Fortis™

The ultimate decision making tool

Exporting Index Data as XML

Revised June 2003

White Paper

Contents

Introduction

XML– Most likely you've heard of it and are wondering what all the buzz is about. Simply put, XML is a meta-markup language used to identify and describe data. The buzz surrounding this technology stems from XML's potential to reduce development time while simplifying data transfer over the Internet in addition to its possibilities for data sharing, application integration, and content management.

With its many possible uses, XML is on the verge of becoming a standard mechanism for data exchange. As a flexible format for data, XML can be used across many industries in countless scenarios. Cross-platform, extensible and license-free, XML is gaining support throughout many organizations as a way to share data across disparate systems and between organizations.

So what does this mean for Fortis? Fortis has helped thousands of customers streamline their business processes, and to further assist customers efficiently exchange data, Fortis provides the ability to export index data in XML format.

This white paper will introduce you to the basic concepts of XML, cover some of the benefits and potential uses for XML and show you how Fortis supports XML output when exporting data. Readers already familiar with the basics of XML can start with XML and Fortis.

Basics of XML

XML, Extensible Markup Language, is used to identify and describe structured data. Despite some misconceptions, XML is not a programming language, nor is it a communication mechanism or a database. It is a way to represent data, just as character-delimited or fixed-width text files are ways of representing data.

Like Hypertext Markup Language (HTML) documents, XML documents are text files composed of markup and content, and XML uses tags (words and/or numbers within a pair of angle brackets "< >") and attributes (such as age="48") to identify and delimit pieces of data. The markup in XML documents looks like the markup in an HTML document.

However, while XML shares similarities with HTML (both are text-based and consist of tags, attributes, and elements), XML is a meta-markup language. XML allows users to create or "invent" their own tags, attributes and elements to accurately describe the data. The tags are self-descriptive, identifying the nature of the data or content they surround. For example, <publisher> tags might identify the publisher of a book or magazine, <calories> tags could describe a food item's caloric content for a nutrition guide, and <status> tags might show if a loan or claim has been approved or denied.

While HTML is a fixed markup language with a finite set of predefined tags, XML is extensible. XML tags are not predefined so that users can define XML elements as they are needed. XML's flexibility in defining elements lets users create tags to meet their own needs and requirements. A legal office can define tags to describe testimonies, reports, wills, deeds, consent forms, correspondence and other legal documents, while a financial department might use tags to describe purchase orders, accounts payable, accounts received, bank statements, contracts, invoices and other financial materials.

While XML tags identify document structure and the inherent meaning of the content, XML does not describe how the content should be displayed. For example, XML does not specify that text should be presented in a certain typeface or weight or that the item should be bold and displayed in a list.

Tags, Elements and Attributes

Tags, elements and attributes are the basic building blocks of XML. XML uses tags to "mark up" content. A tag consists of words and/or numbers within a pair of angle brackets, such as **<Product>**.

An element consists of a pair of tags with data or other elements between them. A tag is placed at the beginning of the element (the start tag) and another tag is placed at the end (the end tag), indicating the beginning and ending of the element, such as **<Product>Fortis</Product>**.

In the element <Product>Fortis</Product>, **<Product>** is the start tag, and **</Product>** is the end tag. Start tags begin with a < (left angle bracket) and end tags begin with a </ (left angle bracket followed by a forward slash). Everything in between the start and end tags is considered content. In this example, the text string Fortis is the content, which is in the form of character data. The element **<Product>Fortis</Product>** is considered the Product element.

Since elements in XML documents can be nested one within another, elements can be described as parent or child elements. For example:

```
<Document_Management>
 <Company_Name>Westbrook Technologies, Inc.</Company_Name>
 <Product>Fortis</Product>
</Document_Management>
```

The Document_Management element is the parent element, which contains two child elements: Company_Name and Product. The two child elements can be considered siblings. The Document_Management element does not have a parent element. It is considered the root or document element. Every XML document has a root element that contains all other elements.

Elements also contain any attributes that a tag might contain, such as <Product **type="client/server"**>. An attribute is a name-value pair (such as profession="writer") that occurs inside start tags and is used to provide additional information that is not part of the data. The name in an attribute is separated from the value by = (equals sign) and values are enclosed in single or double quotes.

```
<Document_Management>
 <Company_Name>Westbrook Technologies, Inc.</Company_Name>
 <Product type="client/server">Fortis</Product>
 <Product type="LAN-based">File Magic 5 Series</Product>
</Document_Management>
```

The first element in an XML document is preceded by an XML declaration that specifies processing instructions and the version of XML being used:

```
<?xml version="1.0" encoding="UTF-8"?>
<Document_Management>
 <Company_Name>Westbrook Technologies, Inc.</Company_Name>
 <Product type="client/server">Fortis</Product>
 <Product type="LAN-based">File Magic 5 Series</Product>
</Document_Management>
```

The header in the example above indicates the XML document conforms to the XML Specification 1.0 and uses UTF-8 encoding for Unicode and ASCII character sets. UTF-8 preserves ASCII character sets and is the default encoding for XML documents. For additional information on the XML Specification, visit www.w3.org/XML.

