Exclusive XML Canonicalization Version 1.0

W3C Recommendation 18 July 2002

This version:
http://www.w3.org/TR/2002/REC-xml-exc-c14n-20020718/
Latest version:
http://www.w3.org/TR/xml-exc-c14n/
Previous version:
http://www.w3.org/TR/2002/PR-xml-exc-c14n-20020524/
Authors/Editors:
John Boyer, PureEdge Solutions Inc., jboyer@PureEdge.com
Donald E. Eastlake 3rd, Motorola, Donald.Eastlake@Motorola.com
Joseph Reagle, W3C, reagle@w3.org

Please see the **errata** for this document, which may include some normative corrections. See also **translations**.

Abstract

Canonical XML [XML-C14N] specifies a standard serialization of XML that, when applied to a subdocument, includes the subdocument's ancestor context including all of the namespace declarations and attributes in the "xml:" namespace. However, some applications require a method which, to the extent practical, excludes ancestor context from a canonicalized subdocument. For example, one might require a digital signature over an XML payload (subdocument) in an XML message that will not break when that subdocument is removed from its original message and/or inserted into a different context. This requirement is satisfied by Exclusive XML Canonicalization.

Status of this document

This document is the W3C Exclusive Canonicalization Recommendation. This document has been reviewed by W3C Members and other interested parties and has been endorsed by the Director as a W3C Recommendation. It is a stable document and may be used as reference material or cited as a normative reference 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.

This specification was produced by the IETF/W3C XML Signature Working Group (W3C Activity Statement) which believes the specification is sufficient for the creation of independent interoperable implementations as demonstrated in the Interoperability Report.

Patent disclosures relevant to this specification may be found on the Working Group's patent

disclosure page, in conformance with W3C policy, and the IETF Page of Intellectual Property Rights Notices, in conformance with IETF policy. At the time of publication, there are no declarations specific to this document.

Please report errors in this document to w3c-ietf-xmldsig@w3.org (archive).

The list of known errors in this specification is available at http://www.w3.org/2002/07/xml-exc-c14n-errata.

The English version of this specification is the only normative version. Information about translations of this document (if any) is available http://www.w3.org/Signature/2002/02/xmldsig-translations

A list of current W3C Technical Reports can be found at http://www.w3.org/TR/.

Table of Contents

---

1. Introduction

The XML Recommendation [XML] specifies the syntax of a class of objects called XML documents. The Namespaces in XML Recommendation [XML-NS] specifies additional syntax and semantics for XML documents. It is normal for XML documents and subdocuments which are equivalent for the purposes of many applications to differ in their physical representation. For example, they may differ in their entity structure, attribute ordering, and character encoding. The goal of this specification is to establish a method for serializing the XPath node-set representation of an XML document or subset such that:

1. The node-set is minimally affected by any XML context which has been omitted.
2. The canonicalization of a node-set representing well-balanced XML [XML-Fragment] will be unaltered by further applications of exclusive canonicalization.
3. It can be determined whether two node-sets are identical except for transformations considered insignificant by this specification under [XML,XML-NS].

An understanding of the Canonical XML Recommendation [XML-C14N] is required.

1.1 Terminology

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 RFC 2119 [Keywords].

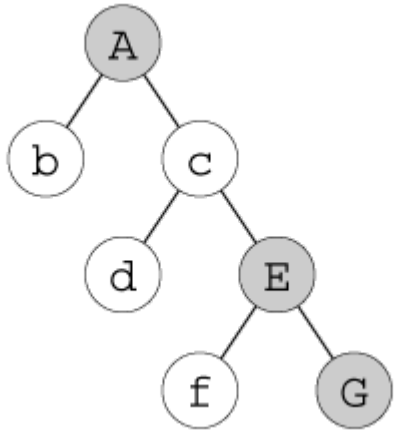
The XPath 1.0 Recommendation [XPath] defines the term *node-set* and specifies a data model for representing an input XML document as a set of nodes of various types (element, attribute, namespace, text, comment, processing instruction, and root). The nodes are included in or excluded from a node-set based on the evaluation of an expression. Within this specification and [XML-C14N], a node-set is used to directly indicate whether or not each node should be rendered in the canonical form (in this sense, it is used as a formal mathematical set). A node that is excluded from the set is not rendered in the canonical form being generated, even if its parent node is included in the node-set. However, an omitted node may still impact the rendering of its descendants (e.g. by affecting the namespace context of the descendants).

