Procedure for Producing JSON Schema from XML Schema

Michael Morris

06/03/2019

Updated

09/14/2020

JSON Schema Procedure

Procedure initiation

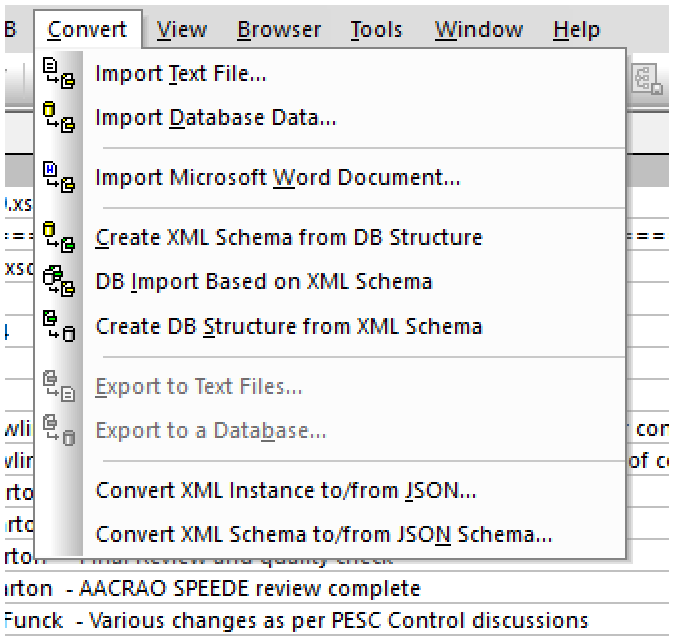
This procedure will start when a new or changed XML standard is approved by the community.

Responsibility

The joint PESC Change Control Board and Technical Advisory Board will be responsible for executing the following procedure.

Procedure Steps

1. The XML Schemas used by the XML standard will be converted to subset schemas using the XSLT in the appendix of this document.
2. A translation tool is used to convert the subset schemas to a single JSON Schema file. The TAB has had success creating a JSON Schema of the College Transcript standard using Altova XMLSpy®:


Convert
View
Browser
Tools
Window
Help
Import Text File...
Import Database Data...
Import Microsoft Word Document...
Create XML Schema from DB Structure
DB Import Based on XML Schema
Create DB Structure from XML Schema
Export to Text Files...
Export to a Database...
Convert XML Instance to/from JSON...
Convert XML Schema to/from JSON Schema...
rton - AACRAO SPEEDE review complete
Funck - Various changes as per PESC Control discussions

3. The namespace prefixes generated by the conversion tool will be removed from the schema document. This will uncover name conflicts that will be resolved manually.

4. The JSON schema will be revised to meet the PESC Compliant JSON standard. This may be accomplished through manual or automated means. Currently, this procedure is not automated.
5. The XML Instance document for the standard will then be converted to JSON instance document. There are several open source and commercial tools that create close but not exact translation consistent with PESC Compliant JSON standard. Currently, the TAB has used Altova XMLSpy® successfully to produce JSON instance documents. However, TAB also has an XSLT that does the conversion. Also, EdExchange can input a document in XML and create JSON output.
6. Any namespace prefixes generated by the conversion tool will be removed from the schema document. This will uncover name conflicts that will be resolved manually.
7. The JSON instance documents will then be modified to meet the JSON standard in the PESC Compliant JSON standard. This may be accomplished by manual or automated means.
8. This JSON instance documents will then be validated by the JSON schema with any discrepancies resolved using the PESC Compliant JSON standard for guidance.
9. Several negative test cases will be added to the instance document to assure rejection of invalid JSON:
    - Added elements
    - Type errors (e.g., a simple value for an array, enumeration errors, etc.)
    - Added attributes (properties that were XML attributes)
10. Once all testing is completed successfully, the JSON schema will be published on the PESC website as an approved standard. It need not go through public comment since it is equivalent to the XML approved standard.

