PROV Python Package Documentation

Release 2.1.1

Trung Dong Huynh

Jun 24, 2025

CONTENTS

Contents:

CHAPTER

ONE

INTRODUCTION

A library for W3C Provenance Data Model supporting PROV-O (RDF), PROV-XML, PROV-JSON import/export

- Free software: MIT license
- Documentation: http://prov.readthedocs.io/.
- Python 3 only.

1.1 Features

- An implementation of the W3C PROV Data Model in Python.
- In-memory classes for PROV assertions, which can then be output as PROV-N
- Serialization and deserialization support: PROV-O (RDF), PROV-XML and PROV-JSON.
- Exporting PROV documents into various graphical formats (e.g. PDF, PNG, SVG).
- Convert a PROV document to a Networkx MultiDiGraph and back.

1.1.1 Uses

See a short tutorial for using this package.

This package is used extensively by ProvStore, a free online repository for provenance documents.

CHAPTER

TWO

INSTALLATION

Python 3 is required.

At the command line:

```
$ python -m pip install prov
```

CHAPTER

THREE

USAGE

3.1 Simple PROV document

```
import prov.model as prov
import datetime

document = prov.ProvDocument()

document.set_default_namespace('http://anotherexample.org/')
document.add_namespace('ex', 'http://example.org/')

e2 = document.entity('e2', (
    (prov.PROV_TYPE, "File"),
    ('ex:path', "/shared/crime.txt"),
    ('ex:creator', "Alice"),
    ('ex:content', "There was a lot of crime in London last month"),
))

a1 = document.activity('a1', datetime.datetime.now(), None, {prov.PROV_TYPE: "edit"})
# References can be qnames or ProvRecord objects themselves
document.wasGeneratedBy(e2, a1, None, {'ex:fct': "save"})
document.wasAssociatedWith('a1', 'ag2', None, None, {prov.PROV_ROLE: "author"})
document.agent('ag2', {prov.PROV_TYPE: 'prov:Person', 'ex:name': "Bob"})

document.get_provn() # =>

# document
#   default <http://anotherexample.org/>
#   prefix ex <http://example.org/>
#
#   entity(e2, [prov:type="File", ex:creator="Alice",
#               ex:content="There was a lot of crime in London last month",
#               ex:path="/shared/crime.txt"])
#   activity(a1, 2014-07-09T16:39:38.795839, -, [prov:type="edit"])
#   wasGeneratedBy(e2, a1, -, [ex:fct="save"])
#   wasAssociatedWith(a1, ag2, -, [prov:role="author"])
#   agent(ag2, [prov:type="prov:Person", ex:name="Bob"])
# endDocument
```

3.2 PROV document with a bundle

```
import prov.model as prov

document = prov.ProvDocument()

document.set_default_namespace('http://example.org/0/')
document.add_namespace('ex1', 'http://example.org/1/')
document.add_namespace('ex2', 'http://example.org/2/')

document.entity('e001')

bundle = document.bundle('e001')
bundle.set_default_namespace('http://example.org/2/')
bundle.entity('e001')

document.get_provn() # =>

# document
#   default <http://example.org/0/>
#   prefix ex2 <http://example.org/2/>
#   prefix ex1 <http://example.org/1/>
#
#   entity(e001)
#   bundle e001
#     default <http://example.org/2/>
#
#     entity(e001)
#   endBundle
# endDocument

document.serialize() # =>

# {"prefix": {"default": "http://example.org/0/", "ex2": "http://example.org/2/", "ex1":
↪"http://example.org/1/"}, "bundle": {"e001": {"prefix": {"default": "http://example.
↪org/2/"}, "entity": {"e001": {}}}}, "entity": {"e001": {}}}
```

3.3 More examples

See prov/tests/examples.py

CHAPTER

FOUR

CONTRIBUTING

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

4.1 Types of Contributions

4.1.1 Report Bugs

Report bugs at https://github.com/trungdong/prov/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

4.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" is open to whoever wants to implement it.

4.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "feature" is open to whoever wants to implement it.

4.1.4 Write Documentation

We could always use more documentation, whether as part of the official prov docs, in docstrings, or even on the web in blog posts, articles, and such.

4.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/trungdong/prov/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

4.2 Get Started!

Ready to contribute? Here's how to set up *prov* for local development.

1. Fork the *prov* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/prov.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv prov
$ cd prov/
$ pip install -r requirements-dev.txt
```

(NOTE: To be updated. The above step is no longer correct.)

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 prov tests
$ python setup.py test
$ tox
```

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

4.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 3.9+ and for PyPy3. Look for the automated checks at the bottom of your pull request and make sure that the tests pass for all supported Python versions.

CHAPTER

FIVE

PROV

5.1 prov package

5.1.1 Subpackages

prov.serializers package

Module contents

class prov.serializers.**Registry**

Bases: object

Registry of serializers.

static load_serializers() → None

Loads all available serializers into the registry.

serializers: dict[str, type[*Serializer*]] = None

Property caching all available serializers in a dict.

class prov.serializers.**Serializer**(*document:* ProvDocument | *None* = *None*)

Bases: ABC

Serializer for PROV documents.

abstractmethod deserialize(*stream: io.IOBase*, ***args: Any*) → *ProvDocument*

Abstract method for deserializing.

Parameters

stream – Stream object to deserialize the document from.

document = None

PROV document to serialise.

abstractmethod serialize(*stream: IOBase*, ***args: Any*) → None

Abstract method for serializing.

Parameters

stream – Stream object to serialize the document into.

prov.serializers.**get**(*format_name: str*) → type[*Serializer*]

Returns the serializer class for the specified format. Raises a DoNotExist

prov.serializers.provjson module

class prov.serializers.provjson.**ProvJSONDecoder**(** (Keyword-only parameters separator (PEP 3102))*, *object_hook=None*, *parse_float=None*, *parse_int=None*, *parse_constant=None*, *strict=True*, *object_pairs_hook=None*)

Bases: JSONDecoder

decode(*s: str*, **args: Any*, ***kwargs: Any*) → Any

Return the Python representation of s (a str instance containing a JSON document).

class prov.serializers.provjson.**ProvJSONEncoder**(***, *skipkeys=False*, *ensure_ascii=True*, *check_circular=True*, *allow_nan=True*, *sort_keys=False*, *indent=None*, *separators=None*, *default=None*)

Bases: JSONEncoder

default(*o: Any*) → Any

Implement this method in a subclass such that it returns a serializable object for o, or calls the base implementation (to raise a TypeError).

For example, to support arbitrary iterators, you could implement default like this:

```
def default(self, o):
    try:
        iterable = iter(o)
    except TypeError:
        pass
    else:
        return list(iterable)
    # Let the base class default method raise the TypeError
    return super().default(o)
```

exception prov.serializers.provjson.**ProvJSONException**

Bases: *Error*

class prov.serializers.provjson.**ProvJSONSerializer**(*document:* ProvDocument | *None* = *None*)

Bases: *Serializer*

PROV-JSON serializer for *ProvDocument*

deserialize(*stream: IOBase*, ***args: Any*) → *ProvDocument*

Deserialize from the PROV JSON representation to a *ProvDocument* instance.

Parameters
stream – Input data.

serialize(*stream: IOBase*, ***args: Any*) → None

Serializes a *ProvDocument* instance to PROV-JSON.

Parameters
stream – Where to save the output.

prov.serializers.provn module

class prov.serializers.provn.**ProvNSerializer**(*document:* ProvDocument | *None* = *None*)

Bases: *Serializer*

PROV-N serializer for ProvDocument

deserialize(*stream: IOBase*, ***args: Any*) → *ProvDocument*

Abstract method for deserializing.

Parameters
stream – Stream object to deserialize the document from.

serialize(*stream: IOBase*, ***args: Any*) → None

Serializes a *prov.model.ProvDocument* instance to a PROV-N.

Parameters
stream – Where to save the output.

prov.serializers.provrdf module

PROV-RDF serializers for ProvDocument

exception prov.serializers.provrdf.**ProvRDFException**

Bases: *Error*

class prov.serializers.provrdf.**ProvRDFSerializer**(*document:* ProvDocument | *None* = *None*)

Bases: *Serializer*

PROV-O serializer for *ProvDocument*

deserialize(*stream: ~io.IOBase*, *rdf_format: str = 'trig'*, *relation_mapper: dict[~rdflib.term.URIRef, str] = {rdflib.term.URIRef('http://www.w3.org/ns/prov#actedOnBehalfOf'): 'delegation', rdflib.term.URIRef('http://www.w3.org/ns/prov#alternateOf'): 'alternate', rdflib.term.URIRef('http://www.w3.org/ns/prov#hadMember'): 'membership', rdflib.term.URIRef('http://www.w3.org/ns/prov#mentionOf'): 'mention', rdflib.term.URIRef('http://www.w3.org/ns/prov#specializationOf'): 'specialization', rdflib.term.URIRef('http://www.w3.org/ns/prov#used'): 'usage', rdflib.term.URIRef('http://www.w3.org/ns/prov#wasAssociatedWith'): 'association', rdflib.term.URIRef('http://www.w3.org/ns/prov#wasAttributedTo'): 'attribution', rdflib.term.URIRef('http://www.w3.org/ns/prov#wasDerivedFrom'): 'derivation', rdflib.term.URIRef('http://www.w3.org/ns/prov#wasEndedBy'): 'end', rdflib.term.URIRef('http://www.w3.org/ns/prov#wasGeneratedBy'): 'generation', rdflib.term.URIRef('http://www.w3.org/ns/prov#wasInfluencedBy'): 'influence', rdflib.term.URIRef('http://www.w3.org/ns/prov#wasInformedBy'): 'communication', rdflib.term.URIRef('http://www.w3.org/ns/prov#wasInvalidatedBy'): 'invalidation', rdflib.term.URIRef('http://www.w3.org/ns/prov#wasStartedBy'): 'start'}*, *predicate_mapper: dict[~rdflib.term.URIRef, ~prov.identifier.QualifiedName] = {rdflib.term.URIRef('http://www.w3.org/2000/01/rdf-schema#label'): <QualifiedName: prov:label>, rdflib.term.URIRef('http://www.w3.org/ns/prov#atLocation'): <QualifiedName: prov:location>, rdflib.term.URIRef('http://www.w3.org/ns/prov#atTime'): <QualifiedName: prov:time>, rdflib.term.URIRef('http://www.w3.org/ns/prov#endedAtTime'): <QualifiedName: prov:endTime>, rdflib.term.URIRef('http://www.w3.org/ns/prov#hadActivity'): <QualifiedName: prov:activity>, rdflib.term.URIRef('http://www.w3.org/ns/prov#hadGeneration'): <QualifiedName: prov:generation>, rdflib.term.URIRef('http://www.w3.org/ns/prov#hadPlan'): <QualifiedName: prov:plan>, rdflib.term.URIRef('http://www.w3.org/ns/prov#hadRole'): <QualifiedName: prov:role>, rdflib.term.URIRef('http://www.w3.org/ns/prov#hadUsage'): <QualifiedName: prov:usage>, rdflib.term.URIRef('http://www.w3.org/ns/prov#startedAtTime'): <QualifiedName: prov:startTime>}*, ***kwargs: ~typing.Any*) → *ProvDocument*

Deserialize from the PROV-O representation to a `ProvDocument` instance.

