 'handle', 'country', 'address', 'postal_code', 'nameservers', 'created', 'updated', 'contacts']. If None, defaults to all.
is_offline	bool	Whether to perform lookups offline. If True, response and asn_data must be provided. Primarily used for testing. Defaults to False.

9.3 Output

If calling via an IPWhois wrapper, the NIR results are added to the RDAP/WHOIS result dictionary under the key 'nir'.

9.3.1 Results Dictionary

The NIR output dictionary (key: nir) from IPWhois.lookup() or IPWhois.lookup_whois() results.

Key	Type	Description
query	str	The IP address input
nets	list	List of network dictionaries. See *Network Dictionary*.
raw	str	Raw NIR whois results if inc_raw is True.

Network Dictionary

The dictionary mapped to the nets key in the *Results Dictionary*.

Key	Type	Description
cidr	str	Network routing block an IP address belongs to.
range	str	Network range an IP address belongs to.
name	str	The identifier assigned to the network registration for an IP address.
handle	str	Unique identifier for a registered network.
country	str	Country code registered with the NIR in ISO 3166-1 format.
address	str	The mailing address for a registered network.
postal_code	str	The postal code for a registered network.
name-servers	list	The nameservers listed for a registered network.
created	str	Network registration date in ISO 8601 format.
updated	str	Network registration updated date in ISO 8601 format.
contacts	dict	Dictionary with keys: admin, tech. Values map to contact dictionaries if found. See *Contact Dictionary*.

Contact Dictionary

The contact information dictionary registered to a NIR network object. This is 'contacts' -> 'admin'/'tech' key in *Network Dictionary*.

Key	Type	Description
name	str	The contact's name.
organization	str	The contact's organization.
division	str	The contact's division of the organization.
email	str	Contact email address.
reply_email	str	Contact reply email address.
updated	str	Updated date in ISO 8601 format.
phone	str	Contact phone number.
fax	str	Contact fax number.
title	str	The contact's position or job title.

9.4 Usage Examples

9.4.1 Basic usage

inc_nir defaults to true in IPWhois.lookup_*(), but I will set it here to show the usage and results.

```
>>>> from ipwhois import IPWhois
>>>> from pprint import pprint

>>>> obj = IPWhois('133.1.2.5')
>>>> results = obj.lookup_whois(inc_nir=True)
>>>> pprint(results)

{
"asn": "4730",
"asn_cidr": "133.1.0.0/16",
"asn_country_code": "JP",
"asn_date": "",
"asn_description": "ODINS Osaka University, JP",
"asn_registry": "apnic",
"nets": [
    {
        "address": "Urbannet-Kanda Bldg 4F\n3-6-2 Uchi-Kanda\nChiyoda-ku, Tokyo 101-
↪0047,Japan",
        "cidr": "133.0.0.0/8",
        "city": None,
        "country": "JP",
        "created": None,
        "description": "Japan Network Information Center",
        "emails": [
            "hm-changed@apnic.net",
            "hostmaster@nic.ad.jp",
            "ip-apnic@nic.ad.jp"
        ],
        "handle": "JNIC1-AP",
        "name": "JPNIC-NET-JP-ERX",
        "postal_code": None,
        "range": "133.0.0.0 - 133.255.255.255",
        "state": None,
        "updated": "20120828"
    }
],
"nir": {
    "nets": [
        {
            "address": None,
            "cidr": "133.1.0.0/16",
            "contacts": {
                "admin": {
                    "division": "Department of Information and Communications␣
↪Technology Services",
                    "email": "odins-room@odins.osaka-u.ac.jp",
                    "fax": "06-6879-8988",
                    "name": "Yoshihide, Minami",
                    "organization": "Osaka University",
                    "phone": "06-6879-8815",
                    "reply_email": "reg@jpdirect.jp",
```

(continues on next page)

(continued from previous page)

```
                    "title": "Specialist",
                    "updated": "2015-08-13T09:08:34"
                },
                "tech": {
                    "division": "Department of Information and Communications␣
↪Technology Services",
                    "email": "odins-room@odins.osaka-u.ac.jp",
                    "fax": "06-6879-8988",
                    "name": "Yoshihide, Minami",
                    "organization": "Osaka University",
                    "phone": "06-6879-8815",
                    "reply_email": "reg@jpdirect.jp",
                    "title": "Specialist",
                    "updated": "2015-08-13T09:08:34"
                }
            },
            "country": "JP",
            "created": None,
            "handle": "OSAKAU-NET",
            "name": "Osaka University",
            "nameservers": [
                "a.osaka-u.ac.jp",
                "b.osaka-u.ac.jp",
                "dns-x.sinet.ad.jp"
            ],
            "postal_code": None,
            "range": "133.1.0.1 - 133.1.255.255",
            "updated": "2015-01-14T02:50:03"
        }
    ],
    "query": "133.1.2.5",
    "raw": None
},
"query": "133.1.2.5",
"raw": None,
"raw_referral": None,
"referral": None
}

>>>> results = obj.lookup_rdap(depth=1, inc_nir=True)
>>>> pprint(results)

{
"asn": "4730",
"asn_cidr": "133.1.0.0/16",
"asn_country_code": "JP",
"asn_date": "",
"asn_description": "ODINS Osaka University, JP",
"asn_registry": "apnic",
"entities": [
    "JNIC1-AP"
],
"network": {
    "cidr": "133.0.0.0/8",
    "country": "JP",
    "end_address": "133.255.255.255",
    "events": [
```

(continues on next page)

(continued from previous page)


```
        {
            "action": "last changed",
            "actor": None,
            "timestamp": "2009-10-30T00:51:09Z"
        }
    ],
    "handle": "133.0.0.0 - 133.255.255.255",
    "ip_version": "v4",
    "links": [
        "http://rdap.apnic.net/ip/133.0.0.0/8"
    ],
    "name": "JPNIC-NET-JP-ERX",
    "notices": [
        {
            "description": "Objects returned came from source\nAPNIC",
            "links": None,
            "title": "Source"
        },
        {
            "description": "This is the APNIC WHOIS Database query service. The␣
↪objects are in RDAP format.",
            "links": [
                "http://www.apnic.net/db/dbcopyright.html"
            ],
            "title": "Terms and Conditions"
        }
    ],
    "parent_handle": None,
    "raw": None,
    "remarks": [
        {
            "description": "Japan Network Information Center",
            "links": None,
            "title": "description"
        },
        {
            "description": "133/8 block is an ERX range which transfered from\nARIN␣
↪to APNIC on 2009-10-30\nThe original allocation date was 1997-03-01\nPlease search␣
↪whois.nic.ad.jp for more information\nabout this range\n% whois -h whois.nic.ad.jp␣
↪***.***.***.***/e",
            "links": None,
            "title": "remarks"
        }
    ],
    "start_address": "133.0.0.0",
    "status": None,
    "type": "ALLOCATED PORTABLE"
},
"nir": {
    "nets": [
        {
            "address": None,
            "cidr": "133.1.0.0/16",
            "contacts": {
                "admin": {
                    "division": "Department of Information and Communications␣
↪Technology Services",
```


(continues on next page)

(continued from previous page)


```
                    "email": "odins-room@odins.osaka-u.ac.jp",
                    "fax": "06-6879-8988",
                    "name": "Yoshihide, Minami",
                    "organization": "Osaka University",
                    "phone": "06-6879-8815",
                    "reply_email": "reg@jpdirect.jp",
                    "title": "Specialist",
                    "updated": "2015-08-13T09:08:34"
                },
                "tech": {
                    "division": "Department of Information and Communications␣
↪Technology Services",
                    "email": "odins-room@odins.osaka-u.ac.jp",
                    "fax": "06-6879-8988",
                    "name": "Yoshihide, Minami",
                    "organization": "Osaka University",
                    "phone": "06-6879-8815",
                    "reply_email": "reg@jpdirect.jp",
                    "title": "Specialist",
                    "updated": "2015-08-13T09:08:34"
                }
            },
            "country": "JP",
            "created": None,
            "handle": "OSAKAU-NET",
            "name": "Osaka University",
            "nameservers": [
                "a.osaka-u.ac.jp",
                "b.osaka-u.ac.jp",
                "dns-x.sinet.ad.jp"
            ],
            "postal_code": None,
            "range": "133.1.0.1 - 133.1.255.255",
            "updated": "2015-01-14T02:50:03"
        }
    ],
    "query": "133.1.2.5",
    "raw": None
},
"objects": {
    "JNIC1-AP": {
        "contact": {
            "address": [
                {
                    "type": None,
                    "value": "Urbannet-Kanda Bldg 4F\n3-6-2 Uchi-Kanda\nChiyoda-ku,␣
↪Tokyo 101-0047,Japan"
                }
            ],
            "email": [
                {
                    "type": None,
                    "value": "hostmaster@nic.ad.jp"
                }
            ],
            "kind": "group",
            "name": "Japan Network Information Center",
```


(continues on next page)

(continued from previous page)


```
            "phone": [
                {
                    "type": "voice",
                    "value": "+81-3-5297-2311"
                },
                {
                    "type": "fax",
                    "value": "+81-3-5297-2312"
                }
            ],
            "role": None,
            "title": None
        },
        "entities": None,
        "events": None,
        "events_actor": None,
        "handle": "JNIC1-AP",
        "links": [
            "http://rdap.apnic.net/entity/JNIC1-AP"
        ],
        "notices": None,
        "raw": None,
        "remarks": None,
        "roles": [
            "technical",
            "administrative"
        ],
        "status": None
    }
},
"query": "133.1.2.5",
"raw": None
}
```

CHAPTER 10

IP ASN Lookups

This is new functionality as of v0.15.0. This functionality was migrated from net.Net and is still used by IPWhois.lookup*().

Note: Cymru ASN data should not be considered a primary source for data points like country code.

Message from the Cymru site:

```
The country code, registry, and allocation date are all based on data
obtained directly from the regional registries including: ARIN, RIPE,
AFRINIC, APNIC, LACNIC. The information returned relating to these
categories will only be as accurate as the data present in the RIR
databases.

IMPORTANT NOTE: Country codes are likely to vary significantly from
actual IP locations, and we must strongly advise that the IP to ASN
mapping tool not be used as an IP geolocation (GeoIP) service.
```

https://team-cymru.com/community-services/ip-asn-mapping/

10.1 IP ASN Input

Arguments supported by IPASN.lookup().

