W3C Recommendation

PROV-N: The Provenance Notation

W3C Recommendation 30 April 2013

This version:
http://www.w3.org/TR/2013/REC-prov-n-20130430/
Latest published version:
http://www.w3.org/TR/prov-n/
Implementation report:
http://www.w3.org/TR/2013/NOTE-prov-implementations-20130430/
Previous version:
http://www.w3.org/TR/2013/PR-prov-n-20130312/ (color-coded diff)
Editors:
Luc Moreau, University of Southampton
Paolo Missier, Newcastle University
Contributors:
James Cheney, University of Edinburgh
Stian Soiland-Reyes, University of Manchester

Please refer to the **errata** for this document, which may include some normative corrections.

The English version of this specification is the only normative version. Non-normative translations may also be available.

Abstract

Provenance is information about entities, activities, and people involved in producing a piece of data or thing, which can be used to form assessments about its quality, reliability or trustworthiness. PROV-DM is the conceptual data becifications. PROV-DM distinguishes core structures, forming the essence of provenance information, from extended structures catering for more specific uses of provenance. PROV-DM is organized in six components, respectively dealing with: (1) entities and activities, and the time at which they were created, used, or ended; (2) derivations of entities from entities; (3) agents bearing responsibility for entities that were generated and activities that happened; (4) a notion of bundle, a mechanism to support provenance of provenance; and, (5) properties to link entities that refer to the same thing; (6) collections forming a logical structure for its members.

To provide examples of the PROV data model, the PROV notation (PROV-N) is introduced: aimed at human consumption, PROV-N allows serializations of PROV instances to be created in a compact manner. PROV-N facilitates the mapping of the PROV data model to concrete syntax, and is used as the basis for a formal semantics of PROV. The purpose of this document is to define the PROV-N notation.

The PROV Document Overview describes the overall state of PROV, and should be read before other PROV documents.

Status of This Document

This section describes the status of this document at the time of its publication. Other documents may supersede this document. A list of current W3C publications and the latest revision of this technical report can be found in the W3C technical reports index at http://www.w3.org/TR/.

PROV Family of Documents

This document is part of the PROV family of documents, a set of documents defining various aspects that are necessary to achieve the vision of inter-operable interchange of provenance information in heterogeneous

environments such as the Web. These documents are listed below. Please consult the [PROV-OVERVIEW] for a guide to reading these documents.

- PROV-OVERVIEW (Note), an overview of the PROV family of documents [PROV-OVERVIEW];
- PROV-PRIMER (Note), a primer for the PROV data model [PROV-PRIMER];
- PROV-O (Recommendation), the PROV ontology, an OWL2 ontology allowing the mapping of the PROV data model to RDF [PROV-O];
- PROV-DM (Recommendation), the PROV data model for provenance [PROV-DM];
- PROV-N (Recommendation), a notation for provenance aimed at human consumption (this document);
- PROV-CONSTRAINTS (Recommendation), a set of constraints applying to the PROV data model [PROV-CONSTRAINTS];
- PROV-XML (Note), an XML schema for the PROV data model [PROV-XML];
- PROV-AQ (Note), mechanisms for accessing and querying provenance [PROV-AQ];
- PROV-DICTIONARY (Note) introduces a specific type of collection, consisting of key-entity pairs [PROV-DICTIONARY];
- PROV-DC (Note) provides a mapping between PROV-O and Dublin Core Terms [PROV-DC];
- PROV-SEM (Note), a declarative specification in terms of first-order logic of the PROV data model [PROV-SEM];
- PROV-LINKS (Note) introduces a mechanism to link across bundles [PROV-LINKS].

Endorsed By W3C

This document has been reviewed by W3C Members, by software developers, and by other W3C groups and interested parties, and is endorsed by the Director as a W3C Recommendation. It is a stable document and may be used as reference material or cited from another document. W3C's role in making the Recommendation is to draw attention to the specification and to promote its widespread deployment. This enhances the functionality and interoperability of the Web.

Please Send Comments

This document was published by the Provenance Working Group as a Recommendation. If you wish to make comments regarding this document, please send them to public-prov-comments@w3.org (subscribe, archives). All comments are welcome.

This document was produced by a group operating under the 5 February 2004 W3C Patent Policy. W3C maintains a public list of any patent disclosures made in connection with the deliverables of the group; that page also includes instructions for disclosing a patent. An individual who has actual knowledge of a patent which the individual believes contains Essential Claim(s) must disclose the information in accordance with section 6 of the W3C Patent Policy.

Table of Contents

1. Introduction

Provenance is a record that describes the people, institutions, entities, and activities, involved in producing, influencing, or delivering a piece of data or a thing in the world. Two companion specifications respectively define PROV-DM, a data model for provenance, allowing provenance descriptions to be expressed [PROV-DM] and a set of constraints that provenance descriptions are expected to satisfy [PROV-CONSTRAINTS].

1.1 Purpose of this Document and Target Audience

A key goal of PROV is the specification of a machine-processable data model for provenance. However, communicating provenance between humans is also important when teaching, illustrating, formalizing, and discussing provenance-related issues. With these two requirements in mind, this document introduces PROV-N, the PROV notation, a syntax designed to write instances of the PROV data model according to the following design principles:

- Technology independence. PROV-N provides a simple syntax that can be mapped to several technologies.
- Human readability. PROV-N follows a functional syntax style that is meant to be easily human-readable so it can be used in illustrative examples, such as those presented in the PROV documents suite.
- Formality. PROV-N is defined through a formal grammar amenable to be used with parser generators.

PROV-N has several known uses:

- It is the notation used in the examples found in [PROV-DM], as well as in the definition of PROV constraints [PROV-CONSTRAINTS];
- It is a source language for the encoding of PROV data model instances into a variety of target languages, including amongst others RDF [PROV-RDF] and XML [PROV-XML];
- It provides the basis for a formal semantics of the PROV data model [PROV-SEM], in which an interpretation is given to each element of the PROV-N language.

This document introduces the PROV-N grammar along with examples of its usage.

Its target audience is twofold:

- Developers of provenance management applications, as well as implementors of new PROV data model encodings, and thus in particular of PROV-N parsers. These readers may be interested in the entire structure of the grammar, starting from the top level nonterminal `document`.
- Readers of the [PROV-DM] and of [PROV-CONSTRAINTS] documents, who are interested in the details of the formal language underpinning the notation used in the examples and in the definition of the constraints. Those readers may find the `expression` nonterminal a useful entry point into the grammar.

1.2 Compliance with this Document

For the purpose of compliance, all sections of this document are normative, except Appendix A, Appendix B, and Appendix C.2.

- Information in tables is normative.
- Text in boxes labeled "Example" is informative.
- Productions (displayed in boxes) are normative, as opposed to the separate file grouping all productions for convenience of programmers, which is informative.

1.3 Structure of this Document

This document is structured as follows.

Section 2 provides general consideration about the PROV-N grammar.

Section 3 presents the grammar of all expressions of the language grouped according to the PROV data model components.

Section 4 defines the grammar of document, a house-keeping construct of PROV-N capable of packaging up PROV-N expressions and namespace declarations.

Section 5 defines the extensibility mechanism for the PROV-N notation.

Section 6 defines media type for the PROV-N notation.

1.4 Notational Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

The following namespaces prefixes are used throughout this document.

Table 1: Prefix and Namespaces used in this specification

prefix	namespace uri	definition
prov	http://www.w3.org/ns/prov#	The PROV namespace (see Section 3.7.4)
xsd	http://www.w3.org/2000/10/XMLSchema#	XML Schema Namespace [XMLSCHEMA11-2]
(others)	(various)	All other namespace prefixes are used in examples only. In particular, URIs starting with "http://example.com" represent some application-dependent URI [RFC3986]

2. General grammar considerations

For convenience, all productions presented in this document have been grouped in a separate file.

2.1 Functional-style Syntax

PROV-N adopts a functional-style syntax consisting of a predicate name and an ordered list of terms.

All PROV data model types have an identifier. Furthermore, some expressions also admit additional elements that further characterize it.

Example 1

The following expression should be read as "entity `e1`".

```
entity(e1)
```

Example 2

The following expression should be read as "activity `a2`, which occurred between `2011-11-16T16:00:00` and `2011-11-16T16:00:01`".

```
entity(e1)
activity(a2, 2011-11-16T16:00:00, 2011-11-16T16:00:01)
```

All PROV data model relations involve two primary elements, the *subject* and the *object*, in this order. Furthermore, some expressions also admit additional elements that further characterize it.

Example 3

The following expression should be read as "`e2` was derived from `e1`". Here `e2` is the subject, and `e1` is the object.

```
wasDerivedFrom(e2, e1)
```

Example 4

The following expression expands the above derivation relation by providing additional elements the optional activity `a`, the generation `g2`, and the usage `u1`:

```
wasDerivedFrom(e2, e1, a, g2, u1)
```

2.2 EBNF Grammar

The grammar is specified using a subset of the Extended Backus-Naur Form (EBNF) notation, as defined in Extensible Markup Language (XML) 1.1 [XML11] section 6 Notation.