Parameters

- **stream** – Input data.
- **rdf_format** – The RDF format of the input data, default: TRiG.

serialize(*stream: ~io.IOBase*, *rdf_format: str = 'trig'*, *PROV_N_MAP: dict[~prov.identifier.QualifiedName, str] = {<QualifiedName: prov:Activity>: 'activity', <QualifiedName: prov:Agent>: 'agent', <QualifiedName: prov:Alternate>: 'alternateOf', <QualifiedName: prov:Association>: 'wasAssociatedWith', <QualifiedName: prov:Attribution>: 'wasAttributedTo', <QualifiedName: prov:Bundle>: 'bundle', <QualifiedName: prov:Communication>: 'wasInformedBy', <QualifiedName: prov:Delegation>: 'actedOnBehalfOf', <QualifiedName: prov:Derivation>: 'wasDerivedFrom', <QualifiedName: prov:End>: 'wasEndedBy', <QualifiedName: prov:Entity>: 'entity', <QualifiedName: prov:Generation>: 'wasGeneratedBy', <QualifiedName: prov:Influence>: 'wasInfluencedBy', <QualifiedName: prov:Invalidation>: 'wasInvalidatedBy', <QualifiedName: prov:Membership>: 'hadMember', <QualifiedName: prov:Mention>: 'mentionOf', <QualifiedName: prov:Specialization>: 'specializationOf', <QualifiedName: prov:Start>: 'wasStartedBy', <QualifiedName: prov:Usage>: 'used'}*, ***kwargs: ~typing.Any*) → None

Serializes a `ProvDocument` instance to PROV-O.

Parameters

- **stream** – Where to save the output.

- **rdf_format** – The RDF format of the output, default to TRiG.

prov.serializers.provrdf.**walk**(*children: list*, *level: int = 0*, *path: dict = None*, *usename: bool = True*) → Generator[dict]

Generate all the full paths in a tree, as a dict.

Example

```
>>> from prov.serializers.provrdf import walk
>>> iterables = [('a', lambda: [1, 2]), ('b', lambda: [3, 4])]
>>> [val['a'] for val in walk(iterables)]
[1, 1, 2, 2]
>>> [val['b'] for val in walk(iterables)]
[3, 4, 3, 4]
```

prov.serializers.provxml module

exception prov.serializers.provxml.**ProvXMLException**

Bases: *Error*

class prov.serializers.provxml.**ProvXMLSerializer**(*document:* ProvDocument | *None = None*)

Bases: *Serializer*

PROV-XML serializer for *ProvDocument*

deserialize(*stream: IOBase*, ***kwargs: Any*) → *ProvDocument*

Deserialize from PROV-XML representation to a *ProvDocument* instance.

Parameters
stream – Input data.

deserialize_subtree(*xml_doc: _Element*, *bundle:* ProvDocument | ProvBundle) → *ProvDocument* | *ProvBundle*

Deserialize an etree element containing a PROV document or a bundle and write it to the provided internal object.

Parameters

- **xml_doc** – An etree element containing the information to read.
- **bundle** – The bundle object to write to.

serialize(*stream: IOBase*, *force_types: bool = False*, ***kwargs: Any*) → None

Serializes a *ProvDocument* instance to PROV-XML.

Parameters

- **stream** – Where to save the output.
- **force_types** (*boolean, optional*) – Will force xsd:types to be written for most attributes mainly PROV-"attributes", e.g. tags not in the PROV namespace. Off by default meaning xsd:type attributes will only be set for prov:type, prov:location, and prov:value as is done in the official PROV-XML specification. Furthermore the types will always be set if the Python type requires it. False is a good default and it should rarely require changing.

serialize_bundle(*bundle:* ProvBundle, *element: _Element | None = None*, *force_types: bool = False*) → _Element

Serializes a bundle or document to PROV XML.

Parameters

- **bundle** – The bundle or document.
- **element** – The XML element to write to. Will be created if None.
- **force_types** (*boolean, optional*) – Will force xsd:types to be written for most attributes mainly PROV-"attributes", e.g. tags not in the PROV namespace. Off by default meaning xsd:type attributes will only be set for prov:type, prov:location, and prov:value as is done in the official PROV-XML specification. Furthermore the types will always be set if the Python type requires it. False is a good default and it should rarely require changing.

5.1.2 Submodules

5.1.3 prov.constants module

5.1.4 prov.dot module

Graphical visualisation support for prov.model.

This module produces graphical visualisation for provenanve graphs. Requires pydot module and Graphviz.

References:

- pydot homepage: https://github.com/erocarrera/pydot
- Graphviz: http://www.graphviz.org/
- DOT Language: http://www.graphviz.org/doc/info/lang.html

prov.dot.**htlm_link_if_uri**(*value: Any*) → str

prov.dot.**prov_to_dot**(*bundle:* ProvBundle, *show_nary: bool = True*, *use_labels: bool = False*, *direction: str = 'BT'*, *show_element_attributes: bool = True*, *show_relation_attributes: bool = True*) → Dot

Convert a provenance bundle/document into a DOT graphical representation.

Parameters

- **bundle** (ProvBundle) – The provenance bundle/document to be converted.
- **show_nary** (*bool*) – shows all elements in n-ary relations.
- **use_labels** (*bool*) – uses the prov:label property of an element as its name (instead of its identifier).
- **direction** – specifies the direction of the graph. Valid values are "BT" (default), "TB", "LR", "RL".
- **show_element_attributes** (*bool*) – shows attributes of elements.
- **show_relation_attributes** (*bool*) – shows attributes of relations.

Returns

pydot.Dot – the Dot object.

5.1.5 prov.graph module

prov.graph.**graph_to_prov**(*g: MultiDiGraph*) → *ProvDocument*

Convert a MultiDiGraph that was previously produced by *prov_to_graph()* back to a *ProvDocument*.

Parameters

g – The graph instance to convert.

prov.graph.**prov_to_graph**(*prov_document:* ProvDocument) → MultiDiGraph

Convert a *ProvDocument* to a MultiDiGraph instance of the NetworkX library.

Parameters
prov_document – The *ProvDocument* instance to convert.

5.1.6 prov.identifier module

class prov.identifier.**Identifier**(*uri: str*)

Bases: object

Base class for all identifiers and also represents xsd:anyURI.

provn_representation() → str

Returns the PROV-N representation of the URI.

Returns:
str: The PROV-N representation of the URI.

property uri: str

Returns the URI associated with the current identifier.

Returns:
str: The URI representing the resource identifier.

class prov.identifier.**Namespace**(*prefix: str*, *uri: str*)

Bases: object

PROV Namespace.

contains(*identifier:* Identifier) → bool

Indicates whether the identifier provided is contained in this namespace.

Parameters
identifier – Identifier to check.

Returns
bool

property prefix: str

Namespace prefix.

qname(*identifier: str* | Identifier) → *QualifiedName* | None

Returns the qualified name of the identifier given using the namespace prefix.

Parameters
identifier – Identifier to resolve to a qualified name.

Returns
QualifiedName

property uri: str

Namespace URI.

class prov.identifier.**QualifiedName**(*namespace:* Namespace, *localpart: str*)

Bases: *Identifier*

Represents a qualified name, which combines a namespace and a local part for use in identifying entities in a namespace-aware context.

This class facilitates handling and manipulation of qualified names, which combine a namespace and a local identifier. It supports string representation, hashing, and retrieval of individual components (namespace or local part).

property localpart: str

Local part of qualified name.

property namespace: *Namespace*

Namespace of qualified name.

provn_representation() → str

PROV-N representation of qualified name in a string.

5.1.7 prov.model module

Python implementation of the W3C Provenance Data Model (PROV-DM), including support for PROV-JSON import/export

References:

PROV-DM: http://www.w3.org/TR/prov-dm/ PROV-JSON: https://openprovenance.org/prov-json/

class prov.model.**Literal**(*value: Any*, *datatype:* QualifiedName | *None* = *None*, *langtag: str* | *None* = *None*)

Bases: object

property datatype: *QualifiedName* | None

has_no_langtag() → bool

property langtag: str | None

provn_representation() → str

property value: str

class prov.model.**NamespaceManager**(*namespaces: dict[str, str]* | *Iterable[*Namespace*]* | *None* = *None*, *default: str* | *None* = *None*, *parent:* NamespaceManager | *None* = *None*)

Bases: dict

Manages namespaces for PROV documents and bundles.

add_namespace(*namespace:* Namespace) → *Namespace*

Adds a namespace (if not available, yet).

Parameters
namespace – *Namespace* to add.

add_namespaces(*namespaces: dict[str, str]* | *Iterable[*Namespace*]*) → None

Add multiple namespaces into this manager.

Parameters
namespaces (List of *Namespace* or dict of {prefix: uri}.) – A collection of namespace(s) to add.

Returns
None

get_anonymous_identifier(*local_prefix: str* = *'id'*) → *Identifier*

Returns an anonymous identifier (without a namespace prefix).

Parameters
local_prefix – Optional local namespace prefix as a string (default: 'id').

Returns
Identifier

get_default_namespace() → *Namespace* | None

Returns the default namespace.

Returns
Namespace

get_namespace(*uri: str*) → *Namespace* | None

Returns the namespace prefix for the given URI.

Parameters
uri – Namespace URI.

Returns
Namespace.

get_registered_namespaces() → Iterable[*Namespace*]

Returns all registered namespaces.

Returns
Iterable of *Namespace*.

parent: *NamespaceManager* | None = None

Parent *NamespaceManager* this manager one is a child of.

set_default_namespace(*uri: str*) → None

Sets the default namespace to the one of a given URI.

Parameters
uri – Namespace URI.

valid_qualified_name(*qname:* QualifiedName | *str* | Identifier) → *QualifiedName* | None

Resolves an identifier to a valid qualified name.

Parameters
qname – Qualified name as *QualifiedName* or a tuple (namespace, identifier).

Returns
QualifiedName or None in case of failure.

class prov.model.**ProvActivity**(*bundle:* ProvBundle, *identifier:* QualifiedName | *None*, *attributes: dict[*QualifiedName | *str* | Identifier*, Any]* | *Iterable[tuple[*QualifiedName | *str* | Identifier*, Any]]* | *None = None*)

Bases: *ProvElement*

Provenance Activity element.

FORMAL_ATTRIBUTES: tuple[*QualifiedName*, ...] = (<QualifiedName: prov:startTime>, <QualifiedName: prov:endTime>)

Formal attributes names of this record type, in the expected order.

get_endTime() → datetime | None

Returns the time the activity ended.

Returns
datetime.datetime

get_startTime() → datetime | None

Returns the time the activity started.

Returns
datetime.datetime

set_time(*startTime: datetime | None = None*, *endTime: datetime | None = None*) → None

Sets the time this activity took place.

Parameters

- **startTime** – Start time for the activity. Either a datetime.datetime object or a string that can be parsed by dateutil.parser().
- **endTime** – Start time for the activity. Either a datetime.datetime object or a string that can be parsed by dateutil.parser().

used(*entity: ProvEntity | QualifiedName | str | Identifier*, *time: datetime | str | None = None*, *attributes: dict[QualifiedName | str | Identifier, Any] | Iterable[tuple[QualifiedName | str | Identifier, Any]] | None = None*) → *ProvActivity*

Creates a new usage record for this activity.

Parameters

- **entity** – Entity or string identifier of the entity involved in the usage relationship (default: None).
- **time** – Optional time for the usage (default: None). Either a datetime.datetime object or a string that can be parsed by dateutil.parser().
- **attributes** – Optional other attributes as a dictionary or list of tuples to be added to the record optionally (default: None).

wasAssociatedWith(*agent: ProvAgent | ProvEntity | ProvActivity | QualifiedName | str | Identifier*, *plan: ProvEntity | QualifiedName | str | Identifier | None = None*, *attributes: dict[QualifiedName | str | Identifier, Any] | Iterable[tuple[QualifiedName | str | Identifier, Any]] | None = None*) → *ProvActivity*

Creates a new association record for this activity.

