FOR IMMEDIATE RELEASE
February 14, 2017
Contact:
Jennifer Kim
PESC Membership Services Director
+1.202.261.6516

JSON TASK FORCE TO LAUNCH AT PESC

INAUGURAL CONVENING AT PESC SPRING 2017 DATA SUMMIT

(Washington DC) – PESC is pleased to announce its latest initiative, the formation of a ***JSON TASK FORCE***. This Task Force is being established to advise PESC Members and the PESC Board of Directors on the impact and utility of JSON in the education domain and its relationship to XML. JavaScript Object Notation (JSON) has become a popular alternative to XML for various reasons, highest among them that JSON is less verbose than XML, has simpler syntax than XML and is more easily generated and consumed.

PESC's Technical Advisory Board (TAB) began discussions on JSON in 2014 and prepared a research paper entitled, *Use of JSON to Supplement XML*, which is attached to this announcement. Under the continued leadership of the PESC TAB and with support of PESC's Change Control Board and Standards Development Forum for Education, this Task Force will continue the discussions and ultimately recommend what action, if any, PESC should undertake as a result of the emergence of JSON.

Specifically, the Task Force is charged with producing a white paper that:

- Describes JSON
- Identifies how JSON is being used across education and throughout various other industries
- Details the values and benefits of JSON
- Describes how JSON and PESC Approved Standards in XML can be used together
- Recommends if PESC should establish PESC Approved Standards in JSON

The inaugural convening of this Task Force will occur at PESC's Spring 2017 Data Summit, taking place May 3-5, 2017 in Washington, D.C. at the Embassy Row Hotel in Dupont Circle. The general public is welcome to register and attend the Spring 2017 Data Summit and participation on this Task Force is open to the general public as well.

To join PESC's JSON Task Force, please visit and sign up here: http://www.pesc.org/contact-us-1.html.

"JASON" "J–S–O–N" "JAY–SAWN"

How do you pronounce JSON? Douglas Crockford of Yahoo, JSON creator, sets the record straight. Listen to his pronunciation here https://www.youtube.com/watch?v=zhVdWQWKRqM.

For more information about PESC and the Spring 2017 Data Summit, please visit www.PESC.org.

ABOUT PESC

Established in 1997 at the National Center for Higher Education and headquartered in Washington, D.C., PESC is an international, 501 (c)(3) non-profit, community-based, umbrella association of data, software and education technology service providers; schools, districts, colleges and universities; college, university and state systems; local, state/province and federal government agencies; professional, commercial and non-profit organizations; and non-profit associations & foundations.

Through open and transparent community participation, PESC enables cost-effective connectivity between data systems to accelerate performance and service, to simplify data access and research, and to improve data quality along the Education lifecycle. PESC envisions global interoperability within the Education domain, supported by a trustworthy, inter-connected network built by and between communities of interest in which data flows digitally and seamlessly from one community or system to another and throughout the entire eco-system when and where needed without compatibility barriers but in a safe, secure, reliable, legal, and efficient manner.

While PESC promotes the implementation and usage of data exchange standards, PESC does not set (create or establish) policies related to privacy and security. Organizations and entities using PESC Approved Standards and services should ensure they comply with FERPA and all local, state, federal and international rules on privacy and security as applicable. For more information, see www.PESC.org.

# # #

USE OF JSON TO SUPPLEMENT XML

Originally Prepared by the PESC Technical Advisory Board
Michael Morris, ACT
May 21, 2015

PROBLEM

JavaScript Object Notation (JSON) has become a popular alternative to XML for various reasons: It is less verbose, has simpler syntax than XML and is easily generated and consumed by client side JavaScript. JSON has found extensive use in **RE**presentational **ST**ate Services (REST or RESTful Web services) as the format for sending and receiving structured data for web applications. The older XML-based web services based upon Simple Object Access Protocol (SOAP) has fallen out of favor and is used with few new applications. While some RESTful web services still provide a choice of XML or JSON, many of web services provide only JSON.

To prepare for the same trend occurring in the data exchange realm, PESC must be able to support JSON when the education community requires it. To do this, PESC must have JSON solutions available when needed rather than be forced to react to industry trends.

This document, identifies the issues in translating between the two notations, and provides a recommendation on how PESC might proceed in incorporating JSON into its standards.

CONVERSION ISSUES

There are several differences between XML and JSON that may make translation difficult:

1. JSON Objects are not equivalent to XML Complex elements. In JSON, the order of the object properties is not required to be maintained like an XML sequence. Also, JSON object property names need to be unique while XML sub-elements can be duplicated.
2. Processing instructions and comments do not have an equivalent structure in JSON.
3. JSON does not have a date type, and thus, a string representing a date in JSON may need to be recognized as a date and translated into an XML date type format.
4. JSON has less strict naming rules than XML. Direct translation of names from JSON to XML may require the creation of new element names.
5. XML has standard validation specifications while JSON does not. While there is currently activity on creating a JSON schema, XML Schema language is the best supported method to specify and validate instance documents.

RECOMMENDATION

The TAB recommends that JSON conformant exchanges be sanctioned by PESC under the following conditions: For those applications requiring JSON exchange, the exchange can be classified as PESC conformant if the JSON is derived from a PESC schema validated XML instance document using the translation rules specified in this document (or using the PESC sanctioned XSLT conversion program).

In addition, PESC will provide an XSLT stylesheet that implements the conversion rules for use by the education community. The TAB has found an open source XSLT that can be used for this purpose.

TRANSLATION RULES

The rules below are recommended for adoptions as a PESC standard:

Object	Source example	Rule	Result
Element with only text	<A>text</A>	The element name is represented as JSON property name and the text as the JSON property value	"A":"text"
Namespace	<ns:A>text</ns:A)	The element name includes the namespace prefix.	"ns:A:":"text"

Element with only an attribute	<A x="att value"/>	The element is represented as a JSON object with the attribute represented as a name value pair property. To allow future potential two way conversion the attribute name is prefixed by "@".	"A":{
			"@x":"att value"
			}

Element with attribute(s) and text	<A x="att1 value" y="att2 value">text</A>	The property name "$" is used as the name of the property that has a value of the content of the element.	"A":{
			"@x":"att1 value",
			"@y":"att2 value",
			"$":"text"
			}

Null Elements	<A/>	The value of null elements is the JSON null unless there is an attribute.	"A":null

Complex Elements	<A>
	<B>text 1</B>
	<C>
	<D>text 2</D>
	</C>
	</A>
		Complex element contents will be treated as properties of the object that is named after the top level element	"A":{
			"B":"text 1",
			"C":{
			"D":"text 2"
			}
			}

Repeated Elements	<A>
	<B>text 1</B>
	<B>text 2</B>
	<C>
	<D>text 3</D>
	</C>
	<C>
	<D>text 4</D>
	</C>
	<A>
		Repeated elements use the element name as a JSON array name and the values in the array as the text or child elements of the parent element.	"A":{
			"B":["text 1",
			"text 2"],
			"C":["D":"text 3",
			"D":"text 4"]
			}

