



Live Linx Extensible Solutions Ltd. <http://www.livelinx.com>, info@livelinx.com
RMPE House, POB 45168, Har Hotzvim, Jerusalem 91450 Israel, Tel. 972-2-532-8580, Fax. 972-2-532-8320

Automatic Generation of Meaningful XML from Unstructured Content

White Paper
By Joe Gelb
Chief Technology Officer
Live Linx Extensible Solutions
www.livelinx.com

Introduction: The Need for Automatic XML Generation

Many organizations have made the strategic decision to use XML or structured content. They have gone through the process of adopting a specific DTD which defines the exact structure and rules that their XML content must comply to. This DTD represents the document hierarchy and meaning of each aspect of the XML content, broken down into elements and attributes.

Some organizations will convert their legacy content to XML from Microsoft Word, Adobe FrameMaker or some other format, and start maintaining that content using XML editors. Other organizations will choose to continue authoring in Word or unstructured FrameMaker, and need a way to provide that content in valid XML conforming to their DTD in a quick and easy way, either for internal use or for delivery to end users or other applications.

In either case, there is a need for an automated way to convert content into XML – either for a one-time conversion, or as part of a production path. Conversion is often required from a range of source documents written in different templates. An automatic solution must provide a consistent and straightforward way to configure the converter for multiple templates (e.g. Word or FrameMaker) into the new DTD or suite of DTDs, without requiring custom programming.

An automated solution provides the following benefits:

- Substantially reduce conversion time
- Convert content as needed
- Improved conversion quality (no human intervention)
- Seamless integration with content management systems, such as ConteX CMS
- Minimize and limit conversion costs - one price for unlimited XML conversion
- Less headache – no need to manage conversion projects with outside vendors

The Challenge of Automating Conversion to XML

There are many hurdles that need to be overcome to arrive at a truly automatic, consistent and reliable conversion tool. An automatic tool must be able to distinguish between elements of text in the source documents in order to derive meaning and use the proper XML coding.

Elements of text may be simple titles, paragraphs, and lists. They may also be information displayed in the headers and footers of the documents, variables and their values, Word bookmarks or FrameMaker markers, FrameMaker conditional tags and their usage, document properties, change bars, and other subtle information included in source files. Some of that content will need to be coded into XML elements, and some into attributes of specific elements. Hierarchy will need to be created from essentially “flat” source documents.

In addition to coding the text in XML, graphics need to be identified and referenced properly. Cross-references and hyperlinks within source files and between the files in the documentation set need to be resolved. Special characters, such as extended ASCII or Unicode, need to be identified and coded properly into the proper entities defined through the DTD.

An automated conversion process must also provide the necessary flexibility to adapt to the reality that source files may come from different writers or groups within the organization, and may not be 100% consistent across the documentation set. The conversion tool must be able to be fine tuned without completely redefining the tool, or requiring staff members with programming skills.

Above all, the resultant XML must always be valid and conform to the DTD. Writers or other users of the automated tool must not be expected to manually edit the XML files to solve subtle validation errors if the converted XML fails to validate against the DTD.

The XML Generator solution developed by Live Linx recognizes and answers these challenges as essential requirements for an XML conversion engine. The following sections describe these challenges in more detail and how the XML Generator solution meets these challenges.

Change bar →

Paragraph styles are general, and do not distinguish between the meaning of different content.

These subsections all need to be represented as different XML elements

Information stored in the footer needs to be coded as attributes for some text elements shown on the page →

