This is a follow-up to my previous post about using VBA on Word to create a template and to reformat documents. This summarizes a few of the issues I had using VBA. I’m sure a lot of these are because I am relatively unfamiliar with VBA and the Word object model. On the other hand, there do seem to be a few idiosyncrasies in the way BA interacts with Word documents and the documentation is not always all that clear.

## Adding Paragraphs

I found it necessary to create an example of a paragraph formatted in each style as it was formatted. This is because some attributes of a style can only be set if the style is in use. But that’s OK, as it’s useful to have an example of each style in the template anyway. As I wanted to handle the built-in styles and new styles in a consistent way, I wrote a function to insert a paragraph, apply the specified style (creating it if necessary) and return a style object :

' Get a style by name; create it if it doesn't already exist

Private Function GetParaStyle(name As String) As Style

Dim s As Style

Dim p As Paragraph

' Turn off error handling so we can tell when a style doesn't already exist

On Error Resume Next

Set s = ActiveDocument.Styles(name)

If (s Is Nothing) Or (s = Empty) Or (s.Type <> wdStyleTypeParagraph) Then

' Create the style if it doesn't exist

Set s = ActiveDocument.Styles.Add(name, wdStyleTypeParagraph)

End If

' Create an example of the style: partly for reference, also

' some features don't get set if the style isn't used

With ActiveDocument.Paragraphs

Set p = .Add

Set p = .Add

p.Range.Text = name

p.Style = s

End With

‘ Return the style object

Set GetParaStyle = s

End Function

While doing this, I found that it was necessary to call the Paragraphs.Add() method twice to insert a new paragraph – otherwise the previously inserted paragraph was just overwritten.

## Paragraph & List Numbering

This is a real can of worms. The idea of separating out the style of the numbering to be applied to a paragraph is quite a good one in practice but doesn’t seem to have been implemented too well.

SIDEBAR: Word has many features for doing documentation “properly”, e.g. using paragraph and character styles for all formatting (with no manual overrides). Unfortunately most of these features are fairly buggy and unusable. I’m sure this because they are rarely used by the vast majority of Word users. I think about 90% of documents I see are entirely formatted using the “Normal” paragraph style, with manual overrides used to create headings and all other formatting. Word goes out of its way to make this easy and makes it harder to do things the right way.

As an example, a comment I came across when trying to understand list numbering was:

“I'm sorry, but I have no experience with list styles, and very few Word experts do because we haven't figured out how they are supposed to work (mostly they just don't, apparently).” [source] And this from a long-time Word expert who has been given Microsoft’s “Most Valued Professional” status.

It is pretty tricky to ensure that a numbering style is used consistently with a given paragraph. For example, resetting numbering to start from 1 can change the style of numbering assigned to a paragraph which can do anything from change the indentation to breaking the sequence of numbering in *other* paragraphs. The key thing is to ensure that a different and, ideally, named numbering template is used for each separate paragraph style. This should avoid unwanted interactions between the numbering of different paragraph styles. This was particularly important for the non-standard numbered paragraph formats created for Figure and Table titles.

### Creating a Named List Template

The following code snippet shows how to create a named numbering template and associate it with a given paragraph style. First a function to get or create a named list number template:

Private Function GetTableListTemplate() As ListTemplate

Dim lt As ListTemplate

On Error Resume Next

' Get existing template

Set lt = ActiveDocument.ListTemplates("TableListTemplate")

If (lt Is Nothing) Then

' Or create if doesn't exist

Set lt = ActiveDocument.ListTemplates.Add(False, "TableListTemplate") ' \*\*Note\*\*

End If

' Set up the number format

With lt.ListLevels(1)

.NumberStyle = wdListNumberStyleArabic

.NumberFormat = "Table %1."

.TrailingCharacter = wdTrailingSpace

End With

' Return the list template

Set GetTableListTemplate = lt

End Function

Note: it seems that the name of the list template has to be string literal, not a variable (in Word 2000, at least).

This can then be applied to the “Table Title” paragraph style when it is created:

Dim s As Style

Dim lt As ListTemplate

Set s = GetParaStyle("TableTitle")

Set lt = GetTableListTemplate()

Call s.LinkToListTemplate(lt)

### Heading Styles

Heading styles are handled similarly. The main difference is that they have multiple levels of numbering. The numbering format we wanted was “1.2.3 Heading text”; this was implemented as follows:

' Initial number format

s = "%1"

With ListGalleries(wdOutlineNumberGallery).ListTemplates(1)

For i = 1 To .ListLevels.Count

With .ListLevels(i)

.NumberStyle = wdListNumberStyleArabic

.NumberFormat = s

.TrailingCharacter = wdTrailingTab

.NumberPosition = 0

.TabPosition = MillimetersToPoints(StandardLeftIndent)

.TextPosition = MillimetersToPoints(StandardLeftIndent)

.Alignment = wdListLevelAlignLeft

.LinkedStyle = "Heading " & i

' Update number format for next level

s = s & ".%" & i + 1

End With

Next

End With

### Numbered Lists

For the numbered lists, I found it more reliable to format the numbering template already assigned to the default list styles (“List Number”, etc). For example:

Dim lt As ListLevel

Dim n As String

For i = 1 To 5

n = "List Number"

If i > 1 Then n = n & " " & i

Dim s As Style

Set s = GetParaStyle(n)

With s

.BaseStyle = ActiveDocument.Styles("Normal")

' Set up numbering

Set lt = .ListTemplate.ListLevels(1)

With lt

.NumberStyle = wdListNumberStyleArabic

.NumberFormat = "%1."

.TrailingCharacter = wdTrailingTab

.Alignment = wdListLevelAlignLeft

.NumberPosition = MillimetersToPoints(StandardLeftIndent + 3 + (i - 1) \* 8)

.TabPosition = MillimetersToPoints(StandardLeftIndent + 3 + i \* 8)

.TextPosition = MillimetersToPoints(StandardLeftIndent + 3 + i \* 8)

End With

'Call .LinkToListTemplate(lt)

' Set the indentation for this level of list numbering

With .ParagraphFormat

' calculate appropriate indents for each level:

' number indented by 3mm

' +8mm at each level

.LeftIndent = MillimetersToPoints(StandardLeftIndent + 3 + i \* 8)

.FirstLineIndent = -MillimetersToPoints(8)

.TabStops.ClearAll

.TabStops.Add (MillimetersToPoints(StandardLeftIndent + 3 + i \* 8))

End With

End With

Next