

Display the XML Files for Disclosure to Public by Using User-defined XSL

Zhiping Yan, BeiGene, Beijing, China

Huadan Li, BeiGene, Beijing, China

ABSTRACT

US Food and Drug Administration Amendments Act (FDAAA) and European Medicines Agency (EMA) require that clinical trial results should be made publicly available through ClinicalTrials.gov and EU Clinical Trials Register (EU CTR). Previously, clinical trial results were usually prepared based on Clinical Study Report (CSR) and manually entered into regulatory website. Right now, more and more pharmaceutical companies are developing SAS macros to automate the process by generating XML files which can be uploaded to PRS (Protocol Registration and Results System) and EudraCT. There are several papers introduced the automation process and even how to develop SAS macros. However, it's hard to review and verify the XML file. This paper will take Adverse Event section as an example and present how to use user-defined XSL to display XML file in a three-lines table format so that we can view and check XML files against CSR tables.

INTRODUCTION

ClinicalTrials.gov results database was launched in September 2008 in US to implement section 801 of Food and Drug Administration Amendments Act (FDAAA) which requires the submission of “basic results” for certain clinical trials. Public can use this website^[1] to learn about registered clinical trial information. While sponsor can use ClinicalTrials.gov Protocol Registration and Results System (PRS)^[2], a web-based tool provided by ClinicalTrials.gov, to submit clinical study information. Transfer of data into ClinicalTrials.gov can be conducted by either manual entry or XML uploads through PRS web interface. XML file, which may contain protocol registration information and results information, must be created according to PRS XML schemas^[3] to pass validation process of PRS.

Similar to ClinicalTrials.gov, EU Clinical Trials Register (EU CTR)^[4] is also a web-based resource providing a free and accurate search of clinical trials to public. The difference is that public can only view information of clinical trials conducted in European Union (EU) and European Economic Area (EEA). Equivalent of PRS is EudraCT^[5] through which sponsor representatives can post, validate result data sets, and upload summary XML attachments to EudraCT database. EudraCT also provides XML schema and validation rules^[6] to guide sponsor in the creation of XML.

XML stands for Extensible Markup Language and is designed to store and transport data which will be wrapped in tags. Start tag, end tag and data placed between them is called an XML element. A lot of different XML elements are composed of an XML. Tag definition will not be included in XML. Description of tags and XML document structure can be found in XML schema. XML not meeting rules described in XML schema cannot be uploaded successfully into regulatory websites.

XML files can be viewed in all major browsers. However, browsers only display plain XML documents with color-coded elements. It cannot be naturally read by humans. By linking XML with XSL, browsers can display XML files as HTML pages. Thus, it will be easy to review XML and check information to be disclosed against CSR tables.

SAMPLE XML FOLLOWING XML SCHEMA FROM CLINICALTRIALS.GOV AND EUDRACT

Among all result modules (including Baseline Characteristics, End points and Adverse Events, etc.), only Adverse Event module can be uploaded alone and thus is a good example for demonstration here.

With similar approach that has been introduced by Daniel^[7] and Juan^[8], we created sample XML of Adverse Event module for both ClinicalTrials.gov and EudraCT. Figure 1 shows how XML is displayed in web browser. Minus sign symbol and plus sign symbol can be used to collapse and toggle XML children elements. XML elements like frequentAdverseEvents containing large amount of detailed AE information are toggled for a better understanding of whole document structure.

