ComponentOne

Word for UWP

ComponentOne, a division of GrapeCity
201 South Highland Avenue, Third Floor
Pittsburgh, PA 15206 USA

Website: http://www.componentone.com
Sales: sales@componentone.com
Telephone: 1.800.858.2739 or 1.412.681.4343 (Pittsburgh, PA USA Office)

Trademarks

Warranty

ComponentOne warrants that the media on which the software is delivered is free from defects in material and workmanship, assuming normal use, for a period of 90 days from the date of purchase. If a defect occurs during this time, you may return the defective media to ComponentOne, along with a dated proof of purchase, and ComponentOne will replace it at no charge. After 90 days, you can obtain a replacement for the defective media by sending it and a check for $2 5 (to cover postage and handling) to ComponentOne.

Except for the express warranty of the original media on which the software is delivered is set forth here, ComponentOne makes no other warranties, express or implied. Every attempt has been made to ensure that the information contained in this manual is correct as of the time it was written. ComponentOne is not responsible for any errors or omissions. ComponentOne's liability is limited to the amount you paid for the product. ComponentOne is not liable for any special, consequential, or other damages for any reason.

Copying and Distribution

Table of Contents

Word for UWP Overview

ComponentOne Studio introduces **Word for UWP** with rich API to create Word documents with advanced features. **Word for UWP** creates, reads, writes Word documents with Microsoft Word Open XML format (*.docx) and RTF Documents with Rich Text format (*.rtf) extension.

Word for UWP uses C1WordDocument class (member of C1.UWP.Word assembly), which provides all advanced properties and methods required to generate both Microsoft Word and RTF Documents. The documents generated using **Word for UWP** can be easily stored in file system and exported in standard Microsoft Word document format.

Help with UWP Edition

For information on installing **ComponentOne Studio UWP Edition**, licensing, technical support, namespaces, and creating a project with the controls, please visit Getting Started with ComponentOne Studio UWP Edition.

Word for UWP Key Features

The key features of **Word for UWP** are as follows:

- **Rich Object Model**
  Word for UWP provides a rich and powerful object model which is easily programmable. All you have to use is C1WordDocument class that provides all advanced properties and methods, to create both Microsoft Word and RTF Documents.
- **Advanced Library**
  Word for UWP as advanced methods to add images, text, graphics, quotes, and table in Word documents.
- **Draw and Play**
  Draw and play Windows Runtime User Interface (UI) objects (FrameworkElements) in Word document using **Word for UWP**.
- **Different Paper Sizes**
  Create a document with multiple paper sizes and orientation using **Word for UWP**.
- **Draw Text**
  Word for UWP allows you to draw text in different fonts and use font properties in your Word documents.
- **Add Tables**
  Word for UWP includes RTFTable object that helps you add data to table cells.
- **Add TOC**
  Word for UWP allows adding Table of Content (TOC) to a document containing headers along with the text. It also allows you to move to different pages within the document.
- **Hyperlinks and bookmarks**
  Word for UWP provides you bookmarks to navigate within the document and hyperlinks to navigate to different URLs.
- **Text Flow**
  Word for UWP allows you to flow text into columns and pages in a word document.

Object Model Summary

Word for UWP provides a rich and powerful object model which is easily programmable. The C1WordDocument class provides all advanced properties and methods to create both Microsoft Word and RTF Documents.

C1WordDocument Object
Add, AddBookmark, AddBookmarkStart, AddBookmarkEnd, AddLink, AddParagraph, AddPicture, ColumnBreak, DrawArc, DrawRectangle, DrawString, FillPie, LoadAsync, Count, Current, Hyperlink, ShapesWord2007Compatible
Font
Bold, FontFamily, Italic, Name, Size, Strikeout, Style, Underline
Pen
Color, DashPattern, DashStyle
RTFPageSize Object
A0, A1, A4, A10, B1, B5, HalfLetter, Legal, Letter
StringFormat
Alignment, Angle, LineAlignment, LineSpacing

Quick Start: Word for UWP

The quick start guide familiarizes you with **Word for UWP**. In this section, you learn to create a new UWP project in Visual Studio. You would require to add the **C1Word** reference (dll) and a button in the application to create and save a document.

