What I Learned This Month: XML vs. JSON

Scott Chapman
American Electric Power

XML (eXtensible Markup Language) is generally seen as the "lingua franca" of data interchange today. But XML does have certain issues. In particular XML documents tend to be large relative to the data that they contain. The relatively verbose nature of XML's tag structure means that a few characters of data may be surrounded by an equal or greater number of characters of metadata. Larger data structures have performance and capacity impacts on networks, storage hardware, and systems. You can limit this impact by using entity names that are short, but that may also limit XML's strengths of human readability and self-describing nature. Additionally, parsing data out of XML documents can be (in some cases) complicated for the application programmer. There may also be performance concerns when parsing large or complicated XML documents, particularly for web applications written in JavaScript.

One alternative to XML is JSON (JavaScript Object Notation). This is a particularly useful solution when the application is written in JavaScript. JavaScript seems to have largely supplanted Java as the language of choice for the client side of web-based applications and JavaScript is being used more and more on the server side as well.[1] Java versions 6 and onwards even include support for running JavaScript within the JVM, effectively allowing you to mix and match languages within a single JVM.

XML enjoys broad support across all the major programming languages today. JSON is natively supported in JavaScript but there are libraries to interpret JSON in other popular languages such as Java, Python, and Ruby.

XML and JSON are not directly comparable: XML describes documents and JSON describes objects. So while there's a number of utilities and frameworks available to convert between the two, there's no standard methodology for doing so. If you need to convert between the two, you need to either write your own utility function or find one that works for your particular situation.

I'm in the process of re-writing my real-time system and application performance monitor[2] using Helma as the server-side application environment. Since this means the application is written almost entirely in JavaScript, I questioned whether I should store XML retrieved from RMF (or other sources) as XML or if I should convert it to JSON before parsing, transmitting and storing the data. While in most cases I simply need to retrieve a single value from the XML, in some cases a significant subset of the XML document needs to be stored.

[1] Some of the most popular server-side JavaScript application servers can be found at: Helma.org, RingoJS.org and NodeJS.org

[2] See www.cmg.org/proceedings/2007/7905.pdf

In the end, I compromised. Where I need to simply extract individual values from an XML document, it was easiest to do this while it was still an XML document. But when I'm storing reports for later display, converting them to JSON results in less data to be stored and it will be easier for the JavaScript running in the browser to interpret.

I did some measurements to support this decision. The following table summarizes the results for one test set of 22 RMF DELAYS and PROCU reports from multiple systems.

	XML	JSON	Comp XML	Comp JSON
Total of 22 reports	269,176	80,937	21,294	17,728
Average / report	12,235	3,679	968	806
Compress savings			92%	78%
JSON savings		70%		17%

In this test case, the JSON data was 70% smaller than the XML data. However, when compressed, the difference was much smaller, with JSON having only a 17% size advantage. In this particular example, the absolute difference of about 3.5KB is arguably immaterial from a disk storage perspective. But that data is going to be read potentially over and over again, going through decompression each time. I figure decompressing smaller things is arguably faster than decompressing larger things, to at least some degree. I didn't take the time to try to measure that; I suspect the benefit is relatively small. But the JSON format has later ease-of-use benefits for this particular application programmer (me) as well, so there's really no reason not to use it.

While the application I'm working on is pretty unique, the idea of having an entire application (both server and client sides) written in JavaScript is gaining in popularity. Where the data is going to be consumed by JavaScript code, using JSON for the data format instead of XML likely makes a lot of sense. I am, of course, not the first to notice this: the major web players (such as Google) have been making their data feeds available in JSON or JSON-P.

XML still has its place, of course. For the skilled XML practitioner, tools such as XPath, XQuery, and XSLT allow for the relatively easy traversal, querying, and transformation of XML documents. There are likely some use cases where XSLT itself may be a strong argument for XML instead of JSON. In fact, I considered that very issue before deciding to store the data in JSON. I decided that since my XSLT skills are a bit rusty and I would only consider XSLT useful for a limited set of application functions, I can probably code a JavaScript-based solution faster than I can remember how to use XSLT.

This is a topic where "your mileage *will* vary", but, as always, if you think I got it all wrong or have questions or comments, you can reach me via email at sachapman@aep.com.

Postscript :

For CMG 2012, I co-authored a paper about data visualization with Nicole Arksey. I met her at CMG 2011 and we had a great discussion about the topic and subsequently decided to write a paper for this year's conference. I am pleased to say that our paper was accepted for CMG 2012, so now we're starting to think about our presentation. We want to show some common data visualizations and ideas about how to improve them. While we have our own personal favorite examples, we're sure there are more out there. So if you have a particularly poor or problematic data visualization that you don't mind us making an example of, please send it our way! We look forward to the challenge!

Postscript 2:

Some of you may be wondering, "what does this JSON stuff look like relative to the XML?" Here are a couple of snippets of some of the RMF data I was working with to illustrate the difference between the formats. Just the first couple of rows from an RMF DELAYS report is shown.

XML

```
<row refno="1">
<col>*SYSTEM</col><col/><col/><col/><col>32</col><col>2</col><col>4</co
l><col>20</col><col>74</col><col>4</col><col>0</col><col>0</col><col>0<
/col><col>0</col><col>1</col><col>0</col><col>0</col><col>0</col><col>0
</col><col>0</col><col>0</col><col>0</col><col>4</col><col>0</col><col>
0</col><col/></row>
<row refno="3">
<col>*BATCH</col><col/><col/><col/><col>31</col><col>12</col><col>28</c
ol><col>0</col><col>61</col><col>23</col><col>0</col><col>0</col><col>0
</col><col>0</col><col>5</col><col>0</col><col>0</col><col>0</col><col>
0</col><col>0</col><col>0</col><col>0</col><col>23</col><col>0</col><co
l>0</col><col/>
```

JSON

```
{"row":[{"col":["*SYSTEM","","","",32,2,4,20,74,4,0,0,0,0,1,0,0,0,0,0,0
,0,4,0,0,""],"refno":1},
{"col":["*BATCH","","","",31,12,28,0,61,23,0,0,0,0,5,0,0,0,0,0,0,0,23,0
,0,""],"refno":3},
```

Note that this simple example also illustrates the difference in how data is handled in XML and in JSON.

XML: "We have a document. In that document, we have rows, each of which has a reference number. Within each row is some number of columns with data, which we will list individually."

JSON: "We have an object, called "row". It has an array of objects. Each object in that array has a property "col" and a property "refno". The "col" property is an array of values, and the "refno" property is a particular value."

While in this case, the distinction between those different ways of thinking about the data is not very important, in other cases, the difference is more significant and the methodology for converting XML to JSON may need to be carefully chosen. I simply used the org.json.XML convertor because it worked and I liked the resultant JSON best of the three or four convertors that I looked at.