Entities	<A>"Testing"</A>	If the entity must be escaped in JSON then it is preceded by a slash. Other characters will be translated directly.	"A":"\"Testing\""
Special Characters	<A>C:\\</A>	If a character must be escaped in JSON, it will be preceded by a slash	"A":"C:\\\\"
Comments	<!--Comment-->	These will not be converted to JSON	
Processing Instructions	<?xml version="1.0" encoding="UTF-8"?>	These will not be converted to JSON	

PRODUCTION RULES FOR TRANSLATING XML TO JSON

Another way of expressing these rules is to use production rules for translating XML structures into JSON:

<element> ::= "<element name>":<element value>| "$":"<element text>"
<element value> ::= <element>|"<element text>"|<complex element>|<repeated element>|null
<complex element> ::= {<attribute list><element value>}
<repeated element> ::= [<element value>,<repeated element>|<element value>]
<attribute list> ::= <attribute>,<attribute list>|<attribute>
<attribute> ::= "@<attribute name>":"<attribute value>"
<attribute value> ::="<attribute value text>"

APPENDIX

JSON DEFINITION

This link provides the complete syntax for JSON:

http://json.org/

CONVERSION BETWEEN XML AND JSON:

http://www.xml.com/pub/a/2006/05/31/converting-between-xml-and-json.html

http://badgerfish.ning.com/

http://wiki.open311.org/index.php?title=JSON_and_XML_Conversion

http://www.bramstein.com/projects/xsltjson/

http://code.google.com/p/xml2json-xslt/

http://json-lib.sourceforge.net/index.html

We also tested the Altova XML-Spy conversion:

http://www.altova.com/xmlspy.html.

The conversion lost much of the XML structure in a round trip. It made attributes into elements and lost all processing instructions and comments.

JSON CONVERSION TO AND FROM POJO

https://json-processing-spec.java.net/

http://www.javaworld.com/article/2074650/core-java/javaone-2012--jsr-353--java-api-for-json-processing.html

http://examples.javacodegeeks.com/enterprise-java/rest/resteasy/json-example-with-resteasy-jaxb-jettison/

OTHER XML ORGANIZATIONS USING JSON

OASIS

The following link lists OASIS Technical Committees that either currently feature JSON and/or REST in their charters, or are discussing REST or JSON:

https://www.oasis-open.org/resources/topics/rest-json

OASIS has also recently approved version 4.0 of the Open Data Protocol (OData) and the OData JSON Format. A press release on MarketWatch is here: http://www.marketwatch.com/story/oasis-approves-odata-40-standards-for-an-open-programmable-web-2014-03-17

LANGUAGES THAT SUPPORT JSON PARSING

- C (jason-parser)
- awk (json.awk)
- C++ (a bunch, including JSONKit, JSON++ and libjson)
- C# (JSON for .net, JSONSharp, Manatee Json)
- Javascript (JSON, kson2.js, clarinet)
- Java (JSON Tools, google-gson, Argo, SOJO, XStream, Json-lib, jjson)
- Objective C (JSONKit, NSJSONSerialization, json-framework, ObjFW)
- Perl (CPAN, perl-JSON-SL)
- PHP (native in 5.2, Services_JSON, json)
- PL/SQL (pljson, Librarie-JSON)
- Python (standard library, simplejson, pyson, ultraison)
- Ruby (built-in)
- Visual Basic (VB-JSON, PW.JSON)

```
<ns0:CollegeTranscript xmlns:ns0="urn:org:pesc:message:CollegeTranscript:v1.6.0">
<TransmissionData>
<DocumentID>418</DocumentID>
<CreatedDateTime>2017-06-05T22:07:22.425Z</CreatedDateTime>
<DocumentTypeCode>StudentRequest</DocumentTypeCode>
<TransmissionType>Original</TransmissionType>
<Source>
<Organization>
<FICE>008073</FICE>
<OrganizationName>Butte College</OrganizationName>
<Contacts/>
</Organization>
</Source>
<Destination>
<Organization>
<MutuallyDefined>7</MutuallyDefined>
<OrganizationName>Data by Touch</OrganizationName>
<Contacts>
<Address>
<AddressLine>4417 Vico Way</AddressLine>
<City>Sacramento</City>
<StateProvinceCode>CA</StateProvinceCode>
<PostalCode>95864</PostalCode>
</Address>
<Phone>
<AreaCityCode>916</AreaCityCode>
<PhoneNumber>7610603</PhoneNumber>
</Phone>
<Email>
<EmailAddress>jwhetstone@ccctechcenter.org</EmailAddress>
</Email>
</Contacts>
</Organization>
</Destination>
<RequestTrackingID>163</RequestTrackingID>
</TransmissionData>
<Student>
<Person>
<SchoolAssignedPersonID>0</SchoolAssignedPersonID>
<SSN>000000000</SSN>
<PartialSSN>0000</PartialSSN>
<Birth>
<BirthDate>1995-01-01Z</BirthDate>
</Birth>
<Name>
<FirstName>John</FirstName>
<LastName>Doe</LastName>
</Name>
```

```
<HighSchool>
<OrganizationName>McClatchy High School</OrganizationName>
<CEEBACT>052705</CEEBACT>
</HighSchool>
<Contacts>
<Address>
<AddressLine>2406 Capitol Ave. #4</AddressLine>
<City>Sacramento</City>
<StateProvinceCode>CA</StateProvinceCode>
<PostalCode>95816</PostalCode>
</Address>
<Phone>
<PhoneNumber>0000000000</PhoneNumber>
</Phone>
<Email>
<EmailAddress>john.doe@edexchange.edu</EmailAddress>
</Email>
</Contacts>
</Person>
<AcademicRecord>
<School>
<OrganizationName>Sacramento City College</OrganizationName>
<FICE>001233</FICE>
</School>
<StudentLevel>
<StudentLevelCode>CollegeSophomore</StudentLevelCode>
</StudentLevel>
<AcademicSession>
<AcademicSessionDetail>
<SessionDesignator>2015-09</SessionDesignator>
<SessionName>Fall Semester 2015</SessionName>
<SessionBeginDate>2015-09-01Z</SessionBeginDate>
<SessionEndDate>2015-12-01Z</SessionEndDate>
</AcademicSessionDetail>
<StudentLevel>
<StudentLevelCode>CollegeSophomore</StudentLevelCode>
</StudentLevel>
<AcademicProgram>
<ProgramCIPCode>11.0101</ProgramCIPCode>
<AcademicProgramType>Major</AcademicProgramType>
<AcademicProgramName>Computer and Information Sciences,
General</AcademicProgramName>
</AcademicProgram>
<Course>
<CourseCreditBasis>Regular</CourseCreditBasis>
<CourseCreditUnits>Semester</CourseCreditUnits>
<CourseCreditLevel>LowerDivision</CourseCreditLevel>
<CourseCreditValue>4.0</CourseCreditValue>
```

