
FrameMaker and XML: What Can I Do Now?

by Alan Houser

About the Speaker: Alan Houser is principal partner in Group Wellesley, a Pittsburgh PA-based company that provides documentation and content management consulting services to technology-based businesses. Alan holds degrees in electrical engineering and professional writing from Carnegie Mellon University, Pittsburgh, PA. He has more than 10 years of technical writing experience, including six years of experience with FrameMaker+SGML. Alan has presented seminars and corporate training courses in SGML- and XML-related topics. He has designed both SGML- and XML-based publishing solutions to support both single-source publishing for multiple audiences and personalized content for specific reader profiles. You can reach Alan at arh@groupwellesley.com.

Introduction

FrameMaker version 5.5.6 introduced the capability of saving documents as XML. This paper presents an overview of the capabilities for generating XML from FrameMaker and FrameMaker+SGML, including strengths and limitations of these capabilities. We will discuss differences in XML export between FrameMaker and FrameMaker+SGML. We will also discuss the situations in which you may wish to save FrameMaker or FrameMaker+SGML documents as XML, as well as template design techniques to maximize the usefulness of the XML created by FrameMaker and FrameMaker+SGML.

Why XML?

You have recently noticed more requests for re-using the content of your documentation. Your customer service department wants to create a database of troubleshooting procedures. Your sales staff wants to provide online documentation to each customer that exactly reflects the products and options that the customer ordered. Not only does your field support staff want hardware repair procedures on their micro-notebook computer screens, they want to easily list all required parts for each procedure. They also mention, “By the way, it would be really great if we could order a replacement part by clicking on a part number.”

Each of these requests can be handled by building an application around your content — an application that parses, filters, manipulates, and selects the appropriate information for the appropriate purpose. Not only might this application present information, it might provide other functionality that can be triggered through your content — like the ability to place an order by clicking on the part number that appears in a procedure.

Of course, you could create each of these applications manually, or perhaps through conventional single-sourcing methods. For example, you could manually (or perhaps through conditional text) create a document set that consists only of your procedures. You could manually create customized documentation sets to reflect the options that each customer has purchased. You could even manually key information into a database.

You might have created subsets of your documentation in the past. You have probably re-used components of your documentation for other purposes. For example, many online documents heavily leverage the content of their printed counterparts. However, creating customized documentation components and re-using content for other purposes is tedious and time-consuming. Manual processes take significant amounts of time, and tend to be error-prone. Re-publishing or re-use of information is typically only done when the value of such re-use is high enough to justify the requisite effort.

Imagine if your documents were more like databases. You can automatically identify each part number. You can easily publish only procedures. You can create a customized online documentation set for any combination of your company's products. XML is the technology that makes these things possible.

A Quick Overview of XML

XML is not itself a language, but a syntax for specifying a language. An XML document typically resembles an HTML document, with “tags” that begin with a less-than sign and end with a greater-than sign, such as: <p>. One major difference between XML and HTML, however, is that HTML specifies a set of legal tag names —
, <p>, <table>, etc., while XML does not. In XML, you use tag names of your choice (or your department's choice, or your company's choice, or your industry's choice), to represent the meaning of your content. These tags are included in your XML source files. For example, a part number in an XML document might look like this:

```
<partNumber>5005-2116</partNumber>.
```

A procedure title might look like this:

```
<procedureTitle difficulty="high">Replacing an  
engine</procedureTitle>.
```

Not only does your XML document contain text; it contains tags that provide information about the text. These tags provide information about the information in your document. This information is known as meta-data, or literally data about data. What can you do with the meta-data that's embedded in your XML documents? You can certainly use it to format your information, either in print, for a Web browser, for a personal digital assistant (PDA), or even for devices that you are not yet aware of or do not yet exist. Just as importantly, however, you can use it to select and manipulate the information you need to provide in any particular application. Your technical support staff may only need procedures. Part numbers (perhaps with links to price and availability information) may be particularly important to your field technicians. The possibilities are limited only by your organization's imagination.

