XML Inclusions (XInclude) Version 1.0

W3C Working Draft 16 May 2001

This version:
http://www.w3.org/TR/2001/WD-xinclude-20010516
Latest version:
http://www.w3.org/TR/xinclude/
Previous versions:
http://www.w3.org/TR/2000/WD-xinclude-20001026/
http://www.w3.org/TR/2000/WD-xinclude-20000717
http://www.w3.org/TR/2000/WD-xinclude-20000322
http://www.w3.org/TR/1999/NOTE-xinclude-19991123

Editors:
Jonathan Marsh (Microsoft) <jmarsh@microsoft.com>
David Orchard (Jamcracker) <dorchard@jamcracker.com>

Abstract

This document specifies a processing model and syntax for general purpose inclusion. Inclusion is accomplished by merging a number of XML Infosets into a single composite Infoset. Specification of the XML documents (infosets) to be merged and control over the merging process is expressed in XML-friendly syntax (elements, attributes, URI References).

Status of this document

The XML Core Working Group, with this 2001 May 16 XInclude Last Call working draft, invites comment on this specification. The Last Call period begins May 16 and ends June 5 2001.

The W3C Membership and other interested parties are invited to review the specification and report early implementation experience. Please send comments to www-xml-xinclude-comments@w3.org, which is publicly archived.

While we welcome implementation experience reports, the XML Core Working Group will not allow early implementation to constrain its ability to make changes to this specification prior to final release.

For background on this work, please see the XML Activity Statement.

A list of current W3C Recommendations and other technical documents can be found at http://www.w3.org/TR/.

Table of Contents

Appendices

1. Introduction

Many programming languages provide an inclusion mechanism to facilitate modularity. Markup languages also often have need of such a mechanism. This proposal introduces a generic mechanism for merging XML documents (as represented by their information sets) for use by applications that need such a facility. The syntax leverages existing XML constructs - elements, attributes, and URI references.

The requirements used to guide the development of XInclude may be found in the XML Inclusion Proposal W3C Note of 23 November 1999 [XInclude].

1.1. Relationship to XLink

XInclude differs from the linking features described in the XML Linking Language [XLink], specifically links with the attribute value `show="embed"`. Such links provide a media-type independent syntax for indicating that a resource is to be embedded graphically within the display of the document. XLink does not specify a specific processing model, but simply facilitates the detection of links and recognition of associated metadata by a higher level application.

XInclude, on the other hand, specifies a media-type specific (XML into XML) transformation. It defines a specific processing model for merging information sets. XInclude processing occurs at a low level, often by a generic XInclude processor which makes the resulting information set available to higher level applications.

Simple node inclusion as described in this specification differs from transclusion, which preserves contextual information such as style.

1.2. Relationship to XML External Entities

There are a number of differences between XInclude and XML external entities [XML] which make them complementary technologies.

Processing of external entities (as with the rest of DTDs) occurs at parse time. XInclude operates on information sets and thus is orthogonal to parsing.

Declaration of external entities requires a DTD or internal subset. This places a set of dependencies on inclusion, for instance, the syntax for the DOCTYPE declaration requires that the document element be named - orthogonal to inclusion in many cases. Validating parsers must have a complete content model defined. XInclude is orthogonal to validation and the name of the document element.

External entities provide a level of indirection - the external entity must be declared and named, and separately invoked. XInclude uses direct references. Applications which generate XML output incrementally can benefit from not having to pre-declare inclusions.

The syntax for an internal subset is cumbersome to many authors of simple well-formed XML documents. XInclude syntax is based on familiar XML constructs.

1.3. Relationship to DTDs

XInclude defines no relationship to DTD validation. XInclude describes an infoset-to-infoset transformation and not a change in XML 1.0 parsing behavior. XInclude does not define a mechanism for DTD validation of the resulting infoset.

1.4. Relationship to XML Schemas

XInclude defines no relationship to the augmented infosets produced by applying an XML Schema. Such an augmented infoset can be supplied as the input infoset, or such augmentation may be applied to the infoset resulting from the inclusion.

1.5. Relationship to Grammar-Specific Inclusions

Special-purpose inclusion mechanisms have been introduced into specific XML grammars. XInclude provides a generic mechanism for recognizing and processing inclusions, and as such can offer a simpler overall authoring experience, greater performance, and less code redundancy.

