Menu

Data Conversion Lab
About DCL
- Why go to DCL?
- Clients
- Company Background
- Management
- DCL in the News
- Events
- Mission

DCL News
- Current Issue
- Back Issues
- Subscribe

Technology
- Technology Resources
- FAQ's
- Glossary
- Presentations
- DCL Work Tracking

Press Info
Clients' Area
Contact DCL
- Directions
- Request Estimate
- Positions

Books2Bytes

Popular Links
- DCL featured in The Columbia Guide to Digital Publishing
- Slash Document Costs
- Ann Rockley on ROI in CM
- PDF Resources
- XML Conversion Resources
- Roundtrip Document Conversion
- DCL Resources Library

Send this Page to a Friend

Printer-Friendly Format

DCL's FAQ

This FAQ sheet has been developed in parallel with *DCLnews*, DCL's newsletter. Readers of *DCLnews* and visitors to our website have submitted many interesting questions which we will be answering here.

About XML

- What is XML?
- What is XML used for?
- XML examples?
- Is XML ready for prime time?
- What is DCL's experience with XML?
- Does XML require a DTD?
- Table Conversions: What are the difficulties of converting tabular material to an XML/SGML format?

SGML & XML

- XML and SGML; what's the difference?
- XML vs. SGML - which should I start with?
- Why did SGML fail?
- I'm in SGML. Should I migrate to XML?
- I'm just starting out. Do I go to SGML or XML?
- I'm already in SGML; have I lost my investment?

PDF

- How do I choose between PDF or SGML conversions?
- Is all PDF created equal?
- Converting PDF to XML - can it be done easily?

Quark, eBooks, and other topics

- How can I convert my files from Quark to XML?
- Can the look and feel of the print book be retained in eBook format?

Q What is XML?

A XML is a new representation for text and database

- Converting Legacy Data...
- Eliminating redundant data
- Aviation & Aerospace
- PDF Conversion to XML & MS-Word
- PDF Conversion
- Quark to XML
- Getting Content into XML

Fact Sheets

- Content Reuse Assessment
- Document Conversion
- SPL - Pharmaceutical Industry
- Harmonizer™
- Jeppesen Map Revision Service

Technical Papers

- Why STM Publishers Should Use XML...
- Department of Defense and the Power of XML
- Your Data in XML
- SGML to SGML 1
- SGML to SGML 2
- Quark to XML
- Plan Ahead
- Do it Yourself?
- Encyclopedia

Presentations

- Conversion to XML: Documents versus Data (11/2003)

information. Based on the industry standard SGML, XML focuses on the structure and content of information. Text or database elements are "fielded" – assigned tags that identify the kind of information they contain, and how different elements interrelate.

Q What is XML used for?

A XML is used whenever organization is important. Because XML facilitates intelligent searching, and because it enables flexible redefinition of appearance, it's become the new standard for publishing on the Web, replacing HTML. But because it's platform and media independent, it's also become the new standard for other forms of traditional and electronic publication. Effectively, XML is replacing SGML for these applications.

Q Can you give some examples of how XML can help me?

A The following really applies to both XML *and* SGML. Example 1: You do a web search on "disk". Instead of a massive list containing everything from music to orthopedics, with XML you only get relevant hits. Example 2: You need to disseminate both a layman's and a professional's version of an article. Instead of maintaining two separate versions of the article, with XML you could maintain a common text database and the tagging information will determine which paragraphs appear in which version.

Q Is XML ready for prime time? Browsers don't yet fully support it. Are you assuming that most users will upgrade to IE 5?

A XML is relatively new on the scene and while it may not quite be ready for prime time, it is getting there rapidly.

While XML may not yet have the support that older approaches like HTML, PDF and SGML do, there is industry-wide support for it. XML today seems to be at the stage SGML was at a few years back. At that time, companies were scrambling to develop supporting applications and tools. A look at Goldfarb's new XML Handbook reveals that companies like Microsoft, IBM, Arbor Text and Sun Microsystems are all lending their full support to XML.

XML overcomes a number of limitations of both HTML and PDF, although at a higher cost. It also provides most of the benefits of SGML at lower cost and with easier implementation. An upcoming FAQ will address the applicability of these alternatives to various kinds of applications.

