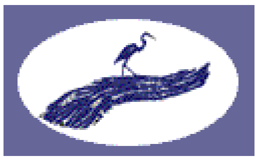
XML Linking Language(XLink)

Creating Powerful and Flexible Hypertext structures



Simon St.Laurent

Geek Cruises XML Excursion - January 2001

Table of Contents

Hypertext for the 1970s

XLink promises a basic level of functionality, but more than HTML

- **Myth** HTML's hypertext functionality provided developers with a magical new set of tools for weaving hypertext webs.
- **Reality** HTML's hypertext tools are remarkably simple, even crude. As sites grow larger, link management becomes incredibly complicated.
- **Promise** XLink will provide a next generation of linking tools for XML that are more sophisticated than HTML but still simple enough for people and programs to manage.

The Problem with Hypertext

- **Complex for Both Readers and Authors** Building hypertexts that are more sophisticated than simple outlines can be very difficult, requiring authors to keep track of potentially moving links. Readers may simply get lost in the links.
- **Interface Creation is Difficult** Formatting links can be complex, especially links that may lead to multiple locations. If readers can't distinguish links from other content, they may quickly grow frustrated. (Sometimes this is fun to do deliberately, however.)
- **Management is difficult** Keeping track of links is extremely difficult when the texts they connect are in flux.
- **Ambition** It's easy to get ambitious with hypertext and try to set up things like 'conditional' links that only become active when other links have been followed. This makes management even harder.

HTML's Magical Simplification

Doing More by Doing Less

- **Unidirectional Links** HTML links are simple - they go from 'here' to 'there'. The 'there' is specified by a URL,and the 'here' is the link's location in the document.
- **Every Page an Island** Pages don't need to know anything about links directed toward them. Their URLs are sufficient identifiers, though authors *may* supplement those with internal anchors.
- **No Nested Links** HTML prohibits links within links, insisting upon a simple model for one-dimensional links. There is no 'default' link for a page, or multiple layers of linking possibilities.
- **Simplicity Encourages Development** HTML finally made hypertext easy enough that it could reach critical mass.

Exploding Webs of Hypertext

Critical Mass Creates an Unmanageable Explosion

- **The New Flexibility** HTML made it possible to create an enormous number of different site architectures. The hierarchical models of Gopher and FTP directories remained available, but any kind of cross-linking was possible. Perhaps most important was the ability to link to other sites without having to coordinate linking, which helped the Web bind tighter and made splitting up site design simpler.
- **Enough Power to be Pretty** The IMG tag, which opened enormous new commercial and other possibilities, is itself a hyperlink, including images in documents by reference.
- **Everyone an Architect** HTML lowered the barriers to hypertext design significantly, letting all kinds of people, from programmers to graphic designers, become Web site architects.

The Pain of Document Management

How much do broken links cost?

- **Sites Keep Growing** In 1996, a project manager I knew kept a 10,000-page site and its links in her head, managing changes throughout the site remarkably well. 10,000 pages is no longer unusual, but people who can manage 10,000 pages by hand are.

- **Link Tracking** One of the the most popular early tools for HTML management was the link checker, which would explore the site and find problems. Unfortunately, it could rarely suggest ways of fixing them.
- **404** As sites grow larger and material gathers dust or is discarded, broken links have become more and more common, even (perhaps especially) on sites that focus on links.

Moving Beyond HTML

Hypertext Needs a Sounder Foundation

- **Building on HTML** The enormous number of developers using HTML has familiarized a large audience with the use of markup technology. < and > are familiar to Web developers, as are the concepts of elements, attributes, and nesting.
- **Moving Beyond Documents** HTML was designed for use with a particular document format, and has significant limitations even for documents. Data applications that need to reference information using hypertext have been forced into this usually inappropriate document format.
- **Customizable yet Standardized** XML promises open markup vocabularies that use generic syntax and processing, making it possible to move beyond HTML yet still have generic hypertext processing.

