

Extensible Markup Language (XML)

Part 2: Linking

Version 1.0

W3C Working Draft April-06-97

This version

<http://www.w3.org/pub/WWW/TR/WD-xml-link-970406.html>

Previous versions

Latest version

<http://www.textuality.com/sgml-erb/WD-xml-link.html>

Editors

Tim Bray, Textuality (tbray@textuality.com)

Steve DeRose, Inso Electronic Publishing Solutions (sderose@inso.com)

Status of this document

Please be advised that the draft you are now reading is unusually volatile. The debating and balloting process which determines the material contents is far from complete, and is nonetheless substantially ahead of the editing process that turns the material contents into usable specification language.

This is a W3C Working Draft for review by W3C members and other interested parties. It is a draft document and may be updated, replaced or obsoleted by other documents at any time. It is inappropriate to use W3C Working Drafts as reference material or to cite them as other than "work in progress". A list of current W3C working drafts can be found at <http://www.w3.org/pub/WWW/TR>.

Note: Since working drafts are subject to frequent change, you are advised to reference the above URL, rather than the URLs for working drafts themselves.

This work is part of the W3C SGML Activity (for current status, see <http://www.w3.org/pub/WWW/MarkUp/SGML/Activity>).

Abstract

This document specifies a simple set of constructs that may be inserted into XML documents to describe links between objects. It is a goal to use the power of XML to create a structure that can describe both the simple unidirectional hyperlinks of today's HTML, as well as more sophisticated multi-ended, typed, self-describing links.

Table of Contents

1. Introduction	1
1.1 Origin and Goals.....	1
1.2 Relationship to Existing Standards	1
1.3 Unfinished Work	1
1.3.1 Structured Labels	1
1.3.2 Element/Attribute Remapping	1
1.4 Terminology	1
1.5 Notation.....	2
1.6 Types of link types.....	2
2. Link Recognition.....	3
2.1 Operational Issues Concerning Link Recognition.....	3
3. Linking Elements	3
3.1 Information Associated With Links	3
3.2 Simple Links	4
3.3 Extended link.....	4
4. Link Behavior	5
4.1 The SHOW Axis.....	5
4.2 The ACTUATE Axis	5
5. Addressing	5
5.1 Locator Syntax in General.....	5
5.2 Locator Syntax for XML Resources	6
5.3 TEI Extended Pointers	6
5.3.1 XPointer Structure	7
5.3.1.1 The ROOT Keyword.....	7
5.3.1.2 The HERE Keyword	7
5.3.1.3 The DITTO Keyword.....	7
5.3.1.4 Addressing by ID Attribute	8
5.3.2 Location Terms - The Keyword Phase	8
5.3.3 Location Terms - The Steps Phase.....	8
5.3.4 Interaction of Keywords and Steps	10
5.3.4.1 The DESCENDANT Keyword.....	10
5.3.4.2 The ANCESTOR Keyword	11
5.3.4.3 The PREVIOUS Keyword.....	11
5.3.4.4 The NEXT Keyword	11
5.3.4.5 The PRECEDING Keyword.....	12
5.3.4.6 The FOLLOWING Keyword.....	12
6. Extended Link Groups.....	12
7. References.....	12

1. Introduction

This document specifies a set of constructs which may be inserted in XML documents to describe links between objects. A link, as the term is used here, is a relationship which is asserted to exist between two or more data objects or portions of data objects. This specification is concerned with the syntax used to assert link existence and describe link characteristics. Implicit (unasserted) relationships, for example that of one word to the next, or that of a word in a text to its entry in an on-line dictionary, are outside its scope. Explicitly asserted links do not constitute the only useful kind of link, but this specification is intended neither to provide machinery for every possible kind of link nor to preclude the use of such machinery.

The existence of links is asserted by the presence of elements contained in XML documents. They may or may not reside at the locations of, or in the same documents with, the objects which they serve to connect.

1.1 Origin and Goals

This specification aims to provide an effective, compact structure for representing links that can be in documents or external to them, that can have multiple typed locators, indirection, and precise specification of resource locations in XML and SGML data. It also aims to represent the abstract structure and significance of links, leaving formatting issues to stylesheets or other mechanisms as far as practical.