AIR CONDITIONING - INSPECTION/CHECK

1. ECS Ducts and Components Leak Check
 - A. References
 - (1) Engine Start, Aircraft Flight Manual
 - (2) Engine Shut down, Aircraft Flight Manual
 - B. Access
 - (1) Baggage Compartment Door
 - C. Job Set-up
 - (1) Prepare aircraft for safe ground maintenance.
 - (2) Open baggage compartment door.
 - D. Procedure
 - (1) Close airstair door and DV window.
 - (2) Start right engine. See Engine Start, Aircraft Flight Manual.
 - (3) Turn CABIN AIR selector to R ENG position (Figure 601).
 - (4) Select manual mode by pressing MAN PRESS pushbutton.
 - (5) Select minimum rate on manual rate control knob.
 - (6) Position manual control toggle lever CABIN ALT to DEC (decrease position).
 - (7) Set RATE control knob to a position that is comfortable.
 - (8) Pressurize cabin to 8.5 psid.
 - WARNING:** DO NOT POSITION HAND CLOSE TO HOT DUCTS OR COMPONENTS. SEVERE BURNS CAN RESULT IF HOT DUCTS OR COMPONENTS ARE TOUCHED.
 - (9) Check ducts and components for indication of leaks.
 - E. Job Closure
 - (1) Inspect for the presence of foreign objects or debris.
 - (2) Close baggage compartment door.
 - (3) Record all maintenance actions in accordance with current regulations of the local governing authority.

EFFECTIVITY: ALL

21-00-00

PAGE 601
MAR 31/2005

← Cross-reference

Sample source document

<p>Attributes → coded from additional information in the document</p> <p>3 different container elements coded from the same non-unique style usage: - reflist - access - procedure</p>	<pre> - <root> - <icpgblk chapnbr="21" sectnbr="00" subjnbr="00" key=""> <pbtitle chg="r">Air Conditioning - Inspection&sol;Check</pbtitle> - <ictask chapnbr="21" sectnbr="00" subjnbr="00" pgbkknbr="6" key="" chg="r"> ← <title chg="r">ECS Ducts and Components Leak Check</title> - <reflist key=""> <title chg="r">References</title> - <prctopic key=""> - <prcitem key=""> <para key="" chg="r">Engine Start, Aircraft Flight Manual</para> </prcitem> - <prcitem key=""> <para key="" chg="r">Engine Shut down, Aircraft Flight Manual</para> </prcitem> </prctopic> </reflist> - <access key=""> <title chg="r">Access</title> - <prctopic key=""> - <prcitem key=""> <para key="" chg="r">Baggage Compartment Door</para> </prcitem> </prctopic> </access> - <procedure key=""> <title chg="r">Job Set-up</title> - <prctopic key=""> - <prcitem key=""> <para key="" chg="r">Prepare aircraft for safe ground maintenance.</para> </prcitem> - <prcitem key=""> <para key="" chg="r">Open baggage compartment door.</para> </prcitem> </prctopic> </procedure> </pre>	<p>Change information coded by detecting change bars</p> <p>Hierarchical structure coded based on DTD: a <procedure> element has <prctopic> and <prcitem>, which has the content coded in the <para> subelement</p>
--	---	---

Sample XML output

Distinguishing between different types of content

For content in the XML to be properly ordered, classified, indexed, searched and published, XML files generally store different types of information in specific elements and attributes. These elements and attributes are well defined in the DTD. Being that Word and FrameMaker are primarily authoring tools that focus on *formatting*, there is no straightforward way to infer meaning from various paragraphs and other text in the source documents.

Theoretically, text can be formatted with separate paragraph and character styles to represent the meaning of different types of content. However, templates for the source documents generally do *not* have separate style tags to distinguish between these various types of information. Rather, the same styles may represent different types of information. To further complicate matters, depending on the definition of the DTD, information may be populated into particular elements, or attributes of those elements.

The XML Generator solution allows you to identify different types of information based on paragraph and character styles. Or, in some cases, where documents are not rigorously tagged using paragraph styles to differentiate between different types of content, such as titles, body text and lists, the XML Generator allows you to use pattern matching to detect regularities in formatting, and regular expressions to detect textual cues and keywords to distinguish between the different elements of content. This style, formatting and phrase/keyword usage may be mapped to particular XML elements *or* attributes. In addition, the *context* or positioning of the instances of style usage may also be used to further fine tune the mapping to one particular XML code or the other, to make the usage of XML more meaningful and more accurate.