The text below provides an introduction to the EBNF notation used in this document.

EBNF specifies a series of production rules (***production***). A production rule in the grammar defines a symbol `expr` (***nonterminal symbol***) using the following form:

`expr` ::= *term*

Symbols are written with an initial capital letter if they are the start symbol of a regular language, otherwise with an initial lowercase letter. A production rule in the grammar defines a symbol `<TERMINAL>` (***terminal symbol***) using the following form:

`<TERMINAL>` ::= *term*

Within the term on the right-hand side of a rule, the following *term*s are used to match strings of one or more characters:

- `expr`: matches production for nonterminal symbol `expr`
- `TERMINAL`: matches production for terminal symbol `TERMINAL`
- `"abc"`: matches the literal string inside the single quotes.
- `(term)?`: optional, matches *term* or nothing.
- `(term)+`: matches one or more occurrences of *term*.
- `(term)*`: matches zero or more occurrences of *term*.
- `(term | term)`: matches one of the two *term*s.

Where suitable, the PROV-N grammar reuses production and terminal names of the SPARQL grammar [RDF-SPARQL-QUERY].

2.3 Main Productions

Two productions are entry points to the grammar.

The production `expression` provides the structure for the *core expressions* of PROV-N.

[2] `expression` ::= (entityExpression | activityExpression | generationExpression | usageExpression | startExpression | endExpression | invalidationExpression | communicationExpression | agentExpression | associationExpression | attributionExpression | delegationExpression | derivationExpression | influenceExpression | alternateExpression | specializationExpression | membershipExpression | extensibilityExpression)

Each of the symbols included in `expression` above, i.e., `entityExpression`, `activityExpression` etc., corresponds to one concept (e.g., Entity, Activity, etc.) of the PROV data model.

Alternatively, the production rule `document` provides the overall structure of PROV-N descriptions. It is a wrapper for a set of expressions, such that the text for an element matches the corresponding `expression` production, and some namespace declarations.

2.4 Optional terms in expressions

Some terms in an expression may be optional. For example:

Example 5

```
wasDerivedFrom(e2, e1, a, g2, u1)
wasDerivedFrom(e2, e1)
```

In a derivation expression, the activity, generation, and usage are optional terms. They are specified in the first derivation, but not in the second.

Example 6

```
activity(a2, 2011-11-16T16:00:00, 2011-11-16T16:00:01)
activity(a1)
```

The start and end times for an activity are optional. They are specified in the first expression, but not in the second.

The general rule for optionals is that, if *none* of the optionals are used in the expression, then they are simply omitted, resulting in a simpler expression as in the examples above.

However, it may be the case that only some of the optional terms are omitted. Because the position of the terms in the expression matters, an additional marker must be used to indicate that a particular term is not available. The symbol `'-'` is used for this purpose.

Example 7

In the first expression below, all optionals are specified. However in the second and third, only one optional is specified, forcing the use of the marker for the missing terms.

```
wasDerivedFrom(e2, e1, a, g2, u1)
wasDerivedFrom(e2, e1, -, -, u1)
wasDerivedFrom(e2, e1, a, -, -)
```

Note that the more succinct form is just shorthand for a complete expression with all the markers specified:

Example 8

```
activity(a1)
activity(a1, -, -)
```

2.5 Identifiers and attributes

Almost all expressions defined in the grammar include an identifier (see Section 3.7.1 for the full syntax of identifiers). Most expressions can also include a set of attribute-value pairs, delimited by square brackets. Identifiers are optional except for Entities, Activities, and Agents. Identifiers are always the first term in any expression. *Optional* identifiers MUST be separated using a semi-colon ';', but where the identifiers are required, a regular comma ',' MUST be used. This makes it possible to completely omit an optional identifier with no ambiguity arising. Also, if the set of attribute-value pairs is present, it is always the last term in any expression.

Example 9

Derivation has an optional identifier. In the first expression, the identifier is not available, while it is explicit in the

second. The third example shows that one can optionally indicate the missing identifier using the - marker. This is equivalent to the first expression.

```
wasDerivedFrom(e2, e1)
wasDerivedFrom(d; e2, e1)
wasDerivedFrom(-; e2, e1)
```

Lack of attributes can be equivalently expressed by omitting the list, or by using an empty list.

Example 10

The first and second activity expressions do not specify any attributes, and are equivalent. The third activity expression specifies two attributes.

```
activity(ex:a1)
activity(ex:a1, [])
activity(ex:a1, [ex:param1="a", ex:param2="b"])
```

2.6 Comments

Comments in PROV-N take two forms:

- '//' outside an IRI_REF or STRING_LITERAL; such comments continue to the end of line (marked by characters U+000D or U+000A) or end of file if there is no end of line after the comment marker.
- '/*' ... '*/', outside an IRI_REF or STRING_LITERAL.

Comments are treated as white space.

3. PROV-N Productions per Component

This section introduces grammar productions for each expression, followed by small examples of expressions illustrating the grammar. Strings conforming to the grammar are valid expressions in the PROV-N language.

3.1 Component 1: Entities and Activities

3.1.1 Entity

```
[3] entityExpression            ::= "entity" "(" identifier optionalAttributeValuePairs ")"
[4] optionalAttributeValuePairs ::= ( "," "[" attributeValuePairs "]" )?
[5] attributeValuePairs         ::= ( | attributeValuePair ( "," attributeValuePair )* )
[6] attributeValuePair          ::= attribute "=" literal
```

The following table summarizes how each constituent of a PROV-DM Entity maps to a PROV-N syntax element.

Entity	Non-Terminal
id	identifier
attributes	optionalAttributeValuePairs

Example 11

```
entity(tr:WD-prov-dm-20111215, [ prov:type="document" ])
```

Here `tr:WD-prov-dm-20111215` is the entity identifier, and `[ prov:type="document" ]` groups the optional attributes, only one in this example, with their values.

```
entity(tr:WD-prov-dm-20111215)
```

Here, the optional attributes are absent.

3.1.2 Activity

```
[7] activityExpression ::= "activity" "(" identifier ( "," timeOrMarker "," timeOrMarker )?
                           optionalAttributeValuePairs ")"
```

```
[8]  timeOrMarker        ::=  ( time | "-" )
```

The following table summarizes how each constituent of a PROV-DM Activity maps to a PROV-N syntax element.

Activity	Non-Terminal
id	identifier
startTime	timeOrMarker
endTime	timeOrMarker
attributes	optionalAttributeValuePairs

Example 12

```
activity(ex:a10, 2011-11-16T16:00:00, 2011-11-16T16:00:01, [prov:type="createFile"])
```

Here `ex:a10` is the activity identifier, `2011-11-16T16:00:00` and `2011-11-16T16:00:01` are the optional start and end times for the activity, and `[prov:type="createFile"]` is a list of optional attributes.

The remaining examples show cases where some of the optionals are omitted.

```
activity(ex:a10)
activity(ex:a10, -, -)
activity(ex:a10, -, -, [prov:type="edit"])
activity(ex:a10, -, 2011-11-16T16:00:00)
activity(ex:a10, 2011-11-16T16:00:00, -)
activity(ex:a10, 2011-11-16T16:00:00, -, [prov:type="createFile"])
activity(ex:a10, [prov:type="edit"])
```

3.1.3 Generation

```
[9]   generationExpression  ::=  "wasGeneratedBy" "(" optionalIdentifier eIdentifier ( "," aIdentifierOrMarker
                                 "," timeOrMarker )? optionalAttributeValuePairs ")"
[10]  optionalIdentifier    ::=  ( identifierOrMarker ";" )?
[11]  identifierOrMarker    ::=  ( identifier | "-" )
```

The following table summarizes how each constituent of a PROV-DM Generation maps to a PROV-N syntax element.

Generation	Non-Terminal
id	optionalIdentifier
entity	eIdentifier
activity	aIdentifierOrMarker
time	timeOrMarker
attributes	optionalAttributeValuePairs

Example 13

```
wasGeneratedBy(ex:g1; tr:WD-prov-dm-20111215, ex:edit1, 2011-11-16T16:00:00,  [ex:fct="save"])
```

Here `ex:g1` is the optional generation identifier, `tr:WD-prov-dm-20111215` is the identifier of the entity being generated, `ex:edit1` is the optional identifier of the generating activity, `2011-11-16T16:00:00` is the optional generation time, and `[ex:fct="save"]` is a list of optional attributes.

The remaining examples show cases where some of the optionals are omitted.

```
wasGeneratedBy(e2, a1, -)
wasGeneratedBy(e2, a1, 2011-11-16T16:00:00)
wasGeneratedBy(e2, a1, -, [ex:fct="save"])
wasGeneratedBy(e2, [ex:fct="save"])
wasGeneratedBy(ex:g1; e)
wasGeneratedBy(ex:g1; e, a, tr:WD-prov-dm-20111215)
```

Additional semantic rules (Section 3.7.5) apply to generationExpression.