A **document subset** is a portion of an XML document indicated by an XPath node-set that may not include all of the nodes in the document. As *defined* in [XPath] every node (e.g., element, attribute, and namespace), has exactly one **parent**, which is either an element node or the root node. An **apex node** is an element node in a document subset having no element node ancestor in the document subset. An **orphan node** is an element node whose parent element node is not in the document subset. The **output parent** of an orphan node that is not an apex node is the nearest ancestor element of the orphan node that is in the document subset; an *apex node* has no *output parent*. The output parent of a non-orphan node is the parent of the node. An **output ancestor** is any ancestor element node in the document subset.

For example given a document tree with three generations under the root node `A` and where capitalization denotes the node is in the document subset (`A,E,G`).

Pictorial Representation:

```
<A>
 <b />
 <c>
  <d />
  <E>
   <f />
   <G />
  </E>
 </c>
</A>
```



Textual Representation:

```
A-+-b
  `-c-+-d
      `-E-+-f
          `-G
```

The following characteristics apply:

- `A` is an apex node, output parent of `E`, and output ancestor of (`E,G`);
- `E` is an orphan node and the output parent of `G`.

An element *E* in a document subset **visibly utilizes** a namespace declaration, i.e. a namespace prefix *P* and bound value *V*, if *E* or an attribute node in the document subset with parent *E* has a qualified name in which *P* is the namespace prefix. A similar definition applies for an element *E* in a document subset that *visibly utilizes* the default namespace declaration, which occurs if *E* has no namespace prefix.

The namespace axis of an element contains nodes for all non-default namespace declarations made within the element as well as non-default namespace declarations inherited from ancestors of the element. The namespace axis also contains a node representing the default namespace if it is not the empty string, whether the default namespace was declared within the element or by an ancestor of the element. Any subset of the nodes in a namespace axis can be included in a document subset.

The method of canonicalization described in this specification receives an **InclusiveNamespaces PrefixList** parameter, which lists namespace prefixes that are handled in the manner described by the Canonical XML Recommendation [XML-C14N].

The **exclusive canonical form** of a document subset is a physical representation of the XPath node-set, as an octet sequence, produced by the method described in this specification. It is as defined in the Canonical XML Recommendation [XML-C14N] except for the changes summarized as follows:

- attributes in the XML namespace, such as `xml:lang` and `xml:space` are not imported into orphan nodes of the document subset, and
- namespace nodes that are not on the *InclusiveNamespaces PrefixList* are expressed only in start tags where they are visible and if they are not in effect from an *output ancestor* of that tag.

The term **exclusive canonical XML** refers to XML that is in exclusive canonical form. The **exclusive XML canonicalization method** is the algorithm defined by this specification that generates the exclusive canonical form of a given XML document subset. The term **exclusive XML canonicalization** refers to the process of applying the exclusive XML canonicalization method to an XML document subset.

1.2 Applications

The applications of Exclusive XML Canonicalization are very similar to those for Canonical XML [XML-C14N]. However, exclusive canonicalization, or equivalent means of excluding most XML context, is necessary for signature applications where the XML context of signed XML will change. This sort of change is typical of many protocol applications.

Note that in the case of the `SignedInfo` element of [XML-DSig], the specification of an appropriate canonicalization method is the only technique available to protect the signature from insignificant changes in physical form and changes in XML context.

