
importlib_metadata Documentation

Release 0.1

Brett Cannon, Barry Warsaw

Jun 26, 2020

CONTENTS:

1	Using <code>importlib_metadata</code>	3
1.1	Overview	3
1.2	Functional API	4
1.3	Distributions	6
1.4	Extending the search algorithm	6
2	<code>importlib_metadata</code> NEWS	7
2.1	v1.7.0	7
2.2	v1.6.1	7
2.3	v1.6.0	7
2.4	v1.5.2	7
2.5	v1.5.1	8
2.6	v1.5.0	8
2.7	v1.4.0	8
2.8	v1.3.0	8
2.9	v1.2.0	8
2.10	v1.1.3	8
2.11	v1.1.2	9
2.12	v1.1.1	9
2.13	v1.1.0	9
2.14	v1.0.0	9
2.15	0.23	9
2.16	0.22	10
2.17	0.21	10
2.18	0.20	10
2.19	0.19	10
2.20	0.18	10
2.21	0.17	11
2.22	0.16	11
2.23	0.15	11
2.24	0.14	11
2.25	0.13	11
2.26	0.12	11
2.27	0.11	12
2.28	0.10	12
2.29	0.9	12
2.30	0.8	12
2.31	0.7	13
2.32	0.6	13
2.33	0.5	13

2.34	0.4	13
2.35	0.3	14
2.36	0.2	14
2.37	0.1	14
3	Project details		15
4	Indices and tables		17
	Index		19

`importlib_metadata` is a library which provides an API for accessing an installed package's metadata (see [PEP 566](#)), such as its entry points or its top-level name. This functionality intends to replace most uses of `pkg_resources` [entry point API](#) and [metadata API](#). Along with `importlib.resources` in Python 3.7 and newer (backported as `importlib_resources` for older versions of Python), this can eliminate the need to use the older and less efficient `pkg_resources` package.

`importlib_metadata` is a backport of Python 3.8's standard library `importlib.metadata` module for Python 2.7, and 3.4 through 3.7. Users of Python 3.8 and beyond are encouraged to use the standard library module. When imported on Python 3.8 and later, `importlib_metadata` replaces the `DistributionFinder` behavior from the stdlib, but leaves the API in tact. Developers looking for detailed API descriptions should refer to the Python 3.8 standard library documentation.

The documentation here includes a general *usage* guide.

USING IMPORTLIB_METADATA

`importlib_metadata` is a library that provides for access to installed package metadata. Built in part on Python's import system, this library intends to replace similar functionality in the [entry point API](#) and [metadata API](#) of `pkg_resources`. Along with `importlib.resources` in Python 3.7 and newer (backported as `importlib_resources` for older versions of Python), this can eliminate the need to use the older and less efficient `pkg_resources` package.

By “installed package” we generally mean a third-party package installed into Python's `site-packages` directory via tools such as `pip`. Specifically, it means a package with either a discoverable `dist-info` or `egg-info` directory, and metadata defined by [PEP 566](#) or its older specifications. By default, package metadata can live on the file system or in zip archives on `sys.path`. Through an extension mechanism, the metadata can live almost anywhere.

1.1 Overview

Let's say you wanted to get the version string for a package you've installed using `pip`. We start by creating a virtual environment and installing something into it:

```
$ python3 -m venv example
$ source example/bin/activate
(example) $ pip install importlib_metadata
(example) $ pip install wheel
```

You can get the version string for `wheel` by running the following:

```
(example) $ python
>>> from importlib_metadata import version
>>> version('wheel')
'0.32.3'
```

You can also get the set of entry points keyed by group, such as `console_scripts`, `distutils.commands` and others. Each group contains a sequence of [EntryPoint](#) objects.

You can get the *metadata for a distribution*:

```
>>> list(metadata('wheel'))
['Metadata-Version', 'Name', 'Version', 'Summary', 'Home-page', 'Author', 'Author-
→email', 'Maintainer', 'Maintainer-email', 'License', 'Project-URL', 'Project-URL',
→'Project-URL', 'Keywords', 'Platform', 'Classifier', 'Classifier', 'Classifier',
→'Classifier', 'Classifier', 'Classifier', 'Classifier', 'Classifier', 'Classifier',
→'Classifier', 'Classifier', 'Classifier', 'Requires-Python', 'Provides-Extra',
→'Requires-Dist', 'Requires-Dist']
```

You can also get a *distribution's version number*, list its *constituent files*, and get a list of the distribution's *Distribution requirements*.

1.2 Functional API

This package provides the following functionality via its public API.