Regarding browser support, Internet Explorer 5 already provides some levels of support for XML and we anticipate that all the

DCL Calendar

Pharmaceutical Labeling and Product Identification, Whippany, NJ, June 16-17. DCL's Don Bridges will deliver a presentation on "Structured Product Labeling (SPL) and the Implications of Implementing an XML Solution." More...

Recent News

AUGI 2005, Orlando, FL, May 18-20. DCL's Don Bridges delivered a presentation on "Put Your Data on a Diet: Implementing Content Reuse." More...

DIA 18th Annual Electronic Document Management Conference, Philadelphia, PA, February 15-18. More...

Electronic Regulatory Submissions, San Diego, CA, December 6-7, 2004. DCL's Don Bridges delivered presentations on "Legacy Data Conversion: Where, When, and How" and "Migration to XML: 7½ Secrets to Success." More...

ATA e-Business Forum, Atlanta, GA, October 27-29, 2004. More...

Digital Library Federation Fall Forum 2004, Baltimore, MD, October 25-27, 2004. DCL president Mark Gross participated in a panel on "The future of re-keying, mark-up and services." More...

XyUser Group, Boston, MA, September 19-22, 2004. DCL's Don Bridges delivered a presentation on "Converting Legacy Data into XML" and "Content Reuse—The Joy of Less". More...

Tri-XML Conference, Raleigh, NC, July 22-23. DCL exhibited at the conference and hosted a reception. DCL's Don Bridges gave an overview of our content

- Data Migration Considerations (6/2003)
- Technology for Cost-Containment and Efficiency (4/2003)
- Converting Textbooks to Meet the National XML Standard for Accessibility (3/2003)
- More Presentations

browsers will have support for XML in the near future. Will people be upgrading browsers and other software to keep up with these advances? That seems to be a fair assumptions based on what's happened historically, and the fact that most of the viewing software is available at low or no cost.

Q I've heard that XML does not require a DTD. That makes XML very simple to implement, right?

A Although the XML standard makes supplying a DTD optional, for most serious applications, we believe it is prudent to build a DTD to go along with your application. XML allows you to define your own document elements. Without a DTD, you can verify that a document is well-formed, but not which particular document elements you want to allow, and in what order. Having a DTD allows you to use XML tools (many of which are free and available in the public domain) to validate your documents.

A validating processor can check for much more than just well-formedness. In particular, it can check the document structure against a set of declarations to make sure that the structure contains all the parts (and no extra parts) required by those declarations. This makes it easy to use XML to define a format, and then test documents against that format to make sure they meet all requirements laid out by the declarations.

In DCL's experience, documentation tends to vary greatly in consistency. While it is easy to give people documentation guidelines, it is usually difficult to enforce these guidelines. Supplying a DTD to validate your XML documents allows at least some level of consistency checking. Because most XML documents eventually need to be transformed into an appearance-based viewable format, the use of a DTD to constrain your documents will also assist in ensuring that the final documents are easily and correctly viewable.

Q What is DCL's experience with XML?

A Although XML has only been in existence for a little over two years, all the SGML that DCL has produced over the past ten years has been XML-compliant. This means that we've effectively converted over 50,000,000 pages to XML.

Q Table Conversions: I've often heard people talk about the difficulty of converting tabular material to an XML/SGML format. Why is this such a big deal?

A Well, to begin with, tables tend to be complicated. A proper representation of a table needs to contain the correct contents of each row and cell, vertical and horizontal spanning and alignment,

reuse solution. More...

SCIENTECH Sixteenth Annual Procedure Symposium, Clearwater Beach, Florida, June 21-25, 2004. DCL's Don Bridges delivered a presentation on "Data Reuse". More...

XML/SGML Forum of New York Inaugural Event, New York, NY, May 11, 2004. More...

More...

vertical and horizontal separators, as well as distinctions between header rows and body rows. Potentially, there is a lot that can go wrong. And if you do get any table elements wrong in the conversion process, the amount of effort to clean up an incorrect table can be substantial.

If the tables were originally composed using a word processor/ desktop publishing system with support for table editing, and the table tool was used properly, then the conversion can be fairly straightforward. But in many cases, either a tool didn't exist, authors didn't know how to use it, or they didn't use it properly. The result will be something that looks like a table in print, but will involve a good deal of guessing when it comes to conversion.

>>> What are some of the uglier kinds of tables I should expect to encounter?