3.1.4 Usage

[12] usageExpression ::= "used" "(" optionalIdentifier aIdentifier ("," eIdentifierOrMarker "," timeOrMarker)? optionalAttributeValuePairs ")"

The following table summarizes how each constituent of a PROV-DM Usage maps to a PROV-N syntax element.

Usage	Non-Terminal
id	optionalIdentifier
activity	aIdentifier
entity	eIdentifierOrMarker
time	timeOrMarker
attributes	optionalAttributeValuePairs

Example 14

```
used(ex:u1; ex:act2, ar3:0111, 2011-11-16T16:00:00, [ex:fct="load"])
```

Here `ex:u1` is the optional usage identifier, `ex:act2` is the identifier of the using activity, `ar3:0111` is the identifier of the entity being used, `2011-11-16T16:00:00` is the optional usage time, and `[ex:fct="load"]` is a list of optional attributes.

The remaining examples show cases where some of the optionals are omitted.

```
used(ex:act2)
used(ex:act2, ar3:0111, 2011-11-16T16:00:00)
used(a1,e1, -, [ex:fct="load"])
used(ex:u1; ex:act2, ar3:0111, -)
```

Additional semantic rules (Section 3.7.5) apply to usageExpression.

3.1.5 Communication

[13] communicationExpression ::= "wasInformedBy" "(" optionalIdentifier aIdentifier "," aIdentifier optionalAttributeValuePairs ")"

The following table summarizes how each constituent of a PROV-DM Communication maps to a PROV-N syntax element.

Communication	Non-Terminal
id	optionalIdentifier
informed	aIdentifier
informant	aIdentifier
attributes	optionalAttributeValuePairs

Example 15

```
wasInformedBy(ex:inf1; ex:a1, ex:a2, [ex:param1="a", ex:param2="b"])
```

Here `ex:inf1` is the optional communication identifier, `ex:a1` is the identifier of the informed activity, `ex:a2` is the identifier of the informant activity, and `[ex:param1="a", ex:param2="b"]` is a list of optional attributes.

The remaining examples show cases where some of the optionals are omitted.

```
wasInformedBy(ex:a1, ex:a2)
wasInformedBy(ex:a1, ex:a2, [ex:param1="a", ex:param2="b"])
wasInformedBy(ex:i; ex:a1, ex:a2)
wasInformedBy(ex:i; ex:a1, ex:a2, [ex:param1="a", ex:param2="b"])
```

3.1.6 Start

[14] startExpression ::= "wasStartedBy" "(" optionalIdentifier aIdentifier ("," eIdentifierOrMarker "," aIdentifierOrMarker "," timeOrMarker)? optionalAttributeValuePairs ")"

The following table summarizes how each constituent of a PROV-DM Start maps to a PROV-N syntax element.

Start	Non-Terminal
id	optionalIdentifier
activity	aIdentifier
trigger	eIdentifierOrMarker
starter	aIdentifierOrMarker
time	timeOrMarker
attributes	optionalAttributeValuePairs

Example 16

```
wasStartedBy(ex:start; ex:act2, ex:trigger, ex:act1, 2011-11-16T16:00:00, [ex:param="a"])
```

Here `start` is the optional start identifier, `ex:act2` is the identifier of the started activity, `ex:trigger` is the optional identifier for the entity that triggered the activity start, `ex:act1` is the optional identifier for the activity that generated the (possibly unspecified) entity `ex:trigger`, `2011-11-16T16:00:00` is the optional start time, and `[ex:param="a"]` is a list of optional attributes.

The remaining examples show cases where some of the optionals are omitted.

```
wasStartedBy(ex:act2, -, ex:act1, -)
wasStartedBy(ex:act2, -, ex:act1, 2011-11-16T16:00:00)
wasStartedBy(ex:act2, -, -, 2011-11-16T16:00:00)
wasStartedBy(ex:act2, [ex:param="a"])
wasStartedBy(ex:start; ex:act2, e, ex:act1, 2011-11-16T16:00:00)
```

Additional semantic rules (Section 3.7.5) apply to `startExpression`.

3.1.7 End

[15] `endExpression` ::= `"wasEndedBy" "(" optionalIdentifier aIdentifier ( "," eIdentifierOrMarker "," aIdentifierOrMarker "," timeOrMarker )? optionalAttributeValuePairs ")"`

The following table summarizes how each constituent of a PROV-DM End maps to a PROV-N syntax element.

End	Non-Terminal
id	optionalIdentifier
activity	aIdentifier
trigger	eIdentifierOrMarker
ender	aIdentifierOrMarker
time	timeOrMarker
attributes	optionalAttributeValuePairs

Example 17

```
wasEndedBy(ex:end; ex:act2, ex:trigger, ex:act3,  2011-11-16T16:00:00, [ex:param="a"])
```

Here `end` is the optional end identifier, `ex:act2` is the identifier of the ending activity, `ex:trigger` is the identifier of the entity that triggered the activity end, `ex:act3` is the optional identifier for the activity that generated the (possibly unspecified) entity `ex:trigger`, `2011-11-16T16:00:00` is the optional usage time, and `[ex:param="a"]` is a list of optional attributes.

The remaining examples show cases where some of the optionals are omitted.

```
wasEndedBy(ex:act2, ex:trigger, -, -)
wasEndedBy(ex:act2, ex:trigger, -, 2011-11-16T16:00:00)
wasEndedBy(ex:act2, -, -, 2011-11-16T16:00:00)
wasEndedBy(ex:act2, -, -, 2011-11-16T16:00:00, [ex:param="a"])
wasEndedBy(ex:end; ex:act2)
wasEndedBy(ex:end; ex:act2, ex:trigger, -, 2011-11-16T16:00:00)
```

Additional semantic rules (Section 3.7.5) apply to `endExpression`.

3.1.8 Invalidation

[16] `invalidationExpression` ::= `"wasInvalidatedBy" "(" optionalIdentifier eIdentifier ( "," aIdentifierOrMarker "," timeOrMarker )? optionalAttributeValuePairs ")"`

The following table summarizes how each constituent of a PROV-DM Invalidation maps to a PROV-N syntax element.

Invalidation	Non-Terminal
id	`optionalIdentifier`
entity	`eIdentifier`
activity	`aIdentifierOrMarker`
time	`timeOrMarker`
attributes	`optionalAttributeValuePairs`

Example 18

```
wasInvalidatedBy(ex:inv; tr:WD-prov-dm-20111215, ex:edit1, 2011-11-16T16:00:00,  [ex:fct="save"])
```

Here `ex:inv` is the optional invalidation identifier, `tr:WD-prov-dm-20111215` is the identifier of the entity being invalidated, `ex:edit1` is the optional identifier of the invalidating activity, `2011-11-16T16:00:00` is the optional invalidation time, and `[ex:fct="save"]` is a list of optional attributes.

The remaining examples show cases where some of the optionals are omitted.

```
wasInvalidatedBy(tr:WD-prov-dm-20111215, ex:edit1, -)
wasInvalidatedBy(tr:WD-prov-dm-20111215, ex:edit1, 2011-11-16T16:00:00)
wasInvalidatedBy(e2, a1, -, [ex:fct="save"])
wasInvalidatedBy(e2, -, -, [ex:fct="save"])
wasInvalidatedBy(ex:inv; tr:WD-prov-dm-20111215, ex:edit1, -)
wasInvalidatedBy(tr:WD-prov-dm-20111215, ex:edit1, -)
```

Additional semantic rules (Section 3.7.5) apply to `invalidationExpression`.

3.2 Component 2: Derivations

3.2.1 Derivation

[17] `derivationExpression` ::= `"wasDerivedFrom" "(" optionalIdentifier eIdentifier "," eIdentifier ( "," aIdentifierOrMarker "," gIdentifierOrMarker "," uIdentifierOrMarker )? optionalAttributeValuePairs ")"`

The following table summarizes how each constituent of a PROV-DM Derivation maps to a PROV-N syntax element.

Derivation	Non-Terminal
id	`optionalIdentifier`
generatedEntity	`eIdentifier`
usedEntity	`eIdentifier`
activity	`aIdentifierOrMarker`
generation	`gIdentifierOrMarker`
usage	`uIdentifierOrMarker`
attributes	`optionalAttributeValuePairs`

Example 19

```
wasDerivedFrom(ex:d; e2, e1, a, g2, u1, [ex:comment="a righteous derivation"])
```

Here `d` is the optional derivation identifier, `e2` is the identifier for the entity being derived, `e1` is the identifier of the

entity from which `e2` is derived, `a` is the optional identifier of the activity which used/generated the entities, `g2` is the optional identifier of the generation, `u1` is the optional identifier of the usage, and `[ex:comment="a righteous derivation"]` is a list of optional attributes.