Extracting information from documents

XML files generally contain not only content, but information *about* the content. For example, information about the content creators (e.g. author, publisher), configuration management (e.g. date of publication, which information was revised), the applicability of the content (e.g. to a particular product, model, customer).

In the XML, this information is often coded as attributes to content elements, or in special elements whose purpose is to describe metadata about the content. But often, this information is not present in the body of the source document. Rather, it must be harvested from other areas of the document. The following are some common sources of information in documents that may be used in creating meaningful XML:

- Headers and footers often contain important information which sometimes changes depending on the content found in the body on a particular page or section. The XML Generator can identify information in headers and footers, and relate it to the content found in the body of the pages.
- Documents often have variables in order to use technical parameters or names more consistently across the document set. Or, sometimes one document is meant to be used in more than one context, depending on the values of a set of variables. The XML Generator does not simply convert the variable text, but can maintain the usage of variables as XML entities.
- Authors often assign conditional tags to text in a document to allow for conditional publishing. In XML, this information can be retained as attributes on the content elements for later filtering. The XML Generator can identify usage of conditional text and code that usage as XML attributes, according to the definition of the DTD.
- General information about the document is often entered by the author or automatically populated by a document management system in the framework of document properties. The XML Generator can pick up document properties and map them to XML elements and attributes according to the DTD.
- Bookmarks in Word and markers in FrameMaker can be used in source documents to code information about the text inside the body of the document. They can represent an anchor or target for a hyperlink or cross-reference, or may signify other information such as context sensitivity targets for online help, mapping content to the applications they describe. The XML Generator can use bookmark names and ranges in Word documents, or marker types, content and positions in FrameMaker documents to code information into the resultant XML.
- Authors use automatic change tracking or other change marking techniques to signify when content has changed in the current version. Often, an XML element or attribute is used for the same purpose. The XML Generator identifies change tracking information in source documents and marks the changes in the XML using the proper coding.

Creating hierarchy from flat documents

Nested XML elements create hierarchy and categorization of information in XML documents. For example, a document body may contain many sections, each of which may contain sub-sections, each of which may contain its own sub-structure of content depending on the subject matter that the content is addressing. This structure, or hierarchy, is created using XML elements, which may themselves have no textual content, but exist only to serve as containers for other information.

The difficulty arises in that the body of source documents is basically a “flat” list of paragraphs. Although indentation and other formatting techniques may be used in the source documents to simulate some type

of hierarchy or nesting, there is no real hierarchy coded into the documents. Even where style usage can be used to infer the beginning of a new section, this usually is not nearly enough information to build the necessary degree of hierarchy in the proper context to meet the requirements of the DTD.

The XML Generator engine uses a unique mathematical model and algorithms developed in universities to create hierarchy based on the hierarchical definition of the DTD. The usage, order and context of lower level elements, which correspond to textual elements in the source documents, are expanded into a structure that is created based on the DTD. Multiple levels of nested hierarchy elements are initiated and closed as needed, following the rules of the DTD. Resultant XML will therefore necessarily always conform to that DTD.

Ensuring valid and conforming XML

XML DTDs determine exactly which elements and attributes are allowed, in which order, and in which number. Many times, the sequence of paragraphs in the source documents do not exactly follow the sequence called for in the DTD. There may be too few or too many paragraphs of a certain type than what the DTD deems as permissible.

Converting documents to XML following a simple one-to-one mapping of styled paragraphs to XML elements, as many conversion tools do, put the writers or other operators in the dubious position of having to decipher XML validation errors and attempt to add, remove or relocate XML elements in order to pacify the parser. This is a very frustrating, time consuming and error prone process.

The XML Generator's mathematical algorithms always result in valid and conforming XML. Sometimes elements or attributes will be created or ordered to comply with the DTD rule set. The result is that the XML is guaranteed to conform to the DTD. In some cases, authors may want to change the XML output using their XML editor in order to optimize the XML. But this becomes a matter of improving the output rather than necessity. The starting point will always be legal, conforming XML.