Step 1 of 3: Setting up the Application

You begin with creating a UWP application in Visual Studio, adding **C1Word** reference and a button control to your application using the following steps:

1. Create a new UWP project in Visual Studio.
2. Add the **C1Word** reference (dll) to your application.
3. Add the following namespaces in the code view:

   Visual Basic

   ```
   Imports C1.Xaml.Word
   Imports C1.Xaml.Word.Objects
   ```

   C#

   ```
   using C1.Xaml.Word;
   using C1.Xaml.Word.Objects;
   ```

4. Switch to design view and add a Button control to the Form in your application to start using the **Word for UWP**. Set the **Content** property to a suitable text such as, **Text**, **Name** property to **btnText** and the **Click** event to **btnText_Click**.
5. Double-click the **btnText_Click** event from the properties window.
   The **btnText_Click** event gets created in the code view.

Step 2 of 3: Adding Text

While you are still in code view in the Visual Studio project, add the following code within the **btnText_Click** event:

Visual Basic

```
Dim word As New C1WordDocument()

' use landscape for more impact
word.Landscape = True

' measure and show some text
Dim text = "Hello!! This is a sample text."
Dim font = New Font("Segoe UI Light", 20, RtfFontStyle.Italic)

' add paragraph
word.AddParagraph(text, font, Colors.BlueViolet, RtfHorizontalAlignment.Justify)

Dim picker As New FileSavePicker()
picker.FileTypeChoices.Add("Word Open XML Format (.docx)", New List(Of String)()
From { _
        ".docx" _
})
picker.FileTypeChoices.Add("RTF Format (.rtf)", New List(Of String)() From { _
```

```
        ".rtf" _
})

picker.DefaultFileExtension = ".docx"
picker.SuggestedStartLocation = PickerLocationId.DocumentsLibrary
Dim file As StorageFile = Await picker.PickSaveFileAsync()
If file IsNot Nothing Then
        Await word.SaveAsync(file, If(file.Name.ToLower().EndsWith(".docx"),
FileFormat.OpenXml, FileFormat.Rtf))

        Dim dlg As New MessageDialog("File saved")
        Await dlg.ShowAsync()
End If
```

C#

```
C1WordDocument word = new C1WordDocument();

// use landscape for more impact
word.Landscape = true;

// measure and show some text
var text = "Hello!! This is a sample text.";
var font = new Font("Segoe UI Light", 20, RtfFontStyle.Italic);

// add paragraph
word.AddParagraph(text, font, Colors.BlueViolet, RtfHorizontalAlignment.Justify);

FileSavePicker picker = new FileSavePicker();
picker.FileTypeChoices.Add("Word Open XML Format (.docx)", new List < string > () {
  ".docx"
});
picker.FileTypeChoices.Add("RTF Format (.rtf)", new List < string > () {
  ".rtf"
});

picker.DefaultFileExtension = ".docx";
picker.SuggestedStartLocation = PickerLocationId.DocumentsLibrary;
StorageFile file = await picker.PickSaveFileAsync();
if (file != null) {
  await word.SaveAsync(file, file.Name.ToLower().EndsWith(".docx") ?
FileFormat.OpenXml : FileFormat.Rtf);

  MessageDialog dlg = new MessageDialog("File saved");
  await dlg.ShowAsync();
}
```

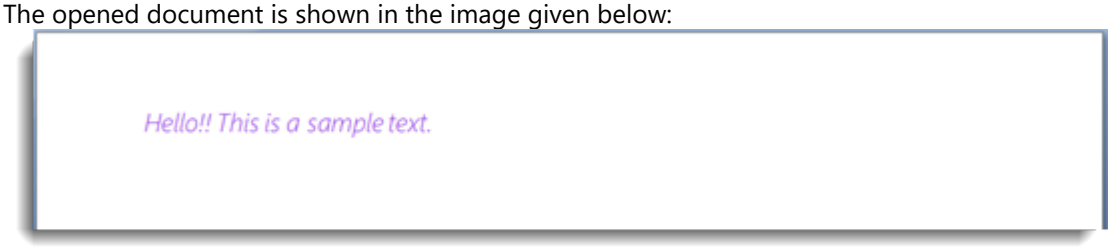
In the above code, a word document is created in landscape mode with some text added to it using AddParagraph method. It allows you to save the created document to the location of your choice. You can select the desired location and write the name of your document as well. The document is then saved with the same name you provided it.

