XML Linking Language (XLink) Version 1.1

W3C Recommendation 06 May 2010

This version:
http://www.w3.org/TR/2010/REC-xlink11-20100506/
Latest version:
http://www.w3.org/TR/xlink11/
Previous versions:
http://www.w3.org/TR/2010/PR-xlink11-20100225/ http://www.w3.org/TR/2008/WD-xlink11-20080331/
Editors:
Steve DeRose, Brown University Scholarly Technology Group
Eve Maler, Sun Microsystems
David Orchard, Jamcracker
Norman Walsh, Mark Logic Corporation - Version 1.1

Please refer to the **errata** for this document, which may include normative corrections.

See also **translations**.

This document is also available in these non-normative formats: XML.

Abstract

This specification defines the XML Linking Language (XLink) Version 1.1, which allows elements to be inserted into XML documents in order to create and describe links between resources. It uses XML syntax to create structures that can describe links similar to the simple unidirectional hyperlinks of today's HTML, as well as more sophisticated links.

Status of this Document

This section describes the status of this document at the time of its publication. Other documents may supersede this document. A list of current W3C publications and the latest revision of this technical report can be found in the W3C technical reports index at http://www.w3.org/TR/.

This document is a W3C Recommendation. It implements all of the XLink 1.1 requirements documented in [Extending XLink 1.0]. This document is a product of the XML Core Working Group as part of the W3C XML Activity.

This edition supersedes the previous W3C Recommendation of 27 June 2001.

Please report errors in this document to the public www-xml-linking-comments@w3.org mailing list; public archives are available.

There is an Implementation Report for XLink 1.1. A Test Suite is maintained to help assessing the conformance to this specification.

This document has been reviewed by W3C Members, by software developers, and by other W3C groups and interested parties, and is endorsed by the Director as a W3C Recommendation. It is a stable document and may be used as reference material or cited 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 document was produced by a group operating under the 5 February 2004 W3C Patent Policy. W3C maintains a public list of any patent disclosures made in connection with the deliverables of the group; that page also includes instructions for disclosing a patent. An individual who has actual knowledge of a patent which the individual believes contains Essential Claim(s) must disclose the information in accordance with section 6 of the W3C Patent Policy.

Table of Contents

---

1 Introduction

This specification defines the XML Linking Language (XLink), which allows elements to be inserted into XML documents in order to create and describe links between resources.

XLink provides a framework for creating both basic unidirectional links and more complex linking structures. It allows XML documents to:

- Assert linking relationships among more than two resources
- Associate metadata with a link
- Express links that reside in a location separate from the linked resources

An important application of XLink is in hypermedia systems that have hyperlinks. A simple case of a hyperlink is an HTML `A` element, which has these characteristics:

- The hyperlink uses IRIs as its locator technology.
- The hyperlink is expressed at one of its two ends.
- The hyperlink identifies the other end (although a server may have great freedom in finding or dynamically creating that destination).
- Users can initiate traversal only from the end where the hyperlink is expressed to the other end.
- The hyperlink's effect on windows, frames, go-back lists, style sheets in use, and so on is determined by user agents, not by the hyperlink itself. For example, traversal of `A` links normally replaces the current view, perhaps with a user option to open a new window.

This set of characteristics is powerful, but the model that underlies them limits the range of possible hyperlink functionality. The model defined in this specification shares with HTML the use of IRI technology, but goes beyond HTML in offering features, previously available only in dedicated hypermedia systems, that make hyperlinking more scalable and flexible. Along with providing linking data structures, XLink provides a minimal link behavior model; higher-level applications layered on XLink will often specify alternate or more sophisticated rendering and processing treatments.

Integrated treatment of specialized links used in other technical domains, such as foreign keys in relational databases and reference values in programming languages, is outside the scope of this specification.

For languages, such as [CSS], that wish to identify hypertext links in a document, we suggest that any local element from which XLink specifies that traversal is possible, and which the application treats as if it specified `actuate="onRequest"`, be treated as a hyperlink source anchor.

1.1 Origin and Goals

The design of XLink has been informed by knowledge of established hypermedia systems and standards. The following standards have been especially influential:

- *HTML* [HTML]: Defines several element types that represent links.
- *HyTime* [ISO/IEC 10744]: Defines inline and inbound and third-party link structures and some semantic features, including traversal control and presentation of objects.
- *Text Encoding Initiative Guidelines* [TEI]: Provides structures for creating links, aggregate objects, and link collections.

Many other linking systems have also informed the design of XLink, especially [Dexter], [FRESS], [OHS], [MicroCosm], and [Intermedia].

See the XLink Requirements Document [XLREQ] for a thorough explanation of requirements for the design of XLink.

2 XLink Concepts

This section describes the terms and concepts that are essential to understanding XLink, without discussing the syntax used to create XLink constructs. A few additional terms are introduced in later parts of this specification.

2.1 Links and Resources

[Definition: An XLink **link** is an explicit relationship between resources or portions of resources.] [Definition: It is made explicit by an XLink **linking element**, which is an XLink-conforming XML element that asserts the existence of a link.] There are six XLink elements; only two of them are considered linking elements. The others provide various pieces of information that describe the characteristics of a link. (The term "link" as used in this specification refers only to an XLink link, though nothing prevents non-XLink constructs from serving as links.)

The notion of resources is universal to the World Wide Web. [Definition: As discussed in [RFC 3986], a **resource** is any addressable unit of information or service.] Examples include files, images, documents, programs, and query results. The means used for addressing a resource is a IRI (Internationalized Resource Identifier) reference (described more in **5.4 Locator Attribute (href)**). It is possible to address a portion of a resource. For example, if the whole resource is an XML document, a useful portion of that resource might be a particular element inside the document. Following a link to it might result, for example, in highlighting that element or scrolling to that point in the document.

[Definition: When a link associates a set of resources, those resources are said to **participate** in the link.] Even though XLink links must appear in XML documents, they are able to associate all kinds of resources, not just XML-encoded ones.

One of the common uses of XLink is to create hyperlinks. [Definition: A **hyperlink** is a link that is intended primarily for presentation to a human user.] Nothing in XLink's design, however, prevents it from being used with links that are intended solely for consumption by computers.

2.2 Arcs, Traversal, and Behavior

[Definition: Using or following a link for any purpose is called **traversal**.] Even though some kinds of link can associate arbitrary numbers of resources, traversal always involves a pair of resources (or portions of them); [Definition: the source from which traversal is begun is the **starting resource**] and [Definition: the destination is the **ending resource**]. Note that the term "resource" used in this fashion may at times apply to a resource portion, not a whole resource.

[Definition: Information about how to traverse a pair of resources, including the direction of traversal and possibly application behavior information as well, is called an **arc**]. If two arcs in a link specify the same pair of resources, but they switch places as starting and ending resources, then the link is multidirectional, which is not the same as merely "going back" after traversing a link.

2.3 Resources in Relation to the Physical Location of a Linking Element

[Definition: A **local resource** is an XML element that participates in a link by virtue of having as its parent, or being itself, a linking element]. [Definition: Any resource or resource portion that participates in a link by virtue of being addressed with an IRI reference is considered a **remote resource**, even if it is in the same XML document as the link, or even inside the same linking element.] Put another way, a local resource is specified "by value," and a remote resource is specified "by reference."

[Definition: An arc that has a local starting resource and a remote ending resource goes **outbound**, that is, away from the linking element.] (Examples of links with such an arc are the HTML `A` element, HyTime "clinks," and Text Encoding Initiative `XREF` elements.) [Definition: If an arc's ending resource is local but its starting resource is remote, then the arc goes **inbound**.] [Definition: If neither the starting resource nor the ending resource is local, then the arc is a **third-party** arc.] Though it is not required, any one link typically specifies only one kind of arc throughout, and thus might be referred to as an inbound, outbound, or third-party link.

To create a link that emanates from a resource to which you do not have (or choose not to exercise) write access, or from a resource that offers no way to embed linking constructs, it is necessary to use an inbound or third-party arc. When such arcs are used, the requirements for discovery of the link are greater than for outbound arcs. [Definition: Documents containing collections of inbound and third-party links are called link databases, or **linkbases**.]

3 XLink Processing and Conformance

This section details processing and conformance requirements on XLink applications and markup.

[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].]

3.1 Processing Dependencies

XLink processing depends on [XML], [XML Names], [XML Base], [RFC 3987], and [CharMod Fundamentals].

3.2 Markup Conformance

An XML element conforms to XLink if:

1. It has a `type` attribute from the XLink namespace whose value is one of "simple", "extended", "locator", "arc", "resource", "title", or "none", and it adheres to the conformance constraints imposed by the chosen XLink element type, as prescribed in this specification, *or*

2. it does not have a `type` attribute from the XLink namespace and it adheres to the conformance constraints imposed by the XLink simple element type, as prescribed in this specification.

3. It does not have any attributes in the XLink namespace other than `type`, `href`, `role`, `arcrole`, `title`, `show`, `actuate`, `label`, `from`, and `to`.

4. Content conforming to XLink must conform to [CharMod Fundamentals].

This specification imposes no particular constraints on schemas; conformance applies only to elements and attributes.

3.3 Application Conformance

An XLink application is any software module that interprets well-formed XML documents containing XLink elements and attributes, or XML information sets [XIS] containing information items and properties corresponding to XLink elements and attributes. (This document refers to elements and attributes, but all specifications herein apply to their information set equivalents as well.)

XLink defines two conformance levels for an XLink application, simple and full.

3.3.1 Full Conformance

An application satisfies the constraints of full conformance if:

1. It observes the mandatory conditions ("must") for applications set forth in this specification, and
2. for any recommended or optional conditions ("should" and "may") it chooses to observe, it observes them in the way prescribed, and
3. it performs markup conformance testing according to all the conformance constraints appearing in this specification.
4. It applies XLink semantics only to those elements which satisfy the markup conformance criteria outlined in **3.2 Markup Conformance**.
5. Applications implementing XLink must conform to [CharMod Fundamentals].

3.3.2 Simple Conformance

An application satisfies the constraints of simple conformance if it is fully conformant with respect to simple links. In other words:

- The processor may ignore any link which specifies an `xlink:type` other than “simple”.
- If the `xlink:href` attribute is specified and the `xlink:type` attribute is not specified, the element must be processed as if `xlink:type` specified “simple”.

An application which claims simple conformance may ignore all other XLink elements.

4 XLink Markup Design

This section describes the design of XLink's markup vocabulary.

Link markup needs to be recognized reliably by XLink applications in order to be traversed and handled properly. XLink uses the mechanism described in the Namespaces in XML Recommendation [XML Names] to accomplish recognition of the constructs in the XLink vocabulary.