1.2 Relationship to Existing Standards

Three standards have been especially influential:

- HTML: Defines several SGML element types that represent links, as well as popularizing a location specifier type, the URL, mainly focused on pointing to entire data objects, though with some provision for linking to elements with IDs, regions in graphics, and so on.
- HyTime: Defines location specifier types applicable to all kinds of data, as well as in-line and out-of-line link structures and some semantic features including traversal control and placement of objects into a display or other space.
- Text Encoding Initiative *Guidelines* (TEI P3): Provide a formal syntax for location specifiers for structured data, graphics, and other data, and structures for creating links and link collections out of them.

Many other linking systems have also informed this design, including Dexter, MicroCosm, and InterMedia.

1.3 Unfinished Work

1.3.1 Structured Labels

The simple labelling mechanism described in this draft is insufficiently flexible to cope with internationalization or the use of multimedia in link labels. A future version will provide a mechanism for the use of structured link labels.

1.3.2 Element/Attribute Remapping

The technique whereby linking elements are recognized based on the use of special attributes is described with insufficient thoroughness. Furthermore, the idea of allowing the attributes of linking elements, as well as their element types, to be remapped to user-chosen values, is under serious consideration.

1.4 Terminology

The following basic terms apply in this document.

- resource: In the abstract sense, an addressable unit of information or service which is participating in a link. Examples include files, images, documents, programs, and query results. Concretely, anything which happens to be reachable by the use of a locator in some linking element.
- linking element: An element which asserts the existence and describes the characteristics of a link.
- locator: A character string appearing in a linking element, which may be used to locate a resource.
- label: A caption associated with a resource, suitable for showing users as a means of explaining the significance of the part played in the link by that resource.
- traversal: The action of using a link; that is, of accessing a resource. Traversal may be initiated by a user action (commonly initiated by clicking on a displayed portion of a linking element) or occur under program control.
- multi-directional link: A link that can be traversed starting at more than one of its resources. Note that being able to "go back" after following a one-directional link does not make the link multi-directional.
- in-line link: A link which serves as one of its own resources. HTML `<A>`, HyTime *clink*, and TEI `<XREF>` are all examples of in-line links.
- out-of-line link: A link which does not serve as one of its own resources (except perhaps by chance). Such links only make sense given a notion like link groups, which instruct applications where to look for links. Nevertheless, out-of-line links are required to support multi-directional traversal and for creating links with resources which can be traversed starting from read-only data objects.

1.5 Notation

The formal grammar for locators is given using a simple Extended Backus-Naur Form (EBNF) location, as described in Part 1.

1.6 Types of link types

There is an extensive literature on link typology. Some well-known axes are:

- link relationships: Links express various kinds of relationships between the data objects or portions they connect, in terms of conceptual significance to the author and user. Some links may be criticisms, others add support or background, while others have a very different meaning such as providing access to demographic information about a data object (its author's name, version number, etc), or to navigational tools such as index, glossary, and summary.
- link topology: In-line and out-of-line links differ in their structure, as do links involving varying numbers of resources.
- locator language: There are many languages available for use in locating resources; examples are URLs, SQL queries, and file names.
- formatting: Links may be presented in a variety of ways. The discussion of this area is complicated by the fact that link formatting and link behavior are inextricably linked. This specification does not discuss, nor provide mechanisms for, the provision or use of link formatting information.
- link behavior: Links may have a wide variety of effects when traversed, such as opening, closing, or scrolling windows or panes; displaying the data from various resources in various ways; testing, authenticating, or logging user and context information; executing various programs. Ideally, link behavior should be determined by a semantic specification based on link types, resource roles, user circumstances, and other factors; just as element formatting is determined by a stylesheet based on element type, context, and other factors.

2. Link Recognition

The existence of a link is asserted by a linking element. These must be recognized reliably by software in order to provide appropriate display and behavior. XML linking elements are recognized based on the use of a designated attribute named `XML-LINK`. Possible values are `SIMPLE`, `EXTENDED`, `LOCATOR`, `GROUP`, and `DOCUMENT`, signalling in each case that the element in whose start-tag the attribute appears is to be treated as an element of the indicated type, as described in this specification.

