Why URI Declarations?
A comparison of architectural approaches

David Booth
HP Software
dbooth@hp.com

Latest version of this document: http://dbooth.org/2008/irsw/

Abstract. When a Semantic Web application encounters a new URI in an RDF statement, how should it determine what resource that URI is intended to denote, and learn more about it? Since assertions are the currency of Semantic Web applications, in practical terms this question can be viewed as asking: What additional assertions should be used if the application wishes to learn more about the URI's denoted resource, and what mechanism should be used to find those assertions? This paper compares two architectural approaches from this perspective: one based on the notion of *URI declarations*, the other based on a marketplace of *competing definitions*. It argues that the URI declarations approach offers more desirable architectural characteristics for the Semantic Web, largely because it reduces URI collision.

Key words: Semantic Web, RDF, identity, URI declaration, URI definition

1 Introduction

"When I *use a word it means just what I choose it to mean -- neither more nor less."*
-- Humpty Dumpty, in Lewis Carroll's Through the Looking Glass [11]

The goal of the Semantic Web is to enable data from multiple sources to be readily combined based on common URIs. Various Semantic Web practitioners have advocated the practice of serving a set of assertions that should be accessible when a URI (or its racine -- the part before the fragment identifier -- if it contains a fragment identifier) is dereferenced. For example, Dan Connelly recommends [1]:

> "1. To mint a term in the community, choose a URI of the form *doc#id* and publish at *doc* some information that motivates others to use the term in a manner that is consistent with your intended meaning(s).
> 2. Use of a URI of the form. *doc#id* implies agreement to information published at *doc*."

The term "URI declaration" was coined [2] to help crystalize, explain and promote this practice (though extended to also accommodate hashless 303-URIs [3]). However, this practice has not yet been universally accepted in the Semantic Web community. In particular, a less constrained architectural approach, which we will call the "competing definitions" approach, is sometimes used.

To compare the merits of these approaches, we will make certain assumptions about how Semantic Web applications use URIs to denote resources. For simplicity, we will restrict our attention to URIs that denote non-information resources [4], however the comparison could be extended to URIs that denote information resources. Furthermore, although the examples show hash URIs [3], the analysis applies equally to hashless 303-URIs [3].

2 The "meaning" of a URI as a set of assertions

Suppose a Semantic Web application reads an N3 [8] statement, *S1*, such as

```
:thermostat :adjust <http://alice.example#foo> .
```

involving a previously unknown URI, http://alice.example#foo, that is intended to indicate whether a thermostat should be adjusted up or down. In an RDF statement like this, the URI is treated as a name -- not merely as a literal string of characters -- for the denoted resource. In order to understand what the statement "means", the application needs to know what http://alice.example#foo "means", i.e., it needs to know what resource the URI was intended to denote.

What does this mean operationally to the application? The answer may of course depend on the application. But for convenience we will assume that the question of what the URI "means" or denotes can be viewed as asking: What set of assertions defines the "meaning" of that URI? In other words, what additional assertions should the application use in conjunction with this URI?

For convenience, we will call these *identifying assertions*, because presumably they serve to constrain the denoted resource's identity; however, we make no requirement that they actually do. Thus, from an application's perspective, the identity of the denoted resource is assumed to be solely determined by the set of identifying assertions that constrain it: if two set sets of identifying assertions are equivalent, then they indicate the same resource. Given this assumption, the application's task upon discovering this new URI is to locate the identifying assertions that constitute the appropriate definition for that URI.

So far so good. But suppose there are multiple sets of assertions available that all involve this URI, any of which might potentially be interpreted as "identifying assertions". Does it matter which set the application chooses? Clearly it does: if the application does not choose the set that the author of statement *S1* intended, then it may misinterpret the author's intent, perhaps adjusting the thermostat up instead of down. Less obviously, even if the sets differ only slightly -- one being more restrictive than the other -- the application may still have trouble if it chooses a different set than the author intended.

For example, if the identifying assertions for http://example#dbooth were taken to be only the following N3 [8] assertions (omitting the foaf: [10] namespace declaration):

```
<http://example#dbooth> foaf:name "David Booth" .
<http://example#dbooth> foaf:workplaceHomepage
"http://www.hp.com/" .
```

instead of the more restrictive set that was intended:

```
<http://example#dbooth> foaf:name "David Booth" .
<http://example#dbooth> foaf:workplaceHomepage
"http://www.hp.com/" .
<http://example#dbooth> foaf:mbox "dbooth@hp.com" .
```

then a statement involving <http://example#dbooth> would apply to *any* of the three people named "David Booth" who work for HP, which may not be what the statement author intended.