<CourseCreditEarned>4.0</CourseCreditEarned>
<CourseAcademicGradeScaleCode>25</CourseAcademicGradeScaleCode>
<CourseAcademicGrade>A</CourseAcademicGrade>
<CourseCIPCode>11.0101</CourseCIPCode>
<CourseQualityPointsEarned>16.0</CourseQualityPointsEarned>
<CourseLevel>LowerDivision</CourseLevel>
<CourseGPAApplicabilityCode>Applicable</CourseGPAApplicabilityCode>
<CourseSubjectAbbreviation>CISC</CourseSubjectAbbreviation>
<CourseNumber>360</CourseNumber>
<CourseTitle>Information Technology 360</CourseTitle>
</Course>
<Course>
<CourseCreditBasis>Regular</CourseCreditBasis>
<CourseCreditUnits>Semester</CourseCreditUnits>
<CourseCreditLevel>LowerDivision</CourseCreditLevel>
<CourseCreditValue>4.0</CourseCreditValue>
<CourseCreditEarned>4.0</CourseCreditEarned>
<CourseAcademicGradeScaleCode>25</CourseAcademicGradeScaleCode>
<CourseAcademicGrade>A</CourseAcademicGrade>
<CourseCIPCode>11.0201</CourseCIPCode>
<CourseQualityPointsEarned>16.0</CourseQualityPointsEarned>
<CourseLevel>LowerDivision</CourseLevel>
<CourseGPAApplicabilityCode>Applicable</CourseGPAApplicabilityCode>
<CourseSubjectAbbreviation>CISP</CourseSubjectAbbreviation>
<CourseNumber>301</CourseNumber>
<CourseTitle>Algorithm Design and 4 Units Implementation</CourseTitle>
</Course>
<Course>
<CourseCreditBasis>Regular</CourseCreditBasis>
<CourseCreditUnits>Semester</CourseCreditUnits>
<CourseCreditLevel>LowerDivision</CourseCreditLevel>
<CourseCreditValue>5.0</CourseCreditValue>
<CourseCreditEarned>5.0</CourseCreditEarned>
<CourseAcademicGradeScaleCode>25</CourseAcademicGradeScaleCode>
<CourseAcademicGrade>B</CourseAcademicGrade>
<CourseCIPCode>27.0102</CourseCIPCode>
<CourseQualityPointsEarned>15.0</CourseQualityPointsEarned>
<CourseLevel>LowerDivision</CourseLevel>
<CourseGPAApplicabilityCode>Applicable</CourseGPAApplicabilityCode>
<CourseSubjectAbbreviation>MATH</CourseSubjectAbbreviation>
<CourseNumber>120</CourseNumber>
<CourseTitle>Intermediate Algebra</CourseTitle>
</Course>
<AcademicSummary>
<AcademicSummaryType>SenderOnly</AcademicSummaryType>
<AcademicSummaryLevel>LowerDivision</AcademicSummaryLevel>
<GPA>
<CreditHoursAttempted>13.0</CreditHoursAttempted>

<CreditHoursEarned>13.0</CreditHoursEarned>
<CreditUnit>Semester</CreditUnit>
<GradePointAverage>4.00</GradePointAverage>
<TotalQualityPoints>52.0</TotalQualityPoints>
<CreditHoursforGPA>47.0</CreditHoursforGPA>
<GPARangeMinimum>0.0</GPARangeMinimum>
<GPARangeMaximum>4.0</GPARangeMaximum>
</GPA>
<AcademicHonors>
<HonorsTitle>Dean's List</HonorsTitle>
<HonorsLevel>SecondHighest</HonorsLevel>
</AcademicHonors>
</AcademicSummary>
</AcademicSession>
<AcademicSession>
<AcademicSessionDetail>
<SessionDesignator>2016-01</SessionDesignator>
<SessionName>Spring Semester 2016</SessionName>
<SessionBeginDate>2016-01-15Z</SessionBeginDate>
<SessionEndDate>2016-06-15Z</SessionEndDate>
</AcademicSessionDetail>
<StudentLevel>
<StudentLevelCode>CollegeSophomore</StudentLevelCode>
</StudentLevel>
<AcademicProgram>
<ProgramCIPCode>11.0101</ProgramCIPCode>
<AcademicProgramType>Major</AcademicProgramType>
<AcademicProgramName>Computer and Information Sciences,
General</AcademicProgramName>
</AcademicProgram>
<Course>
<CourseCreditBasis>Regular</CourseCreditBasis>
<CourseCreditUnits>Semester</CourseCreditUnits>
<CourseCreditLevel>LowerDivision</CourseCreditLevel>
<CourseCreditValue>5.0</CourseCreditValue>
<CourseCreditEarned>5.0</CourseCreditEarned>
<CourseAcademicGradeScaleCode>25</CourseAcademicGradeScaleCode>
<CourseAcademicGrade>A</CourseAcademicGrade>
<CourseCIPCode>27.0102</CourseCIPCode>
<CourseQualityPointsEarned>20.0</CourseQualityPointsEarned>
<CourseLevel>LowerDivision</CourseLevel>
<CourseGPAApplicabilityCode>Applicable</CourseGPAApplicabilityCode>
<CourseSubjectAbbreviation>MATH</CourseSubjectAbbreviation>
<CourseNumber>100</CourseNumber>
<CourseTitle>Elementary Algebra</CourseTitle>
</Course>
<Course>
<CourseCreditBasis>Regular</CourseCreditBasis>