2. Terminology

[Definition:] The key words **must**, **must not**, **required**, **shall**, **shall not**, **should**, **should not**, **recommended**, **may**, and **optional** in this specification are to be interpreted as described in RFC 2119 [IETF RFC 2119].

In this document the term *infoset* is used as a synonyn for *information set*.

3. Syntax

XInclude defines a namespace associated with the URI http://www.w3.org/2001/XInclude. The XInclude namespace contains a single element with the local name `include`. For convenience this element is refered to as `xi:include` within this specification.

The `xi:include` element has the following attributes:

href
: A URI Reference indicating the location of the resource to include. This attribute is required.

parse
: An enumeration specifying whether to include the resource as parsed XML or as text. A value of "xml" indicates that the resource must be parsed as XML and the infosets merged. A value of "text" indicates that the resource must be included as the contents of a text node. This attribute is optional.

When omitted, the value of "xml" is implied (even in the absence of a default value declaration). Values other than "xml" and "text" are errors.

encoding

When `parse="text"`, it may be impossible to correctly detect the encoding of the text file. The `encoding` attribute specifies how the resource is to be translated. The value of this attribute is an EncName as defined in XML 1.0 spec., section 4.3.3, rule [81]. The `encoding` attribute has no effect when `parse="xml"`.

Attributes from other namespaces may be placed on the `xi:include` element. Unqualified attribute names are reserved for future versions of this specification, and must be ignored by XInclude 1.0 processors.

The content of the `xi:include` element is not constrained by this specification.

The following (non-normative) XML Schema illustrates the content model of the `xi` namespace:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:xi="http://www.w3.org/2001/XInclude"
           targetNamespace="http://www.w3.org/2001/XInclude">

  <xs:element name="include">
    <xs:complexType mixed="true">
      <xs:attribute name="href" type="xs:anyURI" use="required"/>
      <xs:attribute name="parse">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:enumeration value="xml"/>
            <xs:enumeration value="text"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
      <xs:attribute name="encoding" type="xs:string"/>
      <xs:anyAttribute />
    </xs:complexType>
  </xs:element>

</xs:schema>
```

The following (non-normative) DTD fragment illustrates a sample declaration for the `xi:include` element:

```
<!ELEMENT xi:include EMPTY>
<!ATTLIST xi:include
    xmlns:xi    #FIXED   "http://www.w3.org/2001/XInclude"
    href        CDATA                                  #REQUIRED
    parse       (xml|text)                             "xml"
    encoding    CDATA                                  #IMPLIED
>
```

4. Processing Model

Inclusion as defined in this document is a specific type of XML Information Set [XML Infoset] transformation.

[Definition:] The input for the inclusion transformation consists of a **source infoset**. [Definition:] The output, called the **result infoset**, is a new infoset which merges the source infoset with the infosets of resources identified by URI references appearing in `xi:include` elements. Thus a mechanism to resolve URIs and return the identified resources as infosets is assumed. Well-formed XML entities that do not have defined infosets (e.g. an external entity file with multiple top-level elements) are outside the scope of this specification, either for use as a source infoset or the result infoset.

Inclusion is indicated by the presence of `xi:include` elements in the source infoset. [Definition:] The information items located by the `xi:include` element are called the **included items**. The result infoset is essentially a copy of the source infoset, with each `xi:include` element replaced by its corresponding included items.

4.1. The Include Location

The value of the `href` attribute is interpreted as a IURI reference. [Definition:] An **internationalized URI reference**, or IURI [IURI], is a URI reference that directly uses Unicode characters [Unicode]. IURI references allow a superset of the characters of fully escaped URI references, but must have normal occurrences of the percent sign (%) escaped because it is the character used for escaping in URIs and IURIs.

The base URI for relative IURIs is the base URI of the `xi:include` element as specified in XML Base [XML Base]. [Definition:] The IURI resulting from resolution to absolute IURI form is called the **include location**.

The set of characters allowed in an `href` attribute is the same as for XML, namely [Unicode]. However, some Unicode characters are disallowed from URI references. Thus the disallowed characters in URI references must ultimately be encoded and escaped in order for URI resolver software to be able to handle them.

The disallowed characters include all non-ASCII characters, plus the excluded characters listed in Section 2.4 of [IETF RFC 2396], except for the number sign (#) and percent sign (%) characters and the square bracket characters re-allowed in [IETF RFC 2732]. Disallowed characters are escaped as follows:

1. Each disallowed character is converted to UTF-8 [IETF RFC 2279] as one or more bytes.

2. Any bytes corresponding to a disallowed character are escaped with the URI

escaping mechanism (that is, converted to %HH, where HH is the hexadecimal notation of the byte value).

3. The original character is replaced by the resulting character sequence.

4.2. Included Items when parse="xml"

When `parse="xml"`, the include location is dereferenced and the resource is fetched and an infoset is created.

> **NOTE:** The specifics of how an infoset is created are intentionally unspecified, to allow for flexibility by implementations and to avoid defining a particular processing model for components of the XML architecture. Particulars of whether DTD or XML Schema validation are performed, for example, are not constrained by this spec.

> **NOTE:** The character encodings of the including and included resources can be different. This does not affect the resulting infoset, but may need to be taken into account during any subsequent serialization.

[Definition:] `xi:include` elements in this infoset are recursively processed to create the **acquired infoset**.

Resources that are unavailable for any reason (for example the resource doesn't exist, connection difficulties or security restrictions prevent it from being fetched, the URI scheme isn't a fetchable one, or a syntax error in an XPointer) result in an error. Resources that contain non-well-formed XML result in an error.

When parsing as XML, the fragment part of the URI reference is interpreted as an XPointer [XPointer], regardless of the media type of the resource. The XPointer indicates a subresource as the target for inclusion.

The set of included items is derived from the acquired infoset as follows.

4.2.1. Document Information Items

An include location might identify the **document information item** (for instance, a URI reference without an XPointer, or an XPointer specifically locating the document root. In this case, the set of included items is the **[children]** of the acquired infoset's document information item, except for the **document type declaration information item** child, if one exists.

> **NOTE:** The XML Information Set specification does not provide for preservation of whitespace outside the document element. XInclude makes no further provision to preserve this whitespace.

4.2.2. Multiple Nodes

An include location having an XPointer might identify a subresource that consists of more than a single node. In this case the set of included items is the set of information items from the acquired infoset corresponding to the nodes referred to by the XPointer, in the order in which they appear in the acquired infoset.

If the document (top-level) element in the source infoset is an `xi:include` element, it is an error to attempt to replace it with more than a single element.

4.2.3. Range Locations

An include location having an XPointer might identify a location set that represents a range or a set of ranges.

Each range corresponds to a set of information items in the acquired infoset. [Definition:] An information item is said to be **selected** by a range if it occurs after (in document order) the starting point of the range and before the ending point of the range. [Definition:] An information item is said to be **partially selected** by a range if it contains only the starting point of the range, or only the ending point of the range. By definition, a character information item cannot be partially selected.

The set of included items is the union, in document order with duplicates removed, of the information items either selected or partially selected by the range. The **[children]** property of selected information items is not modified. The **[children]** property of partially selected information items is the set of information items that are in turn either selected or partially selected, and so on.

4.2.4. Element, Comment, and Processing Instruction Information Items

An include location having an XPointer might identify an element node, a comment node, or a processing instruction node, respectively representing an **element information item**, a **comment information item**, or a **processing instruction information item**. In this case the set of included items consists of the information item corresponding to the element, comment, or processing instruction node in the acquired infoset.

4.2.5. Attribute and Namespace Declaration Information Items

An include location having an XPointer might identify an attribute node or a namespace node. Attempting to include such a node is an error.

4.2.6. Inclusion Loops

When recursively processing an `xi:include` element, it is an error to process another `xi:include` element with an include location that has already been processed in the inclusion chain.

In other words, the following are all legal:

- An `xi:include` element may reference the document containing the include element, when `parse="text"`.
- An `xi:include` element may identify a different part of the same local resource.
- Two non-nested `xi:include` elements may identify a resource which itself contains an `xi:include` element.

The following are illegal:

- An `xi:include` element pointing to itself or any ancestor thereof, when `parse="xml"`.
- An `xi:include` element pointing to any include element or ancestor thereof which has already been processed at a higher level.

4.3. Included Items when parse="text"

When `parse="text"`, the include location is dereferenced and the resource is fetched. This resource is treated as plain text and converted to a set of character information items without attempting to parse the resource as XML. This feature facilitates the inclusion of working XML examples, as well as other text-based formats.

The encoding of such a resource is determined by:

- external encoding information, if available, otherwise
- if the media type of the file is `text/xml`, `application/xml`, or matches the conventions `text/*+xml` or `application/*+xml` as described in XML Media Types [IETF RFC 3023], the encoding is recognized as specified in XML 1.0, otherwise
- the value of the `encoding` attribute if one exists, otherwise
- UTF-8.

Byte sequences outside the range allowed by the encoding are an error. Characters that are not permitted in XML documents also are an error.

[Definition:] A range of characters (the **selected range**) may be identified by a fragment identifier. The syntax of the fragment identifier is interpreted using the syntax of the fragment identifier for the media type text/plain. In the absence of a fragment identifier, the selected range contains all the characters in the document.

> **NOTE:** There is currently no standard defining fragment identifiers for the media type text/plain.

The set of characters in the selected range is converted to a set of included items by creating a **character information item** with the **[character code]** set to the character code representing the character in ISO 10646 encoding, and the **[element content whitespace]** set to false.

In accordance with the Character Model [Character Model], when the text format is a Unicode encoding, the XInclude processor must fail the inclusion when the text in the selected range is non-normalized. When transcoding characters to a Unicode encoding from a legacy encoding, a normalizing transcoder must be used.

4.4. Creating the Result Infoset

The result infoset is a copy of the source infoset, with each `xi:include` element processed as follows:

The information item for the `xi:include` element is found. [Definition:] The **[parent]** property of this item refers to an information item called the **include parent**. The **[children]** property of the include parent is modified by replacing the `xi:include` element information item with the included items. The **[parent]** property of each included item is set to the include parent.

Intra-document references within `xi:include` elements must be resolved against the source infoset. The effect of this is that the order in which `xi:include` elements are processed does not affect the result.

In the following example, the second include always points to the first `xi:include` element and not to itself, regardless of the order in which the includes are processed. Thus the result of this inclusion is two copies of `something.xml`, and does not produce an inclusion loop error.

