INNODATA ISOGEN

White Paper

Don’t Break the Link: Avoid Four Costly Pitfalls in Linking and Reuse

Brandon Jockman, Senior Consultant

Executive Summary

When it comes to technical publishing, today’s enterprises have an unprecedented ability to create and publish content quickly and cost-effectively. The ability to create content once and publish it in multiple formats that can be accessed through different channels is now standard practice. At the same time, creating links to all types of content from a vast range of sources is an expected feature in technical and Web-based publications. The technical advances that support this new world of business publishing also present significant technical challenges.

Organizations striving to realize the full advantages of dynamic publishing tools face an inherent challenge: deploy a system that conforms to standards and embraces best practices, while also avoiding costly pitfalls that could offset those benefits. Publishing groups, for example, would rather focus on the core tasks of creating compelling content that is accurate and accessible to a wide range of audiences than struggle with the nuances of a cumbersome publishing system. Yet, by failing to address such fundamental issues as ensuring link integrity, they may be deploying a system that ultimately turns out to be unstable. To that end, many organizations are beginning to understand the value of link management as a key component of an overall content management and publishing strategy.

Unfortunately, quality information on link management is hard to find and can be either too technical or too vague. Even with the best intentions, publishing groups can be overwhelmed in trying to manage the complex links and relationships in their documents.

This paper will help organizations understand the challenges of link management and offer practical advice for incorporating this critical strategy into their publishing environments. This paper covers:

- The business case for implementing link management
- An overview of link mechanisms and management strategies
- Four pitfalls to avoid
- A real-world case study of the advantages of link management
- Technical references to consult for more information

Introduction: Link Management

Link management plays a vital role in establishing the overall quality of an XML-based system. Too often, organizations underestimate the significance of this key requirement – causing systems to fall short in providing the full suite of link management services required by modern enterprises. The global transition of documents into XML prompts project requirements related to authoring, single-source/multiple-output publication, content management and workflow. The ability to establish links between content units and to reuse content across documents plays a pivotal role in these systems. Link management defines how links interact with the systems around them to ensure that they remain valid and useful throughout the life of a document.

Business Criteria

Companies across the globe are rapidly transitioning business-critical information from proprietary-format silos into non-proprietary, international XML-standard format. To maximize business investment for those information assets, companies are developing complex, integrated systems, which include customized authoring clients integrated with content management systems (CMS). Those content management systems contain workflow processes, annotation and review options and Web-based components that can be integrated with publishing systems.

Throughout these complex systems, it is important to synchronize key architectural points. Such key points include document structure (typically DTDs or W3C Schemas), separation of document content from formatting, and link management. While user-facing features receive a great deal of attention and customization, link management is overlooked too often – sometimes with disastrous results.

How then, can organizations avoid the problems that result from poor link management practices? First, they need to view link management as more than a checkbox on a software product. Inadequate analysis of linking requirements can have a significant impact on an XML-based system's overall success. A lack of understanding of the link-related integration points between authoring, content management, workflow, and publication tools can lead to cost overruns, poor ease of use and even the eventual failure of an otherwise highly-functional system. In short, inattention to link management can negate the clear-cut benefits of using XML in the first place.

This paper will address the basic tenets of link management, and offer guidelines to help organizations avoid four pitfalls that can complicate and introduce costly delays in deploying new content management systems.

Link Mechanisms

Linking can mean one of several things, depending on the context in which the term is applied. Most people understand the basic concept of a link. A link generally includes a pair of objects: a source and a target. However, the implications of linking and link management within complex systems are more difficult to understand.

Linking traditionally refers to a simple point-to-point link, such as a hyperlink from one Web page to another or a navigable link between two pages in a PDF document. The XML Linking Language (XLink) W3C specification[1] provides an XML-based linking syntax for traditional links. However, the combination of its cumbersome authoring syntax and general lack of tool support prevents it from being the most commonly used linking syntax.