Root element tag of adverse event XML for ClinicalTrials.gov is <reportedEvents>. While EudraCT requires that root node should be <adverseEvents> together with a namespace **aev**. Namespace is used to provide uniquely named elements and attributes in an XML document. All children elements of <aev:adverseEvents> are considered to be in the **aev** namespaces. Besides information like Assessment method, frequency threshold, key information required to be uploaded by both two websites are divided into three parts: treatment group, non-serious adverse event and serious adverse event.

```

ClinicalTrials.gov
<?xml version="1.0" encoding="UTF-8"?>
- <study_collection>
  - <clinical_study partial_upload="true">
    - <id_info>
      <org_name>BeiGene</org_name>
      <org_study_id>BGB-XXXX-XXX</org_study_id>
    </id_info>
    - <result partialUpload="true">
      - <reportedEvents>
        <assessmentType>Non-systematic Assessment</assessmentType>
        <frequencyReportingThreshold>0</frequencyReportingThreshold>
        + <frequentAdverseEvents>
        + <interventionGroups>
        + <seriousAdverseEvents>
        <sourceVocabulary>MedDRA (19.0)</sourceVocabulary>
      </reportedEvents>
    </result>
  </clinical_study>
</study_collection>

EudraCT
<?xml version="1.0" encoding="UTF-8"?>
- <aev:adverseEvents xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:aev="http://eudract.ema.europa.eu/schema/clinical_trial_result/adverse_events">
  <nonSeriousEventFrequencyThreshold>0</nonSeriousEventFrequencyThreshold>
  <timeFrame>From start of study treatment until 30 days after the last study drug intake</timeFrame>
  - <assessmentMethod>
    <value>ADV_EVT_ASSESS_TYPE.non_systematic</value>
  </assessmentMethod>
  - <dictionary>
    <otherName/>
    <version>19.0</version>
    - <name>
      <value>ADV_EVT_DICTIONARY_NAME.meddra</value>
    </name>
  </dictionary>
  + <reportingGroups>
  + <nonSeriousAdverseEvents>
  + <seriousAdverseEvents>
</aev:adverseEvents>
  
```

Figure 1. Sample XML of Adverse Event Module for Both ClinicalTrials.gov and EudraCT

Partial output of toggled elements is shown as following in Figure 2. Detailed tags presented here will be used later to build XSL stylesheet. Let's look at treatment group section at first. Root element of treatment group for ClinicalTrials.gov and EudraCT is <interventionGroups> and <reportingGroups> respectively. Each <interventionGroup> or <reportingGroup> element corresponds to a specific arm group and includes elements to provide information like arm/group title, number of subjects affected by serious adverse events, number of subject exposed to treatment and number of deaths, etc.

```

- <interventionGroups>
  - <interventionGroup id="ReportedEvents-InterventionGroup.0">
    <title>Placebo</title>
    <description>Description for Treatment Group Placebo.</description>
    <numSubjectsSeriousEvents>0</numSubjectsSeriousEvents>
    <numSubjectsFrequentEvents>10</numSubjectsFrequentEvents>
    <partAtRiskFrequentEvents>15</partAtRiskFrequentEvents>
    <partAtRiskSeriousEvents>15</partAtRiskSeriousEvents>
  </interventionGroup>
</interventionGroups>

- <reportingGroups>
  - <reportingGroup id="ReportingGroup-0">
    <title>Placebo</title>
    <description>Description for Treatment Group Placebo.</description>
    <subjectsAffectedByNonSeriousAdverseEvents>10</subjectsAffectedByNonSeriousAdverseEvents>
    <subjectsAffectedBySeriousAdverseEvents>0</subjectsAffectedBySeriousAdverseEvents>
    <subjectsExposed>15</subjectsExposed>
    <deathsAllCauses>0</deathsAllCauses>
    <deathsResultingFromAdverseEvents>0</deathsResultingFromAdverseEvents>
  </reportingGroup>
</reportingGroups>
  
```

Figure 2. Partial Output of Toggled XML Element for Treatment Group

ClinicalTrials.gov uses <frequentAdverseEvents> as root element of non-serious adverse event section while EudraCT applies <nonSeriousAdverseEvents>. <frequentAdverseEvent> or <nonSeriousAdverseEvent> is designed to contain detailed information of an adverse event. <eventStats> or <value> tag captures number of occurrences for adverse event reported in <term> tag as well as number of subjects affected by this adverse event. These two tags will be repeated for each treatment group and number of subjects exposed to each treatment group is also reported.