Producing tables without a proper table editor is often done in one of three ways:

1. The first, and most common, is to use tabs and various types of tab alignments to simulate the look of a table.
2. The second is what we call "page image" tables, and involves using a combination of spaces and line drawing characters to simulate the look of a table.
3. The third is popular in DTP packages that allow you to lay out frames in different positions on the page, and involves building a table via absolute positioning of text frames next to each other to simulate the appearance of a table.

In all three cases, conversion software is forced to "guess" at what constitutes a column, a row, and some of the other attributes that we mentioned earlier (and even guess what actually *is* a table). All this means we should expect some amount of manual table cleanup after automated tools have been run.

>>> But if my authors used a table editing tool - I should be fine, right?

Well, not exactly. Because, even in those cases, all information needed to properly represent a table via structured markup is not always present. For instance, although table tools typically allow the user to delineate table head from body rows, this is often not done by the table author. We also sometimes see difficult tables to convert where, for instance, what should be a five row, three column table, is done as a three column one row table, with the rows within the table simulated via the use of hard returns at the end of each cell. These kinds of tables are particularly ugly because any single problem that is wrong in the converted table can result in cells, from what were meant to be different rows, coming out on the same line (which really can get ugly). This is another example of tables that requires special tools as well as quality control procedures to ensure a properly converted table.

>>> While I'm on the subject of tables, what table model do I use to contain my XML/SGML marked-up tables?

Over the last few years, there have been many table models built to represent tabular materials. The two models that seem to be

the most popular today are the OASIS table model, which is a standardized version of the CALS table model developed many years ago by the United States Department of Defense to represent tables in military documentation, and the HTML table model, used by Web Browsers to render tabular type materials on web pages.


What's the difference between XML and SGML?


XML and SGML are conceptually the same. The key differences are that for XML, several complex, rarely used features were removed which made it easier to implement, and several features were added to support e-commerce. The features that were removed should not impact most applications, especially those built from scratch in XML.

The key items removed are:

1) Tag minimization is no longer allowed.

2) Aggregate Connectors are no longer allowed in element content models.

3) SGML Inclusions and Exclusions to specific element content models are no longer allowed.

The key items added are:

1) XML introduces a new notation for empty tags, namely the notation.

2) A DTD is not required to produce valid XML, as long as the document is "well formed".

NOTE: A more detailed analysis of the specifics may be found in "SGML to XML Conversion Strategies" by Richard Lander.


XML vs. SGML - which should I start with? I'm starting a new project and considering the use of one of the structured markup standards – XML or SGML. I understand the technical differences between the two, but this does not answer my big question: Which should I use?


This is not necessarily a simple question – XML is often referred to as 'SGML Lite'. XML was developed to enable structured documents to work better than SGML did within the confines of the World Wide Web. So part of the answer is that if you will be simply rendering documents via the Web, then you should probably be looking at using XML. There are many tools that will enable you to work with XML documents and render them directly as web pages. So, for instance, XSLT is a language

transformation language that makes it relatively straightforward to transform an XML document into an HTML document on the fly, and in fact, recent versions of Microsoft and Netscape's browsers support XSLT translation, making document display easy. In addition, because XML has become so popular, many standards are proliferating. For example, XLINK (for linking within documents), and MathML (for rendering mathematical equations) are just two examples of the hundreds of XML based vocabularies that may be useful to use within documents, but can only be used if the documents are XML based.

>>> What about if I'm not simply rendering for the web, but want to use my structured documents for other purposes, like publishing to paper?

Now it gets a little more complicated. The SGML standard included features which were removed from the XML standard which might make your life easier. Certainly, building a DTD in XML is a little harder, as features such as inclusions, exclusions, and certain content model structures have been removed. Entity management is also easier in SGML, so if you stick with XML you'll have to do without these. In addition, if you're building a new DTD for your application, you may not want to reinvent the wheel and start from scratch, but rather base your DTD off of an already extant one. So, for instance, if you want to make use of the DOCBOOK DTD, which is a very popular DTD for software documentation, then both SGML and XML versions exist, so you're okay either way. On the other hand the airline industry (ATA) has build a robust set of DTD's for aircraft documentation, and if you want to base your DTD off of that, you'll need to stick with SGML for the time being.

>>> What about if I've already got a DTD that I need to use?