Step 3 of 3: Running the Application

In previous step, you added code to the **btnText_Click** event to create, add text and save the word document. In this step, you will run the application and view the document created. Follow the given steps to run the application:

1. Press **F5** to run the application.
2. Click the Text Button to view the document you created and saved using **Word for UWP**.

The opened document is shown in the image given below:



Working with Word for UWP

Word for UWP comes with a rich API and object model that enables you to create word documents as well as RTF documents supported in Microsoft Word and other editors. Understand the working with **Word for UWP** in detail in the topics listed below.

Basic Level Working

Using **Word for UWP**, you can add simple illustrations such as images, graphics, quotes or more to your word document. **Word for UWP** enables you to add these illustrations with few lines of code. Following topics walk you through adding simple text and some basic illustrations.

Adding Text

You can add text to a word document using **Word for UWP**. You need to write the desired text and use AddParagraph method to add text. You can also set other properties, such as font style, family, color, and more for the text to be displayed in the word document using Font class and its properties. The implementation of adding text to a word document is given in the code below:

1. Create an object of C1WordDocument class with the following code:

   Visual Basic

   ```
   Dim word As New C1WordDocument()
   ```

   C#

   ```
   C1WordDocument word = new C1WordDocument();
   ```

2. Write the following code to add text to the document:

   Visual Basic

   ```
   Dim text = "Hello!! This is a sample text."
   Dim font = New Font("Segoe UI Light", 20, RtfFontStyle.Italic)
   word.AddParagraph(text, font, Colors.BlueViolet,
   RtfHorizontalAlignment.Justify)
   WordUtils.Save(word)
   ```

   C#

   ```
   var text = "Hello!! This is a sample text.";
   var font = new Font("Segoe UI Light", 20, RtfFontStyle.Italic);
   word.AddParagraph(text, font, Colors.BlueViolet,
   RtfHorizontalAlignment.Justify);
   WordUtils.Save(word);
   ```

The document will look similar to the image below:

Adding Images

You might need to insert images in your word document along with the text to enhance the overall appearance of your document.

Note that a class named **WordUtils** is used in the code given below. It is available in the product sample located at the following location on your system:
Documents\ComponentOne Samples\UWP\WordSamples
You can use these classes in your application from the mentioned location.

To add an image in your document, use the following code that loads an image and sketches it in the document:

Visual Basic

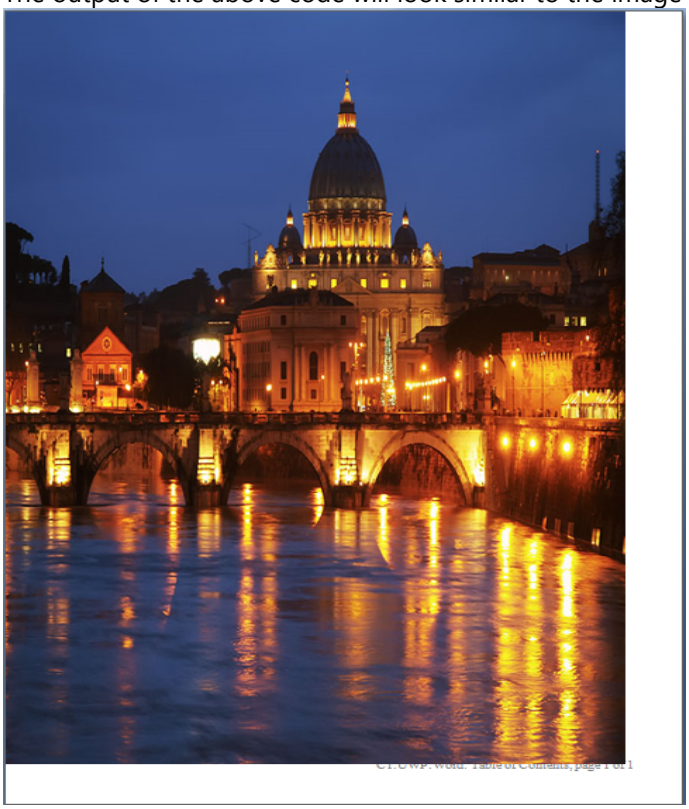
```
' calculate page rect (discounting margins)
Dim rcPage As Rect = WordUtils.PageRectangle(word)
Dim ras As New InMemoryRandomAccessStream()
' load image into writeable bitmap
Dim wb As New WriteableBitmap(880, 660)
Dim file = Await StorageFile.GetFileFromApplicationUriAsync(New Uri("ms-
appx:///WordSamplesLib/Assets/pic.jpg"))

wb.SetSource(Await file.OpenReadAsync())
Dim rcPic As New Rect(New Point(0, 0), New Point(word.PageSize.Width,
word.PageSize.Height))

' draw on page preserving aspect ratio
word.DrawImage(wb, rcPic)
WordUtils.Save(word)
```