The XLink namespace defined by this specification has the following URI:

```
http://www.w3.org/1999/xlink
```

As dictated by [XML Names], the use of XLink elements and attributes requires declaration of the XLink namespace. For example, the following declaration would make the prefix `xlink` available within the `myElement` element to represent the XLink namespace:

```
<myElement
  xmlns:xlink="http://www.w3.org/1999/xlink">
  ...
</myElement>
```

Note:

Most code examples in this specification do not show an XLink namespace declaration. The `xlink` prefix is used throughout to stand for the declaration of the XLink namespace on elements in whose scope the so-marked attribute appears (on the same element that bears the attribute or on some ancestor element), whether or not an XLink namespace declaration is present in the example.

XLink's namespace provides **global attributes** for use on elements that are in any arbitrary namespace. The global attributes are `type`, `href`, `role`, `arcrole`, `title`, `show`, `actuate`, `label`, `from`, and `to`. All other attributes, and all elements, in the XLink namespace are reserved. Document creators use the XLink global attributes to make the elements in their own namespace, or even in a namespace they do not control, recognizable as XLink elements. The `type` attribute indicates the XLink element type (simple, extended, locator, arc, resource, or title); the element type dictates the XLink-imposed constraints that such an element must follow and the behavior of XLink applications on encountering the element.

Following is an example of a `crossReference` element from a non-XLink namespace that has XLink global attributes:

```
<my:crossReference
  xmlns:my="http://example.com/"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xlink:type="simple"
  xlink:href="students.xml"
  xlink:role="http://www.example.com/linkprops/studentlist"
  xlink:title="Student List"
  xlink:show="new"
  xlink:actuate="onRequest">
Current List of Students
</my:crossReference>
```

Using global attributes always requires the use of namespace prefixes on the individual attributes.

In [XML Linking Language (XLink) Version 1.0], XLink elements are identified by the presence of an `xlink:type` attribute. In XLink 1.1, XLink elements are identified by the presence of either an `xlink:type` attribute or an `xlink:href` attribute:

- If an element has an `xlink:type` attribute, then that attribute must have one of the following values: “simple”, “extended”, “locator”, “arc”, “resource”, or “title” and the element must adhere to the conformance constraints imposed by that XLink element type.
- If an element has an `xlink:href` attribute but *does not* have an `xlink:type` attribute, then it is treated exactly as if it had an `xlink:type` attribute with the value “simple”.

4.1 XLink Attribute Usage Patterns

While the XLink attributes are considered global by virtue of their use of the namespace mechanism, their allowed combinations on any one XLink element type depend greatly on the value of the special `type` attribute (see **5.3 XLink Element Type Attribute (type)** for more information) for the element on which they appear. The conformance constraint notes in this specification detail their allowed usage patterns. Following is a summary of the element types (columns) on which the global attributes (rows) are allowed, with an indication of whether a value is required (R) or optional (O):

	simple	**extended**	**locator**	**arc**	**resource**	**title**
type	O*	R	R	R	R	R
href	O*		R			
role	O	O	O		O	
arcrole	O			O		
title	O	O	O	O	O	
show	O			O		
actuate	O			O		
label			O		O	
from				O		
to				O		
* At least one of `type` or `href` must be specified.						

(See also **B Sample DTD**, **C Sample XML Schema**, and **D Sample RELAX NG Grammar** for examples of non-normative schemas that illustrate the allowed patterns of attributes.)

This specification uses the convention "*xxx*-type element" to refer to elements that must adhere to a named set of constraints associated with an XLink element type, no matter what name the element actually has. For example, "`locator`-type element" would refer to all of the following elements:

```
<locator xlink:type="locator" ... />
<loc xlink:type="locator" ... />
<my:pointer xlink:type="locator" ... />
```

4.2 XLink Element Type Relationships

Various XLink element types have special meanings dictated by this specification when they appear as direct children of other XLink element types. Following is a summary of the child element types that play a significant role in particular parent element types. (Other combinations are not conformant.)

Parent type	Significant child types
simple	none
extended	locator, arc, resource, title
locator	title
arc	title
resource	none
title	none

4.3 Attribute Value Defaulting

Using XLink potentially involves using a large number of attributes for supplying important link information. In cases where the values of the desired XLink attributes are unchanging across individual instances in all the documents of a certain type, attribute value defaults (fixed or not) may be added to a DTD or schema so that the attributes do not have to appear physically on element start-tags. For example, if attribute defaults were provided for the `xmlns:xlink`, `xmlns:my`, `type`, `show`, and `actuate` attributes in the example in the introduction to **4 XLink Markup Design**, the example would look as follows:

```
<my:crossReference
  xlink:href="students.xml"
  xlink:role="http://www.example.com/linkprops/studentlist"
  xlink:title="Student List">
Current List of Students
</my:crossReference>
```

Information sets that have been created under the control of a DTD have all attribute values filled in. Note, however, that parsers are not required to process the external subset. Applications, such as web browsers, that do not process the external subset will not be aware of any default values identified therein.

Note also that using the attribute value default technique to specify the XLink namespace declaration has no equivalent in [RELAX NG], [XML Schema Part 1: Structures], or other modern schema languages. While it can be used when DTD-informed parsing is performed, it poses an interoperability risk and should be used with care.

4.4 Integrating XLink Usage with Other Markup

This specification defines only attributes and attribute values in the XLink namespace. There is no restriction on using non-XLink attributes alongside XLink attributes. In addition, most XLink attributes are optional and the choice of simple or extended link is up to the markup designer or document creator, so a DTD or other schema that uses XLink features need not use or declare the entire set of XLink's attributes. Finally, while this specification identifies the minimum constraints on XLink markup, schemas that use XLink are free to tighten these constraints. The use of XLink does not absolve a valid document from conforming to the constraints expressed in its governing schema.

Following is an example of a `crossReference` element with both XLink and non-XLink attributes:

```
<my:crossReference
  xmlns:my="http://example.com/"
  my:lastEdited="2000-06-10"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xlink:type="simple"
  xlink:href="students.xml">
Current List of Students
</my:crossReference>
```

4.5 Using XLink with Legacy Markup

Because XLink's global attributes require the use of namespace prefixes, non-XLink-based links in legacy documents generally do not serve as conforming XLink constructs as they stand, even if attribute value defaulting is used. For example, XHTML 1.0 has an `a` element with an `href` attribute, but because the attribute is a local one attached to the `a` element in the XHTML namespace, it is not the same as an `xlink:href` global attribute in the XLink namespace.

This specification defines XLink markup, that is, what combinations XLink attributes and what arrangements of elements with XLink attributes are conformant. It also defines the XLink semantics of such conformant markup. When XLink is used on elements that have other attributes or related elements with linking semantics, XLink can not and does not attempt to define the semantics of such combinations.

If an element allows such possibly conflicting markup to occur, it should specify the semantics of the result.

5 XLink Elements and Attributes

XLink offers two kinds of links:

Extended links

Extended links offer full XLink functionality, such as inbound and third-party arcs, as well as links that have arbitrary numbers of participating resources. As a result, their structure can be fairly complex, including elements for pointing to remote resources, elements for containing local resources, elements for specifying arc traversal rules, and elements for specifying human-readable resource and arc titles.

XLink defines a way to give an extended link special semantics for finding linkbases; used in this fashion, an extended link helps an XLink application process other links.

Simple links

Simple links offer shorthand syntax for a common kind of link, an outbound link with exactly two participating resources (into which category HTML-style `A` and `IMG` links fall). Because simple links offer less functionality than extended links, they have no special internal structure.

While simple links are conceptually a subset of extended links, they are syntactically different. For example, to convert a simple link into an extended link, several structural changes would be needed.

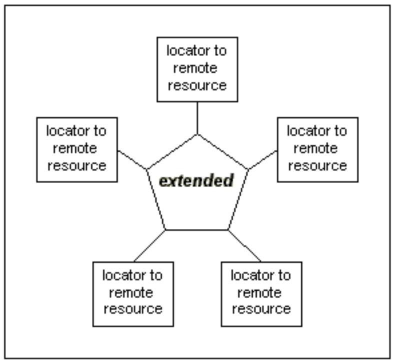
The following sections define the XLink elements and attributes.

5.1 Extended Links (`extended`-Type Element)

[Definition: An **extended link** is a link that associates an arbitrary number of resources. The participating resources may be any combination of remote and local.]

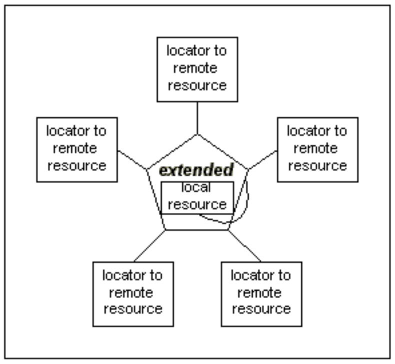
The only kind of link that is able to have inbound and third-party arcs is an extended link. Typically, extended linking elements are stored separately from the resources they associate (for example, in entirely different documents). Thus, extended links are important for situations where the participating resources are read-only, or where it is expensive to modify and update them but inexpensive to modify and update a separate linking element, or where the resources are in formats with no native support for embedded links (such as many multimedia formats).

The following diagram shows an extended link that associates five remote resources. This could represent, for example, information about a student's course load: one resource being a description of the student, another being a description of the student's academic advisor, two resources representing courses that the student is attending, and the last resource representing a course that the student is auditing.


locator to remote resource
locator to remote resource
locator to remote resource
extended
locator to remote resource
locator to remote resource

Without the extended link, the resources might be entirely unrelated; for example, they might be in five separate documents. The lines emanating from the extended link represent the association it creates among the resources. However, notice that the lines do not have directionality. Directionality is expressed with traversal rules; without such rules being provided, the resources are associated in no particular order, with no implication as to whether and how individual resources are accessed.

The following diagram shows an extended link that associates five remote resources and one local resource (a special element inside the extended link element). This could represent the same sort of course-load example as described above, with the addition of the student's grade point average stored locally. Again, the lines represent mere association of the six resources, without traversal directions or behaviors implied.


locator to remote resource
locator to remote resource
locator to remote resource
extended
local resource
locator to remote resource
locator to remote resource

The XLink element type for extended links is any element with an attribute in the XLink namespace called `type` with a value of "extended".

The `extended`-type element may contain a mixture of the following elements in any order, possibly along with other content and markup:

- `locator`-type elements that address the remote resources participating in the link
- `arc`-type elements that provide traversal rules among the link's participating resources
- `title`-type elements that provide human-readable labels for the link
- `resource`-type elements that supply local resources that participate in the link

It is not an error for an `extended`-type element to associate fewer than two resources. If the link has only one participating resource, or none at all, it is simply untraversable. Such a link may still be useful, for example, to associate properties with a single resource by means of XLink attributes, or to provide a placeholder for link information that will be populated eventually.

Subelements of the `simple` or `extended` type anywhere inside a parent `extended`-type element are not conformant. Subelements of the `locator`, `arc`, or `resource` type that are not direct children of an `extended`-type element are not conformant.