Parameters

- **agent** – Agent or string identifier of the agent involved in the association (default: None).
- **plan** – Optionally extra entity to state qualified association through an internal plan (default: None).
- **attributes** – Optional other attributes as a dictionary or list of tuples to be added to the record optionally (default: None).

wasEndedBy(*trigger: ProvEntity | QualifiedName | str | Identifier | None*, *ender: ProvActivity | QualifiedName | str | Identifier | None = None*, *time: datetime | str | None = None*, *attributes: dict[QualifiedName | str | Identifier, Any] | Iterable[tuple[QualifiedName | str | Identifier, Any]] | None = None*) → *ProvActivity*

Creates a new end record for this activity.

Parameters

- **trigger** – Entity triggering the end of this activity.
- **ender** – Optionally extra activity to state a qualified end through which the trigger entity for the end is generated (default: None).

- **time** – Optional time for the end (default: None). Either a `datetime.datetime` object or a string that can be parsed by `dateutil.parser()`.
- **attributes** – Optional other attributes as a dictionary or list of tuples to be added to the record optionally (default: None).

wasInformedBy(*informant:* ProvActivity | QualifiedName | *str* | Identifier, *attributes: dict[*QualifiedName | *str* | Identifier*, Any]* | *Iterable[tuple[*QualifiedName | *str* | Identifier*, Any]]* | *None* = *None*) → *ProvActivity*

Creates a new communication record for this activity.

Parameters

- **informant** – The informing activity (relationship source).
- **attributes** – Optional other attributes as a dictionary or list of tuples to be added to the record optionally (default: None).

wasStartedBy(*trigger:* ProvEntity | QualifiedName | *str* | Identifier | *None*, *starter:* ProvActivity | QualifiedName | *str* | Identifier | *None* = *None*, *time: datetime* | *str* | *None* = *None*, *attributes: dict[*QualifiedName | *str* | Identifier*, Any]* | *Iterable[tuple[*QualifiedName | *str* | Identifier*, Any]]* | *None* = *None*) → *ProvActivity*

Creates a new start record for this activity. The activity did not exist before the start by the trigger.

Parameters

- **trigger** – Entity triggering the start of this activity.
- **starter** – Optional extra activity to state a qualified start through which the trigger entity for the start is generated (default: None).
- **time** – Optional time for the start (default: None). Either a `datetime.datetime` object or a string that can be parsed by `dateutil.parser()`.
- **attributes** – Optional other attributes as a dictionary or list of tuples to be added to the record optionally (default: None).

class prov.model.**ProvAgent**(*bundle:* ProvBundle, *identifier:* QualifiedName | *None*, *attributes: dict[*QualifiedName | *str* | Identifier*, Any]* | *Iterable[tuple[*QualifiedName | *str* | Identifier*, Any]]* | *None* = *None*)

Bases: *ProvElement*

Provenance Agent element.

actedOnBehalfOf(*responsible:* ProvAgent | ProvEntity | ProvActivity | QualifiedName | *str* | Identifier, *activity:* ProvActivity | QualifiedName | *str* | Identifier | *None* = *None*, *attributes: dict[*QualifiedName | *str* | Identifier*, Any]* | *Iterable[tuple[*QualifiedName | *str* | Identifier*, Any]]* | *None* = *None*) → *ProvAgent*

Creates a new delegation record on behalf of this agent.

Parameters

- **responsible** – Agent the responsibility is delegated to.
- **activity** – Optionally extra activity to state qualified delegation internally (default: None).
- **attributes** – Optional other attributes as a dictionary or list of tuples to be added to the record optionally (default: None).

class prov.model.**ProvAlternate**(*bundle:* ProvBundle, *identifier:* QualifiedName | *None*, *attributes:* *dict[*QualifiedName | *str* | Identifier*, Any]* | *Iterable[tuple[*QualifiedName | *str* | Identifier*, Any]]* | *None* = *None*)

Bases: *ProvRelation*

Provenance Alternate relationship.

FORMAL_ATTRIBUTES: tuple[*QualifiedName*, ...] = (<QualifiedName: prov:alternate1>, <QualifiedName: prov:alternate2>)

Formal attributes names of this record type, in the expected order.

class prov.model.**ProvAssociation**(*bundle:* ProvBundle, *identifier:* QualifiedName | *None*, *attributes:* *dict[*QualifiedName | *str* | Identifier*, Any]* | *Iterable[tuple[*QualifiedName | *str* | Identifier*, Any]]* | *None* = *None*)

Bases: *ProvRelation*

Provenance Association relationship.

FORMAL_ATTRIBUTES: tuple[*QualifiedName*, ...] = (<QualifiedName: prov:activity>, <QualifiedName: prov:agent>, <QualifiedName: prov:plan>)

Formal attributes names of this record type, in the expected order.

class prov.model.**ProvAttribution**(*bundle:* ProvBundle, *identifier:* QualifiedName | *None*, *attributes:* *dict[*QualifiedName | *str* | Identifier*, Any]* | *Iterable[tuple[*QualifiedName | *str* | Identifier*, Any]]* | *None* = *None*)

Bases: *ProvRelation*

Provenance Attribution relationship.

FORMAL_ATTRIBUTES: tuple[*QualifiedName*, ...] = (<QualifiedName: prov:entity>, <QualifiedName: prov:agent>)

Formal attributes names of this record type, in the expected order.

class prov.model.**ProvBundle**(*records: Iterable[*ProvRecord*]* | *None* = *None*, *identifier:* QualifiedName | *None* = *None*, *namespaces: dict[str, str]* | *Iterable[*Namespace*]* | *None* = *None*, *document:* ProvDocument | *None* = *None*)

Bases: `object`

PROV Bundle

actedOnBehalfOf(*delegate:* ProvAgent | ProvEntity | ProvActivity | QualifiedName | *str* | Identifier, *responsible:* ProvAgent | ProvEntity | ProvActivity | QualifiedName | *str* | Identifier, *activity:* ProvActivity | QualifiedName | *str* | Identifier | *None* = *None*, *identifier:* QualifiedName | *str* | Identifier | *None* = *None*, *other_attributes: dict[*QualifiedName | *str* | Identifier*, Any]* | *Iterable[tuple[*QualifiedName | *str* | Identifier*, Any]]* | *None* = *None*) → *ProvDelegation*

Creates a new delegation record on behalf of an agent.

Parameters

- **delegate** – Agent delegating the responsibility (relationship source).
- **responsible** – Agent the responsibility is delegated to (relationship destination).
- **activity** – Optionally extra activity to state qualified delegation internally (default: None).
- **identifier** – Identifier for new association record.

- **other_attributes** – Optional other attributes as a dictionary or list of tuples to be added to the record optionally (default: None).

activity(*identifier:* QualifiedName | *str* | Identifier, *startTime: datetime* | *str* | *None* = *None*, *endTime: datetime* | *str* | *None* = *None*, *other_attributes: dict[*QualifiedName | *str* | Identifier*, Any]* | *Iterable[tuple[*QualifiedName | *str* | Identifier*, Any]]* | *None* = *None*) → *ProvActivity*

Creates a new activity.

Parameters

- **identifier** – Identifier for new activity.
- **startTime** – Optional start time for the activity (default: None). Either a `datetime.datetime` object or a string that can be parsed by `dateutil.parser()`.
- **endTime** – Optional start time for the activity (default: None). Either a `datetime.datetime` object or a string that can be parsed by `dateutil.parser()`.
- **other_attributes** – Optional other attributes as a dictionary or list of tuples to be added to the record optionally (default: None).

add_namespace(*namespace_or_prefix:* Namespace | *str*, *uri: str* | *None* = *None*) → *Namespace*

Adds a namespace (if not available, yet).

Parameters

- **namespace_or_prefix** – *Namespace* or its prefix as a string to add.
- **uri** – Namespace URI (default: None). Must be present if only a prefix is given in the previous parameter.

add_record(*record:* ProvRecord) → *ProvRecord*

Adds a new record that to the bundle.

Parameters
record – *ProvRecord* to be added.

agent(*identifier:* QualifiedName | *str* | Identifier, *other_attributes: dict[*QualifiedName | *str* | Identifier*, Any]* | *Iterable[tuple[*QualifiedName | *str* | Identifier*, Any]]* | *None* = *None*) → *ProvAgent*

Creates a new agent.

Parameters

- **identifier** – Identifier for new agent.
- **other_attributes** – Optional other attributes as a dictionary or list of tuples to be added to the record optionally (default: None).

alternate(*alternate1:* ProvEntity | QualifiedName | *str* | Identifier, *alternate2:* ProvEntity | QualifiedName | *str* | Identifier) → *ProvAlternate*

Creates a new alternate record between two entities.

Parameters

- **alternate1** – Entity or a string identifier for the first entity (relationship source).
- **alternate2** – Entity or a string identifier for the second entity (relationship destination).

alternateOf(*alternate1:* ProvEntity | QualifiedName | *str* | Identifier, *alternate2:* ProvEntity | QualifiedName | *str* | Identifier) → *ProvAlternate*

Creates a new alternate record between two entities.

Parameters

- **alternate1** – Entity or a string identifier for the first entity (relationship source).
- **alternate2** – Entity or a string identifier for the second entity (relationship destination).

association(*activity:* ProvActivity | QualifiedName | *str* | Identifier, *agent:* ProvAgent | ProvEntity | ProvActivity | QualifiedName | *str* | Identifier | *None* = *None*, *plan:* ProvEntity | QualifiedName | *str* | Identifier | *None* = *None*, *identifier:* QualifiedName | *str* | Identifier | *None* = *None*, *other_attributes: dict[*QualifiedName | *str* | Identifier*, Any]* | *Iterable[tuple[*QualifiedName | *str* | Identifier*, Any]]* | *None* = *None*) → *ProvAssociation*

Creates a new association record for an activity.

Parameters

- **activity** – Activity or a string identifier for the activity.
- **agent** – Agent or string identifier of the agent involved in the association (default: None).
- **plan** – Optionally extra entity to state qualified association through an internal plan (default: None).
- **identifier** – Identifier for new association record.
- **other_attributes** – Optional other attributes as a dictionary or list of tuples to be added to the record optionally (default: None).

attribution(*entity:* ProvEntity | QualifiedName | *str* | Identifier, *agent:* ProvAgent | ProvEntity | ProvActivity | QualifiedName | *str* | Identifier, *identifier:* QualifiedName | *str* | Identifier | *None* = *None*, *other_attributes: dict[*QualifiedName | *str* | Identifier*, Any]* | *Iterable[tuple[*QualifiedName | *str* | Identifier*, Any]]* | *None* = *None*) → *ProvAttribution*

Creates a new attribution record between an entity and an agent.

Parameters

- **entity** – Entity or a string identifier for the entity (relationship source).
- **agent** – Agent or string identifier of the agent involved in the attribution (relationship destination).
- **identifier** – Identifier for new attribution record.
- **other_attributes** – Optional other attributes as a dictionary or list of tuples to be added to the record optionally (default: None).

property bundles: Iterable[*ProvBundle*]

Returns bundles contained in the document

Returns
Iterable of *ProvBundle*.

collection(*identifier:* QualifiedName | *str* | Identifier, *other_attributes: dict[*QualifiedName | *str* | Identifier*, Any]* | *Iterable[tuple[*QualifiedName | *str* | Identifier*, Any]]* | *None* = *None*) → *ProvEntity*

Creates a new collection record for a particular record.