C#

```
// calculate page rect (discounting margins)
Rect rcPage = WordUtils.PageRectangle(word);
InMemoryRandomAccessStream ras = new InMemoryRandomAccessStream();
// load image into writeable bitmap
WriteableBitmap wb = new WriteableBitmap(880, 660);
var file = await StorageFile.GetFileFromApplicationUriAsync(new Uri("ms-
appx:///WordSamplesLib/Assets/pic.jpg"));

wb.SetSource(await file.OpenReadAsync());
Rect rcPic = new Rect(new Point(0, 0), new Point(word.PageSize.Width,
word.PageSize.Height));

// draw on page preserving aspect ratio
word.DrawImage(wb, rcPic);
WordUtils.Save(word);
```

The image is loaded into writeable bitmap and DrawImage method is used to draw the image in the above code.

The output of the above code will look similar to the image given below:



Drawing Graphics

Adding graphics enhances the appearance of a document and make them visually appealing. You can add various types of shapes in your documents such as, Arc, Beizer, Ellipse, Line, Pie, polygon, PolygonLine, and Rectangle. Use the following code to add graphics such as pie, rectangles, polyline, and beziers along with the text:

Visual Basic

```
Dim word = New C1WordDocument()
' set up to draw
Dim rc As New Rect(100, 100, 300, 200)
Dim text As String = "Hello world of .NET Graphics and Word/RTF." & vbCr & vbLf &
"Nice to meet you."
Dim font As New Font("Times New Roman", 12, RtfFontStyle.Italic Or
RtfFontStyle.Underline)

' draw to word document
Dim penWidth As Integer = 0
Dim penRGB As Byte = 0
word.FillPie(Colors.DarkRed, rc, 0, 20F)
word.FillPie(Colors.Green, rc, 20F, 30F)
word.FillPie(Colors.Teal, rc, 60F, 12F)
word.FillPie(Colors.Orange, rc, -80F, -20F)
For startAngle As Single = 0 To 359 Step 40
        Dim penColor As Color = Color.FromArgb(&Hff, penRGB, penRGB, penRGB)
        Dim pen As New Pen(penColor,
System.Math.Max(System.Threading.Interlocked.Increment(penWidth),penWidth - 1))
        penRGB = CByte(penRGB + 20)
        word.DrawArc(pen, rc, startAngle, 40F)
Next
```

```
word.DrawRectangle(Colors.Red, rc)


' show a Bezier curve
Dim pts = New Point() {New Point(400, 100), New Point(420, 130), New Point(500,
140), New Point(530, 120)}

' draw Bezier
word.DrawBeziers(New Pen(Colors.Green, 4), pts)


' show Bezier control points
word.DrawPolyline(Colors.Gray, pts)
For Each pt As Point In pts
        word.FillRectangle(Colors.Orange, pt.X - 2, pt.Y - 2, 4, 4)
Next

' title
word.DrawString(Strings.Bezier, font, Colors.Black, New Rect(500, 150, 100, 100))
```