The `extended`-type element may have the semantic attributes `role` and `title` (see **5.5 Semantic Attributes (role, arcrole, and title)**). They supply semantic information about the link as a whole; the `role` attribute indicates a property that the entire link has, and the `title` attribute indicates a human-readable description of the entire link. It is not conformant for other XLink attributes to be present on the element. If both a `title` attribute and one or more `title`-type elements are present, they have no XLink-specified relationship; a higher-level application built on XLink will likely want to specify appropriate treatment (for example, precedence) in this case.

Example: Sample `extended`-Type Element Declarations and Instance

Following is a non-normative set of DTD declarations for an `extended`-type element and its subelements. Parts of this example are reused throughout this specification. Note that the `type` attribute and some other attributes are defaulted in the DTD in order to highlight the attributes that are changing on a per-instance basis.

```
<!ELEMENT courseload ((tooltip|person|course|gpa|go)*)>
<!ATTLIST courseload
  xmlns:xlink     CDATA           #FIXED "http://www.w3.org/1999/xlink"
  xlink:type      (extended)      #FIXED "extended"
  xlink:role      CDATA           #IMPLIED
  xlink:title     CDATA           #IMPLIED>

<!ELEMENT tooltip ANY>
<!ATTLIST tooltip
  xlink:type      (title)         #FIXED "title"
  xml:lang        CDATA           #IMPLIED>
```

```
<!ELEMENT person EMPTY>
<!ATTLIST person
  xlink:type      (locator)       #FIXED "locator"
  xlink:href      CDATA           #REQUIRED
  xlink:role      CDATA           #IMPLIED
  xlink:title     CDATA           #IMPLIED
  xlink:label     NMTOKEN         #IMPLIED>

<!ELEMENT course EMPTY>
<!ATTLIST course
  xlink:type      (locator)       #FIXED "locator"
  xlink:href      CDATA           #REQUIRED
  xlink:role      CDATA           #FIXED "http://www.example.com/linkprops/course"
  xlink:title     CDATA           #IMPLIED
  xlink:label     NMTOKEN         #IMPLIED>

<!-- GPA = "grade point average" -->
<!ELEMENT gpa ANY>
<!ATTLIST gpa
  xlink:type      (resource)      #FIXED "resource"
  xlink:role      CDATA           #FIXED "http://www.example.com/linkprops/gpa"
  xlink:title     CDATA           #IMPLIED
  xlink:label     NMTOKEN         #IMPLIED>

<!ELEMENT go EMPTY>
<!ATTLIST go
  xlink:type      (arc)           #FIXED "arc"
  xlink:arcrole   CDATA           #IMPLIED
  xlink:title     CDATA           #IMPLIED
  xlink:show      (new
                  |replace
                  |embed
                  |other
                  |none)          #IMPLIED
  xlink:actuate   (onLoad
                  |onRequest
                  |other
                  |none)          #IMPLIED
  xlink:from      NMTOKEN         #IMPLIED
  xlink:to        NMTOKEN         #IMPLIED>
```

Following is how XML elements using these declarations might look.

```
<courseload>

  <tooltip>Course Load for Pat Jones</tooltip>

  <person
    xlink:href="students/patjones62.xml"
    xlink:label="student62"
    xlink:role="http://www.example.com/linkprops/student"
    xlink:title="Pat Jones" />

  <person
    xlink:href="profs/jaysmith7.xml"
    xlink:label="prof7"
    xlink:role="http://www.example.com/linkprops/professor"
    xlink:title="Dr. Jay Smith" />
  <!-- more remote resources for professors, teaching assistants, etc. -->

  <course
    xlink:href="courses/cs101.xml"
    xlink:label="CS-101"
    xlink:title="Computer Science 101" />
  <!-- more remote resources for courses, seminars, etc. -->

  <gpa xlink:label="PatJonesGPA">3.5</gpa>

  <go
    xlink:from="student62"
    xlink:to="PatJonesGPA"
    xlink:show="new"
    xlink:actuate="onRequest"
    xlink:title="Pat Jones's GPA" />
  <go
    xlink:from="CS-101"
    xlink:arcrole="http://www.example.com/linkprops/auditor"
    xlink:to="student62"
    xlink:show="replace"
    xlink:actuate="onRequest"
    xlink:title="Pat Jones, auditing the course" />
  <go
    xlink:from="student62"
    xlink:arcrole="http://www.example.com/linkprops/advisor"
    xlink:to="prof7"
    xlink:show="replace"
    xlink:actuate="onRequest"
    xlink:title="Dr. Jay Smith, advisor" />

</courseload>
```

5.1.1 Local Resources for an Extended Link (`resource`-Type Element)

An extended link indicates its participating local resources by means of special subelements that appear inside the extended link. An entire subelement, together with all of its contents, makes up a local resource.

The XLink element for local resources is any element with an attribute in the XLink namespace called `type` with a value of "resource".

The `resource`-type element may have any content; whatever content is present has no XLink-specified relationship to the link. It is possible for a `resource`-type element to have no content; in cases where it serves as a starting resource expected to be traversed on request, interactive XLink applications will typically generate some content in order to give the user a way to initiate the traversal. If a `resource`-type element has anything other than an `extended`-type element for a parent, the `resource`-type element is not conformant.

The `resource`-type element may have the semantic attributes `role` and `title` (see **5.5 Semantic Attributes (role, arcrole, and title)**) and the traversal attribute `label` (see **5.7 Traversal Attributes (label, from, and to)**). The semantic attributes supply information about the resource in generic terms, outside of the context of a particular arc that leads to it; the `role` attribute indicates a property of the resource, and the `title` attribute indicates a human-readable description of the resource. The `label` attribute provides a way for an `arc`-type element to refer to it in creating a traversal arc.

Example: Sample `resource`-Type Element Declarations and Instance

Following is a non-normative set of RELAX NG declarations (in the compact syntax) for a `resource`-type element.

```
namespace a = "http://relaxng.org/ns/compatibility/annotations/1.0"
namespace xlink = "http://www.w3.org/1999/xlink"

gpa = element gpa { attlist.gpa, any }

attlist.gpa &=
  [ a:defaultValue = "resource" ] attribute xlink:type { "resource" }?,
  [ a:defaultValue = "http://www.example.com/linkprops/gpa" ]
  attribute xlink:role {
    string "http://www.example.com/linkprops/gpa"
  }?,
  attribute xlink:title { text }?,
  attribute xlink:label { xsd:NMTOKEN }?

any =
  (element * {
     attribute * { text }*,
     any
   }
   | text)*
```

Following is how an XML element using these declarations might look.

```
  <gpa xlink:type="resource"
     xlink:role="http://www.example.com/linkprops/gpa"
     xlink:label="PatJonesGPA">3.5</gpa>
```

5.1.2 Remote Resources for an Extended Link (`locator`-Type Element)

An extended link indicates remote resources that participate in it by means of locator elements.

The XLink element for locators is any element with an attribute in the XLink namespace called `type` with a value of "locator".

The `locator`-type element may have any content. Other than `title`-type elements that are direct children (see **5.1.4 Titles for Extended Links, Locators, and Arcs (title-Type Element)**), whatever content is present has no XLink-specified relationship to the link. If a `locator`-type element contains nested XLink elements, such contained elements have no XLink-specified relationship to the parent link. If a `locator`-type element has anything other than an `extended`-type element for a parent, the `locator`-type element is not conformant.

Constraint: Attributes on Locator Element

The `locator`-type element must have the locator attribute (see **5.4 Locator Attribute (href)**). The locator attribute (`href`) must have a value supplied.

The `locator`-type element may have the semantic attributes `role` and `title` (see **5.5 Semantic Attributes (role, arcrole, and title)**) and the traversal attribute `label` (see **5.7 Traversal Attributes (label, from, and to)**). The locator attribute provides an IRI reference that identifies a remote resource. The semantic attributes supply information about the resource in generic terms, outside of the context of a particular arc that leads to it; the `role` attribute indicates a property that the resource has, and the `title` attribute indicates a human-readable description of the resource. The `label` attribute provides a way for an `arc`-type element to refer to it in creating a traversal arc.

Note:

A `locator`-type element, by itself, does not constitute a link just because it has a locator (`href`) attribute; unlike a `simple`-type element, it does not create an XLink-governed association between itself and the referenced resource.

Example: Sample `locator`-Type Element Declarations and Instance

Following is a non-normative set of XML Schema declarations for a `locator`-type element.

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:xlink="http://www.w3.org/1999/xlink"
           targetNamespace=""
           elementFormDefault="qualified">

  <xs:import namespace="http://www.w3.org/1999/xlink"/>
```

```
    <xs:element name="person">
      <xs:complexType>
        <xs:complexContent>
          <xs:restriction base="xs:anyType">
            <xs:attribute ref="xlink:type" fixed="locator"/>
            <xs:attribute ref="xlink:href" use="required"/>
            <xs:attribute ref="xlink:role"/>
            <xs:attribute ref="xlink:title"/>
            <xs:attribute ref="xlink:label"/>
          </xs:restriction>
        </xs:complexContent>
      </xs:complexType>
    </xs:element>

    <xs:element name="course">
      <xs:complexType>
        <xs:complexContent>
          <xs:restriction base="xs:anyType">
            <xs:attribute ref="xlink:type" fixed="locator"/>
            <xs:attribute ref="xlink:href" use="required"/>
            <xs:attribute ref="xlink:role"
                          fixed="http://www.example.com/linkprops/course"/>
            <xs:attribute ref="xlink:title"/>
            <xs:attribute ref="xlink:label"/>
          </xs:restriction>
        </xs:complexContent>
      </xs:complexType>
    </xs:element>

  </xs:schema>
```

Following is how XML elements using these declarations might look.

