# service_identity Documentation

***Release 17.0.0***

**Hynek Schlawack**

**May 23, 2017**

# Contents

Release v17.0.0 (*What's new?*).

Use this package if:

- you use pyOpenSSL and don't want to be MITMed or
- if you want to verify that a PyCA cryptography certificate is valid for a certain hostname.

`service_identity` aspires to give you all the tools you need for verifying whether a certificate is valid for the intended purposes.

In the simplest case, this means *host name verification*. However, `service_identity` implements RFC 6125 fully and plans to add other relevant RFCs too.

`service_identity`'s documentation lives at Read the Docs, the code on GitHub.

CHAPTER 1

# User's Guide

## Installation and Requirements

### Installation

```
$ pip install service_identity
```

### Requirements

Python 2.7, 3.4 and later, as well as PyPy are supported.

Additionally, the following PyPI packages are required:

- attrs
- pyOpenSSL >= 0.14 (0.12 and 0.13 may work but are not part of CI anymore)
- pyasn1
- pyasn1-modules

Optionally, idna >= 0.6 can be used for internationalized domain names (IDN), i.e. non-ASCII domains. Unfortunately it's required because Python's IDN support in the standard library is outdated even in the latest releases.

If you need Python 3.2 support, you will have to use the latest 0.2.x release. If you need Python 2.6 or 3.3 support, you will have to use the latest 14.0.x release. They will receive bug fix releases if necessary but other than that no further development is planned.

## Implemented Standards

### Present

- dNSName with fallback to CN (DNS-ID, aka host names, RFC 6125).

- `uniformResourceIdentifier` (URI-ID, RFC 6125).
- SRV-ID (RFC 6125)

### Future

- `xmppAddr` (RFC 3920).
- `iPAddress` (RFC 2818).
- `nameConstraints` extensions (RFC 3280).

## API

**Note:** The APIs for RFC 6125 verification beyond DNS-IDs (i.e. hostnames) aren't public yet. They are in place and used by the documented high-level APIs though. Eventually they will become public. If you'd like to play with them and provide feedback have a look at the `verify_service_identity` function in the _common module.

### pyOpenSSL

`service_identity.pyopenssl.`**`verify_hostname`**(*connection*, *hostname*)

Verify whether the certificate of *connection* is valid for *hostname*.

**Parameters**

- **`connection`** (*`OpenSSL.SSL.Connection`*) – A pyOpenSSL connection object.
- **`hostname`** (*`unicode`*) – The hostname that *connection* should be connected to.

**Raises**

- ***`service_identity.VerificationError`*** – If *connection* does not provide a certificate that is valid for *hostname*.
- ***`service_identity.CertificateError`*** – If the certificate chain of *connection* contains a certificate that contains invalid/unexpected data.

**Returns** `None`

In practice, this may look like the following:

```
from __future__ import absolute_import, division, print_function

import socket

from OpenSSL import SSL
from service_identity import VerificationError
from service_identity.pyopenssl import verify_hostname


ctx = SSL.Context(SSL.SSLv23_METHOD)
ctx.set_verify(SSL.VERIFY_PEER, lambda conn, cert, errno, depth, ok: ok)
ctx.set_default_verify_paths()

hostname = u"twistedmatrix.com"
```

```
conn = SSL.Connection(ctx, socket.socket(socket.AF_INET, socket.SOCK_STREAM))
conn.connect((hostname, 443))

try:
   conn.do_handshake()
   verify_hostname(conn, hostname)
   # Do your super-secure stuff here.
except SSL.Error as e:
   print("TLS Handshake failed: {0!r}.".format(e.args[0]))
except VerificationError:
   print("Presented certificate is not valid for {0}.".format(hostname))
finally:
   conn.shutdown()
   conn.close()
```

## PyCA cryptography

service_identity.cryptography.**verify_certificate_hostname**(*certificate*, *hostname*)
Verify whether *certificate* is valid for *hostname*.

---

**Note:** Nothing is verified about the *authority* of the certificate; the caller must verify that the certificate chains to an appropriate trust root themselves.