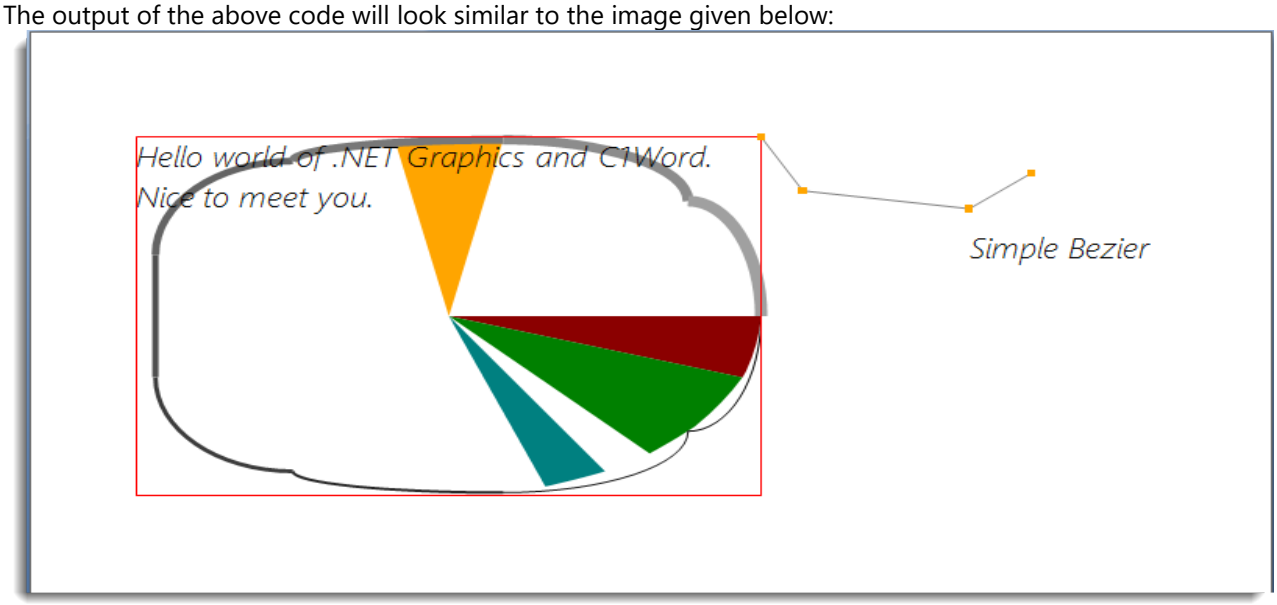
C#

```
var word = new C1WordDocument();
// set up to draw
Rect rc = new Rect(100, 100, 300, 200);
string text = "Hello world of .NET Graphics and Word/RTF.\r\nNice to meet you.";
Font font = new Font("Times New Roman", 12, RtfFontStyle.Italic |
RtfFontStyle.Underline);

// draw to word document
int penWidth = 0;
byte penRGB = 0;
word.FillPie(Colors.DarkRed, rc, 0, 20 f);
word.FillPie(Colors.Green, rc, 20 f, 30 f);
word.FillPie(Colors.Teal, rc, 60 f, 12 f);
word.FillPie(Colors.Orange, rc, -80 f, -20 f);
for (float startAngle = 0; startAngle < 360; startAngle += 40) {
  Color penColor = Color.FromArgb(0xff, penRGB, penRGB, penRGB);
  Pen pen = new Pen(penColor, penWidth++);
  penRGB = (byte)(penRGB + 20);
  word.DrawArc(pen, rc, startAngle, 40 f);
}
word.DrawRectangle(Colors.Red, rc);


// show a Bezier curve
var pts = new Point[] {
  new Point(400, 100), new Point(420, 130),
    new Point(500, 140), new Point(530, 120),
};

// draw Bezier
word.DrawBeziers(new Pen(Colors.Green, 4), pts);


// show Bezier control points
word.DrawPolyline(Colors.Gray, pts);
foreach(Point pt in pts) {
  word.FillRectangle(Colors.Orange, pt.X - 2, pt.Y - 2, 4, 4);
}
```

```
// title
word.DrawString(Strings.Bezier, font, Colors.Black, new Rect(500, 150, 100, 100));
```

In the above code, DrawRectangle, DrawArc, DrawPolyline, and DrawBeziers methods are used to draw graphics of different types such as, line, rectangle, pie, and bezier. Besides, DrawString method is used to draw and show text in the document.

The output of the above code will look similar to the image given below:



Adding Quotes

You can add quotes from another file in your document using **Word for UWP**.

Note that a class named **WordUtils** is used in the code given below. It is available in the product sample located at the following location on your system:
Documents\ComponentOne Samples\UWP\WordSample
You can use these classes in your application from the mentioned location.