```
  <person
    xmlns:xlink="http://www.w3.org/1999/xlink"
    xlink:href="students/patjones62.xml"
    xlink:label="student62"
    xlink:role="http://www.example.com/linkprops/student"
  xlink:title="Pat Jones" />

  <person
    xmlns:xlink="http://www.w3.org/1999/xlink"
    xlink:href="profs/jaysmith7.xml"
    xlink:label="prof7"
    xlink:role="http://www.example.com/linkprops/professor"
    xlink:title="Dr. Jay Smith" />

  <course
    xmlns:xlink="http://www.w3.org/1999/xlink"
    xlink:href="courses/cs101.xml"
    xlink:label="CS-101"
    xlink:title="Computer Science 101" />
```

5.1.3 Traversal Rules for an Extended Link (`arc`-Type Element)

An extended link may indicate rules for traversing among its participating resources by means of a series of optional arc elements.

The XLink element for arcs is any element with an attribute in the XLink namespace called `type` with a value of "arc".

The `arc`-type element may have any content. Other than `title`-type elements that are direct children (see **5.1.4 Titles for Extended Links, Locators, and Arcs (title-Type Element)**), whatever content is present has no XLink-specified relationship to the link. If an `arc`-type element has anything other than an `extended`-type element for its parent, the `arc`-type element is not conformant.

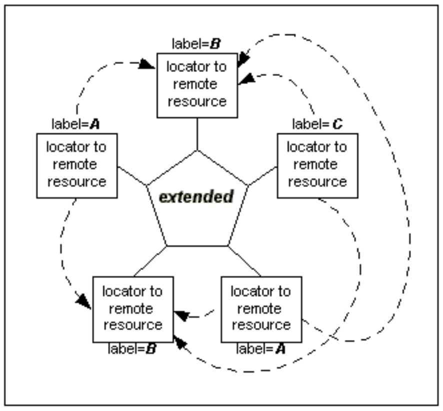
The `arc`-type element may have the traversal attributes `from` and `to` (see **5.7 Traversal Attributes (label, from, and to)**), the behavior attributes `show` and `actuate` (see **5.6 Behavior Attributes (show and actuate)**) and the semantic attributes `arcrole` and `title` (see **5.5 Semantic Attributes (role, arcrole, and title)**).

The traversal attributes define the desired traversal between pairs of resources that participate in the same link, where the resources are identified by their `label` attribute values. The `from` attribute defines resources from which traversal may be initiated, that is, starting resources, while the `to` attribute defines resources that may be traversed to, that is, ending resources. The behavior attributes specify the desired behavior for XLink applications to use when traversing to the ending resource.

The semantic attributes describe the meaning of the arc's ending resource relative to its starting resource. The `arcrole` attribute corresponds to the [RDF] notion of a property, where the role can be interpreted as stating that "*starting-resource* HAS *arc-role ending-resource*." This contextual role can differ from the meaning of an ending resource when taken outside the context of this particular arc. For example, a resource might generically represent a "person," but in the context of a particular arc it might have the role of "mother" and in the context of a different arc it might have the role of "daughter."

When the same resource serves as a starting resource in several arcs (whether in a single link or across many links), traversal-request behavior is unconstrained by this specification, but one possibility for interactive applications is a pop-up menu that lists the relevant arc or link titles.

The following diagram shows an extended link that associates five remote resources and provides rules for traversal among them. All of the arcs specified are third-party arcs; that is, the arcs go exclusively between remote resources. The nondirectional solid lines indicate, as before, that the link is associating the five resources; the new dotted arrows indicate the traversal rules that the link provides. Notice that some resources share the same `label` value.


label=B
locator to remote resource
label=A
locator to remote resource
label=C
locator to remote resource
extended
locator to remote resource
label=B
locator to remote resource
label=A

This diagram reflects directional traversal arcs created by the following settings, where both As and Cs are allowed to initiate traversal to all Bs. Because some labels appear on several resources, each arc specification potentially creates several traversal arcs at once:

```
<go xlink:type="arc" xlink:from="A" xlink:to="B" />
<go xlink:type="arc" xlink:from="C" xlink:to="B" />
```

As another example, assume an extended link that contains five locators, two with `label` values of "parent" and three with `label` values of "child":

```
<extendedlink xlink:type="extended">
  <loc xlink:type="locator" xlink:href="..." xlink:label="parent" xlink:title="p1" />
  <loc xlink:type="locator" xlink:href="..." xlink:label="parent" xlink:title="p2" />
  <loc xlink:type="locator" xlink:href="..." xlink:label="child"  xlink:title="c1" />
  <loc xlink:type="locator" xlink:href="..." xlink:label="child"  xlink:title="c2" />
  <loc xlink:type="locator" xlink:href="..." xlink:label="child"  xlink:title="c3" />
... <!-- arc-type elements would go here -->
</extendedlink>
```

The following specifies traversal from parent resources to child resources, which includes all of p1-c1, p1-c2, p1-c3, p2-c1, p2-c2, and p2-c3:

```
<go xlink:type="arc" xlink:from="parent" xlink:to="child" />
```

If no value is supplied for a `from` or `to` attribute, the missing value is interpreted as standing for *all* the labels supplied on `locator`-type elements in that `extended`-type element. For example, the following specifies traversal from parents to children and also from children to children, which includes all of p1-c1, p1-c2, p1-c3, p2-c1, p2-c2, p2-c3, c1-c1, c1-c2, c1-c3, c2-c1, c2-c2, c2-c3, c3-c1, c3-c2, and c3-c3:

```
<go xlink:type="arc" xlink:to="child" />
```

In this case, note that the traversal rules include arcs from some resources to other resources with the same label (from children to other children), as well as from some resources to themselves (from a child to itself); this is not an error.

If no `arc`-type elements are provided in an extended link, then by extension the missing `from` and `to` values are interpreted as standing for all the labels in that link. This would be equivalent to the following traversal specification:

```
<go xlink:type="arc" />
```

When more than one locator has the same label, the set of locators with the same label are to be understood as individual locators, rather than as referring to an aggregate resource; the traversal behavior of such a link might be the same as for a link where all the locators have different roles and the appropriate arcs are specified to produce the identical traversal pairs.

If the arc traversal rules for an extended link leave out any possible traversal pairs, XLink defines no traversal for these pairs. A higher-level application may perform non-XLink-directed traversals; for example, a link-checking process might traverse all available pairs of resources.

Constraint: No Arc Duplication

Each `arc`-type element must have a pair of `from` and `to` values that does not repeat the `from` and `to` values (respectively) for any other `arc`-type element in the same extended link; that is, each pair in a link must be unique.

Example: Sample `arc`-Type Element Declarations and Instance

Following is a non-normative set of RELAX NG declarations for an `arc`-type element.

```
<grammar xmlns:a="http://relaxng.org/ns/compatibility/annotations/1.0"
         xmlns:xlink="http://www.w3.org/1999/xlink"
         xmlns="http://relaxng.org/ns/structure/1.0"
         datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes">
  <define name="go">
    <element name="go">
      <ref name="attlist.go"/>
      <empty/>
    </element>
  </define>
  <define name="attlist.go" combine="interleave">
    <optional>
```

```
      <attribute name="xlink:type" a:defaultValue="arc">
        <value>arc</value>
      </attribute>
    </optional>
    <optional>
      <attribute name="xlink:arcrole"/>
    </optional>
    <optional>
      <attribute name="xlink:title"/>
    </optional>
    <optional>
      <attribute name="xlink:show">
        <choice>
          <value>new</value>
          <value>replace</value>
          <value>embed</value>
          <value>other</value>
          <value>none</value>
        </choice>
      </attribute>
    </optional>
    <optional>
      <attribute name="xlink:actuate">
        <choice>
          <value>onLoad</value>
          <value>onRequest</value>
          <value>other</value>
          <value>none</value>
        </choice>
      </attribute>
    </optional>
    <optional>
      <attribute name="xlink:from">
        <data type="NCNAME"/>
      </attribute>
    </optional>
    <optional>
      <attribute name="xlink:to">
        <data type="NCNAME"/>
      </attribute>
    </optional>
  </define>
</grammar>
```

Following is how XML elements using these declarations might look.

```
<go
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xlink:type="arc"
  xlink:from="student62"
  xlink:to="PatJonesGPA"
  xlink:show="new"
  xlink:actuate="onRequest"
  xlink:title="Pat Jones's GPA" />
<go
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xlink:type="arc"
  xlink:from="CS-101"
  xlink:arcrole="http://www.example.com/linkprops/auditor"
  xlink:to="student62"
  xlink:show="replace"
  xlink:actuate="onRequest"
  xlink:title="Pat Jones, auditing the course" />
<go
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xlink:type="arc"
  xlink:from="student62"
  xlink:arcrole="http://www.example.com/linkprops/advisor"
  xlink:to="prof7"
  xlink:show="replace"
  xlink:actuate="onRequest"
  xlink:title="Dr. Jay Smith, advisor" />
```

5.1.4 Titles for Extended Links, Locators, and Arcs (`title`-Type Element)

The `extended`-, `locator`-, and `arc`-type elements may have the `title` attribute (more about which see **5.5 Semantic Attributes (role, arcrole, and title)**). However, they may also have a series of one or more `title`-type elements. Such elements are useful, for example, for cases where human-readable label information needs further element markup, or where multiple titles are necessary. One common motivation for using the `title`-type element is to account for internationalization and localization. For example, title markup might be necessary for bidirectional contexts or in East Asian languages, and multiple titles might be necessary for different natural-language versions of a title. See [Ruby Annotation] for examples where markup might be necessary inside a title.

The XLink element for titles is any element with an attribute in the XLink namespace called `type` with a value of "title".

The `title`-type element may have any content. If a `title`-type element contains nested XLink elements, such contained elements have no XLink-specified relationship to the parent link containing the title. If a `title`-type element has anything other than an `extended`-, `locator`-, or `arc`-type element for a parent, the `title`-type element is not conformant.

Example: Sample `title`-Type Element Declarations and Instance

Following is a non-normative set of XML Schema declarations for a `title`-type element. The element has been given the `xml:lang` attribute, which may be used in conjunction with server settings or other contextual information in determining which title to present.

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:xlink="http://www.w3.org/1999/xlink"
           targetNamespace=""
           elementFormDefault="qualified">

  <xs:import namespace="http://www.w3.org/1999/xlink"/>
  <xs:import namespace="http://www.w3.org/XML/1998/namespace"/>

  <xs:element name="advisorname">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="name"/>
      </xs:sequence>
      <xs:attribute ref="xlink:type" fixed="title"/>
      <xs:attribute ref="xml:lang"/>
    </xs:complexType>
  </xs:element>

  <xs:element name="name">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="honorific" minOccurs="0" maxOccurs="1"/>
        <xs:element ref="given"/>
        <xs:element ref="family"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <!-- advisor is a locator type element -->
  <xs:element name="advisor">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="advisorname"/>
      </xs:sequence>
      <xs:attribute ref="xlink:type" fixed="locator"/>
      <xs:attribute ref="xlink:href" use="required"/>
      <xs:attribute ref="xlink:role"/>
      <xs:attribute ref="xlink:title"/>
      <xs:attribute ref="xlink:label"/>
    </xs:complexType>
  </xs:element>

  <!-- Arbitrary declarations for the components of the name -->
  <xs:element name="honorific" type="xs:string"/>
  <xs:element name="given" type="xs:string"/>
  <xs:element name="family" type="xs:string"/>
</xs:schema>
```

Following is how XML elements using these declarations might look.

```
<advisor xlink:href="profs/jaysmith7.xml" ...>
  <advisorname xml:lang="en">
    <name>
      <honorific>Dr.</honorific>
      <given>Jay</given>
      <family>Smith</family>
    </name>
  </advisorname>
</advisor>
```

5.1.5 Locating Linkbases (Special Arc Role)

For an XLink application to traverse from a starting resource to an ending resource, it needs to locate both the starting resource and the link. Locating the two pieces is not a problem in the case of outbound arcs because the starting resource is either the linking element itself or a child of the linking element. However, in the case of inbound and third-party arcs, the XLink application needs to be able to find both pieces somehow.

