Data Model Aware JSON

JSON Task Force

Create: 12/18/2017

Last Updated: 7/3/2018

Mission of JSON Task Force:

In order to align with technology trends in information transfer while still supporting PESC and A4L existing standards adopters, the mission of the JSON task force is to create design rules for both XML and JSON that promote expression of comparable semantics, simplicity of translation, and ease of implementation so that the education community can have a choice of exchange formats without sacrificing interoperability.

Document Purpose

The purpose of this document is establish JSON as a viable format for A4L and PESC data exchanges without sacrificing standardization. There have been many attempts to define translation rules between XML and JSON. Many of these rely on direct syntactical transformations without reference to an underlying data model, thus resulting in difficult interpretations and excessive type checking on the part of the receiving program code. For example, a repeatable element in XML schema that is a single element in an XML instance document, would be rendered in JSON as a name value pair (e.g., {"A": 3} in JSON, but if the element was repeated would be a JSON array (e.g., {"A": [3, 4]}). The programmer then has to do type checking and process the two cases differently. In the data model aware situation, the type would always be an array and type checking would not be needed.

This document is the first step in establishing JSON as a standard of exchange for A4L and PESC. The next step is explore alternative expression of data and validation models that would complement or replace XML Schema Language. Some of the alternatives we will explore include JSON Schema, JSON-LD, and Content Assembly Mechanism (CAM).

Objectives for JSON Generation

* To be compliant with A4L and PESC standards, JSON exchanges must follow the data models as expressed currently in XML schemas. Because JSON syntax is simpler than XML, A4L and PESC provide a set of rules that apply XML Schema data models to JSON exchanges.
* It should not be necessary to use an XML instance document to create JSON. A programing language object model based on XML schemas should be able to generate JSON.
* Any information for translating from JSON back to XML should not be contained in the JSON itself. For example, "@" or "\_" will not appear before attribute names to denote that it was an XML attribute. This will allow programmers to view JSON as they would for any application natively using JSON.
* The rules for data model aware JSON should reduce the need for programmers to do type checking. Some existence checking will still be required since some data objects are optional.

Tools Support

* To assist with the creation of data model aware JSON, various software tools may be used to encode the XML schema rules into language objects that can then be serialized into JSON, XML or other language. Our experience with these tools indicates that they may not enforce all constraints in their objects and that some additional code or post processing may need to be provided.
* Currently, a combination of JAXB (Java object model creation from XML Schema) and MoXY (JSON serialization) have been successfully used to create data model aware JSON. The PESC EdExchange program uses this tool to create JSON for transcript exchanges.
* The xmlschema package for Python has been used to translate between XML instance documents and JSON using the XML schema to drive the translation.

XML Simplification:

In order to align XML and JSON standards and simplify translation between the two, the following guidelines should be used in XML schema development:

* No mixed elements with complex content
* Limit use of attributes
* Limit namespace to the top level element in instance document.
* Do not define global elements in XML schemas as this will require namespace qualification of elements in instance documents.
* An element can be repeatable but an element name should not be used twice in a sequence.

JSON Generation and Translation Rules:

As with XML, it is understood that exchange partners may decide that certain rules as specified below do not fit their business models or tools. The JSON produced would not be considered A4L or PESC compliant and may not work in an exchange expecting compliance, but A4L and PESC would still encourage exchanges using our standards as guidelines even if not compliant. We would also appreciate feedback as to the reasons for the deviations so that we may improve our standards.

Below are rules defining how different data model entities will be serialized in JSON:

| XML feature | Translation | Data Model aware |
| --- | --- | --- |
| Attribute, complex content | Attributes on a complex element with complex content will be treated as another property value pair in the objects properties list.  <A attr="text"><B>text2</B></A> 🡪  "A":{"attr":"text","B":"text2"}  If the attribute or element is optional and does not exist, then leave it out. | No |
| Attribute, simple content | The simple content will be converted into a new object named for the simple element. If the attribute is optional according to the schema, this object will still be used. This property will be reused within the object:  <A attr="text">text2</A> 🡪  "A":{"attr":"text","value":"text2"} | No |
| XML numeric value (e.g., xs: integer) | Use the schema to determine if the value of a attribute or an element is a number or a string:  <A>3</A>  If a XML schema type is numeric  "A":3  If a string or not typed  "A":"3" | Yes |
| XML xs:boolean | Use the XML schema to determine if it is a JSON boolean or a string  <A>true</A>  If xs:boolean  "A": true  If not  "A":"true" | Yes |
| Repeatable element | The values of a repeatable element are translated to a JSON array.  <A>text1</A>  If element is repeatable (maxOccurs > 1):  "A":["text1"]  Otherwise maxOccurs = "1"  "A":"text1"  No matter how long the list is, it should be an array | Yes |
| xs:list | If the schema specifies a list then the space separated list is specified as an array  <A>1 2 3</A> 🡪"A":["1","2","3"] | Yes |
| xs:date  xs:time xs:dateTime | Use the date time format from the XML schema description which uses ISO 8601. For xs:dateTime  <A>1990-09-02T03:03:00-0500</A> 🡪"A":"1990-09-02T03:03:00-0500"  If translating, just copy. If generating, use ISO 8601. | Yes |
| If xs:nillable = "true" and  xsi:nil = "true" | <A xsi:nil = "true"/> or <A xsi:nil = "true"></A>🡪"A":null  Note: Only elements that are nillable in the schema may carry xsi:nil | Yes |
| Missing Tag | <A> is not in XML instance 🡪 exclude property | No |
| Required Empty Tag | If a string, <A/> or <A></A> 🡪 "A": "" or if repeatable <A></A> 🡪"A":[""]  If empty complex sequence then "A":{} if the child elements are optional  Note: By default JAXB will leave out the property in JSON if there is no data to send. | Yes |
| XML comments | <!--comment -->  Do not include comments in JSON. | No |
| XML Processing Instructions | <? xml version = "1.0" encoding = "utf-8" ?>  Do not include in JSON. | No |
| namespaces | Treat as attribute xmlns for any object needing it. | No |
| XPath in XML elements  Proposal use JSONPath | /AcademicEPortfolio/Competencies[CompetencyID="Competency1"] 🡪  $.AcademicEPortfolio.Competencies[?(@.CompencyID == "Comptency1")]  or  $["AcademicEPortfolio"]['Competencies'][?(@.CompetencyID =="Comptency1")] | No |
| xs:sequence | Property names should appear in JSON in the same sequence as XML (after the added attributes) | Yes |
| xs:choice | If maxOccurs is greater that one, keep the order in JSON as listed in the choice definition. | Yes |
| xs:union | The value can be any of the types listed so type checking will be necessary on the receiving program code. | Yes |
| Optional empty tag | If a string, <A/> 🡪 "A": "" otherwise do not translate to JSON | Yes |
| facets | Use XML Schema Language facets to constrain strings and numbers to sizes and patterns:  xs:length, xs:minLength, xs:maxLength, xs:enumeration, xs:pattern, xs:totalDigits, xs:fractionDigits,etc. xs:minInclusive, xs:maxInclusive, xs:whitespace | Yes |