The remaining examples show cases where some of the optionals are omitted.

```
wasDerivedFrom(e2, e1)
wasDerivedFrom(e2, e1, a, g2, u1)
wasDerivedFrom(e2, e1, -, g2, u1)
wasDerivedFrom(e2, e1, a, -, u1)
wasDerivedFrom(e2, e1, a, g2, -)
wasDerivedFrom(e2, e1, a, -, -)
wasDerivedFrom(e2, e1, -, -, u1)
wasDerivedFrom(e2, e1, -, -, -)
wasDerivedFrom(ex:d; e2, e1, a, g2, u1)
wasDerivedFrom(-; e2, e1, a, g2, u1)
```

3.2.2 Revision

PROV-N provides no dedicated syntax for Revision. Instead, a Revision MUST be expressed as a `derivationExpression` with attribute `prov:type='prov:Revision'`.

Example 20

```
wasDerivedFrom(ex:d; e2, e1, a, g2, u1,
               [prov:type='prov:Revision',
                ex:comment="a righteous derivation"])
```

Here, the derivation from Example 19 is extended with a `prov:type` attribute and value `prov:Revision`. The expression `'prov:Revision'` is `convenienceNotation` to denote a `QUALIFIED_NAME` literal (See Section 3.7.3. Literal).

3.2.3 Quotation

PROV-N provides no dedicated syntax for Quotation. Instead, a Quotation MUST be expressed as a `derivationExpression` with attribute `prov:type='prov:Quotation'`.

Example 21

```
wasDerivedFrom(ex:quoteId1; ex:blockQuote,ex:blog, ex:act1, ex:g, ex:u,
               [ prov:type='prov:Quotation' ])
```

Here, the derivation is provided with a `prov:type` attribute and value `prov:Quotation`.

3.2.4 Primary Source

PROV-N provides no dedicated syntax for PrimarySource. Instead, a PrimarySource MUST be expressed as a `derivationExpression` with attribute `prov:type='prov:Primary-Source'`.

Example 22

```
wasDerivedFrom(ex:sourceId1;  ex:e1, ex:e2, ex:act, ex:g, ex:u,
               [ prov:type='prov:PrimarySource' ])
```

Here, the derivation is provided with a `prov:type` attribute and value `prov:PrimarySource`.

3.3 Component 3: Agents, Responsibility, and Influence

3.3.1 Agent

[18] `agentExpression` ::= `"agent" "(" identifier optionalAttributeValuePairs ")"`

The following table summarizes how each constituent of a PROV-DM Agent maps to a PROV-N syntax element.

Agent	Non-Terminal
id	identifier
attributes	optionalAttributeValuePairs

PROV-N provides no dedicated syntax for Person, Organization, SoftwareAgent. Instead, a Person, an Organization, or a SoftwareAgent MUST be expressed as an agentExpression with attribute `prov:type='prov:Person'`, `prov:type='prov:Organization'`, or `prov:type='prov:SoftwareAgent'`, respectively.

Example 23

```
agent(ex:ag4, [ prov:type='prov:Person', ex:name="David" ])
```

Here `ag` is the agent identifier, and `[ prov:type='prov:Person', ex:name="David" ]` is a list of optional attributes.

In the next example, the optional attributes are omitted.

```
agent(ex:ag4)
```

3.3.2 Attribution

[19] attributionExpression ::= "wasAttributedTo" "(" optionalIdentifier eIdentifier "," agIdentifier optionalAttributeValuePairs ")"

The following table summarizes how each constituent of a PROV-DM Attribution maps to a PROV-N syntax element.

Attribution	Non-Terminal
id	optionalIdentifier
entity	eIdentifier
agent	agIdentifier
attributes	optionalAttributeValuePairs

Example 24

```
wasAttributedTo(ex:attr; e, ag, [ex:license='cc:attributionURL' ])
```

Here `attr` is the optional attribution identifier, `e` is an entity identifier, `ag` is the identifier of the agent to whom the entity is ascribed, and `[ex:license='cc:attributionURL' ]` is a list of optional attributes.

The remaining examples show cases where some of the optionals are omitted.

```
wasAttributedTo(e, ag)
wasAttributedTo(e, ag, [ex:license='cc:attributionURL' ])
```

3.3.3 Association

[20] associationExpression ::= "wasAssociatedWith" "(" optionalIdentifier aIdentifier ("," agIdentifierOrMarker "," eIdentifierOrMarker)? optionalAttributeValuePairs ")"

The following table summarizes how each constituent of a PROV-DM Association maps to a PROV-N syntax element.

Association	Non-Terminal
id	optionalIdentifier
activity	aIdentifier
agent	agIdentifierOrMarker
plan	eIdentifierOrMarker

attributes	optionalAttributeValuePairs

PROV-N provides no dedicated syntax for Plan. Instead, a Plan MUST be expressed as an entityExpression with attribute `prov:type='prov:Plan'`.

Example 25

```
wasAssociatedWith(ex:assoc; ex:a1, ex:ag1, ex:e1, [ex:param1="a", ex:param2="b"])
```

Here `ex:assoc` is the optional attribution identifier, `ex:a1` is an activity identifier, `ex:ag1` is the optional identifier of the agent associated to the activity, `ex:e1` is the optional identifier of the plan used by the agent in the context of the activity, and `[ex:param1="a", ex:param2="b"]` is a list of optional attributes.

The remaining examples show cases where some of the optionals are omitted.

```
wasAssociatedWith(ex:a1, -, ex:e1)
wasAssociatedWith(ex:a1, ex:ag1)
wasAssociatedWith(ex:a1, ex:ag1, ex:e1)
wasAssociatedWith(ex:a1, ex:ag1, ex:e1, [ex:param1="a", ex:param2="b"])
wasAssociatedWith(ex:assoc; ex:a1, -, ex:e1)
```

Additional semantic rules (Section 3.7.5) apply to associationExpression.

Example 26

The following expression illustrates a plan.

```
entity(ex:pl1,   [ prov:type='prov:Plan' ])
```

3.3.4 Delegation

[21] delegationExpression ::= "actedOnBehalfOf" "(" optionalIdentifier agIdentifier "," agIdentifier ("," aIdentifierOrMarker)? optionalAttributeValuePairs ")"

The following table summarizes how each constituent of a PROV-DM Delegation maps to a PROV-N syntax element.

Delegation	Non-Terminal
id	optionalIdentifier
delegate	agIdentifier
responsible	agIdentifier
activity	aIdentifierOrMarker
attributes	optionalAttributeValuePairs

Example 27

```
actedOnBehalfOf(ex:del1; ex:ag2, ex:ag1, ex:a, [prov:type="contract"])
```

Here `ex:del1` is the optional delegation identifier, `ex:ag2` is the identifier for the delegate agent, `ex:ag1` is the identifier of the responsible agent, `ex:a` is the optional identifier of the activity for which the delegation link holds, and `[prov:type="contract"]` is a list of optional attributes.

The remaining examples show cases where some of the optionals are omitted.

```
actedOnBehalfOf(ex:ag1, ex:ag2)
actedOnBehalfOf(ex:ag1, ex:ag2, ex:a)
actedOnBehalfOf(ex:ag1, ex:ag2, -, [prov:type="delegation"])
actedOnBehalfOf(ex:ag2, ex:ag3, ex:a, [prov:type="contract"])
actedOnBehalfOf(ex:del1; ex:ag2, ex:ag3, ex:a, [prov:type="contract"])
```

3.3.5 Influence

[22] `influenceExpression` ::= `"wasInfluencedBy" "(" optionalIdentifier eIdentifier "," eIdentifier optionalAttributeValuePairs ")"`

The following table summarizes how each constituent of a PROV-DM Influence maps to a PROV-N syntax element.

Influence	Non-Terminal
id	`optionalIdentifier`
influencee	`eIdentifier`
influencer	`eIdentifier`
attributes	`optionalAttributeValuePairs`

Example 28

```
wasInfluencedBy(ex:infl1;e2,e1,[ex:param="a"])
```

Here, `ex:infl1` is the optional influence identifier, `ex:e2` is an entity identifier, `ex:e1` is the identifier for an ancestor entity that `ex:e2` is influenced by, and `[ex:param="a"]` is the optional set of attributes.

The remaining examples show cases where some of the optionals are omitted.

```
wasInfluencedBy(ex:e2,ex:e1)
wasInfluencedBy(ex:e2,ex:e1,[ex:param="a"])
wasInfluencedBy(ex:infl1; ex:e2,ex:e1)
```

3.4 Component 4: Bundles

3.4.1 Bundle Constructor

[23] `bundle` ::= `"bundle" identifier (namespaceDeclarations)? (expression)* "endBundle"`

Bundles cannot be nested. It is for this reason that a `bundle` is not defined as an `expression`, to prevent the occurrence of a `bundle` inside another `bundle`.

Each identifier occurring in a bundle, including the bundle identifier itself, MUST be interpreted with respect to the namespace declarations of that bundle, or if the identifier's prefix is not declared in the bundle, with respect to the namespace declarations in the document.

The following table summarizes how each constituent of a PROV-DM bundle maps to a PROV-N syntax element.