An example of such a link:

```
<A XML-LINK="SIMPLE" HREF="http://www.w3.org/">The W3C</A>
```

2.1 Operational Issues Concerning Link Recognition

There are two distinct mechanisms that may be used to associate the `XML-LINK` attribute with a linking element. The simplest is to provide it explicitly, as in the example above. However, this practice is verbose, and would be not only cumbersome but wasteful of network bandwidth in the case where there are large numbers of linking elements. Fortunately, XML's facilities for declaring default attribute values can be used to address this problem. For example, suppose one wished to declare the "A" element to be an XML `SIMPLE` link. The following would accomplish this:

```
<!ATTLIST A XML-LINK CDATA #FIXED "SIMPLE">
```

Such a declaration may be placed in either the external or internal subsets of the Document Type Declaration. Placing it in both subsets would be the obvious thing to do for convenient network operation. So doing, at the time of creation of this specification, would cause the document to fail to be valid. Note that the successful completion of the current work on a technical corrigendum to ISO 8879 that is in the process of submission to ISO/IEC JTC1/SC18/WG8 would resolve this problem and allow this practice in valid documents. However, for interoperability, the declaration should not be placed in both subsets.

3. Linking Elements

This specification defines two types of linking elements. First, a simple link, which is always in-line and one-directional, very like the HTML `<A>` element. Second, a much more general extended link (`EXTENDED`) which is out-of-line and may be used for multi-directional links, links into read-only data, and so on.

3.1 Information Associated With Links

This specification describes a variety of information that may be (and in some cases is required to be) associated with linking elements:

Role

Every link has a role, a string used to identify to the application program the meaning of the link. Furthermore, each resource participating in a link may be given its own role. The `ROLE` attribute is used to provide both link and resource roles.

Resource

Every locator must identify a resource in some fashion. This is done using the `HREF` attribute, as described below.

Label

Every locator may be associated with a label in the `TITLE` attribute. This specification does not require that applications make any particular use of the label.

Behavior

The SHOW and ACTUATE attributes may be used for an author to communicate general policies concerning the traversal behavior of the link; this specification defines a small set of policies for this purpose. The BEHAVIOR attribute may be used to communicate detailed instructions for traversal behavior; this specification does not constrain the contents, format, or meaning of this attribute.

3.2 Simple Links

Simple links are very much like HTML <A> or TEI <XREF> elements, but with more general reference capabilities. A simple link may contain only one locator; thus there is no necessity for a separate child element, and the locator attributes are attached directly to the linking element.

Following is a declaration for an XML simple link; note that the element type need not be SIMPLE, since the linking element will be recognized based on the value of the XML-LINK attribute:

```
<!ELEMENT SIMPLE ANY>
<!ATTLIST SIMPLE
    XML-LINK CDATA #FIXED "SIMPLE"
    ROLE CDATA #IMPLIED
    HREF CDATA #REQUIRED
    TITLE CDATA #IMPLIED
    SHOW ( EMBED | REPLACE | NEW ) "REPLACE"
    ACTUATE ( AUTO | USER ) "USER"
    BEHAVIOR CDATA #IMPLIED
>
```

3.3 Extended link

A extended link can involve any number of resources, and need not be co-located with any of them. An application may be expected to provide traversal among all of them (subject to semantic constraints outside the scope of this paper). The key issue with extended links is how to manage and find them, since they do not necessarily co-occur with any of their resources, and often are located in completely separate documents. This process is discussed under Extended Link Groups below.

A extended link's locators are contained in child elements of the linking element, each with its own set of attributes. Once again, in the following declaration, the linking element type need not be EXTENDED and LOCATOR:

```
<!ELEMENT EXTENDED ( #PCDATA | LOCATOR ) * >
<!ELEMENT LOCATOR ANY>
<!ATTLIST EXTENDED
    XML-LINK CDATA #FIXED "EXTENDED"
    ROLE CDATA #IMPLIED
    TITLE CDATA #IMPLIED
    SHOW ( EMBED | REPLACE | NEW ) "REPLACE"
    ACTUATE ( AUTO | USER ) "USER"
    BEHAVIOR CDATA #IMPLIED
>
<!ATTLIST LOCATOR
    XML-LINK CDATA #FIXED "LOCATOR"
    ROLE CDATA #IMPLIED
    HREF CDATA #REQUIRED
    TITLE CDATA #IMPLIED
    SHOW ( EMBED | REPLACE | NEW ) "REPLACE"
    ACTUATE ( AUTO | USER ) "USER"
    BEHAVIOR CDATA #IMPLIED
>
```

Note that many of the attributes may be provided for both the parent linking element and the child locator element. If any such attribute is provided in the linking attribute but not in a locator element, the value

provided in the linking element is to be used in processing the locator element. In other words, the attributes provided in the linking element may serve as defaults for the (possibly many) locator elements.

