SESUG Paper 52-2025

Creating DOCX Tables and Listings using Office Open XML from SAS® Data Null

Eric Qi, Suhas R. Sanjee, Merck & Co., Inc., Rahway, NJ, USA

ABSTRACT

Currently we rely heavily on RTF technology to create tables and listings. Microsoft has transitioned away from RTF in favor of the XML-based Office Open XML (DOCX) format as the default standard for its Office suite. DOCX is based on the Office Open XML standard, which is a modern, internationally-recognized standard for Office documents. DOCX format offers better long-term compatibility and interoperability with other modern Office productivity software. The DOCX format also offers superior support for structured documents. SAS can support generation of DOCX files using ODS. However, this approach doesn't meet all the requirements of our stakeholders. It lacks the level of customization needed to accurately control page breaks and create non-editable tables, which are necessary features to prevent accidental edits. In this paper, we describe an efficient approach using Office Open XML framework, SAS macros and Linux commands to create non-editable tables in DOCX format. A DOCX file consists of multiple XML files compressed into a single package. Using SAS, we can create an XML file with content extracted from a dataset, which can then be combined with other parts of the DOCX package to produce a complete DOCX file. This not only modernizes our table creation programs but also eliminates the current complex conversion step required to render RTF tables non-editable.

1. INTRODUCTION

For many years, the pharmaceutical industry has relied extensively on Rich Text Format (RTF) to generate tables and listings for clinical study reports submitted to regulatory authorities as part of drug and vaccine approval applications. Introduced by Microsoft in 1987, RTF provided a robust solution for cross-platform document exchange with basic formatting capabilities. However, as Microsoft shifted its focus to the XML-based Office Open XML - DOCX format with the release of Office 2007, RTF has become a legacy technology, lacking the advanced features such as locking the table to prevent accidental editing and interoperability demanded by modern workflows. DOCX, built on the internationally recognized Office Open XML (OOXML) standard, offers superior support for structured documents, smaller file sizes, and compatibility with a wide range of office productivity software, making it the preferred format for professional documents.

SAS, a leading tool for clinical trial analysis & reporting (A&R), supports DOCX generation through its Output Delivery System (ODS). While ODS provides a convenient way to export SAS output to DOCX, it often lacks the granular control required for clinical trial A&R. Key limitations include inadequate handling of page breaks and the inability to create non-editable tables which is a feature critical for preventing unintentional modifications in analysis results once the tables, listings and figures (TLF) are used within a clinical study report (CSR). To address these gaps, this paper introduces an innovative approach that utilizes SAS DATA _NULL_ to generate Office Open XML content directly from SAS datasets. By creating document.xml files and integrating them with predefined OOXML style files, we can produce fully customized, non-editable tables and listings in DOCX format. This method leverages SAS macros and Linux commands to streamline the process, eliminating the need for complex RTF conversion steps while meeting stringent formatting and protection requirements. Through a practical example—an adverse event summary table—we illustrate how this approach enhances flexibility, ensures compliance with business needs, and aligns with the modern DOCX standard.

2. OVERVIEW OF OOXML FRAMEWORK

XML is a markup language designed to store and transport data in a structured, human-readable, and machine-readable format. Office Open XML (OOXML) is a family of XML-based file formatting standards developed by Microsoft. OOXML defines how to store and structure office documents (Word documents, spreadsheets, presentations) in a compressed, XML-based format.

OOXML includes a specific schema for Word documents called WordprocessingML (WPML) that defines how tables are structured in a DOCX file. A DOCX file is a ZIP archive containing multiple XML files. Inside the DOCX schema, the main content including tables being in word/document.xml.

WPML focuses on word-processing documents. It defines the XML vocabulary for Word documents, including text, formatting, tables, and more. It offers detailed control over tables (e.g., cell merging, borders, alignment). It is more verbose than generic XML but enables rich formatting. It uses w: to indicate WPML elements as shown in Table 1. WPML provides the detailed schema that Microsoft Word uses to interpret the contents of a DOCX file. It is the language used in document.xml and related XML files such as styles.xml inside a DOCX ZIP archive.

In terms of how they work together for tables, XML foundation provides the syntax for structured data (e.g., <table><row><cell>). OOXML standard defines the DOCX file format as a ZIP archive with XML files, using WPML for content. WPML implementation specifies the exact tags and attributes as shown in Table 1 to create and style tables in a DOCX file.