In the course load example, extended links can associate pairs of remote resources representing students and courses. In order for the system to load and present a "student resource" (such as a description and picture of the person) in a way that offers traversal to related information (for example, by allowing users to click on the student's name to traverse to information about the courses in which she is enrolled), it needs to locate and use the extended links that contain the association.

Linkbases are often used to make link management easier by gathering together a number of related linking elements. XLink provides a way to instruct XLink applications to access potentially relevant linkbases. The instruction takes the form of an arc specification (whether an explicit one in an extended link, or an implicit one in a simple link) that has the following value for its arcrole attribute:

```
http://www.w3.org/1999/xlink/properties/linkbase
```

Constraint: Linkbases Must Be XML

Any linkbase specified as the ending resource of an arc with this special value must be an XML document.

(XLink applications may also use any other means to locate and process additional linkbases.)

The handling of a linkbase arc is much like the handling of a normal arc, except that traversal entails loading the ending resource (the linkbase) to extract its links for later use, rather than to present it to a user or to perform some other processing. Its handling is also special in that XLink applications must suspend traversal of linkbase arcs at user option.

Specifically, on loading a linkbase arc, an XLink application should keep track of what the starting resource is. Whenever a document containing that starting resource is loaded and traversal of the linkbase arc is actuated, the application should access the linkbase and extract any extended links found inside it. In the case that the extracted resource is a portion of a complete XML document, such as a range or a string range, only those extended links completely contained in the extracted portion should be made available.

The timing of linkbase arc traversal depends on the value of the `actuate` attribute on the arc. For example, if the value is "onLoad", the linkbase is loaded and its links extracted as soon as the starting resource is loaded. Any `show` attribute value on a linkbase arc must be ignored, because traversal does not entail presentation in this case.

Linkbases may be chained by virtue of serving as the starting resource of yet another linkbase arc. The application interpreting an initial linkbase arc may choose to limit the number of steps processed in the chain.

An application should maintain a list of extended links retrieved as a result of processing a linkbase, and should not retrieve duplicate resources or links in the case where a cyclic dependency exists. To ease XLink processing, document creators may wish to define linkbase arcs near the beginning of a document.

Example: Annotating a Specification

Following is a non-normative set of declarations for an extended link that specializes in providing linkbase arcs:

```
<!ELEMENT basesloaded ((startrsrc|linkbase|load)*)>
<!ATTLIST basesloaded
  xlink:type      (extended)      #FIXED "extended">

<!ELEMENT startrsrc EMPTY>
<!ATTLIST startrsrc
  xlink:type      (locator)       #FIXED "locator"
  xlink:href      CDATA           #REQUIRED
  xlink:label     NMTOKEN         #IMPLIED>

<!ELEMENT linkbase EMPTY>
<!ATTLIST linkbase
  xlink:type      (locator)       #FIXED "locator"
  xlink:href      CDATA           #REQUIRED
  xlink:label     NMTOKEN         #IMPLIED>

<!ELEMENT load EMPTY>
<!ATTLIST load
  xlink:type      (arc)           #FIXED "arc"
  xlink:arcrole   CDATA           #FIXED "http://www.w3.org/1999/xlink/properties/linkbase"
  xlink:actuate   (onLoad
                  |onRequest
                  |other
                  |none)          #IMPLIED
  xlink:from      NMTOKEN         #IMPLIED
  xlink:to        NMTOKEN         #IMPLIED>
```

Following is how an XML element using these declarations might look. This would indicate that when a specification document is loaded, a linkbase full of annotations to it should automatically be loaded as well, possibly necessitating re-rendering of the entire specification document to reveal any regions within it that serve as starting resources in the links found in the linkbase.

```
<basesloaded>
  <startrsrc xlink:label="spec" xlink:href="spec.xml" />
  <linkbase xlink:label="linkbase" xlink:href="linkbase.xml" />
  <load xlink:from="spec" xlink:to="linkbase" actuate="onLoad" />
</basesloaded>
```

Following is how an XML element using these declarations might look if the linkbase loading were on request. This time, the starting resource consists of the words "Click here to reveal annotations." If the starting resource were the entire document as in the example above, a reasonable behavior for allowing a user to actuate traversal would be a confirmation dialog box.

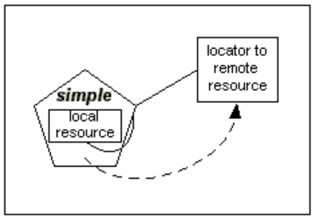
```
<basesloaded>
  <startrsrc
    xlink:label="spec"
    xlink:href="spec.xml#string-range(//*,'Click here to reveal annotations.')" />
  <linkbase xlink:label="linkbase" xlink:href="linkbase.xml" />
  <load xlink:from="spec" xlink:to="linkbase" actuate="onRequest" />
</basesloaded>
```

5.2 Simple Links (`simple`-Type Element)

[Definition: A **simple link** is a link that associates exactly two resources, one local and one remote, with an arc going from the former to the latter. Thus, a simple link is always an outbound link.]

The purpose of a simple link is to be a convenient shorthand for the equivalent extended link. A single simple linking element combines the basic functions of an `extended`-type element, a `locator`-type element, an `arc`-type element, and a `resource`-type element.

The following diagram shows the characteristics of a simple link; it associates one local and one remote resource, and implicitly provides a single traversal arc from the local resource to the remote one. This could represent, for example, the name of a student appearing in text which, when clicked, leads to information about the student.



Example: Simple Link Functionality Done with an Extended Link

A simple link could be represented by an extended link in approximately the following way:

```
<studentlink xlink:type="extended">
  <resource
    xlink:type="resource"
    xlink:label="local">Pat Jones</resource>
  <locator
    xlink:type="locator"
    xlink:href="..."
    xlink:label="remote"
    xlink:role="..."
    xlink:title="..." />

  <go
    xlink:type="arc"
    xlink:from="local"
    xlink:to="remote"
    xlink:arcrole="..."
    xlink:show="..."
    xlink:actuate="..." />
</studentlink>
```

A simple link combines all the features above (except for the types and labels) into a single element. In cases where only this subset of features is required, the XLink simple linking element is available as an alternative to the extended linking element. The features missing from simple links are as follows:

- Supplying arbitrary numbers of local and remote resources
- Specifying an arc from its remote resource to its local resource
- Associating a title with the single hardwired arc
- Associating a role or title with the local resource
- Associating a role or title with the link as a whole

The XLink element for simple links is any element:

- with an attribute in the XLink namespace called `type` with a value of "simple" or
- with an attribute in the XLink namespace called `href` and no attribute in the XLink namespace called `type`. In this case, the value "simple" is implied for the `type` attribute.

In other words, the XLink `type` attribute is optional on XLink simple links.

The simple equivalent of the above extended link would be as follows:

```
<studentlink xlink:href="...">Pat Jones</studentlink>
```

The `simple`-type element may have any content. The `simple`-type element itself, together with all of its content, is the local resource of the link, as if the element were a `resource`-type element. If a `simple`-type element contains nested XLink elements, such contained elements have no XLink-specified relationship to the parent link. It is possible for a `simple`-type element to have no content; in cases where the link is expected to be traversed on request, interactive XLink applications will typically generate some content in order to give the user a way to initiate the traversal.

The `simple`-type element effectively takes the locator attribute `href` and the semantic attributes `role` and `title` from the `locator`-type element, and the behavior attributes `show` and `actuate` and the single semantic attribute `arcrole` from the `arc`-type element.

It is not an error for a `simple`-type element to have no locator (`href`) attribute value. If a value is not provided, the link is simply untraversable. Such a link may still be useful, for example, to associate properties with the resource by means of XLink attributes.

Example: Sample `simple`-Type Element Declarations and Instance

Following is a non-normative set of RELAX NG declarations (in the compact syntax) for a `simple`-type element.

```
namespace a = "http://relaxng.org/ns/compatibility/annotations/1.0"
namespace xlink = "http://www.w3.org/1999/xlink"

studentlink = element studentlink { attlist.studentlink, any }

attlist.studentlink &=
  attribute xlink:type { "simple" }?,
  attribute xlink:href { text }?,
  [ a:defaultValue = "http://www.example.com/linkprops/student" ]
  attribute xlink:role {
    string "http://www.example.com/linkprops/student"
  }?,

  attribute xlink:arcrole { text }?,
  attribute xlink:title { text }?,
  attribute xlink:show {
    "new" | "replace" | "embed" | "other" | "none"
  }?,
  attribute xlink:actuate { "onLoad" | "onRequest" | "other" | "none" }?

any =
  (element * {
     attribute * { text }*,
     any
   }
   | text)*
```

Following is how an XML document might use these declarations.

```
  ..., and <studentlink xlink:href="students/patjones62.xml">Pat
  Jones</studentlink> is popular around the student union.
```

5.3 XLink Element Type Attribute (`type`)

The attribute that identifies XLink element types is `type`.

Constraint: type Value

The value of the `type` attribute must be supplied unless the element is a simple link and an `href` attribute in the XLink namespace is supplied. In the latter case, the value "simple" is implied for the `type` attribute. If a value is supplied for the `type` attribute, its value must be one of "simple", "extended", "locator", "arc", "resource", "title", or "none".

When the value of the `type` attribute is "none", the element has no XLink-specified meaning, and any XLink-related content or attributes have no XLink-specified relationship to the element.

Example: Sample `type` Attribute Declarations

Following is a non-normative attribute-list declaration for `type` on an element intended to be `simple`-type.

```
  <!ATTLIST simple-link-element
    xlink:type      (simple)        #FIXED "simple"
    ...>
```

An analogous declaration in XML Schema is:

```
  <xs:element name="simple-link-element">
    ...
    <xs:attribute ref="xlink:type" fixed="simple"/>
    ...
  </xs:element>
```

In RELAX NG:

```
  element simple-link-element {
    ...
    [ a:defaultValue = "simple" ] attribute xlink:type { "simple" }?,
    ...
  }
```

For an element that serves as an XLink element only on some occasions, one declaration might be as follows, where the document creator sets the value to "simple" in some circumstances and "none" in others. The use of "none" might be useful in helping XLink applications to avoid checking for the presence of an `href` value.

```
  <!ATTLIST commandname
    xlink:type      (simple|none)   #IMPLIED
    xlink:href      CDATA           #IMPLIED>
```

Analogously RELAX NG:

```
  element commandname {
    ...
    attribute xlink:type { "simple" | "none" }?,
    ...
  }
```

The global nature of `xlink:type` makes redefinition on a per-element basis impractical in XML Schema.

5.4 Locator Attribute (`href`)

The attribute that supplies the data that allows an XLink application to find a remote resource (or resource fragment) is `href`. It may be used on `simple`-type elements, and must be used on `locator`-type elements.