Well, in that case, your decision should be easier. If it's an XML DTD, then you want to be using XML. If it's an SGML DTD, then you might consider modifying the DTD so that it no longer makes use of any features, which were removed, from XML. However, do not expect this to be easy - rewriting the DTD if it made significant use of these features can be a fairly monumental task.

Note also that there is now an approved W3C Schema standard – Schemas are an alternative to DTD's and are considerably more powerful. If you want to make use of them, then there's another reason to stick with XML.

>>> Forget all this technical mumbo-jumbo – just answer my question: Do I start with XML or not?

Well, as you can see, a simple yes or no answer is not so simple, because there are 'strings attached'. But the current thinking in the industry seems to be is that if you're starting an XML application from scratch, it makes more sense to start with XML, particularly so you can make use of the many vocabularies and the built in XML support in many popular software applications. If you're constrained to using a non-XML compliant DTD, then you'll probably need to stick with SGML for the time being.

Either way, in reality, there isn't that much difference in which decision you make. Both solutions are workable, and it's the

concept of authoring documents in a structured markup language that is key. We believe that in the long run, the effort it takes to get started with SGML or XML will be far outweighed by the many benefits that will be realized. Good Luck!


Why did SGML fail?


We feel that the rumors of SGML's death are greatly exaggerated. For organizations with complex documentation requirements, SGML is still the mark-up structure of choice because it offers sophistication and because all the tools are in place to support it. Many of the clients listed on our "Clients" webpage are currently using SGML.

XML has gotten support, because, among other reasons, it is easier to get started with. You can often get started without designing DTDs or worrying about some of the other upfront requirements of SGML. XML is sometimes described as a starter kit for SGML because it is so much easier to master. However, to get the same benefits that SGML can deliver, a DTD and other trappings will eventually be needed (especially for complex documentation).

For simpler applications, XML allows you to get started without all the SGML overhead.

Michael Gross, DCL's Director of R&D adds ...

It may be misleading to think of XML as SGML with all of the unnecessary bells and whistles pulled out. The non-reliance on a DTD is designed into XML so that documents can be shipped around the web easily. This renders greater reusability in data, allowing that it be repurposed with tremendous ease. More than merely an SGML starter kit, it's really a less complex SGML (which makes it less confusing).

Also, there may be trade offs involved with using XML. The XML content model doesn't allow certain structures, and building a DTD for XML is not substantially different or easier than doing so with SGML.

Steve Brown of POET Software adds ...

I would like to add that because XML is less complex, it's easier to build tools for it. Easier-to-build tools means more tools ... that means more competition, which brings down tool prices, making the tools more ubiquitous.

I would also have to say that the relative difficulty of SGML has little to do with its use per se: one of the big problems with SGML is that there is a dearth of tools, and the ones that are available are expensive and difficult to use. As a result, SGML is even more

expensive and difficult.

One of the promises of XML is that the tools will become cheaper and easier. When this eventually happens, the complexities of XML will be largely shielded from end-users. *This*, not the complexity under the covers, is what will make XML easier and its use more widespread.

 I'm already in SGML. Do I need to start moving over to XML?

 If you are already in SGML with production systems in place, you've already gotten past the hard part and there is no real immediate purpose in not staying with SGML. The only time you would need to contemplate such a move is if you are considering major changes requiring new software that supports XML to the exclusion of SGML.

 I'm just starting out. Do I go to SGML or XML?

 If you're just starting out, XML should probably be your first choice. Most people will not miss the features that were left out in the transition from SGML to XML. The SGML software vendors have already retooled to support XML. If your documents require formal validation of structure, make use of the optional DTD construct.

 If I've already put my data into SGML, Have I lost my investment?

 Your investment in having built a database in SGML is not lost, and there is no real reason to stop existing SGML projects in order to retool in XML. The transition from SGML to XML, while not trivial, is not a complex one. Most things that need to be done can be done automatically through software filters, the DTD can be made XML compliant, and much existing SGML software has already been ported to support XML. The key features that are not supported in XML and will require cleanup are things like: putting quotes around attributes, proper casing of tags, and removing tag minimization. More details may be found in "SGML to XML Conversion Strategies" by Richard Lander. View the article here.

 How do I choose between PDF or SGML conversions?

 If your sole goal is to disseminate information in its existing form and look, PDF will do an excellent job at much lower cost. PDF is an outstanding choice for reference documents that must

retain their original look, and for documents that would normally be printed.