Table 1: Commonly used WPML tags

WPML Tags	Description
<w:tbl>	Defines a table
<w:tblPr>	Table properties (e.g., width, borders).
<w:tr>	Table row.
<w:tc>	Table cell.
<w:p>	Paragraph inside a cell (required in OOXML for text).
<w:r>	Run (a sequence of text with the same formatting).
<w:t>	Text content.
<w:tblStyle>	Applies a Word table style (e.g., “TableGrid”).
<w:tblGrid>	Defines column widths.
<w:tcPr>	Cell properties (e.g., width, shading).
<w:pPr>	Paragraph properties (e.g., alignment).
<w:rPr>	Run properties (e.g., bold text).

DOCX FORMAT

DOCX is a file format used by Microsoft Word for storing documents. a standard introduced with Microsoft Office 2007 and used in all later versions. It offers improved features like smaller file sizes, better data recovery, and compatibility with modern standards.

KEY FEATURES OF DOCX

DOCX is supported by most modern word processing software such as Word, Google Docs and LibreOffice. DOCX offers robust support for tables, figures, multilevel headings, footnotes, hyperlinks, and VBA macros. This is ideal for complex documents such as manuscripts and CSRs which are typically used to present analysis results from clinical trials. SAS also has the capability to generate reports in DOCX using ODS but doesn’t offer the required flexibility to meet all business needs for reporting of clinical trial data.

A .DOCX ZIP archive schema contains multiple XML files as shown in Figure 1:

- document.xml: The content of the document.
- styles.xml: Defines the styles (e.g., fonts, table formatting).
- numbering.xml: Defines numbering schemes (if needed).

- [Content_Types].xml: Defines content types.
- _rels/. rels: Defines relationships between files.

In addition, there are other XML files, depending on the complexity of the document.

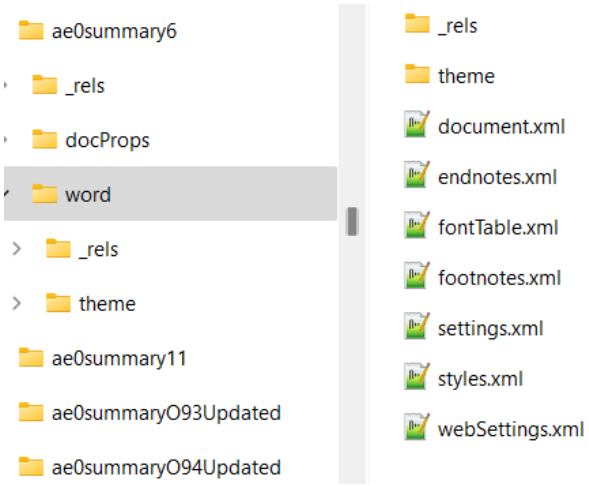

Figure 1: Schema of DOCX ZIP Archive

Simplified WPML content for a table in document.xml is as shown below where you can see how everything comes together to create a simple table (Table 2).

```
<w:tbl>
    <w:tblPr>
        <w:tblStyle w:val="TableGrid"/> <!-- Apply a predefined style -->
        <w:tblW w:w="0" w:type="auto"/> <!-- Auto-width -->
        <w:tblBorders>
            <w:top w:val="single" w:sz="4"/> <!-- Border style -->
        </w:tblBorders>
    </w:tblPr>
    <w:tblGrid>
        <w:gridCol w:w="2000"/> <!-- Column width -->
        <w:gridCol w:w="2000"/>
    </w:tblGrid>
    <w:tr>
        <w:tc>
            <w:tcPr>
                <w:tcW w:w="2000" w:type="dxa"/> <!-- Cell width -->
                <w:shd w:val="solid" w:fill="D3D3D3"/> <!-- Cell shading -->
            </w:tcPr>
            <w:p>
                <w:pPr>
                    <w:jc w:val="center"/> <!-- Center-align text -->
                </w:pPr>
                <w:r>
                    <w:rPr>
                        <w:b/> <!-- Bold -->
                    </w:rPr>
                    <w:t>Header 1</w:t>
                </w:r>
            </w:p>
        </w:tc>
        <w:tc>
            <w:tcPr>
                <w:tcW w:w="2000" w:type="dxa"/>
            </w:tcPr>
            <w:p><w:r><w:t>Header 2</w:t></w:r></w:p>
        </w:tc>
    </w:tr>
    <w:tr>
        <w:tc>
```

```
            <w:tcPr>
                <w:tcW w:w="2000" w:type="dxa"/>
            </w:tcPr>
            <w:p><w:r><w:t>Data 1</w:t></w:r></w:p>
        </w:tc>
        <w:tc>
            <w:tcPr>
                <w:tcW w:w="2000" w:type="dxa"/>
            </w:tcPr>
            <w:p><w:r><w:t>Data 2</w:t></w:r></w:p>
        </w:tc>
    </w:tr>
</w:tbl>
```

The when embedded in DOCX schema and opened using MS Word, the table looks as shown in Table 2.