Key	Type	Description
inc_raw	bool	Whether to include the raw whois results in the returned dictionary. Defaults to False.
retry_count	int	The number of times to retry in case socket errors, timeouts, connection resets, etc. are encountered. Defaults to 3.
extra_org_map	dict	Dictionary mapping org handles to RIRs. This is for limited cases where ARIN REST (ASN fallback HTTP lookup) does not show an RIR as the org handle e.g., DNIC (which is now built in ORG_MAP) e.g., {'DNIC': 'arin'} Valid RIR values are (note the case-sensitive - this is meant to match the REST result): 'ARIN', 'RIPE', 'apnic', 'lacnic', 'afrinic' Defaults to None.
asn_methods	list	ASN lookup types to attempt, in order. If None, defaults to all ['dns', 'whois', 'http'].
get_asn_description	bool	Whether to run an additional query when pulling ASN information via dns, in order to get the ASN description. Defaults to True.

10.2 IP ASN Output

10.2.1 IP ASN Results Dictionary

The output dictionary from IPASN.lookup().

Key	Type	Description
asn	str	The Autonomous System Number
asn_date	str	The ASN Allocation date
asn_registry	str	The assigned ASN registry
asn_cidr	str	The assigned ASN CIDR
asn_country_code	str	The assigned ASN country code
asn_description	str	The ASN description
raw	str	Raw ASN results if inc_raw is True.

10.3 IP ASN Usage Examples

10.3.1 Basic usage

```
>>> from ipwhois.net import Net
>>> from ipwhois.asn import IPASN
>>> from pprint import pprint

>>> net = Net('2001:43f8:7b0::')
>>> obj = IPASN(net)
>>> results = obj.lookup()
>>> pprint(results)

{
"asn": "37578",
"asn_cidr": "2001:43f8:7b0::/48",
"asn_country_code": "KE",
"asn_date": "2013-03-22",
"asn_description": "Tespok, KE",
"asn_registry": "afrinic"
}
```

CHAPTER 11

ASN Origin Lookups

This is new functionality as of v0.15.0.

Both Whois and HTTP protocols are supported.

RADB is the only query destination at the moment.

Parsing is currently limited to the keys in the output *ASN Origin Results Dictionary*. This is assuming that those fields are present.

11.1 ASN Origin Input

Arguments supported by ASNOrigin.lookup().

Key	Type	Description
asn	str	The autonomous system number (ASN) to lookup. May be in format '1234'/'AS1234'
inc_raw	bool	Whether to include the raw whois results in the returned dictionary. Defaults to False.
retry_count	int	The number of times to retry in case socket errors, timeouts, connection resets, etc. are encountered. Defaults to 3.
response	str	Optional response object, this bypasses the Whois lookup. Defaults to None.
field_list	list	If provided, fields to parse: ['description', 'maintainer', 'updated', 'source']. If None, defaults to all.
asn_methods	list	ASN lookup types to attempt, in order. If None, defaults to all ['whois', 'http'].

11.2 ASN Origin Output

11.2.1 ASN Origin Results Dictionary

The output dictionary from ASNOrigin.lookup().

Key	Type	Description
query	str	The ASN input
nets	list	List of network dictionaries. See *ASN Origin Network Dictionary*.
raw	str	Raw ASN origin whois results if inc_raw is True.

ASN Origin Network Dictionary

The dictionary mapped to the nets key in the *ASN Origin Results Dictionary*.

Key	Type	Description
cidr	str	Network routing block an IP address belongs to.
description	str	Description for a registered network.
maintainer	str	The entity that maintains this network.
updated	str	Network registration updated information.
source	str	The source of this network information.

11.3 ASN Origin Usage Examples

11.3.1 Basic usage

```
>>>> from ipwhois.net import Net
>>>> from ipwhois.asn import ASNOrigin
>>>> from pprint import pprint

>>>> net = Net('2001:43f8:7b0::')
>>>> obj = ASNOrigin(net)
>>>> results = obj.lookup(asn='AS37578')
>>>> pprint(results)

{
"nets": [
    {
        "cidr": "196.6.220.0/24",
        "description": "KIXP Nairobi Management Network",
        "maintainer": "TESPOK-MNT",
        "source": "AFRINIC",
        "updated": "***@isoc.org 20160720"
    },
    {
        "cidr": "2001:43f8:7b0::/48",
        "description": "KIXP Nairobi Management Network",
        "maintainer": "TESPOK-MNT",
        "source": "AFRINIC",
        "updated": "***@isoc.org 20160721"
    }
],
"query": "AS37578",
"raw": None
}
```

CHAPTER 12

Utilities

Many useful utilities are provided for IP addresses outside of whois functionality. The following utilities are used throughout the ipwhois library for validation and parsing.

12.1 Country Codes

The legacy country code listing (iso_3166-1_list_en.xml) is no longer available as a free export from iso.org. Support has been added for iso_3166-1.csv, which is now the default.

Use Legacy XML File:

```
>>>> from ipwhois.utils import get_countries
>>>> countries = get_countries(is_legacy_xml=True)
```

12.2 Human Readable Fields

Human readable translations are available for all result fields (RDAP and Legacy Whois). Translations are currently limited to the short name (_short), the name (_name), and the description (_description).

See the ipwhois CLI (ipwhois_utils_cli.py) for an example.

Import the human readable translation dictionaries

```
>>>> from ipwhois.hr import (HR_ASN, HR_ASN_ORIGIN, HR_RDAP_COMMON,
                            HR_RDAP, HR_WHOIS, HR_WHOIS_NIR)
```

12.3 Usage Examples

12.3.1 IPv4 Strip Zeros

Strip leading zeros in each octet of an IPv4 address string.

```
>>>> from ipwhois.utils import ipv4_lstrip_zeros
>>>> print(ipv4_lstrip_zeros('074.125.025.229'))

74.125.25.229
```

12.3.2 CIDR Calculation

Get a list of CIDR range(s) from a start and end IP address.

```
>>>> from ipwhois.utils import calculate_cidr
>>>> print(calculate_cidr('192.168.0.9', '192.168.5.4'))

['192.168.0.9/32', '192.168.0.10/31', '192.168.0.12/30', '192.168.0.16/28',
'192.168.0.32/27', '192.168.0.64/26', '192.168.0.128/25', '192.168.1.0/24',
'192.168.2.0/23', '192.168.4.0/24', '192.168.5.0/30', '192.168.5.4/32']
```

12.3.3 Check if IP is reserved/defined

Check if an IPv4 or IPv6 address is in a reserved/defined pool.

```
>>>> from ipwhois.utils import (ipv4_is_defined, ipv6_is_defined)
>>>> print(ipv4_is_defined('192.168.0.1'))

(True, 'Private-Use Networks', 'RFC 1918')

>>>> print(ipv6_is_defined('fe80::'))

(True, 'Link-Local', 'RFC 4291, Section 2.5.6')
```

12.3.4 Country Code Mapping

Retrieve a dictionary mapping ISO 3166-1 country codes to country names.

```
>>>> from ipwhois import IPWhois
>>>> from ipwhois.utils import get_countries

>>>> countries = get_countries()
>>>> obj = IPWhois('74.125.225.229')
>>>> results = obj.lookup_whois(False)
>>>> print(countries[results['nets'][0]['country']])

United States
```

12.3.5 Iterable to unique elements (order preserved)

List unique elements, preserving the order. This was taken from the itertools recipes.

```
>>>> from ipwhois.utils import unique_everseen
>>>> print(list(unique_everseen(['b', 'a', 'b', 'a', 'c', 'a', 'b', 'c')))

['b', 'a', 'c']
```

12.3.6 Parse IPs/ports from text/file

Search an input string and/or file, extracting and counting IPv4/IPv6 addresses/networks. Summarizes ports with sub-counts.

```
>>>> from ipwhois.utils import unique_addresses
>>>> from pprint import pprint

>>>> input_data = (
        'You can have IPs like 74.125.225.229, or 2001:4860:4860::8888'
        'Put a port at the end 74.125.225.229:80 or for IPv6: '
        '[2001:4860:4860::8888]:443 or even networks like '
        '74.125.0.0/16 and 2001:4860::/32.'
)

>>>> results = unique_addresses(data=input_data, file_path=None)
>>>> pprint(results)

{'2001:4860:4860::8888': {'count': 2, 'ports': {'443': 1}},
 '2001:4860::/32': {'count': 1, 'ports': {}},
 '74.125.0.0/16': {'count': 1, 'ports': {}},
 '74.125.225.229': {'count': 2, 'ports': {'80': 1}}}
```

12.3.7 Generate random IP addresses

Generate random, unique IPv4/IPv6 addresses that are not defined (can be looked up using ipwhois).

```
>>>> from ipwhois.utils import ipv4_generate_random
>>>> for address in ipv4_generate_random(10):
>>>>     print(address)

71.58.89.10
17.206.180.200
156.94.166.94
36.92.169.70
52.214.0.208
174.254.156.179
33.184.228.52
17.58.3.61
101.151.158.16
61.162.38.154

>>>> from ipwhois.utils import ipv6_generate_random
>>>> for address in ipv6_generate_random(10):
>>>>     print(address)
```

(continues on next page)

(continued from previous page)

```
218e:a9ad:aae4:431c:ff16:eb94:f063:47f7
24ba:3185:a26f:fd30:5756:16d5:b4ab:771b
38ad:f797:360a:d98e:4f3b:b1c8:5811:8425
2c0e:9add:6b48:96c4:d22:2674:8067:2de9
3b72:414b:c387:4650:c4a6:eed3:21a8:ba9b
3d24:4053:dd81:d269:2cdc:91c9:b0f8:830e
32a4:8ef8:807:1bf0:e866:c8d7:d69e:2a52
2a2b:eb87:d368:89ee:6861:555:32c6:d552
2ee6:5445:f1ff:b1c6:d68f:3ee1:1e31:fe34
2c6b:393f:ae7:a0f7:1c2:2e19:bab1:af9c
```

CHAPTER 13

CLI

ipwhois_cli.py and ipwhois_utils_cli.py are command line interfaces for the ipwhois library. When using pip to install ipwhois, the CLI scripts are installed to your Python environment Scripts directory.

- ipwhois_cli.py has full ipwhois.py functionality.
- ipwhois_utils_cli.py has full utils.py functionality.
- The others (net.py, rdap.py, whois.py, nir.py, asn.py) will be included in a future release.