There are drawbacks associated with PDF:

- Slower Downloads
- More Scrolling
- Reduced Flexibility

If the materials being converted have long-term value and the goal is to build a repository of information that can be used and adapted to new applications over time, SGML provides advantages that are otherwise unavailable. This is particularly important if the goal is (1) to maintain a library of materials that can be used to create derivative products and (2) to have maximum flexibility in reselling the information through other organization and through aggregators.

We believe that having SGML capabilities offers the following advantages:

- Readers will be able to dynamically reorganize content to fit computer screens and other devices and thus be able to easily read the articles on-screen.
- The data will be adaptable to future uses on devices that today are not yet available.
- Intellectual assets will be easily licensable to various organizations that might re-distribute various literature (medical, scientific, business, etc.) resulting in increased income from these assets.
- Since most aggregators and licensors can more easily handle SGML data, they often pay higher royalties to SGML-based content providers and can load materials more quickly.

The process of going to SGML will normalize your information so that there will be common structures in the data over the multi-year span of the collected works. As a result, it will become much easier to build derivative products that can combine articles from many different parts of your collection.


Is all PDF created equal?


PDF is a print format intended to electronically reproduce the look of a page. But there are three different types of PDF which, while looking similar, have very different characteristics:

- PDF Normal -- This is the best kind of PDF. You get this when your materials have been produced on a modern word processing or publishing system, with a PDF output capability. It contains the full text of the page with appropriate coding to define fonts, sizes, etc. The downloaded files are relatively small, and it will look as good on the screen as the printed version would. (The downsides are described in our PDF vs. SGML FAQ).

If PDF works for your application, and you have the original WP or publishing files, this is the way to go. However, if you are going from legacy materials and don't have suitable electronic files, producing PDF Normal is complex and relatively costly, usually requiring that you first convert to a word processing or publishing format, and from there produce the PDF files.

- Image Only -- This type of PDF is easiest to produce from legacy sources. It is an image of the page in a PDF wrapper and contains no searchable text. Producing it is easy. All you need to do is scan the materials and put the images through an automated PDF loading process. Image Only PDF could be seen as a replacement for microfilm: It is an archival format which can be retrieved. However, there is no ability for text searching and files tend to be fairly large and therefore harder to store and download. The image quality is dependent on the quality of the source materials and the quality of the scanning operation.

- Image & Hidden Text -- This is a good compromise for many legacy applications. It is an image of the page, but with the text portions of the image converted to text for search purposes. In a search application, when the text is found, the image corresponding to the found text is displayed, and the materials can be read in context. This type of PDF is relatively inexpensive to produce since the pages can be scanned and run through an automated OCR process.

  Usually raw OCR is not suitable because accuracy is unlikely to be high enough (raw OCR accuracy is only about 95-99% for most materials). But for search purposes, it is good enough for the majority of applications. Also, since the image needs to be retained, file sizes are larger than PDF Normal and larger than other text formats. If you can live with these constraints, Image & Hidden Text PDF could be a very good compromise. This approach is frequently suitable for library and legal applications.

  Note: Image & Hidden Text PDF allows text to be selected and copied into the Windows' paste buffer to use in other applications. But care needs to be taken during the conversion process because any OCR errors that haven not been "cleaned up" will be seen if someone pastes text. What's more, searches would fail if the text did not OCR properly -- all of which would reflect poorly on the quality of the product.

For more background on PDF, and a discussion of why we might need both PDF and XML, read Bill Kasdorf's article XML and PDF: Why We Need Both.

I have a bunch of PDF documents that I need transformed into XML. Can this be done easily?

A The answer to this question is an unqualified MAYBE. Unfortunately, people are often led to believe that PDF is a publishing format, and therefore it should be easy to convert PDF documents directly into XML. In reality, PDF is more of a description of what the printed page looks like than a description of a document's original document structure. In fact, PDF is often referred to as 'Electronic Paper,' which is a pretty good name for it. And it is a great way to disseminate documentation - Acrobat is an accurate and inexpensive delivery engine.

XML, on the other hand, is a great way to represent documentation in a structured way - because it allows us to specify what 'things' are, NOT how they look, which has many advantages. But this also means that converting documentation from PDF to XML implies inferring a document structure from the page layout of a document, which can be a fairly daunting task.