Table 2: Simple table rendering of the WPML content in document.xml

Header 1	Header 2
Data 1	Data 2

3. TECHNICAL APPROACH

We evaluated two approaches for generating TLF in DOCX format using SAS. The first approach utilized ODS WORD, a feature available in SAS for creating DOCX files. The second approach involved generating XML files that conform to the Office Open XML (OOXML) structure, which are then packaged as DOCX documents.

We decided against using SAS ODS WORD for two main reasons:

1. It does not support the complex table formatting required by our internal standards.
2. ODS WORD is currently in pre-production and not fully supported.

Instead, we adopted an approach using SAS DATA NULL to generate XML files, similar to the method used for generating RTF tables in SAS. This approach allows for greater control over the output and formatting. This methodology is based on techniques described in a 2003 PharmaSUG proceedings paper published by our department [1].

Creation of DOCX files using XML as a backbone involves the following steps:

1. Generation of XML Content: Creation of OOXML compliant XML files that represent the content and structure of the Word document using SAS DATA _NULL.
2. Combining the main content i.e. document.xml with predefined XML files based on DOCX file schema into a .docx file using Linux commans.

These steps can be implemented entirely within SAS in conjunction with Linux commands for ZIP creation and file/folder operations.

3.1 GENERATION OF XML CONTENT USING SAS DATA _NULL_

The DATA _NULL_ step in SAS is a special type of DATA step that is used when you want to process data or generate output without creating a new SAS dataset. It is particularly useful for tasks like writing custom reports, generating files in ASCII formats or for performing data manipulations where the goal is not to store the results in a dataset but to produce some other form of output. It is often used with the FILE and PUT statements for creating ASCII files. This allows full control over the output format, making it ideal for generating structured files in ASCII such as XML. Since no dataset is created, DATA _NULL_ can be more memory-efficient for certain tasks, especially when dealing with large datasets where intermediate storage is not necessary.

Below is an example code snippet to demonstrate the use of DATA _NULL_ to create an XML file representing the content of a simple adverse event (AE) summary table, which can later be combined with other XML files to create a .docx file.

```
/* Use DATA _NULL_ to generate document.xml */
data _null_;
   set _ds_body end=last;

   /* Declare file reference for XML output */
*    file "C:\Reports\document.xml" lrecl=32767;
    file fxmldoco(document.xml)  LRECL=32767;

   /* Variables to track state */
   retain in_table 0;

   /* XML declaration and document start (only written once) */
   if _n_ = 1 then do;
      put '<?xml version="1.0" encoding="UTF-8" standalone="yes"?>';
      put '<w:document xmlns:w="http://schemas.openxmlformats.org/wordprocessingml/2006/main">';
      put '  <w:body>';
      /* Add the table title as a paragraph */
      put '    <w:p>';
      put '      <w:pPr>';
      put '        <w:jc w:val="center"/>';
      put '      </w:pPr>';
      put '      <w:r>';
      put '        <w:t>Adverse Event Summary</w:t>';
      put '      </w:r>';
      put '    </w:p>';
      /* Start the table */
      put '    <w:tbl>';
      put '      <w:tblPr>';
      put '        <w:tblStyle w:val="TableGrid"/>';
      put '        <w:tblW w:w="0" w:type="auto"/>';
      put '      </w:tblPr>';
      put '      <w:tblGrid>';
      put '        <w:gridCol w:w="4000"/>';
... more code

      /* First header row (Placebo, Drug, Total) */
      put '      <w:tr>';
      put '        <w:tc><w:p><w:r><w:t></w:t></w:r></w:p></w:tc>'; /* Empty cell for description */
      put '        <w:tc>';
      put '          <w:tcPr>';
      put '            <w:gridSpan w:val="2"/>';
      put '          </w:tcPr>';
      put '          <w:p>';
      put '            <w:pPr>';
      put '              <w:jc w:val="center"/>';
      put '            </w:pPr>';
      put '            <w:r>';
      put '              <w:t>Placebo</w:t>';
      put '            </w:r>';
      put '          </w:p>';
      put '        </w:tc>';
... more code

      /* Second header row (n, %) */
      put '      <w:tr>';
```

```
      put '       <w:tc><w:p><w:r><w:t></w:t></w:r></w:p></w:tc>'; /* Empty cell for description */
      put '       <w:tc><w:p><w:pPr><w:jc w:val="center"/></w:pPr><w:r><w:t>n</w:t></w:r></w:p></w:tc>';
... more code
   end;

   /* Write data rows */
   put '     <w:tr>';
   /* Description column (indented if ORDER > 1) */
   if ORDER > 1 then do;
      put '       <w:tc><w:p><w:r><w:t>  ' _COL1 +(-1) '</w:t></w:r></w:p></w:tc>';
   end;
   else do;
      put '       <w:tc><w:p><w:r><w:t>' _COL1 +(-1) '</w:t></w:r></w:p></w:tc>';
   end;
   /* Placebo n */
   put '       <w:tc><w:p><w:pPr><w:jc w:val="center"/></w:pPr><w:r><w:t>' CTOT1 +(-1) '</w:t></w:r></w:p></w:tc>';
   /* Placebo % */
   put '       <w:tc><w:p><w:pPr><w:jc w:val="center"/></w:pPr><w:r><w:t>' CPCT1 +(-1) '</w:t></w:r></w:p></w:tc>';
... more code


   /* Close the table and document on the last observation */
   if last then do;
      put '    </w:tbl>';
      put '  </w:body>';
      put '</w:document>';
   end;
run;
```