<pre> - <frequentAdverseEvents> - <frequentEvent> - <adverseEventStats> - <eventStats> <reportingGroupId>ReportedEvents-InterventionGroup.0</reportingGroupId> <numEvents>1</numEvents> <numSubjectsAffected>1</numSubjectsAffected> <numSubjects>15</numSubjects> </eventStats> - <eventStats> <reportingGroupId>ReportedEvents-InterventionGroup.1</reportingGroupId> <numEvents>1</numEvents> <numSubjectsAffected>1</numSubjectsAffected> <numSubjects>15</numSubjects> </eventStats> </adverseEventStats> <assessmentType>Non-systematic Assessment</assessmentType> <organSystemName>Gastrointestinal disorders</organSystemName> <sourceVocabulary>MedDRA (19.0)</sourceVocabulary> <term>Abdominal pain</term> </frequentEvent> </frequentAdverseEvents> </pre> <p style="text-align: center;">ClinicalTrials.gov</p>	<pre> - <nonSeriousAdverseEvents> - <nonSeriousAdverseEvent> <term>Abdominal pain</term> - <organSystem> <eutctId>10000004856</eutctId> <version>19</version> </organSystem> <dictionaryOverridden>false</dictionaryOverridden> - <values> - <value reportingGroupId="ReportingGroup-0"> <occurrences>1</occurrences> <subjectsAffected>1</subjectsAffected> <subjectsExposed>15</subjectsExposed> </value> - <value reportingGroupId="ReportingGroup-1"> <occurrences>1</occurrences> <subjectsAffected>1</subjectsAffected> <subjectsExposed>15</subjectsExposed> </value> </values> </nonSeriousAdverseEvent> </nonSeriousAdverseEvents> </pre> <p style="text-align: center;">EudraCT</p>
---	---

Figure 3. Partial Output of Toggled XML Element for Non-serious Adverse Event

Serious adverse event section is similar to non-serious adverse event section and will not be described in details in this paper.

<pre> - <seriousAdverseEvents> - <seriousEvent> - <adverseEventStats> - <eventStats> <reportingGroupId>ReportedEvents-InterventionGroup.0</reportingGroupId> <numEvents>0</numEvents> <numSubjectsAffected>0</numSubjectsAffected> <numSubjects>15</numSubjects> </eventStats> - <eventStats> <reportingGroupId>ReportedEvents-InterventionGroup.1</reportingGroupId> <numEvents>1</numEvents> <numSubjectsAffected>1</numSubjectsAffected> <numSubjects>15</numSubjects> </eventStats> </adverseEventStats> <assessmentType>Non-systematic Assessment</assessmentType> <organSystemName>Skin and subcutaneous tissue disorders</organSystemName> <sourceVocabulary>MedDRA (19.0)</sourceVocabulary> <term>Erythema multiforme</term> </seriousEvent> </seriousAdverseEvents> </pre> <p style="text-align: center;">ClinicalTrials.gov</p>	<pre> - <seriousAdverseEvents> - <seriousAdverseEvent> <term>Erythema multiforme</term> - <organSystem> <eutctId>10000004858</eutctId> <version>19</version> </organSystem> <dictionaryOverridden>false</dictionaryOverridden> - <values> - <value reportingGroupId="ReportingGroup-0"> <occurrences>0</occurrences> <subjectsAffected>0</subjectsAffected> <subjectsExposed>15</subjectsExposed> <occurrencesCausallyRelatedToTreatment>0</occurrencesCausallyRelatedToTreatment> - <fatalities> <deaths>0</deaths> <deathsCausallyRelatedToTreatment>0</deathsCausallyRelatedToTreatment> </fatalities> </value> - <value reportingGroupId="ReportingGroup-1"> <occurrences>1</occurrences> <subjectsAffected>1</subjectsAffected> <subjectsExposed>15</subjectsExposed> <occurrencesCausallyRelatedToTreatment>1</occurrencesCausallyRelatedToTreatment> - <fatalities> <deaths>0</deaths> <deathsCausallyRelatedToTreatment>0</deathsCausallyRelatedToTreatment> </fatalities> </value> </values> </seriousAdverseEvent> </seriousAdverseEvents> </pre> <p style="text-align: center;">EudraCT</p>
--	--

Figure 4. Partial Output of Toggled XML Element for Serious Adverse Event

XSL SYNTAX

XSL stands for Extensible Stylesheet Language and describes how the XML elements should be displayed. With XSL, you can add, remove, rearrange and sort XML elements. With following XSL basic knowledge, this paper will present how to create a simple user-defined stylesheet for above sample XML of EudraCT version.