Appendix

XSLT for Creating Subset Schemas

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0" xmlns:pescfn="urn:org:pesc:functions" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:fo="http://www.w3.org/1999/XSL/Format" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:fn="http://www.w3.org/2005/xpath-functions" xmlns:act="http://www.act.org">
	<xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes" name="xml"/>
	<xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes" name="xml"/>
	<!--get main file name and path-->
	<!-- stylesheet assumes that import schema schemaLocations are relative to this path-->
	<xsl:variable name="root-schema" select="tokenize(base-uri(.),'/')[last()]"/>
	<xsl:variable name="path" select="substring-before(base-uri(.),$root-schema)"/>
	<xsl:function name="pescfn:get-schema-prefix" as="xs:string">
		<xsl:param name="prefixlist"/>
		<xsl:param name="schema"/>
		<xsl:variable name="target" select="$schema/@targetNamespace"/>
		<xsl:choose>
			<xsl:when test="fn:exists($schema[$target  = namespace-uri-for-prefix($prefixlist[1],$schema)])">
				<xsl:sequence select="$prefixlist[1]"/>
			</xsl:when>
			<xsl:otherwise>
				<xsl:sequence select="pescfn:get-schema-prefix(fn:remove($prefixlist,1),$schema)"/>
			</xsl:otherwise>
		</xsl:choose>
	</xsl:function>
	<xsl:variable name="file-list" as="item()*">
		<xsl:variable name="list" as="item()*">
			<xsl:call-template name="compose-file-list">
				<xsl:with-param name="current-document" select="/"/>
				<xsl:with-param name="file-location" select="$root-schema"/>
			</xsl:call-template>
		</xsl:variable>
		<xsl:for-each select="distinct-values($list)">
			<xsl:sequence select="."/>
		</xsl:for-each>
	</xsl:variable>
	<xsl:variable name="prefix-list" as="item()*">
		<xsl:variable name="list" as="item()*">
```