The value of the href attribute is a [Legacy extended IRIs] (LEIRI). Processing a relative identifier against a base is handled straightforwardly; the algorithms of [RFC 3986] can be applied directly, treating the characters additionally allowed in LEIRIs in the same way that unreserved characters are in URI references.

> **Note:**
>
> XLink 1.0 explicitly did not require applications to check that the value of the `xlink:href` attribute conformed to the syntactic rules of a URI. While [RFC 3986] has clarified the syntactic rules, this specification follows the lead of XLink 1.0 (and many other specifications) and does not impose any new conformance testing requirements on XLink applications in this area.
>
> Although XLink applications need not enforce URI syntactic constraints, XLink applications which use libraries which *do* detect violations of the syntactic rules of [RFC 3986] should not recover silently.

If the value of the `href` attribute is a relative reference (as defined in [RFC 3986], also known as "relative URI" in earlier RFCs), or results in a relative reference after escaping, its absolute version must be computed by the method of [XML Base] before use.

If a locator includes a fragment identifer, the syntax of the fragment identifier is defined by the media type of the representation returned when the locator is dereferenced. For locators into XML resources (that is, resources with the media type "application/xml" or media types that defer to the fragment identifier syntax of "application/xml" media), the syntax of the fragment identifier is expected to be defined by the successor to [RFC 3023]. Technically, there is no fragment identifier syntax for XML resources at the time of this writing, though the [XPointer Framework] and [XPointer element() Scheme] are explicitly supported by several XML vocabularies.

Example: Sample `href` Attribute Declarations

Following is a non-normative attribute-list declaration for `href` on an element intended to be `simple`-type.

```
<!ATTLIST simplelink
  xlink:href      CDATA           #REQUIRED
  ...>
```

5.5 Semantic Attributes (`role`, `arcrole`, and `title`)

The attributes that describe the meaning of resources within the context of a link are `role`, `arcrole`, and `title`. The `role` attribute may be used on `extended`-, `simple`-, `locator`-, and `resource`-type elements. The `arcrole` attribute may be used on `arc`- and `simple`-type elements. The `title` attribute may be used on all of these types of elements.

The value of the `role` or `arcrole` attribute is a [Legacy extended IRIs]. The identifier must not be relative.

The `title` attribute is used to describe the meaning of a link or resource in a human-readable fashion, along the same lines as the `role` or `arcrole` attribute. (However, see also **5.1.4 Titles for Extended Links, Locators, and Arcs (title-Type Element)**.) A value is optional; if a value is supplied, it should contain a string that describes the resource. The use of this information is highly dependent on the type of processing being done. It may be used, for example, to make titles available to applications used by visually impaired users, or to create a table of links, or to present help text that appears when a user lets a mouse pointer hover over a starting resource.

Example: Sample `role` and `title` Attribute Declarations

Following is a non-normative attribute-list declaration for `role` and `title` on an element intended to be `simple`-type.

```
<!ATTLIST simplelink
  ...
  xlink:role      CDATA           #IMPLIED
  xlink:title     CDATA           #IMPLIED
  ...>
```

Following is a non-normative attribute-list declaration for `arcrole` and `title` on an element intended to be `arc`-type.

```
<!ATTLIST go
  ...
  xlink:arcrole   CDATA           #IMPLIED
  xlink:title     CDATA           #IMPLIED
  ...>
```

5.6 Behavior Attributes (`show` and `actuate`)

The behavior attributes are `show` and `actuate`. They may be used on the `simple`- and `arc`-type elements. When used on a `simple`-type element, they signal behavior intentions for traversal to that link's single remote ending resource. When they are used on an `arc`-type element, they signal behavior intentions for traversal to whatever ending resources (local or remote) are specified by that arc.

The `show` and `actuate` attributes are not required.

Example: Sample `show` and `actuate` Attribute Declarations

Following is a non-normative attribute-list declaration for `show` and `actuate` on an element intended to be `simple`-type.

```
<!ATTLIST simplelink
  xlink:type      (simple)        #FIXED "simple"
  ...
  xlink:show      (new
                  |replace
                  |embed
                  |other
                  |none)          #IMPLIED
  xlink:actuate   (onLoad
                  |onRequest
                  |other
                  |none)          #IMPLIED
  ...>
```

Applications encountering `arc`-type elements in linkbase lists must treat the behavior attributes as if they were specified as `show="none"` and `actuate="onLoad"`, even if other values were specified.

5.6.1 `show` Attribute

The `show` attribute is used to communicate the desired presentation of the ending resource on traversal from the starting resource.

Constraint: show Value

If a value is supplied for a `show` attribute, it must be one of the values "new", "replace", "embed", "other", and "none".

Conforming XLink applications should apply the following treatment for `show` values:

"new"

An application traversing to the ending resource should load it in a new window, frame, pane, or other relevant presentation context. This is similar to the effect achieved by the following HTML fragment:

```
<A HREF="http://www.example.org" target="_blank">...</A>
```

"replace"

An application traversing to the ending resource should load the resource in the same window, frame, pane, or other relevant presentation context in which the starting resource was loaded. This is similar to the effect achieved by the following HTML fragment:

```
<A HREF="http://www.example.org" target="_self">...</A>
```

"embed"

An application traversing to the ending resource should load its presentation in place of the presentation of the starting resource. This is similar to the effect achieved by the following HTML fragment:

```
<IMG SRC="http://www.example.org/smiley.gif" ALT=":-)">
```

The presentation of the starting resource typically does not consist of an entire document; it would be the entire document only when the root element of the document is a simple link. Thus, embedding typically has an effect distinct from replacing.

Just as for the HTML `IMG` element, embedding affects only the presentation of the relevant resources; it does not dictate permanent transformation of the starting resource. Put another way, when an embedded XLink is processed, the result of styling the ending resource of the link is merged into the result of styling the resource into which it is embedded. By contrast, when a construct such as an [XInclude] element is resolved, the transformation takes place in the original source document.

The behavior of conforming XLink applications when embedding XML-based ([RFC 3023]) ending resources is not defined in this version of this specification.

The presentation of embedded resources is application dependent.

"other"

The behavior of an application traversing to the ending resource is unconstrained by this specification. The application should look for other markup present in the link to determine the appropriate behavior.

"none"

The behavior of an application traversing to the ending resource is unconstrained by this specification. No other markup is present to help the application determine the appropriate behavior.

If the starting or ending resource consists of multiple non-contiguous locations, such as a series of string ranges in various locations in the resource, then application behavior is unconstrained. (See [XPTR] for more information about selecting portions of XML documents.)

Note:

Some possibilities for application behavior with non-contiguous ending resources might include highlighting of each location, producing a dialog box that allows the reader to choose among the locations as if there were separate arcs leading to each one, concatenating the content of all the locations for presentation, and so on. Application behavior with non-contiguous starting resources might include concatenation and rendering as a single unit, or creating one arc emanating from each contiguous portion.

5.6.2 `actuate` Attribute

The `actuate` attribute is used to communicate the desired timing of traversal from the starting resource to the ending resource..

Constraint: actuate Value

If a value is supplied for an `actuate` attribute, it must be be one of the values "onLoad", "onRequest", "other", and "none".

Conforming XLink applications should apply the following treatment for `actuate` values:

"onLoad"

An application should traverse to the ending resource immediately on loading the starting resource. This is similar to the effect typically achieved by the following HTML fragment, when the user agent is configured to display images:

```
<IMG SRC="http://www.example.org/smiley.gif" ALT=":-)">
```

If a single resource contains multiple arcs whose behavior is set to `show="replace" actuate="onLoad"`, application behavior is unconstrained by XLink.

"onRequest"

An application should traverse from the starting resource to the ending resource only on a post-loading event triggered for the purpose of traversal. An example of such an event might be when a user clicks on the presentation of the starting resource, or a software module finishes a countdown that precedes a redirect.

"other"

The behavior of an application traversing to the ending resource is unconstrained by this specification. The application should look for other markup present in the link to determine the appropriate behavior.

"none"

The behavior of an application traversing to the ending resource is unconstrained by this specification. No other markup is present to help the application determine the appropriate behavior.

5.7 Traversal Attributes (`label`, `from`, and `to`)

The traversal attributes are `label`, `from`, and `to`. The `label` attribute may be used on the `resource`- and `locator`-type elements. The `from` and `to` attributes may be used on the `arc`-type element.

Constraint: `label`, `from`, and `to` Values

The value of a `label`, `from`, or `to` attribute must be an NCName. If a value is supplied for a `from` or `to` attribute, it must correspond to the same value for some `label` attribute on a `locator`- or `resource`-type element that appears as a direct child inside the same `extended`-type element as does the `arc`-type element.

A References

A.1 Normative References

RFC 3023
RFC 3023: XML Media Types. Makoto, M., St. Laurent, S. and D. Kohn, editors. Internet Engineering Task Force, 2001. (See http://www.ietf.org/rfc/rfc3023.txt.)

RFC 3986
RFC 3986: Uniform Resource Identifier (URI): Generic Syntax. Berners-Lee, T., Fielding, R., and Masinter, L, editors. Internet Engineering Task Force, 2005. (See http://www.ietf.org/rfc/rfc3986.txt.)

RFC 3987
RFC 3987: Internationalized Resource Identifiers (IRIs). Internet Engineering Task Force, 2005. (See http://www.ietf.org/rfc/rfc3987.txt.)

Legacy extended IRIs
Legacy extended IRIs for XML resource identification. Henry S. Thompson, Richard Tobin, and Norman Walsh, editors. World Wide Web Consortium, 2008. (See http://www.w3.org/TR/leiri.)

XML
Extensible Markup Language (XML) 1.0 (Fifth Edition). Tim Bray, Jean Paoli, C.M. Sperberg-McQueen, et. al., editors. World Wide Web Consortium, 2008. (See http://www.w3.org/TR/REC-xml/.)

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

XML Base
XML Base (Second Edition). Jonathan Marsh and Richard Tobin, editors. World Wide Web Consortium, 2009. (See http://www.w3.org/TR/xmlbase/.)

XML Names
Namespaces in XML 1.0 (Third Edition). Tim Bray, Dave Hollander, Andrew Layman, et. al. editors. World Wide Web Consortium, 2009. (See http://www.w3.org/TR/REC-xml-names/.)

XPointer Framework
XPointer Framework Grosso, Paul, Eve Maler, Jonathan Marsh, and Norman Walsh, editors. World Wide Web Consortium, 2003. (See http://www.w3.org/TR/xptr-framework/.)

XPointer element() Scheme
XPointer element() Scheme Grosso, Paul, Eve Maler, Jonathan Marsh, and Norman Walsh, editors. World Wide Web Consortium, 2003. (See http://www.w3.org/TR/xptr-element/.)

CharMod Fundamentals
Character Model for the World Wide Web 1.0: Fundamentals Dürst, Martin J., François Yergeau, Richard Ishida, *et. al.*, editors. World Wide Web Consortium, 2005. (See http://www.w3.org/TR/charmod/.)