>>> So, you're saying that this can't be done?

That's not exactly what we mean. More accurately, we're saying that for typical types of documents, this sort of conversion is partially automatable, but you should expect that some part of it will need to be done manually. Of course, it depends on the complexity of the source documents, and the level of markup that is required in the target documents.

The kinds of things that are usually easy for a human to discern from a piece of paper, such as paragraph start and end, headers and footers, multi column text flows, complex tables, mathematical equations, paragraph headings, and footnotes, can be challenging for an automated tool to guess. A conversion tool has to be carefully tailored to the particular structures that exist in the source PDF documents. And if the documents do not share a consistent appearance, the task becomes even more difficult.

To give you an idea of what we are talking about, lets take a look at one simple type of feature that may cause problems, which we hope will illustrate the types of obstacles that can exist. For ease of readability, many publications are printed in a multiple column layout. What this means, of course, is that the PDF, also contains the multiple columns of the document. But since the PDF is basically a page layout format, it contains information about the letters on the page and where they are to be printed. But there is nothing in the PDF document that specifies that certain text is in column one and certain text is in column two (or even that there are in fact two columns on the page). An XML conversion tool must therefore analyze the geometry of the page and attempt to discern a column layout. But where two columns are quite close together, conversion tools can often get confused and miss a multiple column layout, thereby horribly mangling together the text from two totally unrelated paragraphs. Cleaning this up can be uglier than retyping a whole set of paragraphs from scratch.

Examples of other 'basic' text extraction challenges include:

- Attempting to determine where there are spaces between words (getting this wrong can result in single words split into two, or two separate words merged into one)

- Determining which hyphens in the paper page layout need to stay (mother-in-law is better than motherinlaw, although some people might say neither is preferred, but then again, I better leave my personal life out of this!) and where the hyphen can be removed (you wouldn't want to see sub-jective in your converted XML).

Extraction tools tend to make use of dictionaries to help with problems like hyphenation removal, but this solution will normally make some mistakes. And these types of errors, if not caught in cleanup, result in ugly typos in the finished XML.

>>> What tools can I use to do the conversion?

Well, you have a few different choices. Adobe actually makes available for Acrobat 5 a plugin that adds 'save as XML' functionality to Acrobat. It also requires the configuring of a mapping table technology. In addition, several companies sell Acrobat plugins which will allow you to save your PDF documents as plain text, Word, HTML, or various other formats, all of which are stepping stones to get you part of the way to where you need to get to. Our experience is that you need to experiment with various plugins and see which ones work best with your flavors of documents.

>>> Can you give me an idea what types of documents will convert easily?

As a general rule, the simpler the layout of the source documents, the better the converted XML documents will be. For instance, if you are converting novels, since there is typically not much layout in the source documents, you can expect a whole lot of success (and hence very little cleanup) in converting these to XML. If, on the other hand, you've got complex pages such as scientific journal pages, which are likely to contain multiple columns, lots of complex tables, math, footnotes and bibliographies, you should expect have to do a fair amount of cleanup on the converted XML.

>>> Is there anything happening to make PDF to XML conversion easier in the future?

In Acrobat 5, Adobe introduced a concept called 'tagged PDF', which allows a software tool to add to its PDF Writer some level of structure information to the PDF document, so that it contains more than just page layout information. The latest version of Microsoft Word has some support for this, and other tools have promised support for 'tagged PDF'. Adobe, itself, would like to see this information added to as many software packages as possible, so that PDF documents can be more easily re-flowed to fit the smaller screens of PDA's, mobile phones, and other portable devices. So, I guess the good news is that as the support for a more structured PDF format grows, the task of converting PDF to XML will become easier.

How can I convert my files from Quark to XML?


Many people who've already published books, journals and technical documentation in Quark need to convert these documents to XML. Converting Quark to XML presents several challenges. As Quark's file format is proprietary it does not support rich ASCII formats that accurately represent the entire document. Although Quark has a capability to export to a format designated by them as “Express Tags” there are limitations that preclude exporting more than one story at a time making manual intervention of some kind necessary.