1.2.1 Entry points

The `entry_points()` function returns a dictionary of all entry points, keyed by group. Entry points are represented by `EntryPoint` instances; each `EntryPoint` has a `.name`, `.group`, and `.value` attributes and a `.load()` method to resolve the value. There are also `.module`, `.attr`, and `.extras` attributes for getting the components of the `.value` attribute:

```
>>> eps = entry_points()
>>> list(eps)
['console_scripts', 'distutils.commands', 'distutils.setup_keywords', 'egg_info.
↳writers', 'setuptools.installation']
>>> scripts = eps['console_scripts']
>>> wheel = [ep for ep in scripts if ep.name == 'wheel'][0]
>>> wheel
EntryPoint(name='wheel', value='wheel.cli:main', group='console_scripts')
>>> wheel.module
'wheel.cli'
>>> wheel.attr
'main'
>>> wheel.extras
[]
>>> main = wheel.load()
>>> main
<function main at 0x103528488>
```

The group and name are arbitrary values defined by the package author and usually a client will wish to resolve all entry points for a particular group. Read [the setuptools docs](#) for more information on entry points, their definition, and usage.

1.2.2 Distribution metadata

Every distribution includes some metadata, which you can extract using the `metadata()` function:

```
>>> wheel_metadata = metadata('wheel')
```

The keys of the returned data structure¹ name the metadata keywords, and their values are returned unparsed from the distribution metadata:

```
>>> wheel_metadata['Requires-Python']
'>=2.7, !=3.0.*, !=3.1.*, !=3.2.*, !=3.3.*'
```

¹ Technically, the returned distribution metadata object is an `email.message.EmailMessage` instance, but this is an implementation detail, and not part of the stable API. You should only use dictionary-like methods and syntax to access the metadata contents.

1.2.3 Distribution versions

The `version()` function is the quickest way to get a distribution's version number, as a string:

```
>>> version('wheel')
'0.32.3'
```

1.2.4 Distribution files

You can also get the full set of files contained within a distribution. The `files()` function takes a distribution package name and returns all of the files installed by this distribution. Each file object returned is a `PackagePath`, a `pathlib.Path` derived object with additional `dist`, `size`, and `hash` properties as indicated by the metadata. For example:

```
>>> util = [p for p in files('wheel') if 'util.py' in str(p)][0]
>>> util
PackagePath('wheel/util.py')
>>> util.size
859
>>> util.dist
<importlib_metadata._hooks.PathDistribution object at 0x101e0cef0>
>>> util.hash
<FileHash mode: sha256 value: bYkw5oMccfazVCoYQwKkkemoVyMAFoR34mmKBx8R1NI>
```

Once you have the file, you can also read its contents:

```
>>> print(util.read_text())
import base64
import sys
...
def as_bytes(s):
    if isinstance(s, text_type):
        return s.encode('utf-8')
    return s
```

In the case where the metadata file listing files (`RECORD` or `SOURCES.txt`) is missing, `files()` will return `None`. The caller may wish to wrap calls to `files()` in `always_iterable` or otherwise guard against this condition if the target distribution is not known to have the metadata present.

1.2.5 Distribution requirements

To get the full set of requirements for a distribution, use the `requires()` function:

```
>>> requires('wheel')
["pytest (>=3.0.0) ; extra == 'test'", "pytest-cov ; extra == 'test'"]
```

1.3 Distributions

While the above API is the most common and convenient usage, you can get all of that information from the `Distribution` class. A `Distribution` is an abstract object that represents the metadata for a Python package. You can get the `Distribution` instance:

```
>>> from importlib_metadata import Distribution
>>> dist = Distribution('wheel')
```

Thus, an alternative way to get the version number is through the `Distribution` instance:

```
>>> dist.version
'0.32.3'
```

There are all kinds of additional metadata available on the `Distribution` instance:

```
>>> d.metadata['Requires-Python']
'>=2.7, !=3.0.*, !=3.1.*, !=3.2.*, !=3.3.*'
>>> d.metadata['License']
'MIT'
```

The full set of available metadata is not described here. See [PEP 566](#) for additional details.

1.4 Extending the search algorithm

Because package metadata is not available through `sys.path` searches, or package loaders directly, the metadata for a package is found through import system `finders`. To find a distribution package's metadata, `importlib_metadata` queries the list of `meta path finders` on `sys.meta_path`.

By default `importlib_metadata` installs a finder for distribution packages found on the file system. This finder doesn't actually find any *packages*, but it can find the packages' metadata.