You can use following code to add quotes to your document from a text file:

Visual Basic

```
' calculate page rect (discounting margins)
Dim rcPage As Rect = WordUtils.PageRectangle(word)
Dim rc As Rect = rcPage

' initialize output parameters
Dim hdrFont As New Font("Arial", 14, RtfFontStyle.Bold)
Dim titleFont As New Font("Arial", 24, RtfFontStyle.Bold)
Dim txtFont As New Font("Times New Roman", 10, RtfFontStyle.Italic)

' add title
rc = WordUtils.RenderParagraph(word, word.Info.Title, titleFont, rcPage, rc)
' build document
For Each s As String In GetQuotes()
        Dim authorQuote As String() = s.Split(ControlChars.Tab)

        ' render header (author)
        Dim author = authorQuote(0)
        rc.Y += 20
```

```
        rc = WordUtils.RenderParagraph(word, author, hdrFont, rcPage, rc, True)

        ' render body text (quote)
        Dim text As String = authorQuote(1)
        rc.X = rcPage.X + 36
        ' << indent body text by 1/2 inch
        rc.Width = rcPage.Width - 40
        rc = WordUtils.RenderParagraph(word, text, txtFont, rcPage, rc)
        rc.X = rcPage.X
        ' << restore indent
        rc.Width = rcPage.Width ' << add 12pt spacing after each quote
        rc.Y += 12
Next
Private Shared Function GetQuotes() As List
        Dim list = New List()

        Using sr = New StreamReader(GetType(BasicTextPage).GetTypeInfo().Assembly.
                   GetManifestResourceStream("WordSamples.Resources.quotes.txt"))
                Dim quotes = sr.ReadToEnd()
                For Each quote As String In quotes.Split("*"C)
                        Dim pos As Integer = quote.IndexOf(vbCr & vbLf)
                        If pos > -1 Then
                                Dim q = String.Format("{0}" & vbTab & "{1}",
quote.Substring(0, pos),
                                        quote.Substring(pos + 2).Trim())
                                list.Add(q)
                        End If
                Next
        End Using

        Return list
End Function
```

