Principles of XML design: Use XML namespaces with care

Minimizing problems while incorporating namespaces into XML design

Uche Ogbuji

April 06, 2004

XML namespaces are an imperfect solution to a difficult problem. From basic information architecture to difficulties with APIs, namespaces can open up rather painful gotchas if used carelessly. In this article, Uche Ogbuji covers some of the more important design principles which, if followed, can minimize problems with namespaces.

Most regular users of XML are quite familiar with XML namespaces and accept them as a basic part of XML. They shake their heads at the occasional oddities of namespaces, but in general don't give them all that much thought. Among XML experts, however, XML namespaces have been very controversial from day one. This controversy is for good reason: Namespaces solve a difficult problem and are one of many approaches to solving this problem, each of which has its pros and cons. The W3C XML namespaces specification is a compromise, and as with all compromises, it often falls short of addressing each user's needs. Even after all this time, namespaces have proven very difficult to incorporate smoothly into XML information architecture, and lack of care with namespaces can cause a lot of complications for XML processing tools. In this article, I go over design considerations that can help you avoid such problems when using XML namespaces. In general my suggested guidelines will be in boldface.

This article will cover XML namespaces 1.0 (including all errata). XML namespaces 1.1 is mercifully modest in its changes, but it is a brand new specification and isn't yet well supported by the tools. I expect that XML namespaces 1.1 will soon become the norm (unlike, say, XML 1.1 which I'm not sure will ever really catch on).

Basic principles

The mechanism of XML namespaces has several moving parts: local names, namespace URIs, prefixes, and declarations. The most important step in using namespaces effectively is to learn how to keep these straight.
Local names

The point of namespaces is that you can use the best concise name for each element or attribute within each context and then put these names in a namespace that distinguishes the context. The concise part of the name that only need be unique within its own context is the local name. Be sure to take advantage of the distinguishing context and don't repeat in local names information that's already inherent in the namespace itself. For example, you don't need to make the local name of the linking element in the XHTML namespace xhtml-link. Since it is already local to the XHTML namespace just link will do. For historical reasons, the XHTML specifications themselves go against this guideline when naming the root element html; it could just as well have been renamed to document.

Namespace URIs

A namespace is a string with the syntax of a URI (often redundantly called the "namespace URI"). The namespace is an integral part of the element's or attribute's name. The combination of a local name and a namespace is called a universal name. In order to highlight the namespace's importance, XML pioneer James Clark developed a notation for universal names that emphasizes how fundamentally namespace and local name are bound (see Related topics). For example, the universal name with local part customer and namespace http://uche.ogbuji.net/eg/ns is written in Clark's notation as {http://uche.ogbuji.net/eg/ns}customer.

Choosing the namespace URI is important. Whether it's better to use URLs or URNs is the source of some debate. The former have the advantage of familiarity, but people often create namespace URLs that do not have any corresponding resource -- that is, if you browse to the equivalent URL you get a 404 "not found" error. URNs have the advantage that they don't encourage people to try to look them up in browsers. Use URLs for namespaces if you are careful to place some sort of document at the URL that would be useful for a reader. I recommend placing an RDDL 1.0 document (see Related topics) at URLs that correspond to namespaces, unless more specialized conventions apply. For example, in RDF/XML documents, namespaces often lead to RDF schema documents when resolved as URLs. URNs have many classes (classes of URNs are formally called "namespaces", not to be confused with XML namespaces). If you don't wish to use URLs, use URNs if your organization has a means of managing and resolving a suitable class of URN. Examples of URN namespaces include oid (an ISO-sanctioned system for assigning numerically coded identifiers to network nodes) and publicid (formal public identifier entities as defined in SGML and XML).

Prefixes

When specifying a universal name in an XML document, you use an abbreviation based on an optional prefix that's attached to the local name. This abbreviation is called the qualified name or qname. The prefix is optional because a special syntactical form allows you to specify a default namespace which is associated with qnames that have no prefix. The prefix is strictly a syntactic convenience; in general, it is not really a matter of XML language design but rather a matter of author or tool preference. I call such issues instance details and I only cover them in these articles on design when in my experience the designer has no choice but to consider them. I recommend that you publish well-known prefixes for namespaces but never make any prefix
Choose well-known prefixes for a namespace when creating documents but accept any chosen prefix for a namespace when reading documents.

Namespace declarations

The namespace declaration is the syntactic device through which prefixes are assigned to namespaces in an XML document. This is technically an instance detail, but important enough that I devote a section (see below) to guidelines for namespace declarations.

Use and evolution of namespaces

Some designers start out not using namespaces and later on adopt namespaces as they feel the need to mix vocabularies. Such a cautious approach can seem sensible considering how tricky namespaces can be. The problem is that since namespaces are a fundamental part of XML names, this change is more significant than you might realize. It requires extensive changes in tools and other related materials. You can deal with name clashes in other ways. Other than namespaces, the leading approaches are ideas based on SGML architectural forms, in which names are directly declared and modified by tools in case of clashes. Try to think as hard as possible about future developments for your XML design and be decisive about whether to deal with name clashes, and how to do so. I have come to agree with many of the criticisms of XML namespaces and dearly wish for a cleaner mechanism that was well established in tools. For practical reasons based on my experience, these days I use namespaces in almost all of my XML designs.

It is also difficult to decide when to evolve or differentiate a namespace. A namespace can be used for versioning, or to differentiate concepts within a domain. The key to best deciding when to do so is to remember that the namespace is a basic part of the name. Change or differentiate the namespace only when you want to make a real, fundamental distinction that defines each element and attribute. If a version change significantly alters the meaning of names in an XML vocabulary, then a namespace change is probably in order. Otherwise, use other versioning mechanisms such as adding a version attribute to top-level elements.