<CourseCreditUnits>Semester</CourseCreditUnits>
<CourseCreditLevel>LowerDivision</CourseCreditLevel>
<CourseCreditValue>3.0</CourseCreditValue>
<CourseCreditEarned>3.0</CourseCreditEarned>
<CourseAcademicGradeScaleCode>25</CourseAcademicGradeScaleCode>
<CourseAcademicGrade>A</CourseAcademicGrade>
<CourseCIPCode>11.0801</CourseCIPCode>
<CourseQualityPointsEarned>12.0</CourseQualityPointsEarned>
<CourseLevel>LowerDivision</CourseLevel>
<CourseGPAApplicabilityCode>Applicable</CourseGPAApplicabilityCode>
<CourseSubjectAbbreviation>CISW</CourseSubjectAbbreviation>
<CourseNumber>304</CourseNumber>
<CourseTitle>Cascading Style Sheets</CourseTitle>
</Course>
<Course>
<CourseCreditBasis>Regular</CourseCreditBasis>
<CourseCreditUnits>Semester</CourseCreditUnits>
<CourseCreditLevel>LowerDivision</CourseCreditLevel>
<CourseCreditValue>3.0</CourseCreditValue>
<CourseCreditEarned>3.0</CourseCreditEarned>
<CourseAcademicGradeScaleCode>25</CourseAcademicGradeScaleCode>
<CourseAcademicGrade>A</CourseAcademicGrade>
<CourseCIPCode>11.0801</CourseCIPCode>
<CourseQualityPointsEarned>12.0</CourseQualityPointsEarned>
<CourseLevel>LowerDivision</CourseLevel>
<CourseGPAApplicabilityCode>Applicable</CourseGPAApplicabilityCode>
<CourseSubjectAbbreviation>CISW</CourseSubjectAbbreviation>
<CourseNumber>320</CourseNumber>
<CourseTitle>Introduction to Web Site</CourseTitle>
</Course>
<Course>
<CourseCreditBasis>Regular</CourseCreditBasis>
<CourseCreditUnits>Semester</CourseCreditUnits>
<CourseCreditLevel>LowerDivision</CourseCreditLevel>
<CourseCreditValue>4.0</CourseCreditValue>
<CourseCreditEarned>4.0</CourseCreditEarned>
<CourseAcademicGradeScaleCode>25</CourseAcademicGradeScaleCode>
<CourseAcademicGrade>A</CourseAcademicGrade>
<CourseCIPCode>11.0801</CourseCIPCode>
<CourseQualityPointsEarned>16.0</CourseQualityPointsEarned>
<CourseLevel>LowerDivision</CourseLevel>
<CourseGPAApplicabilityCode>Applicable</CourseGPAApplicabilityCode>
<CourseSubjectAbbreviation>CISW</CourseSubjectAbbreviation>
<CourseNumber>400</CourseNumber>
<CourseTitle>Client-side Web Scripting</CourseTitle>
</Course>
<AcademicSummary>
<AcademicSummaryType>SenderOnly</AcademicSummaryType>

