Binding Images

Introduction

A picture content control can be bound to an element containing a base 64 encoded image.

This document describes the behaviour of Microsoft Word, and what docx4j does (including its OpenDoPE extensions).

Microsoft Word

In order for Microsoft Word to resolve the binding, the picture content control must already contain some image.

For example:

```
    <w:sdt>
      <w:sdtPr>
        <w:id w:val="-1271625544"/>
        <w:dataBinding w:prefixMappings="" w:xpath="/data[1]/pic[1]" w:storeItemID="{7DF6A384-01E6-
4765-93A1-579EC703D9A9}"/>
          <w:picture/>
      </w:sdtPr>
      <w:sdtContent>
        <w:p >
          <w:r>
            <w:drawing>
              <wp:inline distT="0" distB="0" distL="0" distR="0" >
               <wp:extent cx="1447800" cy="1905000"/>
               <wp:effectExtent l="0" t="0" r="0" b="0"/>
               <wp:docPr id="2" name="Picture 1"/>
               <wp:cNvGraphicFramePr>
                 <a:graphicFrameLocks xmlns:a="http://schemas.openxmlformats.org/drawingml/2006/main"
noChangeAspect="1"/>
               </wp:cNvGraphicFramePr>
                <a:graphic xmlns:a="http://schemas.openxmlformats.org/drawingml/2006/main">
                  <a:graphicData uri="http://schemas.openxmlformats.org/drawingml/2006/picture">
                    <pic:pic xmlns:pic="http://schemas.openxmlformats.org/drawingml/2006/picture">
                     <pic:nvPicPr>
                       <pic:cNvPr id="0" name="Picture 1"/>
                       <pic:cNvPicPr>
                         <a:picLocks noChangeAspect="1" noChangeArrowheads="1"/>
                       </pic:cNvPicPr>
                     </pic:nvPicPr>
                      <pic:blipFill>
                        <a:blip r:embed="rId9">
                         <a:extLst>
                           <a:ext uri="{28A0092B-C50C-407E-A947-70E740481C1C}">
                             <a14:useLocalDpi
xmlns:a14="http://schemas.microsoft.com/office/drawing/2010/main" val="0"/>
                           </a:ext>
                         </a:extLst>
                        </a:blip>
                      </pic:blipFill>
                    </pic:pic>
                  </a:graphicData>
                </a:graphic>
              </wp:inline>
            </w:drawing>
          </w:r>
        </w:p>
      </w:sdtContent>
```

```
</w:sdt>
```

If you use the Developer menu within Word to add a picture content control, it will add markup like the above (except for the `w:dataBinding` element).

`w:xpath="/data[1]/pic[1]"` points to an element in the Custom XML part containing base 64 encoded data:

```
<data><pic>iVBORw0KGgoAAAANSUhEUgAAADkAAABLCAYAAADK8i9PAAAABGdBTUEAALGPC/xhBQAAAAlwSFlzAAAOwwAAD
sMBx2+..../H/5oAtm6ED26jpwDic8/F0ze/Ld/h71lz8HTlGzm2tT+Po/7aysHN4z549Po4hdv2/HPi8hwQE/2vyP8UFz9X
Bd2U1AAAAAElFTkSuQmCC</pic></data>
```

Word will not resolve the binding if the `w:drawing` element is not present, or `a:blip/@r:embed` does not point to an image part.

If you have things set up correctly, Word maintains/updates a **2 way association** between the base 64 encoded data in the Custom XML part, and the image shown in the picture content control:

- *On opening* the document in Word, the content control is updated from the base 64 encoded data, so the user sees whatever the base 64 encoded data provides.
- The update happens in the other direction if you paste a new picture into the content control. In that case, the new picture is base 64 encoded and injected into the XML part.

That behaviour is very similar to what Word does with a plain text data bind.

Using docx4j to resolve the binding

docx4j is also able to update the image which appears in the docx from base 64 encoded data.

It distinguishes 3 cases:

1. if the tag contains `od:Handler=picture`, use picture handling explained below
2. when test="`w:sdtContent//a:blip`", use this existing content (ie size etc), but replace the rel id with something pointing to an image constructed from the binding – this is similar to what Word does when the document is opened
3. otherwise, generate a wp:inline element from scratch.

Caution re floating pictures

It turns out that Microsoft have actively tried to prevent users from making picture content controls "float". There are still ways to do this via the Word user interface, but they've greyed out other ways.

If you do succeed in making your picture content control float in Word (then save, close), then the next time you open the document, Word will report a problem. It'll allow you to proceed, but won't actually "fix" anything! So the problem will still be there next time you open the docx.

In this respect Microsoft Word (2010 and 2013) do not fully implement the Open XML specification.

od:Handler=picture picture handling

To work around this Word issue with picture controls, the new OpenDoPE Word add-in automatically ***uses a rich text control, instead of a picture control, for an image***. This way, you can make the image float without Word complaining.

The tag `od:Handler=picture` signals to docx4j to unencode the bound base64 picture.

BindInverse

docx4j (as of v3.0.1) does not provide an API for injecting the current image from the picture content control into the bound XML element.

The scenario where this might be useful is where a user has edited the document. It ought not be necessary where the editing has taken place in Microsoft Word, since Word itself ought to have updated the Custom XML part with any new image data.

It would however be necessary for a rich text content control with tag `od:Handler=picture`

13 March 2013

Jason Harrop
Plutext Pty Ltd