4. Link Behavior

This specification provides a mechanism for the authors of linking elements to signal their intentions as to the timing and effects of traversal. Such intentions can be expressed along two axes, labeled *SHOW* and *ACTUATE*. These are used to express *policies* rather than *mechanisms*; programs which are processing links in XML documents are free to devise their own mechanisms, best suited to the user environment and processing mode, to implement the requested policies.

In many cases, there will be a requirement for much finer control over the details of traversal behavior; existing hypertext software typically provides such control. Such fine control of link traversal is outside the scope of this specification; however, the *BEHAVIOR* attribute is provided as a standard place for authors to provide, and in which programs should look for, such detailed behavioral instructions.

4.1 The *SHOW* Axis

The *SHOW* attribute is used to express a policy as to the context in which a resource that is traversed to should be displayed or processed. It may take one of three values:

EMBED

directs that upon traversal of the link, the designated resource should be embedded, for the purposes of display or processing, in the body of the resource, and at the location, where the traversal started.

REPLACE

directs that upon traversal of the link, the designated resource should, for the purposes of display or processing, replace the resource where the traversal started.

NEW

directs that upon traversal of the link, the designated resource should be displayed or processed in a new context, not affecting that of the resource where the traversal started.

4.2 The *ACTUATE* Axis

The *ACTUATE* attribute is used to express a policy as to when traversal of a link should occur. It may take one of two values:

AUTO

directs that the link should be traversed when encountered, and that the display or processing of the resource where the traversal started is not considered complete until this is done.

USER

directs that the link should not be traversed until there is an explicit external request for this to happen.

5. Addressing

The locator value for a resource is provided in the *HREF* attribute. *HREF* may have at one point stood for "Hypertext reference"; the name is adopted for compatibility with existing practice.

5.1 Locator Syntax in General

In general, a locator contains a URL, as described in RFC 1738. As that and related RFCs state, the URL may be followed by "?" and a *query*, or by "#" and a *fragment identifier*, with the query interpreted by the

host providing the indicated resource, and the interpretation of the fragment specifier dependent on the data type of the indicated resource. Thus, when a locator in an XML linking element identifies a resource that is not an XML document (for example, an HTML or PDF document), this specification does not constrain the syntax or semantics of the query nor of the fragment specifier.

However, this specification does provide, starting in the next section, a complete description of query and fragment identifier syntax and semantics in the case when the indicated resource is an XML document.

5.2 Locator Syntax for XML Resources

When a locator identifies a resource that is an XML document, the locator value may contain either or both a URL and a TEI Extended Pointer (hereinafter XPointer). Special syntax may be used to request the use of particular processing models in accessing the locator's resource. This is designed to reflect the realities of network operation, where it may or may not be desirable to exercise fine control over the distribution of work between local and remote processors.

Locator

[1]	Locator	::=	URL Connector (XPointer Name) URL Connector (XPointer Name)
[2]	Connector	::=	'#' ' ' '?XML-XPTR='
[3]	URL	::=	URLchar*

In this discussion, the term designated resource refers to the resource participating in the link which the locator serves to locate. The following rules apply:

- The URL, if provided, locates a resource called the containing resource.
- If the URL is not provided, the containing resource is considered to be the document in which the linking element is contained.
- If the Connector is followed by a Name, the Name is shorthand for the XPointer "ID(Name)"; i.e. the sub-resource is the element in the containing resource that has an XML ID attribute whose value matches the Name. This shorthand is to encourage use of the robust ID addressing mode.
- If the XPointer is provided, the designated resource is a "sub-resource" of the containing resource; otherwise the designated resource is the containing resource.
- If the connector is "#", this signals an intent that the containing resource is to be fetched as a whole from the host that provides it, and that the XPointer processing to extract the sub-resource is to be performed on the client, that is to say on the same system where the linking element is recognized and processed.
- If the connector is "?XML-XPTR=", this signals an intent that the entire locator is to be transmitted to the host providing the resource, and that the host should perform the XPointer processing to extract the sub-resource, and that only the sub-resource should be transmitted to the client.
- If the connector is "|", no intent is signaled as to what processing model is to be used to go about accessing the designated resource.

5.3 TEI Extended Pointers

XML uses a locator syntax derived from that for TEI extended pointers. These operate in a straightforward way on the parse tree which is defined by the elements of an XML document.