Well-formed XML

Although XML allows for flexibility in defining tags and elements, XML does have a standardized syntax established by the World Wide Web Consortium (W3C). The W3C publishes the XML Specification detailing the syntax rules that XML documents should follow. When a XML document adheres to the rules, the document is considered *well-formed*.

XML rules regulate the appearance of tags, where they are placed, what element names are acceptable, and so on. However, XML rules are less lenient than those for HTML. All XML documents must be well-formed for an XML parser (or XML processor) to read them. If an XML parser encounters an error, it stops processing the document and reports an error.

The following provides an overview of some of the XML syntax rules:

- **Unlike HTML, XML tags are case sensitive.**
  Upper- or lowercase, or both, is acceptable. However, consistency within any element is important. The tag `<Product>` is different from `<product>`. An element opened with `<Product>` cannot be closed with `</PRODUCT>`. Start and end tags must be written in the same case.

- **All XML elements must have an end tag.**
  Using a start tag while omitting the matching end tag will cause an XML parser to reject the XML document.

- **All XML documents must have a root element and all elements must be properly nested.**
  The root element begins with the first tag in an XML document, and each XML document must contain a start and an end tag to define the root element. All elements must be properly nested within the root element.

- **All elements within an XML document must follow the naming rules.**
  Element names can contain letters, numbers, and other characters; however, element names cannot contain spaces. Using an underscore to separate words in an element is acceptable. Names cannot start with a number or punctuation mark, nor can names start with the letters xml (in any case or combination of upper- and lowercase).

XML Schemas and Validity

Defining and applying a set of XML tags to data is considered an XML application. An XML application is not an application in the sense of executable code; rather, it is simply the application of XML markup to a document.

XML Schemas define what markups and elements are acceptable in an XML application and the type of content or character data an element can contain. An XML Schema provides a framework for enforcing the structure and allowable content in an XML document. Validity is defined by having an XML Schema and XML documents conforming to that schema.

Whereas all XML documents need to be well-formed for XML parsers to read them, not all XML documents need to be valid.

Benefits of XML

XML offers many benefits to organizations. It has the potential to reduce development time, while facilitating pervasive computing and allowing organizations to repurpose data, making business-to-business dating sharing and application-to-application integration easier.

Organizations often store information in computer systems and databases in various data formats that are incompatible. Exchanging data between systems is therefore complex, costly and time-consuming. Converting the data to XML produces data that can be read by many different types of applications. As a cross-platform, non-proprietary and verifiable format, XML offers data portability and reusability and is increasingly becoming a standardized format for exchanging data and documents.

As a flexible and extensible markup language, XML permits the addition of new tags to an existing XML document without breaking the document's structure. Organizations can create new tags as they are needed while maintaining document integrity. In addition, since XML is platform-independent, upgrading and expanding systems with new servers, applications or operating systems won't affect XML documents. Organizations can therefore adapt XML to meet their needs while making their data available to other businesses and mission-critical applications.

Beyond data exchange, XML has many other applications, including Web services, content management, and media-independent publishing.

XML and Fortis

With the Export Data feature of Fortis, users can export index data and document path(s) for a document or a group of documents into an ASCII-delimited file. Users perform the export function from the Query Results window by selecting the documents from which to export the index data and then selecting Export > Data from the File menu.

Users can select to export index data in XML format, as shown in Figure 1.