Resolving cross-references and hyperlinks

Source documents that are to be converted very often have cross-references or hyperlinks that need to be maintained and properly resolved. These cross-references or hyperlinks may be internal – where the target of the reference or link is in the same file. However, many times they are external – where the target is in a different file (e.g. in another chapter), or maybe in an entirely different book.

Many conversion utilities are only able to resolve external cross-references and hyperlinks when the entire document set is converted in a single batch, or conversion project. This way, the utility can keep track of all the target names (i.e. unique ID numbers) during the conversion process.

The obvious problem with this method is that it may not be possible to convert the entire document set in one batch. Either the document set is too large to do at once, and must be converted in pieces, over time and possibly using several different computers. Or, some documents in the document set may not yet be ready for conversion.

The XML Generator solution overcomes this problem by employing a database. As documents are converted, this database records all possible targets for cross-references and hyperlinks found in the document, such as bookmarks for Word files and markers for FrameMaker files, which are then mapped to the unique ID that was generated for that text element. When links are found in the document during conversion that need to be resolved, this database is accessed to provide the correct target XML filename

and element ID. Additionally, the actual link instance is recorded in the database, which may become important later for link tracking.

When a link is found to a target document that has not yet been converted, and therefore does not exist in the database, one of two options are employed. In some cases, the XML Generator can use an acceptable naming convention to already create an ID for the target element, even though the target has not yet been converted. The target element ID and link are recorded in the database as pending. Later, when the target document is actually converted, the pre-created target element ID is used, and the link is now fully resolved. In the case where the target bookmark or marker does not exist, a warning is generated.

In other cases, where the XML Generator is unable to determine a target element ID, then special coding is added to the XML file that notes that there is a link that is pending, and a note is generated for the user. The XML file is still valid and conforms to the DTD, however the cross-reference is disabled. Later, when the target file is successfully converted, then the user may run a special utility on the XML file with the link. This utility now resolves the link using the now available target information, and the cross-reference is now enabled.

Coding special characters

Documents written in Word or FrameMaker often include special characters and symbols – extended ASCII or Unicode. These special characters, for example, diacritics, the degree symbol (°), fraction symbols, scientific symbols, usually need to be encoded in the XML based on entity modules that are referenced by the DTD. That is, each character has a special XML entity mapped to it. For example, the degree symbol may be mapped to the entity °.

Although DTDs often use standard libraries of entity modules, this is not always the case. To make things more difficult, the source Word or FrameMaker files may code these symbols differently, depending on the character encoding, font, language, etc.

The XML Generator uses mapping tables which may be customized based on a specific DTD and the nature of the source documents. When special characters are identified during the conversion, the mapping table is used to lookup the corresponding entity code.

Flexibility for change

Many large documentation sets are made up of different types of documents. For example, an organization may have end user guides, maintenance procedures, installation procedures, system descriptions, data sheets, parts catalogs, troubleshooting guides, research and engineering documentation, sales and marketing literature, etc. These disparate types of documents may have been authored using different templates. Or, similar documents may be created by different writing groups who do not always use the same templates or style guides. In addition, documentation with different functions are often represented using different DTDs. The result is the possibility of a many-to-many relationship between source document templates and XML DTDs.

To expand this problem further, the reality of legacy material is that it is often written over several years, by different authors, using templates or style guides that have evolved over time. The result is a documentation set that is not uniform. There is a need to fine tune the conversion tool as subtle nuances in the source material are identified.

Most conversion utilities use programming scripts that effectively hard-code the conversion based on some initial analysis of the source documents. The result is that a programmer is always needed to code changes into the utility to deal with any slight aberration from the expected input.