1.3 Limitations

Exclusive XML Canonicalization has the limitations of Canonical XML [XML-C14N] plus two additional limitations as follows:

1. The XML being canonicalized may depend on the effect of XML namespace attributes, such as `xml:lang`, `xml:space`, and `xml:base` appearing in ancestor nodes. To avoid problems due to the non-importation of such attributes into an enveloped document subset, either they must be explicitly given in the *apex nodes* of the XML document subset being canonicalized or they must always be declared with an equivalent value in every context in which the XML document subset will be interpreted.
2. Applications that use the XML being canonicalized may depend on the effect of XML namespace declarations where the namespace prefix being bound is not *visibly utilized*. An example would be an attribute whose value is an XPath expression and whose evaluation therefore depends upon namespace prefixes referenced in the expression. Or, an attribute value might be considered a QName [XML-NS] by some applications, but it is only a string-value to XPath:

   ```
   <number xsi:type="xsd:decimal">10.09</number>.
   ```

To avoid problems with such namespace declarations,

- the XML must be modified so that use of the namespace prefix involved is visible, or
- the namespace declarations must appear and be bound to the same values in every context in which the XML will be interpreted, or
- the prefixes for such namespaces must appear in the *InclusiveNamespaces PrefixList*.

2. The Need for Exclusive XML Canonicalization

In some cases, particularly for signed XML in protocol applications, there is a need to canonicalize a subdocument in such a way that it is substantially independent of its XML context. This is because, in protocol applications, it is common to envelope XML in various layers of message or transport elements, to strip off such enveloping, and to construct new protocol messages, parts of which were extracted from different messages previously received. If the pieces of XML in question are signed, they need to be canonicalized in a way such that these operations do not break the signature but the signature still provides as much security as can be practically obtained.

2.1 A Simple Example

As a simple example of the type of problem that changes in XML context can cause for signatures, consider the following document:

```
<n1:elem1 xmlns:n1="http://b.example">
    content
</n1:elem1>
```

this is then enveloped in another document:

```
<n0:pdu xmlns:n0="http://a.example">
   <n1:elem1 xmlns:n1="http://b.example">
       content
   </n1:elem1>
</n0:pdu>
```

The first document above is in canonical form. But assume that document is enveloped as in the second case. The subdocument with `elem1` as its apex node can be extracted from this second case with an XPath expression such as:

```
(//. | //@* | //namespace::*)[ancestor-or-self::n1:elem1]
```

The result of applying Canonical XML to the resulting XPath node-set is the following (except for line wrapping to fit this document):

```
<n1:elem1 xmlns:n0="http://a.example"
          xmlns:n1="http://b.example">
    content
</n1:elem1>
```

Note that the `n0` namespace has been included by Canonical XML because it includes namespace context. This change which would break a signature over `elem1` based on the first version.

2.2 General Problems with re-Enveloping

As a more complete example of the changes in canonical form that can occur when the enveloping context of a document subset is changed, consider the following document:

```
<n0:local xmlns:n0="foo:bar"
          xmlns:n3="ftp://example.org">
   <n1:elem2 xmlns:n1="http://example.net"
             xml:lang="en">
       <n3:stuff xmlns:n3="ftp://example.org"/>
   </n1:elem2>
</n0:local>
```

And the following which has been produced by changing the enveloping of `elem2`:

```
<n2:pdu xmlns:n1="http://example.com"
        xmlns:n2="http://foo.example"
        xml:lang="fr"
        xml:space="retain">
   <n1:elem2 xmlns:n1="http://example.net"
             xml:lang="en">
       <n3:stuff xmlns:n3="ftp://example.org"/>
   </n1:elem2>
</n2:pdu>
```

Assume an XPath node-set produced from each case by applying the following XPath expression:

```
(//. | //@* | //namespace::*)[ancestor-or-self::n1:elem2]
```

Applying Canonical XML to the node-set produced from the first document yields the following serialization (except for line wrapping to fit in this document):

```
<n1:elem2 xmlns:n0="foo:bar"
          xmlns:n1="http://example.net"
          xmlns:n3="ftp://example.org"
          xml:lang="en">
    <n3:stuff></n3:stuff>
</n1:elem2>
```

However, although `elem2` is represented by the same octet sequence in both pieces of external XML above, the Canonical XML version of `elem2` from the second case would be (except for line wrapping so it will fit into this document) as follows:

```
<n1:elem2 xmlns:n1="http://example.net"
          xmlns:n2="http://foo.example"
          xml:lang="en"
          xml:space="retain">
    <n3:stuff xmlns:n3="ftp://example.org"></n3:stuff>
</n1:elem2>
```

Note that the change in context has resulted in lots of changes in the subdocument as serialized by the inclusive Canonical XML [XML-C14N]. In the first example, `n0` had been included from the context and the presence of an identical `n3` namespace declaration in the context had elevated that declaration to the apex of the canonicalized form. In the second example, `n0` has gone away but `n2` has appeared, `n3` is no longer elevated, and an `xml:space` declaration has appeared, due to changes in context. But not all context changes have effect. In the second example, the presence at ancestor nodes of an `xml:lang` and `n1` prefix namespace declaration have no effect because of existing declarations at the `elem2` node.