<xsl:stylesheet>

Though XSL is intended to transform XML, it is also an XML-based language which consists of different elements. The root element that declares the document to be an XSL stylesheet is <xsl:stylesheet>. Prefix "aev" was contained in XML uploaded into EudraCT and corresponding namespace declaration must be included as below in <xsl:stylesheet> root element.

```

<xsl:stylesheet xmlns:aev="http://eudract.ema.europa.eu/schema/clinical_trial_result/adverse_events"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">

```

<xsl:value-of select="">

<xsl:value-of> element can be used to extract value of an XML element and add it to the output stream of the transformation. XPath expression in select attribute is like the path expression which we use with computer file system. It helps navigating XML file system. For example, "aev:adverseEvents/timeFrame" tells XSL to locate parent node - "aev:adverseEvents" - and then go to the second childnode - "timeFrame".

HTML elements can be used to present value selected by <xsl:value-of>. <html> element is the root element of an HTML page. <body> element contains visible page content. <table> tag defines an HTML table. Each table row is defined with <tr> tag while a table cell is defined with <td> tag.

By linking following stylesheet with XML for EudraCT (Figure 1), browser will present readers a table with two rows. The first row contains information of time frame while the second row includes threshold.

```

<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:aev="http://eudract.ema.europa.eu/schema/clinical_trial_result/adverse_events"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:template match="/">
    <html>
      <body>
        <h2>Adverse events information</h2>
        <table border="1">
          <tr>
            <td>Timeframe for adverse event reporting</td>
            <td><xsl:value-of select="aev:adverseEvents/timeFrame"/></td>
          </tr>
          <tr>
            <td>Threshold</td>
            <td><xsl:value-of select="aev:adverseEvents/nonSeriousEventFrequencyThreshold"/></td>
          </tr>
        </table>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>

```

Stylesheet

Adverse events information Result HTML Output

Timeframe for adverse event reporting	From start of study treatment until 30 days after the last study drug intake
Threshold	0

Figure 5. Example to Demonstrate <xsl:value-of>

<xsl:for-each select="">

Similar to Do loop in SAS, <xsl:for-each> element can be used to loop through every XML elements of a specific node. Look at Figure 6, "aev:adverseEvents/reportingGroups/reporting Group" in select attribute means that XSL will display information of every reportingGroup element within reportingGroups element. One <tr> element and four <td> elements indicate that extracted information of each reporting group is placed in one row which consists of four cells. <thead> tag represents table header section and <th> tag represents a header cell.

```

<table border="1">
  <thead>
    <tr>
      <th width="12%">Title</th>
      <th width="40%">Description</th>
      <th width="12%">Subject Affected <br> By Non-Serious AE</br></th>
      <th width="12%">Subject <br> Exposed</br></th>
    </tr>
  </thead>
  <tbody>
    <xsl:for-each select="aev:adverseEvents/reportingGroups/reportingGroup">
      <tr>
        <td><xsl:value-of select="title"/></td>
        <td><xsl:value-of select="description"/></td>
        <td><xsl:value-of select="subjectsAffectedByNonSeriousAdverseEvents"/></td>
        <td><xsl:value-of select="subjectsExposed"/></td>
      </tr>
    </xsl:for-each>
  </tbody>
</table>

```

Stylesheet

Result HTML Output

Title	Description	Subject Affected By Non-Serious AE	Subject Exposed
Placebo	Description for Treatment Group Placebo.	10	15
Study Drug	Description for Treatment Group Study Drug.	13	15

Figure 6. Example to Demonstrate <xsl:for-each>

<xsl:choose>, <xsl: when test=""> and <xsl:otherwise>

Like SELECT-WHEN/OTHEWISE conditions in SAS DATA step, conjunction of <xsl:choose>, <xsl:when>, <xsl:otherwise> elements allow us to perform multiple conditional test against XML file. <xsl:otherwise> follows one or more <xsl:when>. If <xsl:when> is true, the content of <xsl:when> is processed. If no <xsl:when> is true, the content of <xsl:otherwise> is processed. With this feature, XML can be displayed with alternative shading as showed in Figure 7.