This example generates an XML file (document.xml) that defines a simple table of adverse events, structured according to the OOXML specification.

The resulting document.xml file contains the content of the AE table, structured as an WPML document. However, this file alone is not sufficient to create a .docx file; it needs to be combined with other XML files (e.g., styles.xml, numbering.xml, etc.) and packaged into a ZIP archive based on the WPML schema.

3.2 COMBINING THE XML FILES BASED ON DOCX FILE SCHEMA

We used an existing DOCX file, unzip it and saved the files preserving the folder structure in a central location. We used the approach described in section 3.1 to create document.xml and saved it in this folder. Following code snippet was then used to zip these files together, zip file was then renamed to have the extension of .docx to create a Word document.

```
data _null_;
  call system ("cd  /indata/xml02row; zip -r xml02row.zip .;
mv /indata/xml02row/xml02row.zip /output/ae0summary_gk_datanull.docx");
run;
```

The contents of the resulting DOCX file when opened in MS Word is as shown in Table 3.

Table 3: Adverse Event Summary

	Arm 1		Arm 2		Total	
	n	%	n	%	n	%
Subjects in population	12		12		24	
with one or more adverse events	11	(91.7)	12	(100.0)	23	(95.8)
with no adverse event	1	(8.3)	0	(0.0)	1	(4.2)
with drug-related adverse events	11	(91.7)	11	(91.7)	22	(91.7)
with toxicity grade 3-5 adverse events	6	(50.0)	6	(50.0)	12	(50.0)
with toxicity grade 3-5 drug-related adverse events	4	(33.3)	3	(25.0)	7	(29.2)
with non-serious adverse events	11	(91.7)	12	(100.0)	23	(95.8)
with serious adverse events	4	(33.3)	5	(41.7)	9	(37.5)
with serious drug-related adverse events	2	(16.7)	3	(25.0)	5	(20.8)
who died	0	(0.0)	0	(0.0)	0	(0.0)
who died due to a drug-related adverse event	0	(0.0)	0	(0.0)	0	(0.0)
discontinued drug due to an adverse event	2	(16.7)	1	(8.3)	3	(12.5)
discontinued drug due to a drug-related adverse event	2	(16.7)	1	(8.3)	3	(12.5)
discontinued drug due to a serious adverse event	0	(0.0)	0	(0.0)	0	(0.0)
discontinued drug due to a serious drug-related adverse event	0	(0.0)	0	(0.0)	0	(0.0)

3.3 FURTHER ENHANCEMENT TO THE FORMAT AND STYLE

Our stakeholders have specific requirements for table formatting. These include double lines for the top and bottom borders, minimal use of visible horizontal lines, horizontal lines displayed only in selected columns, merged cells where necessary, and the use of the "Times New Roman" font throughout.