Other common approaches to XML linking include ID/IDREF, W3C Schema's key/keyref, or custom links. They have limitations as well. For example, the XML Specification[2] defines a simple ID/IDREF

mechanism. In this mechanism, a link relationship is established when an IDREF attribute of an XML element references another element's ID attribute. The ID/IDREF mechanism has some constraints: both the source and target elements must be within a single XML document, its referenced external entities, or (in some tool-dependent cases) its imported reused content. It does not provide for linking between two separate XML documents (*cross-document linking*). W3C Schema's key/keyref mechanism is similar in nature to ID/IDREF, yet provides more flexible groupings to reduce uniqueness issues via the use of path expressions instead of relying on ID type attribute values.

Some common XML flavors, such as XHTML[3] and DocBook[4] define their own link elements. To support these custom linking syntaxes, some tools provide configurable link definition systems where custom link elements can be defined along with the type of link. However, these non-standard links may not be portable between tools. Unless adequate analysis and preparation takes place, simple linking can be anything but simple in a complex system.

Linking can also include other types of referencing. This type of linking is often called *use-by-reference* or *reuse*. In a reuse link, the point-to-point link relationship has a modifier specifying that content from the target location should be pulled into the source location at a given point in time. The most complete XML-based reuse mechanism is defined by the XML Inclusions (XInclude) W3C specification[5].

In early SGML implementations, a link was a complex relationship of potentially many-to-many items where multiple source items had relationships with many destination targets. Many-to-many links were typically defined outside the documents they linked together. In practice, many-to-many link relationships have proven to occur infrequently and to be difficult to present to users in an intuitive way. Over time, the concept of linking has simplified to the more typical one-to-one relationship, where a location in a source document points to a location in a target document.

Another link mechanism is a reference to an external resource, such as an item in a database, an image or a multimedia file. This is often referred to as a *multimedia* or *asset link*. Multimedia linking issues usually have more to do with asset management concerns than with traditional linking problems.

This paper addresses the first two types of links (traditional and reuse), but not the multimedia or many-to-many style of links.

Fundamentals of Link Management

From an abstract view, most links have a single source and a single target. In a traditional link, the source document contains a markup link element. This link element contains key pieces of information. First, it contains the path to the location of the target document. This may be a file system path, database location or another Universal Resource Identifier (URI)[6]. If the link points to a location within the target document, the second part of the path identifies that location. Additionally, the link element identifies the type of link (or reference) – which is often used to determine how the link will be processed. The link element itself often indicates the type of link (traditional or inclusion).

Link management takes the notion of linking one step further. While a link defines a simple point-to-point relationship, link management breaks down the various aspects of linking as well as how links interact with the systems around them in order to provide control and structure to an otherwise ambiguous area.

One key aspect of link management involves tracking dependencies between different documents. Once a link or reuse relationship has been established between a source document and a target, the integrity of that link must be maintained. If either file is moved, the link address must be updated to remain valid.

Similarly, the target content must not be deleted before removing the link which points to it; otherwise the link in the source document becomes invalid.

Dependency tracking enables *where-used* tracking. By tracking the documents that reuse or link to other content modules, it is possible to display those relationships to end users. An author, editor, or content manager should be able to query a system to see all locations from which a given document or content module is reused or linked. Conversely, he or she should be able to query and see all locations to which a given document reuses or links. This knowledge provides them with key information they need to enable complex link and change management in their systems.

With this basic understanding of the types of links and link management, we can look into some common pitfalls that should be avoided.

Four Key Pitfalls of Link Management

Pitfall #1 – Failing to Maintain Link Paths and Dependencies

Dependency tracking plays a key role in managing links and maintaining their validity. A link relationship includes the source document (which originates the link), the target document, the link path location information, and various other properties. As the source and target XML documents are moved between various computers, systems, repositories, and file locations, the dependencies must be updated and the links must remain valid.