Organ System	Preferred Term	Placebo (N=15)		Study Drug (N=15)	
		Number of AEs	Number of Subjects	Number of AEs	Number of Subjects
100000004856	Abdominal pain	1	1 (6.7%)	1	1 (6.7%)
100000004849	Atrial fibrillation	1	1 (6.7%)	0	0 (0.0%)
100000004848	Blood alkaline phosphatase increased	0	0 (0.0%)	2	1 (6.7%)
100000004859	Bone pain	1	1 (6.7%)	1	1 (6.7%)
100000004856	Cheilitis	0	0 (0.0%)	1	1 (6.7%)
.....					
100000004867	Mucosal inflammation	2	2 (13.3%)	0	0 (0.0%)
100000004859	Myalgia	2	2 (13.3%)	4	3 (20.0%)
100000004855	Nasal congestion	1	1 (6.7%)	0	0 (0.0%)
100000004867	Non-cardiac chest pain	1	1 (6.7%)	0	0 (0.0%)
100000004867	Oedema	1	1 (6.7%)	0	0 (0.0%)
100000004855	Oropharyngeal pain	1	1 (6.7%)	0	0 (0.0%)
100000004859	Pain in extremity	1	1 (6.7%)	0	0 (0.0%)

Figure 7. Partial Output of Non-Serious Adverse Events with Alternative Shading Rows

The attribute style = "background-color:#xxx" in <td> can be used to set background color of table cells. By performing test "position() mod 2 != 0" in <xsl:when>, we can apply gray background color to odd rows only. Even rows not meeting test in <xsl:when> will be processed in <xsl:otherwise> to have no background color.

```

- <tbody>
  - <xsl:for-each
    select="aev:adverseEvents/nonSeriousAdverseEvents/nonSeriousAdverseEvent">
    - <xsl:choose>
      - <xsl:when test="position() mod 2 != 0">
        - <tr>
          - <td style="background-color: #eee">
            <xsl:value-of select="organSystem/eutctId"/>
          </td>
          .....
        </tr>
      </xsl:when>
      - <xsl:otherwise>
        - <tr>
          - <td style="background-color: #fff">
            <xsl:value-of select="organSystem/eutctId"/>
          </td>
          .....
        </tr>
      </xsl:otherwise>
    </xsl:choose>
  </xsl:for-each>
</tbody>

```

Figure 8. <xsl:when> and <xsl:otherwise> to Enable Alternative Shading

<xsl:variable select="">

The <xsl:variable> element can be used to declare a variable. Name attribute specifies the name of user-declared variable while select attribute defines the value of the variable. In Figure 9, variable subjectsAffected represents

number of subjects having a given preferred term. While variable subjectsExposed is total number of subjects from a treatment group. These two declared variables can be used to calculate percentage and construct strings like xx (xx.x%) with format-number and concat functions. When using declared variables, “\$” should be added as a prefix to indicate it is variable.

```

- <tbody>
- <xsl:for-each
  select="aev:adverseEvents/nonSeriousAdverseEvents/nonSeriousAdverseEvent">
- <xsl:choose>
- <xsl:when test="position() mod 2 != 0">
- <tr>
- <td style="background-color: #eee">
  <xsl:value-of select="organSystem/eutctId"/>
</td>
- <td style="background-color: #eee">
  <xsl:value-of select="term"/>
</td>
- <xsl:for-each select="values/value">
  <xsl:variable select="subjectsAffected"
    name="subjectsAffected"/>
  <xsl:variable select="subjectsExposed"
    name="subjectsExposed"/>
- <td style="background-color: #eee">
  <xsl:value-of select="occurrences"/>
</td>
- <td style="background-color: #eee">
  <xsl:value-of select="concat(subjectsAffected,' (' ,string
    (format-number(($subjectsAffected div
      $subjectsExposed * 100),'#0.0'),'%)')"/>
</td>
</xsl:for-each>
</tr>
</xsl:when>

```

Figure 9. <xsl:variable> to Perform Calculation

<xsl:template match="" mode=""> and <xsl:apply-template select="" mode="">

<xsl:template> defines rules applied to a specified node matching the value of match attribute. The value of the match attribute is an XPath expression. For example, match = "aev:adverseEvents/nonSeriousAdverseEvents" in Figure 10 associates rules defined in second template with NonSeriousAdverseEvents element and all of its child elements. Match = "/" defines the whole document.

<xsl:apply-templates> can apply rules defined in <xsl:template> to a node and corresponding child notes. If we add a select attribute to the <xsl:apply-templates> element, it will process only the child element that matches the value of the select attribute.