13.1 ipwhois_cli.py

13.1.1 Usage

ipwhois_cli.py [-h] [–whois] [–exclude_nir] [–json] [–hr] [–show_name] [–colorize] [–timeout TIMEOUT] [–proxy_http “PROXY_HTTP”] [–proxy_https “PROXY_HTTPS”] [–inc_raw] [–retry_count RETRY_COUNT] [–asn_methods “ASN_METHODS”] [–extra_org_map “EXTRA_ORG_MAP”] [–skip_asn_description] [–depth COLOR_DEPTH] [–excluded_entities “EXCLUDED_ENTITIES”] [–bootstrap] [–rate_limit_timeout RATE_LIMIT_TIMEOUT] [–get_referral] [–extra_blacklist “EXTRA_BLACKLIST”] [–ignore_referral_errors] [–field_list “FIELD_LIST”] [–nir_field_list “NIR_FIELD_LIST”] –addr “IP”

ipwhois CLI interface

optional arguments:

-h, --help	show this help message and exit
--whois	Retrieve whois data via legacy Whois (port 43) instead of RDAP (default).
--exclude_nir	Disable NIR whois lookups (JPNIC, KRNIC). This is the opposite of the ipwhois inc_nir, in order to enable inc_nir by default in the CLI.
--json	Output results in JSON format.

Output options:

--hr If set, returns results with human readable key translations.

--show_name If this and –hr are set, the key name is shown in parentheses afterits short value

--colorize If set, colorizes the output using ANSI. Should work in most platform consoles.

IPWhois settings:

--timeout TIMEOUT The default timeout for socket connections in seconds.

--proxy_http PROXY_HTTP The proxy HTTP address passed to request.ProxyHandler. User auth can be passed like "http://user:pass@192.168.0.1:80"

--proxy_https PROXY_HTTPS The proxy HTTPS address passed to request.ProxyHandler. User auth can be passed like "https://user:pass@192.168.0.1:443"

Common settings (RDAP & Legacy Whois):

--inc_raw Include the raw whois results in the output.

--retry_count RETRY_COUNT The number of times to retry in case socket errors, timeouts, connection resets, etc. are encountered.

--asn_methods ASN_METHODS List of ASN lookup types to attempt, in order. Defaults to all ['dns', 'whois', 'http'].

--extra_org_map EXTRA_ORG_MAP Dictionary mapping org handles to RIRs. This is for limited cases where ARIN REST (ASN fallback HTTP lookup) does not show an RIR as the org handle e.g., DNIC (which is now the built in ORG_MAP) e.g., {"DNIC": "arin"}. Valid RIR values are (note the case-sensitive - this is meant to match the REST result): 'ARIN', 'RIPE', 'apnic', 'lacnic', 'afrinic'

--skip_asn_description Don't run an additional query when pulling ASN information via dns (to get the ASN description). This is the opposite of the ipwhois get_asn_description argument, in order to enable get_asn_description by default in the CLI.

RDAP settings:

--depth COLOR_DEPTH If not –whois, how many levels deep to run RDAP queries when additional referenced objects are found.

--excluded_entities EXCLUDED_ENTITIES If not –whois, a comma delimited list of entity handles to not perform lookups.

--bootstrap If not –whois, performs lookups via ARIN bootstrap rather than lookups based on ASN data. ASN lookups are not performed and no output for any of the asn* fields is provided.

--rate_limit_timeout RATE_LIMIT_TIMEOUT If not –whois, the number of seconds to wait before retrying when a rate limit notice is returned via rdap+json.

Legacy Whois settings:

--get_referral If –whois, retrieve referral whois information, if available.

--extra_blacklist EXTRA_BLACKLIST If –whois, A list of blacklisted whois servers in addition to the global BLACKLIST.

--ignore_referral_errors If –whois, ignore and continue when an exception is encountered on referral whois lookups.

--field_list FIELD_LIST If –whois, a list of fields to parse: [‘name’, ‘handle’, ‘description’, ‘country’, ‘state’, ‘city’, ‘address’, ‘postal_code’, ‘emails’, ‘created’, ‘updated’]

NIR (National Internet Registry) settings:

--nir_field_list NIR_FIELD_LIST If not –exclude_nir, a list of fields to parse: [‘name’, ‘handle’, ‘country’, ‘address’, ‘postal_code’, ‘nameservers’, ‘created’, ‘updated’, ‘contact_admin’, ‘contact_tech’]

Input (Required):

--addr IP An IPv4 or IPv6 address as a string.

13.1.2 Usage Examples

Basic usage

```
ipwhois_cli.py --addr 74.125.225.229 --hr --show_name --colorize --depth 1
```

13.2 ipwhois_utils_cli.py

13.2.1 Usage

ipwhois_utils_cli.py [-h] [–ipv4_lstrip_zeros IPADDRESS] [–calculate_cidr IPADDRESS IPADDRESS] [–get_countries] [–get_country COUNTRYCODE] [–ipv4_is_defined IPADDRESS] [–ipv6_is_defined IPADDRESS] [–unique_everseen ITERABLE] [–unique_addresses FILEPATH] [–colorize]

ipwhois utilities CLI interface

optional arguments:

-h, --help show this help message and exit

--ipv4_lstrip_zeros IPADDRESS Strip leading zeros in each octet of an IPv4 address.

--calculate_cidr IPADDRESSRANGE Calculate a CIDR range(s) from a start and end IP address. Separate start and end address arguments by space.

--get_countries Output a dictionary containing ISO_3166-1 country codes to names.

--get_country COUNTRYCODE Output the ISO_3166-1 name for a country code.

--ipv4_is_defined IPADDRESS Check if an IPv4 address is defined (in a reserved address range).

--ipv6_is_defined IPADDRESS Check if an IPv6 address is defined (in a reserved address range).

--ipv4_generate_random TOTAL Generate random, unique IPv4 addresses that are not defined (can be looked up using ipwhois).

--ipv6_generate_random TOTAL Generate random, unique IPv6 addresses that are not defined (can be looked up using ipwhois).

--unique_everseen ITERABLE List unique elements from input iterable, preserving the order.

--unique_addresses FILEPATH Search an input file, extracting, counting, and summarizing IPv4/IPv6 addresses/networks.

Output options:

--colorize If set, colorizes the output using ANSI. Should work in most platform consoles.

13.2.2 Usage Examples

ipv4_lstrip_zeros

```
>>>> ipwhois_utils_cli.py --ipv4_lstrip_zeros 074.125.025.229

74.125.25.229
```

calculate_cidr

```
>>>> ipwhois_utils_cli.py --calculate_cidr 192.168.0.9 192.168.5.4

Found 12 CIDR blocks for (192.168.0.9, 192.168.5.4):
192.168.0.9/32
192.168.0.10/31
192.168.0.12/30
192.168.0.16/28
192.168.0.32/27
192.168.0.64/26
192.168.0.128/25
192.168.1.0/24
192.168.2.0/23
192.168.4.0/24
192.168.5.0/30
192.168.5.4/32
```

get_countries

```
>>>> ipwhois_utils_cli.py --get_countries

Found 252 countries:
AD: Andorra
AE: United Arab Emirates
AF: Afghanistan
AG: Antigua and Barbuda
AI: Anguilla
AL: Albania
AM: Armenia
...
```

get_country

```
>>>> ipwhois_utils_cli.py --get_country US

Match found for country code (US):
United States
```

ipv4_is_defined

```
>>>> ipwhois_utils_cli.py --ipv4_is_defined 192.168.0.1

192.168.0.1 is defined:
Name: Private-Use Networks
RFC: RFC 1918
```

ipv6_is_defined

```
>>>> ipwhois_utils_cli.py --ipv6_is_defined fc00::

fc00:: is defined:
Name: Unique Local Unicast
RFC: RFC 4193
```

ipv4_generate_random

```
>>>> ipwhois_utils_cli.py --ipv4_generate_random 5

119.224.47.74
128.106.183.195
54.97.0.158
52.206.105.37
126.180.201.81
```

ipv6_generate_random

```
>>>> ipwhois_utils_cli.py --ipv6_generate_random 5

3e8c:dc93:49c8:57fd:31dd:2963:6332:426e
2e3d:fd84:b57b:9282:91e6:5d4d:18d5:34f1
21d4:9d25:7dd6:e28b:77d7:7ce9:f85f:b34f
3659:2b9:12ed:1eac:fd40:5756:3753:6d2d
2e05:6d47:83fd:5de8:c6cb:85cb:912:fdb1
```

unique_everseen

```
>>>> ipwhois_utils_cli.py --unique_everseen [4,2,6,4,6,2]

Unique everseen:
[4, 2, 6]
```

unique_addresses

```
>>>> ipwhois_utils_cli.py --unique_addresses /tmp/some.file

Found 477 unique addresses:
```

(continues on next page)

(continued from previous page)

```
74.125.225.229: Count: 5, Ports: {'22': 1}
2001:4860::/32: Count: 4, Ports: {'443': 1, '80': 2}
2001:4860:4860::8888: Count: 3, Ports: {}
...
```

CHAPTER 14

Experimental Functions

Caution: Functions in experimental.py contain new functionality that has not yet been widely tested. Bulk lookup support contained here can result in significant system/network resource utilization. Additionally, abuse of this functionality may get you banned by the various services queried by this library. Use at your own discretion.

14.1 Bulk ASN Lookups

The function for retrieving ASN information for multiple IP addresses from Cymru via port 43/tcp (WHOIS).

ipwhois.experimental.get_bulk_asn_whois()

14.1.1 Input

Arguments supported:

Key	Type	Description
addresses	list	List of IP address strings to lookup.
retry_count	int	The number of times to retry in case socket errors, timeouts, connection resets, etc. are encountered. Defaults to 3.
timeout	int	The default timeout for socket connections in seconds. Defaults to 120.

14.1.2 Output

Outputs a string of the raw ASN bulk data, new line separated. The first line is obsolete.