Bundle	Non-Terminal
id	`identifier`
descriptions	`expression`
namespaceDeclaration	`namespaceDeclarations`

Example 29

```
bundle ex:author-view
  prefix ex  <http://example.org/>
  agent(ex:Paolo,   [ prov:type='prov:Person' ])
  agent(ex:Simon,   [ prov:type='prov:Person' ])
  //...
endBundle
```

Here `ex:author-view` is the name of the bundle.

3.4.2 Bundle Type

When described, a Bundle MUST be expressed as an `entityExpression` with attribute `prov:type='prov:Bundle'`.

Example 30

The bundle of Example 29 can be referred to as an entity, and its provenance described.

```
entity(ex:author-view,   [ prov:type='prov:Bundle' ])
```

3.5 Component 5: Alternate Entities

3.5.1 Alternate

[24] `alternateExpression` ::= `"alternateOf" "(" eIdentifier "," eIdentifier ")"`

The following table summarizes how each constituent of a PROV-DM Alternate maps to a PROV-N syntax element.

Alternate	Non-Terminal
alternate1	`eIdentifier`
alternate2	`eIdentifier`

Example 31

```
alternateOf(tr:WD-prov-dm-20111215,ex:alternate-20111215)
```

Here `tr:WD-prov-dm-20111215` is alternate for `ex:alternate-20111215`.

3.5.2 Specialization

[25] `specializationExpression` ::= `"specializationOf" "(" eIdentifier "," eIdentifier ")"`

The following table summarizes how each constituent of a PROV-DM Specialization maps to a PROV-N syntax element.

Specialization	Non-Terminal
specificEntity	`eIdentifier`
generalEntity	`eIdentifier`

Example 32

```
specializationOf(tr:WD-prov-dm-20111215,tr:prov-dm)
```

Here `tr:WD-prov-dm-20111215` is a specialization of `tr:prov-dm`.

3.6 Component 6: Collections

3.6.1 Collection

PROV-N provides no dedicated syntax for Collection and EmptyCollection. Instead, a Collection or an EmptyCollection MUST be expressed as an `entityExpression` with attribute `prov:type='prov:Collection'`, or `prov:type='prov:EmptyCollection'`, respectively.

Example 33

The following two expressions are about a collection and an empty collection, respectively.

```
entity(ex:col1,   [ prov:type='prov:Collection' ])
entity(ex:col2,   [ prov:type='prov:EmptyCollection' ])
```

3.6.2 Membership

[26] `membershipExpression` ::= `"hadMember" "(" cIdentifier "," eIdentifier ")"`

The following table summarizes how each constituent of a PROV-DM Membership maps to a PROV-N syntax element.

Membership	Non-Terminal
collection	cIdentifier
entity	eIdentifier

Example 34

```
hadMember(ex:c, ex:e1)    // ex:c contained ex:e1
hadMember(ex:c, ex:e2)    // ex:c contained ex:e2
```

Here `ex:c` is the identifier for the collection whose membership is stated, and `ex:e1` and `ex:e2` are the entities that are members of collection `ex:c`.

3.7 Further Expressions

This section defines further expressions of PROV-N.

3.7.1 Identifier

Various kinds of identifiers are used in productions.

```
[27] eIdentifier           ::= identifier
[28] aIdentifier           ::= identifier
[29] agIdentifier          ::= identifier
[30] gIdentifier           ::= identifier
[31] uIdentifier           ::= identifier
[32] cIdentifier           ::= identifier
[33] eIdentifierOrMarker   ::= ( eIdentifier | "-" )
[34] aIdentifierOrMarker   ::= ( aIdentifier | "-" )
[35] agIdentifierOrMarker  ::= ( agIdentifier | "-" )
[36] gIdentifierOrMarker   ::= ( gIdentifier | "-" )
[37] uIdentifierOrMarker   ::= ( uIdentifier | "-" )
[38] identifier            ::= QUALIFIED_NAME
```

A ***qualified name*** is a name subject to namespace interpretation. It consists of a namespace, denoted by an optional prefix, and a local name. The PROV data model stipulates that a qualified name can be mapped to an IRI by concatenating the IRI associated with the prefix and the local part. This section provides the exact details of this procedure for qualified names defined by PROV-N.

A qualified name's prefix is OPTIONAL. If a prefix occurs in a qualified name, the prefix MUST refer to a namespace declared in a namespace declaration. In the absence of prefix, the qualified name belongs to the default namespace.

A PROV-N qualified name (production QUALIFIED_NAME) has a more permissive syntax than XML's QName [XML-NAMES] and SPARQL PrefixedName [RDF-SPARQL-QUERY]. A QUALIFIED_NAME consists of a prefix and a local part. Prefixes follow the production PN_PREFIX defined by SPARQL [RDF-SPARQL-QUERY]. Local parts have to be conformant with PN_LOCAL, which extends the original SPARQL PN_LOCAL definition by allowing further characters (see PN_CHARS_OTHERS):

- an extra set of characters commonly encountered in IRIs;
- %-escaped characters (see PERCENT) to be interpreted as per Section 3.1. Mapping of IRIs to URIs in [RFC3987];
- and \-escaped characters (see PN_CHARS_ESC).

Given that '=' (equal), "'" (single quote), '(' (left bracket), ')' (right bracket), ',' (comma), ':' (colon), ';' (semi-colon), '"' (double quote), '[' (left square bracket), ']' (right square bracket) are used by the PROV notation as delimiters, they are not allowed in local parts. Instead, among those characters, those that are permitted in SPARQL IRI_REF are also allowed in PN_LOCAL if they are escaped by the '\' (backslash character) as per production PN_CHARS_ESC. Furthermore, '.' (dot), ':' (colon), '-' (hyphen) can also be \-escaped.

A PROV-N qualified name QUALIFIED_NAME can be mapped to a valid IRI [RFC3987] by concatenating the namespace denoted its local name PN_PREFIX to the local name PN_LOCAL, whose \-escaped characters have been unescaped by dropping the character '\' (backslash).

```
[52] <QUALIFIED_NAME>     ::=  ( PN_PREFIX ":" )? PN_LOCAL
                               | PN_PREFIX ":"
[53] <PN_LOCAL>           ::=  ( PN_CHARS_U | [0-9] | PN_CHARS_OTHERS ) ( ( PN_CHARS | "." | PN_CHARS_OTHERS )*
                               ( PN_CHARS | PN_CHARS_OTHERS ) )?
[54] <PN_CHARS_OTHERS>    ::=  "/"
                               | "@"
                               | "~"
                               | "&"
                               | "+"
                               | "*"
                               | "?"
                               | "#"
                               | "$"
                               | "!"
                               | PERCENT
                               | PN_CHARS_ESC
[55] <PN_CHARS_ESC>       ::=  "\" ( "=" | "'" | "(" | ")" | "," | "-" | ":" | ";" | "[" | "]" | "." )
[56] <PERCENT>            ::=  "%" HEX HEX
[57] <HEX>                ::=  [0-9]
                               | [A-F]
                               | [a-f]
```

Example 35

Examples of articles on the BBC Web site seen as entities.

```
document
  prefix bbc <http://www.bbc.co.uk/>
  prefix bbcNews <http://www.bbc.co.uk/news/>

  entity(bbc:)                           // bbc site itself
  entity(bbc:news/)                      // bbc news
  entity(bbc:news/world-asia-17507976)   // a given news article

  entity(bbcNews:)                       // an alternative way of referring to the bbc news site

endDocument
```

Example 36

Examples of entities with declared and default namespace.

```
document
  default <http://example.org/2/>
  prefix ex <http://example.org/1/>

  entity(ex:a)      //  corresponds to IRI http://example.org/1/a
  entity(ex:a/)     //  corresponds to IRI http://example.org/1/a/
  entity(ex:a/b)    //  corresponds to IRI http://example.org/1/a/b
  entity(b)         //  corresponds to IRI http://example.org/2/b
  entity(ex:1234)   //  corresponds to IRI http://example.org/1/1234
  entity(4567)      //  corresponds to IRI http://example.org/2/4567
  entity(c/)        //  corresponds to IRI http://example.org/2/c/
  entity(ex:/)      //  corresponds to IRI http://example.org/1//
endDocument
```

Example 37

Examples of \-escaped characters.

```
document
  prefix ex <http://example.org/>
  default <http://example.org/default>

  entity(ex:foo?a\=1)            // corresponds to IRI http://example.org/foo?a=1
  entity(ex:\-)                  // corresponds to IRI http://example.org/-
  entity(ex:?fred\=fish%20soup)  // corresponds to IRI http://example.org/?fred=fish%20soup

  used(-;a1,e1,-)                // identifier not specified for usage
  used(\-;a1,e1,-)               // usage identifier corresponds to http://example.org/default-
endDocument
```

Note: The productions for the terminals QUALIFIED_NAME and PN_PREFIX are conflicting. Indeed, for a tokenizer operating independently of the parse tree, `abc` matches both QUALIFIED_NAME and PN_PREFIX. In the context of a namespaceDeclaration, a tokenizer should give preference to the production PN_PREFIX.