Conversely, if the application chooses a more restrictive set of identifying assertions than the statement author intended, then the application may produce absurd results or incur a logical contradiction when that statement is combined with other statements involving the same URI. For example, suppose an author writes the following statement, *S2*:

```
<http://example#dbooth> a :male .
```

and bases the statement on the following URI definition for http://example#dbooth>:

```
<http://example#dbooth> a :human .
<http://example#dbooth> :hasHairColor :gray .
```

If an application reading *S2* instead uses the following, more constraining URI definition:

```
<http://example#dbooth> a :human .
<http://example#dbooth> :hasHairColor :gray .
<http://example#dbooth> a :female .
```

then the application will erroneously conclude that <http://example#dbooth> is both :male and :female.

Therefore, we will make the further assumption that the application's task in determining the "meaning" of a URI is to locate the *specific* set of identifying assertions that the statement author intended: the *URI definition* relevant to that statement. In summary, we will assume: *When an application needs to determine the "meaning" of a URI used in a statement, this boils down to the task of locating and accepting the specific set of "identifying assertions" that the statement author intended: the "URI definition" corresponding to that use of the URI.*

3 The follow-your-nose algorithm

One obvious way the application might locate the intended URI definition is by "following its nose" from the URI. If the URI contains a fragment identifier -- a so called "hash URI" such as http://alice.example#foo -- this means dereferencing the *racine* (i.e., the part before the fragment identifier: http://alice.example) in search of a URI definition. If the URI does not contain a fragment identifier -- a "hashless" or "303 URI" -- it means dereferencing the URI itself to obtain a new URI, which in turn is dereferenced in search of a URI definition. For brevity, we will call the document obtained by this algorithm the **follow-your-nose document** (or **f-y-n document**), and the identifying assertions that it contains will be called the **follow-your-nose definition** (or **f-y-n definition**) of that URI.

The follow-your-nose algorithm is not the only way that an application might locate the URI definition. If there were a standardized, machine processable convention for the author to indicate where to find it, then that could be used. For example, if, in the same file as *S1*, the author also wrote:

```
   <http://alice.example#foo> rdfs:isDefinedBy
   <http://alice.example/ont> .
```

then a new architectural convention might stipulate that, to locate the URI definition for http://alice.example#foo that pertains to statements contained in the same document, the application should dereference http://alice.example/ont (instead of using the "follow your nose" algorithm). However, to date this use of rdfs:isDefinedBy [5] has not been standardized.

4 URI collision

URI collision [4] occurs when the same URI is used in different contexts to denote different resources. Since we are treating the "meaning" of a URI operationally as indicating what identifying assertions an application should use in conjunction with that URI, the problem of URI collision translates into the problem of having different sets of identifying assertions associated with the URI in different contexts -- alternate URI definitions. For example, if the author of statement *S1* intended the reader to use one URI definition, and the author of statement *S2* intended the reader to use a different definition of the same URI (indicating different identifying assertions), it would constitute a URI collision that would inhibit the ability of an application to use *S1* and *S2* together, because doing so could cause the application to make incorrect inferences or incur a logical contradiction. To avoid this problem, an application in this situation would have to treat *S1* and *S2* as though they were talking about different resources -- as in effect they would be -- thus undermining a key goal of the Semantic Web: the ability to easily combine data based on common URIs. In short, URI collisions are harmful, and in a Semantic Web application, URI collision is manifested as different statements using the same URI but requiring different URI definitions, i.e., different sets of identifying assertions.

Given the above assumptions of how URI meaning can be viewed in terms of sets of identifying assertions, we can now describe and compare two architectural approaches for specifying and using such assertions.

5 URI declarations versus competing definitions

Loosely, the *competing definitions* approach takes the view that all assertions are created equal, and it is up to the community or marketplace to decide which assertions become the prevailing definition of a particular URI. In contrast, the *URI declarations* approach is based on the guiding principle that *use of a URI implies agreement with its follow-your-nose definition*. Thus it takes the view that assertions are *not* created equal: some are special from the outset -- namely, the URI declaration's "core assertions" -- and should be consistently used as the URI's definition in all statements. Although this is the fundamental architectural difference between these approaches, in order to further explain how the URI declarations approach can work under various circumstances we will also make a few more assumptions about the architectural rules that they might involve, in terms of the obligations of the URI owner and those of a statement author wishing to use the URI to make statements.