```
<x xmlns:xinclude="http://www.w3.org/2001/XInclude">
  <xi:include href="something.xml"/>
  <xi:include href="#xmlns(xi=http://www.w3.org/2001/XInclude)
                    xpointer(x/xi:include[1])"
              parse="xml"/>
</x>
```

4.4.1. Unparsed Entities

Any **unparsed entity information item** appearing in the **[references]** property of an attribute in the included items is added to the **[unparsed entities]** property of the source infoset's **document information item**, if it is not a duplicate of an existing member.

Unparsed entity items with the same **[name]**, **[system identifier]**, **[public identifier]**, **[declaration base URI]**, **[notation name]**, and **[notation]** are

considered to be duplicate. An application may also be able to detect that unparsed entities are duplicate through other means. For instance, the URI resulting from combining the system identifier and the declaration base URI is the same.

It is possible that the resulting **[unparsed entities]** property contains **unparsed entity information items** with the same name. These can be disambiguated in useage by the **[references]** property on the **attribute information item** referencing the unparsed entity. Serialization of such a result infoset may require application-specific fixup (for instance, renaming).

4.4.2. Notations

Any **notation information item** appearing in the **[references]** property of an attribute in the included items is added to the **[notations]** property of the source infoset's **document information item**, if it is not a duplicate of an existing member. Likewise, any notation referenced by an unparsed entity added as described in **Unparsed Entities**, is added unless it is a duplicate.

Notation items with the same **[name]**, **[system identifier]**, **[public identifier]**, and **[declaration base URI]** are considered to be duplicate. An application may also be able to detect that notations are duplicate through other means. For instance, the URI resulting from combining the system identifier and the declaration base URI is the same.

It is possible that the resulting **[notations]** property contains **[notations]** with the same name. These can be disambiguated in useage by the **[references]** property. Serialization of such a result infoset may require application-specific fixup (for instance, renaming).

4.4.3. Properties Preserved by the Infoset

As an infoset transformation, XInclude operates on the logical structure of XML documents, not on their text serialization. All properties of an information item other than those specifically modified by this proposal are preserved during inclusion.

> **NOTE:** Properties describing conditions about ancestors, descendants, etc. (for example the PSVI **[validity]** property) may be invalidated by inclusion of a document fragment. This specification does not constrain how processors may handle such properties.
>
> **NOTE:** The `xml:lang` and `xml:space` attributes are not treated specially by XInclude.

4.4.3.1. Namespace Declarations

The **[in-scope namespaces]** property ensures that namespace scope is preserved through inclusion. However, after inclusion, the **[namespace**

attributes] property may not provide a full list of namespace declarations necessary to serialize the result. When serializing a result infoset additional namespace declarations might be necessary.

For example, the following document:

```
<foo xmlns:x="uri1">
  <xi:include href="common.xml#xpointer(a/b)"
              xmlns:xi="http://www.w3.org/2001/XInclude"/>
</foo>
```

including a node from common.xml:

```
<a xmlns:x="uri2">
  <b>
    <x:a/>
  </b>
</a>
```

results in a document that could be serialized as:

```
<foo xmlns:x="uri1">
  <b xmlns:x="uri2">
    <x:a/>
  </b>
</foo>
```

Note that the namespace declaration `xmlns:x="uri2"` was added during serialization and is not added to the **[namespace attributes]** property of the `b` element by the XInclude processor. When this infoset is serialized and then reparsed, the resulting infoset may differ from the infoset resulting from processing inclusions in that the **[namespace attributes]** property reflects the addition of these attributes.

Serialization of the result infoset, and specifically the placement and form of namespace declarations, is not constrained by this specification.

4.4.3.2. Base URI

The base URI property of the acquired infoset is not changed as a result of merging the infoset, and remains unchanged after merging. Thus relative URI references in the included infoset resolve to the same URI despite being included into a document with a potentially different base URI in effect. A serialized result infoset may need to add `xml:base` attributes to indicate this fact.

5. Conformance

5.1. Markup Conformance

An **element information item** conforms to this specification if it meets the structural requirements for include elements defined in this specification. This specification imposes no particular constraints on DTDs or XML Schemas; conformance applies only to elements and attributes.

5.2. Application Conformance

An application conforms to XInclude if it:

- supports XML 1.0, XML Namespaces, the XML Information Set, and XML Base
- observes the mandatory conditions (must) set forth in this specification, and for any optional conditions (should and may) it chooses to observe, observes them in the way prescribed
- performs markup conformance testing according to all the conformance constraints appearing in this specification.

5.3. XML Information Set Conformance

This specification conforms to the XML Information Set [XML Infoset]. The following information items must be present in the input infosets to enable correct processing:

- **Document Information Items** with **[children]** and **[base URI]** properties.
- **Element Information Items** with **[namespace name]**, **[local name]**, **[children]**, **[attributes]**, **[base URI]** and **[parent]** properties.
- **Attribute Information Items** with **[namespace name]**, **[local name]** and **[normalized value]** properties.

Information items and properties present in the input infosets are either processed according to the stated requirements of this specification, or are passed through the include transformation unchanged. Additionally, XInclude processing may generate the following kinds of information items in the result:

- **Character Information Items** with **[character code]**, **[element content whitespace]** and **[parent]** properties.

Appendices

A. References

Character Model
Martin J. Dürst, François Yergeau, Misha Wolf, Asmus Freytag, Tex Texin. *Character Model for the World Wide Web 1.0*. World Wide Web Consortium, 2001. (See http://www.w3.org/TR/charmod/.)

IETF RFC 2119
RFC 2119: Key words for use in RFCs to Indicate Requirement Levels. Internet Engineering Task Force, 1997. (See http://www.ietf.org/rfc/rfc2119.txt.)

IETF RFC 2279
RFC 2279: UTF-8, a transformation format of ISO 10646. Internet Engineering Task Force, 1998. (See http://www.ietf.org/rfc/rfc2279.txt.)

IETF RFC 2396
RFC 2396: Uniform Resource Identifiers. Internet Engineering Task Force, 1995. (See http://www.ietf.org/rfc/rfc2396.txt.)

IETF RFC 2732
RFC 2732: Format for Literal IPv6 Addresses in URL's. Internet Engineering Task Force, 1999. (See http://www.ietf.org/rfc/rfc2732.txt.)

IETF RFC 3023
RFC 3023: XML Media Types. Internet Engineering Task Force, 2001. (See http://www.ietf.org/rfc/rfc3023.txt.)

Unicode
The Unicode Consortium. *The Unicode Standard.*(See http://www.unicode.org/unicode/standard/standard.html.)

XML
Tim Bray, Jean Paoli, and C.M. Sperberg-McQueen, editors. *Extensible Markup Language (XML) 1.0.* World Wide Web Consortium, 1998. (See http://www.w3.org/TR/REC-xml.)

XML Base
Jonathan Marsh, editor. *XML Base*. World Wide Web Consortium, 1999. (See http://www.w3.org/TR/xmlbase/.)

XML Infoset
John Cowan and David Megginson, editors. *XML Information Set*. World Wide Web Consortium, 1999. (See http://www.w3.org/TR/xml-infoset/.)

XML Names
Tim Bray, Dave Hollander, and Andrew Layman, editors. *Namespaces in XML*. Textuality, Hewlett-Packard, and Microsoft. World Wide Web Consortium, 1999. (See http://www.w3.org/TR/REC-xml-names/.)

XPointer
Steve DeRose, Ron Daniel, Eve Maler, editors. *XML Pointer Language (XPointer)*. World Wide Web Consortium, 1999. (See http://www.w3.org/TR/xptr.)

B. References *(Non-Normative)*

IURI
Internationalized URIs. Internet Engineering Task Force, 2000. Expired

Internet-Draft. (See http://www.w3.org/International/2000/03/draft-masinter-url-i18n-05.txt.)

XInclude

Jonathan Marsh, David Orchard, editors. *XML Inclusion Proposal (XInclude).* World Wide Web Consortium, 1999. (See http://www.w3.org/TR/1999/NOTE-xinclude-19991123.)

XLink

Steve DeRose, Eve Maler, David Orchard, and Ben Trafford, editors. *XML Linking Language (XLink).* World Wide Web Consortium, 2000. (See http://www.w3.org/TR/xlink/.)

C. Examples *(Non-Normative)*

C.1. Basic Inclusion Example

The following XML document contains an `xi:include` element which points to an external document.