The basic form of such a locator is a series of location terms, each of which specifies a location, either absolute or (more frequently) relative to the prior one. Each term has a name, such as ID, CHILD, ANCESTOR, and so on, and can be qualified by parameters such as an instance number, element types, or attributes. For example, the locator string CHILD(2,CHAP)(4,SEC)(3) refers to the 3rd child of the 4th SEC within the 2nd CHAP within the referenced document.

Note: Formally, these may be specified as operating on groves as defined in DSSSL, using the grove plan (set of structural information) specified in HyTime. Every construct in such locators has a corresponding expression in DSSSL's SDQL query language, and most also have direct equivalents in the HyTime location module.

The syntax for TEI Extended Pointers has been adjusted in order to allow them to be packaged naturally with URLs without requiring URL-escaping of space characters:

- Parameters in locator terms must be separated by commas, to facilitate including Xpointers within URLs, where spaces would otherwise need to be escaped.
- A locator may contain two Xpointers, separated by the string ". . ". These define the beginning and end of a span which constitutes the resource. This merely combines the capability of the TEI `FROM` and `TO` attributes into the locator syntax.

5.3.1 XPointer Structure

At the heart of the XPointer is the location term. Location terms are designed to work in sequences. Each location term works in the context of a *location source*, a location or element in the document. The location term itself provides instructions which, based on the location source context, navigate to another location in the document, establishing the location source for the next location term.

The location source for the first location term in a locator is, by default, the root element of the XML document which is the containing resource.

The result of evaluating a location term may be (in this version of the specification, always is) an element.

A locator can contain either one or two series of location terms; if there are two, they are separated by the string ". . ". If it contains one, its designated resource is the element or location selected by that series. If it contains two, its designated resource is all of the text from the location, or start of the element, selected by the first series, through to the location, or the end of the element, selected by the second series:

XPointer

[4]	XPointer	::= <i>First</i> (' . . ' <i>Second</i>) ?
[5]	First	::= ('ROOT' 'HERE' <i>IdLoc</i>) (' , ' <i>LocTerm</i> +) ?
[6]	Second	::= ('ROOT' 'HERE' <i>IdLoc</i>) (' , ' <i>LocTerm</i> +) ? 'DITTO', ' <i>LocTerm</i> +
[7]	IdLoc	::= 'ID(' Name ')'

5.3.1.1 The ROOT Keyword

If the first or second series of location terms are preceded by `ROOT`, this means that the location source for the first location term of the series is the root element of the containing resource. This is the default behavior, thus the presence of the `ROOT` keyword has no effect on the interpretation of the locator; it exists in the interests of clarity and comprehensibility of the language design.

5.3.1.2 The HERE Keyword

If the first or second series of location terms are preceded by `HERE`, this means that the location source for the first location term of the series is the linking element containing the locator, rather than the default root element. This allows extended pointers to select items such as "the paragraph immediately preceding the one within which this pointer occurs". It is an error to use `HERE` in a locator where a URL is also provided and identifies a resource different from the document which contains the linking element.

5.3.1.3 The DITTO Keyword

If the second series is preceded by `DITTO`, this means that the location source for the first location term is the location source specified by the entire first series, in order to facilitate relative specification of a span.

5.3.1.4 Addressing by ID Attribute

If the first or second series of location terms are preceded by `ID(Name)`, this means that the location source for the first location term is element in the containing resource which has an attribute of type ID with a value matching the given Name.

For example, the location specification

```
ID(a27)
```

chooses the necessarily unique element of the containing resource which has an attribute declared to be of type ID, whose value is a27.

5.3.2 Location Terms - The Keyword Phase

Each location term specifies a two-stage selection process. The first selection phase is identified by keyword:

Location Term

[8]	LocTerm	::=	<i>Keyword Steps</i>			
[9]	Keyword	::=	'CHILD' 'DESCENDANT' 'ANCESTOR' 'PREVIOUS'			
			'NEXT' 'PRECEDING' 'FOLLOWING'			

The keyword selects zero or more elements, relative to the location source:

CHILD

selects child elements of the location source.

DESCENDANT

selects elements appearing within the content of the location source.

ANCESTOR

selects elements in whose content the location source is found.

PREVIOUS

selects preceding sibling elements of the location source.

NEXT

selects following sibling elements of the location source.

PRECEDING

selects elements which appear before the location source.

FOLLOWING

selects elements which appear after the location source.

The locations or elements selected by the keyword are referred to as the candidate locations.