5.1 The URI owner's obligations

In both **the URI declarations approach** and **the competing definitions approach** we will assume that the URI owner's obligations are the same:

> **Rule A.** A URI owner minting a new URI SHOULD [6] publish a follow-your-nose definition of that URI.

For example, when Alice mints http://alice.example#foo, she should publish its definition at http://alice.example.

Of course, there are many other best practice guidelines that might be recommended also, such as:

- The URI owner should make best efforts to separate "essential" properties [7] from other properties of the resource, which should be published in a separate document from the follow-your-nose definition.

- As a convenience to statement authors, the URI definition should include pointers (perhaps via rdfs:seeAlso) to other known sets of assertions about the resource that statement authors may find useful but are not part of the URI definition.

- The follow-your-nose definition should indicate known relationships between this URI and other URIs. (See comments about this in the "URI relationships" section below.)

- The f-y-n definition's change policy should be clearly indicated, and substantive changes should be avoided, and change can cause URI collision.

- The URI should be persistent [13], such as by use of a Persistent URL (PURL).

- Etc.

However, such additional guidelines are beyond the scope of this paper and can be better covered in other works.

5.2 The statement author's obligations

To be clear, when we speak of an author "using a URI" in a statement, unless otherwise indicated, we mean that the author is using the URI to denote a resource, as http://alice.example#foo does in statement S1 above -- *not* merely as a string literal. Also, before we describe the differences between the URI declarations approach and the competing definitions approach, we will assume that they both involve the following obligation:

> **Rule B:** If a statement is based on a URI definition other than the f-y-n definition, the statement author MUST indicate where that URI definition can be found. This will be called an **alternate URI definition**.

Although in some cases it may be feasible to include an alternate URI definition -- the identifying assertions themselves -- in the same document as the statement that requires it, this is unlikely to be practical in general. Hence, we will assume only that the *location* of the alternate URI definition is specified (as a URL).

The statement author's remaining obligations vary, depending on which of three cases the statement author believes the URI falls under:

Case 1 URI: Normal case.

In this case, the URI has a reasonable follow-your-nose definition, at least for its intended application domain. The identifying assertions in that URI definition might only be useful to, or usable by, some applications -- for example, they may contain approximations that are too imprecise for many applications -- but they are not clearly erroneous.

Case 2 URI: Newly minted URI with missing or erroneous f-y-n definition.
In this case, either a follow-your-nose definition is not available or it is clearly erroneous, and the URI does not have a URI definition that is already entrenched in the community.

Case 3 URI: Entrenched URI with missing or erroneous f-y-n definition.
In this case, the URI has a set of identifying assertions that have been widely accepted in the community, but a follow-your-nose definition is not available or it is clearly erroneous -- perhaps due to a clerical error, or because the domain was hijacked. In this case, the community's URI definition may need to take precedence over an erroneous f-y-n definition.

Under the **competing definitions approach** the statement author's remaining obligations are:

Rule C-CD:
1: A statement using a case 1 URI SHOULD be based on the URI's f-y-n definition.
2: A statement using a case 2 URI MAY be based on any URI definition.
3: A statement using a case 3 URI SHOULD be based on the URI's community-accepted definition.

Under the **URI declarations approach** the statement author's remaining obligations are:

Rule C-UD:
1: A statement using a case 1 URI MUST be based on the URI's f-y-n definition.
2: A statement SHOULD NOT use a case 2 URI.
3: A statement SHOULD NOT use a case 3 URI.

Prohibiting the statement author from using the URI does not prevent the author from saying what he'she wishes to say, it merely means that the author needs to use a different URI to do so. For example, in the case of an erroneous f-y-n definition, a statement author can, if desired, mint a new URI with a corrected URI definition, in which case, as mentioned under "URI owner's obligations", the new URI definition should indicate the new URI's relationship to the old URI.

The intent of rules C-UD-2 and C-UD-3 is to discourage the use of a newly minted URI that has a missing or obviously erroneous follow-your-nose document, thus encouraging the URI owner to turn his/her case 2 URI into a case 1 URI and prevent it from becoming a case 3 URI. One reason this is important is because a statement author considering the use of a URI may not be able to accurately determine whether that URI should fall under case 1, 2 or 3, particularly if the statement author is not very familiar with "the community".

However, rules C-UD-2 and C-UD-3 say "SHOULD NOT" instead of "MUST NOT" largely because:

- It seems reasonable to use a URI if its f-y-n definition is known but temporarily unavailable.