```
<?xml version='1.0'?>
<document xmlns:xi="http://www.w3.org/1999/XML/xinclude">
  <p>120 Mz is adequate for an average home user.</p>
  <xi:include href="disclaimer.xml"/>
</document>
```

disclaimer.xml contains:

```
<?xml version='1.0'?>
<disclaimer>
  <p>The opinions represented herein represent those of the individual
  and should not be interpreted as official policy endorsed by this
  organization.</p>
</disclaimer>
```

The infoset resulting from resolving inclusions on this document is the same as that of the following document:

```
<?xml version='1.0'?>
<document xmlns:xi="http://www.w3.org/1999/XML/xinclude">
  <p>120 Mz is adequate for an average home user.</p>
  <disclaimer>
  <p>The opinions represented herein represent those of the individual
  and should not be interpreted as official policy endorsed by this
  organization.</p>
</disclaimer>
</document>
```

C.2. Range Inclusion Example

The following illustrates the results of including a range specified by an XPointer.

```
<?xml version='1.0'?>
<document>
  <p>The relevant excerpt is:</p>
  <quotation>
    <include xmlns="http://www.w3.org/1999/XML/xinclude"
       href="source.xml#xpointer(string-range(chapter/p[1],'Sentence 2'
             range-to(string-range(chapter/p[2]/i,'3.',1,11)))"/>
  </quotation>
</document>
```

source.xml contains:

```
<chapter>
  <p>Sentence 1.  Sentence 2.</p>
  <p><i>Sentence 3.  Sentence 4.</i>  Sentence 5.</p>
</chapter>
```

The infoset resulting from resolving inclusions on this document is the same as that of the following document:

```
<?xml version='1.0'?>
<document>
  <p>The relevant excerpt is:</p>
  <quotation>
    <p>Sentence 2.</p>
  <p><i>Sentence 3.</i></p>
  </quotation>
</document>
```

C.3. Textual Inclusion Example

The following XML document includes a "working example" into a document.

```
<?xml version='1.0'?>
<document xmlns:xi="http://www.w3.org/1999/XML/xinclude">
  <p>The following is the source of the "data.xml" file:</p>
  <example><xi:include href="data.xml" parse="text"/></example>
</document>
```

data.xml contains:

```
<?xml version='1.0'?>
<data>
  <item><![CDATA[Brooks & Shields]]></item>
</data>
```

The infoset resulting from resolving inclusions on this document is the same as that of the following document:

```
<?xml version='1.0'?>
<document xmlns:xi="http://www.w3.org/1999/XML/xinclude">
  <p>The following is the source of the "data.xml" file:</p>
  <example>&lt;?xml version='1.0'?&gt;
&lt;data&gt;
  &lt;item&gt;&lt;![CDATA[Brooks &amp; Shields]]&gt;&lt;/item&gt;
&lt;/data&gt;</example>
</document>
```