Table 4: Adverse Event Summary enhanced to meet all internal requirements

	Arm 1		Arm 2		Total	
	n	(%)	n	(%)	n	(%)
Subjects in population	12		12		24	
with one or more adverse events	11	(91.7)	12	(100.0)	23	(95.8)
with no adverse event	1	(8.3)	0	(0.0)	1	(4.2)
with drug-related adverse events	11	(91.7)	11	(91.7)	22	(91.7)
with toxicity grade 3-5 adverse events	6	(50.0)	6	(50.0)	12	(50.0)
with toxicity grade 3-5 drug-related adverse events	4	(33.3)	3	(25.0)	7	(29.2)
with non-serious adverse events	11	(91.7)	12	(100.0)	23	(95.8)
with serious adverse events	4	(33.3)	5	(41.7)	9	(37.5)
with serious drug-related adverse events	2	(16.7)	3	(25.0)	5	(20.8)
who died	0	(0.0)	0	(0.0)	0	(0.0)
who died due to a drug-related adverse event	0	(0.0)	0	(0.0)	0	(0.0)
discontinued drug due to an adverse event	2	(16.7)	1	(8.3)	3	(12.5)
discontinued drug due to a drug-related adverse event	2	(16.7)	1	(8.3)	3	(12.5)
discontinued drug due to a serious adverse event	0	(0.0)	0	(0.0)	0	(0.0)
discontinued drug due to a serious drug-related adverse event	0	(0.0)	0	(0.0)	0	(0.0)
Database Cutoff Date: DDMMMYYYY.						

Here are some of the enhancements that were made to document.xml file from Table 3 to create Table 4. Table 5 shows some additional WPML tags for controlling table and cell borders.

To define table borders:

```
<w:tblBorders>
    <w:top w:val="double" w:sz="6" w:space="0" w:color="auto"/>
    <w:left w:val="single" w:sz="6" w:space="0" w:color="auto"/>
    <w:bottom w:val="double" w:sz="6" w:space="0" w:color="auto"/>
    <w:right w:val="single" w:sz="6" w:space="0" w:color="auto"/>
</w:tblBorders>
```

To make horizontal lines after “Arm 1”:

```
<w:tcBorders>
    <w:top w:val="double" w:sz="6" w:space="0" w:color="auto"/>
    <w:left w:val="nil"/>
    <w:bottom w:val="single" w:sz="2" w:space="0" w:color="auto"/>
    <w:right w:val="single" w:sz="2" w:space="0" w:color="auto"/>
</w:tcBorders>
```

Merging cells are accomplished by defining cell width using the following code:

```
<w:tblGrid>
    <w:gridCol w:w="3456"/>
    <w:gridCol w:w="864"/>
    <w:gridCol w:w="864"/>
    <w:gridCol w:w="864"/>
    <w:gridCol w:w="864"/>
    <w:gridCol w:w="864"/>
    <w:gridCol w:w="864"/>
</w:tblGrid>
```

To specify the font as Times New Roman:

```
<w:rPr>
    <w:rFonts w:ascii="Times New Roman" w:hAnsi="Times New Roman" w:cs="Times New Roman"/>
    <w:sz w:val="24"/>
    <w:szCs w:val="24"/>
    <w:lang w:eastAsia="en-US"/>
</w:rPr>
```

Table 5: WPML tags for controlling table and cell borders[2]

WPML Tags	Description
<w:tblBorders>	Defines table borders.
<w:tcBorders>	Defines table column borders.
<w:top>	Defines top border.
<w:left>	Defines left border.
<w:bottom>	Defines bottom border.
<w:right>	Defines right border.

Another key requirement is that the table contents need to be locked. This can be achieved using data _null_ by leveraging the structured document tag defined in WPML.

3.4 STRUCTURED DOCUMENT TAG <W:SDT>

The <w:sdt> element stands for Structured Document Tag. It is part of the WPML schema and is used to define content controls within a Word document.

Content controls are elements or placeholders in a Word document that can be used to structure, restrict, or manage content. They are commonly employed in templates, forms, or documents where specific sections need to be controlled or programmatically manipulated. The <w:sdt> element encapsulates these controls in the XML structure.

The use cases may include

- Forms: To create fillable fields (e.g., text boxes, drop-down lists, checkboxes) that users can interact with.
- Templates: To define placeholders for dynamic content that can be populated programmatically (e.g., via macros or external applications).
- Data Binding: To link the content control to an external data source (e.g., a database or XML data) using custom XML parts in the .docx file.
- Protection: To restrict editing of specific parts of the document while allowing users to fill in designated fields.