On the other hand, using Exclusive XML Canonicalization as specified herein, the physical form of `elem2` as extracted by the XPath expression above is (except for line wrapping so it will fit into this document) as follows:

```
<n1:elem2 xmlns:n1="http://example.net"
          xml:lang="en">
    <n3:stuff xmlns:n3="ftp://example.org"></n3:stuff>
</n1:elem2>
```

in both cases.

3. Specification of Exclusive XML Canonicalization

The data model, processing, input parameters, and output data for Exclusive XML Canonicalization are the same as for Canonical XML [XML-C14N] with the following exceptions:

1. Canonical XML applied to a document subset requires the search of the ancestor nodes of each orphan element node for attributes in the XML namespace, such as `xml:lang` and `xml:space`. These are copied into the element node except if a declaration of the same attribute is already in the attribute axis of the element (whether or not it is included in the document subset). This search and copying are *omitted* from the Exclusive XML Canonicalization method.
2. The Exclusive XML Canonicalization method may receive an additional, possibly null, parameter *InclusiveNamespaces PrefixList* containing a list of namespace prefixes and/or a token indicating the presence of the default namespace. All namespace nodes appearing on this list are handled as provided in Canonical XML [XML-C14N].
3. A namespace node **N** with a prefix that does not appear in the *InclusiveNamespaces PrefixList* is rendered if all of the conditions are met:
    1. Its parent element is in the node-set, and
    2. it is *visibly utilized* by its parent element, and
    3. the prefix has not yet been rendered by any *output ancestor*, or the nearest *output ancestor* of its parent element that *visibly utilizes* the namespace prefix does not have a namespace node in the node-set with the same namespace prefix *and* value as **N**.
4. If the token representing the default namespace is not present in *InclusiveNamespaces PrefixList*, then the rules for rendering `xmlns=""` are changed as follows. When canonicalizing the namespace axis of an element **E** that is in the node-set, output `xmlns=""` if and only if all of the conditions are met:
    1. **E** *visibly utilizes* the default namespace (i.e., it has no namespace prefix), and
    2. it has no default namespace node in the node-set, and
    3. the nearest output ancestor of E that visibly utilizes the default namespace has a default namespace node in the node-set.

    (This step for for `xmlns=""` is necessary because it is not represented in the XPath data model as a namespace node, but as the absence of a namespace node; see §4.7 Propagation of Default Namespace Declaration in Document Subsets [XML-C14N].)

3.1 Constrained Implementation (non-normative)

The following is a (non-normative) method for implementing the Exclusive XML Canonicalization method for many straightforward cases -- it assumes a well-formed subset and that if an element is in the node-set, so is all of its namespace axis; if the element is not in the subset, neither is its namespace axis.

1. Recursively process the *entire* tree (from which the XPath node-set was selected) in document order starting with the root. (The operation of copying ancestor `xml:` namespace attributes into output *apex element nodes* is *not* done.)
2. If the node is not in the XPath subset, continue to process its children element nodes recursively.
3. If the element node is in the XPath subset then output the node in accordance with Canonical XML except for namespace nodes which are rendered as follows:
    1. `ns_rendered` is a copy of a dictionary, off the top of the `state` stack, of prefixes and their

values which have already been rendered by an *output ancestor* of the namespace node's parent element.
2. Render each namespace node if and only if all of the conditions are met:
    1. it is *visibly utilized* by the immediate parent element or one of its attributes, *or* is present in *InclusiveNamespaces PrefixList*, and
    2. its prefix and value *do not* appear in `ns_rendered`.
3. Render `xmlns=""` if and only if all of the conditions are met:
    1. The default namespace is *visibly utilized* by the immediate parent element node, or the default prefix token is present in *InclusiveNamespaces PrefixList*, and
    2. the element does not have a namespace node in the node-set declaring a value for the default namespace, and
    3. the default namespace prefix is present in the dictionary `ns_rendered`.
5. Insert all the rendered namespace nodes (including `xmlns=""`) into the `ns_rendered` dictionary, replacing any existing entries. Push `ns_rendered` onto the state stack and recurse.
6. After the recursion returns, pop the `state` stack.