The abstract class `importlib.abc.MetaPathFinder` defines the interface expected of finders by Python's import system. `importlib_metadata` extends this protocol by looking for an optional `find_distributions` callable on the finders from `sys.meta_path` and presents this extended interface as the `DistributionFinder` abstract base class, which defines this abstract method:

```
@abc.abstractmethod
def find_distributions(context=DistributionFinder.Context()):
    """Return an iterable of all Distribution instances capable of
    loading the metadata for packages for the indicated ``context``.
    """
```

The `DistributionFinder.Context` object provides `.path` and `.name` properties indicating the path to search and name to match and may supply other relevant context.

What this means in practice is that to support finding distribution package metadata in locations other than the file system, subclass `Distribution` and implement the abstract methods. Then from a custom finder, return instances of this derived `Distribution` in the `find_distributions()` method.

IMPORTLIB_METADATA NEWS

2.1 v1.7.0

2020-06-14

- `PathNotFoundError` now has a custom `__str__` mentioning “package metadata” being missing to help guide users to the cause when the package is installed but no metadata is present. Closes #124.

2.2 v1.6.1

2020-06-05

- Added `Distribution._local()` as a provisional demonstration of how to load metadata for a local package. Implicitly requires that [pep517](#) is installed. Ref #42.
- Ensure inputs to `FastPath` are Unicode. Closes #121.
- Tests now rely on `importlib.resources.files` (and backport) instead of the older `path` function.
- Support any iterable from `find_distributions`. Closes #122.

2.3 v1.6.0

2020-03-27

- Added `module` and `attr` attributes to `EntryPoint`

2.4 v1.5.2

2020-03-26

- Fix redundant entries from `FastPath.zip_children`. Closes #117.

2.5 v1.5.1

2020-03-25

- Improve reliability and consistency of compatibility imports for contextlib and pathlib when running tests. Closes #116.

2.6 v1.5.0

2020-01-29

- Additional performance optimizations in FastPath now saves an additional 20% on a typical call.
- Correct for issue where PyOxidizer finder has no `__module__` attribute. Closes #110.

2.7 v1.4.0

2020-01-10

- Through careful optimization, `distribution()` is 3-4x faster. Thanks to Antony Lee for the contribution. Closes #95.
- When searching through `sys.path`, if any error occurs attempting to list a path entry, that entry is skipped, making the system much more lenient to errors. Closes #94.

2.8 v1.3.0

2019-12-10

- Improve custom finders documentation. Closes #105.

2.9 v1.2.0

2019-12-04

- Once again, drop support for Python 3.4. Ref #104.

2.10 v1.1.3

2019-12-04

- Restored support for Python 3.4 due to improper version compatibility declarations in the v1.1.0 and v1.1.1 releases. Closes #104.

2.11 v1.1.2

2019-12-04

- Repaired project metadata to correctly declare the `python_requires` directive. Closes #103.

2.12 v1.1.1

2019-12-04

- Fixed `repr(EntryPoint)` on PyPy 3 also. Closes #102.

2.13 v1.1.0

2019-12-01

- Dropped support for Python 3.4.
- `EntryPoints` are now pickleable. Closes #96.
- Fixed `repr(EntryPoint)` on PyPy 2. Closes #97.

2.14 v1.0.0

2019-11-30

- Project adopts semver for versioning.
- Removed compatibility shim introduced in 0.23.
- For better compatibility with the stdlib implementation and to avoid the same distributions being discovered by the stdlib and backport implementations, the backport now disables the stdlib `DistributionFinder` during initialization (import time). Closes #91 and closes #100.

2.15 0.23

2019-09-16

- Added a compatibility shim to prevent failures on beta releases of Python before the signature changed to accept the “context” parameter on `find_distributions`. This workaround will have a limited lifespan, not to extend beyond release of Python 3.8 final.

2.16 0.22

2019-09-11

- Renamed `package` parameter to `distribution_name` as **recommended** in the following functions: `distribution`, `metadata`, `version`, `files`, and `requires`. This backward-incompatible change is expected to have little impact as these functions are assumed to be primarily used with positional parameters.

2.17 0.21

2019-09-10

- `importlib.metadata` now exposes the `DistributionFinder` metaclass and references it in the docs for extending the search algorithm.
- Add `Distribution.at` for constructing a `Distribution` object from a known metadata directory on the file system. Closes #80.
- `Distribution` finders now receive a context object that supplies `.path` and `.name` properties. This change introduces a fundamental backward incompatibility for any projects implementing a `find_distributions` method on a `MetaPathFinder`. This new layer of abstraction allows this context to be supplied directly or constructed on demand and opens the opportunity for a `find_distributions` method to solicit additional context from the caller. Closes #85.

2.18 0.20

2019-09-02

- Clarify in the docs that calls to `.files` could return `None` when the metadata is not present. Closes #69.
- Return all requirements and not just the first for dist-info packages. Closes #67.