- If a URI is very entrenched in common usage -- to the point where its meaning is hard wired into applications, for example -- it may not be worth the cost of changing to a case 1 URI. However, if a case 3 URI is used then *only* the entrenched, community-accepted URI definition must be used, and ideally the location of the definition should be explicitly indicated, so that an application can verify that the URI is being used according to its entrenched definition.

The precise details of rules C-CD and C-UD could perhaps be improved, but they are good enough for this architectural comparision. The point is that the competing definitions approach willingly permits competing definitions for a URI. It is somewhat like saying: "When *I* use a URI it means just what *I* choose it to mean." In contrast, the URI declaration approach is more like saying: "When *I* use a URI it means just what *the URI owner* chose it to mean." Or, from the URI owner's perspective, it would be like saying: "When *I* mint a URI it means just what *I* choose it to mean."

The rationale behind the greater discretion offered by the competing definitions approach is that it permits the statement author to use an alternate URI definition that, at least in that author's view, is *better* than the follow-your-nose definition. The concommitent assumption is that if competing URI definitions are offered to the community, the community will eventually converge on a common URI definition for that URI, and this will be A Good Thing. In contrast, under the URI declarations approach, these same market or community forces are expected to operate on competing *URIs* (rather than on competing URI *definitions*), with the same kind of beneficial effect. But as explained below, the overall impact of these two architectural approaches is *not* equivalent.

5.3 Application impact

As stipulated above, when a Semantic Web application reads a statement involving a previously unknown URI, and it needs to know more about that URI, its task is to locate the URI definition that the statement author intended. In case 1 (normal case), under the URI declarations approach (rule C-UD-1) the process is simple: the application uses the URI's follow-your-nose definition. But under the competing definitions approach (rule C-CD-1) the application faces two problems:

1. Until there is a standard mechanism for specifying an alternate URI definition (to be used *instead* of any f-y-n definition), the application cannot, without assistance, be assured of getting the right URI definition, hence violating the self-describing Web [12] principle.

2. Alternate URI definitions cause URI collision, as described above.

Of course, problem 1 will go away if such a mechanism is standardized. But problem 2 will not. In short, in the normal case, *any architectural rule that permits statement authors using the same URI to base their statements on different URI definitions leads to harmful URI collision.*

Furthermore, even if the problem of URI collision is viewed as a necessary harm en route to the higher goal of community-accepted URI definitions, the notion of a "community-accepted definition" is tenuous, because different communities might "standardize" the definition of a URI differently. That would be bad, because the Semantic Web should enable serendipitous combinations of data *across* communities. In fact, it may be difficult to even ascertain whether a community-accepted definition had reached the globally accepted stage. Furthermore, as the number of parties increases, the difficulty of reaching agreement increases. In such situations it seems much more likely that the eventual outcome would not be a single, community-accepted definition, but a community-accepted definition for *each* community, thus perpetuating URI collision indefinitely.

5.4 URI-translating proxy

Regardless of which of the three cases in rule C-CD or C-UD a URI fell under when a statement is made, the situation may have changed by the time an application reads that statement: documents, may have been moved or changed, etc. How can the application still use the right URI definition?

One simple implementation strategy is to use a **URI-translating proxy** that transparently redirects from a URI definition's old URI to its new URI. The proxy can be driven by an **exception list** of <*oldURI*, *newURI*> pairs, such that if the application attempts to dereference *oldURI*, the proxy will instead dereference *newURI*, thus insulating the application from the exceptions. The exception list can thus cover cases in which:

- the f-y-n definition is unavailable;
- the f-y-n definition is erroneous (for example, if the domain was hijacked); or
- the f-y-n definition should be superceded (for example, by a community-accepted URI definition).

Such a proxy could be used in either the URI declarations approach or the competing definitions approach, and the exception list would have to be updated as links break, definitions are accepted or deprecated by the community, etc. However, since the competing definitions approach encourages the indefinite accumulation of community-defined URIs, and a URI may more freely change from using the f-y-n definition to a community-accepted definition, maintenance of the exceptions list becomes significantly less burdensome under the URI declarations approach, which discourages such exceptions.

6 URI relationships

Suppose Bob does not wish to agree with one of the identifying assertions in Alice's URI definition for http://alice.example#foo. Bob would like to use a URI definition that is essentially the same as Alice's except that it would omit the offending assertion that Bob doesn't like. He therefore decides to mint a new URI, http://bob.example#foo, with a new URI definition, and indicate its relationship to Alice's URI. Bob's URI would therefore denote a skos:broader [7] concept than Alice's URI denotes.