4. Use in XML Security

Exclusive Canonicalization may be used as a `Transform` or `CanonicalizationMethod` algorithm in XML Digital Signature [XML-DSig] and XML Encryption [XML-Enc].

Identifier:

http://www.w3.org/2001/10/xml-exc-c14n#

http://www.w3.org/2001/10/xml-exc-c14n#WithComments

Just as with [XML-C14N] one may use the "#WithComments" parameter to include the serialization of XML comments. This algorithm also takes an optional explicit parameter of an empty `InclusiveNamespaces` element with a `PrefixList` attribute. The value of this attribute, which may be null, is a white space delimited list of namespace prefixes, and where #default indicates the default namespace, to be handled as per [XML-C14N]. The list is in NMTOKENS format (a white space separated list). For example:

```
<ds:Transform
   Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
   <ec:InclusiveNamespaces PrefixList="dsig soap #default"
       xmlns:ec="http://www.w3.org/2001/10/xml-exc-c14n#"/>
</ds:Transform>
```

indicates the exclusive canonicalization transform, but that namespaces with prefix "dsig" or "soap" and default namespaces should be processed according to [XML-C14N].

Schema Definition:

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE schema
  PUBLIC "-//W3C//DTD XMLSchema 200102//EN" "http://www.w3.org/2001/XMLSchema.dtd"
 [
   <!ATTLIST schema
     xmlns:ec CDATA #FIXED 'http://www.w3.org/2001/10/xml-exc-c14n#'>
   <!ENTITY ec 'http://www.w3.org/2001/10/xml-exc-c14n#'>
   <!ENTITY % p ''>
   <!ENTITY % s ''>
  ]>

<schema xmlns="http://www.w3.org/2001/XMLSchema"
        xmlns:ec="http://www.w3.org/2001/10/xml-exc-c14n#"
        targetNamespace="http://www.w3.org/2001/10/xml-exc-c14n#"
```

```
        version="0.1" elementFormDefault="qualified">

  <element name="InclusiveNamespaces"
           type="ec:InclusiveNamespaces"/>
  <complexType name="InclusiveNamespaces">
     <attribute name="PrefixList" type="NMTOKENS"/>
  </complexType>
</schema>
```

