prov Documentation

Release 1.5.3

Trung Dong Huynh

Nov 20, 2018

Contents

Contents:

CHAPTER 1

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/.

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 2

Installation

At the command line:

```
$ easy_install prov
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv prov
$ pip install prov
```

CHAPTER 3

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"])
```

(continues on next page)

(continued from previous page)

```
#   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 4

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
```

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 2.7, 3.4, 3.5, 3.6, 3.7, and for PyPy/PyPy3. Check https://travis-ci.org/trungdong/prov/pull_requests and make sure that the tests pass for all supported Python versions. (See pyenv for help on setting up multiple versions of Python locally for testing.)

4.4 Tips

To run a subset of tests:

```
$ python -m unittest prov.tests.test_json
```

CHAPTER 5

prov

5.1 prov package

5.1.1 Subpackages

prov.serializers package

Module contents

prov.serializers.**get**(*format_name*)
Returns the serializer class for the specified format. Raises a DoNotExist

prov.serializers.provjson module

class prov.serializers.provjson.**ProvJSONDecoder**(*encoding=None*, *object_hook=None*, *parse_float=None*, *parse_int=None*, *parse_constant=None*, *strict=True*, *object_pairs_hook=None*)
Bases: json.decoder.JSONDecoder

decode(*s*, **args*, ***kwargs*)
Return the Python representation of s (a str or unicode 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*, *encoding='utf-8'*, *default=None*)
Bases: json.encoder.JSONEncoder

default(*o*)
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 JSONEncoder.default(self, o)
```

class prov.serializers.provjson.**ProvJSONSerializer**(*document=None*)
Bases: prov.serializers.Serializer

PROV-JSON serializer for *ProvDocument*

deserialize(*stream*, ***kwargs*)
Deserialize from the PROV JSON representation to a *ProvDocument* instance.

Parameters **stream** – Input data.

serialize(*stream*, ***kwargs*)
Serializes a *ProvDocument* instance to PROV-JSON.

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

prov.serializers.provn module

class prov.serializers.provn.**ProvNSerializer**(*document=None*)
Bases: prov.serializers.Serializer

PROV-N serializer for ProvDocument

serialize(*stream*, ***kwargs*)
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

class prov.serializers.provrdf.**ProvRDFSerializer**(*document=None*)
Bases: prov.serializers.Serializer

PROV-O serializer for *ProvDocument*

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

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=None*, *rdf_format=u'trig'*, *PROV_N_MAP={<QualifiedName: prov:Entity>: u'entity'*, *<QualifiedName: prov:Activity>: u'activity'*, *<QualifiedName: prov:Generation>: u'wasGeneratedBy'*, *<QualifiedName: prov:Usage>: u'used'*, *<QualifiedName: prov:Communication>: u'wasInformedBy'*, *<QualifiedName: prov:Start>: u'wasStartedBy'*, *<QualifiedName: prov:End>: u'wasEndedBy'*, *<QualifiedName: prov:Invalidation>: u'wasInvalidatedBy'*, *<QualifiedName: prov:Derivation>: u'wasDerivedFrom'*, *<QualifiedName: prov:Agent>: u'agent'*, *<QualifiedName: prov:Attribution>: u'wasAttributedTo'*, *<QualifiedName: prov:Association>: u'wasAssociatedWith'*, *<QualifiedName: prov:Delegation>: u'actedOnBehalfOf'*, *<QualifiedName: prov:Influence>: u'wasInfluencedBy'*, *<QualifiedName: prov:Bundle>: u'bundle'*, *<QualifiedName: prov:Alternate>: u'alternateOf'*, *<QualifiedName: prov:Specialization>: u'specializationOf'*, *<QualifiedName: prov:Mention>: u'mentionOf'*, *<QualifiedName: prov:Membership>: u'hadMember'}*, ***kwargs*)

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*, *level=0*, *path=None*, *usename=True*)

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

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

Bases: prov.serializers.Serializer

PROV-XML serializer for *ProvDocument*

deserialize(*stream*, ***kwargs*)

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

Parameters **stream** – Input data.

deserialize_subtree(*xml_doc*, *bundle*)

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*, *force_types=False*, ***kwargs*)

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*, *element=None*, *force_types=False*)

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*)

prov.dot.**prov_to_dot**(*bundle*, *show_nary=True*, *use_labels=False*, *direction=u'BT'*, *show_element_attributes=True*, *show_relation_attributes=True*)

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*)

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*)

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*)

Bases: object

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

provn_representation()
PROV-N representation of qualified name in a string.

uri
Identifier's URI.