Parameters

- **identifier** – Identifier for new collection record.
- **other_attributes** – Optional other attributes as a dictionary or list of tuples to be added to the record optionally (default: None).

communication(*informed:* ProvActivity | QualifiedName | *str* | Identifier, *informant:* ProvActivity | QualifiedName | *str* | Identifier, *identifier:* QualifiedName | *str* | Identifier | *None* = *None*, *other_attributes: dict[*QualifiedName | *str* | Identifier*, Any]* | *Iterable[tuple[*QualifiedName | *str* | Identifier*, Any]]* | *None* = *None*) → *ProvCommunication*

Creates a new communication record for an entity.

Parameters

- **informed** – The informed activity (relationship destination).
- **informant** – The informing activity (relationship source).
- **identifier** – Identifier for new communication record.
- **other_attributes** – Optional other attributes as a dictionary or list of tuples to be added to the record optionally (default: None).

property default_ns_uri: str | None

Returns the default namespace's URI, if any.

Returns
URI as string.

delegation(*delegate:* ProvAgent | ProvEntity | ProvActivity | QualifiedName | *str* | Identifier, *responsible:* ProvAgent | ProvEntity | ProvActivity | QualifiedName | *str* | Identifier, *activity:* ProvActivity | QualifiedName | *str* | Identifier | *None* = *None*, *identifier:* QualifiedName | *str* | Identifier | *None* = *None*, *other_attributes: dict[*QualifiedName | *str* | Identifier*, Any]* | *Iterable[tuple[*QualifiedName | *str* | Identifier*, Any]]* | *None* = *None*) → *ProvDelegation*

Creates a new delegation record on behalf of an agent.

Parameters

- **delegate** – Agent delegating the responsibility (relationship source).
- **responsible** – Agent the responsibility is delegated to (relationship destination).
- **activity** – Optionally extra activity to state qualified delegation internally (default: None).
- **identifier** – Identifier for new association record.
- **other_attributes** – Optional other attributes as a dictionary or list of tuples to be added to the record optionally (default: None).

derivation(*generatedEntity:* ProvEntity | QualifiedName | *str* | Identifier, *usedEntity:* ProvEntity | QualifiedName | *str* | Identifier, *activity:* ProvActivity | QualifiedName | *str* | Identifier | *None* = *None*, *generation:* ProvGeneration | QualifiedName | *str* | Identifier | *None* = *None*, *usage:* ProvUsage | QualifiedName | *str* | Identifier | *None* = *None*, *identifier:* QualifiedName | *str* | Identifier | *None* = *None*, *other_attributes: dict[*QualifiedName | *str* | Identifier*, Any]* | *Iterable[tuple[*QualifiedName | *str* | Identifier*, Any]]* | *None* = *None*) → *ProvDerivation*

Creates a new derivation record for a generated entity from a used entity.

Parameters

- **generatedEntity** – Entity or a string identifier for the generated entity (relationship source).
- **usedEntity** – Entity or a string identifier for the used entity (relationship destination).
- **activity** – Activity or string identifier of the activity involved in the derivation (default: None).
- **generation** – Optionally extra activity to state qualified generation through a generation (default: None).
- **usage** – XXX (default: None).

- **identifier** – Identifier for new derivation record.
- **other_attributes** – Optional other attributes as a dictionary or list of tuples to be added to the record optionally (default: None).

property document: *ProvDocument* **| None**

Returns the parent document, if any.

Returns
ProvDocument.

end(*activity:* ProvActivity | QualifiedName | *str* | Identifier, *trigger:* ProvEntity | QualifiedName | *str* | Identifier | *None* = *None*, *ender:* ProvActivity | QualifiedName | *str* | Identifier | *None* = *None*, *time:* *datetime* | *str* | *None* = *None*, *identifier:* QualifiedName | *str* | Identifier | *None* = *None*, *other_attributes:* *dict[*QualifiedName | *str* | Identifier*, Any]* | *Iterable[tuple[*QualifiedName | *str* | Identifier*, Any]]* | *None* = *None*) → *ProvEnd*

Creates a new end record for an activity.

Parameters

- **activity** – Activity or a string identifier for the entity.
- **trigger** – trigger: Entity triggering the end of this activity.
- **ender** – Optionally extra activity to state a qualified end through which the trigger entity for the end is generated (default: None).
- **time** – Optional time for the end (default: None). Either a `datetime.datetime` object or a string that can be parsed by `dateutil.parser()`.
- **identifier** – Identifier for new end record.
- **other_attributes** – Optional other attributes as a dictionary or list of tuples to be added to the record optionally (default: None).

entity(*identifier:* QualifiedName | *str* | Identifier, *other_attributes:* *dict[*QualifiedName | *str* | Identifier*, Any]* | *Iterable[tuple[*QualifiedName | *str* | Identifier*, Any]]* | *None* = *None*) → *ProvEntity*

Creates a new entity.

Parameters

- **identifier** – Identifier for new entity.
- **other_attributes** – Optional other attributes as a dictionary or list of tuples to be added to the record optionally (default: None).

generation(*entity:* ProvEntity | QualifiedName | *str* | Identifier, *activity:* ProvActivity | QualifiedName | *str* | Identifier | *None* = *None*, *time:* *datetime* | *str* | *None* = *None*, *identifier:* QualifiedName | *str* | Identifier | *None* = *None*, *other_attributes:* *dict[*QualifiedName | *str* | Identifier*, Any]* | *Iterable[tuple[*QualifiedName | *str* | Identifier*, Any]]* | *None* = *None*) → *ProvRecord*

Creates a new generation record for an entity.

Parameters

- **entity** – Entity or a string identifier for the entity.
- **activity** – Activity or string identifier of the activity involved in the generation (default: None).
- **time** – Optional time for the generation (default: None). Either a `datetime.datetime` object or a string that can be parsed by `dateutil.parser()`.
- **identifier** – Identifier for new generation record.

- **other_attributes** – Optional other attributes as a dictionary or list of tuples to be added to the record optionally (default: None).

get_default_namespace() → *Namespace* | None

Returns the default namespace.

Returns
Namespace

get_provn(*_indent_level: int = 0*) → str

Returns the PROV-N representation of the bundle.

Returns
String

get_record(*identifier:* QualifiedName | *str* | Identifier) → list[*ProvRecord*]

Returns one or more records matching a given identifier.

Parameters
identifier – Record identifier.

Returns
List of *ProvRecord*

get_records(*class_or_type_or_tuple: type | tuple[type] | None = None*) → Iterable[*ProvRecord*]

Returns all records. Returned records may be filtered by the optional argument.

Parameters
class_or_type_or_tuple – A filter on the type for which records are to be returned (default: None). The filter checks by the type of the record using the *isinstance* check on the record.

Returns
List of *ProvRecord* objects.

get_registered_namespaces() → Iterable[*Namespace*]

Returns all registered namespaces.

Returns
Iterable of *Namespace*.

hadMember(*collection:* ProvEntity | QualifiedName | *str* | Identifier, *entity:* ProvEntity | QualifiedName | *str* | Identifier) → *ProvMembership*

Creates a new membership record for an entity to a collection.

Parameters

- **collection** – Collection the entity is to be added to.
- **entity** – Entity to be added to the collection.

hadPrimarySource(*generatedEntity:* ProvEntity | QualifiedName | *str* | Identifier, *usedEntity:* ProvEntity | QualifiedName | *str* | Identifier, *activity:* ProvActivity | QualifiedName | *str* | Identifier | *None = None*, *generation:* ProvGeneration | QualifiedName | *str* | Identifier | *None = None*, *usage:* ProvUsage | QualifiedName | *str* | Identifier | *None = None*, *identifier:* QualifiedName | *str* | Identifier | *None = None*, *other_attributes: dict[*QualifiedName | *str* | Identifier*, Any] | Iterable[tuple[*QualifiedName | *str* | Identifier*, Any]] | None = None*) → *ProvDerivation*

Creates a new primary source record for a generated entity from a used entity.

Parameters

- **generatedEntity** – Entity or a string identifier for the generated entity (relationship source).
- **usedEntity** – Entity or a string identifier for the used entity (relationship destination).
- **activity** – Activity or string identifier of the activity involved in the primary source (default: None).
- **generation** – Optionally to state qualified primary source through a generation activity (default: None).
- **usage** – XXX (default: None).
- **identifier** – Identifier for new primary source record.
- **other_attributes** – Optional other attributes as a dictionary or list of tuples to be added to the record optionally (default: None).

has_bundles() → bool

True if the object has at least one bundle, *False* otherwise.

Returns
bool

property identifier: *QualifiedName* | None

Returns the bundle's identifier

influence(*influencee:* ProvEntity | QualifiedName | *str* | Identifier | ProvActivity | ProvAgent, *influencer:* ProvEntity | QualifiedName | *str* | Identifier | ProvActivity | ProvAgent, *identifier:* QualifiedName | *str* | Identifier | *None* = *None*, *other_attributes: dict[*QualifiedName | *str* | Identifier*, Any]* | *Iterable[tuple[*QualifiedName | *str* | Identifier*, Any]]* | *None* = *None*) → *ProvInfluence*

Creates a new influence record between two entities, activities or agents.

Parameters

- **influencee** – Influenced entity, activity or agent (relationship source).
- **influencer** – Influencing entity, activity or agent (relationship destination).
- **identifier** – Identifier for new influence record.
- **other_attributes** – Optional other attributes as a dictionary or list of tuples to be added to the record optionally (default: None).

invalidation(*entity:* ProvEntity | QualifiedName | *str* | Identifier, *activity:* ProvActivity | QualifiedName | *str* | Identifier | *None* = *None*, *time: datetime* | *str* | *None* = *None*, *identifier:* QualifiedName | *str* | Identifier | *None* = *None*, *other_attributes: dict[*QualifiedName | *str* | Identifier*, Any]* | *Iterable[tuple[*QualifiedName | *str* | Identifier*, Any]]* | *None* = *None*) → *ProvInvalidation*

Creates a new invalidation record for an entity.

Parameters

- **entity** – Entity or a string identifier for the entity.
- **activity** – Activity or string identifier of the activity involved in the invalidation (default: None).
- **time** – Optional time for the invalidation (default: None). Either a `datetime.datetime` object or a string that can be parsed by `dateutil.parser()`.
- **identifier** – Identifier for the new invalidation record.
- **other_attributes** – Optional other attributes as a dictionary or list of tuples to be added to the record optionally (default: None).

is_bundle() → bool

True if the object is a bundle, *False* otherwise.

Returns
bool

is_document() → bool

True if the object is a document, *False* otherwise.

Returns
bool

mandatory_valid_qname(*identifier:* QualifiedName | *str* | Identifier) → *QualifiedName*

Determines if the given identifier is a valid qualified name and returns it. If the provided identifier is not valid, an exception is raised.

membership(*collection:* ProvEntity | QualifiedName | *str* | Identifier, *entity:* ProvEntity | QualifiedName | *str* | Identifier) → *ProvMembership*

Creates a new membership record for an entity to a collection.

Parameters

- **collection** – Collection the entity is to be added to.
- **entity** – Entity to be added to the collection.

mention(*specificEntity:* ProvEntity | QualifiedName | *str* | Identifier, *generalEntity:* ProvEntity | QualifiedName | *str* | Identifier, *bundle:* ProvEntity | QualifiedName | *str* | Identifier) → *ProvMention*

Creates a new mention record for a specific from a general entity.

Parameters

- **specificEntity** – Entity or a string identifier for the specific entity (relationship source).
- **generalEntity** – Entity or a string identifier for the general entity (relationship destination).
- **bundle** – XXX

mentionOf(*specificEntity:* ProvEntity | QualifiedName | *str* | Identifier, *generalEntity:* ProvEntity | QualifiedName | *str* | Identifier, *bundle:* ProvEntity | QualifiedName | *str* | Identifier) → *ProvMention*

Creates a new mention record for a specific from a general entity.