Linking and Open Vocabularies

Creating a Linking Vocabulary That Doesn't Conflict

* **'A' doesn't live here anymore** The A, LINK, and IMG elements are just names to XML, which doesn't automatically associate any kind of behavior with element or attributes. Creating a new vocabulary requires building a meta-vocabulary developers can integrate with their own applications.
* **Namespaces** Developers who just want links but don't need to customize them may be able to use the (controversial) XML namespaces to accomodate linking information without having to create their own linking vocabulary. XLink creates a set of *global attributes* for use across documents.
* **Integrated Linking** On the other hand, developers who want to build generic linking may want to combine linking functionality with their own vocabularies. Earlier drafts allowed developers to map XLink attribute names to any attribute names; the current draft requires developers to use the XLink names, though these operate in their own namespace to minimize disturbance. It appears that the earlier requirement of integration has been dropped.
* **Building on Schemas and DTDs** XLink functionality can be added to your application with help from the defaulting features found in XML 1.0 DTDs and the upcoming schemas. There are some risks, however.

A New Approach for XML

Actually, Two Approaches

* **Building on SGML/HyTime** XLink uses ideas from the much more sophisticated (and much more complicated) SGML HyTime standard for hypertext linking, stripping them down to

a much simpler set of tools.

- **Architectural Forms for Integration** One of the key tools from HyTime, architectural forms, provided the original backbone of XLink.
- **Namespaces for Unique Identifiers** More recent drafts of XLink use XML namespaces to keep the XLink vocabulary out of the way of other element names. XLink's set of global attributes can be used with any DTD or schema, and don't have a DTD or schema of their own.

Simple Links

Backward-Compatible and Easy

- **Forward Motion** Simple links allow documents to reference other documents, with no expectation that those documents will reference back to them.
- **No intelligence beyond the page** Applications only need to know about simple links while the page containing them is open. Presenting and processing simple links requires only that a page processor recognize, present, and act upon them.
- **Backward compatible** Simple links allow the recreation of most (though not all) HTML linking capabilities.

Recreating A HREF=

The parts of a simple link

Attribute	Notes
xlink:type	When assigned the value 'simple', identifies the link as a

	'simple' link. The link is inline and unidirectional and contains a single target destination in an href attribute.
xlink:href	Contains the URI that represents the target of this link.
xlink:role	Provides a "description of the link's content". For simple links, could be used to style different kinds of links differently. Value must be a URI.
xlink:arcrole	Describes the resource at the end of the xlink:href URI. For simple links, could be used to style different targets of links differently. Value must be a URI.
xlink:title	Provides a human-readable title for the link, pretty much like the popup text that appears now in some browsers.
xlink:show	Provides the options new (appear in new window), embed (included in text), and replace (appears in the current window) for the link content, as well as other and none.
xlink:actuate	Offers the choices onRequest (wait for user activation) and onLoad (process automatically) to indicate when link processing should happen. other and none provide additional application-dependent options.

Recreating A HREF=

Implied XLink Simple Link

Simple links are specified using an attribute value using xlink:type. An example simple link using xlink:type as an attribute is shown below:

```
<mylink xlink:type="simple" xlink:href="http://www.simonstl.com" xlink:title="Author site"
xlink:role="http://www.simonstl.com/idURIs/authorlink" xlink:show="new"
xlink:actuate="onRequest" >Simon's Site</xlink:simple>
```

This would create a link, containing the text "Simon's Site", that would point to www.simonstl.com. The title, which might pop up on a mouseover, would be "Author site", and its role (in case anyone, perhaps a style sheet cared) would be "http://www.simonstl.com/idURIs/authorlink". When clicked on by a user, a new window would appear with the contents of www.simonstl.com. For the most part, this approach allows to use your own element names, but you are still required to use the attribute names, complete with full namespace prefixes.

Moving from Paths to Sets

- **Multidirectional Links** While simple links do a great job of moving users from one place to another, they require links all over the place - the 'back' button is the most sophisticated navigation tool most browsers provide.
- **Sophisticated Modeling** 'Linking' may be used metaphorically, as well as to describe hypertext connections. By providing a framework for describing sets, extended links permit more powerful modeling.
- **Fewer links, more connections** Authors can write fewer actual links while representing more paths. Because extended links can represent multiple connections between resources, the number of possible connections grows at n^2 rather than n, where n is the number of nodes.
- **Memory** Browsers need to remember links from previous pages (or reload them constantly) for genuinely multi-directional links to work.