14.1.3 Usage Examples

Basic usage

```
>>>> from ipwhois.experimental import get_bulk_asn_whois
>>>> from pprint import pprint

>>>> ip_list = ['74.125.225.229', '2001:4860:4860::8888', '62.239.237.1',
↪'2a00:2381:ffff::1', '210.107.73.73', '2001:240:10c:1::ca20:9d1d', '200.57.141.161',
↪ '2801:10:c000::', '196.11.240.215', '2001:43f8:7b0::', '133.1.2.5', '115.1.2.3']
>>>> results = get_bulk_asn_whois(addresses=ip_list)
>>>> pprint(results.split('\n'))

[
"Bulk mode; whois.cymru.com [2020-09-15 16:42:29 +0000]",
"15169   | 74.125.225.229   | 74.125.225.0/24     | US | arin     | 2007-03-13 |␣
↪GOOGLE, US",
"15169   | 2001:4860:4860::8888                   | 2001:4860::/32      | US | arin␣
↪     | 2005-03-14 | GOOGLE, US",
"2856    | 62.239.237.1     | 62.239.0.0/16       | GB | ripencc  | 2001-01-02 | BT-
↪UK-AS BTnet UK Regional network, GB",
"2856    | 2a00:2381:ffff::1                      | 2a00:2380::/25      | GB |␣
↪ripencc  | 2007-08-29 | BT-UK-AS BTnet UK Regional network, GB",
"3786    | 210.107.73.73    | 210.107.0.0/17      | KR | apnic    | 1997-08-29 |␣
↪LGDACOM LG DACOM Corporation, KR",
"2497    | 2001:240:10c:1::ca20:9d1d              | 2001:240::/32       | JP |␣
↪apnic    | 2000-03-08 | IIJ Internet Initiative Japan Inc., JP",
"19373   | 200.57.141.161   | 200.57.128.0/20     | MX | lacnic   | 2000-12-04 |␣
↪Triara.com, S.A. de C.V., MX",
"NA      | 2801:10:c000::                         | NA                  | CO |␣
↪lacnic   | 2013-10-29 | NA",
"12091   | 196.11.240.215   | 196.11.240.0/24     | ZA | afrinic  | 1994-07-21 |␣
↪MTNNS-1, ZA",
"37578   | 2001:43f8:7b0::                        | 2001:43f8:7b0::/48  | KE |␣
↪afrinic  | 2013-03-22 | Tespok, KE",
"4730    | 133.1.2.5        | 133.1.0.0/16        | JP | apnic    | 1997-03-01 |␣
↪ODINS Osaka University, JP",
"4766    | 115.1.2.3        | 115.0.0.0/12        | KR | apnic    | 2008-07-01 | KIXS-
↪AS-KR Korea Telecom, KR",
""
]
```

14.2 Bulk RDAP Lookups

The function for bulk retrieving and parsing whois information for a list of IP addresses via HTTP (RDAP). This bulk lookup method uses bulk ASN Whois lookups first to retrieve the ASN for each IP. It then optimizes RDAP queries to achieve the fastest overall time, accounting for rate-limiting RIRs.

ipwhois.experimental.bulk_lookup_rdap()

14.2.1 Input

Arguments supported:

Key	Type	Description
addresses	list	List of IP address strings to lookup.
inc_raw	bool	Whether to include the raw whois results in the returned dictionary. Defaults to False.
retry_count	int	The number of times to retry in case socket errors, timeouts, connection resets, etc. are encountered. Defaults to 3.
depth	int	How many levels deep to run queries when additional referenced objects are found. Defaults to 0.
excluded_entities	list	Entity handles to not perform lookups. Defaults to None.
rate_limit_timeout	int	The number of seconds to wait before retrying when a rate limit notice isreturned via rdap+json. Defaults to 60.
socket_timeout	int	The default timeout for socket connections in seconds. Defaults to 10.
asn_timeout	int	The default timeout for bulk ASN lookups in seconds. Defaults to 240.
proxy_openers	list	List of urllib.request.OpenerDirector proxy openers for single/rotating proxy support. Defaults to None.

14.2.2 Output

The output namedtuple from ipwhois.experimental.bulk_lookup_rdap().

Key	Type	Description
results	dict	IP address keys with the values as dictionaries returned by IPWhois.lookup_rdap()
stats	dict	Stats for the lookup containing the keys identified in *Stats Dictionary*

Stats Dictionary

The stats dictionary returned by ipwhois.experimental.bulk_lookup_rdap()

```
{
    'ip_input_total' (int) - The total number of addresses
        originally provided for lookup via the addresses argument.
    'ip_unique_total' (int) - The total number of unique addresses
        found in the addresses argument.
    'ip_lookup_total' (int) - The total number of addresses that
        lookups were attempted for, excluding any that failed ASN
        registry checks.
    'ip_failed_total' (int) - The total number of addresses that
        lookups failed for. Excludes any that failed initially, but
        succeeded after further retries.
    'lacnic' (dict) -
    {
        'failed' (list) - The addresses that failed to lookup.
            Excludes any that failed initially, but succeeded after
            further retries.
        'rate_limited' (list) - The addresses that encountered
            rate-limiting. Unless an address is also in 'failed',
            it eventually succeeded.
        'total' (int) - The total number of addresses belonging to
            this RIR that lookups were attempted for.
    }
    'ripencc' (dict) - Same as 'lacnic' above.
    'apnic' (dict) - Same as 'lacnic' above.
```

(continues on next page)

(continued from previous page)

```
    'afrinic' (dict) - Same as 'lacnic' above.
    'arin' (dict) - Same as 'lacnic' above.
    'unallocated_addresses' (list) - The addresses that are
        unallocated/failed ASN lookups. These can be addresses that
        are not listed for one of the 5 RIRs (other). No attempt
        was made to perform an RDAP lookup for these.
}
```

14.2.3 Usage Examples

Basic usage

```
>>>> from ipwhois.experimental import bulk_lookup_rdap
>>>> from pprint import pprint

>>>> ip_list = ['74.125.225.229', '2001:4860:4860::8888', '62.239.237.1',
↪'2a00:2381:ffff::1', '210.107.73.73', '2001:240:10c:1::ca20:9d1d', '200.57.141.161',
↪ '2801:10:c000::', '196.11.240.215', '2001:43f8:7b0::', '133.1.2.5', '115.1.2.3']
>>>> results, stats = bulk_lookup_rdap(addresses=ip_list)
>>>> pprint(stats)

{
"afrinic": {
    "failed": [],
    "rate_limited": [],
    "total": 2
},
"apnic": {
    "failed": [],
    "rate_limited": [],
    "total": 4
},
"arin": {
    "failed": [],
    "rate_limited": [],
    "total": 2
},
"ip_failed_total": 0,
"ip_input_total": 12,
"ip_lookup_total": 12,
"ip_unique_total": 12,
"lacnic": {
    "failed": [],
    "rate_limited": [],
    "total": 2
},
"ripencc": {
    "failed": [],
    "rate_limited": [],
    "total": 2
},
"unallocated_addresses": []
}
```

CHAPTER 15

Library Structure

class ipwhois.ipwhois.**IPWhois**(*address*, *timeout=5*, *proxy_opener=None*)

The wrapper class for performing whois/RDAP lookups and parsing for IPv4 and IPv6 addresses.

Parameters

- **address** (str/int/IPv4Address/IPv6Address) – An IPv4 or IPv6 address
- **timeout** (int) – The default timeout for socket connections in seconds. Defaults to 5.
- **proxy_opener** (urllib.request.OpenerDirector) – The request for proxy support. Defaults to None.

lookup_rdap(*inc_raw=False*, *retry_count=3*, *depth=0*, *excluded_entities=None*, *bootstrap=False*, *rate_limit_timeout=120*, *extra_org_map=None*, *inc_nir=True*, *nir_field_list=None*, *asn_methods=None*, *get_asn_description=True*, *root_ent_check=True*)

The function for retrieving and parsing whois information for an IP address via HTTP (RDAP).

This is now the recommended method, as RDAP contains much better information to parse.

Parameters

- **inc_raw** (bool) – Whether to include the raw whois results in the returned dictionary. Defaults to False.
- **retry_count** (int) – The number of times to retry in case socket errors, timeouts, connection resets, etc. are encountered. Defaults to 3.
- **depth** (int) – How many levels deep to run queries when additional referenced objects are found. Defaults to 0.
- **excluded_entities** (list) – Entity handles to not perform lookups. Defaults to None.
- **bootstrap** (bool) – If True, performs lookups via ARIN bootstrap rather than lookups based on ASN data. ASN lookups are not performed and no output for any of the asn* fields is provided. Defaults to False.
- **rate_limit_timeout** (int) – The number of seconds to wait before retrying when a rate limit notice is returned via rdap+json. Defaults to 120.

- **extra_org_map** (dict) – Dictionary mapping org handles to RIRs. This is for limited cases where ARIN REST (ASN fallback HTTP lookup) does not show an RIR as the org handle e.g., DNIC (which is now the built in ORG_MAP) e.g., {'DNIC': 'arin'}. Valid RIR values are (note the case-sensitive - this is meant to match the REST result): 'ARIN', 'RIPE', 'apnic', 'lacnic', 'afrinic' Defaults to None.
- **inc_nir** (bool) – Whether to retrieve NIR (National Internet Registry) information, if registry is JPNIC (Japan) or KRNIC (Korea). If True, extra network requests will be required. If False, the information returned for JP or KR IPs is severely restricted. Defaults to True.
- **nir_field_list** (list) – If provided and inc_nir, a list of fields to parse: ['name', 'handle', 'country', 'address', 'postal_code', 'nameservers', 'created', 'updated', 'contacts'] If None, defaults to all.
- **asn_methods** (list) – ASN lookup types to attempt, in order. If None, defaults to all ['dns', 'whois', 'http'].
- **get_asn_description** (bool) – Whether to run an additional query when pulling ASN information via dns, in order to get the ASN description. Defaults to True.
- **root_ent_check** (bool) – If True, will perform additional RDAP HTTP queries for missing entity data at the root level. Defaults to True.

Returns

The IP RDAP lookup results

```
{
    'query' (str) - The IP address
    'asn' (str) - The Autonomous System Number
    'asn_date' (str) - The ASN Allocation date
    'asn_registry' (str) - The assigned ASN registry
    'asn_cidr' (str) - The assigned ASN CIDR
    'asn_country_code' (str) - The assigned ASN country code
    'asn_description' (str) - The ASN description
    'entities' (list) - Entity handles referred by the top
        level query.
    'network' (dict) - Network information which consists of
        the fields listed in the ipwhois.rdap._RDAPNetwork
        dict.
    'objects' (dict) - Mapping of entity handle->entity dict
        which consists of the fields listed in the
        ipwhois.rdap._RDAPEntity dict. The raw result is
        included for each object if the inc_raw parameter
        is True.
    'raw' (dict) - Whois results in json format if the inc_raw
        parameter is True.
    'nir' (dict) - ipwhois.nir.NIRWhois results if inc_nir is
        True.
}
```

Return type dict

lookup_whois(*inc_raw=False*, *retry_count=3*, *get_referral=False*, *extra_blacklist=None*, *ignore_referral_errors=False*, *field_list=None*, *extra_org_map=None*, *inc_nir=True*, *nir_field_list=None*, *asn_methods=None*, *get_asn_description=True*)

The function for retrieving and parsing whois information for an IP address via port 43 (WHOIS).

