**WD-xml-names-19980916**

# Namespaces in XML

### World Wide Web Consortium Working Draft 16-September-1998

This version:
http://www.w3.org/TR/1998/WD-xml-names-19980916
Latest version:
http://www.w3.org/TR/WD-xml-names
Previous version:
http://www.w3.org/TR/1998/WD-xml-names-19980802
http://www.w3.org/TR/1998/WD-xml-names-19980518
http://www.w3.org/TR/1998/WD-xml-names-19980327
Editors:
Tim Bray (Textuality <tbray@textuality.com>
Dave Hollander (Hewlett-Packard Company <dmh@corp.hp.com>
Andrew Layman (Microsoft <andrewl@microsoft.com>



## Status of this document

This draft specification is a work in progress representing the consensus of the W3C XML Working Group. This is a W3C Working Draft for review by W3C members and other interested parties. Publication as a working draft does not imply endorsement by the W3C membership.

With the publication of this draft, the Namespace specification enters "last call." For the period ending October 9, 1998, comments on this draft should be sent to the editors at xml-names-issues@w3.org (archive).

While we do not anticipate substantial changes, we still caution that further changes are possible and therefore we recommend that only experimental software or software that can be easily field-upgraded be implemented to this specification at this time. This is a draft document and may be updated, replaced or obsoleted by other documents at any time. It is inappropriate to cite W3C Working Drafts as other than "work in progress".

## Abstract

XML namespaces provide a simple method for qualifying names used in Extensible Markup Language documents by associating them with namespaces identified by URI.

## Table of Contents

1. Motivation and Summary
2. Declaring Namespaces
3. Qualified Names
4. Using Qualified Names
5. Applying Namespaces to Elements and Attributes
   5.1 Namespace Scoping
   5.2 Namespace Defaulting
   5.3 Uniqueness of Attributes
6. Conformance

### Appendices

A. The Internal Structure of XML Namespaces
   A.1 The Insufficiency of the Traditional Namespace
   A.2 XML Namespace Partitions
   A.3 Expanded Element Types and Attribute Names
   A.4 Unique Expanded Attribute Names
B. Acknowledgements
C. References

---

# 1. Motivation and Summary

We envision applications of Extensible Markup Language (XML) where a single XML document may contain elements and attributes that are defined for and used by multiple software modules. One motivation for this is modularity; if such a markup vocabulary exists which is well-understood and for which there is useful software available, it is better to re-use this markup rather than re-invent it.

Such documents, containing multiple markup vocabularies, pose problems of recognition and collision. Software modules need to be able to recognize the tags and attributes which they are designed to process, even in the face of "collisions" occurring when markup intended for some other software package uses the same element type or attribute name.

These considerations require that document constructs should have universal names, whose scope extends beyond their containing document. This specification describes a mechanism *XML namespaces*, which accomplishes this.

[Definition:] An **XML namespace** is a collection of names, identified by a URI, which are used in XML documents as element types and attribute names. XML namespaces differ from the "namespaces" conventionally used in computing disciplines in that the XML version has internal structure and is not, mathematically speaking, a set. These issues are discussed in "A. The Internal Structure of XML Namespaces".

Names from XML namespaces may appear as qualified names, which contain a single colon, separating the name into a namespace prefix and a local part. The prefix, which is mapped to a URI [URI], selects a namespace. The combination of the universally managed URI namespace and the document's own namespace produces identifiers that are universally unique. Mechanisms are provided for prefix scoping and defaulting to avoid clutter and improve readability.

URIs can contain characters not allowed in names, so cannot be used directly as namespace prefixes. Therefore, the namespace prefix serves as a proxy for a URI. An attribute-based syntax described below is used to declare the association of the namespace prefix with a URI; software which supports this namespace proposal must recognize and act on these declarations and prefixes.

## 2. Declaring Namespaces

Note that many of the nonterminals in the productions in this specification are defined not here but in the XML specification [XML]. When nonterminals defined here have the same names as nonterminals defined in the XML specification, the productions here in all cases match a subset of the strings matched by the corresponding ones there.