Converting tables from Quark to XML presents the biggest problem. Ideally, it is best to convert all tables in the source document to a table structure such as HTML or CALS. Since Quark does not include a table editor it means that you'll need to simulate tables. Many people simply use tabs and frames to achieve that look. Although it gets the job done for print purposes it's not a great solution and can cause larger problems when conversion to the web is done. In terms of conversion, you may not be able to tell what's actually a table. The translation from printed page to the web may look like text separated by tabs, spaces and forced spans. If you're building software to help automate a large conversion project you will be stuck attempting to guess what is a table with its related structure. As such, you'll need to apply logic to files that were often formatted without any logic at all!

There are plug-ins to Quark (such as Tableworks) that let you build tables within Quark as a true table structure. The advantage is that you'll end up with more structured Quark files. The disadvantage is that these plug-ins still don't allow you to export the table structure, so, it will still be a “guessing” game. Quark has a tool called Avenue that attempts to export from Quark to XML. This is good for simple documents or ones that are particularly well-styled in the input file. Tricky conversion features such as cross-referencing, special characters and tables are still hard to do with general purpose tools and will, more than likely, require a customized conversion.


Can the look and feel of the print book be retained in eBook format?


If it's a simple book such as a novel or even a news story this probably wouldn't be an issue but in a more complex book there are problems converting certain elements that would enable those elements to look like the printed book. Converting to an eBook format is moving to a different genre, like when a movie is made into a play. In a movie, for example, there can be scene changes every 20 seconds including complex special effects. However, imagine converting those special effects to the confines and limitations that a stage poses. Would the movie “Star Wars” look the same on stage as it does in film? The answer is that it's just not possible for a variety of reasons. Changes and accommodations would need to be made to simulate some of the special effects while others wouldn't “translate” properly and couldn't be used. The visuals may be different but the story would

be the same.

In the educational, scholarly, scientific and technical markets there are more complex elements that need to be considered as unique components. Unique situations require customized answers but as a general rule of thumb the books with graphic images, tables, equations, sidebars and bullets are the kind of problems you may encounter when converting to an eBook format.

Converting a textbook with sidebars, bulleted lists with margins that have lots of paragraphs, indentations and images to an eBook format will mean giving up some of the way the book looks visually. In electronic books you have practical limitations. There are fewer pixels on a page, whereas, books can be printed at 1200 pixels per inch. To start with, there are two perspectives:

- Screen limitations
- Space limitations

Publishers of finely crafted text, professional and scientific books are used to seeing the printed, finished product with very fine gradations and very fine lines. In an eBook there just aren't enough pixels to make those fine distinctions. There are fewer pixels on a whole screen than in a square inch on a fine book. For example, a 2 x 2 inch graphic in a book is the equivalent of 12 inches wide. If you were to capture the full resolution it would be larger than the screen.

Also, space limitations need to be considered especially in eBook devices. A lot of print books have high quality artwork and the eBook devices used today have limited space. Most of them don't have a hard drive so everything is stored in the RAM. Even the popular Microsoft eBook Reader has limitations to how an eBook can be viewed.

What are some other issues that may affect the look and feel of the final product?

Other limitations include:

- Size
- Graphics
- Color

For example, if there are 3 columns on the printed page they won't fit on a screen. If it were to be done as 3 columns there would be constant scanning and spanning across so that you wouldn't be able to find things. If the publisher is trying to get this image on the page it needs to be readable. If made too small it's not going to be readable and will take up half the screen. The presumption may be that the publisher doesn't want to take up half the screen with an image that is probably not a critical part of the book.

Color is another component to consider. If someone is building a book for an eBook device they would need to use a limited set of colors. OeB defines under 10 colors that will be supported on all devices. If you want to stay universal in what's being produced you have to stay with a limited range of colors. If you start getting in to subtle colors, as you're used to in print, it won't be supported

on all devices.

As a member of the Open eBook Forum Working Group Standards Committee, Data Conversion Laboratory is an active part of the discussions to develop markup standards that would allow publishers to follow a universal method of conversion for files to be loaded on multiple devices that are compatible. Such standards would help publishers define the conversion of eBook projects so that the format would be readable on any device.

Included in this FAQ are only some of the questions and issues that arise when a publisher tries to maintain the exact replication of a print book. Watch for future issues of DCLnews that will cover other areas of eBook conversions.

Table of Contents

 Send this Page to a Friend

 Printer-Friendly Format

Data Conversion Laboratory, Inc. 61-18 190th St., 2nd Floor, Fresh Meadows, NY 11365 718-357-8700 *convert@dclab.com*