Moving Links Out of the Way

'Out-of-line' links

- **Easier to Manage** Out-of-line links allow developers to centralize link descriptions. Instead of including the links in the document content, developers can create links that only describe the resources they connect, and let the application handle the actual connecting.

- **Requires Extra Processing** While they make life easier for humans, out-of-line links require applications to do more work, possibly involving a second pass over the document structure to mark links.
- **'Resource' Mechanism** The latest XLink draft uses 'resource' type child elements to identify in-line links. Resources are pretty much locators without an xlink:href attribute.

Creating Extended Links

- **Sets of Locators** Extended links use containment to identify sets of locators (and occasionally local resources). The set is defined by a parent element, which contains the properties for the set, while elements within that parent may be locators or resources.
- **Set Properties** The set itself has a role and a title. This information can be used to build link directories outside of the context of particular documents and to define the type of link within the document.
- **Locator Properties** Locators contain the all-important href attribute, as well as role, title, and IDs. The href attribute is really the identifying feature of a locator, providing a URL pointer to resources.
- **Arcs** Extended links may also contain arcs, which identify connections between particular locators. They provide detailed information on traversal paths, including show and actuate information.

Creating Extended Links

Extended Linking Element Attributes

Name	Notes
xlink:type	Elements containing the attribute xlink:type with the value 'extended' should be treated as extended links by XLink processors.
xlink:role	Provides a "description of the link's content". For extended links, might be used to specify different kinds of processing for different types of locator sets. The role must be a URI.
xlink:title	Provides a human-readable title for the link, pretty much like the popup text that appears now in some browsers. Child elements with an xlink:type attribute set to title may also be used in cases where multiple titles are needed, perhaps for multi-language scenarios.

Creating Extended Links

Locator Element Attributes

Name	Notes
xlink:type	Elements containing the attribute xlink:type with the value 'locator' should be treated as locator elements by XLink processors.
xlink:href	Contains the URI, the target of this link.
xlink:role	Provides an identifier describing the locator's content. Must be a URI.
xlink:label	Provides an identifier that can be used to identify how different types of resources and locators are connected (by arcs) within a link.
xlink:title	Provides a human-readable title for the locator, pretty much like the popup text that appears now in some browsers.

Creating Extended Links

Resource Element Attributes

Name	Notes
xlink:type	Elements containing the attribute xlink:type with the value 'resource' should be treated as resource elements by XLink processors
xlink:role	Provides an identifier describing the resource's content. Must be a URI.
xlink:label	Provides an identifier that can be used to identify how different types of resources and locators are connected (by arcs) within a link.
xlink:title	Provides a human-readable title for the locator, pretty much like the popup text that appears now in some browsers.

Creating Extended Links

Attribute-identified XLink Extended Link

An extended link, described using xlink:type, is shown below:

```
<pictures xlink:type="extended" xlink:title="Wedding Pictures - Charlie"
xlink:role="http://www.simonstl.com/idURIs/scrapbook">
<photo xlink:type="locator" xlink:href="toast.scr" xlink:label="toast" xlink:title="Charlie
leads the toast" />
<photo xlink:type="locator" xlink:href="garter.scr" xlink:label="garter" xlink:title="Charlie
brings out the garter"/>
<photo xlink:type="locator" xlink:href="cakemess.scr" xlink:label="cake" xlink:title="Cake
down Charlie's shirt" />
<address xlink:type="locator" xlink:href="address.adr" xlink:label="address"
xlink:title="Charlie's Address" />
</pictures>
```

Note that we're not doing anything with the labels here, but we will be using them as hooks later.

Demonstration - Image Maps

- **Demonstration** available at http://www.simonstl.com/projects/xlinkfilter/code.html#example

The Behavior Question

show and actuate?

- **An architectural question** Some in the XML community oppose the inclusion of 'show' and 'actuate', arguing that this functionality is more appropriate to the realm of style sheets and scripts.
- **Not always there** Style sheets aren't used in all XML processing, nor is script processing always available. (The previous demonstration, for example.) On the other hand, applications may not always be able to support the choices presented by XLink, either.
- **A useful hook** While it may not meet everyone's standards for architectural purity, the show and actuate attributes may prove useful as a foundation on which 'real' behavior can be built. Recent drafts allowing applications to use other values open these possibilities further.