FrameMaker and XML

Adobe does not yet sell a product that can edit existing XML documents. However, you can create XML documents with FrameMaker and FrameMaker+SGML. The XML documents that you create can be the basis for document-based applications such as those described above. This capability allows you to maintain your current, familiar workflow — so that you can continue to author and publish using FrameMaker or FrameMaker+SGML. While maintaining this workflow, you can also generate XML versions of your content that can be published to XML-capable Web browsers or used to create document-based applications.

Considerations For your XML to be useful, you must define (or use, if defined by somebody else) a set of XML tags (an XML vocabulary) that is rich and descriptive enough to support the applications that will use your documentation. For example, it would be very difficult to create an application that allows your field staff to order parts by clicking on a part number, if you don't have a unique XML tag to denote a part number, and use that tag consistently throughout your documentation set. It is impossible to publish only procedures if you can't identify your procedure content from the XML.

Constraints Your XML vocabulary must be descriptive enough to support any applications based on your XML documents. However, the tool you use to generate your XML may place limitations on the XML that you can create. For example, standard FrameMaker can only create XML documents of very limited complexity. Particularly if you are creating XML from standard FrameMaker, you must be sure that the XML you create will support the applications that you have in mind.

Capabilities The quality and characteristics of FrameMaker's XML export varies with both product (FrameMaker or FrameMaker+SGML, versions 5.5.6 or 6.0), and whether the XML is exported directly from FrameMaker or is generated by WebWorks Publisher (which is included with FrameMaker and FrameMaker+SGML version 6.0). The following sections describe the capabilities of each product combination.

Capabilities of Each Product

This section details the capabilities and the constraints that FrameMaker (with and without WebWorks Publisher) and FrameMaker+SGML offer for creating XML.

FrameMaker Overview

FrameMaker offers a “Save as XML” option for creating XML documents. When you save a FrameMaker document as XML, FrameMaker maps paragraph and character formats to XML elements. For example, assume that your authors apply a `procedureTitle` paragraph format to the titles of procedures, and a `partNum` character format to part numbers. If so, the document fragment:

is exported to XML as:

```
<procedureTitle>Installing the Acme Widget (<partNum>5005-2116</partNum>)</procedureTitle>
```

Limitations

FrameMaker's XML export functionality presents a number of limitations. Most of these are not specific to FrameMaker itself, but are constraints present when exporting XML from any authoring tool that does not support a hierarchical document structure.

Limited nesting of XML elements. You probably assume that the previous example (the title of a procedure) occurs within a larger context — perhaps within a procedure body within a chapter within a troubleshooting manual. XML is very good at representing these nested relationships, and these nested relationships tend to be very useful when building XML-based applications. However, because of FrameMaker's “flat” document structure (i.e., you cannot nest or overlap paragraph formats), its exported XML cannot nest elements more than two levels deep.

When building XML applications around FrameMaker, your XML application developer must rely on order and proximity, instead of nesting, to deduce relationships between elements. For example, the <procedureTitle> in the previous example would probably occur next to, instead of nested within, a <procedureBody> in the exported XML.

Lack of support for XML attributes. Another way to represent the previous example is:

```
<procedureTitle>Installing the <part number="5005-2116">Acme  
Widget</part></procedureTitle>
```

In this case, the part number is represented as an XML attribute (number="5005-2116") instead of as an XML element.

XML attributes provide a mechanism for adding information to an XML element. Alas, however, there is no way to represent nor export XML attributes from FrameMaker.

FrameMaker and WebWorks Publisher Standard Edition

At version 6.0, Adobe includes Quadralay WebWorks Publisher Standard Edition with FrameMaker and FrameMaker+SGML. Instead of saving as XML from FrameMaker, you can create XML from WebWorks Publisher. WebWorks Publisher's XML export capabilities offer several advantages over those of FrameMaker's built-in XML export.

Ability to remap formats. WebWorks Publisher allows you to map FrameMaker paragraph and character styles to XML element names. For example, when exporting an XML representation of a procedures guide, perhaps you know in advance that each "H2" paragraph format instance is applied to a procedure title. You could map "H2" to "procTitle" in WebWorks Publisher, perhaps minimizing or eliminating the need to rename formats in your FrameMaker template.

