

2021 - All about



OOXML Document Analysis

By Gülşah Köse

Software Engineer  Collabora
Online

@gulsahkse

How do we proceed when we receive a problematic OOXML document?

Reproduce the problem

There may be more than one problem in same document. We must make sure we produce the right one.

Simplify the document

The document may contain texts, images, sounds. But the problem might just about the images. If possible, other elements in the document can be cleaned.

Read the XML

OOXML documents (docx, pptx, xlsx etc) contains compressed XML files. Result of the “unzip test.docx” command will be seen like this.

```
.
├── [Content_Types].xml
├── docProps
│   ├── app.xml
│   └── core.xml
├── _rels
└── word
    ├── document.xml
    ├── fontTable.xml
    ├── _rels
    │   └── document.xml.rels
    ├── settings.xml
    └── styles.xml
```

Detect the XML part problematic element

For example: A paragraph element will look like as following

```
<w:p>
  <w:pPr>
    <w:pStyle w:val="Normal"/>
    <w:bidl w:val="0"/>
    <w:jc w:val="left"/>
    <w:rPr/>
  </w:pPr>
  <w:r>
    <w:rPr/>
    <w:t>Hello</w:t>
  </w:r>
</w:p>
```

Read the OOXML specs

It can be understood at this stage whether the syntax is appropriate or not. Apart from that, it allows us to understand the element in more detail than what I see in the interface.

Pick the keywords

We will need this to understand how Collabora Office handles these elements.

grep, read, debug loop

It cannot be formulated after this stage.

Reading code → detecting suspicious parts → debug them is a loop.

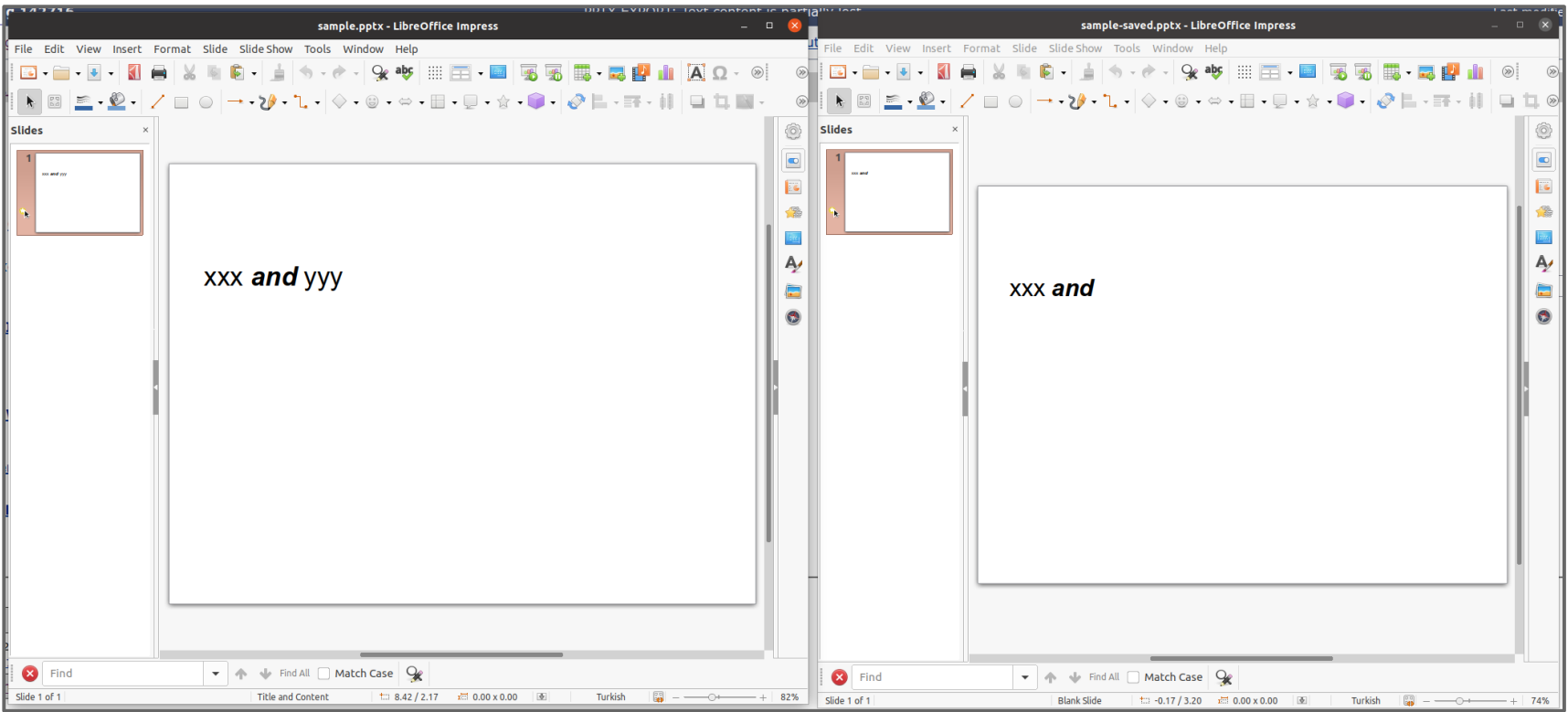
Often writing code is a small part of that proces :)