Parameters

- **inc_raw** (bool) – Whether to include the raw whois results in the returned dictionary. Defaults to False.
- **retry_count** (int) – The number of times to retry in case socket errors, timeouts, connection resets, etc. are encountered. Defaults to 3.
- **get_referral** (bool) – Whether to retrieve referral whois information, if available. Defaults to False.
- **extra_blacklist** (list) – Blacklisted whois servers in addition to the global BLACKLIST. Defaults to None.
- **ignore_referral_errors** (bool) – Whether to ignore and continue when an exception is encountered on referral whois lookups. Defaults to False.
- **field_list** (list) – If provided, a list of fields to parse: ['name', 'handle', 'description', 'country', 'state', 'city', 'address', 'postal_code', 'emails', 'created', 'updated'] If None, defaults to all.
- **extra_org_map** (dict) – Dictionary mapping org handles to RIRs. This is for limited cases where ARIN REST (ASN fallback HTTP lookup) does not show an RIR as the org handle e.g., DNIC (which is now the built in ORG_MAP) e.g., {'DNIC': 'arin'}. Valid RIR values are (note the case-sensitive - this is meant to match the REST result): 'ARIN', 'RIPE', 'apnic', 'lacnic', 'afrinic' Defaults to None.
- **inc_nir** (bool) – Whether to retrieve NIR (National Internet Registry) information, if registry is JPNIC (Japan) or KRNIC (Korea). If True, extra network requests will be required. If False, the information returned for JP or KR IPs is severely restricted. Defaults to True.
- **nir_field_list** (list) – If provided and inc_nir, a list of fields to parse: ['name', 'handle', 'country', 'address', 'postal_code', 'nameservers', 'created', 'updated', 'contacts'] If None, defaults to all.
- **asn_methods** (list) – ASN lookup types to attempt, in order. If None, defaults to all ['dns', 'whois', 'http'].
- **get_asn_description** (bool) – Whether to run an additional query when pulling ASN information via dns, in order to get the ASN description. Defaults to True.

Returns

The IP whois lookup results

```
{
    'query' (str) - The IP address
    'asn' (str) - The Autonomous System Number
    'asn_date' (str) - The ASN Allocation date
    'asn_registry' (str) - The assigned ASN registry
    'asn_cidr' (str) - The assigned ASN CIDR
    'asn_country_code' (str) - The assigned ASN country code
    'asn_description' (str) - The ASN description
    'nets' (list) - Dictionaries containing network
        information which consists of the fields listed in the
        ipwhois.whois.RIR_WHOIS dictionary.
    'raw' (str) - Raw whois results if the inc_raw parameter
        is True.
    'referral' (dict) - Referral whois information if
        get_referral is True and the server is not blacklisted.
        Consists of fields listed in the ipwhois.whois.RWHOIS
        dictionary.
```

(continues on next page)

(continued from previous page)

```
    'raw_referral' (str) - Raw referral whois results if the
        inc_raw parameter is True.
    'nir' (dict) - ipwhois.nir.NIRWhois() results if inc_nir
        is True.
}
```

Return type dict

class ipwhois.net.**Net**(*address*, *timeout=5*, *proxy_opener=None*)

The class for performing network queries.

Parameters

- **address** (str/int/IPv4Address/IPv6Address) – An IPv4 or IPv6 address
- **timeout** (int) – The default timeout for socket connections in seconds. Defaults to 5.
- **proxy_opener** (urllib.request.OpenerDirector) – The request for proxy support. Defaults to None.

Raises IPDefinedError – The address provided is defined (does not need to be resolved).

get_asn_dns()

The function for retrieving ASN information for an IP address from Cymru via port 53 (DNS).

Returns The raw ASN data.

Return type list

Raises ASNLookupError – The ASN lookup failed.

get_asn_http(*retry_count=3*)

The function for retrieving ASN information for an IP address from Arin via port 80 (HTTP). Currently limited to fetching asn_registry through a Arin whois (REST) lookup. The other values are returned as None to keep a consistent dict output. This should be used as a last chance fallback call behind ASN DNS & ASN Whois lookups.

Parameters **retry_count** (int) – The number of times to retry in case socket errors, timeouts, connection resets, etc. are encountered. Defaults to 3.

Returns The ASN data in json format.

Return type dict

Raises ASNLookupError – The ASN lookup failed.

get_asn_origin_whois(*asn_registry='radb'*, *asn=None*, *retry_count=3*, *server=None*, *port=43*)

The function for retrieving CIDR info for an ASN via whois.

Parameters

- **asn_registry** (str) – The source to run the query against (asn.ASN_ORIGIN_WHOIS).
- **asn** (str) – The AS number (required).
- **retry_count** (int) – The number of times to retry in case socket errors, timeouts, connection resets, etc. are encountered. Defaults to 3.
- **server** (str) – An optional server to connect to.
- **port** (int) – The network port to connect on. Defaults to 43.

Returns The raw ASN origin whois data.

Return type str

Raises

- `WhoisLookupError` – The ASN origin whois lookup failed.
- `WhoisRateLimitError` – The ASN origin Whois request rate limited and retries were exhausted.

get_asn_verbose_dns(*asn=None*)

The function for retrieving the information for an ASN from Cymru via port 53 (DNS). This is needed since IP to ASN mapping via Cymru DNS does not return the ASN Description like Cymru Whois does.

Parameters **asn** (`str`) – The AS number (required).

Returns The raw ASN data.

Return type str

Raises `ASNLookupError` – The ASN lookup failed.

get_asn_whois(*retry_count=3*)

The function for retrieving ASN information for an IP address from Cymru via port 43/tcp (WHOIS).

Parameters **retry_count** (`int`) – The number of times to retry in case socket errors, timeouts, connection resets, etc. are encountered. Defaults to 3.

Returns The raw ASN data.

Return type str

Raises `ASNLookupError` – The ASN lookup failed.

get_host(*retry_count=3*)

The function for retrieving host information for an IP address.

Parameters **retry_count** (`int`) – The number of times to retry in case socket errors, timeouts, connection resets, etc. are encountered. Defaults to 3.

Returns

hostname (str) The hostname returned mapped to the given IP address.

aliaslist (list) Alternate names for the given IP address.

ipaddrlist (list) IPv4/v6 addresses mapped to the same hostname.

Return type namedtuple

Raises `HostLookupError` – The host lookup failed.

get_http_json(*url=None*, *retry_count=3*, *rate_limit_timeout=120*, *headers=None*)

The function for retrieving a json result via HTTP.

Parameters

- **url** (`str`) – The URL to retrieve (required).
- **retry_count** (`int`) – The number of times to retry in case socket errors, timeouts, connection resets, etc. are encountered. Defaults to 3.
- **rate_limit_timeout** (`int`) – The number of seconds to wait before retrying when a rate limit notice is returned via rdap+json or HTTP error 429. Defaults to 60.
- **headers** (`dict`) – The HTTP headers. The Accept header defaults to 'application/rdap+json'.

Returns The data in json format.

Return type dict

Raises

- `HTTPLookupError` – The HTTP lookup failed.
- `HTTPRateLimitError` – The HTTP request rate limited and retries were exhausted.

get_http_raw(*url=None*, *retry_count=3*, *headers=None*, *request_type='GET'*, *form_data=None*)
The function for retrieving a raw HTML result via HTTP.

Parameters

- **url** (`str`) – The URL to retrieve (required).
- **retry_count** (`int`) – The number of times to retry in case socket errors, timeouts, connection resets, etc. are encountered. Defaults to 3.
- **headers** (`dict`) – The HTTP headers. The Accept header defaults to 'text/html'.
- **request_type** (`str`) – Request type 'GET' or 'POST'. Defaults to 'GET'.
- **form_data** (`dict`) – Optional form POST data.

Returns The raw data.

Return type str

Raises `HTTPLookupError` – The HTTP lookup failed.

get_whois(*asn_registry='arin'*, *retry_count=3*, *server=None*, *port=43*, *extra_blacklist=None*)
The function for retrieving whois or rwhois information for an IP address via any port. Defaults to port 43/tcp (WHOIS).

Parameters

- **asn_registry** (`str`) – The NIC to run the query against. Defaults to 'arin'.
- **retry_count** (`int`) – The number of times to retry in case socket errors, timeouts, connection resets, etc. are encountered. Defaults to 3.
- **server** (`str`) – An optional server to connect to. If provided, asn_registry will be ignored.
- **port** (`int`) – The network port to connect on. Defaults to 43.
- **extra_blacklist** (`list` of `str`) – Blacklisted whois servers in addition to the global BLACKLIST. Defaults to None.

Returns The raw whois data.

Return type str

Raises

- `BlacklistError` – Raised if the whois server provided is in the global BLACKLIST or extra_blacklist.
- `WhoisLookupError` – The whois lookup failed.
- `WhoisRateLimitError` – The Whois request rate limited and retries were exhausted.

class ipwhois.rdap.**RDAP**(*net*)
The class for parsing IP address whois information via RDAP: https://tools.ietf.org/html/rfc7483 https://www.arin.net/resources/rdap.html

Parameters **net** (*ipwhois.net.Net*) – The network object.

Raises

- `NetError` – The parameter provided is not an instance of ipwhois.net.Net
- `IPDefinedError` – The address provided is defined (does not need to be resolved).

_get_entity(*entity=None*, *roles=None*, *inc_raw=False*, *retry_count=3*, *asn_data=None*, *bootstrap=False*, *rate_limit_timeout=120*)

The function for retrieving and parsing information for an entity via RDAP (HTTP).

Parameters

- **entity** (`str`) – The entity name to lookup.
- **roles** (`dict`) – The mapping of entity handles to roles.
- **inc_raw** (`bool`, optional) – Whether to include the raw results in the returned dictionary. Defaults to False.
- **retry_count** (`int`) – The number of times to retry in case socket errors, timeouts, connection resets, etc. are encountered. Defaults to 3.
- **asn_data** (`dict`) – Result from *ipwhois.asn.IPASN.lookup*. Optional if the bootstrap parameter is True.
- **bootstrap** (`bool`) – If True, performs lookups via ARIN bootstrap rather than lookups based on ASN data. Defaults to False.
- **rate_limit_timeout** (`int`) – The number of seconds to wait before retrying when a rate limit notice is returned via rdap+json. Defaults to 120.

Returns

result (dict) Consists of the fields listed in the ipwhois.rdap._RDAPEntity dict. The raw result is included for each object if the inc_raw parameter is True.

roles (dict) The mapping of entity handles to roles.