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

PROV Namespace.

contains(*identifier*)
Indicates whether the identifier provided is contained in this namespace.

Parameters **identifier** – Identifier to check.

Returns bool

prefix
Namespace prefix.

qname(*identifier*)
Returns the qualified name of the identifier given using the namespace prefix.

Parameters **identifier** – Identifier to resolve to a qualified name.

Returns *QualifiedName*

uri
Namespace URI.

class prov.identifier.**QualifiedName**(*namespace*, *localpart*)
Bases: *prov.identifier.Identifier*

Qualified name of an identifier in a particular namespace.

localpart
Local part of qualified name.

namespace
Namespace of qualified name.

provn_representation()
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://provenance.ecs.soton.ac.uk/prov-json/

class prov.model.**Literal**(*value*, *datatype=None*, *langtag=None*)
Bases: object

datatype

has_no_langtag()

langtag

provn_representation()

value

class prov.model.**NamespaceManager**(*namespaces=None*, *default=None*, *parent=None*)
Bases: dict

Manages namespaces for PROV documents and bundles.

add_namespace(*namespace*)
Adds a namespace (if not available, yet).

Parameters **namespace** – *Namespace* to add.

add_namespaces(*namespaces*)
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=u'id'*)
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()
Returns the default namespace.

Returns *Namespace*

get_namespace(*uri*)
Returns the namespace prefix for the given URI.

Parameters **uri** – Namespace URI.

Returns *Namespace*.

get_registered_namespaces()
Returns all registered namespaces.

Returns Iterable of *Namespace*.

parent = None
Parent *NamespaceManager* this manager one is a child of.

set_default_namespace(*uri*)
Sets the default namespace to the one of a given URI.

Parameters **uri** – Namespace URI.

valid_qualified_name(*qname*)
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*, *identifier*, *attributes=None*)
Bases: *prov.model.ProvElement*

Provenance Activity element.

FORMAL_ATTRIBUTES = (<QualifiedName: prov:startTime>, <QualifiedName: prov:endTime>)

get_endTime()
Returns the time the activity ended.

Returns datetime.datetime

get_startTime()
Returns the time the activity started.

Returns datetime.datetime

set_time(*startTime=None*, *endTime=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*, *time=None*, *attributes=None*)
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*, *plan=None*, *attributes=None*)
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*, *ender=None*, *time=None*, *attributes=None*)
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*, *attributes=None*)
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*, *starter=None*, *time=None*, *attributes=None*)
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** – 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()`.
- **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*, *identifier*, *attributes=None*)
Bases: *prov.model.ProvElement*

Provenance Agent element.

actedOnBehalfOf(*responsible*, *activity=None*, *attributes=None*)
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*, *identifier*, *attributes=None*)
Bases: *prov.model.ProvRelation*

Provenance Alternate relationship.

FORMAL_ATTRIBUTES = (<QualifiedName: prov:alternate1>, <QualifiedName: prov:alternat

class prov.model.**ProvAssociation**(*bundle*, *identifier*, *attributes=None*)
Bases: *prov.model.ProvRelation*

Provenance Association relationship.

FORMAL_ATTRIBUTES = (<QualifiedName: prov:activity>, <QualifiedName: prov:agent>, <Q

class prov.model.**ProvAttribution**(*bundle*, *identifier*, *attributes=None*)
Bases: *prov.model.ProvRelation*

Provenance Attribution relationship.

FORMAL_ATTRIBUTES = (<QualifiedName: prov:entity>, <QualifiedName: prov:agent>)

class prov.model.**ProvBundle**(*records=None*, *identifier=None*, *namespaces=None*, *document=None*)
Bases: `object`

PROV Bundle

actedOnBehalfOf(*delegate*, *responsible*, *activity=None*, *identifier=None*, *other_attributes=None*)
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*, *startTime=None*, *endTime=None*, *other_attributes=None*)
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*, *uri=None*)
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*)
Adds a new record that to the bundle.