Real example

Bug report: https://bugs.documentfoundation.org/show_bug.cgi?id=142716
(PPTX EXPORT: Text content is partially lost)

Before

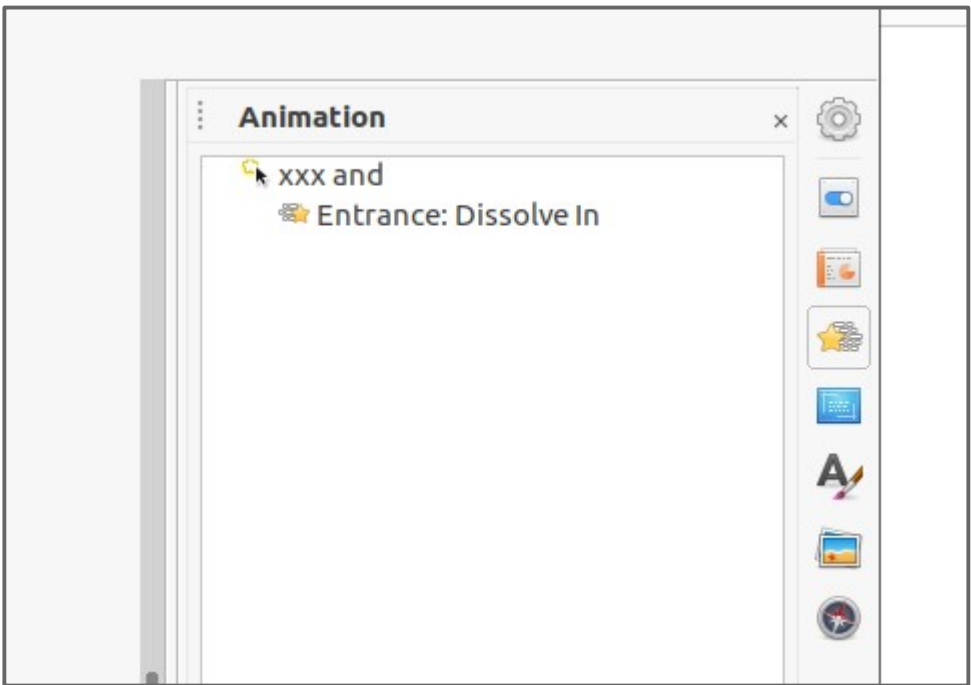
After



```

<p:txBody>
  <a:bodyPr/>
  <a:lstStyle/>
  <a:p>
    <a:r>
      <a:rPr lang="tr-TR" dirty="0" smtClean="0"/>
      <a:t>xxx</a:t>
    </a:r>
    <a:r>
      <a:rPr lang="en-NZ" dirty="0" smtClean="0"/>
      <a:t> </a:t>
    </a:r>
    <a:r>
      <a:rPr lang="en-NZ" b="1" i="1" dirty="0" smtClean="0"/>
      <a:t>and </a:t>
    </a:r>
    <a:r>
      <a:rPr lang="tr-TR" dirty="0" smtClean="0"/>
      <a:t>yyy</a:t>
    </a:r>
    <a:endParaRPr lang="en-US" dirty="0"/>
  </a:p>
</p:txBody>

```



```
<p:animEffect transition="in" filter="dissolve">
  <p:cBhvr>
    <p:cTn id="7" dur="500"/>
    <p:tgtEl>
      <p:spTgt spid="3">
        <p:txEl>
          <p:pRg st="0" end="0"/>
        </p:txEl>
      </p:spTgt>
    </p:tgtEl>
  </p:cBhvr>
</p:animEffect>
```

```

1875
1876 uno::Reference< text::XTextRange > SAL_CALL SvUnoTextBase::getEnd()
1877 {
1878     return SvUnoTextRangeBase::getEnd();
1879 }
1880
1881 OUString SAL_CALL SvUnoTextBase::getString()
1882 {
1883     return SvUnoTextRangeBase::getString();
1884 }
1885
1886 void SAL_CALL SvUnoTextBase::setString( const OUString& aString )
1887 {
1888     SvUnoTextRangeBase::setString(aString);
1889 }
1890
1891 // XEnumerationAccess
1892 uno::Reference< container::XEnumeration > SAL_CALL SvUnoTextBase::createEnumeration()
1893 {
1894     SolarMutexGuard aGuard;
1895     if( maSelection == ESelection(0,0,0,0) || maSelection == ESelection(EE_PARA_MAX_COUNT,0,0,0) )
1896     {
1897         ESelection aSelection;
1898         ::GetSelection( aSelection, GetEditSource()->GetTextForwarder() );
1899         return new SvUnoTextContentEnumeration(*this, aSelection);
1900     }
1901     else
1902     {
1903         return new SvUnoTextContentEnumeration(*this, maSelection);
1904     }
1905 }
1906
1907 // XElementAccess ( container::XEnumerationAccess )
1908 uno::Type SAL_CALL SvUnoTextBase::getElementType( )
1909 {
1910     return cppu::UnoType<text::XTextRange>::get();
1911 }
1912
1913 sal_Bool SAL_CALL SvUnoTextBase::hasElements( )
1914 {
1915