Return type namedtuple

lookup(*inc_raw=False*, *retry_count=3*, *asn_data=None*, *depth=0*, *excluded_entities=None*, *response=None*, *bootstrap=False*, *rate_limit_timeout=120*, *root_ent_check=True*)

The function for retrieving and parsing information for an IP address via RDAP (HTTP).

Parameters

- **inc_raw** (`bool`, optional) – Whether to include the raw results in the returned dictionary. Defaults to False.
- **retry_count** (`int`) – The number of times to retry in case socket errors, timeouts, connection resets, etc. are encountered. Defaults to 3.
- **asn_data** (`dict`) – Result from *ipwhois.asn.IPASN.lookup*. Optional if the bootstrap parameter is True.
- **depth** (`int`) – How many levels deep to run queries when additional referenced objects are found. Defaults to 0.
- **excluded_entities** (`list`) – Entity handles to not perform lookups. Defaults to None.
- **response** (`str`) – Optional response object, this bypasses the RDAP lookup.
- **bootstrap** (`bool`) – If True, performs lookups via ARIN bootstrap rather than lookups based on ASN data. Defaults to False.

- **rate_limit_timeout** (int) – The number of seconds to wait before retrying when a rate limit notice is returned via rdap+json. Defaults to 120.
- **root_ent_check** (bool) – If True, will perform additional RDAP HTTP queries for missing entity data at the root level. Defaults to True.

Returns

The IP RDAP lookup results

```
{
    'query' (str) - The IP address
    'entities' (list) - Entity handles referred by the top
        level query.
    'network' (dict) - Network information which consists of
        the fields listed in the ipwhois.rdap._RDAPNetwork
        dict.
    'objects' (dict) - Mapping of entity handle->entity dict
        which consists of the fields listed in the
        ipwhois.rdap._RDAPEntity dict. The raw result is
        included for each object if the inc_raw parameter
        is True.
}
```

Return type dict

class ipwhois.rdap.**_RDAPCommon**(*json_result*)

The common class for parsing RDAP objects: https://tools.ietf.org/html/rfc7483#section-5

Parameters **json_result** (dict) – The JSON response from an RDAP query.

Raises ValueError – vcard is not a known RDAP object.

_parse()

The function for parsing the JSON response to the vars dictionary.

summarize_events(*events_json*)

The function for summarizing RDAP events in to a unique list. https://tools.ietf.org/html/rfc7483#section-4.5

Parameters **events_json** (dict) – A json mapping of events from RDAP results.

Returns

Unique RDAP events information:

```
 [{
     'action' (str) - The reason for an event.
     'timestamp' (str) - The timestamp for when an event
         occured.
     'actor' (str) - The identifier for an event initiator.
}]
```

Return type list of dict

summarize_links(*links_json*)

The function for summarizing RDAP links in to a unique list. https://tools.ietf.org/html/rfc7483#section-4.2

Parameters **links_json** (dict) – A json mapping of links from RDAP results.

Returns Unique RDAP links.

Return type list of str

summarize_notices(*notices_json*)
The function for summarizing RDAP notices in to a unique list. https://tools.ietf.org/html/rfc7483#section-4.3

Parameters **notices_json** (dict) – A json mapping of notices from RDAP results.

Returns

Unique RDAP notices information:

```
[{
    'title' (str) - The title/header of the notice.
    'description' (str) - The description/body of the notice.
    'links' (list) - Unique links returned by
        :obj:`ipwhois.rdap._RDAPCommon.summarize_links()`.
}]
```

Return type list of dict

class ipwhois.rdap.**_RDAPContact**(*vcard*)
The class for parsing RDAP entity contact information objects: https://tools.ietf.org/html/rfc7483#section-5.1 https://tools.ietf.org/html/rfc7095

Parameters **vcard** (list of list) – The vcard list from an RDAP IP address query.

Raises InvalidEntityContactObject – vcard is not an RDAP entity contact information object.

_parse_address(*val*)
The function for parsing the vcard address.

Parameters **val** (list) – The value to parse.

_parse_email(*val*)
The function for parsing the vcard email addresses.

Parameters **val** (list) – The value to parse.

_parse_kind(*val*)
The function for parsing the vcard kind.

Parameters **val** (list) – The value to parse.

_parse_name(*val*)
The function for parsing the vcard name.

Parameters **val** (list) – The value to parse.

_parse_phone(*val*)
The function for parsing the vcard phone numbers.

Parameters **val** (list) – The value to parse.

_parse_role(*val*)
The function for parsing the vcard role.

Parameters **val** (list) – The value to parse.

_parse_title(*val*)
The function for parsing the vcard title.

Parameters **val** (list) – The value to parse.

parse()
The function for parsing the vcard to the vars dictionary.

class ipwhois.rdap.**_RDAPEntity**(*json_result*)
The class for parsing RDAP entity objects: https://tools.ietf.org/html/rfc7483#section-5.1

Parameters **json_result** (dict) – The JSON response from an RDAP query.

Raises InvalidEntityObject – json_result is not an RDAP entity object.

parse()
The function for parsing the JSON response to the vars dictionary.

class ipwhois.rdap.**_RDAPNetwork**(*json_result*)
The class for parsing RDAP network objects: https://tools.ietf.org/html/rfc7483#section-5.4

Parameters **json_result** (dict) – The JSON response from an RDAP IP address query.

Raises InvalidNetworkObject – json_result is not an RDAP network object.

parse()
The function for parsing the JSON response to the vars dictionary.

class ipwhois.whois.**Whois**(*net*)
The class for parsing via whois

Parameters **net** (*ipwhois.net.Net*) – The network object.

Raises

- NetError – The parameter provided is not an instance of ipwhois.net.Net
- IPDefinedError – The address provided is defined (does not need to be resolved).

get_nets_arin(*response*)
The function for parsing network blocks from ARIN whois data.

Parameters **response** (str) – The response from the ARIN whois server.

Returns

Mapping of networks with start and end positions.

```
[{
    'cidr' (str) - The network routing block
    'start' (int) - The starting point of the network
    'end' (int) - The endpoint point of the network
}]
```

Return type list of dict

get_nets_lacnic(*response*)
The function for parsing network blocks from LACNIC whois data.

Parameters **response** (str) – The response from the LACNIC whois server.

Returns

Mapping of networks with start and end positions.

```
[{
    'cidr' (str) - The network routing block
    'start' (int) - The starting point of the network
    'end' (int) - The endpoint point of the network
}]
```

Return type list of dict

get_nets_other(*response*)
The function for parsing network blocks from generic whois data.

Parameters **response** (str) – The response from the whois/rwhois server.

Returns

Mapping of networks with start and end positions.

```
[{
    'cidr' (str) - The network routing block
    'start' (int) - The starting point of the network
    'end' (int) - The endpoint point of the network
}]
```

Return type list of dict

lookup(*inc_raw=False*, *retry_count=3*, *response=None*, *get_referral=False*, *extra_blacklist=None*, *ignore_referral_errors=False*, *asn_data=None*, *field_list=None*, *is_offline=False*)
The function for retrieving and parsing whois information for an IP address via port 43/tcp (WHOIS).

Parameters

- **inc_raw** (bool, optional) – Whether to include the raw results in the returned dictionary. Defaults to False.
- **retry_count** (int) – The number of times to retry in case socket errors, timeouts, connection resets, etc. are encountered. Defaults to 3.
- **response** (str) – Optional response object, this bypasses the NIR lookup. Required when is_offline=True.
- **get_referral** (bool) – Whether to retrieve referral whois information, if available. Defaults to False.
- **extra_blacklist** (list) – Blacklisted whois servers in addition to the global BLACKLIST. Defaults to None.
- **ignore_referral_errors** (bool) – Whether to ignore and continue when an exception is encountered on referral whois lookups. Defaults to False.
- **asn_data** (dict) – Result from *ipwhois.asn.IPASN.lookup* (required).
- **field_list** (list of str) – If provided, fields to parse. Defaults to:

```
['name', 'handle', 'description', 'country', 'state',
'city', 'address', 'postal_code', 'emails', 'created',
'updated']
```

- **is_offline** (bool) – Whether to perform lookups offline. If True, response and asn_data must be provided. Primarily used for testing. Defaults to False.

Returns

The IP whois lookup results

```
{
    'query' (str) - The IP address
    'asn' (str) - The Autonomous System Number
    'asn_date' (str) - The ASN Allocation date
    'asn_registry' (str) - The assigned ASN registry
```

(continues on next page)

(continued from previous page)

```
    'asn_cidr' (str) - The assigned ASN CIDR
    'asn_country_code' (str) - The assigned ASN country code
    'asn_description' (str) - The ASN description
    'nets' (list) - Dictionaries containing network
        information which consists of the fields listed in the
        ipwhois.whois.RIR_WHOIS dictionary.
    'raw' (str) - Raw whois results if the inc_raw parameter
        is True.
    'referral' (dict) - Referral whois information if
        get_referral is True and the server is not blacklisted.
        Consists of fields listed in the ipwhois.whois.RWHOIS
        dictionary.
    'raw_referral' (str) - Raw referral whois results if the
        inc_raw parameter is True.
}
```

Return type dict

parse_fields(*response*, *fields_dict*, *net_start=None*, *net_end=None*, *dt_format=None*, *field_list=None*)

The function for parsing whois fields from a data input.

Parameters

- **response** (str) – The response from the whois/rwhois server.
- **fields_dict** (dict) – The mapping of fields to regex search values (required).
- **net_start** (int) – The starting point of the network (if parsing multiple networks). Defaults to None.
- **net_end** (int) – The ending point of the network (if parsing multiple networks). Defaults to None.
- **dt_format** (str) – The format of datetime fields if known. Defaults to None.
- **field_list** (list of str) – If provided, fields to parse. Defaults to:

```
['name', 'handle', 'description', 'country', 'state',
'city', 'address', 'postal_code', 'emails', 'created',
'updated']
```

Returns

A dictionary of fields provided in fields_dict, mapping to the results of the regex searches.

Return type dict

class ipwhois.nir.**NIRWhois**(*net*)

The class for parsing whois data for NIRs (National Internet Registry). JPNIC and KRNIC are currently the only NIRs supported. Output varies based on NIR specific whois formatting.

Parameters **net** (*ipwhois.net.Net*) – The network object.

Raises

- NetError – The parameter provided is not an instance of ipwhois.net.Net
- IPDefinedError – The address provided is defined (does not need to be resolved).

get_contact(*response=None*, *nir=None*, *handle=None*, *retry_count=3*, *dt_format=None*)

The function for retrieving and parsing NIR whois data based on NIR_WHOIS contact_fields.