```
DTD:
<!ELEMENT InclusiveNamespaces    EMPTY >
<!ATTLIST InclusiveNamespaces
   PrefixList    NMTOKENS    #REQUIRED >
```

5. Security Considerations

This specification is used to serialize an XPath node-set under certain assumptions given in [XML-C14N] and this specification. Three such examples include:

1. implementations of [XML-C14N] and this specification do not render an XML declaration;
2. implementations of this specification only render attributes from the "XML" namespace (e.g., `xml:lang`, `xml:space`, and `xml:base`) when they are in the subset being serialized;
3. implementations of this specification do not consider the appearance of a namespace prefix within an attribute value to be *visibly utilized*.

While such choices are consistent with other XML specifications and satisfy the Working Group's application requirements it is important that an XML application carefully construct its transforms such that the result is meaningful and unambiguous in its application context. In addition to this section, the Limitations of this specification, the Resolutions of [XML-C14N], and the Security Considerations of [XML-DSig] should be carefully attended to.

5.1 Target Context

The requirement of this specification is to satisfy applications that "require a method which, to the extent practical, excludes ancestor context from a canonicalized subdocument." Given a fragment being removed from its source instance, this specification satisfies this requirement by excluding from the fragment any context from its ancestors that is not utilized. Consequently, a signature [XML-DSig] over that fragment will remain valid in its source context, removed from the source context, and even in a new target context. However, this specification does not insulate the fragment against confused interpretation in a target context.

For example, if the `<Foo/>` element is signed in its source instance of `<Bar/><Foo/></Bar>` and then removed and placed in the target instance `<Baz xmlns="http://example.org/bar"/><Foo/></Baz>`, the signature should still be valid, but won't be if `<Foo/>` is interpreted as belonging to the `http://example.org/bar` namespace: this is dependent on how nodes are processed.

This specification does not define mechanisms of removing, inserting, and "fixing up" a node-set. (For an example of this sort of specification, see the processing required of Creating the Result Infoset (section 4.5) when an [XInclude] is performed.) Instead, applications must carefully specify the XML (i.e., source, fragment, and target) or define the node-set processing (i.e., removal, replacement, and insertion) with respect to default namespace declarations (e.g., `xmlns=""`) and XML attributes (e.g., `xml:lang`, `xml:space`, and `xml:base`).

5.2 "Esoteric" Node-sets

Consider an application that might use this specification or [XML-C14N] to serialize a single attribute node. An implementation of either specification will *not* emit a namespace declaration for that single

attribute node. Consequently, a "carefully constructed" transform should create a node-set containing the attribute and the relevant namespace declaration for serialization.

This example is provided to caution that as one moves beyond well-formed [XML] and then well-balanced XML [XML-Fragment], it becomes increasingly difficult to create a result that "is meaningful and unambiguous in its application context."

6. References

Keywords
RFC 2119. *Key words for use in RFCs to Indicate Requirement Levels*. S. Bradner. Best Current Practice, March 1997.
Available at http://www.ietf.org/rfc/rfc2119.txt

URI
RFC 2396 . *Uniform Resource Identifiers (URI): Generic Syntax.* T. Berners-Lee, R. Fielding, and L. Masinter. Standards Track, August 1998.
Available at http://www.ietf.org/rfc/rfc2396.txt

XML
Extensible Markup Language (XML) 1.0 (Second Edition). T. Bray, E. Maler, J. Paoli, and C. M. Sperberg-McQueen. W3C Recommendation, October 2000.
Available at http://www.w3.org/TR/2000/REC-xml-20001006 .

XML-C14N
Canonical XML. J. Boyer. W3C Recommendation, March 2001.
Available at http://www.w3.org/TR/2001/REC-xml-c14n-20010315
Available at http://www.ietf.org/rfc/rfc3076.txt

XML-DSig
XML-Signature Syntax and Processing. D. Eastlake, J. Reagle, and D. Solo. IETF Draft Standard/W3C Recommendation, August 2001.
Available at http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/

XML-Fragment
XML Fragment Interchange. P. Grosso, and D. Veillard. W3C Candidate Recommendation, February 2001.
Available at http://www.w3.org/TR/2001/CR-xml-fragment-20010212

XInclude
XML Inclusions (XInclude) Version 1.0. J. Marsh, and D. Orchad. W3C Candidate Recommendation, February 2002.
Available at http://www.w3.org/TR/2002/CR-xinclude-20020221/

XML-NS
Namespaces in XML. T. Bray, D. Hollander, and A. Layman. W3C Recommendation, January 1999.
Available at http://www.w3.org/TR/1999/REC-xml-names-19990114/

XML-Enc
XML Encryption Syntax and Processing. D. Eastlake, and J. Reagle. W3C Candidate Recommendation, March 2002.
Available at http://www.w3.org/TR/2002/CR-xmlenc-core-20020304/

XML-schema
XML Schema Part 1: Structures D. Beech, M. Maloney, N. Mendelsohn, and H. Thompson. W3C Recommendation, May 2001.
Available at http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/

XPath
XML Path Language (XPath) Version 1.0. J. Clark and S. DeRose. W3C Recommendation, November 1999.
Available at http://www.w3.org/TR/1999/REC-xpath-19991116.

7. Acknowledgements (Informative)

The following people provided valuable feedback that improved the quality of this specification:

- Merlin Hughes, Baltimore
- Thomas Maslen, DSTC
- Paul Denning, MITRE
- Christian Geuer-Pollmann, University Siegen
- Bob Atkinson, Microsoft