3.7.2 Attribute

```
[39] attribute ::= QUALIFIED_NAME
```

The reserved attributes in the PROV namespace are the following. Their meaning is explained by [PROV-DM] (see Section 5.7.2: Attribute).

1. prov:label
2. prov:location
3. prov:role
4. prov:type
5. prov:value

3.7.3 Literal

```
[40] literal                  ::= typedLiteral
                                | convenienceNotation
[41] typedLiteral             ::= STRING_LITERAL "%%" datatype
[42] datatype                 ::= QUALIFIED_NAME
[43] convenienceNotation      ::= STRING_LITERAL (LANGTAG)?
                                | INT_LITERAL
                                | QUALIFIED_NAME_LITERAL
[58] <STRING_LITERAL>         ::= STRING_LITERAL2
                                | STRING_LITERAL_LONG2
[60] <INT_LITERAL>            ::= ("-")? (DIGIT)+
[62] <DIGIT>                  ::= [0-9]
[61] <QUALIFIED_NAME_LITERAL> ::= "'" QUALIFIED_NAME "'"
```

In production datatype, the QUALIFIED_NAME is used to denote a PROV data type [PROV-DM].

The non terminals STRING_LITERAL, INT_LITERAL, and QUALIFIED_NAME_LITERAL are syntactic sugar for quoted strings with datatype xsd:string, xsd:int, and prov:QUALIFIED_NAME respectively.

In particular, a Literal may be an IRI-typed string (with datatype `xsd:anyURI`); such IRI has no specific interpretation in the context of PROV.

Note: The productions for terminals QUALIFIED_NAME and INT_LITERAL are conflicting. Indeed, for a tokenizer operating independently of the parse tree, `1234` matches both INT_LITERAL and QUALIFIED_NAME (local name without prefix). In the context of a convenienceNotation, a tokenizer should give preference to the production INT_LITERAL.

Example 38

The following examples illustrate convenience notations.

The two following expressions are strings; if datatype is not specified, it is `xsd:string`.

```
"abc" %% xsd:string
"abc"
```

The two following expressions are integers. For convenience, numbers, expressed as digits optionally preceded by a minus sign, can occur without quotes.

```
"1234" %% xsd:integer
1234
"-1234" %% xsd:integer
-1234
```

The two following expressions are qualified names. Values of type qualified name can be conveniently expressed within single quotes.

```
"ex:value" %% prov:QUALIFIED_NAME
'ex:value'
```

Example 39

The following examples respectively are the string "abc", the string (in French) "bonjour", the integer number 1, and the IRI "http://example.org/foo".

```
"abc"
"bonjour"@fr
"1" %% xsd:integer
"http://example.org/foo" %% xsd:anyURI
```

The following examples respectively are the floating point number 1.01 and the boolean true.

```
"1.01" %% xsd:float
"true" %% xsd:boolean
```

3.7.3.1 Reserved Type Values

The reserved type values in the PROV namespace are the following. Their meaning is defined [PROV-DM] (see Section 5.7.2.4: prov:type).

1. `prov:Bundle`
2. `prov:Collection`
3. `prov:EmptyCollection`
4. `prov:Organization`
5. `prov:Person`
6. `prov:Plan`
7. `prov:PrimarySource`
8. `prov:Quotation`
9. `prov:Revision`
10. `prov:SoftwareAgent`

Example 40

The agent `ag` is a person (type: `prov:Person`), whereas the entity `pl` is a plan (type: `prov:Plan`).

```
agent(ag,  [ prov:type='prov:Person' ])
entity(pl, [ prov:type='prov:Plan' ])
```

3.7.3.2 Time Values

Time instants are defined according to xsd:dateTime [XMLSCHEMA11-2].

[44] `time` ::= `DATETIME`

Example 41

The third argument in the following usage expression is a time instance, namely 4pm on 2011-11-16.

```
used(ex:act2, ar3:0111, 2011-11-16T16:00:00)
```

3.7.4 Namespace Declaration

[45]	`namespaceDeclarations`	::=	`(` `defaultNamespaceDeclaration` | `namespaceDeclaration` `)` `(namespaceDeclaration)*`
[46]	`namespaceDeclaration`	::=	`"prefix"` `PN_PREFIX` `namespace`
[47]	`defaultNamespaceDeclaration`	::=	`"default"` `IRI_REF`
[48]	`namespace`	::=	`IRI_REF`

A `namespaceDeclaration` consists of a binding between a prefix and a namespace. Every qualified name with this prefix in the scope of this declaration belongs to this namespace. A `defaultNamespaceDeclaration` consists of a namespace. Every qualified name without prefix in the scope of this declaration belongs to this namespace. Scope of a prefix-namespace declaration is specified as follows:

- The scope of a prefix-namespace declaration directly occurring in a `bundle` is the `bundle` itself.
- The scope of a prefix-namespace declaration directly occurring in a `document` is the `document` including the `bundles` it contains but excluding those `bundles` that re-declare this prefix.

A set of namespace declarations `namespaceDeclarations` MUST NOT re-declare the same prefix.

A set of namespace declarations `namespaceDeclarations` occurring in a bundle MAY re-declare a prefix declared in a surrounding document.

A namespace declaration `namespaceDeclaration` MUST NOT declare prefixes `prov` and `xsd` (see Table 1 for their IRI).

Example 42

The following example declares three namespaces, one default, and two with explicit prefixes `ex1` and `ex2`.

```
document
  default <http://example.org/0/>
  prefix ex1 <http://example.org/1/>
  prefix ex2 <http://example.org/2/>
...
endDocument
```

Example 43

In the following example, a document declares a default namespace and the occurrence of `e001` directly occurring in the document refers to that namespace. A nested bundle also declares a default namespace, but with a different IRI. In that bundle, the occurrences of `e001`, including the bundle name, refer to the latest default namespace.

```
document
  default <http://example.org/1/>
  entity(e001)                              // IRI: http://example.org/1/e001

  bundle e001                               // IRI: http://example.org/2/e001
    default <http://example.org/2/>
    entity(e001)                            // IRI: http://example.org/2/e001
  endBundle
endDocument
```

Example 44

In the following example, a document declares a namespace with prefix `ex` and the occurrence of `ex:e001` directly occurring in the document refers to that namespace. In a nested bundle, the occurrence of `ex:e001` also refers to the same namespace.

```
document
  prefix ex <http://example.org/1/>
  entity(ex:e001)                             // IRI: http://example.org/1/e001

  bundle b
    entity(ex:001)                            // IRI: http://example.org/1/e001
  endBundle
endDocument
```

3.7.5 Summary of additional semantic rules

Some of the grammar productions allow for expressions that are syntactically correct, and yet according to [PROV-DM] they are not acceptable, because additional semantic rules are defined for those expressions. The following table provides a summary of such expressions along with examples of syntactically correct but unacceptable forms, and the additional semantic rules.

Table 2: Summary of additional semantic rules for grammar productions

Production	Examples of syntactically correct expressions	Additional semantic rule
Generation expression	`wasGeneratedBy(e2, -, -)` `wasGeneratedBy(-; e2, -, -)`	At least one of *id*, *activity*, *time*, and *attributes* MUST be present.
Usage expression	`used(a2, -, -)`	At least one of *id*, *entity*, *time*, and *attributes* MUST be

	`used(-; a2, -, -)`	present
Start expression	`wasStartedBy(e2, -, -, -)` `wasStartedBy(-; e2, -, -, -)`	At least one of *id*, *trigger*, *starter*, *time*, and *attributes* MUST be present
End expression	`wasEndedBy(e2, -, -, -)` `wasEndedBy(-; e2, -, -, -)`	At least one of *id*, *trigger*, *ender*, *time*, and *attributes* MUST be present
Invalidation expression	`wasInvalidatedBy(e2, -, -)` `wasInvalidatedBy(-; e2, -, -)`	At least one of *id*, *activity*, *time*, and *attributes* MUST be present
Association expression	`wasAssociatedWith(a, -, -)` `wasAssociatedWith(-; a, -, -)`	At least one of *id*, *agent*, *plan*, and *attributes* MUST be present

4. Document

A ***document*** is a house-keeping construct of PROV-N capable of packaging up PROV-N expressions and namespace declarations. A document forms a self-contained package of provenance descriptions for the purpose of *exchanging* them. A document may be used to package up PROV-N expressions in response to a request for the provenance of something ([PROV-AQ]).

Given its status of house-keeping construct for the purpose of exchanging provenance expressions, a document is not defined as a PROV-N expression (production `expression`).

A document's text matches the `document` production.

```
[1] document ::= "document" (namespaceDeclarations)? (expression)* (bundle)* "endDocument"
```

A document contains:

- *namespaceDeclarations*: a set of namespace declarations `namespaceDeclarations`, declaring namespaces and associated prefixes, which can be used in attributes and identifiers occurring inside *expressions* or *bundles*;
- *expressions*: a set of expressions, each matching `expression`;
- *bundles*: a set of expressions, each matching `bundle`.

