XML Binary Infosets: Position Paper from Tarari

W3C Workshop on Binary Interchange of XML Information Item Sets
September 24-26, 2003, Santa Clara, California

Eric Lemoine, XML Architect (Eric.Lemoine@tarari.com)
Michael Leventhal, Director of XML Products (Michael.Leventhal@tarari.com)

Tarari
10908 Technology Place
San Diego, CA 92127
www.tarari.com
858-385-5131

Tarari makes hardware devices (a PCI card with FPGAs) for accelerating XML processing. The current product, XML-CP (content processor) performs XML tokenization and is designed to be integrated into an XML parsing framework. Our experience in implementing this technology was presented at Extreme Markup recently. This presentation will be mailed separately. This experience, combined with a our unique position as pure hardware company dedicated to accelerating XML, gives us a unique perspective on the issue of XML Binary Infoset which we would like to bring to the Workshop.

The Tarari XML tokenizer produces a list of tokens and pointers which fully represents the original document. At first glance it might seem that Tarari would naturally not consider an XML binary infoset that eliminates or reduces parsing overhead to be in its business interest since our current product is designed precisely to eliminate such overhead. This is not, however, our position. Due to the I/O cost of communication between the software layer and the card we must return an intermediate data structure representing the tokenized equivalent of the entire XML file which is then subsequently interpreted by the software and mapped into the objects required by the higher levels of the XML parser. This structure has a lot in common with some of the proposals we have seen for XML binary infosets which are *not* compressed and also has some similarities with the Millau project and the XMill compressor. Of course, our structure is entirely proprietary. We don't plan to present our tokenization structure as a potential standard binary infoset as it was designed primarily to enable efficient mapping to other representations and not in-place manipulation.

The cost of mapping our tokenized XML structure into event objects or other types of XML objects used in standard XML parsers is very high, especially in languages such as Java which intensively manage objects at the virtual machine level. We have integrated our tokenizer into the Java version of Xerces, for example, and have found that in many cases the acceleration we gain with our card is almost completely washed out by the cost of creating SAX event objects. The card itself is capable of accelerating tokenization by up to 50X compared to software-only approaches. Taking into account bus I/O and other

management and other overhead, this translates into approximately a 5X improvement in XML parsing. However, in our benchmarks, this 5X improvement is reduced to an average of just 30% when we pay the penalty of Java object creation. We can present more detailed benchmark data at the workshop.

Our solution is, effectively, to directly expose our own API into those structures which we can deliver efficiently to the software layer. We recognize, however, that this runs contrary to some of the core principles of XML as well as the wishes of much of the XML user community. It is also limits the potential market for our technology. On the other hand, in our target market (Web Services as well as other high volume, transactional XML processing areas) we see intense demand for XML acceleration, with an important segment of the market ready to accept proprietary solutions in order to meet their performance needs.

So we see a standardized XML binary infoset as the best way, ultimately, to continue to guarantee that XML remains open, interoperable, and non-proprietary. In terms of our own interest as an XML vendor we feel that the XML binary infoset will open up far more opportunities for the possibilities inherent in our technology than either the status quo or the situation where users will be forced to choose from among proprietary technologies in order to gain performance. Naturally, we are also very concerned to ensure that the XML binary infoset is fully compatible with hardware approaches to processing and accelerating XML.

The above text describes well, we hope, what work our organization has done in this area as well as outlining our basic interests and position. The presentation delivered at Extreme Markup (and mailed separately) gives more background on our technology and implementation experience. We would be pleased to make a short presentation to the workshop on our experience or to simply participate as an interested party in the discussions around existing strawman proposals. We respond to the other questions below.

What goals do you believe are most important in this area? (e.g. reducing bandwidth usage; reducing parse time; simplifying APIs or data structures or other goals)
The first goal is to reduce parse time but this is also deeply related to upstream processes. We think the goal is to process XML very efficiently which is related to both the in-memory structures and the API for accessing those structure. The DOM doesn't really need to be simplified, but we need a DOM that doesn't impose an inefficient internal representation. We have not ourselves seen that bandwidth usage is as critical, although our technology could benefit by a significant reduction in bandwidth. Our thought about bandwidth reduction is that if it is emphasized, care should be taken so that the binary format enables a hierarchy of compression schemes enabling applications to chose between ease of parsing vs ultimate size reduction.

What sort of documents have you studied the most? (e.g. gigabyte-long relational database table dumps; 20-MByte telephone exchange repair manuals; 2 Kbytes web service requests)

1k-100k SOAP messages, database extracts, standard XSL and XSD files – with our current technology. We are mainly focused on high-volume, transactional XML with small to medium size files. Mostly, we are not focused on RPC style Web Services requests but on document-style Web Services message exchange. We are interested longer term in larger business documents in transactional environments (e.g., ebXML) which, based on our experience will be in the 500kb range.

What sorts of applications did you have in mind?
We are mainly application agnostic but our early customers are using our technology for document style Web Services and other types of high-volume, transactional XML.

If you implemented something, how did you ensure that internationalization and accessibility were not compromised?
We currently and intend to continue to fully support UTF-8 and UTF-16.

How does your proposal differ from using gzip on raw XML?
We don't currently do compression. We are not as concerned about bandwidth as we are with parsing time and efficient APIs and in-memory structures. We are not convinced that gzip will be beneficial unless un-compression also delivers a parsed version of the XML.

Does your solution work with any XML? How is it affected by choice of Schema language? (e.g. W3C XML Schema, DTD, Relax NG)
Our current tokenization only supports non-validating SAX events and does not use the schema. It is our intention to provide support for at least XML Schema in the future. It does work with any XML. We are interested in processing methods which can be made more efficient with *a priori* knowledge of the XML documents that could be derived from the schema.

How important to you are random access within a document, dynamic update and streaming, and how do you see a binary format as impacting these issues?
Random access within a document is a key feature of any binary format. But not all the features of a XML document have to be necessarily granted a random access. A binary format should offer a greater facility to implement fast document traversal and dynamic update.