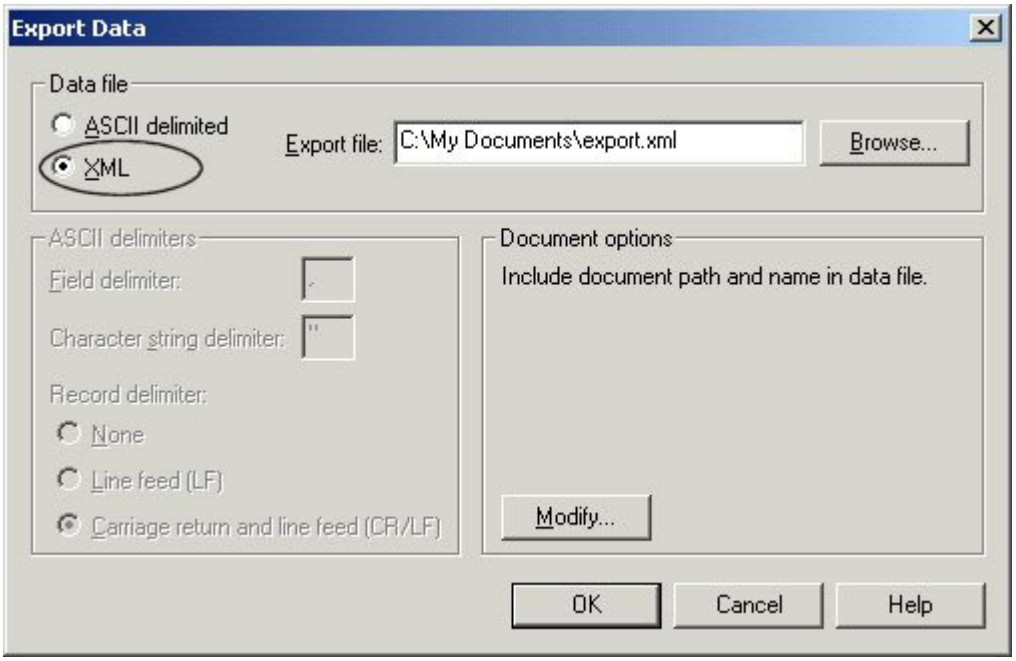

Figure 1: Export Data dialog box with XML option selected

The Export as XML feature allows only index data to be exported in XML format, *not documents*. Users can continue to export documents as Fortis documents, as Adobe PDFs, or in the document's original file format.

XML Format

The Fortis XML Schema is dynamic. The following table lists the XML elements generated during export:

Element	Description/Content
`Data`	root element
`DocTypeName`	parent element for `FieldName`, `Volume-Name` and `File-Name` elements *actual element name is dependent on the Document Type
`FieldName`	child element of `DocTypeName` that contains index field data *actual element name is dependent on the index field

Element	Description/Content
Volume-Name	child element of DocTypeName containing the volume label of the document file (if requested on output)
File-Name	child element of DocTypeName containing the file name or full path to the document file (if requested on output)

The Data element is included in each XML document, as it is the root element. The DocTypeName and FieldName elements are dependent on the Document Type. DocTypeName identifies the name of the Fortis Document Type with tags specific to the Document Type, such as <Invoice> or <Loan_App>. FieldName identifies the index fields associated with the Document Type, with tags such as <Vendor> or <Loan_Type>.

The File-Name element is only produced if either the document name or the document name and path are requested. The Volume-Name element is only produced if the document name or document name and path are requested and the file is not being exported. Volume-Name identifies the volume of the MAG file.

Selecting XML when exporting data produces an XML document that appears as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<Data>
 <MedicalRecord>
  <PatientNumber>123-394</PatientNumber>
  <PatientName>John Smith</PatientName>
  <Volume-Name>DOCS</Volume-Name>
  <File-Name>D:\ARCHIVE\WTI0\00000033.MAG</File-Name>
 </MedicalRecord>
</Data>
```

In this example, exporting data for a document of the Medical Record Document Type produces the MedicalRecord element containing four child elements: PatientNumber and PatientName, which correspond to the index fields, and Volume-Name and File-Name, with the option to include the document path selected and the document file not being exported.

Advantages to Exporting in XML Format

There are several advantages to exporting index data in XML format. Data integrity is maintained because data in the file is logically described for others to interpret. With an ASCII file, users and other applications can only assume data exists where it is supposed to exist. Data formatted with XML can be validated, preventing incorrect updates and misinterpretations of the data. Off-the-shelf XML parsers can parse the data, doing most of the work of validating and extracting the data.

Since XML is platform-independent, users can share data across systems, thus eliminating time-consuming data conversions. For example, an organization can take the XML output from Fortis and import it into a mission-critical application. Data from captured invoices in Fortis can be exported in XML format and then transmitted to an accounting application.

In addition, when exporting index data for multiple Document Types, differentiating between Document Types is much easier. With XML, where the data for one Document Type begins and another ends is easy to distinguish, as shown in the following example:

```
<?xml version="1.0" encoding="UTF-8"?>
<Data>
 <Resume>
  <Applicant_Name>Charles Babbage</Applicant_Name>
  <Telephone>5092278172</Telephone>
  <Dept_Name>R and D</Dept_Name>
  <Status>Active</Status>
  <Salary>145000</Salary>
  <File-Name>C:\export\X.MAG</File-Name>
 </Resume>
 <MSDS>
  <Chemical_Name>Hydroxide Alumaticenate</Chemical_Name>
  <Manufacturer>Hoffman Enterprises, Inc.</Manufacturer>
  <Physical_State>Gas</Physical_State>
  <Carcinogenic>Yes</Carcinogenic>
  <Flammable>Yes</Flammable>
  <CAS_NA_No>29526</CAS_NA_No>
  <File-Name>C:\export\4X.MAG</File-Name>
 </MSDS>
</Data>
```

Exporting in XML produces a human-readable data format that is also easy for XML processors to parse and validate. In addition, XML's simple syntax and support for nesting make it an obvious choice for data storage and transmission.

Summary

XML is a text-based meta-markup language used to describe structured data through the use of tags, elements and attributes. XML is similar to HTML; however, XML lets users define their own tags to explicitly describe content. XML tags can be created as they are needed.

Fortis provides the option to export index data in XML format. The XML elements generated are dependent on the Document Type. Data exported in XML format can be exchanged with other systems or application, enhancing data portability and making integration between systems easier. By creating data that disparate systems can work with, Fortis promotes the exchange of information between applications and between organizations.