Parameters

- **response** (str) – Optional response object, this bypasses the lookup.
- **nir** (str) – The NIR to query ('jpnic' or 'krnic'). Required if response is None.
- **handle** (str) – For NIRs that have separate contact queries (JPNIC), this is the contact handle to use in the query. Defaults to None.
- **retry_count** (int) – The number of times to retry in case socket errors, timeouts, connection resets, etc. are encountered. Defaults to 3.
- **dt_format** (str) – The format of datetime fields if known. Defaults to None.

Returns

Mapping of the fields provided in contact_fields, to their parsed results.

Return type dict

get_nets_jpnic(*response*)

The function for parsing network blocks from jpnic whois data.

Parameters **response** (str) – The response from the jpnic server.

Returns

Mapping of networks with start and end positions.

```
[{
    'cidr' (str) - The network routing block
    'start' (int) - The starting point of the network
    'end' (int) - The endpoint point of the network
}]
```

Return type list of dict

get_nets_krnic(*response*)

The function for parsing network blocks from krnic whois data.

Parameters **response** (str) – The response from the krnic server.

Returns

Mapping of networks with start and end positions.

```
[{
    'cidr' (str) - The network routing block
    'start' (int) - The starting point of the network
    'end' (int) - The endpoint point of the network
}]
```

Return type list of dict

lookup(*nir=None*, *inc_raw=False*, *retry_count=3*, *response=None*, *field_list=None*, *is_offline=False*)

The function for retrieving and parsing NIR whois information for an IP address via HTTP (HTML scraping).

Parameters

- **nir** (str) – The NIR to query ('jpnic' or 'krnic'). Required if response is None.
- **inc_raw** (bool, optional) – Whether to include the raw results in the returned dictionary. Defaults to False.

- **retry_count** (int) – The number of times to retry in case socket errors, timeouts, connection resets, etc. are encountered. Defaults to 3.
- **response** (str) – Optional response object, this bypasses the NIR lookup. Required when is_offline=True.
- **field_list** (list of str) – If provided, fields to parse. Defaults to ipwhois.nir.BASE_NET.
- **is_offline** (bool) – Whether to perform lookups offline. If True, response and asn_data must be provided. Primarily used for testing.

Returns

The NIR whois results:

```
{
    'query' (str) - The IP address.
    'nets' (list of dict) - Network information which consists
        of the fields listed in the ipwhois.nir.NIR_WHOIS
        dictionary.
    'raw' (str) - Raw NIR whois results if the inc_raw
        parameter is True.
}
```

Return type dict

parse_fields(*response*, *fields_dict*, *net_start=None*, *net_end=None*, *dt_format=None*, *field_list=None*, *hourdelta=0*, *is_contact=False*)

The function for parsing whois fields from a data input.

Parameters

- **response** (str) – The response from the whois/rwhois server.
- **fields_dict** (dict) – The mapping of fields to regex search values (required).
- **net_start** (int) – The starting point of the network (if parsing multiple networks). Defaults to None.
- **net_end** (int) – The ending point of the network (if parsing multiple networks). Defaults to None.
- **dt_format** (str) – The format of datetime fields if known. Defaults to None.
- **field_list** (list of str) – If provided, fields to parse. Defaults to ipwhois.nir.BASE_NET if is_contact is False. Otherwise, defaults to ipwhois.nir.BASE_CONTACT.
- **hourdelta** (int) – The timezone delta for created/updated fields. Defaults to 0.
- **is_contact** (bool) – If True, uses contact information field parsing. Defaults to False.

Returns

A dictionary of fields provided in fields_dict, mapping to the results of the regex searches.

Return type dict

class ipwhois.asn.**ASNOrigin**(*net*)

The class for parsing ASN origin whois data

Parameters **net** (*ipwhois.net.Net*) – A ipwhois.net.Net object.

Raises `NetError` – The parameter provided is not an instance of ipwhois.net.Net

get_nets_radb(*response*, *is_http=False*)
The function for parsing network blocks from ASN origin data.

Parameters

- **response** (`str`) – The response from the RADB whois/http server.
- **is_http** (`bool`) – If the query is RADB HTTP instead of whois, set to True. Defaults to False.

Returns

A list of network block dictionaries

```
[{
    'cidr' (str) - The assigned CIDR
    'start' (int) - The index for the start of the parsed
        network block
    'end' (int) - The index for the end of the parsed network
        block
}]
```

Return type list

lookup(*asn=None*, *inc_raw=False*, *retry_count=3*, *response=None*, *field_list=None*, *asn_methods=None*)
The function for retrieving and parsing ASN origin whois information via port 43/tcp (WHOIS).

Parameters

- **asn** (`str`) – The ASN (required).
- **inc_raw** (`bool`) – Whether to include the raw results in the returned dictionary. Defaults to False.
- **retry_count** (`int`) – The number of times to retry in case socket errors, timeouts, connection resets, etc. are encountered. Defaults to 3.
- **response** (`str`) – Optional response object, this bypasses the Whois lookup. Defaults to None.
- **field_list** (`list`) – If provided, fields to parse: [‘description’, ‘maintainer’, ‘updated’, ‘source’] If None, defaults to all.
- **asn_methods** (`list`) – ASN lookup types to attempt, in order. If None, defaults to all [‘whois’, ‘http’].

Returns

The ASN origin lookup results

```
{
    'query' (str) - The Autonomous System Number
    'nets' (list) - Dictionaries containing network
        information which consists of the fields listed in the
        ASN_ORIGIN_WHOIS dictionary.
    'raw' (str) - Raw ASN origin whois results if the inc_raw
        parameter is True.
}
```

Return type dict

Raises

- `ValueError` – methods argument requires one of whois, http.
- `ASNOriginLookupError` – ASN origin lookup failed.

parse_fields(*response*, *fields_dict*, *net_start=None*, *net_end=None*, *field_list=None*)
The function for parsing ASN whois fields from a data input.

Parameters

- **response** (`str`) – The response from the whois/rwhois server.
- **fields_dict** (`dict`) – Mapping of fields->regex search values.
- **net_start** (`int`) – The starting point of the network (if parsing multiple networks). Defaults to None.
- **net_end** (`int`) – The ending point of the network (if parsing multiple networks). Defaults to None.
- **field_list** (`list`) – If provided, a list of fields to parse: ['description', 'maintainer', 'updated', 'source'] If None, defaults to all fields.

Returns A dictionary of fields provided in fields_dict.

Return type dict

class ipwhois.asn.**IPASN**(*net*)
The class for parsing ASN data for an IP address.

Parameters **net** (*ipwhois.net.Net*) – A ipwhois.net.Net object.

Raises `NetError` – The parameter provided is not an instance of ipwhois.net.Net

lookup(*inc_raw=False*, *retry_count=3*, *extra_org_map=None*, *asn_methods=None*, *get_asn_description=True*)
The wrapper function for retrieving and parsing ASN information for an IP address.

Parameters

- **inc_raw** (`bool`) – Whether to include the raw results in the returned dictionary. Defaults to False.
- **retry_count** (`int`) – The number of times to retry in case socket errors, timeouts, connection resets, etc. are encountered. Defaults to 3.
- **extra_org_map** (`dict`) – Mapping org handles to RIRs. This is for limited cases where ARIN REST (ASN fallback HTTP lookup) does not show an RIR as the org handle e.g., DNIC (which is now the built in ORG_MAP) e.g., {'DNIC': 'arin'}. Valid RIR values are (note the case-sensitive - this is meant to match the REST result): 'ARIN', 'RIPE', 'apnic', 'lacnic', 'afrinic' Defaults to None.
- **asn_methods** (`list`) – ASN lookup types to attempt, in order. If None, defaults to all: ['dns', 'whois', 'http'].
- **get_asn_description** (`bool`) – Whether to run an additional query when pulling ASN information via dns, in order to get the ASN description. Defaults to True.

Returns

The ASN lookup results

```
{
    'asn' (str) - The Autonomous System Number
    'asn_date' (str) - The ASN Allocation date
    'asn_registry' (str) - The assigned ASN registry
    'asn_cidr' (str) - The assigned ASN CIDR
    'asn_country_code' (str) - The assigned ASN country code
    'asn_description' (str) - The ASN description
    'raw' (str) - Raw ASN results if the inc_raw parameter is
        True.
}
```

Return type dict

Raises

- `ValueError` – methods argument requires one of dns, whois, http.
- `ASNRegistryError` – ASN registry does not match.

parse_fields_dns(*response*)

The function for parsing ASN fields from a dns response.

Parameters **response** (`str`) – The response from the ASN dns server.

Returns

The ASN lookup results

```
{
    'asn' (str) - The Autonomous System Number
    'asn_date' (str) - The ASN Allocation date
    'asn_registry' (str) - The assigned ASN registry
    'asn_cidr' (str) - The assigned ASN CIDR
    'asn_country_code' (str) - The assigned ASN country code
    'asn_description' (None) - Cannot retrieve with this
        method.
}
```

Return type dict

Raises

- `ASNRegistryError` – The ASN registry is not known.
- `ASNParseError` – ASN parsing failed.

parse_fields_http(*response*, *extra_org_map=None*)

The function for parsing ASN fields from a http response.

Parameters

- **response** (`str`) – The response from the ASN http server.
- **extra_org_map** (`dict`) – Dictionary mapping org handles to RIRs. This is for limited cases where ARIN REST (ASN fallback HTTP lookup) does not show an RIR as the org handle e.g., DNIC (which is now the built in ORG_MAP) e.g., {'DNIC': 'arin'}. Valid RIR values are (note the case-sensitive - this is meant to match the REST result): 'ARIN', 'RIPE', 'apnic', 'lacnic', 'afrinic'. Defaults to None.

Returns

The ASN lookup results

```
{
    'asn' (None) - Cannot retrieve with this method.
    'asn_date' (None) - Cannot retrieve with this method.
    'asn_registry' (str) - The assigned ASN registry
    'asn_cidr' (None) - Cannot retrieve with this method.
    'asn_country_code' (None) - Cannot retrieve with this
        method.
    'asn_description' (None) - Cannot retrieve with this
        method.
}
```

Return type dict

Raises

- ASNRegistryError – The ASN registry is not known.
- ASNParseError – ASN parsing failed.

parse_fields_verbose_dns(*response*)

The function for parsing ASN fields from a verbose dns response.

Parameters **response** (str) – The response from the ASN dns server.