Parameters **record** – *ProvRecord* to be added.

agent(*identifier*, *other_attributes=None*)
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*, *alternate2*)
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*, *alternate2*)
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*, *agent=None*, *plan=None*, *identifier=None*, *other_attributes=None*)
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*, *agent*, *identifier=None*, *other_attributes=None*)
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).

bundles
Returns bundles contained in the document

Returns Iterable of *ProvBundle*.

collection(*identifier*, *other_attributes=None*)
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*, *informant*, *identifier=None*, *other_attributes=None*)
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).

default_ns_uri
Returns the default namespace's URI, if any.

Returns URI as string.

delegation(*delegate*, *responsible*, *activity=None*, *identifier=None*, *other_attributes=None*)
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*, *usedEntity*, *activity=None*, *generation=None*, *usage=None*, *identifier=None*, *other_attributes=None*)
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).

document
Returns the parent document, if any.

Returns *ProvDocument*.

end(*activity*, *trigger=None*, *ender=None*, *time=None*, *identifier=None*, *other_attributes=None*)
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*, *other_attributes=None*)
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*, *activity=None*, *time=None*, *identifier=None*, *other_attributes=None*)
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()
Returns the default namespace.

Returns *Namespace*

get_provn(*_indent_level=0*)
Returns the PROV-N representation of the bundle.

Returns String

get_record(*identifier*)
Returns a specific record matching a given identifier.

Parameters **identifier** – Record identifier.

Returns *ProvRecord*

get_records(*class_or_type_or_tuple=None*)
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()
Returns all registered namespaces.

Returns Iterable of *Namespace*.

hadMember(*collection*, *entity*)
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*, *usedEntity*, *activity=None*, *generation=None*, *usage=None*, *identifier=None*, *other_attributes=None*)
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()
True if the object has at least one bundle, *False* otherwise.

Returns bool

identifier
Returns the bundle's identifier

influence(*influencee*, *influencer*, *identifier=None*, *other_attributes=None*)
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*, *activity=None*, *time=None*, *identifier=None*, *other_attributes=None*)
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 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()
True if the object is a bundle, *False* otherwise.

Returns bool

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

Returns bool

membership(*collection*, *entity*)
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*, *generalEntity*, *bundle*)
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*, *generalEntity*, *bundle*)
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

namespaces
Returns the set of registered namespaces.

Returns Set of *Namespace*.

new_record(*record_type*, *identifier*, *attributes=None*, *other_attributes=None*)
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=None*, *show_nary=True*, *use_labels=False*, *show_element_attributes=True*, *show_relation_attributes=True*)
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*, *usedEntity*, *activity=None*, *generation=None*, *usage=None*, *identifier=None*, *other_attributes=None*)
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*, *usedEntity*, *activity=None*, *generation=None*, *usage=None*, *identifier=None*, *other_attributes=None*)
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).

records
Returns the list of all records in the current bundle

revision(*generatedEntity*, *usedEntity*, *activity=None*, *generation=None*, *usage=None*, *identifier=None*, *other_attributes=None*)
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*)
Sets the default namespace through a given URI.

Parameters **uri** – Namespace URI.

specialization(*specificEntity*, *generalEntity*)
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*, *generalEntity*)
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*, *trigger=None*, *starter=None*, *time=None*, *identifier=None*, *other_attributes=None*)
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()
Unifies all records in the bundle that haves same identifiers

Returns *ProvBundle* – the new unified bundle.

update(*other*)
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*, *entity=None*, *time=None*, *identifier=None*, *other_attributes=None*)
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*, *entity=None*, *time=None*, *identifier=None*, *other_attributes=None*)
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*)

wasAssociatedWith(*activity*, *agent=None*, *plan=None*, *identifier=None*, *other_attributes=None*)
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*, *agent*, *identifier=None*, *other_attributes=None*)
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*, *usedEntity*, *activity=None*, *generation=None*, *usage=None*, *identifier=None*, *other_attributes=None*)
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*, *trigger=None*, *ender=None*, *time=None*, *identifier=None*, *other_attributes=None*)
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*, *activity=None*, *time=None*, *identifier=None*, *other_attributes=None*)
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*, *influencer*, *identifier=None*, *other_attributes=None*)
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*, *informant*, *identifier=None*, *other_attributes=None*)
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*, *activity=None*, *time=None*, *identifier=None*, *other_attributes=None*)
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 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*, *usedEntity*, *activity=None*, *generation=None*, *usage=None*, *identifier=None*, *other_attributes=None*)
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*, *usedEntity*, *activity=None*, *generation=None*, *usage=None*, *identifier=None*, *other_attributes=None*)
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*, *trigger=None*, *starter=None*, *time=None*, *identifier=None*, *other_attributes=None*)
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*, *identifier*, *attributes=None*)
Bases: *prov.model.ProvRelation*

Provenance Communication relationship.