Generally speaking, <xsl:template> likes macro development while <xsl:apply-templates> is similar to calling a macro in SAS macro language. And to make it work, value in match attribute and value in select attribute should be the same. If there are multiple templates defined for the same element, mode attribute can be used to distinguish among them. Values of mode attribute in <xsl:template> and <xsl:apply-attribute> should also be the same.

<xsl:template> and <xsl:apply-template> will not trigger error if no node matching value of match attribute or select attribute. They just will not be processed. Therefore, we can put different <xsl:template> and <xsl:apply-template> together to build one XSL file which later can transform XML files following different XML schema. This is the reason why we can develop one XSL that can transform XML for both ClinicalTrials.gov and EudraCT.

In Figure 10, <xsl:template match="aev:adverseEvents/reportingGroups"> and <xsl:template match="aev:adverseEvents/nonSeriousAdverseEvents"> are defined for EudraCT version. The first one is to build table header while the later one is to construct table body. They are applied in <thead> and <tbody> tag, respectively, to build whole table showed in Figure 7. The remaining two <xsl:template> and <xsl:apply-template> are defined for ClinicalTrials version. They are responsible for transforming XML of ClinicalTrials version into output as that showed in Figure 7.

```

- <xsl:template match="/">
- <html>
  <!--Define stylesheet for table output-->
  + <head>
    <!--Output HTML table-->
  - <body>
    + <p>
    - <table>
      <!--Apply Template for Table Header-->
      - <thead>
        <!--EudraCT-->
        <xsl:apply-templates select="aev:adverseEvents/reportingGroups"/>
        <!--ClinicalTrials-->
        <xsl:apply-templates
          select="study_collection/clinical_study/result/reportedEvents/interventionGroups"/>
        </thead>
      <!--Apply Template for Table Body-->
      - <tbody>
        <!--EudraCT-->
        <xsl:apply-templates select="aev:adverseEvents/nonSeriousAdverseEvents"/>
        <!--ClinicalTrials-->
        <xsl:apply-templates
          select="study_collection/clinical_study/result/reportedEvents/frequentAdverseEvents"/>
        </tbody>
      </table>
    </body>
  </html>
</xsl:template>
<!--Template of Table Header for EudraCT version-->
+ <xsl:template match="aev:adverseEvents/reportingGroups">
  <!--Template of Table Body: Non-Serious Adverse Event for EudraCT version-->
+ <xsl:template match="aev:adverseEvents/nonSeriousAdverseEvents">
  <!--Template of Table Header for ClinicalTrials version-->
+ <xsl:template match="study_collection/clinical_study/result/reportedEvents/interventionGroups">
  <!--Template of Table Body: Non-Serious Adverse Event for ClinicalTrials version-->
+ <xsl:template match="study_collection/clinical_study/result/reportedEvents/frequentAdverseEvents">

```

Figure 10. <xsl:template> and <xsl:apply-template> to build XSL for both EudraCT and ClinicalTrials

SAMPLE XSL

Figure 11 shows corresponding XSL that can transform XML into table showed in Figure 7 for a better understanding.

```

<?xml version="1.0" encoding="UTF-8"?>
- <xsl:stylesheet xmlns:aev="http://eudract.ema.europa.eu/schema/clinical_trial_result/adverse_events"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  - <xsl:template match="/">
    - <html>
      <!--Define stylesheet for table output-->
      + <head>
        <!--Output HTML table-->
      - <body>
        + <p>
        - <table>
          <!--Apply Template for Table Header-->
          - <thead>
            <!--EudraCT-->
            <xsl:apply-templates select="aev:adverseEvents/reportingGroups"/>
            <!--ClinicalTrials-->
            <xsl:apply-templates
              select="study_collection/clinical_study/result/reportedEvents/interventionGroups"/>
            </thead>
          <!--Apply Template for Table Body-->
          - <tbody>
            <!--EudraCT-->
            <xsl:apply-templates select="aev:adverseEvents/nonSeriousAdverseEvents"/>
            <!--ClinicalTrials-->
            <xsl:apply-templates
              select="study_collection/clinical_study/result/reportedEvents/frequentAdverseEvents"/>
            </tbody>
          </table>
        </body>
      </html>
    </xsl:template>