Thus, bundles MAY occur inside a document, but do not appear inside other bundles.

Example 45

The following document contains expressions related to the provenance of entity `e2`.

```
document
  default <http://anotherexample.org/>
  prefix ex <http://example.org/>

  entity(e2, [ prov:type="File", ex:path="/shared/crime.txt", ex:creator="Alice",
               ex:content="There was a lot of crime in London last month."])
  activity(a1, 2011-11-16T16:05:00, -, [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
```

This container could, for instance, be returned as the result of a query to a provenance store for the provenance of entity `e2` [PROV-AQ].

5. Extensibility

The PROV data model is extensible by means of attributes `prov:type` and `prov:role` allowing subtyping of expressions. For some applications, novel syntax may also be convenient. Hence, the normative requirements are as follow.

- PROV-N compliant parsers MUST be able to parse expressions matching the `extensibilityExpression` production defined below.
- As PROV provides no definition for these expressions, PROV compliant implementations MAY ignore these expressions.
- Extensions to PROV and PROV-N MAY specify more specific productions and interpretations for these

expressions, which applications MAY elect to follow.

```
[49]  extensibilityExpression  ::=  QUALIFIED_NAME "(" optionalIdentifier extensibilityArgument ( "," extensibilityArgument )* optionalAttributeValuePairs ")"
[50]  extensibilityArgument    ::=  ( identifierOrMarker | literal | time | extensibilityExpression | extensibilityTuple )
[51]  extensibilityTuple       ::=  "{" extensibilityArgument ( "," extensibilityArgument )* "}"
                                  | "(" extensibilityArgument ( "," extensibilityArgument )* ")"
```

Expressions compatible with the `extensibilityExpression` production follow a general form of functional syntax, in which the predicate MUST be a `qualifiedName` with a non-empty `prefix`.

Example 46

Collections are sets of entities, whose membership can be expressed using the `hadMember` relation. The following example shows how one can express membership for *dictionaries*, an illustrative extension of Collections consisting of sets of key-entity pairs, where a key is a `literal`. The notation is a variation of that used for Collections membership, allowing multiple member elements to be declared, and in which the elements are pairs. The name of the relation is qualified with the extension-specific namespace `http://example.org/dictionaries`.

```
prefix dictExt <http://example.org/dictionaries#>
dictExt:hadMembers(mId; d, {("k1",e1), ("k2",e2), ("k3",e3)}, [])
```

Note that the generic `extensibilityExpression` production above allows for alternative notations to be used for expressing membership, if the designers of the extensions so desire. Here is an alternate syntax that is consistent with the productions:

```
prefix dictExt <http://example.org/dictionaries#>
dictExt:hadMembers(mid; d, dictExt:set(dictExt:pair("k1",e1),
                                       dictExt:pair("k2",e2),
                                       dictExt:pair("k3",e3)),
                           [dictExt:uniqueKeys="true"])
```

6. Media Type

The media type of PROV-N is `text/provenance-notation`. The content encoding of PROV-N content is UTF-8.

Contact:
Ivan Herman
See also:
How to Register a Media Type for a W3C Specification
Internet Media Type registration, consistency of use
TAG Finding 3 June 2002 (Revised 4 September 2002)

The Internet Media Type / MIME Type for PROV-N is "text/provenance-notation".

It is recommended that PROV-N files have the extension ".provn" (all lowercase) on all platforms.

It is recommended that PROV-N files stored on Macintosh HFS file systems be given a file type of "TEXT".

The information that follows has been registered with IANA.

Type name:
text
Subtype name:
provenance-notation
Required parameters:
`charset` — the value of `charset` MUST always be `UTF-8`.
Optional parameters:
None
Encoding considerations: 8bit
The syntax of PROV-N is expressed over code points in Unicode [UNICODE]. The encoding is always UTF-8 [UTF-8].
Unicode code points may also be expressed using an \uXXXX (U+0 to U+FFFF) or \UXXXXXXXX syntax (for U+10000 onwards) where X is a hexadecimal digit [0-9A-F]
Security considerations:

PROV-N is a general-purpose language for describing the provenance of things; applications may evaluate given data to infer more descriptions or to dereference URIs, invoking the security considerations of the scheme for that URI. Note in particular, the privacy issues in [RFC3023] section 10 for HTTP URIs. Data obtained from an inaccurate or malicious data source may lead to inaccurate or misleading conclusions, as well as the dereferencing of unintended URIs. Care must be taken to align the trust in consulted resources with the sensitivity of the intended use of the data.

PROV-N is used to express the provenance of arbitrary application data; security considerations will vary by domain of use. Security tools and protocols applicable to text (e.g. PGP encryption, MD5 sum validation, password-protected compression) may also be used on PROV-N documents. Security/privacy protocols must be imposed which reflect the sensitivity of the embedded information.

PROV-N can express data which is presented to the user, for example, by means of label attributes. Application rendering strings retrieved from untrusted PROV-N documents must ensure that malignant strings may not be used to mislead the reader. The security considerations in the media type registration for XML ([RFC3023] section 10) provide additional guidance around the expression of arbitrary data and markup.

PROV-N is a language for describing the provenance of things, and therefore a PROV-N document is metadata for other resources. Untrusted PROV-N documents may mislead its consumers by indicating that a third-party resource has a reputable lineage, when it has not. Provenance of PROV-N document should be sought.

PROV-N uses qualified names mappable to IRIs as term identifiers. Applications interpreting data expressed in PROV-N should address the security issues of Internationalized Resource Identifiers (IRIs) [RFC3987] Section 8, as well as Uniform Resource Identifier (URI): Generic Syntax [RFC3986] Section 7.

Multiple IRIs may have the same appearance. Characters in different scripts may look similar (a Cyrillic "o" may appear similar to a Latin "o"). A character followed by combining characters may have the same visual representation as another character (LATIN SMALL LETTER E followed by COMBINING ACUTE ACCENT has the same visual representation as LATIN SMALL LETTER E WITH ACUTE). Any person or application that is writing or interpreting data in PROV-N must take care to use the IRI that matches the intended semantics, and avoid IRIs that make look similar. Further information about matching of similar characters can be found in Unicode Security Considerations [UNISEC] and Internationalized Resource Identifiers (IRIs) [RFC3987] Section 8.

PROV-N offers an extensibility mechanism, which in turn may introduce additional security considerations. For example, predicates in extensibility expressions use qualified names, mappable to IRIs, and appropriate security considerations for IRIs apply too.

Interoperability considerations:

There are no known interoperability issues.

Published specification:

PROV-N: The Provenance Notation, Moreau, Missier, (eds), Cheney, Soiland-Reyes http://www.w3.org/TR/prov-n/, 2012.

Applications which use this media type:

It may be used by any application for publishing provenance information. This format is designed to be a human-readable form of provenance.

Additional information:

Magic number(s):

PROV-N documents may have the strings 'document' near the beginning of the document.

File extension(s):

".provn"

Base URI:

There are no constructs in the PROV-N Syntax to change the Base IRI.

Macintosh file type code(s):

"TEXT"

Person & email address to contact for further information:

Ivan Herman, ivan@w3.org

Intended usage:

COMMON

Restrictions on usage:

None

Author/Change controller:

The PROV-N specification is the product of the World Wide Web Consortium's Provenance Working Group. The W3C has change control over this specification.

A. Change Log

This section is non-normative.

A.1 Changes since Proposed Recommendation

This section is non-normative.

- Changed the status of this document section.
- Changed all URLs to PROV documents.
- Updated URL to IANA registered text/provenance-notation mime type.
- Added ex: prefix to some identifiers in examples.
- The mapping table for the bundle production incorrectly referred to optionalIdentifier, which was not present in the production. It was replaced by identifier.
- Replaced "are optional attributes" by the grammatically more correct "is a list of optional attributes".

A.2 Changes since Candidate Recommendation

This section is non-normative.

- Checked that all fragments resolved.
- Changed the status of this document section: added new documents to the PROV Family of Document, and removed the how to read section, referring instead to PROV-OVERVIEW.
- Changed all URLs to PROV documents.
- Clarified a sentence about the (non) nesting of bundles.
- Added html link to provenance.
- Updated security consideration for extensibility feature of PROV-N following IANA feedback.

A.3 Changes since last call

This section is non-normative.

Please see the Responses to Public Comments on the Last Call Working Draft for more details about the justification of these changes.

- ISSUE-543: Uses key-entity pair terminology for example 43.
- ISSUE-538: Rephrasing of text in example 10.
- ISSUE-547: Aligned terminology and productions with prov-constraints document, renaming 'toplevel bundle' to 'document', and renaming 'named bundle' to 'bundle'.
- ISSUE-496: For each prov-n term, added a cross-reference to corresponding prov-dm term (see table following each production). Rephrased text related to scope of a namespace declaration. Added an informative file with all productions.
- ISSUE-573: Updated Media Type section.
- ISSUE-589: Updated scoping rule for prefix-namespace declaration. Also refrained from saying bundles are self-contained.
- Moved feature at Risk mention into a separate note.
- Editorial's pass pre-publication.