[Definition:] A namespace is **declared** using an attribute whose prefix is `xmlns` as follows:

**Namespace declaration using attributes**

```
[1]     NSDecl ::= PrefixDef Eq AttValue                    [ NSC: Empty URI ]
[2]  PrefixDef ::= 'xmlns' (':' NCName)?                    [ NSC: Leading "XML" ]
[3]     NCName ::= (Letter | '_') (NCNameChar)*             /* An XML Name, minus
                                                               the ":" */
[4] NCNameChar ::= Letter | Digit | '.' | '-' | '_'
                   | CombiningChar | Extender
```

[Definition:] The `AttValue` in the `NSDecl` production is a URI which functions as a **namespace name** to identify the namespace. The namespace name, to serve its intended purpose, should have the characteristics of uniqueness and persistence. It is not a goal that it be directly usable for retrieval of a schema (if any exists). An example of a syntax that is designed with these goals in mind is that for Uniform Resource Names [RFC2141]. However, it should be noted that ordinary URLs can be managed in such a way as to achieve these same goals.

[Definition:] In the `PrefixDef` production, if the optional colon and `NCName` are provided, then that `NCName` gives the **namespace prefix**, used to associate names with this namespace in the scope of the element to which the declaration is attached.

[Definition:] If the colon and `NCName` are not provided, then the associated namespace name is that of the **default namespace** in the scope of the element to which the declaration is attached.

**Namespace Constraint: Empty URI**
The `AttValue` may be empty only if the `PrefixDef` is simply `xmlns`, i.e. is declaring a default namespace. Default namespaces and overriding of declarations are discussed in "5. Applying Namespaces to Elements and Attributes".

**Namespace Constraint: Leading "XML"**
Prefixes beginning with the three-letter sequence `x`, `m`, `l`, in any case combination, are reserved for use by XML and XML-related specifications.

An example namespace declaration:

```
<?xml version="1.0"?>
  <x xmlns:edi='http://ecommerce.org/schema'>
    <!-- the edi namespace applies to the "x" element and contents -->
  </x>
```

# 3. Qualified Names

[Definition:] In XML documents conforming to this specification, some names (constructs corresponding to the nonterminal Name) may be given as **qualified names**, defined as follows:

**Qualified Name**

```
[5]      QName ::= (Prefix ':')? LocalPart
[6]     Prefix ::= NCName
[7] LocalPart ::= NCName
```

The Prefix provides the namespace prefix part of the qualified name, and must be associated with a namespace URI in a namespace declaration. [Definition:] The LocalPart provides the **local part** of the qualified name.

Note that the prefix functions *only* as a placeholder for a namespace name. Applications should use the namespace name, not the prefix, in constructing names whose scope extends beyond the containing document.

# 4. Using Qualified Names

In XML documents conforming to this specification, element types are given as qualified names, as follows:

**Element Types and Attribute Names**

```
 [8]         STag ::= '<' QName (S Attribute)* S? '>'  [ NSC: Prefix Declared ]
 [9]         ETag ::= '</' QName S? '>'                [ NSC: Prefix Declared ]
[10] EmptyElemTag ::= '<' QName (S Attribute)* S? '/>' [ NSC: Prefix Declared ]
```

Attribute names are given as qualified names, as follows:

**Attribute**

```
[11] Attribute ::= QName Eq AttValue [ NSC: Prefix Declared ]
```

**Namespace Constraint: Prefix Declared**
The namespace prefix, unless it is `xml` or `xmlns`, must have been declared in a namespace declaration attribute in either the start-tag of the element where the prefix is used or in an an ancestor element (i.e. an element in whose content the prefixed markup occurs). The prefix `xml` is by definition bound to the namespace name `urn:Connolly:input:required`. The prefix `xmlns` is used only for namespace bindings and is not itself bound to any namespace name.

This constraint may lead to operational difficulties in the case where the namespace declaration attribute is provided, not directly in the XML document entity, but via a default attribute declared in an external entity. Such declarations may not be read by software which is based on a non-validating XML processor. Many XML applications, presumably including namespace-sensitive ones, fail to require validating processors. For correct operation with such applications, namespace declarations must be provided either directly or via default attributes declared in the internal subset of the DTD.