---

**Parameters**

- **certificate** (*cryptography.x509.Certificate*) – A cryptography X509 certificate object.
- **hostname** (*unicode*) – The hostname that *certificate* should be valid for.

**Raises**

- ***service_identity.VerificationError*** – If *certificate* is not valid for *hostname*.
- ***service_identity.CertificateError*** – If *certificate* contains invalid/unexpected data.

**Returns** None

## Universal Errors and Warnings

**exception** service_identity.**VerificationError**(*errors*)
Service identity verification failed.

**exception** service_identity.**CertificateError**
Certificate contains invalid or unexpected data.

**exception** service_identity.**SubjectAltNameWarning**
Server Certificate does not contain a SubjectAltName.

Hostname matching is performed on the CommonName which is deprecated.

# Project Information

## Backward Compatibility

`service_identity` has a very strong backward compatibility policy. Generally speaking, you shouldn't ever be afraid of updating.

If breaking changes are needed do be done, they are:

1. ...announced in the *History*.
2. ...the old behavior raises a `DeprecationWarning` for a year.
3. ...are done with another announcement in the *History*.

## License

`service_identity` is licensed under the MIT license. The full license text can be also found in the source code repository.

## Authors

`service_identity` is written and maintained by Hynek Schlawack.

The development is kindly supported by Variomedia AG.

Other contributors can be found in GitHub's overview.

## How To Contribute

Every open source project lives from the generous help by contributors that sacrifice their time and `service_identity` is no different.

Here are a few guidelines to get you started:

- Try to limit each pull request to one change only.
- To run the test suite, all you need is a recent tox. It will ensure the test suite runs with all dependencies against all Python versions just as it will on Travis CI. If you lack some Python version, you can can always limit the environments like `tox -e py27,py35` (in that case you may want to look into pyenv that makes it very easy to install many different Python versions in parallel).
- Make sure your changes pass our CI. You won't get any feedback until it's green unless you ask for it.
- If your change is noteworthy, add an entry to the changelog.
- No contribution is too small; please submit as many fixes for typos and grammar bloopers as you can!
- Don't break backward compatibility.
- *Always* add tests and docs for your code. This is a hard rule; patches with missing tests or documentation won't be merged.
- Write good test docstrings.
- Obey PEP 8 and PEP 257.
- If you address review feedback, make sure to bump the pull request. Maintainers don't receive notifications if you push new commits.

Please note that this project is released with a Contributor Code of Conduct. By participating in this project you agree to abide by its terms. Please report any harm to Hynek Schlawack in any way you find appropriate. We can usually be found in the `#cryptography-dev` channel on freenode.

Thank you for considering to contribute to `service_identity`!

## Contributor Covenant Code of Conduct

### Our Pledge

In the interest of fostering an open and welcoming environment, we as contributors and maintainers pledge to making participation in our project and our community a harassment-free experience for everyone, regardless of age, body size, disability, ethnicity, gender identity and expression, level of experience, nationality, personal appearance, race, religion, or sexual identity and orientation.

### Our Standards

Examples of behavior that contributes to creating a positive environment include:

- Using welcoming and inclusive language
- Being respectful of differing viewpoints and experiences
- Gracefully accepting constructive criticism
- Focusing on what is best for the community
- Showing empathy towards other community members

Examples of unacceptable behavior by participants include:

- The use of sexualized language or imagery and unwelcome sexual attention or advances
- Trolling, insulting/derogatory comments, and personal or political attacks
- Public or private harassment
- Publishing others' private information, such as a physical or electronic address, without explicit permission
- Other conduct which could reasonably be considered inappropriate in a professional setting

### Our Responsibilities

Project maintainers are responsible for clarifying the standards of acceptable behavior and are expected to take appropriate and fair corrective action in response to any instances of unacceptable behavior.

Project maintainers have the right and responsibility to remove, edit, or reject comments, commits, code, wiki edits, issues, and other contributions that are not aligned to this Code of Conduct, or to ban temporarily or permanently any contributor for other behaviors that they deem inappropriate, threatening, offensive, or harmful.