Programming for Links

Building Sets of Graphs

- **Generic Linking** If and when the world standardizes on XLink, developing processors for hypertext linking shouldn't be the custom job it is today. Link extraction and processing *should* become a task for libraries rather than precisely crafted software.
- **Interpretation** The current specification leaves a lot of questions open, especially about processing extended links. Answering those questions in a way that can move across applications is (presently) difficult.
- **Linking as Integration** XLink promises to let developers treat the relationships beween XML documents much the way they treat XML: as part of a solution that can be mapped into many different application areas. XLink provides a layer on top of XML that allows developers to work with sets of documents as well as individual documents.

Styles for Links

Linking as Transformation Tool

- **How strong a stylesheet?** Advocates of a minimalist XLink have long argued that extended link processing - the conversion of sets of resources into traversable paths - is something that should be handled by another layer, most likely the style sheet layer. While Extensible Stylesheet Language (XSL) may be capable of handling some of this work, Cascading Style Sheets (CSS) is definitely not oriented this direction.
- **Memory** 'Multi-directional' links require applications to keep track of links as users traverse them. At present, there is no way for an XSL stylesheet to keep track of this kind of information, though extensions may allow applications to help follow the information.

- **Multiple trees** XSL processing is complicated by the fact that link locations in the original document need to be mapped to the results of the transformation.
- **Are links presentation or content?** No one really knows - or at least most everyone disagrees.

The Question of Arcs

Moving Toward Complete Extended Link Processing

- **From Sets to Graphs** After considerable debate on the xlxp-dev mailing list and internally at the W3C, the XLink Working Group added Arcs to extended links. Arcs (represented by `xlink:type="arc"`) allow developers to describe paths between the resources identified by locators within an extended link.
- **XArc** One proposal would have replaced the entire framework of XLink with a simple origin-destination scheme. The distinction between simple and extended links would have vaporized, and links could be in-line or out-of-line as appropriate.
- **Compromise** The XLink Working Group's proposal allows the use of arcs inside of extended links, and connects them to label values on locators and resources. If wildcarding isn't used, so specification of traversal paths is n^2 where n is the number of labels. (This is much better than previous versions, which used ID values for *each* locator.) Missing values for `from` and `to` are considered to represent 'all'.

The Question of Arcs

Moving Part-way Toward Extended Link Processing

Name	Notes
xlink:type	Elements containing the attribute xlink:type and the value 'arc' should also be treated as arcs by XLink processors.
xlink:from	Identifies the origin, using a label value from one of the locator or resource elements in the extended link. (If no value appears, defaults to all locators and resources.)
xlink:to	Identifies the destination, using a label value from one of the locator or resource elements in the extended link. (If no value appears, defaults to all locators and resources.)
xlink:arcrole	Provides an identifier describing the arc. Must be a URI.
xlink:title	Provides a human-readable title for the arc, pretty much like the popup text that appears now in some browsers.
xlink:show	Provides the options new (appear in new window), embed (included in text), and replace (appears in the current window) for the link content, as well as other and none.
xlink:actuate	Offers the choices onRequest (wait for user activation) and onLoad (process automatically) to indicate when link processing should happen. other and none provide additional application-dependent options.

The Question of Arcs

Attribute-identified XLink Extended Link, Plus Arcs

```
<pictures xlink:type="extended: xlink:title="Wedding Pictures - Charlie"
xlink:role="scrapbook">
<photo xlink:type="locator" xlink:href="toast.scr" xlink:title="Charlie leads the toast"
xlink:label="toast" />
<photo xlink:type="locator" xlink:href="garter.scr" xlink:title="Charlie brings out the garter"
xlink:label="garter"/>
<photo xlink:type="locator" xlink:href="cakemess.scr" xlink:title="Cake down Charlie's shirt"
xlink:label="cake" />
<address xlink:type="locator" xlink:href="address.adr" xlink:title="Charlie's Address"
xlink:label="address"/>
<link xlink:type="arc" from="toast" to="address" />
<link xlink:type="arc" from="garter" to="address" />
<link xlink:type="arc" from="cake" to="address" />
</pictures>
```

Note the references to `xlink:label` values. Connections here are between the pictures and the address only.