Element names and attribute types are also given as qualified names when they appear in declarations in the DTD:

**Qualified Names in Declarations**

```
[12] doctypedecl ::= '<!DOCTYPE' S QName (S ExternalID)? S? ('[' (markupdecl
                     | PEReference | S)* ']' S?)? '>'
[13] elementdecl ::= '<!ELEMENT' S QName S contentspec S? '>'
[14]          cp ::= (QName | choice | seq) ('?' | '*' | '+')?
[15]       Mixed ::= '(' S? '#PCDATA' (S? '|' S? QName)* S? ')*'
                     | '(' S? '#PCDATA' S? ')'
[16] AttlistDecl ::= '<!ATTLIST' S QName AttDef* S? '>'
[17]      AttDef ::= S QName S AttType S DefaultDecl
```

# 5. Applying Namespaces to Elements and Attributes

## 5.1 Namespace Scoping

The namespace declaration is considered to apply to the element where it is specified and to all elements within the content of that element, unless overridden by another namespace declaration with the same `PrefixDef` part:

```
<?xml version="1.0"?>
<!-- everything here is explicitly in the HTML namespace -->
<html:html xmlns:html='http://www.w3.org/TR/REC-html40'>
  <html:head><html:title>Frobnostication</html:title></html:head>
  <html:body><html:p>Moved to
    <html:a href='http://frob.com'>here.</html:a></html:p></html:body>
  </html:html>
```

Multiple namespace prefixes can be declared as attributes of a single element, as shown in this example:

```
<?xml version="1.0"?>
<!-- both namespace prefixes are available throughout -->
<bk:book xmlns:bk='urn:loc.gov:books'
         xmlns:isbn='urn:ISBN:0-395-36341-6'>
    <bk:title>Cheaper by the Dozen</bk:title>
    <isbn:number>1568491379</isbn:number>
</bk:book>
```

## 5.2 Namespace Defaulting

A default namespace is considered to apply to the element where it is declared (if that element has no namespace prefix), and to all elements with no prefix within the content of that element. If the URI in a default namespace declaration is empty, then unprefixed elements in the scope of the declaration are not considered to be in any namespace. Note that default namespaces do not apply directly to attributes.

```
<?xml version="1.0"?>
<!-- everything is in the HTML namespace, in this case by default -->
<html xmlns='http://www.w3.org/TR/REC-html40'>
  <head><title>Frobnostication</title></head>
  <body><p>Moved to
    <a href='http://frob.com'>here</a>.</p></body>
  </html>
```

```
<?xml version="1.0"?>
<!-- unprefixed names are from "books" -->
<book xmlns='urn:loc.gov:books'
      xmlns:isbn='urn:ISBN:0-395-36341-6'>
    <title>Cheaper by the Dozen</title>
    <isbn:number>1568491379</isbn:number>
</book>
```

A larger example of namespace scoping:

```
<?xml version="1.0"?>
<!-- initially, the default namespace is "books" -->
<book xmlns='urn:loc.gov:books'
      xmlns:isbn='urn:ISBN:0-395-36341-6'>
    <title>Cheaper by the Dozen</title>
    <isbn:number>1568491379</isbn:number>
    <notes>
      <!-- drop the default into HTML for some commentary -->
      <p xmlns='urn:w3-org-ns:HTML'>
          This is a <i>funny</i> book!
      </p>
    </notes>
</book>
```

The default namespace, once declared, may be overridden:

```
<?xml version='1.0'?>
<Beers>
  <!-- the default namespace is now that of HTML -->
  <table xmlns='http://www.w3.org/TR/REC-html40'>
   <tr><td>Name</td><td>Origin</td><td>Description</td></tr>
   <tr>
     <!-- drop the HTML namespace inside table cells -->
     <td><brandName xmlns="">Huntsman</brandName></td>
     <td><origin xmlns="">Bath, UK</origin></td>
     <td>
       <details xmlns=""><class>Bitter</class><hop>Fuggles</hop>
         <pro>Wonderful hop, light alcohol, good summer beer</pro>
         <con>Fragile; excessive variance pub to pub</con>
         </details>
       </td>
     </tr>
    </table>
  </Beers>
```

### 5.3 Uniqueness of Attributes

In XML documents conforming to this specification, no start-tag may contain two attributes which::

1. have identical names, or
2. have qualified names with the same local part and with prefixes which have been bound to namespace names which are lexically equivalent. Note that namespace names are URIs, the governing RFCs for which contain rules for establishing lexical equivalence.

For example, each of the `bad` start-tags is illegal in the following:

```
<!-- http://www.w3.org is bound to n1 and n2 -->
<x xmlns:n1="http://www.w3.org"
   xmlns:n2="http://www.w3.org" >
  <bad a="1"     a="2" />
  <bad n1:a="1"  n2:a="2" />
</x>
```

However, each of the following is legal:

```
<!-- http://www.w3.org is bound to n2 and is the default -->
<x xmlns:n1="http://www.w3.org"
   xmlns="http://www.w3.org" />
  <good a="1"     b="2" />
  <good a="1"     n1:a="2" />
</x>
```

# 6. Conformance

In XML documents which conform to this specification, element types and attribute names must match the production either for `NSDecl` or `QName` and must satisfy the "Namespace Constraints".