Further discussion of the detailed interpretation of each of these keywords appears in a later section.

5.3.3 Location Terms - The Steps Phase

The second phase of a location term's selection process uses a *Steps* expression:

Steps

[10]	Steps	::=	<i>Step+</i>			
[11]	Step	::=	'(' Instance (',' Element (',' Attr (',' Val)*)? ')'			

Each Step expression is used in order to select elements or locations from the candidate locations, generating a new set of candidate candidates.

Instance

[12] Instance ::= 'ALL' | (('+' | '-')? Digit+)

When the value of instance is the number N, it selects the Nth of the candidate locations. If the special value ALL is given, then *all* the candidate locations are selected. Negative numbers count from the last candidate location to the first; numbers out of range constitute an error.

Candidates can be selected by element type as well as number:

Element

[13] Element ::= '*CDATA' /* selects text pseudo-elements */
 | '*' /* any element type */
 | Name

The Element gives an XML element type; only elements of the type indicated will be selected from among the candidate locations. For example, the location term

```
CHILD(3,DIV1)(4,DIV2)(29,P)
```

selects the 29th paragraph of the fourth sub-division of the third major division of the location source.

The XPointer

```
DESCENDANT(-1,EXAMPLE)
```

selects the last example in the document.

Selection by type is strongly recommended because it makes links more perspicuous and more robust. It is perspicuous because humans typically refer to things by type: as "the second section", "the third paragraph", etc. It is robust because it increases the chance of detecting breakage if (due to document editing) the target originally pointed at no longer exists.

The type may be specified by Name or using the values "*CDATA" or "*". If the type is specified as "*", any element type is matched; this means that the following are synonymous:

```
CHILD(2,*)
CHILD(2)
```

If the type is specified as "*CDATA", the location term selects only untagged sub-portions of an element with mixed content.

The location term

```
CHILD(3,*CDATA)
```

thus chooses the third span of character data directly contained by the current location source. If the location source is a paragraph containing

1. a sentence (A) with no embedded child elements
2. an embedded quotation, tagged with <Q> and </Q>
3. another sentence (B) without embedded elements
4. an embedded note, tagged with <NOTE> and </NOTE>
5. another sentence (C) without embedded elements

6. a second embedded quotation, tagged as the first

then `CHILD(3 , *CDATA)` selects sentence C, while `CHILD(3)` selects sentence B.

Candidates can be selected based on attribute name and value:

Attribute

[14] Attr	::= ' * '	<i>/* any attribute name */</i>
	Name	
[15] Val	::= ' *IMPLIED'	<i>/* no value specified, no default */</i>
	' * '	<i>/* any value */</i>
	Name	<i>/* case and space normalized */</i>
	SkipLit	<i>/* exact match */</i>

The Attr and Val are used to provide attribute names and values to use in selecting among candidates.

If specified within quotation marks, the attribute-value parameter is case-sensitive; otherwise not.

As with generic identifiers, attribute names may be specified as "*" in location terms in the (unlikely) event that an attribute value constitutes a constraint regardless of what attribute name it is a value for.

For example, the location term

```
CHILD( 1 , * , TARGET , * )
```

selects the first child of the location source for which the attribute TARGET has a value.

For example, the location specification

```
CHILD( 1 , * , N , 2 ) ( 1 , * , N , 1 )
```

chooses an element using the N attribute. Beginning at the location source, the first child (whatever element type it is) with an N attribute having the value 2 is chosen; then that element's first child element having the value 1 for the same attribute is chosen.

The location specification

```
CHILD( 1 , FS , RESP , *IMPLIED )
```

selects the first child of the location source which is an FS element for which the RESP attribute has been left unspecified.

5.3.4 Interaction of Keywords and Steps

5.3.4.1 The DESCENDANT Keyword

The location specification

```
ID( a23 ) DESCENDANT( 2 , TERM , LANG , DE )
```

selects the second TERM element with a LANG of DE occurring within the element with an ID of A23. The search for matching elements occurs in the same order as the XML data stream (depth-first, left-to-right).

If an instance number is negative, the search is depth-first right-to-left, in which the right-most, deepest matching element is numbered -1, etc. The location specification

```
DESCENDANT( -1 , NOTE )
```

thus chooses the last NOTE element in the document, that is, the one with the rightmost start-tag.

