

# XML and Content Processes

---

## IMERGE Consulting, Inc

---

Gary Gershon

March / April 2006

Gary.gershon@imergeconsult.com

As Appeared in e-Doc Magazine

www.imergeconsult.com

Extensible Markup Language (XML) has garnered a central role in computer applications as a means to represent business data, transaction requests and responses, and even for scripting processing flow—particularly in the world of Web services. When a business process dictates that XML transactions need to be accompanied by content files, such as a spreadsheet or scanned image, a thoughtful implementation can make a significant difference in transfer speed, resource consumption, and, ultimately, in the scalability and viability of the solution. Over the next thousand words or so, we'll examine these issues, point out some pitfalls, and highlight alternative solutions.

Much of the popularity and power of XML stems from its design as a text-based mechanism to identify the beginning and ending point for data elements and provide human-readable names for these elements, commonly referred to as tags. This is an immense convenience, since we can display and change XML files with standard editors, as well as format and print these files for analysis and diagnosing problems.

If the data we wish to transfer is purely textual, that is, consists of letters, numbers, and symbols, but does not contain special control characters (i.e., non-printing character values such as “line feed”) or binary data (i.e., internal computer representations of numbers), then XML works quite well of and by itself. By utilizing a modern text encoding standard for the XML file, such as UTF-8, we can faithfully and efficiently transmit essentially all the printable glyphs of any known language in the world. Standard XML parsing software will automatically accommodate a handful of important character substitutions to prevent the receiving system from being confused when the text contains any of the XML delimiters. For example, if the text includes the symbol “<” then the sequence “<” will be transparently inserted when a parser creates an XML file and similarly transparently removed when the recipient parses it.

When we need to transfer a file that is not purely textual, or what is usually referred to as a “binary” file, then we need to take a different approach since XML doesn't support sending all binary values. This would be the case with including images, pictures, as well as most office documents since generally these are not yet stored in XML-friendly text formats.

For transporting binary data there are three basic approaches to consider:

1. Encode the binary data into characters (separate from the UTF-8 text encoding discussed above);
2. Use an attachment scheme, similar to how email systems handle mail attachments; or, < file a to URL as such file, attachment each reference>

**Encoding Binary Data** On the Web, binary information is most commonly converted into a text format by using standard software routines for “base 64 encoding.” Each consecutive set of three source bytes of data is replaced by four printable characters. Similar approaches include converting the binary data to “hex”—using two printable characters for each source byte.

While these encodings effectively eliminate the problem of special characters, they do so at an expense. Base-64 encoding increases the amount of data to be transmitted by a third; hex doubles it. The encoding and decoding are processor intensive, and both the source and encoded data are likely to be consuming memory simultaneously.

For small amounts of data these concerns are minor, and indeed, processor cycles and bandwidth continue to be more affordable. However, for large files and production volumes, the impact can be considerable and cumulative—particularly in Service-oriented Architecture (SOA) environments where the XML may be parsed, transformed, and otherwise handled multiple times within a business process transaction.

**XML Attachments** For transferring larger files, particularly for Web services, a better approach is to extract the binary data from the middle of the XML file and append it as one or more attachments. The XML contains reference file names or URIs (a URI is a resource identifier which may not necessarily provide a resource location, as does a URL) to link the tagged data structure to the correct attachment. The following attachment schemes all eliminate the need to encode the binary data as characters and thus don't expand the number of content bytes to be transmitted as do base-64 and hex encodings.

- SwA—SOAP Messages with Attachments. A popular way to attach files to a SOAP XML transmission using the text-marker delimited file packaging approach (referred to as MIME Multipart) that was established for email. SOAP (no longer an acronym) refers to a standards-based approach of using an XML envelope to transmit a request or response body, which is also XML).
  - DIME—Direct Internet Message Encapsulation. The basis for the WS-Attachments specification, DIME provides record lengths for direct offsets of files, rather than text-marker separators. WS-Attachments refers to one of a growing and evolving series of Web service (WS) specifications efforts intended to foster interoperability.
  - MTOM—SOAP Message Transmission Optimization Mechanism. The most recent of the three, MTOM builds on SwA as an approach conformant with the WS-Security specification, for those applications that need digital signatures or encryption now, or might have these needs in the future. We expect DIME to be supplanted over time by MTOM.
3. Many vendor-supplied content management and BPM systems support one or more of these attachment mechanisms, and software support in specific Web service programming environments is a practical consideration. Processing systems need to be designed to separate and hold these attachments for subsequent access when the XML is ultimately processed.

**Document References** In many XML applications a third option exists. The various problems of either encoding or attaching binary files can often be avoided by including in the XML a reference to the content, rather than packaging the content within or attaching it to the XML—typically using a URL. This is, of course, the way Web browsers process HTML Web pages; the style sheet and graphics are retrieved separately using URLs. In a similar fashion many of us send information in emails by including URL links rather than using attachments.

The utility of using content references depends on having the ability to store the content in an appropriate repository that supports URL retrieval (e.g., HTTP, FTP, or perhaps queued messaging ids) references. This approach must also take into consideration that the content may be updated and thus revisiting the transaction at a later time could result in accessing different data. This consideration can be solved effectively by appending a version indicator to the URL.

**Which Approach is Best?** In recent years, we have architected content management and BPM solutions using a mix of these approaches, both for intranet applications as well as those between business partners over the Internet. In an SOA environment, having the ability to reference your repository documents using URLs can be a very useful capability that supports both performance and agility. In some cases, it is nonetheless imperative to package the XML and the content files together in a single transmission. Only by thoroughly understanding the benefits and drawbacks of the alternatives for XML content delivery in your environment will the integrity of your transactions be maintained while avoiding unnecessary performance impacts.

Gary Gershon is a Principal of IMERGE Consulting, Inc. [www.imergeconsult.com](http://www.imergeconsult.com). *[His practice focuses on effectively architecting and building agile SOA business solutions incorporating content and process management. Reach him at 203-431-9328 or \[gary.gershon@imergeconsult.com\]\(mailto:gary.gershon@imergeconsult.com\).](#)*