JSON & FpML: A Discussion Document

FpML Architect Working Group – June 2016

In recent months both ISO and FIX Protocol have announced projects to look into representing financial data using JavaScript Object Notation (JSON) encoding. It has been suggested that FpML too should investigate its potential use with FpML.

What is JSON?

JSON is a language independent format commonly used in Internet communication between servers and browser based applications. Whilst it derives originally from JavaScript many programming languages now have tools to support its generation and parsing.

JSON is a relatively simple text based format consisting of lists of 'name : value' pairs. By allowing values to be arrays or sub-lists a nested data description can be built up. For example:

```
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 25,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    },
    {
      "type": "mobile",
      "number": "123 456-7890"
    }
  ],
  "children": [],
  "spouse": null
}
```

The order of data pairs within the document is not important and all access to values is performed via the 'name' (e.g. the 'name : value' pairs are usually held in hash table). Only arrays of values or sub-lists maintain their order.

Building and accessing JSON structures in program code is typically easy compared using the standard DOM API for XML.[1]

[1] DOM was designed to be a standard API across several programming languages and as such is not optimal for any of them. This has led to the creation of better tailored and language specific APIs such as JDOM for Java.

JSON Schema

Traditionally JSON data is not validated through a standard tool rather it is left to each processing application to test the data content of the document to see if it is sufficient for its purpose. The loose nature of JSON documents means that extraneous information is unlikely to be tested for.

There have been efforts to define a JSON based schema for JSON documents. The latest draft (version 4) expired on August 3, 2013 but no International standard has yet been defined.[2] There are some tools that support version 3 and/or 4 of the IETF specification but standard implementations such as Java's API for JSON (JSR-353) do not support validation against a schema.[3]

JSON Schema support similar features to XML Schema. It is possible to define constraints on values through patterns; repeating values can have limits on the cardinality; properties can be marked as optional or mandatory and there is support for grammatical like constraints using 'allOf', 'oneOf' and 'anyOf' constructs but this must be used in conjunction either with a simple extension like mechanism or in the definition of sub-lists. It does not appear to be possible to mix the 'allOf' and 'one of' constructs in the same way that 'sequence' and 'choice' can be in XML Schema.

A quick review of some sample JSON Schemas available on one online JSON validation website found that most are simple lists of optional and mandatory named values with a little nesting and repetition.[4] None of the examples were large or complex schemas comparable to FpML or ISO 20022 messages.

We could find no tool support for JSON Schema design as there are for XML. Some XML tools have support for editing JSON files and doing simple well-formedness checks, transformation to/from XML and validation but with varying results.[5] Neither could we find any tools to automate the conversion of XML Schema to a JSON equivalent.

Conversion to/from XML

Bi-directional conversion between XML and JSON is possible provided some conventions are adopted in the JSON for handling attributes, repeating elements and element order. For example, the following XML fragment contains all the commonly used features of XML.

```
<ns:element xmlns:ns="urn:somewhere">
   <emptyElement attr="flag"/>
   <repeatingValue>one</repeatingValue>
   <repeatingValue>two</repeatingValue>
   <repeatingValue>three</repeatingValue>
   <repeatingElement>
      <value>one</value>
   </repeatingElement>
   <repeatingElement>
      <value>two</value>
   </repeatingElement>
</ns:element>
```

This could be converted into the following JSON representation if attribute names are prefixed with

[2] See http://json-schema.org/documentation.html
[3] See https://jsonp.java.net/
[4] See http://www.jsonschemavalidator.net/
[5] XML Spy 2015 cannot 'round-trip' XML/JSON transformations if attributes are present, '<c/>' becomes '<a>true<c/>'

a '@' character and the conversion process differentiates between elements containing simple values and those containing sub-elements.[6]

```
{
   "ns:element": {
      "@xmlns:ns": "urn:somewhere",
      "emptyElement": {
         "@attr": "flag"
      },
      "repeatingValue": [ "one", "two", "three" ],
      "repeatingElement": [ {
         "value": "one"
      }, {
         "value": "two"
      } ]
   }
}
```

JSON does not have to maintain the order of the data pairs so when documents are converted back to XML extra processing would be needed to return the generated data to the order expected by its schema.

There are some features of XML that cannot be easily translated, such as comments, processing instructions, or mixed content, but all the business data content of a data-centric structured messages, like FpML documents, can be preserved.

There does not appear to be defined standard for XML/JSON mapping. Some of the tools append a '@' to attribute names to differentiate them from normal elements while others create a special 'attributes' sub-list to hold them separately from the element data.