### Scope

This Code of Conduct applies both within project spaces and in public spaces when an individual is representing the project or its community. Examples of representing a project or community include using an official project e-mail address, posting via an official social media account, or acting as an appointed representative at an online or offline event. Representation of a project may be further defined and clarified by project maintainers.

### Enforcement

Instances of abusive, harassing, or otherwise unacceptable behavior may be reported by contacting the project team at hs@ox.cx. all complaints will be reviewed and investigated and will result in a response that is deemed necessary and appropriate to the circumstances. The project team is obligated to maintain confidentiality with regard to the reporter of an incident. Further details of specific enforcement policies may be posted separately.

Project maintainers who do not follow or enforce the Code of Conduct in good faith may face temporary or permanent repercussions as determined by other members of the project's leadership.

### Attribution

This Code of Conduct is adapted from the Contributor Covenant, version 1.4, available at http://contributor-covenant.org/version/1/4.

## History

Versions are year-based with a strict backwards-compatibility policy. The third digit is only for regressions.

### 17.0.0 (2017-05-23)

#### Deprecations:

- Since Chrome 58 and Firefox 48 both don't accept certificates that contain only a Common Name, its usage is hereby deprecated in `service_identity` too. We have been raising a warning since 16.0.0 and the support will be removed in mid-2018 for good.

#### Changes:

- When `service_identity.SubjectAltNameWarning` is raised, the Common Name of the certificate is now included in the warning message. #17
- Added `cryptography.x509` backend for verifying certificates. #18
- Wildcards (`*`) are now only allowed if they are the leftmost label in a certificate. This is common practice by all major browsers. #19

---

### 16.0.0 (2016-02-18)

#### Backward-incompatible changes:

- Python 3.3 and 2.6 aren't supported anymore. They may work by chance but any effort to keep them working has ceased.

  The last Python 2.6 release was on October 29, 2013 and isn't supported by the CPython core team anymore. Major Python packages like Django and Twisted dropped Python 2.6 a while ago already.

  Python 3.3 never had a significant user base and wasn't part of any distribution's LTS release.
- pyOpenSSL versions older than 0.14 are not tested anymore. They don't even build on recent OpenSSL versions. Please note that its support may break without further notice.

#### Changes:

- Officially support Python 3.5.
- `service_identity.SubjectAltNameWarning` is now raised if the server certicate lacks a proper `SubjectAltName`. #9
- Add a `__str__` method to `VerificationError`.
- Port from `characteristic` to its spiritual successor attrs.

---

### 14.0.0 (2014-08-22)

#### Changes:

- Switch to year-based version numbers.
- Port to `characteristic` 14.0 (get rid of deprecation warnings).
- Package docs with sdist.

---

### 1.0.0 (2014-06-15)

#### Backward-incompatible changes:

- Drop support for Python 3.2. There is no justification to add complexity and unnecessary function calls for a Python version that nobody uses.

#### Changes:

- Move into the Python Cryptography Authority's GitHub account.
- Move exceptions into `service_identity.exceptions` so tracebacks don't contain private module names.
- Promoting to stable since Twisted 14.0 is optionally depending on `service_identity` now.
- Use characteristic instead of a home-grown solution.
- `idna` 0.6 did some backward-incompatible fixes that broke Python 3 support. This has been fixed now therefore `service_identity` only works with `idna` 0.6 and later. Unfortunately since `idna` doesn't offer version introspection, `service_identity` can't warn about it.

---

### 0.2.0 (2014-04-06)

#### Backward-incompatible changes:

- Refactor into a multi-module package. Most notably, `verify_hostname` and `extract_ids` live in the `service_identity.pyopenssl` module now.
- `verify_hostname` now takes an `OpenSSL.SSL.Connection` for the first argument.

### Changes:

- Less false positives in IP address detection.
- Officially support Python 3.4 too.
- More strict checks for URI_IDs.

---

### 0.1.0 (2014-03-03)

Initial release.

CHAPTER 2

# Indices and tables

- genindex
- search

# Index