### Roundtripping Alternate Content

Office Open [XML](XML.docx) defines a mechanism for the storage of content which is not defined by this Office Open [XML](XML.docx) Standard, for example extensions developed by future software applications which leverage the Open [XML](XML.docx) formats. This mechanism allows for the storage of a series of alternative representations of content, of which the consuming application should use the first alternative whose requirements are met.

[Example: Consider an application which creates a new paragraph property intended to make the colors of its text change colors randomly when it is displayed. This functionality is not defined in this Office Open [XML](XML.docx) Standard, and so the application might choose to create an alternative representation setting a different manual [color](color.docx) on each character for clients which do not understand this extension using an AlternateContent block as follows:

<ve:AlternateContent xmlns:ve="…">
 <ve:Choice Requires="colors" xmlns:colors="urn:randomTextColors">
 <w:[p](p.docx)>
 <w:[pPr](pPr.docx)>
 <colors:random colors:val="true" />
 </w:[pPr](pPr.docx)>
 <w:[r](r.docx)>
 <w:[t](t.docx)>Random colors!</w:[t](t.docx)>
 </w:[r](r.docx)>
 </w:[p](p.docx)>
 </ve:Choice>
 <ve:Fallback>
 <w:[p](p.docx)>
 <w:[r](r.docx)>
 <w:[rPr](rPr.docx)>
 <w:[color](color.docx) w:val="FF0000" />
 </w:[rPr](rPr.docx)>
 <w:[t](t.docx)>R</w:[t](t.docx)>
 </w:[r](r.docx)>
 <w:[r](r.docx)>
 <w:[rPr](rPr.docx)>
 <w:[color](color.docx) w:val="00FF00" />
 </w:[rPr](rPr.docx)>
 <w:[t](t.docx)>a</w:[t](t.docx)>
 </w:[r](r.docx)>
 …
 </w:[p](p.docx)>
 </ve:Fallback>
</ve:AlternateContent>

The Choice element that requires the new [color](color.docx) extensions uses the random element in its namespace, and the Fallback element allows clients that do not support this namespace to see an appropriate alternative representation. end example]

These alternate content blocks may occur at any location within a WordprocessingML document, and applications shall handle and process them appropriately (taking the appropriate choice).

However, WordprocessingML does not explicitly define a set of locations where applications shall attempt to store and roundtrip all non-taken choices whenever possible. This [behavior](behavior.docx) is therefore application-defined.

[Example: If an application does not understand the colors extension, the resulting file (if alternate choices are to be preserved would appear as follows:

<ve:AlternateContent xmlns:ve="…">
 <ve:Choice Requires="colors" xmlns:colors="urn:randomTextColors">
 …
 </ve:Choice>
 <ve:Fallback>
 …
 </ve:Fallback>
</ve:AlternateContent>

The file would then appear as follows after the choice is processed:

<w:[p](p.docx)>
 <w:[r](r.docx)>
 <w:[rPr](rPr.docx)>
 <w:[color](color.docx) w:val="FF0000" />
 </w:[rPr](rPr.docx)>
 <w:[t](t.docx)>R</w:[t](t.docx)>
 </w:[r](r.docx)>
 <w:[r](r.docx)>
 <w:[rPr](rPr.docx)>
 <w:[color](color.docx) w:val="00FF00" />
 </w:[rPr](rPr.docx)>
 <w:[t](t.docx)>a</w:[t](t.docx)>
 </w:[r](r.docx)>
 …
</w:[p](p.docx)>

The state of the alternate choices (preserved or not) is dependent on the application hosting the file. Preserving the content involves storing each non-taken choice while the file is being edited, and writing out the file with an AlternateContent block when it is resaved. end example]