An XML document conforms to this specification if all other tokens in the document which are required, for XML conformance, to match the XML production for `Name`, match this specification's production for `NCName`.

The effect of conformance is that in such a document:

- All element types and attribute names contain either zero or one colon.
- No entity names, PI targets, or notation names contain any colons.

Strictly speaking, attribute values declared to be of types `ID`, `IDREF(S)`, `ENTITY(IES)`, and `NOTATION` are also `Names`, and thus should be colon-free. However, the declared type of attribute values is in principle only available in documents which have been validated. Thus, in well-formed XML documents, there can be no assurance that the contents of attribute values have been checked for conformance to this specification.

---

# Appendices

# A. The Internal Structure of XML Namespaces

## A.1 The Insufficiency of the Traditional Namespace

In the computing disciplines, the term "namespace" conventionally refers to a *set* of names, i.e. a collection containing no duplicates. However, treating the names used in XML markup as such a namespace would greatly impair their usefulness. The primary use of such names in XML documents is to enable identification of logical structures in documents by software modules such as query processors, stylesheet-driven rendering engines, and schema-driven validators. Consider the following example:

```
<section><title>Book-Signing Event</title>
<signing>
  <author title="Mr" name="Vikram Seth" />
  <book title="A Suitable Boy" price="$22.95" /></signing>
<signing>
  <author title="Dr" name="Oliver Sacks" />
  <book title="The Island of the Color-Blind" price="$12.95" /></signing>
</section>
```

In this example, there are three occurrences of the name `title` within markup, and the name alone clearly provides insufficient information to allow correct processing by a software module.

Another problematic area comes from the use of "global" attributes, as illustrated by this example, a fragment of an XML document which is to be displayed using a CSS stylesheet:

```
<RESERVATION>
 <NAME HTML:CLASS="largeSansSerif">Layman, A</NAME>
 <SEAT CLASS="Y" HTML:CLASS="largeMonotype">33B</SEAT>
 <DEPARTURE>1997-05-24T07:55:00+1</DEPARTURE></RESERVATION>
```

In this case, the `CLASS` attribute, which describes the fare basis and takes values such as "J", "Y", and "C", is distinct at all semantic levels from the `HTML:CLASS` attribute, which is used to achieve CSS formatting effects.

XML 1.0 does not provide a built-in way to declare "global" attributes; items such as the HTML `CLASS` attribute are global only in their prose description and their interpretation by HTML applications. However, such attributes, an important distinguishing feature of which is that their names are unique, are commonly observed to occur in a variety of applications.

## A.2 XML Namespace Partitions

In order to support the goal of making both qualified and unqualified names useful in meeting their intended purpose, we identify the names appearing in an XML namespace as belonging to one of several disjoint traditional (i.e. set-structured) namespaces, called namespace partitions. The partitions are:

**The All Element Types Partition**

All element types in an XML namespace appear in this partition. Each has a unique local part; the combination of the namespace name and the local part uniquely identifies the element type.

**The Global Attribute Partition**

This partition contains the names of all attributes which are defined, in this namespace, to be global. The only required characteristic of a global attribute is that its name be unique in the global attribute partition. This specification makes no assertions as to the proper usage of such attributes. The combination of the namespace name and the attribute name uniquely identifies the global attribute.

**The Per-Element-Type Partitions**

Each type in the All Element Types Partition has an associated namespace in which appear the names of the unqualified attributes that are provided for that element. This is a traditional namespace because the appearance of duplicate attribute names on an element is forbidden by XML 1.0. The combination of the attribute name with the element's type and namespace name uniquely identifies each unqualified attribute.

In XML documents conforming to this specification, the names of all qualified (prefixed) attributes are assigned to the global attribute partition, and the names of all unqualified attributes are assigned to the appropriate per-element-type partition.

## A.3 Expanded Element Types and Attribute Names

For convenience in specifying rules and in making comparisons, we define an expanded form, expressed here in XML element syntax, for each element type and attribute name in an XML document.

[Definition:] An **expanded element type** is expressed as an empty XML element of type `ExpEType`. It has a required `type` attribute which gives the type's `LocalPart`, and an optional `ns` attribute which, if the element is qualified, gives its namespace name.