B. Acknowledgements

This section is non-normative.

This document has been produced by the Provenance Working Group, and its contents reflect extensive discussion within the Working Group as a whole. The editors extend special thanks to Sandro Hawke (W3C/MIT) and Ivan Herman (W3C/ERCIM), W3C contacts for the Provenance Working Group.

The editors acknowledge valuable contributions from the following: Tom Baker, David Booth, Robert Freimuth, Satrajit Ghosh, Ralph Hodgson, Renato Iannella, Jacek Kopecky, James Leigh, Jacco van Ossenbruggen, Alan Ruttenberg, Reza Samavi, and Antoine Zimmermann.

Members of the Provenance Working Group at the time of publication of this document were: Ilkay Altintas (Invited expert), Reza B'Far (Oracle Corporation), Khalid Belhajjame (University of Manchester), James Cheney (University of Edinburgh, School of Informatics), Sam Coppens (iMinds - Ghent University), David Corsar (University of Aberdeen, Computing Science), Stephen Cresswell (The National Archives), Tom De Nies (iMinds - Ghent University), Helena Deus (DERI Galway at the National University of Ireland, Galway, Ireland), Simon Dobson (Invited expert), Martin Doerr (Foundation for Research and Technology - Hellas(FORTH)), Kai Eckert (Invited expert), Jean-Pierre EVAIN (European Broadcasting Union, EBU-UER), James Frew (Invited expert), Irini Fundulaki (Foundation for Research and Technology - Hellas(FORTH)), Daniel Garijo (Universidad Politécnica de Madrid), Yolanda Gil (Invited expert), Ryan Golden (Oracle Corporation), Paul Groth (Vrije Universiteit), Olaf Hartig (Invited expert), David Hau (National Cancer Institute, NCI), Sandro Hawke (W3C/MIT), Jörn Hees (German Research Center for Artificial Intelligence (DFKI) Gmbh), Ivan Herman, (W3C/ERCIM), Ralph Hodgson (TopQuadrant), Hook Hua (Invited expert), Trung Dong Huynh (University of Southampton), Graham Klyne (University of Oxford), Michael Lang (Revelytix, Inc.), Timothy Lebo (Rensselaer Polytechnic Institute), James McCusker (Rensselaer Polytechnic Institute), Deborah

McGuinness (Rensselaer Polytechnic Institute), Simon Miles (Invited expert), Paolo Missier (School of Computing Science, Newcastle university), Luc Moreau (University of Southampton), James Myers (Rensselaer Polytechnic Institute), Vinh Nguyen (Wright State University), Edoardo Pignotti (University of Aberdeen, Computing Science), Paulo da Silva Pinheiro (Rensselaer Polytechnic Institute), Carl Reed (Open Geospatial Consortium), Adam Retter (Invited Expert), Christine Runnegar (Invited expert), Satya Sahoo (Invited expert), David Schaengold (Revelytix, Inc.), Daniel Schutzer (FSTC, Financial Services Technology Consortium), Yogesh Simmhan (Invited expert), Stian Soiland-Reyes (University of Manchester), Eric Stephan (Pacific Northwest National Laboratory), Linda Stewart (The National Archives), Ed Summers (Library of Congress), Maria Theodoridou (Foundation for Research and Technology - Hellas(FORTH)), Ted Thibodeau (OpenLink Software Inc.), Curt Tilmes (National Aeronautics and Space Administration), Craig Trim (IBM Corporation), Stephan Zednik (Rensselaer Polytechnic Institute), Jun Zhao (University of Oxford), Yuting Zhao (University of Aberdeen, Computing Science).

C. References

C.1 Normative references

[PROV-CONSTRAINTS]
James Cheney; Paolo Missier; Luc Moreau; eds. *Constraints of the PROV Data Model*. 30 April 2013, W3C Recommendation. URL: http://www.w3.org/TR/2013/REC-prov-constraints-20130430/

[PROV-DM]
Luc Moreau; Paolo Missier; eds. *PROV-DM: The PROV Data Model*. 30 April 2013, W3C Recommendation. URL: http://www.w3.org/TR/2013/REC-prov-dm-20130430/

[PROV-O]
Timothy Lebo; Satya Sahoo; Deborah McGuinness; eds. *PROV-O: The PROV Ontology*. 30 April 2013, W3C Recommendation. URL: http://www.w3.org/TR/2013/REC-prov-o-20130430/

[RDF-SPARQL-QUERY]
Andy Seaborne; Eric Prud'hommeaux. *SPARQL Query Language for RDF*. 15 January 2008. W3C Recommendation. URL: http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115

[RFC2119]
S. Bradner. *Key words for use in RFCs to Indicate Requirement Levels.* March 1997. Internet RFC 2119. URL: http://www.ietf.org/rfc/rfc2119.txt

[RFC3023]
M. Murata; S. St.Laurent; D. Kohn. *XML Media Types (RFC 3023)*. January 2001. RFC. URL: http://www.ietf.org/rfc/rfc3023.txt

[RFC3986]
T. Berners-Lee; R. Fielding; L. Masinter. *Uniform Resource Identifier (URI): Generic Syntax (RFC 3986)*. January 2005. RFC. URL: http://www.ietf.org/rfc/rfc3986.txt

[RFC3987]
M. Dürst; M. Suignard. *Internationalized Resource Identifiers (IRIs) (RFC 3987)*. January 2005. RFC. URL: http://www.ietf.org/rfc/rfc3987.txt

[UNICODE]
The Unicode Consortium. *The Unicode Standard.* Defined by: The Unicode Standard, Version 6.2.0, (Mountain View, CA: The Unicode Consortium, 2012. ISBN 978-1-936213-07-8) , as updated from time to time by the publication of new versions URL: http://www.unicode.org/unicode/standard/versions/enumeratedversions.html

[UTF-8]
F. Yergeau. *UTF-8, a transformation format of ISO 10646*. IETF RFC 3629. November 2003. URL: http://www.ietf.org/rfc/rfc3629.txt

[XML-NAMES]
Richard Tobin et al. *Namespaces in XML 1.0 (Third Edition)*. 8 December 2009. W3C Recommendation. URL: http://www.w3.org/TR/2009/REC-xml-names-20091208/

[XML11]
Eve Maler et al. *Extensible Markup Language (XML) 1.1 (Second Edition)*. 16 August 2006. W3C Recommendation. URL: http://www.w3.org/TR/2006/REC-xml11-20060816

[XMLSCHEMA11-2]
Henry S. Thompson et al. *W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes*. 5 April 2012. W3C Recommendation. URL: http://www.w3.org/TR/2012/REC-xmlschema11-2-20120405/

C.2 Informative references

[PROV-AQ]
Graham Klyne; Paul Groth; eds. *Provenance Access and Query*. 30 April 2013, W3C Note. URL: http://www.w3.org/TR/2013/NOTE-prov-aq-20130430/

[PROV-DC]
Daniel Garijo; Kai Eckert; eds. *Dublin Core to PROV Mapping*. 30 April 2013, W3C Note. URL:

http://www.w3.org/TR/2013/NOTE-prov-dc-20130430/

[PROV-DICTIONARY]

Tom De Nies; Sam Coppens; eds. *PROV Dictionary: Modeling Provenance for Dictionary Data Structures*. 30 April 2013, W3C Note. URL: http://www.w3.org/TR/2013/NOTE-prov-dictionary-20130430/

[PROV-LINKS]

Luc Moreau; Timothy Lebo; eds. *Linking Across Provenance Bundles*. 30 April 2013, W3C Note. URL: http://www.w3.org/TR/2013/NOTE-prov-links-20130430/

[PROV-OVERVIEW]

Paul Groth; Luc Moreau; eds. *PROV-OVERVIEW: An Overview of the PROV Family of Documents*. 30 April 2013, W3C Note. URL: http://www.w3.org/TR/2013/NOTE-prov-overview-20130430/

[PROV-PRIMER]

Yolanda Gil; Simon Miles; eds. *PROV Model Primer*. 30 April 2013, W3C Note. URL: http://www.w3.org/TR/2013/NOTE-prov-primer-20130430/

[PROV-RDF]

James Cheney *PROV-RDF Mapping* 2012, Working in Progress. URL: http://www.w3.org/2011/prov/wiki/ProvRDF

[PROV-SEM]

James Cheney; ed. *Semantics of the PROV Data Model*. 30 April 2013, W3C Note. URL: http://www.w3.org/TR/2013/NOTE-prov-sem-20130430.

[PROV-XML]

Hook Hua; Curt Tilmes; Stephan Zednik; eds. *PROV-XML: The PROV XML Schema*. 30 April 2013, W3C Note. URL: http://www.w3.org/TR/2013/NOTE-prov-xml-20130430/

[UNISEC]

Mark Davis; Michel Suignard. *Unicode Security Considerations*. 4 August 2010. URL: http://www.unicode.org/reports/tr36/