Better browser representation. If you want to display your exported XML in an XML-capable Web browser (such as Microsoft Internet Explorer version 5.0 or later), generating XML from WebWorks Publisher is likely to yield much better results than saving a document as XML from FrameMaker. WebWorks Publisher's XML output is better for Web browser display for several reasons:

- WebWorks Publisher generates a cascading style sheet that more accurately reflects the original FrameMaker document format.
- WebWorks Publisher converts FrameMaker tables to HTML tables.
- WebWorks Publisher converts graphics links to HTML graphics links.
- WebWorks Publisher converts cross-references to HTML cross-references.

While these features only minimally affect the usefulness of the XML output for repurposing and republishing, they do make it possible to deliver your XML content via a Web browser, as long as Microsoft Internet Explorer 5.0 or later is the only browser you need to support.

Cascading Style Sheet or XSL Style Sheet Options. WebWorks Publisher offers the choice of representing the original FrameMaker format of an XML document by one of two mechanisms: A cascading style sheet or an XSL style sheet. A cascading style sheet specifies a format for each XML element, while an XSL style sheet provides a set of rules for transforming XML to HTML within the Web browser.

Using the CSS option, WebWorks Publisher's XML output includes a significant amount of formatting information within the exported XML documents. This is somewhat odd, because a major idea behind XML is to separate content and format. This is, however, not necessarily problematic; an XML programmer can fairly easily create a transformation to strip out the formatting information.

WebWorks claims that the XSL option includes less formatting information in the exported XML documents. (It is not obvious to me that this is true; I still see a significant number of formatting directives when using the XSL option.) Furthermore, the XSL option creates a style sheet based on early working draft of XML stylesheet language, which has changed significantly in the final version. Although Microsoft Internet Explorer version 5.0 and 5.5 support the working draft version, other vendors' XML style sheet (XSLT) processors have been updated to support the final version. If you use the XSL option, be aware that the WebWorks XSL style sheets will probably not work with other XML style sheet processors.

FrameMaker+SGML Capabilities

If you wish to generate XML documents and are currently working with FrameMaker+SGML, you're in luck. FrameMaker+SGML is capable of exporting rich XML documents that can be easily repurposed and used as the basis for XML applications.

Generating XML from SGML Elements. Because FrameMaker+SGML uses SGML elements to XML elements, the XML generated by FrameMaker+SGML will have the following advantages over the XML generated by standard FrameMaker:

- Support for nesting XML elements.
- Support for creating and setting element attributes.

Likewise, because XML is essentially a subset of SGML, your FrameMaker+SGML documents probably already have a hierarchical structure and descriptive element names that are helpful, even necessary, for generating XML documents for repurposing.

Ability to remap names through read/write rules. FrameMaker+SGML allows you to map SGML element names to XML element names through the read/write rules file. This gives you another mechanism for using meaningful element names in your exported XML.

Generating style sheets based on element names. If you wish to export XML for Web display, you should be aware of a limitation of FrameMaker+SGML's mechanism for creating cascading style sheets. FrameMaker+SGML creates style

sheet formats based on paragraph and character formats, not based on element formats. In the likely even that the elements in your FrameMaker+SGML source documents do not use the same names as your paragraph and character formats, the cascading style sheet produced by FrameMaker+SGML will yield very poor results.

There are two work-arounds available:

- Install Adobe's XMLcss plug-in, which will create a cascading style sheet from the format characteristics of your elements. This plug-in is available from the Adobe Web site for FrameMaker+SGML version 5.5.6, and is included with FrameMaker+SGML version 6.0.
- Use WebWorks Publisher to create XML.

[FrameMaker+SGML and WebWorks Publisher Standard Edition](#). You can use WebWorks Publisher to map SGML element names to XML element names. This capability is also available, however, through FrameMaker+SGML's read/write rules.

As with standard FrameMaker, WebWorks Publisher creates XML that displays better in an XML-aware Web browser (Microsoft Internet Explorer version 5.0 or later) than does XML exported directly from FrameMaker+SGML. WebWorks publisher exports tables, graphics, and cross-references such that they render well in a Web browser, and produces a more accurate cascading style sheet than does FrameMaker+SGML when exporting XML directly.