Managing Links - Extended Link Groups

Centralizing Links for Easier Management

- **Documents linking other documents** Extended link groups allow developers to put all of their links in a single file, rather than having links spread out among other files. All links in this model are extended and out-of-line.
- **Just an arcrole** To identify a linkbase, create an extended link that includes a locator for the linkbase, and then create an arc whose `xlink:arcrole` attribute is set to `http://www.w3.org/1999/xlink/properties/linkbase` .

Legal and Political Implications of Multidirectional Links

When Out-of-line is 'out of line'

- **Linking to content you don't control** Multidirectional links could conceivably originate from content outside of your site. Other users can provide annotations on their own pages, which users traveling from those pages to your pages will see.
- **Already enough lawyers** Even 'traditional' HTML linking has been enough to generate lawsuits. Being able to link documents externally and reference fragments (through XPointer) is going to bring out a whole new set of issues.
- **Potentially offensive content** What's annotation and what's graffiti? You decide.

Alternative Approaches - Weaknesses of XLink

Staying on the W3C's bus?

- **The IMG problem** Even simple-seeming elements like the HTML IMG element can't be represented by XLink, as IMG contains two locators, SRC and LONGDESC. It doesn't seem unreasonable to have to locators as attributes, but XLink is too brittle to support it. (Even in earlier versions of XLink, which were more flexible, this was a problem.)
- **The RDF problem** Reconciling XLink and RDF, both of

which describe relationships among resources, is a fairly complex and controversial task, best avoided.

- **The Stylesheet problem** Are graphic designers really the people who should be designing stylesheets for handling link processing?
- **Waiting for the W3C** The XLink group spent 15 months between working drafts, though recent drafts have moved much more quickly.

Alternative Approaches - Non-XLink Solutions

- **XArc** Gabe Beged-Dov originally created XArc in response to discussions on the XLink/XPointer (xlxp-dev) mailing list. XArc was a radical simplification, providing a from/to locator approach. It's been integrated with XLink to some extent.
- **RDF** The W3C's Resource Description Format (RDF) is widely regarded as the hardest piece of XML. Nonetheless, it's simple foundations on directed graphs and freer rules for describing properties make it as capable, perhaps more capable, than XLink for describing complex relationships. For the most part, though, it's stuck in metadata.

Moving Toward Generic Hypertext Applications

Still hoping for a silver lining

- **W3C Support** The W3C's own standards are still barely using XLink. Getting XLink built into W3C standards seems like a good first step, but it may be a long road. Similarly, even the W3C's Amaya browser lacks generic XML and XLink support.

- **Browser Support** Microsoft has remained very quiet about XLink support. Mozilla does support simple links, though for an older draft.
- **Application Support** XLink has potential for a wide range of data-oriented XML tasks within traditionally non-hypertext applications. Whether or not XLink provides enough tools for that work and whether application developers will see XLink as a plus remains to be seen.

Questions

- The XLink Specification - http://www.w3.org/TR/xlink/ - is the root of all of this work.
- The XPointer Specification - http://www.w3.org/TR/xptr - is also worth a look, providing tools used to specify locators smaller than a complete document (and way beyond #name).
- The XArc Specification - http://www.jfinity.com/xarc/ - (not a W3C specification, but an open project) was one alternative to XLink.
- Robin Cover's XLL page - http://www.oasis-open.org/cover/xll.html - lists many resources for XLink development.
- Lloyd Rutlege's Hypermedia Research Links - http://www.cwi.nl/~lloyd/Hypermedia/index.html - include a wealth of Web and multimedia information.
- Elliotte Rusty Harold did a presentation - http://metalab.unc.edu/xml/slides/xmlsig0899/ - for the New York Object Developers' Group on XLink, XPointer, and XPath on 24 August 1999.
- Archives of xlxp-dev - http://www.fsc.fujitsu.com/hybrick/xlxp-dev/maillist.html - (XLink/XPointer Discussion)
- Archives of http://lists.w3.org/Archives/Public/www-xml-linking-comments/ - the W3C XLink list