```
                    <xsl:call-template name="compose-prefix-list">
                          <xsl:with-param name="current-document" select="/"/>
                    </xsl:call-template>
              </xsl:variable>
              <xsl:for-each select="distinct-values($list)">
                    <xsl:sequence select="."/>
              </xsl:for-each>
        </xsl:variable>
        <xsl:variable name="combined-schema" as="document-node()*">
              <xsl:for-each select="$file-list">
                    <xsl:sequence select="document(concat($path,.))"/>
              </xsl:for-each>
        </xsl:variable>
        <!-- tempate to create sequence of documents -->
        <xsl:template name="compose-file-list">
              <xsl:param name="current-document"/>
              <xsl:param name="file-location"/>
              <xsl:sequence select="$file-location"/>
              <xsl:for-each select="$current-document/xs:schema/xs:import">
                    <xsl:variable name="file-name" select="./@schemaLocation"/>
                    <xsl:variable name="new-document" select="document($file-name)"/>
                    <xsl:call-template name="compose-file-list">
                          <xsl:with-param name="current-document" select="$new-document"/>
                          <xsl:with-param name="file-location" select="$file-name"/>
                    </xsl:call-template>
              </xsl:for-each>
        </xsl:template>
        <!--template compose-prefix-list gets the prefix for the current-document @targetNamespace-->
        <xsl:template name="compose-prefix-list">
              <xsl:param name="current-document"/>
              <xsl:sequence select="pescfn:get-schema-prefix(in-scope-prefixes($current-document/xs:schema),$current-
document/xs:schema)"/>
              <xsl:for-each select="$current-document/xs:schema/xs:import">
                    <xsl:variable name="file-name" select="./@schemaLocation"/>
                    <xsl:variable name="new-document" select="document($file-name)"/>
                    <xsl:call-template name="compose-prefix-list">
                          <xsl:with-param name="current-document" select="$new-document"/>
                    </xsl:call-template>
              </xsl:for-each>
        </xsl:template>
        <!--template process-types iterates through types or groups elements and follows the type or ref declaration to
a new type or group-->
        <xsl:template name="process-types">
                <xsl:param name="type"/>
                <xsl:param name="target-prefix"/>
                <xsl:for-each select="$type//xs:element|$type//xs:group|$type//xs:restriction|$type//xs:extension">
                        <xsl:choose>
                                <xsl:when test="exists(./@type)">
                                        <xsl:variable name="prefix" select="substring-before(./@type,':')"/>
                                        <xsl:variable name="type-name">
                                                <xsl:choose>
                                                        <xsl:when test="$prefix !=''">
                                                                <xsl:value-of select="substring-after(./@type,':')"/>
                                                        </xsl:when>
                                                        <xsl:otherwise>
                                                                <xsl:value-of select="./@type"/>
                                                        </xsl:otherwise>
                                                </xsl:choose>
                                        </xsl:variable>
                                        <xsl:variable name="next-type" select="$combined-schema[index-of($prefix-
list,$prefix)]/xs:schema/element()[@name=$type-name]"/>
                                        <xsl:if test="$prefix=$target-prefix">
                                                <xsl:sequence select="$next-type/@name"/>
                                        </xsl:if>
                                        <xsl:call-template name="process-types">
                                                <xsl:with-param name="type" select="$next-type"/>
                                                <xsl:with-param name="target-prefix" select="$target-prefix"/>
                                        </xsl:call-template>
                                </xsl:when>
                                <!-- de-reference and store elements and groups-->
                                <xsl:when test="exists(./@ref)">
                                        <xsl:variable name="prefix" select="substring-before(./@ref,':')"/>
                                        <xsl:variable name="ref-name">
                                                <xsl:choose>
                                                        <xsl:when test="$prefix !=''">
                                                                <xsl:value-of select="substring-after(./@ref,':')"/>
                                                        </xsl:when>
                                                        <xsl:otherwise>
                                <xsl:value-of select="./@ref"/>
                            </xsl:otherwise>
                        </xsl:choose>
                    </xsl:variable>
                    <xsl:variable name="ref" select="$combined-schema[index-of($prefix-
list,$prefix)]/xs:schema/element()[@name=$ref-name]"/>
                    <xsl:if test="$prefix=$target-prefix">
                        <xsl:sequence select="$ref/@name"/>
                    </xsl:if>
                    <xsl:call-template name="process-types">
                        <xsl:with-param name="type" select="$ref"/>
                        <xsl:with-param name="target-prefix" select="$target-prefix"/>
                    </xsl:call-template>
                </xsl:when>
                <!--look for type in base attribute-->
                <xsl:when test="exists(./@base)">
                    <xsl:variable name="prefix" select="substring-before(./@base,':')"/>
                    <xsl:variable name="base-name">
                        <xsl:choose>
                            <xsl:when test="$prefix !=''">
                                <xsl:value-of select="substring-after(./@base,':')"/>
                            </xsl:when>
                            <xsl:otherwise>
                                <xsl:value-of select="./@base"/>
                            </xsl:otherwise>
                        </xsl:choose>
                    </xsl:variable>
                    <xsl:if test="$prefix!='xs'">
                        <xsl:variable name="base" select="$combined-schema[index-of($prefix-
list,$prefix)]/xs:schema/element()[@name=$base-name]"/>
                        <xsl:if test="$prefix = $target-prefix">
                            <xsl:sequence select="$base/@name"/>
                        </xsl:if>
                        <xsl:call-template name="process-types">
                            <xsl:with-param name="type" select="$base"/>
                            <xsl:with-param name="target-prefix" select="$target-prefix"/>
                        </xsl:call-template>
                    </xsl:if>
                </xsl:when>
                </xsl:choose>
            </xsl:for-each>
        </xsl:template>
        <!-- template for testing variable assignments -->
        <xsl:template name="test">
            <xsl:comment>
                <xsl:for-each select="$file-list">
                    <xsl:value-of select="."/>
                    <xsl:text> | </xsl:text>
                </xsl:for-each>
                <xsl:for-each select="$prefix-list">
                    <xsl:value-of select="."/>
                    <xsl:text> | </xsl:text>
                </xsl:for-each>
                <xsl:for-each select="$combined-schema/xs:schema/@targetNamespace">
                    <xsl:value-of select="."/>
                </xsl:for-each>
            </xsl:comment>
        </xsl:template>
        <!-- start walking the element tree to gather all types and groups for the imported schemas-->
        <xsl:template match="/">
            <xsl:for-each select="$prefix-list[position()>1]">
                <xsl:variable name="target-prefix" select="."/>
                <xsl:variable name="target-schema" select="$combined-schema[index-of($prefix-list,$target-
prefix)]/xs:schema"/>
                <!-- assume that first element is the root-->
                <xsl:variable name="root" select="$combined-schema[1]/xs:schema/xs:element[1]"/>
                <xsl:variable name="subset-schema" as="item()*">
                    <xsl:call-template name="process-types">
                        <xsl:with-param name="type" select="$combined-schema[1]/xs:schema"/>
                        <xsl:with-param name="target-prefix" select="$target-prefix"/>
                    </xsl:call-template>
                </xsl:variable>
                <xsl:result-document href="{concat($target-prefix,'.xsd')}" format="xml">
                    <xsl:element name="xs:schema" namespace="http://www.w3.org/2001/XMLSchema">
                        <xsl:namespace name="{$target-prefix}" select="$combined-schema[index-of($prefix-
list,$target-prefix)]/xs:schema/@targetNamespace"/>
                        <xsl:for-each select="$target-schema/@*">
                            <xsl:copy-of select="."/>
```

```
                                        </xsl:for-each>
                                        <xsl:for-each select="$target-schema/xs:import">
                                              <xsl:copy-of select="." copy-namespaces="yes"/>
                                        </xsl:for-each>
                                        <xsl:for-each select="distinct-values($subset-schema)">
                                              <xsl:variable name="type-name" select="."/>
                                              <xsl:copy-of select="$combined-schema[index-of($prefix-list,$target-
prefix)]/xs:schema/*[@name =$type-name]"/>
                                        </xsl:for-each>
                                  </xsl:element>
                            </xsl:result-document>
                      </xsl:for-each>
                </xsl:template>
          </xsl:stylesheet>
```