<AcademicSummaryLevel>LowerDivision</AcademicSummaryLevel>
<GPA>
<CreditHoursAttempted>15.0</CreditHoursAttempted>
<CreditHoursEarned>15.0</CreditHoursEarned>
<CreditUnit>Semester</CreditUnit>
<GradePointAverage>4.00</GradePointAverage>
<TotalQualityPoints>60.0</TotalQualityPoints>
<CreditHoursforGPA>60.0</CreditHoursforGPA>
<GPARangeMinimum>0.0</GPARangeMinimum>
<GPARangeMaximum>4.0</GPARangeMaximum>
</GPA>
<AcademicHonors>
<HonorsTitle>Dean's List</HonorsTitle>
<HonorsLevel>FirstHighest</HonorsLevel>
</AcademicHonors>
</AcademicSummary>
</AcademicSession>
</AcademicRecord>
</Student>
</ns0:CollegeTranscript>

```
{
   "CollegeTranscript" : {
      "TransmissionData" : {
         "DocumentID" : "204",
         "CreatedDateTime" : "2017-06-12T16:37:24.684Z",
         "DocumentTypeCode" : "StudentRequest",
         "TransmissionType" : "Original",
         "Source" : {
            "Organization" : {
               "OPEID" : "00133400",
               "NCHELPID" : "00133400",
               "IPEDS" : "125028",
               "ATP" : "493111",
               "FICE" : "001334",
               "ACT" : "0476",
               "CEEBACT" : "493111",
               "MutuallyDefined" : "32960",
               "OrganizationName" : [ "Ventura College" ],
               "Contacts" : [ {
               } ]
            }
         },
         "Destination" : {
            "Organization" : {
               "DUNS" : "013900836",
               "OrganizationName" : [ "Parchment Inc." ],
               "Contacts" : [ {
                  "Address" : [ {
                     "AddressLine" : [ "7001 N Scottsdale Rd, Ste. 1050" ],
                     "City" : "Scottsdale",
                     "StateProvinceCode" : "AZ",
                     "PostalCode" : "85250"
                  } ]
               } ]
            }
         },
         "RequestTrackingID" : "202"
      },
      "Student" : {
         "Person" : {
            "SchoolAssignedPersonID" : "0",
            "SSN" : "000000000",
            "PartialSSN" : "0000",
            "Birth" : {
               "BirthDate" : "1995-01-01"
            },
            "Name" : {
               "FirstName" : "John",
```

```
            "LastName" : "Doe"
         },
         "HighSchool" : {
            "OrganizationName" : "McClatchy High School",
            "CEEBACT" : "052705"
         },
         "Contacts" : [ {
            "Address" : [ {
               "AddressLine" : [ "2406 Capitol Ave. #4" ],
               "City" : "Sacramento",
               "StateProvinceCode" : "CA",
               "PostalCode" : "95816"
            } ],
            "Phone" : [ {
               "PhoneNumber" : "0000000000"
            } ],
            "Email" : [ {
               "EmailAddress" : "john.doe@edexchange.edu"
            } ]
         } ]
      },
      "AcademicRecord" : [ {
         "School" : {
            "OrganizationName" : "Sacramento City College",
            "FICE" : "001233"
         },
         "StudentLevel" : {
            "StudentLevelCode" : "CollegeSophomore"
         },
         "AcademicSession" : [ {
            "AcademicSessionDetail" : {
               "SessionDesignator" : "2015-09",
               "SessionName" : "Fall Semester 2015",
               "SessionBeginDate" : "2015-09-01",
               "SessionEndDate" : "2015-12-01"
            },
            "StudentLevel" : {
               "StudentLevelCode" : "CollegeSophomore"
            },
            "AcademicProgram" : [ {
               "ProgramCIPCode" : "11.0101",
               "AcademicProgramType" : "Major",
               "AcademicProgramName" : "Computer and Information Sciences,
General"
            } ],
            "Course" : [ {
               "CourseCreditBasis" : "Regular",
               "CourseCreditUnits" : "Semester",
```

```
   "CourseCreditLevel" : "LowerDivision",
   "CourseCreditValue" : 4.0,
   "CourseCreditEarned" : 4.0,
   "CourseAcademicGradeScaleCode" : "25",
   "CourseAcademicGrade" : "A",
   "CourseCIPCode" : "11.0101",
   "CourseQualityPointsEarned" : 16.0,
   "CourseLevel" : "LowerDivision",
   "CourseGPAApplicabilityCode" : "Applicable",
   "CourseSubjectAbbreviation" : "CISC",
   "CourseNumber" : "360",
   "CourseTitle" : "Information Technology 360"
}, {
   "CourseCreditBasis" : "Regular",
   "CourseCreditUnits" : "Semester",
   "CourseCreditLevel" : "LowerDivision",
   "CourseCreditValue" : 4.0,
   "CourseCreditEarned" : 4.0,
   "CourseAcademicGradeScaleCode" : "25",
   "CourseAcademicGrade" : "A",
   "CourseCIPCode" : "11.0201",
   "CourseQualityPointsEarned" : 16.0,
   "CourseLevel" : "LowerDivision",
   "CourseGPAApplicabilityCode" : "Applicable",
   "CourseSubjectAbbreviation" : "CISP",
   "CourseNumber" : "301",
   "CourseTitle" : "Algorithm Design and 4 Units Implementation"
}, {
   "CourseCreditBasis" : "Regular",
   "CourseCreditUnits" : "Semester",
   "CourseCreditLevel" : "LowerDivision",
   "CourseCreditValue" : 5.0,
   "CourseCreditEarned" : 5.0,
   "CourseAcademicGradeScaleCode" : "25",
   "CourseAcademicGrade" : "B",
   "CourseCIPCode" : "27.0102",
   "CourseQualityPointsEarned" : 15.0,
   "CourseLevel" : "LowerDivision",
   "CourseGPAApplicabilityCode" : "Applicable",
   "CourseSubjectAbbreviation" : "MATH",
   "CourseNumber" : "120",
   "CourseTitle" : "Intermediate Algebra"
} ],
"AcademicSummary" : [ {
   "AcademicSummaryType" : "SenderOnly",
   "AcademicSummaryLevel" : "LowerDivision",
   "GPA" : {
      "CreditHoursAttempted" : 13.0,
```

```
                  "CreditHoursEarned" : 13.0,
                  "CreditUnit" : "Semester",
                  "GradePointAverage" : 4.00,
                  "TotalQualityPoints" : 52.0,
                  "CreditHoursforGPA" : 47.0,
                  "GPARangeMinimum" : 0.0,
                  "GPARangeMaximum" : 4.0
               },
               "AcademicHonors" : [ {
                  "HonorsTitle" : "Dean's List",
                  "HonorsLevel" : "SecondHighest"
               } ]
            } ]
         }, {
            "AcademicSessionDetail" : {
               "SessionDesignator" : "2016-01",
               "SessionName" : "Spring Semester 2016",
               "SessionBeginDate" : "2016-01-15",
               "SessionEndDate" : "2016-06-15"
            },
            "StudentLevel" : {
               "StudentLevelCode" : "CollegeSophomore"
            },
            "AcademicProgram" : [ {
               "ProgramCIPCode" : "11.0101",
               "AcademicProgramType" : "Major",
               "AcademicProgramName" : "Computer and Information Sciences,
General"
            } ],
            "Course" : [ {
               "CourseCreditBasis" : "Regular",
               "CourseCreditUnits" : "Semester",
               "CourseCreditLevel" : "LowerDivision",
               "CourseCreditValue" : 5.0,
               "CourseCreditEarned" : 5.0,
               "CourseAcademicGradeScaleCode" : "25",
               "CourseAcademicGrade" : "A",
               "CourseCIPCode" : "27.0102",
               "CourseQualityPointsEarned" : 20.0,
               "CourseLevel" : "LowerDivision",
               "CourseGPAApplicabilityCode" : "Applicable",
               "CourseSubjectAbbreviation" : "MATH",
               "CourseNumber" : "100",
               "CourseTitle" : "Elementary Algebra"
            }, {
               "CourseCreditBasis" : "Regular",
               "CourseCreditUnits" : "Semester",
               "CourseCreditLevel" : "LowerDivision",
```

```
   "CourseCreditValue" : 3.0,
   "CourseCreditEarned" : 3.0,
   "CourseAcademicGradeScaleCode" : "25",
   "CourseAcademicGrade" : "A",
   "CourseCIPCode" : "11.0801",
   "CourseQualityPointsEarned" : 12.0,
   "CourseLevel" : "LowerDivision",
   "CourseGPAApplicabilityCode" : "Applicable",
   "CourseSubjectAbbreviation" : "CISW",
   "CourseNumber" : "304",
   "CourseTitle" : "Cascading Style Sheets"
}, {
   "CourseCreditBasis" : "Regular",
   "CourseCreditUnits" : "Semester",
   "CourseCreditLevel" : "LowerDivision",
   "CourseCreditValue" : 3.0,
   "CourseCreditEarned" : 3.0,
   "CourseAcademicGradeScaleCode" : "25",
   "CourseAcademicGrade" : "A",
   "CourseCIPCode" : "11.0801",
   "CourseQualityPointsEarned" : 12.0,
   "CourseLevel" : "LowerDivision",
   "CourseGPAApplicabilityCode" : "Applicable",
   "CourseSubjectAbbreviation" : "CISW",
   "CourseNumber" : "320",
   "CourseTitle" : "Introduction to Web Site"
}, {
   "CourseCreditBasis" : "Regular",
   "CourseCreditUnits" : "Semester",
   "CourseCreditLevel" : "LowerDivision",
   "CourseCreditValue" : 4.0,
   "CourseCreditEarned" : 4.0,
   "CourseAcademicGradeScaleCode" : "25",
   "CourseAcademicGrade" : "A",
   "CourseCIPCode" : "11.0801",
   "CourseQualityPointsEarned" : 16.0,
   "CourseLevel" : "LowerDivision",
   "CourseGPAApplicabilityCode" : "Applicable",
   "CourseSubjectAbbreviation" : "CISW",
   "CourseNumber" : "400",
   "CourseTitle" : "Client-side Web Scripting"
} ],
"AcademicSummary" : [ {
   "AcademicSummaryType" : "SenderOnly",
   "AcademicSummaryLevel" : "LowerDivision",
   "GPA" : {
      "CreditHoursAttempted" : 15.0,
      "CreditHoursEarned" : 15.0,
```

```
                  "CreditUnit" : "Semester",
                  "GradePointAverage" : 4.00,
                  "TotalQualityPoints" : 60.0,
                  "CreditHoursforGPA" : 60.0,
                  "GPARangeMinimum" : 0.0,
                  "GPARangeMaximum" : 4.0
               },
               "AcademicHonors" : [ {
                  "HonorsTitle" : "Dean's List",
                  "HonorsLevel" : "FirstHighest"
               } ]
            } ]
         } ]
      } ]
   }
}
```

PESC

FOR THE COMMUNITY, BY THE COMMUNITY
IN SUPPORT OF STUDENT ACHIEVEMENT

Technical Advisory Board

JSON Initiative

PESC

FOR THE COMMUNITY, BY THE COMMUNITY
IN SUPPORT OF STUDENT ACHIEVEMENT

Quotes that I live by

- "I would never die for my beliefs because I might be wrong."
- "To die for an idea: it is unquestionably noble. But how much nobler it would be if men died for ideas that were true."
- "It's tough to make predictions, especially about the future."
- "The older I grow the more I distrust the familiar doctrine that age brings wisdom."
- "There are no solutions, only trade-offs"

PESC

FOR THE COMMUNITY, BY THE COMMUNITY
IN SUPPORT OF STUDENT ACHIEVEMENT

JavaScript Object Notation (JSON)

- In use by most RESTful web services
- Can be marshaled and unmarshaled directly with JavaScript
- All major programming languages provide JSON APIs.
- JSON Schema standard supported by third party tools (XML Spy)

PESC

FOR THE COMMUNITY, BY THE COMMUNITY
IN SUPPORT OF STUDENT ACHIEVEMENT

Issues

- Mapping difficulties: Multiple solutions
  - <e attr="value"/>
  - <e>text</e>
  - <e attr="value">text</e>
  - <e>text</e>
  - <e>texttext</e>
- JSON Schema is not a W3C standard
- Security Standards are not defined (https only)