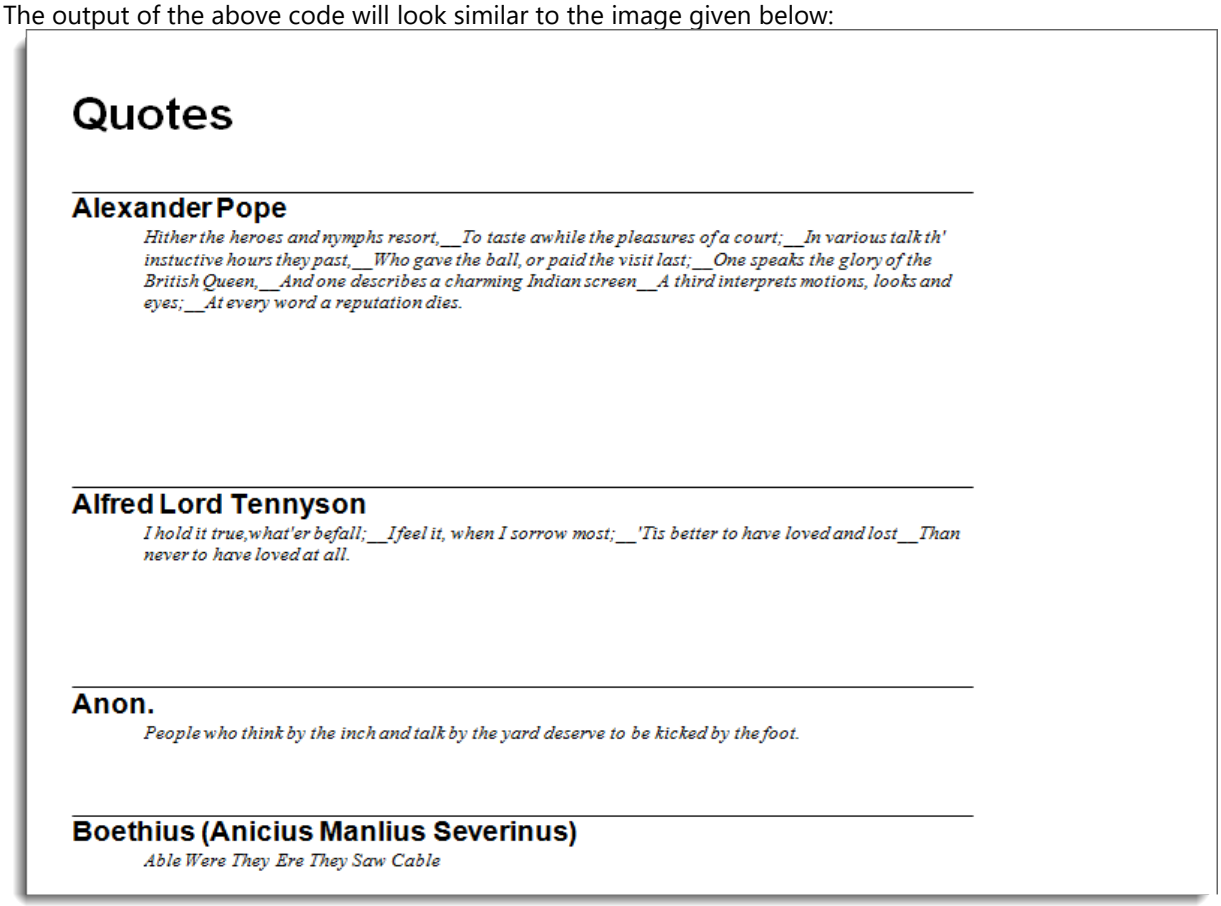
C#

```
// calculate page rect (discounting margins)
Rect rcPage = WordUtils.PageRectangle(word);
Rect rc = rcPage;

// initialize output parameters
Font hdrFont = new Font("Arial", 14, RtfFontStyle.Bold);
Font titleFont = new Font("Arial", 24, RtfFontStyle.Bold);
Font txtFont = new Font("Times New Roman", 10, RtfFontStyle.Italic);

// add title
rc = WordUtils.RenderParagraph(word, word.Info.Title, titleFont, rcPage, rc);

// build document
foreach(string s in GetQuotes()) {
  string[] authorQuote = s.Split('\t');

  // render header (author)
  var author = authorQuote[0];
  rc.Y += 20;
  rc = WordUtils.RenderParagraph(word, author, hdrFont, rcPage, rc, true);

  // render body text (quote)
  string text = authorQuote[1];
  rc.X = rcPage.X + 36; // << indent body text by 1/2 inch
  rc.Width = rcPage.Width - 40;
  rc = WordUtils.RenderParagraph(word, text, txtFont, rcPage, rc);
```

```
    rc.X = rcPage.X; // << restore indent
    rc.Width = rcPage.Width;
    rc.Y += 12; // << add 12pt spacing after each quote
  }

  static List GetQuotes() {
    var list = new List();

    using(var sr = new StreamReader(typeof(BasicTextPage).GetTypeInfo().Assembly.
                   GetManifestResourceStream("WordSamples.Resources.quotes.txt"))) {
      var quotes = sr.ReadToEnd();
      foreach(string quote in quotes.Split('*')) {
        int pos = quote.IndexOf("\r\n");
        if (pos > -1) {
          var q = string.Format("{0}\t{1}", quote.Substring(0, pos),
  quote.Substring(pos + 2).Trim());
          list.Add(q);
        }
      }
    }

    return list;
  }
```

The above code reads the quotes from a text file and writes them in a document. It adds title to the document at first, then renders the header and body text, and writes the text in the document.

The output of the above code will look similar to the image given below:



Advanced Level Working

Usually a word document contains text and adding images, illustrations, tables or metafiles to the document makes it look visually appealing and more descriptive. The listed topics will walk you through adding these advanced features to your word document using **Word for UWP**:

Inserting Table

Table is used in word documents to present data in a well formatted form with the help of rows and columns. **Word for UWP** enables you to add table to a word document using the following code:

Visual Basic

```
' add table
word.LineBreak()
Dim rows As Integer = 4
Dim cols As Integer = 2
Dim table As New RtfTable(rows, cols)
word.Add(table)
table.Rows(0).Cells(0).SetMerged(1, 2)

For row As Integer = 0 To rows - 1
        If row = 0 Then
                table.Rows(row).Height = 50
        End If
        For col As Integer = 0 To cols - 1
                If row = 0 AndAlso col = 0 Then
                        text = Strings.DocumentBasicText2
                        table.Rows(row).Cells(col).Alignment =
ContentAlignment.MiddleCenter
                        table.Rows(row).Cells(col).BackFilling = Colors.LightPink
                Else
                        text = String.Format("table cell {0}:{1}.", row, col)
                        table.Rows(row).Cells(col).BackFilling = Colors.LightYellow
                End If
                table.Rows(row).Cells(col).Content.Add(New RtfString(text))
                table.Rows(row).Cells(col).BottomBorderWidth = 2
                table.Rows(row).Cells(col).TopBorderWidth = 2
                table.Rows(row).Cells(col).LeftBorderWidth = 2
                table.Rows(row).Cells(col).RightBorderWidth = 2
                If col = cols - 1 Then
                        table.Rows(row).Cells(col).Alignment =
ContentAlignment.BottomRight
                End If
        Next
Next
```