FORMAL_ATTRIBUTES = (<QualifiedName: prov:informed>, <QualifiedName: prov:informant>

class prov.model.**ProvDelegation**(*bundle*, *identifier*, *attributes=None*)
Bases: *prov.model.ProvRelation*

Provenance Delegation relationship.

FORMAL_ATTRIBUTES = (<QualifiedName: prov:delegate>, <QualifiedName: prov:responsibl

class prov.model.**ProvDerivation**(*bundle*, *identifier*, *attributes=None*)
Bases: *prov.model.ProvRelation*

Provenance Derivation relationship.

FORMAL_ATTRIBUTES = (<QualifiedName: prov:generatedEntity>, <QualifiedName: prov:use

class prov.model.**ProvDocument**(*records=None*, *namespaces=None*)
Bases: *prov.model.ProvBundle*

Provenance Document.

add_bundle(*bundle*, *identifier=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*)
Returns a new bundle from the current document.

Parameters **identifier** – The identifier to use for the bundle.

Returns *ProvBundle*

bundles
Returns bundles contained in the document

Returns Iterable of *ProvBundle*.

static deserialize(*source=None*, *content=None*, *format=u'json'*, ***args*)
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()
Flattens the document by moving all the records in its bundles up to the document level.

Returns *ProvDocument* – the (new) flattened document.

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

Returns bool

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

Returns bool

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

Returns bool

serialize(*destination=None*, *format=u'json'*, ***args*)
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()
Returns a new document containing all records having same identifiers unified (including those inside bundles).

Returns *ProvDocument*

update(*other*)
Append all the records of the *other* document/bundle into this document. Bundles having 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*, *identifier*, *attributes=None*)
Bases: *prov.model.ProvRecord*

Provenance Element (nodes in the provenance graph).

is_element()
True, if the record is an element, False otherwise.

Returns bool

exception prov.model.**ProvElementIdentifierRequired**
Bases: *prov.model.ProvException*

Exception for a missing element identifier.

class prov.model.**ProvEnd**(*bundle*, *identifier*, *attributes=None*)
Bases: *prov.model.ProvRelation*

Provenance End relationship.

FORMAL_ATTRIBUTES = (<QualifiedName: prov:activity>, <QualifiedName: prov:trigger>,

class prov.model.**ProvEntity**(*bundle*, *identifier*, *attributes=None*)
Bases: *prov.model.ProvElement*

Provenance Entity element

alternateOf(*alternate2*)
Creates a new alternate record between this and another entity.

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

hadMember(*entity*)
Creates a new membership record to an entity for a collection.

Parameters **entity** – Entity to be added to the collection.

specializationOf(*generalEntity*)
Creates a new specialisation record for this from a general entity.

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

wasAttributedTo(*agent*, *attributes=None*)
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*, *activity=None*, *generation=None*, *usage=None*, *attributes=None*)
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** – Optionally extra activity to state qualified derivation through an internal generation (default: None).
- **usage** – Optionally extra entity 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*, *time=None*, *attributes=None*)
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*, *time=None*, *attributes=None*)
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: *prov.Error*

Base class for PROV model exceptions.

exception prov.model.**ProvExceptionInvalidQualifiedName**(*qname*)
Bases: *prov.model.ProvException*

Exception for an invalid qualified identifier name.

qname = None
Intended qualified name.

class prov.model.**ProvGeneration**(*bundle*, *identifier*, *attributes=None*)
Bases: *prov.model.ProvRelation*

Provenance Generation relationship.

FORMAL_ATTRIBUTES = (<QualifiedName: prov:entity>, <QualifiedName: prov:activity>, <

class prov.model.**ProvInfluence**(*bundle*, *identifier*, *attributes=None*)
Bases: *prov.model.ProvRelation*

Provenance Influence relationship.