A.2 Non-Normative References

Extending XLink 1.0
: *Extending XLink 1.0*, Norman Walsh, Editor. World Wide Web Consortium, 27 Jan 2005. (See http://www.w3.org/TR/xlink10-ext.)

XML Linking Language (XLink) Version 1.0
: *XML Linking Language (XLink) Version 1.0*, Steven DeRose, David Orchard, and Eve Maler, Editors. World Wide Web Consortium, 27 Jun 2001. (See http://www.w3.org/TR/xlink/.)

Dexter
: "The Dexter Hypertext Reference Model." Halasz, Frank. 1994. In Communications of the Association for Computing Machinery 37 (2), February 1994: 30-39.

FRESS
: Steven J. DeRose and Andries van Dam. 1999. "Document structure in the FRESS Hypertext System." Markup Languages 1 (1) Winter. Cambridge: MIT Press: 7-32.

HTML
: *HTML 4.01 Specification*. World Wide Web Consortium, 1999. (See http://www.w3.org/TR/1999/REC-html401-19991224/.)

Intermedia
: "Intermedia: The Concept and the Construction of a Seamless Information Environment." Yankelovich, Nicole, Bernard J. Haan, Norman K. Meyrowitz, and Steven M. Drucker. 1988. IEEE Computer 21 (January, 1988): 81-96.

ISO/IEC 10744
: *ISO/IEC 10744-1992 (E). Information technology-Hypermedia/Time-based Structuring Language (HyTime).* [Geneva]: International Organization for Standardization, 1992. *Extended Facilities Annex.* ISO (International Organization for Standardization). [Geneva]: International Organization for Standardization, 1996. (See http://www.y12.doe.gov/sgml/wg8/document/1920.htm.)

MicroCosm
: *Rethinking Hypermedia: The Microcosm Approach.* Hall, Wendy, Hugh Davis, and Gerard Hutchings. 1996. Boston: Kluwer Academic Publishers. ISBN 0-7923-9679-0.

OHS
: "The Role of XML in Open Hypermedia Systems." van Ossenbruggen, Jacco, Anton Eliëns and Lloyd Rutledge. Position paper for the 4th Workshop on Open Hypermedia Systems, ACM Hypertext '98.

RDF
: *RDF Primer*. Manola, Frank and Eric Miller, editors. World Wide Web Consortium, 2004. (See http://www.w3.org/TR/rdf-primer/.)

TEI
: *Guidelines for Electronic Text Encoding and Interchange*. C. M. Sperberg-McQueen and Lou Burnard, editors. Association for Computers and the Humanities (ACH), Association for Computational Linguistics (ACL), and Association for Literary and Linguistic Computing (ALLC). Chicago, Oxford: Text Encoding Initiative, 1994.

XIS
: *XML Information Set*. John Cowan and Richard Tobin, editors. World Wide Web Consortium, 2004. (See http://www.w3.org/TR/xml-infoset/.)

XInclude
: *XML Inclusions (XInclude) Version 1.0 (Second Edition)*. Jonathan Marsh, David Orchard, and Daniel Veillard, editors. World Wide Web Consortium, 2006. (See http://www.w3.org/TR/xinclude/.)

XLREQ
: *XML XLink Requirements Version 1.0*. Steven DeRose, editor. World Wide Web Consortium, 1999. (See http://www.w3.org/TR/1999/NOTE-xlink-req-19990224/.)

XPTR
: *XML Pointer Language (XPointer)*. Ron Daniel, Steve DeRose, Eve Maler, *et. al.* editors. World Wide Web Consortium, 2002. (See http://www.w3.org/TR/xptr/.)

XML Schema Part 1: Structures
: *XML Schema Part 1: Structures*. David Beech, Noah Mendelsohn, Murray Maloney, and Henry S. Thompson, Editors. World Wide Web Consortium, 2004.

RELAX NG
: *ISO/IEC 19757-2:2008 Information technology -- Document Schema Definition Language (DSDL) -- Part 2: Regular-grammar-based validation -- RELAX NG*. International Organization for Standardization, 2008.

CSS
: *Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification*. Bert Bos, Tantek Çelik, Ian Hickson, et. al., editors. World Wide Web Consortium, 2009. (See http://www.w3.org/TR/CSS2.)

Ruby Annotation
: *Ruby Annotation*. Michel Suignard, Tex Texin, Marcin Sawicki, *et. al.*, Editors. World Wide Web Consortium, 2001. (See http://www.w3.org/TR/ruby/.)

B Sample DTD (Non-Normative)

The following DTD makes invalid (for purposes of argument) all XLink constructs for which this specification does not specify behavior. It is provided only as a convenience for application developers; it has no normative status.

The following assumptions hold for this DTD:

- Only constructs that have XLink-defined meaning are allowed.
- No "foreign" vocabularies are mixed in, since DTDs do not work well with namespaces.
- The use of **ANY** means there is typically content provided in the element that is used by XLink in some way.
- The use of the `(title*)` construct means that any non-title content provided has no XLink-defined use.
- Elements are named after the XLink element types they represent.

Other assumptions and conditions appear as comments in the DTD.

```
<!ENTITY % showActuate
  "xlink:show      (new
                   |replace
                   |embed
                   |other
                   |none)          #IMPLIED
```

```
     xlink:actuate   (onLoad
                     |onRequest
                     |other
                     |none)          #IMPLIED">

  <!ENTITY % simpleAttrs
    'xlink:type      (simple)        "simple"
     xlink:href      CDATA           #IMPLIED
     xlink:role      CDATA           #IMPLIED
     xlink:arcrole   CDATA           #IMPLIED
     xlink:title     CDATA           #IMPLIED
     %showActuate;'>

  <!ELEMENT simple ANY>
  <!ATTLIST simple
     xmlns:xlink     CDATA           #FIXED "http://www.w3.org/1999/xlink"
     %simpleAttrs;>

  <!ENTITY % extendedContent.extras "">

  <!ENTITY % extendedModel
    "(title|resource|locator|arc %extendedContent.extras;)*">

  <!ENTITY % extendedAttrs
    'xlink:type      (extended)      #FIXED "extended"
     xlink:role      CDATA           #IMPLIED
     xlink:title     CDATA           #IMPLIED'>

  <!ELEMENT extended (%extendedModel;)>
  <!ATTLIST extended
     xmlns:xlink     CDATA           #FIXED "http://www.w3.org/1999/xlink"
     %extendedAttrs;>

  <!ENTITY % titleAttrs
    'xlink:type      (title)         #FIXED "title"
     xml:lang        CDATA           #IMPLIED'>

  <!ELEMENT title ANY>
  <!-- xml:lang is not required, but provides much of the motivation
        for title elements in addition to attributes, and so is provided
        here for convenience -->
  <!ATTLIST title
     %titleAttrs;>

  <!ENTITY % resourceAttrs
    'xlink:type      (resource)      #FIXED "resource"
     xlink:role      CDATA           #IMPLIED
     xlink:title     CDATA           #IMPLIED
     xlink:label     NMTOKEN         #IMPLIED'>

  <!ELEMENT resource ANY>
  <!ATTLIST resource
     %resourceAttrs;>

  <!ENTITY % locatorAttrs
    'xlink:type      (locator)       #FIXED "locator"
     xlink:href      CDATA           #REQUIRED
     xlink:role      CDATA           #IMPLIED
     xlink:title     CDATA           #IMPLIED
     xlink:label     NMTOKEN         #IMPLIED'>

  <!ELEMENT locator (title*)>
  <!-- label is not required, but locators have no particular XLink
       function if they are not labeled -->
  <!ATTLIST locator
     %locatorAttrs;>

  <!ENTITY % arcAttrs
    'xlink:type      (arc)           #FIXED "arc"
     xlink:arcrole   CDATA           #IMPLIED
     xlink:title     CDATA           #IMPLIED
     xlink:from      NMTOKEN         #IMPLIED
     xlink:to        NMTOKEN         #IMPLIED
     %showActuate;'>

  <!ELEMENT arc (title*)>
  <!-- from and to have default behavior when values are missing -->
  <!ATTLIST arc
     %arcAttrs;>
```

C Sample XML Schema (Non-Normative)

The following XML Schema document, per [XML Schema Part 1: Structures], provides XLink-1.1-specific declarations and definitions for use in defining linking elements which conform to this specification.