[Definition:] An **expanded attribute name** is expressed as an empty XML element of type `ExpAName`. It has a required `name` attribute which gives the name. If the attribute is global, it has a required `ns` attribute which gives the namespace name; otherwise, it has a required attribute `eltype` which gives the type of the attached element, and an optional attribute `elns` which gives the namespace name, if known, of the attached element.

Slight variations on the examples given above will illustrate the working of expanded element types and attribute names. The following two fragments are each followed by a table showing the expansion of the names:

```
<!-- 1 --> <section xmlns='urn:com:books-r-us'>
<!-- 2 -->   <title>Book-Signing Event</title>
<!-- 3 -->   <signing>
<!-- 4 -->     <author title="Mr" name="Vikram Seth" />
<!-- 5 -->     <book title="A Suitable Boy" price="$22.95" />
             </signing>
           </section>
```

The names would expand as follows:

| Line | Name | Expanded |
|---|---|---|
| 1 | section | <ExpEType type="section" ns="urn:com:books-r-us" /> |
| 2 | title | <ExpEType type="title" ns="urn:com:books-r-us" /> |
| 3 | signing | <ExpEType type="signing" ns="urn:com:books-r-us" /> |
| 4 | author | <ExpEType type="author" ns="urn:com:books-r-us" /> |
| 4 | title | <ExpAName name='title' eltype="author" elns="urn:com:books-r-us" /> |
| 4 | name | <ExpAName name='name' eltype="author" elns="urn:com:books-r-us" /> |
| 5 | book | <ExpEType type="book" ns="urn:com:books-r-us" /> |
| 5 | title | <ExpAName name='title' eltype="book" elns="urn:com:books-r-us" /> |
| 5 | price | <ExpAName name='price' eltype="book" elns="urn:com:books-r-us" /> |

```
<!-- 1 --> <RESERVATION xmlns:HTML="http://www.w3.org/TR/REC-html40">
<!-- 2 --> <NAME HTML:CLASS="largeSansSerif">Layman, A</NAME>
<!-- 3 --> <SEAT CLASS="Y" HTML:CLASS="largeMonotype">33B</SEAT>
<!-- 4 --> <HTML:A HREF='/cgi-bin/ResStatus'>Check Status</HTML:A>
<!-- 5 --> <DEPARTURE>1997-05-24T07:55:00+1</DEPARTURE></RESERVATION>
```

| | | |
|---|---|---|
| 1 | RESERVATION | <ExpEType type="RESERVATION" /> |
| 2 | NAME | <ExpEType type="NAME" /> |
| 2 | HTML:CLASS | <ExpAName name="CLASS" ns=http://www.w3.org/TR/REC-html40 /> |
| 3 | SEAT | <ExpEType type="SEAT" /> |
| 3 | CLASS | <ExpAName name="CLASS" eltype="SEAT"> |
| 3 | HTML:CLASS | <ExpAName name="CLASS" ns="http://www.w3.org/TR/REC-html40" /> |
| 4 | HTML:A | <ExpEType type="A" ns="http://www.w3.org/TR/REC-html40" /> |
| 4 | HREF | <ExpAName name="HREF" eltype="A" elns="http://www.w3.org/TR/REC-html40" /> |
| 5 | DEPARTURE | <ExpEType type="DEPARTURE" /> |

## A.4 Unique Expanded Attribute Names

The constraint expressed by "5.3 Uniqueness of Attributes" above may straightforwardly be implemented by requiring that no element have two attributes whose expanded names are equivalent, i.e. have the same attribute-value pairs and child elements with the same content.

# B. Acknowledgements

This work reflects input from a very large number of people, including especially the members of the World Wide Web Consortium XML Working Group and Special Interest Group and the participants in the W3C Metadata Activity.

# C. References

RFC21 41

*URN Syntax*, ed. R. Moats.
IETF RFC 2141 May 1997.

XML

*Extensible Markup Language (XML) 1.0*, eds. Tim Bray, Jean Paoli, and C. M. Sperberg-McQueen. W3C Recommendation 10 February 1998.
Available a thttp://www.w3.org/TR/1998/REC-xml-19980210.

URI

*Uniform Resource Identifiers (URI): Generic Syntax* eds. T. Berners-Lee, R. Fielding, L. Masinter, IETF RFC 2396, August 1998.