Parameters

- **specificEntity** – Entity or a string identifier for the specific entity (relationship source).
- **generalEntity** – Entity or a string identifier for the general entity (relationship destination).
- **bundle** – XXX

property namespaces: set[*Namespace*]

Returns the set of registered namespaces.

Returns
Set of *Namespace*.

new_record(*record_type:* QualifiedName, *identifier:* QualifiedName | *str* | Identifier | *None*, *attributes:* *dict[*QualifiedName | *str* | Identifier*, Any]* | *Iterable[tuple[*QualifiedName | *str* | Identifier*, Any]]* | *None* = *None*, *other_attributes: dict[*QualifiedName | *str* | Identifier*, Any]* | *Iterable[tuple[*QualifiedName | *str* | Identifier*, Any]]* | *None* = *None*) → *ProvRecord*

Creates a new record.

Parameters

- **record_type** – Type of record (one of PROV_REC_CLS).
- **identifier** – Identifier for new record.
- **attributes** – Attributes as a dictionary or list of tuples to be added to the record optionally (default: None).
- **other_attributes** – Optional other attributes as a dictionary or list of tuples to be added to the record optionally (default: None).

plot(*filename: str* | *bytes* | *PathLike* | *None* = *None*, *show_nary: bool* = *True*, *use_labels: bool* = *False*, *show_element_attributes: bool* = *True*, *show_relation_attributes: bool* = *True*) → None

Convenience function to plot a PROV document.

Parameters

- **filename** (*String*) – The filename to save to. If not given, it will open an interactive matplotlib plot. The filetype is determined from the filename ending.
- **show_nary** (*bool*) – Shows all elements in n-ary relations.
- **use_labels** (*bool*) – Uses the *prov:label* property of an element as its name (instead of its identifier).
- **show_element_attributes** (*bool*) – Shows attributes of elements.
- **show_relation_attributes** (*bool*) – Shows attributes of relations.

primary_source(*generatedEntity:* ProvEntity | QualifiedName | *str* | Identifier, *usedEntity:* ProvEntity | QualifiedName | *str* | Identifier, *activity:* ProvActivity | QualifiedName | *str* | Identifier | *None* = *None*, *generation:* ProvGeneration | QualifiedName | *str* | Identifier | *None* = *None*, *usage:* ProvUsage | QualifiedName | *str* | Identifier | *None* = *None*, *identifier:* QualifiedName | *str* | Identifier | *None* = *None*, *other_attributes: dict[*QualifiedName | *str* | Identifier*, Any]* | *Iterable[tuple[*QualifiedName | *str* | Identifier*, Any]]* | *None* = *None*) → *ProvDerivation*

Creates a new primary source record for a generated entity from a used entity.

Parameters

- **generatedEntity** – Entity or a string identifier for the generated entity (relationship source).
- **usedEntity** – Entity or a string identifier for the used entity (relationship destination).
- **activity** – Activity or string identifier of the activity involved in the primary source (default: None).
- **generation** – Optionally to state qualified primary source through a generation activity (default: None).
- **usage** – XXX (default: None).
- **identifier** – Identifier for new primary source record.
- **other_attributes** – Optional other attributes as a dictionary or list of tuples to be added to the record optionally (default: None).

quotation(*generatedEntity:* ProvEntity | QualifiedName | *str* | Identifier, *usedEntity:* ProvEntity | QualifiedName | *str* | Identifier, *activity:* ProvActivity | QualifiedName | *str* | Identifier | *None* = *None*, *generation:* ProvGeneration | QualifiedName | *str* | Identifier | *None* = *None*, *usage:* ProvUsage | QualifiedName | *str* | Identifier | *None* = *None*, *identifier:* QualifiedName | *str* | Identifier | *None* = *None*, *other_attributes: dict[*QualifiedName | *str* | Identifier*, Any]* | *Iterable[tuple[*QualifiedName | *str* | Identifier*, Any]]* | *None* = *None*) → *ProvDerivation*

Creates a new quotation record for a generated entity from a used entity.

Parameters

- **generatedEntity** – Entity or a string identifier for the generated entity (relationship source).
- **usedEntity** – Entity or a string identifier for the used entity (relationship destination).
- **activity** – Activity or string identifier of the activity involved in the quotation (default: None).
- **generation** – Optionally to state qualified quotation through a generation activity (default: None).
- **usage** – XXX (default: None).
- **identifier** – Identifier for new quotation record.
- **other_attributes** – Optional other attributes as a dictionary or list of tuples to be added to the record optionally (default: None).

property records: list[*ProvRecord*]

Returns the list of all records in the current bundle

revision(*generatedEntity:* ProvEntity | QualifiedName | *str* | Identifier, *usedEntity:* ProvEntity | QualifiedName | *str* | Identifier, *activity:* ProvActivity | QualifiedName | *str* | Identifier | *None* = *None*, *generation:* ProvGeneration | QualifiedName | *str* | Identifier | *None* = *None*, *usage:* ProvUsage | QualifiedName | *str* | Identifier | *None* = *None*, *identifier:* QualifiedName | *str* | Identifier | *None* = *None*, *other_attributes: dict[*QualifiedName | *str* | Identifier*, Any]* | *Iterable[tuple[*QualifiedName | *str* | Identifier*, Any]]* | *None* = *None*) → *ProvDerivation*

Creates a new revision record for a generated entity from a used entity.

Parameters

- **generatedEntity** – Entity or a string identifier for the generated entity (relationship source).
- **usedEntity** – Entity or a string identifier for the used entity (relationship destination).
- **activity** – Activity or string identifier of the activity involved in the revision (default: None).
- **generation** – Optionally to state qualified revision through a generation activity (default: None).
- **usage** – XXX (default: None).
- **identifier** – Identifier for new revision record.
- **other_attributes** – Optional other attributes as a dictionary or list of tuples to be added to the record optionally (default: None).

set_default_namespace(*uri: str*) → None

Sets the default namespace through a given URI.

Parameters
uri – Namespace URI.

specialization(*specificEntity:* ProvEntity | QualifiedName | *str* | Identifier, *generalEntity:* ProvEntity | QualifiedName | *str* | Identifier) → *ProvSpecialization*

Creates a new specialisation record for a specific from a general entity.

Parameters

- **specificEntity** – Entity or a string identifier for the specific entity (relationship source).
- **generalEntity** – Entity or a string identifier for the general entity (relationship destination).

specializationOf(*specificEntity:* ProvEntity | QualifiedName | *str* | Identifier, *generalEntity:* ProvEntity | QualifiedName | *str* | Identifier) → *ProvSpecialization*

Creates a new specialisation record for a specific from a general entity.

Parameters

- **specificEntity** – Entity or a string identifier for the specific entity (relationship source).
- **generalEntity** – Entity or a string identifier for the general entity (relationship destination).

start(*activity:* ProvActivity | QualifiedName | *str* | Identifier, *trigger:* ProvEntity | QualifiedName | *str* | Identifier | *None = None*, *starter:* ProvActivity | QualifiedName | *str* | Identifier | *None = None*, *time:* *datetime* | *str* | *None = None*, *identifier:* QualifiedName | *str* | Identifier | *None = None*, *other_attributes: dict[*QualifiedName | *str* | Identifier*, Any]* | *Iterable[tuple[*QualifiedName | *str* | Identifier*, Any]]* | *None = None*) → *ProvStart*

Creates a new start record for an activity.

Parameters

- **activity** – Activity or a string identifier for the entity.
- **trigger** – Entity triggering the start of this activity.
- **starter** – Optionally extra activity to state a qualified start through which the trigger entity for the start is generated (default: None).
- **time** – Optional time for the start (default: None). Either a `datetime.datetime` object or a string that can be parsed by `dateutil.parser()`.
- **identifier** – Identifier for new start record.
- **other_attributes** – Optional other attributes as a dictionary or list of tuples to be added to the record optionally (default: None).

unified() → *ProvBundle*

Unifies all records in the bundle that haves same identifiers

Returns
ProvBundle – the new unified bundle.

update(*other:* ProvBundle) → None

Append all the records of the *other* ProvBundle into this bundle.

Parameters
other (*ProvBundle*) – the other bundle whose records to be appended.

Returns
None.

usage(*activity:* ProvActivity | QualifiedName | *str* | Identifier, *entity:* ProvEntity | QualifiedName | *str* | Identifier | *None* = *None*, *time: datetime* | *str* | *None* = *None*, *identifier:* QualifiedName | *str* | Identifier | *None* = *None*, *other_attributes: dict[*QualifiedName | *str* | Identifier*, Any]* | *Iterable[tuple[*QualifiedName | *str* | Identifier*, Any]]* | *None* = *None*) → *ProvUsage*

Creates a new usage record for an activity.

Parameters

- **activity** – Activity or a string identifier for the entity.
- **entity** – Entity or string identifier of the entity involved in the usage relationship (default: None).
- **time** – Optional time for the usage (default: None). Either a `datetime.datetime` object or a string that can be parsed by `dateutil.parser()`.
- **identifier** – Identifier for new usage record.
- **other_attributes** – Optional other attributes as a dictionary or list of tuples to be added to the record optionally (default: None).

used(*activity:* ProvActivity | QualifiedName | *str* | Identifier, *entity:* ProvEntity | QualifiedName | *str* | Identifier | *None* = *None*, *time: datetime* | *str* | *None* = *None*, *identifier:* QualifiedName | *str* | Identifier | *None* = *None*, *other_attributes: dict[*QualifiedName | *str* | Identifier*, Any]* | *Iterable[tuple[*QualifiedName | *str* | Identifier*, Any]]* | *None* = *None*) → *ProvUsage*

Creates a new usage record for an activity.

Parameters

- **activity** – Activity or a string identifier for the entity.
- **entity** – Entity or string identifier of the entity involved in the usage relationship (default: None).
- **time** – Optional time for the usage (default: None). Either a `datetime.datetime` object or a string that can be parsed by `dateutil.parser()`.
- **identifier** – Identifier for new usage record.
- **other_attributes** – Optional other attributes as a dictionary or list of tuples to be added to the record optionally (default: None).

valid_qualified_name(*identifier:* QualifiedName | *str* | Identifier) → *QualifiedName* | None

wasAssociatedWith(*activity:* ProvActivity | QualifiedName | *str* | Identifier, *agent:* ProvAgent | ProvEntity | ProvActivity | QualifiedName | *str* | Identifier | *None* = *None*, *plan:* ProvEntity | QualifiedName | *str* | Identifier | *None* = *None*, *identifier:* QualifiedName | *str* | Identifier | *None* = *None*, *other_attributes: dict[*QualifiedName | *str* | Identifier*, Any]* | *Iterable[tuple[*QualifiedName | *str* | Identifier*, Any]]* | *None* = *None*) → *ProvAssociation*

Creates a new association record for an activity.

Parameters

- **activity** – Activity or a string identifier for the activity.
- **agent** – Agent or string identifier of the agent involved in the association (default: None).
- **plan** – Optionally extra entity to state qualified association through an internal plan (default: None).
- **identifier** – Identifier for new association record.

- **other_attributes** – Optional other attributes as a dictionary or list of tuples to be added to the record optionally (default: None).

wasAttributedTo(*entity:* ProvEntity | QualifiedName | *str* | Identifier, *agent:* ProvAgent | ProvEntity | ProvActivity | QualifiedName | *str* | Identifier, *identifier:* QualifiedName | *str* | Identifier | *None* = *None*, *other_attributes: dict[*QualifiedName | *str* | Identifier*, Any]* | *Iterable[tuple[*QualifiedName | *str* | Identifier*, Any]]* | *None* = *None*) → *ProvAttribution*

Creates a new attribution record between an entity and an agent.