A permanent copy of this schema document is available at http://www.w3.org/XML/2008/06/xlink.xsd. Another copy is available at http://www.w3.org/1999/xlink.xsd. At the time of publication these two copies are identical, but the version at `.../1999/xlink.xsd` may change in the future to reflect subsequent editions or versions of XLink or of XML Schema.

```
  <?xml version='1.0' encoding='UTF-8'?>
  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" targetNamespace="http://www.w3.org/1999/xlink" xmlns:xlink="http://www.w3.o

   <xs:annotation>
    <xs:documentation>This schema document provides attribute declarations and
```

```
attribute group, complex type and simple type definitions which can be used in
the construction of user schemas to define the structure of particular linking
constructs, e.g.
<![CDATA[
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:xl="http://www.w3.org/1999/xlink">

 <xs:import namespace="http://www.w3.org/1999/xlink"
            location="http://www.w3.org/1999/xlink.xsd">

 <xs:element name="mySimple">
  <xs:complexType>
   ...
   <xs:attributeGroup ref="xl:simpleAttrs"/>
   ...
  </xs:complexType>
 </xs:element>
 ...
</xs:schema>]]></xs:documentation>
 </xs:annotation>

 <xs:import namespace="http://www.w3.org/XML/1998/namespace" schemaLocation="http://www.w3.org/2001/xml.xsd"/>

 <xs:attribute name="type" type="xlink:typeType"/>

 <xs:simpleType name="typeType">
  <xs:restriction base="xs:token">
   <xs:enumeration value="simple"/>
   <xs:enumeration value="extended"/>
   <xs:enumeration value="title"/>
   <xs:enumeration value="resource"/>
   <xs:enumeration value="locator"/>
   <xs:enumeration value="arc"/>
  </xs:restriction>
 </xs:simpleType>

 <xs:attribute name="href" type="xlink:hrefType"/>

 <xs:simpleType name="hrefType">
  <xs:restriction base="xs:anyURI"/>
 </xs:simpleType>

 <xs:attribute name="role" type="xlink:roleType"/>

 <xs:simpleType name="roleType">
  <xs:restriction base="xs:anyURI">
   <xs:minLength value="1"/>
  </xs:restriction>
 </xs:simpleType>

 <xs:attribute name="arcrole" type="xlink:arcroleType"/>

 <xs:simpleType name="arcroleType">
  <xs:restriction base="xs:anyURI">
   <xs:minLength value="1"/>
  </xs:restriction>
 </xs:simpleType>

 <xs:attribute name="title" type="xlink:titleAttrType"/>

 <xs:simpleType name="titleAttrType">
  <xs:restriction base="xs:string"/>
 </xs:simpleType>

 <xs:attribute name="show" type="xlink:showType"/>

 <xs:simpleType name="showType">
  <xs:restriction base="xs:token">
   <xs:enumeration value="new"/>
   <xs:enumeration value="replace"/>
   <xs:enumeration value="embed"/>
   <xs:enumeration value="other"/>
   <xs:enumeration value="none"/>
  </xs:restriction>
 </xs:simpleType>

 <xs:attribute name="actuate" type="xlink:actuateType"/>

 <xs:simpleType name="actuateType">
  <xs:restriction base="xs:token">
   <xs:enumeration value="onLoad"/>
   <xs:enumeration value="onRequest"/>
   <xs:enumeration value="other"/>
   <xs:enumeration value="none"/>
  </xs:restriction>
 </xs:simpleType>

 <xs:attribute name="label" type="xlink:labelType"/>

 <xs:simpleType name="labelType">
  <xs:restriction base="xs:NCName"/>
 </xs:simpleType>

 <xs:attribute name="from" type="xlink:fromType"/>

 <xs:simpleType name="fromType">
  <xs:restriction base="xs:NCName"/>
```

```
</xs:simpleType>

<xs:attribute name="to" type="xlink:toType"/>

<xs:simpleType name="toType">
 <xs:restriction base="xs:NCName"/>
</xs:simpleType>

<xs:attributeGroup name="simpleAttrs">
 <xs:attribute ref="xlink:type" fixed="simple"/>
 <xs:attribute ref="xlink:href"/>
 <xs:attribute ref="xlink:role"/>
 <xs:attribute ref="xlink:arcrole"/>
 <xs:attribute ref="xlink:title"/>
 <xs:attribute ref="xlink:show"/>
 <xs:attribute ref="xlink:actuate"/>
</xs:attributeGroup>

<xs:group name="simpleModel">
 <xs:sequence>
  <xs:any processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
 </xs:sequence>
</xs:group>

<xs:complexType mixed="true" name="simple">
 <xs:annotation>
  <xs:documentation>
   Intended for use as the type of user-declared elements to make them
   simple links.
  </xs:documentation>
 </xs:annotation>
 <xs:group ref="xlink:simpleModel"/>
 <xs:attributeGroup ref="xlink:simpleAttrs"/>
</xs:complexType>

<xs:attributeGroup name="extendedAttrs">
 <xs:attribute ref="xlink:type" fixed="extended" use="required"/>
 <xs:attribute ref="xlink:role"/>
 <xs:attribute ref="xlink:title"/>
</xs:attributeGroup>

<xs:group name="extendedModel">
  <xs:choice>
   <xs:element ref="xlink:title"/>
   <xs:element ref="xlink:resource"/>
   <xs:element ref="xlink:locator"/>
   <xs:element ref="xlink:arc"/>
 </xs:choice>
</xs:group>

<xs:complexType name="extended">
 <xs:annotation>
  <xs:documentation>
   Intended for use as the type of user-declared elements to make them
   extended links.
   Note that the elements referenced in the content model are all abstract.
   The intention is that by simply declaring elements with these as their
   substitutionGroup, all the right things will happen.
  </xs:documentation>
 </xs:annotation>
 <xs:group ref="xlink:extendedModel" minOccurs="0" maxOccurs="unbounded"/>
 <xs:attributeGroup ref="xlink:extendedAttrs"/>
</xs:complexType>

<xs:element name="title" type="xlink:titleEltType" abstract="true"/>

<xs:attributeGroup name="titleAttrs">
 <xs:attribute ref="xlink:type" fixed="title" use="required"/>
 <xs:attribute ref="xml:lang">
  <xs:annotation>
   <xs:documentation>
    xml:lang is not required, but provides much of the
    motivation for title elements in addition to attributes, and so
    is provided here for convenience.
   </xs:documentation>
  </xs:annotation>
 </xs:attribute>
</xs:attributeGroup>

<xs:group name="titleModel">
 <xs:sequence>
  <xs:any processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
 </xs:sequence>
</xs:group>

<xs:complexType mixed="true" name="titleEltType">
 <xs:group ref="xlink:titleModel"/>
 <xs:attributeGroup ref="xlink:titleAttrs"/>
</xs:complexType>

<xs:element name="resource" type="xlink:resourceType" abstract="true"/>

<xs:attributeGroup name="resourceAttrs">
 <xs:attribute ref="xlink:type" fixed="resource" use="required"/>
 <xs:attribute ref="xlink:role"/>
 <xs:attribute ref="xlink:title"/>
 <xs:attribute ref="xlink:label"/>
```

```
  </xs:attributeGroup>

  <xs:group name="resourceModel">
   <xs:sequence>
    <xs:any processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
   </xs:sequence>
  </xs:group>

  <xs:complexType mixed="true" name="resourceType">
   <xs:group ref="xlink:resourceModel"/>
   <xs:attributeGroup ref="xlink:resourceAttrs"/>
  </xs:complexType>

  <xs:element name="locator" type="xlink:locatorType" abstract="true"/>

  <xs:attributeGroup name="locatorAttrs">
   <xs:attribute ref="xlink:type" fixed="locator" use="required"/>
   <xs:attribute ref="xlink:href" use="required"/>
   <xs:attribute ref="xlink:role"/>
   <xs:attribute ref="xlink:title"/>
   <xs:attribute ref="xlink:label">
    <xs:annotation>
     <xs:documentation>
      label is not required, but locators have no particular
      XLink function if they are not labeled.
     </xs:documentation>
    </xs:annotation>
   </xs:attribute>
  </xs:attributeGroup>

  <xs:group name="locatorModel">
   <xs:sequence>
    <xs:element ref="xlink:title" minOccurs="0" maxOccurs="unbounded"/>
   </xs:sequence>
  </xs:group>

  <xs:complexType name="locatorType">
   <xs:group ref="xlink:locatorModel"/>
   <xs:attributeGroup ref="xlink:locatorAttrs"/>
  </xs:complexType>

  <xs:element name="arc" type="xlink:arcType" abstract="true"/>

  <xs:attributeGroup name="arcAttrs">
   <xs:attribute ref="xlink:type" fixed="arc" use="required"/>
   <xs:attribute ref="xlink:arcrole"/>
   <xs:attribute ref="xlink:title"/>
   <xs:attribute ref="xlink:show"/>
   <xs:attribute ref="xlink:actuate"/>
   <xs:attribute ref="xlink:from"/>
   <xs:attribute ref="xlink:to">
    <xs:annotation>
     <xs:documentation>
      from and to have default behavior when values are missing
     </xs:documentation>
    </xs:annotation>
   </xs:attribute>
  </xs:attributeGroup>

  <xs:group name="arcModel">
   <xs:sequence>
    <xs:element ref="xlink:title" minOccurs="0" maxOccurs="unbounded"/>
   </xs:sequence>
  </xs:group>

  <xs:complexType name="arcType">
   <xs:group ref="xlink:arcModel"/>
   <xs:attributeGroup ref="xlink:arcAttrs"/>
  </xs:complexType>

 </xs:schema>
```

Note: The Working Group acknowledges the work of the XBRL Consortium in producing a W3C XML Schema for XLink 1.0, which was useful input into the design of the schema document for XLink 1.1.

D Sample RELAX NG Grammar (Non-Normative)

The following [RELAX NG] Grammars (expressed in the compact syntax) validate XLink 1.1. They are provided only as a convenience for application developers; they have no normative status.

`xlink11-simple.rnc`:

```
 namespace xlink = "http://www.w3.org/1999/xlink"

 start = anyElement
 anyElement = simple | foreignElement
 foreignElement = element * - xlink:* { foreign.att*, (anyElement | text)* }

 simple.type = attribute xlink:type {"simple"}
 href.att = attribute xlink:href {xsd:anyURI}
 role.att = attribute xlink:role {xsd:anyURI}
 arcrole.att = attribute xlink:arcrole {xsd:anyURI}
 title.att = attribute xlink:title {text}
 show.att = attribute xlink:show {"new"|"replace"|"embed"|"other"|"none"}
```

```
  actuate.att = attribute xlink:actuate {"onLoad"|"onRequest"|"other"|"none"}
  foreign.att = attribute * - xlink:*  {text}

  simple = element * {
                  (simple.type | href.att | (simple.type, href.att)),
                  foreign.att*, role.att?, arcrole.att?, title.att?,
                  show.att?, actuate.att?,
                  (anyElement | text)*
                  }
```

`xlink11.rnc`:

```
  namespace xlink = "http://www.w3.org/1999/xlink"

  include "xlink11-simple.rnc" {
          anyElement = simple | extended | foreignElement
          }

  label.att = attribute xlink:label {xsd:NCName}
  from.att = attribute xlink:from {xsd:NCName}
  to.att = attribute xlink:to {xsd:NCName}

  extended = element * {
                  attribute xlink:type {"extended"},
                  foreign.att*, role.att?, title.att?,
                  (title | resource | locator | arc | anyElement | text)*
                  }

  title = element * {
                  attribute xlink:type {"title"}, foreign.att*,
                  (anyElement | text)*
                  }

  resource = element * {
                  attribute xlink:type {"resource"}, foreign.att*,
                  role.att?, title.att?, label.att?,
                  (anyElement | text)*
                  }

  locator = element * {
                  attribute xlink:type {"locator"}, foreign.att*,
                  href.att, role.att?, title.att?, label.att?,
                  (title | anyElement | text)*
                  }

  arc = element * {
                  attribute xlink:type {"arc"}, foreign.att*,
                  arcrole.att?, title.att?, from.att?, to.att?,
                  show.att?, actuate.att?,
                  (title | anyElement | text)*
                  }
```

E Changes from XLink 1.0 (Non-Normative)

This specification implements the changes described in [Extending XLink 1.0]. These changes make XLink more useful in the places where it is already being used and make it practical in a variety of similar vocabularies. It differs from [XML Linking Language (XLink) Version 1.0] in the following ways:

1. The `xlink:type` attribute is no longer required for simple links. In the absence of any `xlink:type` attribute, an XLink is treated as a simple link.
2. Where [XML Linking Language (XLink) Version 1.0] referred to URIs, this specification refers to IRIs. This allows a broader range of values for for those properties that are identified with a resource identifier.
3. This specification includes non-normative sample XML Schema and RELAX NG grammars to complement the existing, non-normative sample DTD.

In addition, a few editorial changes have also been made.

1. Some bibliographic references have been updated to point to more recent specifications.
2. The conformance language has been rewritten to support a new, simple conformance level for applications that only expect to process simple links.
3. Several of the examples have been changed to highlight the presence of non-normative grammars other than the DTD.
4. The text describing the interpretation of the locator attribute (`xlink:href`) has been moved into a separate specification ([Legacy extended IRIs]) so that it may more easily be reused. The locator attribute is now described with reference to that specification.