How should Bob indicate the relationship between his URI and Alice's URI? If Bob were to naively write something like the following:

```
<http://alice.example#foo> skos:broader
<http://bob.example#foo> . # WRONG!
```

then his use of Alice's URI would be imply agreement with Alice's offending assertion! To avoid this problem, Bob needs to indicate the relationship without using Alice's URI in its usual denotational way. This can be done by using Alice's URI as a string literal, and using a property such as log:uri [14] to relate Alice's URI as a string literal to the resource that Alice's URI normally denotes:

```
# Right:
_:aliceFoo log:uri "http://alice.example#foo" .
_:aliceFoo skos:broader <http://bob.example#foo> .
```

`where` the log:uri property relates a URI to the resource it denotes, such that for any URI *u*, if *u* is used to denote a resource, then the following relationship is implied:

```
<u> log:uri "u"^^xsd:anyURI .
```

This use of a blank node and log:uri shows one way the relationship between Bob's URI and Alice's URI can be expressed without implying agreement to the assertions in Alices's URI definition. Other quoting mechanisms may work also.

7 Conclusions

This comparison has shown that under certain assumptions that view "meaning" as sets of assertions, the URI declarations approach has architectural properties that better support the Semantic Web than the competing definitions approach. Of course, some readers may disagree with some of the assumptions made herein, and the comparison of pros and cons may lack criteria that some readers find important. Furthermore, there are many other architectural policies that could have been compared instead of those chosen to represent the "competing definitions" approach, and perhaps they would have been better choices for comparison. But this comparison can at least act as a starting point -- a concrete stake in the ground -- in the discussion of how Semantic Web architecture should work.

8 Acknowledgements

Thanks to Pat Hayes for his always interesting discussion on these ideas.

9 References

1. Connelly, Dan: A Pragmatic Theory of Reference for the Web, IRW 2006, 26-May-2006, http://www.w3.org/2006/04/irw65/urisym

2. Booth, David: URI Declaration Versus Use, 25-July-2007, http://dbooth.org/2007/uri-decl/

3. Sauermann, Leo; Cyganiak, Richard: Cool URIs for the Semantic Web, W3C Working Draft 17-Dec-2007, http://www.w3.org/TR/cooluris

4. Jacobs, Ian; Walsh, Norman: Architecture of the World Wide Web, Volume One, W3C Recommendation 15-Dec-2004, http://www.w3.org/TR/webarch/

5. Brickley, Dan; Guha, R.V.: RDF Vocabulary Description Language 1.0: RDF Schema, W3C Recommendation 10-Feb-2004, http://www.w3.org/TR/rdf-schema/

6. Bradner, S.: Key words for use in RFCs to Indicate Requirement Levels, RFC 2119, March 1997, http://www.ietf.org/rfc/rfc2119.txt

7. Miles, Alistair; Bechhofer, Sean: SKOS Simple Knowledge Organization System Reference, W3C Working Draft 25-January-2008, http://www.w3.org/TR/skos-reference

8. Berners-Lee, Tim: Notation 3, W3C, first published 1998, http://www.w3.org/DesignIssues/Notation3.html

10. Brickley, Dan; Miller, Libby: FOAF Vocabulary Specification 0.91, 2-Nov-2007, http://xmlns.com/foaf/spec/

11. Carroll, Lewis: Through the Looking Glass, http://www.sabian.org/Alice/lgchap06.htm

12. Mendelsohn, Noah: The Self-Describing Web, W3C Draft TAG Finding, 08-Feb-2008, http://www.w3.org/2001/tag/doc/selfDescribingDocuments

13. Berners-Lee, Tim: Cool URIs don't change, 1998, http://www.w3.org/Provider/Style/URI

14. Berners-Lee, Tim: log.n3 - n3 definition of some Semantic Web terms, 2006, http://www.w3.org/2000/10/swap/log.n3 . Also available in RDF/XML at http://www.w3.org/2000/10/swap/log .

24-Apr-2008: Changed section 6 example to be clearer; minor edits.
30-May-2008: Changed section 2 explanation to the effect that "equivalent identifying assertions => same resource" instead of "different identifying assertions => different resource", because I'm starting to suspect that even if :a and :b have non-equivalent URI declarations -- presumably one broader than the other -- it is fine to say ":a owl:sameAs :b .".
13-Mar-2008: Initial version