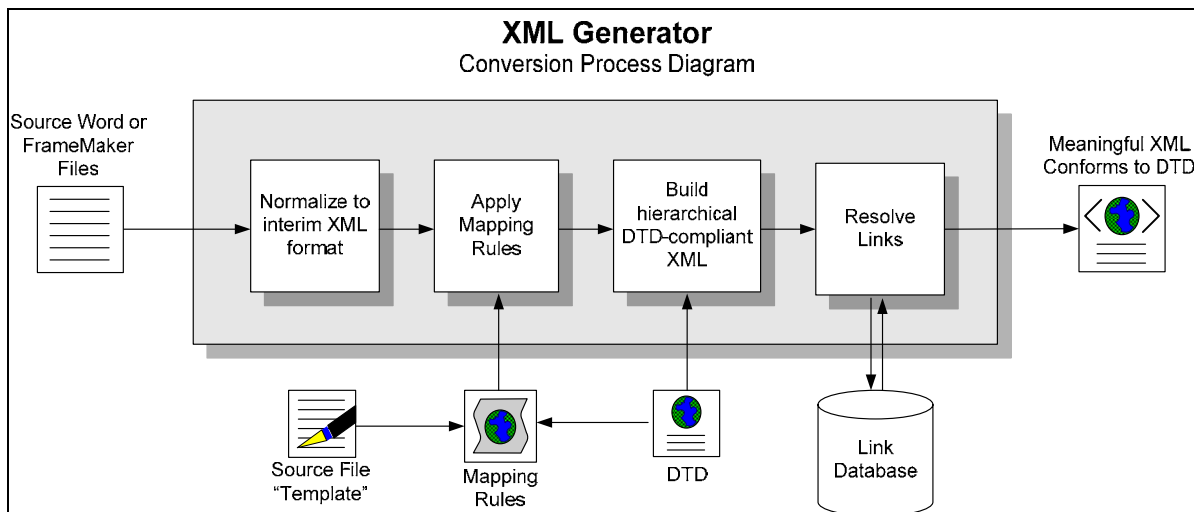
The XML Generator provides flexibility on many levels. The XML Generator is a generic tool that may be configured for one or more source templates and one or more DTDs. Therefore, you may create conversion mappings that allow for the many-to-many relationships between input and output described above.

Because the XML Generator is rule-driven rather than based on custom scripts, additional mapping rules may be created as nuances in source documents are found. Programming expertise is not needed to configure the mapping rules, which means maximum flexibility and independence to meet the challenges of a large and diverse documentation set.

XML Generator Conversion Process

The XML Generator creates meaningful XML conforming to a specific DTD using the following process:

1. Convert the Word or FrameMaker file to a normalized interim XML format that encapsulates the content, style and formatting information, relative positioning of all content elements, and other properties and information found in the source document.
2. Apply mapping rules that map the source template or common style and formatting usage to elements and attributes in the DTD. These mapping rules can include format pattern matching, employ regular expressions to detect patterns in the content with cues to determine meaning, detect keyword usage, and other tools.
3. Build the hierarchical XML based on the rules of the DTD.
4. Resolve cross-references and hyperlinks.
5. Optionally, you may develop and run pre-processing scripts to help normalize aspects of the content before the conversion, and/or post-processing scripts to help make global changes to the XML content during the conversion stage.



Implementing the XML Generator

The XML Generator is an engine that may be implemented in a variety of configurations.

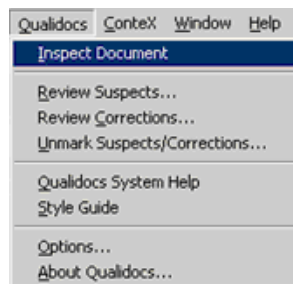
- Client Wizard interface – The XML Generator may be implemented as a client application with a Wizard interface. The Wizard guides users to choose Word or FrameMaker files to convert and to choose an output folder. Files are converted to XML and a log file is produced.
- Service – The XML Generator may be implemented on a shared computer as a service with a queuing system. Source Word or FrameMaker files are submitted to the queue for conversion by users or another application. The service picks conversion jobs from the queue in sequence and converts the source documents to XML.
- Integration with a CMS – The XML Generator may be integrated with a CMS system, such as Live Linx ConteX, to produce XML files on demand via the CMS interface. Documents may be converted either as a one-time conversion or as part of a publishing workflow.
- OEM – The XML Generator engine may be implemented with third party applications as part of an OEM framework.