2.19 0.19

2019-07-28

- Restrain over-eager egg metadata resolution.
- Add support for entry points with colons in the name. Closes #75.

2.20 0.18

2019-06-09

- Parse entry points case sensitively. Closes #68
- Add a version constraint on the backport `configparser` package. Closes #66

2.21 0.17

2019-05-29

- Fix a permission problem in the tests on Windows.

2.22 0.16

2019-05-29

- Don't crash if there exists an EGG-INFO directory on sys.path.

2.23 0.15

2019-05-24

- Fix documentation.

2.24 0.14

2019-05-24

- Removed `local_distribution` function from the API. **This backward-incompatible change removes this behavior summarily.** Projects should remove their reliance on this behavior. A replacement behavior is under review in the [pep517](#) project. Closes #42.

2.25 0.13

2019-05-18

- Update docstrings to match PEP 8. Closes #63.
- Merged modules into one module. Closes #62.

2.26 0.12

2019-05-14

- Add support for eggs. !65; Closes #19.

2.27 0.11

2019-05-09

- Support generic zip files (not just wheels). Closes #59
- Support zip files with multiple distributions in them. Closes #60
- Fully expose the public API in `importlib_metadata.__all__`.

2.28 0.10

2019-05-07

- The `Distribution ABC` is now officially part of the public API. Closes #37.
- Fixed support for older single file egg-info formats. Closes #43.
- Fixed a testing bug when `$CWD` has spaces in the path. Closes #50.
- Add Python 3.8 to the `tox` testing matrix.

2.29 0.9

2019-03-25

- Fixed issue where entry points without an attribute would raise an Exception. Closes #40.
- Removed unused `name` parameter from `entry_points()`. Closes #44.
- `DistributionFinder` classes must now be instantiated before being placed on `sys.meta_path`.

2.30 0.8

2019-01-01

- This library can now discover/enumerate all installed packages. **This backward-incompatible change alters the protocol finders must implement to support distribution package discovery.** Closes #24.
- The signature of `find_distributions()` on custom installer finders should now accept two parameters, `name` and `path` and these parameters must supply defaults.
- The `entry_points()` method no longer accepts a package name but instead returns all entry points in a dictionary keyed by the `EntryPoint.group`. The `resolve` method has been removed. Instead, call `EntryPoint.load()`, which has the same semantics as `pkg_resources` and `entrypoints`. **This is a backward incompatible change.**
- Metadata is now always returned as Unicode text regardless of Python version. Closes #29.
- This library can now discover metadata for a 'local' package (found in the current-working directory). Closes #27.
- Added `files()` function for resolving files from a distribution.
- Added a new `requires()` function, which returns the requirements for a package suitable for parsing by `packaging.requirements.Requirement`. Closes #18.

- The top-level `read_text()` function has been removed. Use `PackagePath.read_text()` on instances returned by the `files()` function. **This is a backward incompatible change.**
- Release dates are now automatically injected into the changelog based on SCM tags.

2.31 0.7

2018-11-27

- Fixed issue where packages with dashes in their names would not be discovered. Closes #21.
- Distribution lookup is now case-insensitive. Closes #20.
- Wheel distributions can no longer be discovered by their module name. Like Path distributions, they must be indicated by their distribution package name.

2.32 0.6

2018-10-07

- Removed `importlib_metadata.distribution` function. Now the public interface is primarily the utility functions exposed in `importlib_metadata.__all__`. Closes #14.
- Added two new utility functions `read_text` and `metadata`.

2.33 0.5

2018-09-18

- Updated README and removed details about `Distribution` class, now considered private. Closes #15.
- Added test suite support for Python 3.4+.
- Fixed `SyntaxErrors` on Python 3.4 and 3.5. !12
- Fixed errors on Windows joining Path elements. !15

2.34 0.4

2018-09-14

- Housekeeping.

2.35 0.3

2018-09-14

- Added usage documentation. Closes #8
- Add support for getting metadata from wheels on `sys.path`. Closes #9

2.36 0.2

2018-09-11

- Added `importlib_metadata.entry_points()`. Closes #1
- Added `importlib_metadata.resolve()`. Closes #12
- Add support for Python 2.7. Closes #4

2.37 0.1

2018-09-09

- Initial release.

PROJECT DETAILS

- Project home: https://gitlab.com/python-devs/importlib_metadata
- Report bugs at: https://gitlab.com/python-devs/importlib_metadata/issues
- Code hosting: https://gitlab.com/python-devs/importlib_metadata.git
- Documentation: http://importlib_metadata.readthedocs.io/

INDICES AND TABLES

- genindex
- modindex
- search

P

Python Enhancement Proposals

PEP 566, 1, 3, 6