August 11, 2003

Position on the Binary Interchange of XML Infosets

Oliver Goldman
Jon Ferriaolo
Larry Masinter
Adobe Systems, Inc.

Summary Position

The current XML encoding has strengths and weaknesses. We believe that there may be value in an alternate encoding of the XML infoset that compromises one of the strengths (simple text encoding) in order to address some of the weaknesses. However, too many encodings of XML infosets, or encodings which are not isomorphic to the XML infoset, will lead to poor interoperability and confusion. Thus, we believe that the W3C should endorse at most one alternative encoding of XML infosets.

Adobe uses XML extensively in its products and services for communication of a wide range of material. While XML protocol applications are included in these, we are also concerned with the use of XML for large documents and documents containing binary data, such as images. We believe any alternate encoding of XML infosets must be a viable alternative for all uses and applications, and that this should be part of the charter of any standardization effort.

Strengths and weaknesses aside, the current textual encoding of XML has proved viable. To improve upon it will require careful consideration of many requirements and a variety of designs. Any effort to create an alternate encoding must, in order to be successful, proceed deliberately and with due process.

Background and Experience

Adobe has institutional expertise and experience in the design and implementation of document formats for desktop applications (i.e., FrameMaker, InDesign), for data interchange (i.e., FDF), and for portability (i.e., PDF). We have adopted XML throughout our product line. Table 1, below, describes just some of Adobe's XML-enabled products and technologies.

Product or Technology	**Description**
Adobe Acrobat	PDF creation and collaboration product. Supports web service invocation, export of XML from PDF content, XMP metadata.
Adobe Document Server	Document generation services. Formats XSL-FO as PDF. Provides web service interface.
Adobe Forms Server	Forms automation services. Supports XML form data. Provides web service interface.
Adobe FrameMaker	A WYSIWYG XML authoring and publishing solution.
Adobe GoLive	Web application designer. Includes XML editing support.
Adobe Graphics Server	Graphics creation and manipulation services. Renders SVG as raster or PDF.
Adobe Illustrator	Vector-based editing product. Supports SVG import and export.
Adobe InDesign	Desktop publishing product. Supports import and export XML content.
Adobe PDF	Open, standards-based portable document format. Supports XML form data, XMP metadata.
Adobe SVG Viewer	Free SVG renderer for Windows and Macintosh browsers.
XMP	XML- and RDF-based metadata technology that allows metadata to be embedded within various file formats. Freely available specification and SDK.

Partial list of Adobe's XML-enabled products and technologies

PDF is a standards-based document format which Adobe first created over ten years ago. It supports very large file sizes (one gigabyte and larger) and media-rich documents containing images, video, an other non-text data. PDF documents can contain not only page descriptions, but also structured content, form data, annotations, and metadata, all of which have XML representations. The PDF standard is freely available from Adobe and has been implemented by a variety of products from multiple vendors.

PDF has demonstrated its applicability to a wide range of documents and various related technologies, such as digital signatures. The properties that enable this are further discussed under *XML for Documents*, below.

Adobe's XML-based solutions scale from small to large documents. For example, the XFDF format permits existing Acrobat Forms to be integrated with XML-based workflows and is designed to be used for the efficient submission of data over low-bandwidth links. At the other end of the spectrum, we support the structured editing and formatting of large documents in FrameMaker and the Adobe Document Server.

Adobe does not have experience with any binary XML encodings.

XML for Documents

There are certain desirable properties of a document format which the current encoding of XML does not possess:

- Random access to elements within the document, where "random access" is defined as performance better than linear in the size of the document.
- Compactness, in that the number of bytes required to encode a document should not be excessive with regard to the information present.
- Non-destructive incremental update, that is, the ability to update a portion of the document without modifying more than a fraction of the entire document and while preserving all information in the previous version of the document.

These properties do not align with the original goals of XML and are, to varying extents, at odds with any textual encoding. Nor can these properties be achieved by simply compressing XML documents, e.g., with gzip.

Nonetheless, these properties are critical for certain classes of documents and document-related technologies. In keeping with our position that there should be at most one additional XML encoding and that it should be broadly applicable, we believe a binary XML encoding must support these properties.

Binary Data

It is frequently desirable to embed certain elements within documents, such as images and fonts, which have native binary encodings. In order to include such an elements in an XML document, they must be transformed into an allowed character set, i.e., via base64 encoding. Such transformations reduce efficiency with regard to space (the size of the document) and time (to encode and decode).

It is sometimes suggested that a binary encoding of the XML infoset could "solve" this problem. We believe this is not the case because the infoset itself does not recognize binary data, and any encoding that recognized binary data would no longer be isomorphic with XML.

There are at least three alternatives for handling binary data:

1. By extending the Infoset to accommodate binary data.

2. By defining a binary encoding on the post schema validation infoset (PSVI), which is aware of binary data.
3. By the use of a packaging mechanism, such as multi-part MIME, which permits binary data to be associated with an XML document.

In the first option, both the text encoding of XML defined by and any binary encoding of the Infoset would then be updated to accommodate this addition. The binary encoding would offer the advantage of not requiring a transformation on the binary data.

We believe an encoding defined only on the PSVI, as in the second option, is problematic; see *Relationship to Infoset and Schema*, below.

The use of a packaging mechanism, as in the third option, has advantages beyond the goals of binary data support. However, with respect to binary data, it has the drawback of failing to represent the binary data within either the Infoset or PSVI of the document; this makes it inaccessible to standards such as XQuery.

Documents vs.Web Services

We are aware of some recent proposals which focus on binary encodings designed to enhance the performance of web service implementations. We are concerned that such designs may focus on small data sizes and stream-oriented process to the detriment of document-oriented use cases.

One might suggest that different encodings might be appropriate for web services vs. documents. This contradicts our experience, which indicates that the ability to exchange large documents via web services is itself desirable. This, in addition to our stated concerns with regard to a proliferation of encodings, leads us to prefer the development of a single binary encoding for all use cases.

(We note also that some desirable properties of a web services encoding, such as “chunking” output in order to limit sender buffer sizes, are similar in implementation to what is required for document properties such as non-destructive incremental update.)

Relationship to Infoset and Schema

Any binary encoding of XML must, because XML defines both a syntax and encoding, first separate the two such that an alternate encoding for the “same” syntax can be defined. There are at least two obvious choices for syntax: the XML Information Item Set (infoset), and the post schema validated infoset (PSVI).

Encoding the PSVI instead of the infoset has the advantage that an encoding can be more compact, because information provided by schema validation can be removed entirely from the encoding.

PSVI encoding also has several practical drawbacks:

- It requires the definition of a schema and validation of a document before encoding is possible. In practice, many XML documents are never validated, and many are used without formal grammars ever being defined.
- It is not clear that schemas and namespaces can yet be combined in a practical way. Yet, many XML documents already depend on the use of multiple namespaces.

A binary encoding which operates directly on the XML infoset thus has much wider applicability. We fell any binary encoding must at least permit an encoding based entirely on the infoset and without reference to a schema.