FORMAL_ATTRIBUTES = (<QualifiedName: prov:influencee>, <QualifiedName: prov:influenc

class prov.model.**ProvInvalidation**(*bundle*, *identifier*, *attributes=None*)
Bases: *prov.model.ProvRelation*

Provenance Invalidation relationship.

FORMAL_ATTRIBUTES = (<QualifiedName: prov:entity>, <QualifiedName: prov:activity>, <

class prov.model.**ProvMembership**(*bundle*, *identifier*, *attributes=None*)
Bases: *prov.model.ProvRelation*

Provenance Membership relationship.

FORMAL_ATTRIBUTES = (<QualifiedName: prov:collection>, <QualifiedName: prov:entity>)

class prov.model.**ProvMention**(*bundle*, *identifier*, *attributes=None*)
Bases: *prov.model.ProvSpecialization*

Provenance Mention relationship (specific Specialization).

FORMAL_ATTRIBUTES = (<QualifiedName: prov:specificEntity>, <QualifiedName: prov:gene

class prov.model.**ProvRecord**(*bundle*, *identifier*, *attributes=None*)
Bases: object

Base class for PROV records.

FORMAL_ATTRIBUTES = ()

add_asserted_type(*type_identifier*)
Adds a PROV type assertion to the record.

Parameters **type_identifier** – PROV namespace identifier to add.

add_attributes(*attributes*)
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.

args
All values of the record's formal attributes.

Returns Tuple

attributes
All record attributes.

Returns List of tuples (name, value)

bundle
Bundle of the record.

Returns *ProvBundle*

copy()
Return an exact copy of this record.

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

Returns Tuple of tuples (name, value)

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

Returns Tuple of tuples (name, value)

get_asserted_types()
Returns the set of all asserted PROV types of this record.

get_attribute(*attr_name*)
Returns the attribute of the given name.

Parameters **attr_name** – Name of the attribute.

Returns Tuple (name, value)

get_provn()
Returns the PROV-N representation of the record.

Returns String

get_type()
Returns the PROV type of the record.

identifier
Record's identifier.

is_element()
True, if the record is an element, False otherwise.

Returns bool

is_relation()
True, if the record is a relation, False otherwise.

Returns bool

label
Identifying label of the record.

value
Value of the record.

class prov.model.**ProvRelation**(*bundle*, *identifier*, *attributes=None*)
Bases: *prov.model.ProvRecord*

Provenance Relationship (edge between nodes).

is_relation()
True, if the record is a relation, False otherwise.

Returns bool

class prov.model.**ProvSpecialization**(*bundle*, *identifier*, *attributes=None*)
Bases: *prov.model.ProvRelation*

Provenance Specialization relationship.

FORMAL_ATTRIBUTES = (<QualifiedName: prov:specificEntity>, <QualifiedName: prov:gene

class prov.model.**ProvStart**(*bundle*, *identifier*, *attributes=None*)
Bases: *prov.model.ProvRelation*

Provenance Start relationship.

FORMAL_ATTRIBUTES = (<QualifiedName: prov:activity>, <QualifiedName: prov:trigger>,

class prov.model.**ProvUsage**(*bundle*, *identifier*, *attributes=None*)
Bases: *prov.model.ProvRelation*

Provenance Usage relationship.

FORMAL_ATTRIBUTES = (<QualifiedName: prov:activity>, <QualifiedName: prov:entity>, <

exception prov.model.**ProvWarning**
Bases: exceptions.Warning

Base class for PROV model warnings.

prov.model.**encoding_provn_value**(*value*)

prov.model.**first**(*a_set*)

prov.model.**parse_boolean**(*value*)

prov.model.**parse_xsd_datetime**(*value*)

prov.model.**parse_xsd_types**(*value*, *datatype*)

prov.model.**sorted_attributes**(*element*, *attributes*)
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: exceptions.Exception

Base class for all errors in this package.

prov.**read**(*source*, *format=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 6

Credits

6.1 Development Lead

- Trung Dong Huynh (@trungdong)

6.2 Contributors

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

CHAPTER 7

History

7.1 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

7.2 1.5.2 (2018-02-06)

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

7.3 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

7.4 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

7.5 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

7.6 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)

7.7 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

7.8 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

7.9 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

7.10 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

7.11 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*

7.12 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 8

Indices and tables

- genindex
- modindex
- search

Python Module Index

p

Index

G

H

I

L

M

N

P

Q

R

S

U

V

W