The location paths that identify link or reuse targets are typically contained within the XML file that is the source of the link. These location identifiers typically contain two parts. The first part defines where to locate the target file. When applicable, the second part of the identifier identifies the location of the desired content within the target XML document.

Often the path portion of the location identifier mimics file system paths, such as **C:/folder/target.xml** or **../folder/target.xml**. These identifiers are examples of two ways to specify a target location: the first uses a full file path whereas the second uses a relative path. The internal address portion of the location may be an ID attribute or it may be an XPath[12] or XPointer[10] expression. For Web-based systems, the location identifier may target a Web resource, such as **http://www.company.com/folder/target.xml**. This same XML file may be stored in many different physical locations at different points in time.

Consider the following scenario. Initially, both the source XML file (containing the link information) and the target XML file are on an author's local file system. When the source document is imported into a content management system, its location changes from the local computer to an internal database or other structure on a server. In order for the link to maintain its validity inside the system, the target XML file also must be imported. Upon import, the location information must be updated to be accurate within the repository. While inside the repository, the source or target file may move to a different relative location. Later, the files may be exported to another author's computer for editing. The files may be exported to a server or other location for publication to PDF or HTML. Finally, the files may be exported to a Web-accessible location.

At each point in the scenario above, the link information in the source XML document must be accurate. Not every file system will share the same drive letters or full paths, so systems that only allow you to define full paths do not work. If there is a fixed directory structure for a project, link information defined with a relative path may work on all of the author and publication export locations. However, while the XML document is inside the CMS, the CMS must manage the path information to maintain the integrity of the link. If it does not, then the link information breaks and becomes invalid when the source and target

documents are moved from their original relative locations. Finally, when exporting to a web-accessible location, the link information in the XML document must be rewritten with the correct URL structure.

The key point is that you must maintain link path information in an XML document throughout import/export cycles, while the document remains within the CMS, and when the relative locations between the source and target change. Many out-of-the-box systems break down at some point in this scenario.

Pitfall #2 -- Confusing Entities with Reuse

The XML specification defines the ability to include content from another file by referencing *external entities*. While this is a part of the XML specification, it is not a suitable mechanism for reuse in a complex system. The external entity reference mechanism merely provides the ability to import plain text content, verbatim, into your XML file. An entity file is not an XML file. The contents of an entity are not treated as XML, are not validated separately, and have no other special reuse properties. Many systems make the fundamental mistake of considering this to be a robust XML reuse mechanism – it is not.

Beyond the fundamental mistake of treating external entities as true XML reuse, many tools fail to handle entities properly and consistently. Some XML authoring applications break when they encounter a document containing nested entities, while other applications will not show the content from a referenced entity inline. Furthermore, authoring applications have varying levels of support for defining whether the content from entities should be read-only or editable. Often, content management systems that use entity-based approaches require an arcane catalog mechanism that uses custom, proprietary rules. These content management systems must generate additional files to track relationships and properties related to the files. This is because entity files are not stand-alone XML files; but rather are merely plain text files that may or may not contain XML syntax. Relying on external entities for reuse is fraught with problems that not only cost companies vital time and money but also annoy end users.

True XML-based reuse is provided by the W3C XInclude recommendation[7]. It defines a clear syntax and processing model for valid XML content reuse. Systems that provide full support for XInclude are preferable to systems that rely on entity-based approaches.

Pitfall #3 – Falling Short on Standards Conformance

XML-centric authoring, publication and content management products are built upon open international standards. In theory, standards conformance is a baseline requirement and benefit of current XML-based systems. Surprisingly, many XML systems vary in the level of conformance and in how they implement standards. This is particularly important to linking and reuse standards.

Many XML-based products provide only partial support for required standards. Furthermore, there may be constraints on that support. Several popular authoring tools break when dealing with various entity-related scenarios. Many content management systems still rely on entities instead of true XML reuse mechanisms. Likewise, several XML content management systems still use proprietary syntaxes for defining links and reuse. Even a few publication systems have limitations in the link and reuse syntax they support.