Getting Useful XML from FrameMaker

Your ability to develop useful applications for repurposing and republishing your XML source files will depend on the quality of the XML files that you produce. This section outlines some tips for maximizing the usefulness of the XML documents that you export from FrameMaker or FrameMaker+SGML.

Choose meaningful names in your document templates

The names of the XML elements that you export from FrameMaker are based on the names of your paragraph and character formats. If you want to re-use this information in any meaningful way, it would be helpful to use meaningful paragraph and character format names. For example, the paragraph format "procedureBody" would probably be more useful when writing an XML application than simply "Body." Likewise, define meaningful character format names for strings (like product names or part numbers) that you wish to have available to your XML application.

If you are using FrameMaker+SGML, the names of your XML elements will be based on the names of your SGML elements. You should likewise define and use meaningful element names.

Use your templates consistently

Defining meaningful names for template formats or elements will have no effect if your authors don't use your templates consistently. Let's assume that your XML output is going to be used in an application that lets your tech support staff order

parts by clicking on the part name or number. If your authors do not consistently apply the correct format to part names or part numbers, this system will not work as expected.

Keep your XML content in the main text flow

FrameMaker will only export XML content from the main text flow. If your document uses multiple text flows, your exported XML will only include content from the main flow.

Check graphic insertion method

FrameMaker will only export links to graphics that are placed in anchored frames. If you want graphic links in your exported XML, be sure that your graphics are placed in anchored frames within your FrameMaker documents.

What's Missing

A paper about FrameMaker and XML would not be complete without a discussion about what's missing in FrameMaker and FrameMaker+SGML support for XML. The following are some of the problem areas:

No support for importing XML

FrameMaker nor FrameMaker+SGML will not import XML documents. XML support is by export only; if you wish to modify your XML documents, you must modify your original FrameMaker or FrameMaker+SGML source and re-export.

You can work around this limitation in FrameMaker+SGML by creating SGML instances from your XML documents, and importing the SGML instances. You may also need to create an SGML version of your XML DTD. I do not recommend this route unless you are very familiar with both XML and SGML syntax.

No support for XML DTDs

You can export “well-formed” XML from FrameMaker and FrameMaker+SGML. A well-formed XML document conforms to the syntax rules of the XML specification, but is not proven to conform to a document type definition (DTD). A “valid” XML document is guaranteed to conform to a DTD. If you need to create “valid” XML documents from FrameMaker or FrameMaker+SGML, you will need to use another application to validate your XML documents.

You may need to create an XML DTD, perhaps to support the people who will be working with your XML documents. If so, I suggest using a tool such as XML Authority by Extensibility (<http://www.extensibility.com>), that will create an XML DTD from an example XML document.

Other Limitations

At the time of this writing, several key XML-related specifications are still under development. Adobe's lack of XML support in these areas is easily excusable and justified because of the lack of final specifications. These areas include:

- Ability to save page layout information via an XML file (per the Extensible Stylesheet Language: Formatting Objects — XSL:FO — specification).
- Ability to create and save XML extended links (per the XLink and XPointer specifications).

- Ability to work with XML schemas (per the XML Schema specification).

I hope that Adobe provides, minimally, FrameMaker+SGML support in these areas within a reasonable amount of time (given typical software development life cycles, six months to a year) after these specifications are finalized by the World Wide Web Consortium. You can check the status of these and other specifications at the W3C Web site, <http://www.w3.org>.

Conclusion There has been some amount of criticism in the FrameMaker community over Adobe's perceived lack of support for XML, particularly in FrameMaker and FrameMaker+SGML. However, at the time of this writing, several key XML-related specifications are not yet finalized, including the specifications for linking XML documents and the specification for formatting XML documents for printing. At this time, I believe that it is premature to expect full XML support from full-featured technical publishing packages like FrameMaker and FrameMaker+SGML.

If your organization is starting to republish content and build XML-based applications for republishing and reusing your content, you can create XML documents through Adobe's "save as XML" functionality. This capability allows you to continue to use your familiar authoring environment and workflow, while producing XML content for republishing and reuse.

Although standard FrameMaker is somewhat limited in the XML that it can produce, it may be able to meet the requirements of some XML-based applications. FrameMaker+SGML, however, provides the capability for creating complex XML documents that are suitable for any XML-based application.