```

multi-thre Thread 0x7fffe90d50 In: SvUnoTextBase::createEnumeration

L1894 PC: 0x7ffff6c212f0

Thread 1 "soffice.bin" hit Breakpoint 1, animcore::(anonymous namespace)::AnimationNode::createEnumeration (this=0x55555a0a4e70) at /media/gulsah/EK/libreoffice-public-2/animations/source/animcore/animcore.cxx:1906

(gdb) n
(gdb) c
Continuing.

Thread 1 "soffice.bin" hit Breakpoint 1, animcore::(anonymous namespace)::AnimationNode::createEnumeration (this=0x555556f647b0) at /media/gulsah/EK/libreoffice-public-2/animations/source/animcore/animcore.cxx:1906

Continuing.

Thread 1 "soffice.bin" hit Breakpoint 1, animcore::(anonymous namespace)::AnimationNode::createEnumeration (this=0x55555711f270) at /media/gulsah/EK/libreoffice-public-2/animations/source/animcore/animcore.cxx:1906

(gdb) c
Continuing.

Thread 1 "soffice.bin" hit Breakpoint 1, animcore::(anonymous namespace)::AnimationNode::createEnumeration (this=0x55555a20ed10) at /media/gulsah/EK/libreoffice-public-2/animations/source/animcore/animcore.cxx:1906

Continuing.

Thread 1 "soffice.bin" hit Breakpoint 1, non-virtual thunk to SvUnoTextBase::createEnumeration() () at /media/gulsah/EK/libreoffice-public-2/include/editeng/unotext.hxx:451

(gdb) c
Continuing.

Thread 1 "soffice.bin" hit Breakpoint 1, SvUnoTextBase::createEnumeration (this=0x555559e75ab8) at /media/gulsah/EK/libreoffice-public-2/editeng/source/uno/unotext.cxx:1894

(gdb) |

Solution: <https://gerrit.libreoffice.org/c/core/+/-/117557>

```

8 void SAL_CALL SvxUnoTextBase::insertString( const uno::Reference< text::XTextRange >& xRange, const
OUString& aString, sal_Bool bAbsorb )
9 {
10     SolarMutexGuard aGuard;
11
12     if( !xRange.is() )
13         return;
14
15     if (GetEditSource())
16     {
17         ESelection aSelection;
18         ::GetSelection( aSelection, GetEditSource()->GetTextForwarder() );
19         SetSelection( aSelection );
20     }
21
22     SvxUnoTextRangeBase* pRange = comphelper::getUnoTunnelImplementation<SvxUnoTextRange>( xRange );
23     if(!pRange)
24         return;
25
26     // setString on SvxUnoTextRangeBase instead of itself QuickInsertText
27     // and UpdateData, so that the selection will be adjusted to
28     // SvxUnoTextRangeBase. Actually all cursor objects of this Text must
29     // to be statement to be adapted!
30
31     if (!bAbsorb) // do not replace -> append on tail
32         pRange->CollapseToEnd();
33
34     pRange->setString( aString );
35
36     pRange->CollapseToEnd();
37 }

```

```

1727
1728 void SAL_CALL SvxUnoTextBase::insertString( const uno::Reference< text::XTextRange >& xRange, const
OUString& aString, sal_Bool bAbsorb )
1729 {
1730     SolarMutexGuard aGuard;
1731
1732     if( !xRange.is() )
1733         return;
1734
1735     SvxUnoTextRangeBase* pRange = comphelper::getUnoTunnelImplementation<SvxUnoTextRange>( xRange );
1736     if(!pRange)
1737         return;
1738
1739     // setString on SvxUnoTextRangeBase instead of itself QuickInsertText
1740     // and UpdateData, so that the selection will be adjusted to
1741     // SvxUnoTextRangeBase. Actually all cursor objects of this Text must
1742     // to be statement to be adapted!
1743
1744     if (!bAbsorb) // do not replace -> append on tail
1745         pRange->CollapseToEnd();
1746
1747     pRange->setString( aString );
1748
1749     pRange->CollapseToEnd();
1750
1751     if (GetEditSource())
1752     {
1753         ESelection aSelection;
1754         ::GetSelection( aSelection, GetEditSource()->GetTextForwarder() );
1755         SetSelection( aSelection );
1756     }
1757 }

```


2021 - All about



Collabora Online

Thanks !



Collabora
Online

By Gülşah Köse

@CollaboraOffice
hello@collaboraoffice.com
Collaboraoffice.com