While some constraints on standards implementation are acceptable under certain circumstances, the real problems become evident when you combine authoring tools, content management systems and publication environments. An entity-based reuse mechanism from the CMS may not work with the authoring tool because of inconsistent standards conformance. The CMS standards for exporting link

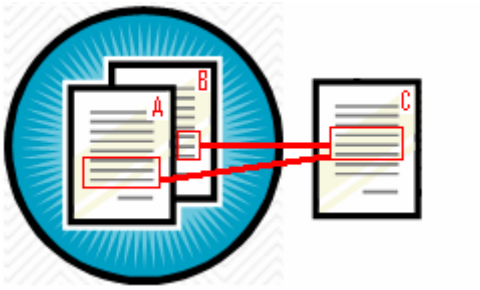
information may not align with the publication engine standards. The link information that comes out of the CMS may not work with any other tool in existence.

This presents a complicated set of problems. Before making a considerable investment in authoring, content management or publication solutions, an organization should engage an experienced analyst to confirm that system-wide requirements share a common approach to standards conformance across a suite of tools.

Pitfall #4 – Overlooking the Complexities of Concurrent Revisions, Versioning, and Change Management

Technical writing departments move at a feverish pace. Many documents may have different versions under concurrent revision in more than one workflow. Companies in this situation want to leverage XML to reuse content chunks across various releases of the documents they are currently authoring. However, there are hidden complexities that, if not discovered through careful analysis, can cause systems to fail to meet their core requirements.

Consider a scenario where both document A and document B link to the same content in document C. All three documents are under concurrent revision. Document B requires a change to the content in document C that is also linked from document A. Document A, however, hasn't been updated so that the change in C makes sense, and may not get this update for several revisions. Which document wins? Should the change to document C wait until document A is updated so that all three documents are still in sync? What if document A needs the content of document C to remain unchanged?



Problems of this nature require linking and change management capabilities that are not present in all content management systems. One potential solution is to branch document C into two documents. That, however, defeats the purpose of reusing a single source and may require an eventual content merger of the two versions of C at a later date. Another potential solution is to lock the link from A to a particular document version of C. What if A requires a change to C that conflicts with the changes required by document B?

Linking and reuse-related change management requirements must be analyzed and compared to the capabilities of content management or workflow systems prior to acquiring them. Otherwise, companies run the risk of purchasing software that won't meet their fundamental needs. The expense of patching or replacing an existing system may be quite costly.

Historical Perspective

Case Study

Challenge… When a major aerospace company won the contract to design and build a multi-purpose jet fighter, it also acquired significant link management challenges. Each model required extensive modifications, which would need to be documented in training manuals that could be easily understood by aviators and repairmen. The solution required linking data on each component so authors could make changes that could be reviewed almost instantaneously in the field

Solution … To streamline this process, the company turned to Innodata Isogen. The content solutions provider had gained extensive experience linking complex data structures and then making that data available electronically via off-the-shelf technology.

A link management tool was chosen to manage the structured documents, which would allow for secure, collaborative XML authoring and sharing among work groups. This tool also complemented the other publishing tools, providing browser-based access — which allowed authors to package content for customized assembly and multi-channel publishing.

While each of the fighter's versions is distinctly different, components are shared as much as possible to hold down costs of producing the jets. Illustration and graphic links were installed so that illustrations could be reused just as related text sections can be recycled. This also reduces the chance of errors in redrawing illustrations from manual to manual.

Results… The system propelled the aerospace company to raise the bar in delivering training and maintenance data as quickly as it is needed. Moreover, electronic distribution has eliminated the vast time and expense involved in printing paper manuals. With its highly-managed and interactive links, the electronic manual can be updated easily and accurately. The company’s new efficiency and speed could literally mean the difference between winning and losing battles in war zones.