PESC

FOR THE COMMUNITY, BY THE COMMUNITY
IN SUPPORT OF STUDENT ACHIEVEMENT

TAB Activities

- Open Source (BSD license) XSLT modified to translate XML instance documents into JSON. Set of flags for different formats
- XSLT to translate XML Schema to JSON Schema
  - Missing Features: choice mapping, documentation (description), code definition

PESC

FOR THE COMMUNITY, BY THE COMMUNITY
IN SUPPORT OF STUDENT ACHIEVEMENT

TAB JSON Original Recommendation 2015

- PESC will recognize that a JSON message exchange conforms to PESC standards:
  - If the JSON instance was derived from a PESC schema validated XML instance document, and the transformation uses PESC specified transformation rules
- When JSON has a recognized schema approach, PESC can then provide direct standards for JSON

PESC

FOR THE COMMUNITY, BY THE COMMUNITY
IN SUPPORT OF STUDENT ACHIEVEMENT

Questions for Task Force

- Is there a need to exchange education information in JSON format?
- Should PESC provide a translation standard between JSON and XML?
- Should we add the additional linking support of JSON-LD?
- Should exchange definitions include RESTful web service definitions as part of the standard or use EdExchange

PESC

FOR THE COMMUNITY, BY THE COMMUNITY
IN SUPPORT OF STUDENT ACHIEVEMENT

Possible approaches (trade-offs)

- XSLT from valid XML instance to JSON instance
- XSLT to transform XML Schema to JSON Schema
- JSON Schema from scratch using PESC specification
- JSON-LD implementation
- Swagger or other web service specification for web service definition
- AI “interlingua” for future dialects

PESC

FOR THE COMMUNITY, BY THE COMMUNITY
IN SUPPORT OF STUDENT ACHIEVEMENT

Goessner Notation (A4L)

Pattern	XML	JSON	Access
1	<e/>	"e": null	o.e
2	<e>text</e>	"e": "text"	o.e
3	<e name="value" />	"e":{"@name": "value"}	o.e["@name"]
4	<e name="value">text</e>	"e": { "@name": "value", "#text": "text" }	o.e["@name"] o.e["#text"]
5	<e> <a>text text </e>	"e": { "a": "text", "b": "text" }	o.e.a o.e.b
6	<e> <a>text <a>text </e>	"e": { "a": ["text", "text"] }	o.e.a[0] o.e.a[1]
7	<e> text <a>text </e>	"e": { "#text": "text", "a": "text" }	o.e["#text"] o.e.a

http://www.xml.com/pub/a/2006/05/31/converting-between-xml-and-json.html

PESC

FOR THE COMMUNITY, BY THE COMMUNITY
IN SUPPORT OF STUDENT ACHIEVEMENT

Example PESC JSON

```
{
    "TransmissionData": {
        "DocumentID": "12345CV",
        "CreatedDateTime": "2016-02-29",
        "TransmissionType": "Resubmission",
        "DocumentTypeCode": "Application",
        "Source": {},
        "Destination": {},
        "NoteMessage": [
            "First Message", "Second Message"
        ]
    }
}
```

PESC

FOR THE COMMUNITY, BY THE COMMUNITY
IN SUPPORT OF STUDENT ACHIEVEMENT

Task Force Activities

- Set up meeting time with Doodle
- Review each possible approach and determine viability
- If multiple or phase solutions, lay out an implementation plan

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="2.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:json="http://json.org/">

  <xsl:output indent="no" omit-xml-declaration="yes" method="text"
encoding="utf-8"/>
  <xsl:strip-space elements="*"/>

  <!--
     XSLTJSON v1.0.93.

     You can use these parameters to control the output by supplying them to
     stylesheet. Consult the manual of your XSLT processor for instructions
     on how to pass parameters to a stylesheet.

     * debug           -  Enable or disable the output of the temporary
                          XML tree used to generate JSON output.
     * use-rabbitfish  -  Output basic JSON with a '@' to indicate XML
                          attributes.
     * use-badgerfish  -  Use the BadgerFish (http://badgerfish.ning.com/)
                          convention to output JSON without XML namespaces.
     * use-rayfish     -  Use the RayFish
(http://onperl.org/blog/onperl/page/rayfish)
                          convention to output JSON without XML namespaces.
     * use-namespaces  -  Output XML namespaces according to the
                          BadgerFish convention.
     * skip-root       -  Skip the root XML element.
     * jsonp           -  Enable JSONP; the JSON output will be prepended with
                          the value of the jsonp parameter and wrapped in
parentheses.

     Credits:
       Chick Markley (chick@diglib.org) - Octal number & numbers with terminating
period.
       Torben Schreiter (Torben.Schreiter@inubit.com) - Suggestions for skip root
and node list.
       Michael Nilsson - Bug report and unit tests for json:force-array feature.
       Frank Schwichtenberg - Namespace prefix name bug.
       Wilson Cheung - Bug report and fix for invalid number serialization.
       Danny Cohn - Bug report and fix for invalid floating point number
serialization.

     Copyright:
      2006-2014, Bram Stein

      Licensed under the new BSD License.
```

```
      All rights reserved.
  -->
  <xsl:param name="debug" as="xs:boolean" select="false()"/>
  <xsl:param name="use-rabbitfish" as="xs:boolean" select="true()"/>
  <xsl:param name="use-badgerfish" as="xs:boolean" select="true()"/>
  <xsl:param name="use-namespaces" as="xs:boolean" select="false()"/>
  <xsl:param name="use-rayfish" as="xs:boolean" select="false()"/>
  <xsl:param name="jsonp" as="xs:string" select="''"/>
  <xsl:param name="skip-root" as="xs:boolean" select="false()"/>

  <!--
    If you import or include the stylesheet in your own stylesheet you
    can use this function to transform any XML node to JSON.
  -->
  <xsl:function name="json:generate" as="xs:string">
    <xsl:param name="input" as="node()"/>

    <xsl:variable name="json-tree">
      <json:object>
        <xsl:copy-of select="if (not($use-rayfish)) then json:create-node($input,
false()) else json:create-simple-node($input)"/>
      </json:object>
    </xsl:variable>

    <xsl:variable name="json-mtree">
      <xsl:choose>
        <xsl:when test="$skip-root">
          <xsl:copy-of
select="$json-tree/json:object/json:member/json:value/child::node()"/>
        </xsl:when>
        <xsl:otherwise>
          <xsl:copy-of select="$json-tree"/>
        </xsl:otherwise>
      </xsl:choose>
    </xsl:variable>