```

```

    <!--Template of Table Header for EudraCT version-->
- <xsl:template match="aev:adverseEvents/reportingGroups">
- <tr>
    <th/>
    <th/>
    - <xsl:for-each select="reportingGroup">
      - <th align="center" colspan="2">
        <xsl:value-of select="title"/>
      </th>
      <th width="1px"/>
    </xsl:for-each>
  </tr>
- <tr>
    <th/>
    <th/>
    - <xsl:for-each select="reportingGroup">
      <xsl:variable select="subjectsExposed" name="subjectsExposed"/>
      - <th colspan="2" class="bot">
        <xsl:value-of select="concat('(N=',subjectsExposed,')')"/>
      </th>
      <th width="1px"/>
    </xsl:for-each>
  </tr>
- <tr>
    <th class="trm">Organ System</th>
    <th class="trm">Preferred Term</th>
    - <xsl:for-each select="reportingGroup">
      <th class="lef">Number of AEs</th>
      <th class="lef">Number of Subjects</th>
      <th width="1px" class="lef"/>
    </xsl:for-each>
  </tr>
</xsl:template>
    <!--Template of Table Body: Non-Serious Adverse Event for EudraCT version-->
- <xsl:template match="aev:adverseEvents/nonSeriousAdverseEvents">
- <xsl:for-each select="nonSeriousAdverseEvent">
- <xsl:choose>
  - <xsl:when test="position() mod 2 != 0">
    - <tr>
      - <td class="bg1">
        <xsl:value-of select="organSystem/eutctId"/>
      </td>
      - <td class="bg1">
        <xsl:value-of select="term"/>
      </td>
      - <xsl:for-each select="values/value">
        <xsl:variable select="subjectsAffected" name="subjectsAffected"/>
        <xsl:variable select="subjectsExposed" name="subjectsExposed"/>
        - <td class="bg1">
          <xsl:value-of select="occurrences"/>
        </td>
        - <td class="bg1">
          <xsl:value-of select="concat(subjectsAffected,'(',string(format-number
            (($subjectsAffected div $subjectsExposed * 100),'#0.0'),'%'))')"/>
        </td>
        <td width="1px" class="bg1"/>
      </xsl:for-each>
    </tr>
  </xsl:when>
  - <xsl:otherwise>
    - <tr>
      - <td class="bg2">
        <xsl:value-of select="organSystem/eutctId"/>
      </td>
      - <td class="bg2">
        <xsl:value-of select="term"/>
      </td>
      - <xsl:for-each select="values/value">
        <xsl:variable select="subjectsAffected" name="subjectsAffected"/>
        <xsl:variable select="subjectsExposed" name="subjectsExposed"/>
        - <td class="bg2">
          <xsl:value-of select="occurrences"/>
        </td>
      </xsl:for-each>
    </tr>
  </xsl:otherwise>
</xsl:choose>
</xsl:for-each>

```

```

- <td class="bg2">
  <xsl:value-of select="concat(subjectsAffected,' (',string(format-number
    (($subjectsAffected div $subjectsExposed * 100),'##0.0'),'%)')"/>
  </td>
  <td width="1px" class="bg2"/>
</xsl:for-each>
</tr>
</xsl:otherwise>
</xsl:choose>
</xsl:for-each>
</xsl:template>
<!--Template of Table Header for ClinicalTrials version-->
- <xsl:template match="study_collection/clinical_study/result/reportedEvents/interventionGroups">
- <tr>
  <th/>
  <th/>
  - <xsl:for-each select="interventionGroup">
    - <th align="center" colspan="2">
      <xsl:value-of select="title"/>
    </th>
    <th width="1px"/>
  </xsl:for-each>
</tr>
- <tr>
  <th/>
  <th/>
  - <xsl:for-each select="../frequentAdverseEvents/frequentEvent[1]/adverseEventStats/eventStats">
    <xsl:variable select="numSubjects" name="numSubjects"/>
    - <th colspan="2" class="bot">
      <xsl:value-of select="concat('(N=',numSubjects,')')"/>
    </th>
    <th width="1px"/>
  </xsl:for-each>
</tr>
- <tr>
  <th class="trm">Organ System</th>
  <th class="trm">Preferred Term</th>
  - <xsl:for-each select="interventionGroup">
    <th class="lef">Number of AEs</th>
    <th class="lef">Number of Subjects</th>
    <th width="1px" class="lef"/>
  </xsl:for-each>
</tr>
</xsl:template>
<!--Template of Table Body: Non-Serious Adverse Event for ClinicalTrials version-->
- <xsl:template match="study_collection/clinical_study/result/reportedEvents/frequentAdverseEvents">
- <xsl:for-each select="frequentEvent">
  - <xsl:choose>
    - <xsl:when test="position() mod 2 != 0">
      - <tr>
        - <td class="bg1">
          <xsl:value-of select="organSystemName"/>
        </td>
        - <td class="bg1">
          <xsl:value-of select="term"/>
        </td>
        - <xsl:for-each select="adverseEventStats/eventStats">
          <xsl:variable select="numSubjectsAffected" name="numSubjectsAffected"/>
          <xsl:variable select="numSubjects" name="numSubjects"/>
          - <td class="bg1">
            <xsl:value-of select="numEvents"/>
          </td>
          - <td class="bg1">
            <xsl:value-of select="concat(numSubjectsAffected,' (',string(format-number
              (($numSubjectsAffected div $numSubjects * 100),'##0.0'),'%)')"/>
          </td>
          <td width="1px" class="bg1"/>
        </xsl:for-each>
      </tr>
    </xsl:when>
    - <xsl:otherwise>
      - <tr>
        - <td class="bg2">