Conclusion

Organizations often fail to consider link management as a key requirement in an XML-based publishing system until they are already well into the deployment phase. Underestimating the importance of link management often leads to systems that don't provide the full suite of link management services that today's organizations require.

By focusing on link management during the early stages of a deployment, organizations can take the necessary steps to ensure that links remain valid and useful throughout the life of a document. Successful link management requires a good understanding of link mechanisms and management strategies. Expert analysis of tools, workflow, and standards in an organization all contribute to deploying a successful approach to link management. Without this expert analysis, simple linking can be anything but simple in a complex system.

Inadequate analysis of linking requirements can also have a significant impact on the overall success of an XML-based system. Lack of understanding of the link-related integration points between authoring, content management, workflow and publishing tools can result in cost overruns and poor content usability. Failing to pay attention to basic link management practices can and will negate many of the clear-cut benefits of using XML in a publishing environment.

On the other hand, organizations that incorporate a link management solution as part of an overall content management system will avoid the four pitfalls and spend more time on analysis, planning and deploying the system, which can often spell the difference between success and failure. Many companies would be well advised to consult an experienced link management practitioner to ensure that the integrity of their links will be maintained prior to engaging in critical stages of implementation.

Done right, link management increases the accuracy, effectiveness and usability of publications and technical content. That means organizations will not only save money creating and publishing content, they will also ensure that consumers of that content are more likely to trust its accuracy and find it more useful and accessible.

Innodata Isogen and Link Management

Innodata Isogen consultants have played a critical role throughout the history of standards-based content linking. Their consultants include a co-author of the historical HyTime standard[8], which defined a complete understanding and mechanisms for tracking and managing advanced linking within SGML.

As content solutions switched from SGML to XML Innodata Isogen consultants helped pave the way. Their consultants participated in the XML specification committee, as well as wrote and delivered the W3C XML Indirection Facility (XIndirect)[9], a mechanism for adding indirection to addressing without requiring changes to XLink or XPointer[10]. In many ways, XLink is a descendant of the HyTime specification, simplified for the web-focused XML community.

For more information on the depth of Innodata Isogen's link management solutions, please view the white papers available on the corporate Web site[11].

References

[1] XML Linking Language (XLink) Version 1.0, http://www.w3.org/TR/xlink/, W3C Recommendation, 27 June 2001

[2] Extensible Markup Language (XML) 1.0 (Third Edition), http://www.w3.org/TR/REC-xml/, W3C Recommendation, 04 February 2004

[3] XHTML 1.0 The Extensible HyperText Markup Language (Second Edition), http://www.w3.org/TR/xhtml1, W3C Recommendation, 26 January 2000, revised 1 August 2002

[4] Docbook.org – The Official Homepage for *DocBook: The Definitive Guide*, http://www.docbook.org

[5] XML Inclusions (XInclude) Version 1.0, http://www.w3.org/TR/xinclude/, W3C Recommendation, 20 December 2004

[6] Universal Resource Identifier, http://www.w3.org/Addressing/URL/URI_Overview.html, W3C

[7] XML Inclusions (XInclude) Version 1.0, http://www.w3.org/TR/xinclude/, W3C Recommendation, 20 December 2004

[8] HyTime Second Edition, http://www.ornl.gov/sgml/wg8/docs/n1920/html/n1920.html

[9] XML Indirection Facility, http://www.w3.org/TR/2003/NOTE-XIndirect-20030612/, W3C Note, 12 June 2003

[10] XML Pointer Language (XPointer) Version 1.0, http://www.w3.org/TR/WD-xptr, W3C Working Draft, 8 January 2001

[11] Innodata Isogen – Resource Cooperative, http://www.innodata-isogen.com/resources

[12] XML Path Language (XPath) Version 1.0, http://www.w3.org/TR/xpath, W3C Recommendation, 16 November 1999