The pitfalls of using namespaces to make distinctions within a domain are best illustrated by example. In 1999 XHTML 1.0 became a finalized proposal. It was really just an XML variation on HTML 4.01, which has three separate DTDs: strict, transitional, and frameset. The XHTML working group decided to use three separate namespaces for the corresponding XHTML DTDs. This decision was met with an uproar in the XML community. The main problem was that even though three separate DTDs existed, the meaning of each element didn't change significantly from one to another; a code element in the XHTML transitional DTD essentially means the same thing as a code element in the XHTML strict DTD. By changing the names in each case, the XHTML design was working against this fact. In the end, the XHTML working group corrected things by issuing new specifications that used a single namespace across the XHTML 1.0 domain. You should heed this lesson well. Make distinctions in XML namespaces only when there are truly distinctions between the things being named.

Unfortunately, things are rarely black and white. A common situation is when a new version of a vocabulary adds new elements. The meaning of the carried over elements may not have changed
and so a namespace change may seem improper. But if you use the same old namespace, it may also seem improper to place the elements added in the new vocabulary in the original namespace. Using a different namespace for only the new elements is rarely a sensible option. In the end, you have to use your judgment to decide whether or not to evolve the namespace with the vocabulary. Some tricks with namespaces may give you other options (see Related topics for a tip on using Namespaces for versioning), but you should use even these with care.

**The Joe English metaphors of namespace sanity**

XML namespace declarations are scoped, meaning that the declared prefix (or default namespace) is in force for the element on which the declaration occurs (as well as its descendant elements), except where another namespace declaration overrides it. However, this flexibility can cause some problems with processing. Joe English, an XML expert working for Advanced Rotorcraft Technology, Inc., famously explained these problems using mental health metaphors which I copy below (see Related topics for the original article). The following are usage patterns for namespaces that I suggest avoiding.

In a **borderline** document (presumably a reference to Borderline Personality Disorder), more than one prefix maps to one namespace:

```xml
<org>
  <a:employee xmlns:a='urn:bogus:ns'>EP</a:employee>
  <b:employee xmlns:b='urn:bogus:ns'>TSE</b:employee>
</org>
```

In a **neurotic** document, the same prefix is used for more than one namespace:

```xml
<h:memo xmlns:a='urn:bogus:ns'>
  <h:body xmlns:a='http://www.w3.org/1999/xhtml'>
    Now hear <h:i>this</h:i>
  </h:body>
</h:memo>
```

In my experience, this pattern is most common where the author goes to some lengths to avoid prefixes. Take the following example, which is neurotic because the default namespace is different depending on where you are in the document:

```xml
<memo xmlns='urn:bogus:ns'>
  <body xmlns='http://www.w3.org/1999/xhtml'>
    Now hear <i>this</i>
  </body>
</memo>
```

In a **psychotic** document, two different prefixes are declared for the same namespace in the same scope:

```xml
<org xmlns:a='urn:bogus:ns' xmlns:b='urn:bogus:ns'>
  <a:employee>EP</a:employee>
  <b:employee>TSE</b:employee>
</org>
```
A document is in **namespace normal** form if all namespace declarations are on the root element and no two prefixes are declared for the same namespace:

```xml
<memo xmlns='urn:bogus:ns' xmlns:html='http://www.w3.org/1999/xhtml'>
   <html:body>
      Now hear <html:i>this</html:i>
   </html:body>
</memo>
```

Avoid borderline, neurotic, and psychotic documents. Try to stick to documents in namespace normal form wherever possible because they are simplest to read and to process.

**Wrap-up**

XML namespaces seem simple on their face, but buried in their nuances is the danger of real complexity and clumsiness if you don't take care while using them. Understand thoroughly the meaning, rules, and implications of the various concepts that make up the XML namespaces mechanism, and stick consistently to simple conventions while designing vocabularies using namespaces and creating actual instance documents.
Related topics

- Don't miss the any of articles in this series on XML design:
  - "When to use elements versus attributes" (developerWorks, March 2004).
  - "Element structures for names and addresses" (developerWorks, August 2004).
- Get the authoritative word on XML namespaces in the W3C's XML Namespaces 1.0 and XML Namespaces 1.1 recommendations.
- Read James Clark’s essay "XML Namespaces," which examines namespaces and introduces a popular notation for describing namespaces.
- If you prefer introduction through a series of examples, check out ZVON's XML namespaces tutorial.
- Bookmark the XML Namespaces FAQ, maintained by Ronald Bourret.
- Read the author's "Tip: Namespaces and versioning" (developerWorks, June 2002) which introduces a mechanism for using XML namespaces to mark the version of XML formats.
- For another look at the subject, read David Marston's articles "Plan to use XML namespaces, Part 1" and "Plan to use XML namespaces, Part 2" (developerWorks, November 2002).
- Read Joe English's important post "A plea for Sanity" on the XML developer's mailing list.
- Learn more about RDDL in Elliotte Rusty Harold's introduction "RDDL Me This: What Does a Namespace URL Locate?," or just go to the RDDL 1.0 specification, which is simple and readable.
- Find details on URIs in RFC 2396: Uniform Resource Identifiers.
- Find more XML resources on the developerWorks XML zone, including Uche Ogbuji's Thinking XML column.
- Find out how you can become an IBM Certified Developer.

© Copyright IBM Corporation 2004
Trademarks
(www.ibm.com/developerworks/ibm/trademarks/)