```

```
        <xsl:value-of select="organSystemName"/>
      </td>
    - <td class="bg2">
      <xsl:value-of select="term"/>
    </td>
    - <xsl:for-each select="adverseEventStats/eventStats">
      <xsl:variable select="numSubjectsAffected" name="numSubjectsAffected"/>
      <xsl:variable select="numSubjects" name="numSubjects"/>
      - <td class="bg2">
        <xsl:value-of select="numEvents"/>
      </td>
      - <td class="bg2">
        <xsl:value-of select="concat(numSubjectsAffected,' (' ,string(format-number
          (($numSubjectsAffected div $numSubjects * 100),'#0.0'),'%'))"/>
      </td>
      <td width="1px" class="bg2"/>
    </xsl:for-each>
  </tr>
</xsl:otherwise>
</xsl:choose>
</xsl:for-each>
</xsl:template>
</xsl:stylesheet>
```

Figure 11. Whole XSL that can transform XML into table showed in Figure 7

HOW TO LINK UP XML WITH XSL

Suppose that full name of XSL is aexml.xsl. XML file can be linked up with XSL by adding `<?xml-stylesheet type="text/xsl" href="aexml.xsl"?>` after `<?xml version="1.0" encoding="UTF-8"?>`. XSL file and XML file should be placed in the same folder.

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="aexml.xsl"?>
```

CONCLUSION

We have successfully created XSL file that can transform XML for both EudraCT and ClinicalTrials into three lines table. Only elements matching value in select/match attribute can be processed and displayed in browser. By applying different templates, we can choose to display different data even in different format. Perhaps in future, we only need to submit XML and regulatories can transform XML data into different tables with different XSL files.

REFERENCES

- [1] ClinicalTrials.gov: <https://clinicaltrials.gov/>
- [2] Protocol Registration and Results System (PRS): <https://prsinfo.clinicaltrials.gov/>
- [3] PRS XML schemas: <https://prsinfo.clinicaltrials.gov/prs-xml-schemas.html>
- [4] EU Clinical Trials Register: <https://www.clinicaltrialsregister.eu/ctr-search/search>
- [5] EudraCT: <https://eudract.ema.europa.eu/eudract-web/index.faces>
- [6] EudraCT Results related documentation: <https://eudract.ema.europa.eu/result.html>
- [7] Daniel Boisvert, Andy Illidge, Troy Ruth. ClinicalTrials.gov: Creating an XML upload from SAS, NESUG 2010
- [8] Juan Wu, Min Lai. Experience of Generating an XML file and Uploading Serious and Frequent Adverse Events to ClinicalTrials.gov, PharmaSUG 2012

ACKNOWLEDGMENTS

I would like to thank Yashan Zhou (Covance) who gave good suggestions in the development of XSL.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Name: Zhiping Yan

Enterprise: BeiGene
E-mail: zhiping.yan@beigene.com

Name: Huadan Li
Enterprise: BeiGene
E-mail: huadan.li@beigene.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.