The <w:sdt> element typically contains several child elements that define its properties and content. Here's a breakdown of its common structure:

```
<w:sdt>
    <w:sdtPr>
        <w:rPr>
            <w:rFonts w:ascii="Times New Roman" w:hAnsi="Times New Roman"/>
            <w:sz w:val="18"/>
            <w:szCs w:val="18"/>
        </w:rPr>
        <w:id w:val="-1478911714"/>
        <w:lock w:val="sdtContentLocked"/>
        <w:placeholder>
            <w:docPart w:val="DefaultPlaceholder_-1854013440"/>
        </w:placeholder>
    </w:sdtPr>
    <w:sdtContent>
    ...
    </w:sdtContent>
</w:sdt>
```

<w:sdtPr>: The **Structured Document Tag Properties** element defines metadata and behavior for the content control, such as its type (e.g., text, date, combo box), identifier, label, and restrictions.

<w:sdtContent>: The **Structured Document Tag Content** element contains the actual content (e.g., text, paragraphs, or other Word elements) that appears within the control.

Common Attributes and Child Elements

Here are some key child elements and attributes you might find within <w:sdt>:

1. **<w:sdtPr> Child Elements**:

   <w:id>: A unique identifier for the content control (e.g., <w:id w:val="123456"/>).

   <w:tag>: A programmatic tag for referencing the control in code (e.g., <w:tag w:val="custName"/>).

   <w:lock>: Specifies locking behavior (e.g., w:val="sdtLocked" prevents deletion of the control, w:val="contentLocked" prevents editing of the content).

<w:text>: Indicates a plain text control (empty element, no attributes).

<w:dropDownList>: Defines a drop-down list with <w:listItem> elements for options.

2. **<w:sdtContent>**:

   Contains standard WPML elements like <w:p> (paragraph), <w:r> (run), and <w:t> (text) that represent the content displayed within the control.

These tags can be used to lock the contents within a table to prevent inadvertent changes during CSR authoring.

4. CONCLUSION

The shift from RTF to DOCX represents a significant evolution in table generation, driven by the need for modern, structured, and interoperable file formats. While SAS ODS offers a straightforward path to generate table in DOCX format, its limitations in customization and content protection necessitate alternative solutions for specialized reporting needs. The approach presented in this paper using SAS DATA _NULL_ to generate OOXML content provides a powerful and flexible framework for creating DOCX tables and listings tailored to stakeholder requirements. By producing document.xml files from SAS datasets and combining them with predefined style files (e.g., styles.xml), we achieve precise control over table formatting, such as specific border lines and font styles (e.g., Times New Roman), while incorporating structured document tags (<w:sdt>) to lock table contents against edits. The integration of SAS macros and Linux commands further automates the process, packaging the XML components into a complete DOCX file efficiently.

This method not only modernizes the table creation process but also eliminates the inefficiencies of converting RTF files into non-editable formats, a challenge in our current workflow. The adverse event summary table example demonstrates the practical application of this approach, showcasing its ability to deliver professional, compliant outputs with enhanced formatting and security features. Furthermore, this method can be extended to the creation of listings and figures, as OOXML provides a flexible framework to support these outputs. In summary, this technique provides a scalable and forward-looking toolset for TLF creation in clinical trial A&R.

5. REFERENCES

1. Eric Qi and Liping Zhang. 2003. “%RTFTable - A Powerful SAS Tool to Produce Rich Text Format Tables.” *Proceedings of the Pharmaceutical SAS Users Group 2003*
2. ISO/IEC 29500-1 “Information technology — Document description and processing languages — Office Open XML File Formats. Part 1: Fundamentals and Markup Language” https://www.iso.org/standard/71691.html

6. ACKNOWLEDGMENTS

The authors would like to thank Raghava Pamulapati and Hui Liu from our organization for their efforts in evaluating ODS Word for DOCX creation. Hui Liu also contributed by refining the Linux command for ZIP creation to ensure compliance with the DOCX schema. We also extend our gratitude to Susan Kramlik and Amy Gillespie from statistical programming leadership team for their valuable review and support.

7. CONTACT INFORMATION

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

Author Name: Eric Qi
Company : Merck & Co., Inc.
Address : 351 Sumneytown Pike
City: North Wales State: PA Zip: 1945
Email:eric_qi@merck.com

Author Name: Suhas R. Sanjee
Company : Merck & Co., Inc.
Address : 351 Sumneytown Pike
City: North Wales State: PA Zip: 1945
Email:suhas_sanjee@merck.com