Other Tools

Much of the utility of XML comes from the family of associated tools that have grown around it for data storage (e.g. XML databases), retrieval (e.g. XPath, XQuery) and transformation (e.g. XSLT).

As JSON's internal representation is similar to the XML DOM some tools can be leveraged to operate on JSON documents (e.g. jQuery) and some NoSQL database will accept it but as most JSON users see it as just a transient representation for data being passed from one computer to another they have not bothered to create tools for more document centric tasks.

JSON vs XML

For the data transfers expressed in FpML, FIX and ISO messages there is little difference in the expressiveness of JSON and XML for actual business data.

JSON messages are perceived as being easier to build and access in program code, having a simple design and none of the complications of XML such as namespaces. The APIs to support it are correspondingly simpler and better integrated with the programming languages than those for XML.

[6] Some translating tools elect to emit attributes in a special sub-list to keep them separate from element content.

JSON documents will always be a little smaller than the equivalent XML as object structures do not have a verbose closing tag and the simpler syntax means that JSON will usually be parsed quicker than XML.

However the wide availability of XML Schema parsers makes validating XML documents to ensure they match the standard expected by an implementation easier than can be achieved in JSON.

JSON lacks a truly standard schema language and support for a diverse set of data types, especially for temporal values. Even if a of the third party JSON schema validator is used to check message content additional coding would be needed in every implementation to bring it up to the same standard as XML Schema validation (e.g. checks on data types beyond those possible with regular expressions, structural relationships between object properties that could be encoded as part of an XML schema grammar, etc.).

When all this extra processing is taken into account the time to parse XML and JSON is probably not significantly different, however the time and cost to maintain a XSD based solution will be less than one that relies on handwritten code.

JSON messages tend to be small and relatively simple in structure – We could find no publically accessible large examples or complex schemas. The kinds of data the Finance community describes in XML can be significantly larger and more complicated. The perceived advantages of JSON may not be so evident when applied to complex data models.

Usefulness to FpML

In principle FpML information could be represented using JSON as the underlying data structures and capabilities are not that different: The key issues would be:

- Whilst a standard conversion from XML to JSON could be relatively easily for FpML data the reverse JSON to XML transformation would be very complex requiring the reordering of data properties to match the ordered expected by the XSD Schema. Automated tooling would be needed to create the transformation from the XML Schema definitions.

- The creation of an equivalent JSON Schema for FpML would also require the development of automated tooling for the task to be completed accurately. We suspect that in order to create a JSON Schema from the XSD Schema either the grammar would need to be simplified (e.g. a complex grammar like '(spotRate, (forwardPoints, pointValue?)?)?' would need to be simplified to 'spotRate?, forwardPoints?, pointValue?' which accepts the same valid data but allows some invalid combinations) additional levels of nesting would need to be added to preserve the constraints (e.g.'{ … "spotDetails" : { "spotRate" : 1.4500, "forwardDetails" : { "forwardPoints" : 0.0001 } } … }').

- There are numerous structural and data-type validations in the FpML we currently get for free with XSD Schema which would need to be documented carefully to allow them to be consistently coded in implementations.

Conclusions

FpML has always been an XML based standard and made extensive use of grammars to define its trade and product models, first using DTDs and later with XSD Schema. In this regard FpML is quite different from other Financial messaging standards like FIX (which is primarily based on structured key/value pairs) and ISO 20022 (which uses UML with a bespoke meta-model) which derive their XSD Schemas from a separate non-XML model.

FpML uses constraints expressed in grammar far more than other standards allowing a simple validating XML parse to detect a higher percentage of invalid documents than the other standards which are more reliant on post parse validation embedded within the processing applications code. [7]

We could create a JSON Schema automatically from our XML Schema if we are prepared to allow it to implement a far lower quality of the validation. We would need to define a far larger collection of business rules to make up for the loosening of the schema to check the quality of the resulting documents matches the XML. The simple data models which underlie FIX and ISO 20022 will more easily be translatable to JSON Schema.[8]

Whilst it is possible to represent FpML data in JSON the size and complexity of the resulting messages and the complexity of the transformation back to XML if the data is to be sent to an external service provider or regulator is such that it is probably easier just to use XML representations throughout.

Please send any comments or questions regarding this paper to archwgchair@fpml.org.

[7] FIXML makes extensive use of attributes to minimise document size. XML Schema only supports validation of data values and optional/mandatory cardinality on attributes.

[8] In an ISO XML Schemas sequences and choices are never nested within each other to form complex grammar.