    <xsl:variable name="output">
      <xsl:choose>
        <xsl:when test="normalize-space($jsonp)">
          <xsl:value-of select="$jsonp"/><xsl:text>(</xsl:text><xsl:apply-templates
select="$json-mtree" mode="json"/><xsl:text>)</xsl:text>
        </xsl:when>
        <xsl:otherwise>
          <xsl:text/><xsl:apply-templates select="$json-mtree"
mode="json"/><xsl:text/>
        </xsl:otherwise>
      </xsl:choose>
    </xsl:variable>
```

```
    <xsl:sequence select="$output"/>
  </xsl:function>

  <!--
    Template to match the root node so that the stylesheet can also
    be used on the command line.
  -->

  <xsl:template match="/*">
      <xsl:choose>
        <xsl:when test="$debug">
          <xsl:variable name="json-tree">
            <json:object>
              <xsl:copy-of select="if (not($use-rayfish)) then json:create-node(.,
false()) else json:create-simple-node(.)"/>
            </json:object>
          </xsl:variable>

          <debug>
            <xsl:copy-of select="$json-tree"/>
          </debug>
          <xsl:apply-templates select="$json-tree" mode="json"/>
        </xsl:when>
        <xsl:otherwise>
          <xsl:value-of select="json:generate(.)"/>
        </xsl:otherwise>
      </xsl:choose>
  </xsl:template>

  <!--
    All methods below are private methods and should not be used
    standalone.
  -->
  <xsl:template name="json:build-tree">
    <xsl:param name="input" as="node()"/>
    <json:object>
      <xsl:copy-of select="if (not($use-rayfish)) then json:create-node($input,
false()) else json:create-simple-node($input/child::node())"/>
    </json:object>
  </xsl:template>

  <xsl:function name="json:create-simple-node-member" as="node()">
    <xsl:param name="type" as="xs:string"/>
    <xsl:param name="value"/>
    <json:member>
      <json:name><xsl:value-of select="$type"/></json:name>
      <json:value><xsl:copy-of select="$value"/></json:value>
```

```
    </json:member>
  </xsl:function>

  <xsl:function name="json:create-simple-node" as="node()*">
    <xsl:param name="node" as="node()"/>

    <xsl:copy-of select="json:create-simple-node-member('#name',
$node/local-name())"/>
    <xsl:copy-of select="json:create-simple-node-member('#text',
$node/child::text())"/>

    <xsl:variable name="empty-array">
      <json:array/>
    </xsl:variable>

    <xsl:variable name="children">
      <json:array>
        <xsl:for-each select="$node/@*">
          <json:array-value>
            <json:value>
              <json:object>
                <xsl:copy-of select="json:create-simple-node-member('#name',
concat('@',./local-name()))"/>
                <xsl:copy-of select="json:create-simple-node-member('#text',
string(.))"/>
                <xsl:copy-of select="json:create-simple-node-member('#children',
$empty-array)"/>
              </json:object>
            </json:value>
          </json:array-value>
        </xsl:for-each>
        <xsl:for-each select="$node/child::element()">
          <json:array-value>
            <json:value>
              <json:object>
                <xsl:copy-of select="json:create-simple-node(.)"/>
              </json:object>
            </json:value>
          </json:array-value>
        </xsl:for-each>
      </json:array>
    </xsl:variable>
    <xsl:copy-of select="json:create-simple-node-member('#children', $children)"/>
  </xsl:function>

  <xsl:function name="json:create-node" as="node()">
    <xsl:param name="node" as="node()"/>
    <xsl:param name="in-array" as="xs:boolean"/>
```

```
    <xsl:choose>
      <xsl:when test="$in-array">
        <json:array-value>
          <json:value>
            <xsl:copy-of select="json:create-children($node)"/>
          </json:value>
        </json:array-value>
      </xsl:when>
      <xsl:otherwise>
        <json:member>
          <xsl:copy-of select="json:create-string($node)"/>
          <json:value>
            <xsl:copy-of select="json:create-children($node)"/>
          </json:value>
        </json:member>
      </xsl:otherwise>
    </xsl:choose>
  </xsl:function>

  <xsl:function name="json:create-children">
    <xsl:param name="node" as="node()"/>
    <xsl:choose>
      <xsl:when test="exists($node/child::text()) and count($node/child::node()) eq
1">
        <xsl:choose>
          <xsl:when test="(count($node/namespace::*) gt 0 and $use-namespaces) or
count($node/@*[not(../@json:force-array) or count(.|../@json:force-array)=2]) gt 0">
            <json:object>
              <xsl:copy-of select="json:create-namespaces($node)"/>
              <xsl:copy-of select="json:create-attributes($node)"/>
              <json:member>
                <json:name>$</json:name>
                <json:value><xsl:value-of select="$node"/></json:value>
              </json:member>
            </json:object>
          </xsl:when>
          <xsl:otherwise>
            <xsl:copy-of select="json:create-text-value($node)"/>
          </xsl:otherwise>
        </xsl:choose>
      </xsl:when>
      <xsl:when test="exists($node/child::text())">
        <xsl:choose>
          <xsl:when test="(count($node/namespace::*) gt 0 and $use-namespaces) or
count($node/@*[not(../@json:force-array) or count(.|../@json:force-array)=2]) gt 0">
            <json:object>
              <xsl:copy-of select="json:create-namespaces($node)"/>
              <xsl:copy-of select="json:create-attributes($node)"/>
```

```
            <json:member>
              <json:name>$</json:name>
              <json:value>
                <xsl:copy-of select="json:create-mixed-array($node)"/>
              </json:value>
            </json:member>
          </json:object>
        </xsl:when>
        <xsl:otherwise>
          <xsl:copy-of select="json:create-mixed-array($node)"/>
        </xsl:otherwise>
      </xsl:choose>
    </xsl:when>
    <xsl:when test="exists($node/child::node()) or ((count($node/namespace::*) gt
0 and $use-namespaces) or count($node/@*[not(../@json:force-array) or
count(.|../@json:force-array)=2]) gt 0)">
      <json:object>
        <xsl:copy-of select="json:create-namespaces($node)"/>
        <xsl:copy-of select="json:create-attributes($node)"/>
        <xsl:for-each-group select="$node/child::node()"
group-adjacent="local-name()">
          <xsl:choose>
            <xsl:when test="count(current-group()) eq 1 and
(not(exists(./@json:force-array)) or ./@json:force-array eq 'false')">
              <xsl:copy-of select="json:create-node(current-group()[1],
false())"/>
            </xsl:when>
            <xsl:otherwise>
              <json:member>
                <json:name><xsl:value-of select="if($use-namespaces) then
current-group()[1]/name() else current-group()[1]/local-name()"/></json:name>
                <json:value>
                  <json:array>
                    <xsl:for-each select="current-group()">
                      <xsl:copy-of select="json:create-node(.,true())"/>
                    </xsl:for-each>
                  </json:array>
                </json:value>
              </json:member>
            </xsl:otherwise>
          </xsl:choose>
        </xsl:for-each-group>
      </json:object>
    </xsl:when>
  </xsl:choose>
</xsl:function>