Parameters

- **entity** – Entity or a string identifier for the entity (relationship source).
- **agent** – Agent or string identifier of the agent involved in the attribution (relationship destination).
- **identifier** – Identifier for new attribution record.
- **other_attributes** – Optional other attributes as a dictionary or list of tuples to be added to the record optionally (default: None).

wasDerivedFrom(*generatedEntity:* ProvEntity | QualifiedName | *str* | Identifier, *usedEntity:* ProvEntity | QualifiedName | *str* | Identifier, *activity:* ProvActivity | QualifiedName | *str* | Identifier | *None* = *None*, *generation:* ProvGeneration | QualifiedName | *str* | Identifier | *None* = *None*, *usage:* ProvUsage | QualifiedName | *str* | Identifier | *None* = *None*, *identifier:* QualifiedName | *str* | Identifier | *None* = *None*, *other_attributes: dict[*QualifiedName | *str* | Identifier*, Any]* | *Iterable[tuple[*QualifiedName | *str* | Identifier*, Any]]* | *None* = *None*) → *ProvDerivation*

Creates a new derivation record for a generated entity from a used entity.

Parameters

- **generatedEntity** – Entity or a string identifier for the generated entity (relationship source).
- **usedEntity** – Entity or a string identifier for the used entity (relationship destination).
- **activity** – Activity or string identifier of the activity involved in the derivation (default: None).
- **generation** – Optionally extra activity to state qualified generation through a generation (default: None).
- **usage** – XXX (default: None).
- **identifier** – Identifier for new derivation record.
- **other_attributes** – Optional other attributes as a dictionary or list of tuples to be added to the record optionally (default: None).

wasEndedBy(*activity:* ProvActivity | QualifiedName | *str* | Identifier, *trigger:* ProvEntity | QualifiedName | *str* | Identifier | *None* = *None*, *ender:* ProvActivity | QualifiedName | *str* | Identifier | *None* = *None*, *time: datetime* | *str* | *None* = *None*, *identifier:* QualifiedName | *str* | Identifier | *None* = *None*, *other_attributes: dict[*QualifiedName | *str* | Identifier*, Any]* | *Iterable[tuple[*QualifiedName | *str* | Identifier*, Any]]* | *None* = *None*) → *ProvEnd*

Creates a new end record for an activity.

Parameters

- **activity** – Activity or a string identifier for the entity.
- **trigger** – trigger: Entity triggering the end of this activity.

- **ender** – Optionally extra activity to state a qualified end through which the trigger entity for the end is generated (default: None).
- **time** – Optional time for the end (default: None). Either a `datetime.datetime` object or a string that can be parsed by `dateutil.parser()`.
- **identifier** – Identifier for new end record.
- **other_attributes** – Optional other attributes as a dictionary or list of tuples to be added to the record optionally (default: None).

wasGeneratedBy(*entity:* ProvEntity | QualifiedName | *str* | Identifier, *activity:* ProvActivity | QualifiedName | *str* | Identifier | *None* = *None*, *time: datetime* | *str* | *None* = *None*, *identifier:* QualifiedName | *str* | Identifier | *None* = *None*, *other_attributes: dict[*QualifiedName | *str* | Identifier*, Any]* | *Iterable[tuple[*QualifiedName | *str* | Identifier*, Any]]* | *None* = *None*) → *ProvRecord*

Creates a new generation record for an entity.

Parameters

- **entity** – Entity or a string identifier for the entity.
- **activity** – Activity or string identifier of the activity involved in the generation (default: None).
- **time** – Optional time for the generation (default: None). Either a `datetime.datetime` object or a string that can be parsed by `dateutil.parser()`.
- **identifier** – Identifier for new generation record.
- **other_attributes** – Optional other attributes as a dictionary or list of tuples to be added to the record optionally (default: None).

wasInfluencedBy(*influencee:* ProvEntity | QualifiedName | *str* | Identifier | ProvActivity | ProvAgent, *influencer:* ProvEntity | QualifiedName | *str* | Identifier | ProvActivity | ProvAgent, *identifier:* QualifiedName | *str* | Identifier | *None* = *None*, *other_attributes: dict[*QualifiedName | *str* | Identifier*, Any]* | *Iterable[tuple[*QualifiedName | *str* | Identifier*, Any]]* | *None* = *None*) → *ProvInfluence*

Creates a new influence record between two entities, activities or agents.

Parameters

- **influencee** – Influenced entity, activity or agent (relationship source).
- **influencer** – Influencing entity, activity or agent (relationship destination).
- **identifier** – Identifier for new influence record.
- **other_attributes** – Optional other attributes as a dictionary or list of tuples to be added to the record optionally (default: None).

wasInformedBy(*informed:* ProvActivity | QualifiedName | *str* | Identifier, *informant:* ProvActivity | QualifiedName | *str* | Identifier, *identifier:* QualifiedName | *str* | Identifier | *None* = *None*, *other_attributes: dict[*QualifiedName | *str* | Identifier*, Any]* | *Iterable[tuple[*QualifiedName | *str* | Identifier*, Any]]* | *None* = *None*) → *ProvCommunication*

Creates a new communication record for an entity.

Parameters

- **informed** – The informed activity (relationship destination).
- **informant** – The informing activity (relationship source).
- **identifier** – Identifier for new communication record.

- **other_attributes** – Optional other attributes as a dictionary or list of tuples to be added to the record optionally (default: None).

wasInvalidatedBy(*entity:* ProvEntity | QualifiedName | *str* | Identifier, *activity:* ProvActivity | QualifiedName | *str* | Identifier | *None* = *None*, *time: datetime* | *str* | *None* = *None*, *identifier:* QualifiedName | *str* | Identifier | *None* = *None*, *other_attributes: dict[*QualifiedName | *str* | Identifier*, Any]* | *Iterable[tuple[*QualifiedName | *str* | Identifier*, Any]]* | *None* = *None*) → *ProvInvalidation*

Creates a new invalidation record for an entity.

Parameters

- **entity** – Entity or a string identifier for the entity.
- **activity** – Activity or string identifier of the activity involved in the invalidation (default: None).
- **time** – Optional time for the invalidation (default: None). Either a `datetime.datetime` object or a string that can be parsed by `dateutil.parser()`.
- **identifier** – Identifier for the new invalidation record.
- **other_attributes** – Optional other attributes as a dictionary or list of tuples to be added to the record optionally (default: None).

wasQuotedFrom(*generatedEntity:* ProvEntity | QualifiedName | *str* | Identifier, *usedEntity:* ProvEntity | QualifiedName | *str* | Identifier, *activity:* ProvActivity | QualifiedName | *str* | Identifier | *None* = *None*, *generation:* ProvGeneration | QualifiedName | *str* | Identifier | *None* = *None*, *usage:* ProvUsage | QualifiedName | *str* | Identifier | *None* = *None*, *identifier:* QualifiedName | *str* | Identifier | *None* = *None*, *other_attributes: dict[*QualifiedName | *str* | Identifier*, Any]* | *Iterable[tuple[*QualifiedName | *str* | Identifier*, Any]]* | *None* = *None*) → *ProvDerivation*

Creates a new quotation record for a generated entity from a used entity.

Parameters

- **generatedEntity** – Entity or a string identifier for the generated entity (relationship source).
- **usedEntity** – Entity or a string identifier for the used entity (relationship destination).
- **activity** – Activity or string identifier of the activity involved in the quotation (default: None).
- **generation** – Optionally to state qualified quotation through a generation activity (default: None).
- **usage** – XXX (default: None).
- **identifier** – Identifier for new quotation record.
- **other_attributes** – Optional other attributes as a dictionary or list of tuples to be added to the record optionally (default: None).

wasRevisionOf(*generatedEntity:* ProvEntity | QualifiedName | *str* | Identifier, *usedEntity:* ProvEntity | QualifiedName | *str* | Identifier, *activity:* ProvActivity | QualifiedName | *str* | Identifier | *None* = *None*, *generation:* ProvGeneration | QualifiedName | *str* | Identifier | *None* = *None*, *usage:* ProvUsage | QualifiedName | *str* | Identifier | *None* = *None*, *identifier:* QualifiedName | *str* | Identifier | *None* = *None*, *other_attributes: dict[*QualifiedName | *str* | Identifier*, Any]* | *Iterable[tuple[*QualifiedName | *str* | Identifier*, Any]]* | *None* = *None*) → *ProvDerivation*

Creates a new revision record for a generated entity from a used entity.

Parameters

- **generatedEntity** – Entity or a string identifier for the generated entity (relationship source).
- **usedEntity** – Entity or a string identifier for the used entity (relationship destination).
- **activity** – Activity or string identifier of the activity involved in the revision (default: None).
- **generation** – Optionally to state qualified revision through a generation activity (default: None).
- **usage** – XXX (default: None).
- **identifier** – Identifier for new revision record.
- **other_attributes** – Optional other attributes as a dictionary or list of tuples to be added to the record optionally (default: None).

wasStartedBy(*activity:* ProvActivity | QualifiedName | *str* | Identifier, *trigger:* ProvEntity | QualifiedName | *str* | Identifier | *None* = *None*, *starter:* ProvActivity | QualifiedName | *str* | Identifier | *None* = *None*, *time:* *datetime* | *str* | *None* = *None*, *identifier:* QualifiedName | *str* | Identifier | *None* = *None*, *other_attributes:* *dict[*QualifiedName | *str* | Identifier*, Any]* | *Iterable[tuple[*QualifiedName | *str* | Identifier*, Any]]* | *None* = *None*) → *ProvStart*

Creates a new start record for an activity.

Parameters

- **activity** – Activity or a string identifier for the entity.
- **trigger** – Entity triggering the start of this activity.
- **starter** – Optionally extra activity to state a qualified start through which the trigger entity for the start is generated (default: None).
- **time** – Optional time for the start (default: None). Either a `datetime.datetime` object or a string that can be parsed by `dateutil.parser()`.
- **identifier** – Identifier for new start record.
- **other_attributes** – Optional other attributes as a dictionary or list of tuples to be added to the record optionally (default: None).

class prov.model.**ProvCommunication**(*bundle:* ProvBundle, *identifier:* QualifiedName | *None*, *attributes:* *dict[*QualifiedName | *str* | Identifier*, Any]* | *Iterable[tuple[*QualifiedName | *str* | Identifier*, Any]]* | *None* = *None*)

Bases: *ProvRelation*

Provenance Communication relationship.

FORMAL_ATTRIBUTES: tuple[*QualifiedName*, ...] = (<QualifiedName: prov:informed>, <QualifiedName: prov:informant>)

Formal attributes names of this record type, in the expected order.

class prov.model.**ProvDelegation**(*bundle:* ProvBundle, *identifier:* QualifiedName | *None*, *attributes:* *dict[*QualifiedName | *str* | Identifier*, Any]* | *Iterable[tuple[*QualifiedName | *str* | Identifier*, Any]]* | *None* = *None*)

Bases: *ProvRelation*

Provenance Delegation relationship.

FORMAL_ATTRIBUTES: tuple[*QualifiedName*, ...] = (<QualifiedName: prov:delegate>, <QualifiedName: prov:responsible>, <QualifiedName: prov:activity>)

Formal attributes names of this record type, in the expected order.

class prov.model.**ProvDerivation**(*bundle:* ProvBundle, *identifier:* QualifiedName | *None*, *attributes:* *dict[*QualifiedName | *str* | Identifier*, Any]* | *Iterable[tuple[*QualifiedName | *str* | Identifier*, Any]]* | *None* = *None*)

Bases: *ProvRelation*

Provenance Derivation relationship.