Returns

The ASN lookup results

```
{
    'asn' (str) - The Autonomous System Number
    'asn_date' (str) - The ASN Allocation date
    'asn_registry' (str) - The assigned ASN registry
    'asn_cidr' (None) - Cannot retrieve with this method.
    'asn_country_code' (str) - The assigned ASN country code
    'asn_description' (str) - The ASN description
}
```

Return type dict

Raises

- ASNRegistryError – The ASN registry is not known.
- ASNParseError – ASN parsing failed.

parse_fields_whois(*response*)

The function for parsing ASN fields from a whois response.

Parameters **response** (str) – The response from the ASN whois server.

Returns

The ASN lookup results

```
{
    'asn' (str) - The Autonomous System Number
    'asn_date' (str) - The ASN Allocation date
    'asn_registry' (str) - The assigned ASN registry
    'asn_cidr' (str) - The assigned ASN CIDR
    'asn_country_code' (str) - The assigned ASN country code
    'asn_description' (str) - The ASN description
}
```

Return type dict

Raises

- `ASNRegistryError` – The ASN registry is not known.
- `ASNParseError` – ASN parsing failed.

`ipwhois.utils.`**`calculate_cidr`**(*start_address*, *end_address*)
The function to calculate a CIDR range(s) from a start and end IP address.

Parameters

- **`start_address`** (`str`) – The starting IP address.
- **`end_address`** (`str`) – The ending IP address.

Returns The calculated CIDR ranges.

Return type list of str

`ipwhois.utils.`**`get_countries`**(*is_legacy_xml=False*)
The function to generate a dictionary containing ISO_3166-1 country codes to names.

Parameters **`is_legacy_xml`** (`bool`) – Whether to use the older country code list (iso_3166-1_list_en.xml).

Returns

A mapping of country codes as the keys to the country names as the values.

Return type dict

`ipwhois.utils.`**`ipv4_generate_random`**(*total=100*)
The generator to produce random, unique IPv4 addresses that are not defined (can be looked up using ipwhois).

Parameters **`total`** (`int`) – The total number of IPv4 addresses to generate.

Yields *str* – The next IPv4 address.

`ipwhois.utils.`**`ipv4_is_defined`**(*address*)
The function for checking if an IPv4 address is defined (does not need to be resolved).

Parameters **`address`** (`str`) – An IPv4 address.

Returns

is_defined (bool) True if given address is defined, otherwise False

ietf_name (str) IETF assignment name if given address is defined, otherwise ''

ietf_rfc (str) IETF assignment RFC if given address is defined, otherwise ''

Return type namedtuple

`ipwhois.utils.`**`ipv4_lstrip_zeros`**(*address*)
The function to strip leading zeros in each octet of an IPv4 address.

Parameters **`address`** (`str`) – An IPv4 address.

Returns The modified IPv4 address.

Return type str

`ipwhois.utils.`**`ipv6_generate_random`**(*total=100*)
The generator to produce random, unique IPv6 addresses that are not defined (can be looked up using ipwhois).

Parameters **`total`** (`int`) – The total number of IPv6 addresses to generate.

Yields *str* – The next IPv6 address.

ipwhois.utils.**ipv6_is_defined**(*address*)
The function for checking if an IPv6 address is defined (does not need to be resolved).

Parameters **address** (str) – An IPv6 address.

Returns

is_defined (bool) True if given address is defined, otherwise False

ietf_name (str) IETF assignment name if given address is defined, otherwise ''

ietf_rfc (str) IETF assignment RFC if given address is defined, otherwise ''

Return type namedtuple

ipwhois.utils.**unique_addresses**(*data=None*, *file_path=None*)
The function to search an input string and/or file, extracting and counting IPv4/IPv6 addresses/networks. Summarizes ports with sub-counts. If both a string and file_path are provided, it will process them both.

Parameters

- **data** (str) – The data to process.
- **file_path** (str) – An optional file path to process.

Returns

The addresses/networks mapped to ports and counts:

```
{
    '1.2.3.4' (dict) - Each address or network found is a
        dictionary:
        {
            'count' (int) - Total number of times seen.
            'ports' (dict) - Mapping of port numbers as keys and
                the number of times seen for this ip as values.
        }
}
```

Return type dict

Raises ValueError – Arguments provided are invalid.

ipwhois.utils.**unique_everseen**(*iterable*, *key=None*)
The generator to list unique elements, preserving the order. Remember all elements ever seen. This was taken from the itertools recipes.

Parameters

- **iterable** (iter) – An iterable to process.
- **key** (callable) – Optional function to run when checking elements (e.g., str.lower)

Yields The next unique element found.

exception ipwhois.exceptions.**ASNLookupError**
An Exception for when the ASN lookup failed.

exception ipwhois.exceptions.**ASNOriginLookupError**
An Exception for when the ASN origin lookup failed.

exception ipwhois.exceptions.**ASNParseError**
An Exception for when the ASN parsing failed.

exception ipwhois.exceptions.**ASNRegistryError**
An Exception for when the ASN registry does not match one of the five expected values (arin, ripencc, apnic, lacnic, afrinic).

exception ipwhois.exceptions.**BaseIpwhoisException**
Base exception for all the ipwhois custom ones.

exception ipwhois.exceptions.**BlacklistError**
An Exception for when the server is in a blacklist.

exception ipwhois.exceptions.**HTTPLookupError**
An Exception for when the RDAP lookup failed.

exception ipwhois.exceptions.**HTTPRateLimitError**
An Exception for when HTTP queries exceed the NIC's request limit and have exhausted all retries.

exception ipwhois.exceptions.**HostLookupError**
An Exception for when the host lookup failed.

exception ipwhois.exceptions.**IPDefinedError**
An Exception for when the IP is defined (does not need to be resolved).

exception ipwhois.exceptions.**InvalidEntityContactObject**
An Exception for when JSON output is not an RDAP entity contact information object: https://tools.ietf.org/html/rfc7483#section-5.4

exception ipwhois.exceptions.**InvalidEntityObject**
An Exception for when JSON output is not an RDAP entity object: https://tools.ietf.org/html/rfc7483#section-5.1

exception ipwhois.exceptions.**InvalidNetworkObject**
An Exception for when JSON output is not an RDAP network object: https://tools.ietf.org/html/rfc7483#section-5.4

exception ipwhois.exceptions.**NetError**
An Exception for when a parameter provided is not an instance of ipwhois.net.Net.

exception ipwhois.exceptions.**WhoisLookupError**
An Exception for when the whois lookup failed.

exception ipwhois.exceptions.**WhoisRateLimitError**
An Exception for when Whois queries exceed the NIC's request limit and have exhausted all retries.

ipwhois.experimental.**bulk_lookup_rdap**(*addresses=None*, *inc_raw=False*, *retry_count=3*, *depth=0*, *excluded_entities=None*, *rate_limit_timeout=60*, *socket_timeout=10*, *asn_timeout=240*, *proxy_openers=None*)
The function for bulk retrieving and parsing whois information for a list of IP addresses via HTTP (RDAP). This bulk lookup method uses bulk ASN Whois lookups first to retrieve the ASN for each IP. It then optimizes RDAP queries to achieve the fastest overall time, accounting for rate-limiting RIRs.

Parameters

- **addresses** (list of str) – IP addresses to lookup.
- **inc_raw** (bool, optional) – Whether to include the raw whois results in the returned dictionary. Defaults to False.
- **retry_count** (int) – The number of times to retry in case socket errors, timeouts, connection resets, etc. are encountered. Defaults to 3.
- **depth** (int) – How many levels deep to run queries when additional referenced objects are found. Defaults to 0.

- **excluded_entities** (list of str) – Entity handles to not perform lookups. Defaults to None.
- **rate_limit_timeout** (int) – The number of seconds to wait before retrying when a rate limit notice is returned via rdap+json. Defaults to 60.
- **socket_timeout** (int) – The default timeout for socket connections in seconds. Defaults to 10.
- **asn_timeout** (int) – The default timeout for bulk ASN lookups in seconds. Defaults to 240.
- **proxy_openers** (list of OpenerDirector) – Proxy openers for single/rotating proxy support. Defaults to None.

Returns

results (dict) IP address keys with the values as dictionaries returned by IPWhois.lookup_rdap().

stats (dict) Stats for the lookups:

```
{
    'ip_input_total' (int) - The total number of addresses
        originally provided for lookup via the addresses argument.
    'ip_unique_total' (int) - The total number of unique addresses
        found in the addresses argument.
    'ip_lookup_total' (int) - The total number of addresses that
        lookups were attempted for, excluding any that failed ASN
        registry checks.
    'ip_failed_total' (int) - The total number of addresses that
        lookups failed for. Excludes any that failed initially, but
        succeeded after further retries.
    'lacnic' (dict) -
    {
        'failed' (list) - The addresses that failed to lookup.
            Excludes any that failed initially, but succeeded after
            further retries.
        'rate_limited' (list) - The addresses that encountered
            rate-limiting. Unless an address is also in 'failed',
            it eventually succeeded.
        'total' (int) - The total number of addresses belonging to
            this RIR that lookups were attempted for.
    }
    'ripencc' (dict) - Same as 'lacnic' above.
    'apnic' (dict) - Same as 'lacnic' above.
    'afrinic' (dict) - Same as 'lacnic' above.
    'arin' (dict) - Same as 'lacnic' above.
    'unallocated_addresses' (list) - The addresses that are
        unallocated/failed ASN lookups. These can be addresses that
        are not listed for one of the 5 RIRs (other). No attempt
        was made to perform an RDAP lookup for these.
}
```

Return type namedtuple

Raises ASNLookupError – The ASN bulk lookup failed, cannot proceed with bulk RDAP lookup.

ipwhois.experimental.**get_bulk_asn_whois**(*addresses=None*, *retry_count=3*, *timeout=120*)

The function for retrieving ASN information for multiple IP addresses from Cymru via port 43/tcp (WHOIS).

Parameters

- **addresses** (list of str) – IP addresses to lookup.
- **retry_count** (int) – The number of times to retry in case socket errors, timeouts, connection resets, etc. are encountered. Defaults to 3.
- **timeout** (int) – The default timeout for socket connections in seconds. Defaults to 120.

Returns The raw ASN bulk data, new line separated.

Return type str

Raises

- ValueError – addresses argument must be a list of IPv4/v6 address strings.
- ASNLookupError – The ASN bulk lookup failed.

Python Module Index

i

Index

Symbols

A

B

C

G

H

I

L

N

P

R

S

U

W