<xsl:function name="json:create-mixed-array" as="node()">
```

```
  <xsl:param name="node" as="node()"/>
  <json:array>
    <xsl:for-each select="$node/child::node()">
      <json:array-value>
        <json:value>
          <xsl:choose>
            <xsl:when test="self::text()">
              <xsl:copy-of select="json:create-text-value(.)"/>
            </xsl:when>
            <xsl:otherwise>
              <json:object>
                <xsl:copy-of select="json:create-node(.,false())"/>
              </json:object>
            </xsl:otherwise>
          </xsl:choose>
        </json:value>
      </json:array-value>
    </xsl:for-each>
  </json:array>
</xsl:function>

<xsl:function name="json:create-text-value" as="node()">
  <xsl:param name="node" as="node()"/>
  <xsl:choose>
    <xsl:when test="$use-badgerfish">
      <json:object>
        <json:member>
          <json:name>$</json:name>
          <json:value>
            <xsl:value-of select="$node"/>
          </json:value>
        </json:member>
      </json:object>
    </xsl:when>
    <xsl:otherwise>
      <xsl:value-of select="$node"/>
    </xsl:otherwise>
  </xsl:choose>
</xsl:function>

<xsl:function name="json:create-string" as="node()">
  <xsl:param name="node" as="node()"/>
  <xsl:choose>
    <xsl:when test="$use-namespaces">
      <json:name><xsl:value-of select="$node/name()"/></json:name>
    </xsl:when>
    <xsl:otherwise>
      <json:name><xsl:value-of select="$node/local-name()"/></json:name>
```

```
      </xsl:otherwise>
    </xsl:choose>
  </xsl:function>

  <xsl:function name="json:create-attributes" as="node()*">
    <xsl:param name="node" as="node()"/>
    <xsl:for-each select="$node/@*[not(../@json:force-array) or
count(.|../@json:force-array)=2]">
      <json:member>
        <json:name><xsl:if test="$use-badgerfish or
$use-rabbitfish">@</xsl:if><xsl:value-of select="if($use-namespaces) then name()
else local-name()"/></json:name>
        <json:value><xsl:value-of select="."/></json:value>
      </json:member>
    </xsl:for-each>
  </xsl:function>

  <xsl:function name="json:create-namespaces" as="node()*">
    <xsl:param name="node" as="node()"/>
    <xsl:if test="$use-namespaces">
      <xsl:if test="count($node/namespace::*) gt 0">
        <json:member>
          <json:name><xsl:if test="$use-badgerfish or
$use-rabbitfish">@</xsl:if>xmlns</json:name>
          <json:value>
            <json:object>
              <xsl:for-each select="$node/namespace::*">
                <json:member>
                  <xsl:choose>
                    <xsl:when test="local-name(.) eq ''">
                      <json:name>$</json:name>
                    </xsl:when>
                    <xsl:otherwise>
                      <json:name><xsl:value-of select="local-name(.)"/></json:name>
                    </xsl:otherwise>
                  </xsl:choose>
                  <json:value><xsl:value-of select="."/></json:value>
                </json:member>
              </xsl:for-each>
            </json:object>
          </json:value>
        </json:member>
      </xsl:if>
    </xsl:if>
  </xsl:function>

  <!--
    These are output functions that transform the temporary tree
```

```
    to JSON.
  -->
  <xsl:template match="json:parameter" mode="json">
    <xsl:variable name="parameters"><xsl:apply-templates
mode="json"/></xsl:variable>
    <xsl:value-of select="string-join($parameters/parameter, ', ')"/>
  </xsl:template>

  <xsl:template match="json:object" mode="json">
    <xsl:variable name="members"><xsl:apply-templates mode="json"/></xsl:variable>
    <parameter>
      <xsl:text/>{<xsl:text/>
        <xsl:value-of select="string-join($members/member,',')"/>
      <xsl:text/>}<xsl:text/>
    </parameter>
  </xsl:template>

  <xsl:template match="json:member" mode="json">
    <xsl:text/><member><xsl:apply-templates mode="json"/></member><xsl:text/>
  </xsl:template>

  <xsl:function name="json:encode-string" as="xs:string">
    <xsl:param name="string" as="xs:string"/>
    <xsl:sequence select="replace(
          replace(
          replace(
          replace(
          replace(
          replace(
          replace(
          replace(
          replace($string,
            '\\','\\\\'),
            '/', '\\/'),
            '&quot;', '\\&quot;'),
            '&#xA;','\\n'),
            '&#xD;','\\r'),
            '&#x9;','\\t'),
            '\n','\\n'),
            '\r','\\r'),
            '\t','\\t')"/>
  </xsl:function>

  <xsl:template match="json:name" mode="json">
    <xsl:text/>"<xsl:value-of select="json:encode-string(.)"/>":<xsl:text/>
  </xsl:template>

  <xsl:template match="json:value" mode="json">
```

```
    <xsl:choose>
      <xsl:when test="node() and not(text())">
        <xsl:apply-templates mode="json"/>
      </xsl:when>
      <xsl:when test="text()">
        <xsl:choose>
          <!--
              A value is considered a string if the following conditions are met:
               * There is whitespace/formatting around the value of the node.
               * The value is not a valid JSON number (i.e. '01', '+1', '1.', and
'.5' are not valid JSON numbers.)
               * The value does not equal the any of the following strings: 'false',
'true', 'null'.
          -->
          <xsl:when test="normalize-space(.) ne . or not((string(.) castable as
xs:integer  and not(starts-with(string(.),'+')) and not(starts-with(string(.),'0')
and not(. = '0'))) or (string(.) castable as xs:decimal  and
not(starts-with(string(.),'+')) and not(starts-with(.,'-.')) and
not(starts-with(.,'.')) and not(starts-with(.,'-0') and not(starts-with(.,'-0.')))
and not(ends-with(.,'.')) and not(starts-with(.,'0') and not(starts-with(.,'0.')))
)) and not(. = 'false') and not(. = 'true') and not(. = 'null')">
            <xsl:text/>"<xsl:value-of select="json:encode-string(.)"/>"<xsl:text/>
          </xsl:when>
          <xsl:otherwise>
            <xsl:text/><xsl:value-of select="."/><xsl:text/>
          </xsl:otherwise>
        </xsl:choose>
      </xsl:when>
      <xsl:otherwise>
        <xsl:text/>null<xsl:text/>
      </xsl:otherwise>
    </xsl:choose>
  </xsl:template>

  <xsl:template match="json:array-value" mode="json">
    <xsl:text/><value><xsl:apply-templates mode="json"/></value><xsl:text/>
  </xsl:template>

  <xsl:template match="json:array" mode="json">
    <xsl:variable name="values">
      <xsl:apply-templates mode="json"/>
    </xsl:variable>
    <xsl:text/>[<xsl:text/>
      <xsl:value-of select="string-join($values/value,',')"/>
    <xsl:text/>]<xsl:text/>
  </xsl:template>
</xsl:stylesheet>
```