FORMAL_ATTRIBUTES: tuple[*QualifiedName*, ...] = (<QualifiedName: prov:generatedEntity>, <QualifiedName: prov:usedEntity>, <QualifiedName: prov:activity>, <QualifiedName: prov:generation>, <QualifiedName: prov:usage>)

Formal attributes names of this record type, in the expected order.

class prov.model.**ProvDocument**(*records: Iterable[*ProvRecord*]* | *None* = *None*, *namespaces: dict[str, str]* | *Iterable[*Namespace*]* | *None* = *None*)

Bases: *ProvBundle*

Provenance Document.

add_bundle(*bundle:* ProvBundle, *identifier:* QualifiedName | *None* = *None*) → None

Add a bundle to the current document.

Parameters

- **bundle** (*ProvBundle*) – The bundle to add to the document.
- **identifier** – The (optional) identifier to use for the bundle (default: None). If none given, use the identifier from the bundle itself.

bundle(*identifier:* QualifiedName | *str* | Identifier) → *ProvBundle*

Returns a new bundle from the current document.

Parameters
identifier – The identifier to use for the bundle.

Returns
ProvBundle

property bundles: Iterable[*ProvBundle*]

Returns bundles contained in the document

Returns
Iterable of *ProvBundle*.

static deserialize(*source: IOBase* | *str* | *bytes* | *PathLike* | *None* = *None*, *content: str* | *bytes* | *None* = *None*, *format: str* = *'json'*, ***args: Any*) → *ProvDocument*

Deserialize the *ProvDocument* from source (a stream or a file path) or directly from a string content.

Available serializers can be queried by the value of *:py:attr:~prov.serializers.Registry.serializers* after loading them via *:py:func:~prov.serializers.Registry.load_serializers()*.

Note: Not all serializers support deserialization.

Parameters

- **source** – Stream object to deserialize the PROV document from (default: None).
- **content** – String to deserialize the PROV document from (default: None).

- **format** – Serialization format (default: 'json'), defaulting to PROV-JSON.

Returns
ProvDocument

flattened() → *ProvDocument*

Flattens the document by moving all the records in its bundles up to the document level.

Returns
ProvDocument – the (new) flattened document.

has_bundles() → bool

True if the object has at least one bundle, *False* otherwise.

Returns
bool

is_bundle() → bool

True if the object is a bundle, *False* otherwise.

Returns
bool

is_document() → bool

True if the object is a document, *False* otherwise.

Returns
bool

serialize(*destination: IOBase | str | bytes | PathLike | None = None*, *format: str = 'json'*, ***args: Any*) → str | None

Serialize the *ProvDocument* to the destination.

Available serializers can be queried by the value of *:py:attr:~prov.serializers.Registry.serializers* after loading them via *:py:func:~prov.serializers.Registry.load_serializers()*.

Parameters

- **destination** – Stream object to serialize the output to. Default is *None*, which serializes as a string.
- **format** – Serialization format (default: 'json'), defaulting to PROV-JSON.

Returns
Serialization in a string if no destination was given, None otherwise.

unified() → *ProvDocument*

Returns a new document containing all records having the same identifiers unified (including those inside bundles).

Returns
ProvDocument

update(*other:* ProvBundle) → None

Append all the records of the *other* document/bundle into this document. Bundles having the same identifiers will be merged.

Parameters
other (*ProvDocument* or *ProvBundle*) – The other document/bundle whose records to be appended.

Returns
None.

class prov.model.**ProvElement**(*bundle:* ProvBundle, *identifier:* QualifiedName | *None*, *attributes:* *dict[*QualifiedName | *str* | Identifier*, Any]* | *Iterable[tuple[*QualifiedName | *str* | Identifier*, Any]]* | *None* = *None*)

Bases: *ProvRecord*

Provenance Element (nodes in the provenance graph).

is_element() → bool

True, if the record is an element, False otherwise.

Returns
bool

exception prov.model.**ProvElementIdentifierRequired**

Bases: *ProvException*

Exception for a missing element identifier.

class prov.model.**ProvEnd**(*bundle:* ProvBundle, *identifier:* QualifiedName | *None*, *attributes:* *dict[*QualifiedName | *str* | Identifier*, Any]* | *Iterable[tuple[*QualifiedName | *str* | Identifier*, Any]]* | *None* = *None*)

Bases: *ProvRelation*

Provenance End relationship.

FORMAL_ATTRIBUTES: tuple[*QualifiedName*, ...] = (<QualifiedName: prov:activity>, <QualifiedName: prov:trigger>, <QualifiedName: prov:ender>, <QualifiedName: prov:time>)

Formal attributes names of this record type, in the expected order.

class prov.model.**ProvEntity**(*bundle:* ProvBundle, *identifier:* QualifiedName | *None*, *attributes:* *dict[*QualifiedName | *str* | Identifier*, Any]* | *Iterable[tuple[*QualifiedName | *str* | Identifier*, Any]]* | *None* = *None*)

Bases: *ProvElement*

Provenance Entity element

alternateOf(*alternate2:* ProvEntity | QualifiedName | *str* | Identifier) → *ProvEntity*

Creates a new alternate record between this and another entity.

Parameters
alternate2 – Entity or a string identifier for the second entity.

hadMember(*entity:* ProvEntity | QualifiedName | *str* | Identifier) → *ProvEntity*

Creates a new membership record to an entity for a collection.

Parameters
entity – Entity to be added to the collection.

specializationOf(*generalEntity:* ProvEntity | QualifiedName | *str* | Identifier) → *ProvEntity*

Creates a new specialisation record for this from a general entity.

Parameters
generalEntity – Entity or a string identifier for the general entity.

wasAttributedTo(*agent:* ProvAgent | ProvEntity | ProvActivity | QualifiedName | *str* | Identifier, *attributes:* *dict[*QualifiedName | *str* | Identifier*, Any]* | *Iterable[tuple[*QualifiedName | *str* | Identifier*, Any]]* | *None* = *None*) → *ProvEntity*

Creates a new attribution record between this entity and an agent.

Parameters

- **agent** – Agent or string identifier of the agent involved in the attribution.
- **attributes** – Optional other attributes as a dictionary or list of tuples to be added to the record optionally (default: None).

wasDerivedFrom(*usedEntity:* ProvEntity | QualifiedName | *str* | Identifier, *activity:* ProvActivity | QualifiedName | *str* | Identifier | *None* = *None*, *generation:* ProvGeneration | QualifiedName | *str* | Identifier | *None* = *None*, *usage:* ProvUsage | QualifiedName | *str* | Identifier | *None* = *None*, *attributes: dict[*QualifiedName | *str* | Identifier*, Any]* | *Iterable[tuple[*QualifiedName | *str* | Identifier*, Any]]* | *None* = *None*) → *ProvEntity*

Creates a new derivation record for this entity from a used entity.

Parameters

- **usedEntity** – Entity or a string identifier for the used entity.
- **activity** – Activity or string identifier of the activity involved in the derivation (default: None).
- **generation** – Optional generation record to state qualified derivation through an internal generation (default: None).
- **usage** – Optional usage record to state qualified derivation through an internal usage (default: None).
- **attributes** – Optional other attributes as a dictionary or list of tuples to be added to the record optionally (default: None).

wasGeneratedBy(*activity:* ProvActivity | QualifiedName | *str* | Identifier | *None* = *None*, *time: datetime* | *str* | *None* = *None*, *attributes: dict[*QualifiedName | *str* | Identifier*, Any]* | *Iterable[tuple[*QualifiedName | *str* | Identifier*, Any]]* | *None* = *None*) → *ProvEntity*

Creates a new generation record to this entity.

Parameters

- **activity** – Activity or string identifier of the activity involved in the generation (default: None).
- **time** – Optional time for the generation (default: None). Either a `datetime.datetime` object or a string that can be parsed by `dateutil.parser()`.
- **attributes** – Optional other attributes as a dictionary or list of tuples to be added to the record optionally (default: None).

wasInvalidatedBy(*activity:* ProvActivity | QualifiedName | *str* | Identifier | *None*, *time: datetime* | *str* | *None* = *None*, *attributes: dict[*QualifiedName | *str* | Identifier*, Any]* | *Iterable[tuple[*QualifiedName | *str* | Identifier*, Any]]* | *None* = *None*) → *ProvEntity*

Creates a new invalidation record for this entity.

Parameters

- **activity** – Activity or string identifier of the activity involved in the invalidation (default: None).

- **time** – Optional time for the invalidation (default: None). Either a `datetime.datetime` object or a string that can be parsed by `dateutil.parser()`.
- **attributes** – Optional other attributes as a dictionary or list of tuples to be added to the record optionally (default: None).

exception prov.model.**ProvException**

Bases: *Error*

Base class for PROV model exceptions.

exception prov.model.**ProvExceptionInvalidQualifiedName**(*qname: Any*)

Bases: *ProvException*

Exception for an invalid qualified identifier name.

qname = None

Intended qualified name.

class prov.model.**ProvGeneration**(*bundle:* ProvBundle, *identifier:* QualifiedName | *None*, *attributes:* *dict[*QualifiedName | *str* | Identifier*, Any]* | *Iterable[tuple[*QualifiedName | *str* | Identifier*, Any]]* | *None = None*)

Bases: *ProvRelation*

Provenance Generation relationship.

FORMAL_ATTRIBUTES: tuple[*QualifiedName*, ...] = (<QualifiedName: prov:entity>, <QualifiedName: prov:activity>, <QualifiedName: prov:time>)

Formal attributes names of this record type, in the expected order.

class prov.model.**ProvInfluence**(*bundle:* ProvBundle, *identifier:* QualifiedName | *None*, *attributes:* *dict[*QualifiedName | *str* | Identifier*, Any]* | *Iterable[tuple[*QualifiedName | *str* | Identifier*, Any]]* | *None = None*)

Bases: *ProvRelation*

Provenance Influence relationship.

FORMAL_ATTRIBUTES: tuple[*QualifiedName*, ...] = (<QualifiedName: prov:influencee>, <QualifiedName: prov:influencer>)

Formal attributes names of this record type, in the expected order.

class prov.model.**ProvInvalidation**(*bundle:* ProvBundle, *identifier:* QualifiedName | *None*, *attributes:* *dict[*QualifiedName | *str* | Identifier*, Any]* | *Iterable[tuple[*QualifiedName | *str* | Identifier*, Any]]* | *None = None*)

Bases: *ProvRelation*

Provenance Invalidation relationship.

FORMAL_ATTRIBUTES: tuple[*QualifiedName*, ...] = (<QualifiedName: prov:entity>, <QualifiedName: prov:activity>, <QualifiedName: prov:time>)

Formal attributes names of this record type, in the expected order.

class prov.model.**ProvMembership**(*bundle:* ProvBundle, *identifier:* QualifiedName | *None*, *attributes:* *dict[*QualifiedName | *str* | Identifier*, Any]* | *Iterable[tuple[*QualifiedName | *str* | Identifier*, Any]]* | *None = None*)

Bases: *ProvRelation*

Provenance Membership relationship.

FORMAL_ATTRIBUTES: tuple[*QualifiedName*, ...] = (<QualifiedName: prov:collection>, <QualifiedName: prov:entity>)

Formal attributes names of this record type, in the expected order.

class prov.model.**ProvMention**(*bundle:* ProvBundle, *identifier:* QualifiedName | *None*, *attributes:* *dict[*QualifiedName | *str* | Identifier*, Any]* | *Iterable[tuple[*QualifiedName | *str* | Identifier*, Any]]* | *None* = *None*)

Bases: *ProvSpecialization*

Provenance Mention relationship (specific Specialization).