Checking Documents Before Conversion

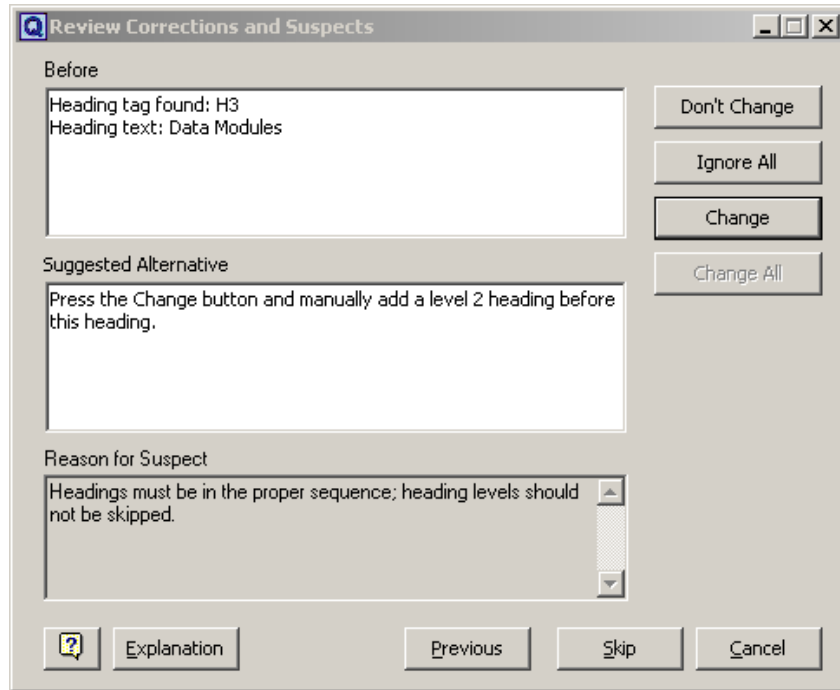
A successful conversion to meaningful XML depends on the quality of the source documents. Often, common formatting inconsistencies or mistakes can create resultant XML that is not optimal. Qualidocs is a plug-in to FrameMaker and Word that writers can use to check their documents for potential problems before the conversion.

Qualidocs is a rule-driven utility configured to verify that documents conform to your predefined template(s), conventions and style guide. Because the verification is done on the source files using Word or FrameMaker *before* the conversion, using Qualidocs can significantly improve XML production quality.

Authors use the Qualidocs menu inside Word or FrameMaker to start the inspection. When the inspection is complete, a user-friendly dialog box (similar to spell check) guides the writer through suspected problems, explains what the problem is and how to fix it.



The Qualidocs Menu



Reviewing Suspects

Common problems that Qualidocs can identify include:

- Paragraph and character style usage that is not consistent with the template
- Manual formatting overrides
- Headings that are not hierarchical; for example, a heading level 1 followed by a heading level 3 without a level 2
- Empty headings or long paragraphs styled mistakenly as headings
- Usage of markers, variables, conditional text, table formats and cross-reference formats that are not recognized by the style guide
- Tables and figures that do not have a caption when a caption is required
- Common text usage mistakes that the spell checker will not catch.

Qualidocs can be configured with new rules that are specific to your organization, template and DTD requirements.

Conclusion

Although the challenges for automated conversion to meaningful XML are substantial, they no longer need be prohibitive. While conversion to XML is never a simple process, the technology does exist in the Live Linx XML Generator that combines extensive experience in document creation, document processing, automated conversion utilities and unique mathematical algorithms into a practical automated XML conversion solution.

About Live Linx

For more information on creating meaningful XML and implementing XML Generator and Qualidocs at your organization, please email us at info@livelinx.com or visit our site at <http://www.livelinx.com>.