5.3.4.2 The ANCESTOR Keyword

The ANCESTOR location term selects an element from among the direct ancestors of the location source. The parameters are as for CHILD. However, the ANCESTOR keyword selects elements from the list of containing elements or "ancestors" of the location source, counting upwards >from the parent of the location source (which is ancestor number 1) to the root of the document instance (which is ancestor number -1).

For example, the location term

```
ANCESTOR(1,*,N,1)(1,DIV)
```

first chooses the smallest element properly containing the location source and having attribute N with value 1; and then the smallest DIV element properly containing it.

5.3.4.3 The PREVIOUS Keyword

The PREVIOUS keyword selects an element or character-data string from among those which precede the location source within the same parent element. We speak of the elements and character-data strings contained by the same parent element as siblings; those which precede a given element or string in the document are its elder siblings; those which follow it are its younger siblings.

The instance number in the location value of a PREVIOUS term designates the nth elder sibling of the location source, counting from most recent to less recent.

```
ID(a23)PREVIOUS(1)
```

thus designates the element immediately preceding the element with an ID of a23. Negative instance numbers also designate elder siblings, but counting from the eldest left sibling to the youngest. If the location source has at least one elder sibling, then the location term

```
PREVIOUS(-1)
```

designates the very eldest sibling and is synonymous with

```
ANCESTOR(1)CHILD(1)
```

The value ALL may be used to select the entire range of elder siblings of an element:

```
ID(a23)PREVIOUS(ALL)
```

thus designates the set of elements preceding the element with an ID of a23 and contained by the same parent.

5.3.4.4 The NEXT Keyword

The keyword NEXT behaves like PREVIOUS, but selects from the younger siblings of the location source, not the elder siblings. The location ladder

```
ID(a23)NEXT(1)
```

thus designates the element or string immediately following the element which has an ID of A23. Negative instance numbers designate younger siblings counting from the youngest sibling toward the location source. If the location source has at least one younger sibling, then the location term

```
NEXT(-1)
```

designates its youngest sibling.

5.3.4.5 The PRECEDING Keyword

The PRECEDING keyword selects an element or character-data string from among those which precede the location source, without being limited to the same containing element. The set of elements and strings which may be selected is the set of all elements and strings in the entire document which occur or begin before the location source. (For purposes of the keywords PRECEDING and FOLLOWING, elements are interpreted as occurring where they start.) The PRECEDING keyword thus resembles PREVIOUS but differs in searching a larger set of strings and elements; its result is not guaranteed to be a subset of its location source.

The instance number in the location value of a preceding term designates the nth element or character-data string preceding the location source, counting from most recent to less recent. The location ladder

```
ID(a23)PRECEDING(5)
```

thus designates the fifth element or string before the element with an ID of a23. Negative instance numbers also designate preceding elements or strings, counting from the eldest to the youngest. The location source must have at least as many elder siblings as the absolute value of the instance number; otherwise, the PRECEDING term fails. The value ALL may be used to select the entire portion of the document preceding the beginning of the location source.

5.3.4.6 The FOLLOWING Keyword

The keyword FOLLOWING behaves like PRECEDING, but selects from the portion of the document following the location source, not preceding it.

6. Extended Link Groups

XML describes the syntax of link elements embedded in documents. Many applications, when processing a document, may wish to process not only the links embedded in that document, but links in other documents which point into it. For example, it may be desirable to highlight the resources of such links to make the linkage network's existence apparent. In other words, it may be appropriate to process a group of interlinked documents, rather than a single document.

In these cases, the Extended Link Group element may be used to store a list of links to other documents that together constitute an interlinked document group. Each such document is identified using the HREF attribute of an Extended Link Document element, which is a child element of the GROUP. The value of the HREF attribute is a locator, with the same interpretation as described above.

These elements, just as with EXTENDED, SIMPLE, or LOCATOR elements, are recognized by the use of the XML-LINK attribute with the values GROUP or DOCUMENT.

Here are declarations for the GROUP and DOCUMENT elements:

```
<!ELEMENT GROUP (DOCUMENT*)>
<!ELEMENT DOCUMENT EMPTY>
<!ATTLIST GROUP
    XML-LINK CDATA #FIXED "GROUP"
>
<!ATTLIST DOCUMENT
    XML-LINK CDATA #FIXED "DOCUMENT"
    HREF CDATA #REQUIRED
>
```

7. References

C. M. Sperberg-McQueen and Lou Burnard, editors. *Guidelines for Electronic Text Encoding and Interchange* 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.