FORMAL_ATTRIBUTES: tuple[*QualifiedName*, ...] = (<QualifiedName: prov:specificEntity>, <QualifiedName: prov:generalEntity>, <QualifiedName: prov:bundle>)

Formal attributes names of this record type, in the expected order.

class prov.model.**ProvRecord**(*bundle:* ProvBundle, *identifier:* QualifiedName | *None*, *attributes:* *dict[*QualifiedName | *str* | Identifier*, Any]* | *Iterable[tuple[*QualifiedName | *str* | Identifier*, Any]]* | *None* = *None*)

Bases: `object`

Base class for PROV records.

FORMAL_ATTRIBUTES: tuple[*QualifiedName*, ...] = ()

Formal attributes names of this record type, in the expected order.

add_asserted_type(*type_identifier:* QualifiedName) → None

Adds a PROV type assertion to the record.

Parameters
type_identifier – PROV namespace identifier to add.

add_attributes(*attributes: dict[*QualifiedName | *str* | Identifier*, Any]* | *Iterable[tuple[*QualifiedName | *str* | Identifier*, Any]]*) → None

Add attributes to the record.

Parameters
attributes – Dictionary of attributes, with keys being qualified identifiers. Alternatively, an iterable of tuples (key, value) with the keys satisfying the same condition.

property args: tuple

All values of the record's formal attributes.

Returns
Tuple

property attributes: list[tuple[*QualifiedName*, Any]]

All record attributes.

Returns
List of tuples (name, value)

property bundle: *ProvBundle*

Bundle of the record.

Returns
ProvBundle

copy() → *ProvRecord*

Return an exact copy of this record.

property extra_attributes: tuple[tuple[*QualifiedName*, Any], ...]

All names and values of the record's attributes that are not formal attributes.

Returns
Tuple of tuples (name, value)

property formal_attributes: tuple[tuple[*QualifiedName*, Any], ...]

All names and values of the record's formal attributes.

Returns
Tuple of tuples (name, value)

get_asserted_types() → set[*QualifiedName*]

Returns the set of all asserted PROV types of this record.

get_attribute(*attr_name:* QualifiedName | *str* | Identifier) → set

Returns the attribute values (if any) for the specified attribute name).

Parameters
attr_name – Name of the attribute.

Returns
Set of value(s) of the specified attribute.

Return type
set

get_provn() → str

Returns the PROV-N representation of the record.

Returns
String

get_type() → *QualifiedName*

Returns the PROV type of the record.

property identifier: *QualifiedName* | None

Record's identifier.

is_element() → bool

True, if the record is an element, False otherwise.

Returns
bool

is_relation() → bool

True, if the record is a relation, False otherwise.

Returns
bool

property label: str

Identifying label of the record.

property value: Any

Value of the record.

class prov.model.**ProvRelation**(*bundle:* ProvBundle, *identifier:* QualifiedName | *None*, *attributes:* *dict[*QualifiedName | *str* | Identifier*, Any]* | *Iterable[tuple[*QualifiedName | *str* | Identifier*, Any]]* | *None* = *None*)

Bases: *ProvRecord*

Provenance Relationship (edge between nodes).

is_relation() → bool

True, if the record is a relation, False otherwise.

Returns
bool

class prov.model.**ProvSpecialization**(*bundle:* ProvBundle, *identifier:* QualifiedName | *None*, *attributes:* *dict[*QualifiedName | *str* | Identifier*, Any]* | *Iterable[tuple[*QualifiedName | *str* | Identifier*, Any]]* | *None* = *None*)

Bases: *ProvRelation*

Provenance Specialization relationship.

FORMAL_ATTRIBUTES: tuple[*QualifiedName*, ...] = (<QualifiedName: prov:specificEntity>, <QualifiedName: prov:generalEntity>)

Formal attributes names of this record type, in the expected order.

class prov.model.**ProvStart**(*bundle:* ProvBundle, *identifier:* QualifiedName | *None*, *attributes:* *dict[*QualifiedName | *str* | Identifier*, Any]* | *Iterable[tuple[*QualifiedName | *str* | Identifier*, Any]]* | *None* = *None*)

Bases: *ProvRelation*

Provenance Start relationship.

FORMAL_ATTRIBUTES: tuple[*QualifiedName*, ...] = (<QualifiedName: prov:activity>, <QualifiedName: prov:trigger>, <QualifiedName: prov:starter>, <QualifiedName: prov:time>)

Formal attributes names of this record type, in the expected order.

class prov.model.**ProvUsage**(*bundle:* ProvBundle, *identifier:* QualifiedName | *None*, *attributes:* *dict[*QualifiedName | *str* | Identifier*, Any]* | *Iterable[tuple[*QualifiedName | *str* | Identifier*, Any]]* | *None* = *None*)

Bases: *ProvRelation*

Provenance Usage relationship.

FORMAL_ATTRIBUTES: tuple[*QualifiedName*, ...] = (<QualifiedName: prov:activity>, <QualifiedName: prov:entity>, <QualifiedName: prov:time>)

Formal attributes names of this record type, in the expected order.

exception prov.model.**ProvWarning**

Bases: Warning

Base class for PROV model warnings.

prov.model.**encoding_provn_value**(*value: str* | *datetime* | *float* | *bool* | QualifiedName) → str

prov.model.**first**(*a_set: set[Any]*) → Any | None

prov.model.**parse_boolean**(*value: str*) → bool | None

prov.model.**parse_xsd_datetime**(*value: str*) → datetime | None

prov.model.**parse_xsd_types**(*value: str*, *datatype:* QualifiedName) → str | datetime | float | int | bool | *Identifier* | None

prov.model.**sorted_attributes**(*element:* QualifiedName, *attributes: Iterable[tuple[*QualifiedName*, Any]]*) → list[tuple[*QualifiedName*, Any]]

Helper function sorting attributes into the order required by PROV-XML.

Parameters

- **element** – The prov element used to derive the type and the attribute order for the type.
- **attributes** – The attributes to sort.

5.1.8 Module contents

exception prov.**Error**

Bases: Exception

Base class for all errors in this package.

prov.**read**(*source: str | bytes | os.PathLike*, *format: str | None = None*) → *ProvDocument* | None

Convenience function returning a ProvDocument instance.

It does a lazy format detection by simply using try/except for all known formats. The deserializers should fail fairly early when data of the wrong type is passed to them thus the try/except is likely cheap. One could of course also do some more advanced format auto-detection but I am not sure that is necessary.

The downside is that no proper error messages will be produced, use the format parameter to get the actual traceback.

CHAPTER

SIX

CREDITS

6.1 Development Lead

- Trung Dong Huynh (@trungdong)

6.2 Contributors

- Satrajit Ghosh @satra (*prov.serializers.provrdf* module)
- Lion Krischer @krischer (*prov.serializers.provxml* module and Python 3 support)
- Sam Millar

CHAPTER

SEVEN

HISTORY

7.1 2.1.1 (2025-06-24)

- No change - fixing the previous botched release

7.2 2.1.0 (2025-06-24)

- Added type annotations and mypy checks
- Added support for Python 3.13

7.3 2.0.2 (2025-06-07)

- Removed support for EOL Python 3.8
- Using pyproject.toml for project configurations (instead of setup.py)

7.4 2.0.1 (2024-06-10)

- Removed support for EOL Python 3.6 and 3.7
- Minor documentation update (#153)
- Stopped using deepcopy when duplicating Namespace (#158)
- Restricting rdflib package version to "<7" (#156)
- Raise an exception when an empty URI is registered as a namespace (#142)
- Ensure rdflib 6+ returns bytes when serializing tests (fixed #151)
- Removed fancy label output for bundle

7.5 2.0.0 (2020-11-01)

- Removed support for EOL Python 2
- Testing against Python 3.6+ and Pypy3

7.6 1.5.3 (2018-11-20)

- Reorganised source code to /src
- Added Python 3.7 support
- Removed Python 3.3 support due to end-of-life
- plus minor improvements and bug fixes

7.7 1.5.2 (2018-02-06)

- Fixed association relation in RDF serialisation
- Fixed compatibility with networkx 2.0+

7.8 1.5.1 (2017-07-18)

- Replaced pydotplus with pydot (see #111)
- Fixed datetime and bundle error in RDF serialisation
- Tested against Python 3.6
- Improved documentation

7.9 1.5.0 (2016-10-19)

- Added: Support for PROV-O (RDF) serialization and deserialization
- Added: *direction* option for *prov.dot.prov_to_dot()*
- Added: *prov.graph.graph_to_prov()* to convert a MultiDiGraph back to a *ProvDocument*
- Testing with Python 3.5
- Various minor bug fixes and improvements

7.10 1.4.0 (2015-08-13)

- Changed the type of qualified names to prov:QUALIFIED_NAME (fixed #68)
- Removed XSDQName class and stopped supporting parsing xsd:QName as qualified names
- Replaced pydot dependency with pydotplus
- Removed support for Python 2.6
- Various minor bug fixes and improvements

7.11 1.3.2 (2015-06-17)

- Added: prov-compare script to check equivalence of two PROV files (currently supporting JSON and XML)
- Fixed: deserialising Python 3's bytes objects (issue #67)

7.12 1.3.1 (2015-02-27)

- Fixed unicode issue with deserialising text contents
- Set the correct version requirement for six
- Fixed format selection in prov-convert script

7.13 1.3.0 (2015-02-03)

- Python 3.3 and 3.4 supported
- Updated prov-convert script to support XML output
- Added missing test JSON and XML files in distributions

7.14 1.2.0 (2014-12-19)

- Added: *prov.graph.prov_to_graph()* to convert a *ProvDocument* to a MultiDiGraph
- Added: PROV-N serializer
- Fixed: None values for empty formal attributes in PROV-N output (issue #60)
- Fixed: PROV-N representation for xsd:dateTime (issue #58)
- Fixed: Unintended merging of Identifier and QualifiedName values
- Fixed: Cloning the records when creating a new document from them
- Fixed: incorrect SoftwareAgent records in XML serialization

7.15 1.1.0 (2014-08-21)

- Added: Support for PROV-XML serialization and deserialization
- A *ProvRecord* instance can now be used as the value of an attributes
- Added: convenient assertions methods for *ProvEntity*, *ProvActivity*, and *ProvAgent*
- Added: *prov.model.ProvDocument.update()* and *prov.model.ProvBundle.update()*
- Fixed: Handling default namespaces of bundles when flattened

7.16 1.0.1 (2014-08-18)

- Added: Default namespace inheritance for bundles
- Fixed: *prov.model.NamespaceManager.valid_qualified_name()* did not support `XSDQName`
- Added: Convenience *prov.read()* method with a lazy format detection
- Added: Convenience *plot()* method on the *ProvBundle* class (requiring matplotlib).
- Changed: The previous `add_record()` method renamed to *new_record()*
- Added: *add_record()* function which takes one argument, a *ProvRecord*, has been added
- Fixed: Document flattening (see *flattened()*)
- Added: `__hash__()` function added to *ProvRecord* (**at risk**: to be removed as *ProvRecord* is expected to be mutable)

- Added: *extra_attributes* added to mirror existing *formal_attributes*

7.17 1.0.0 (2014-07-15)

- The underlying data model has been rewritten and is **incompatible** with pre-1.0 versions.
- References to PROV elements (i.e. entities, activities, agents) in relation records are now QualifiedName instances.
- A document or bundle can have multiple records with the same identifier.
- PROV-JSON serializer and deserializer are now separated from the data model.
- Many tests added, including round-trip PROV-JSON encoding/decoding.
- For changes pre-1.0, see CHANGES.txt.

CHAPTER

EIGHT

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

p

INDEX

G

H

I

L

M

N

P

Q

R

S

U

V

W