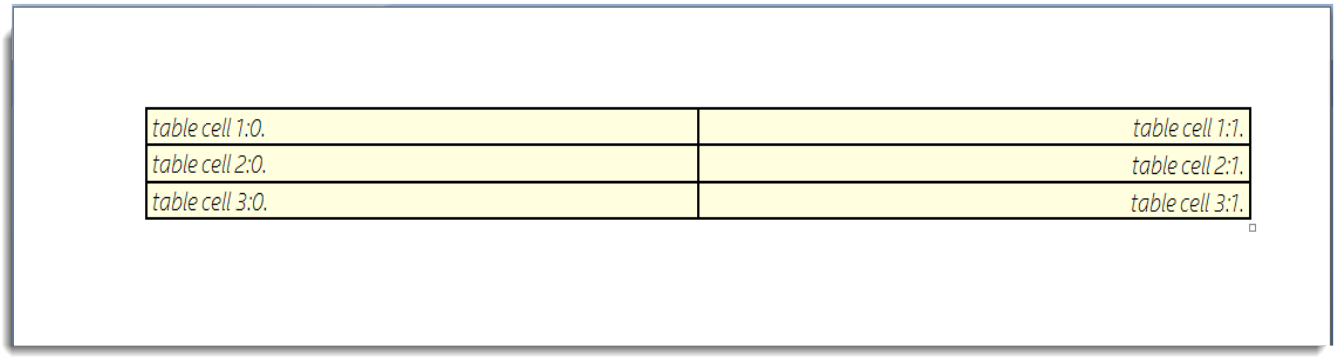
C#

```
// add table
word.LineBreak();
int rows = 4;
int cols = 2;
RtfTable table = new RtfTable(rows, cols);
word.Add(table);
table.Rows[0].Cells[0].SetMerged(1, 2);

for (int row = 0; row < rows; row++) {
  if (row == 0) {
    table.Rows[row].Height = 50;
```

```
    }
    for (int col = 0; col < cols; col++) {
      if (row == 0 && col == 0) {
        text = Strings.DocumentBasicText2;
        table.Rows[row].Cells[col].Alignment = ContentAlignment.MiddleCenter;
        table.Rows[row].Cells[col].BackFilling = Colors.LightPink;
      } else {
        text = string.Format("table cell {0}:{1}.", row, col);
        table.Rows[row].Cells[col].BackFilling = Colors.LightYellow;
      }
      table.Rows[row].Cells[col].Content.Add(new RtfString(text));
      table.Rows[row].Cells[col].BottomBorderWidth = 2;
      table.Rows[row].Cells[col].TopBorderWidth = 2;
      table.Rows[row].Cells[col].LeftBorderWidth = 2;
      table.Rows[row].Cells[col].RightBorderWidth = 2;
      if (col == cols - 1) {
        table.Rows[row].Cells[col].Alignment = ContentAlignment.BottomRight;
      }
    }
  }
```

The above code creates a table with proper indentation and alignment in a word document.

The output of the above code will look similar to the image given below:



Adding TOC

Word for UWP enables you to add TOC to a word document containing text with the respective headers. These headers can then be used to create a TOC.

Note that a class named **WordUtils** is used in the code given below. It is available in the product sample located at the following location on your system:
Documents\ComponentOne Samples\UWP\WordSample
You can use these classes in your application from the mentioned location.

The following code creates a TOC for a word document with text and headers in it:

Visual Basic

```
Dim _doc As New C1WordDocument()

' add title
Dim titleFont As New Font("Tahoma", 24, RtfFontStyle.Bold)
Dim rcPage As Rect = WordUtils.PageRectangle(doc)
Dim rc As Rect = WordUtils.RenderParagraph(doc, doc.Info.Title, titleFont, rcPage,
rcPage, False)
rc.Y += 12
```

```
' create nonsense document
Dim bkmk = New List()
Dim headerFont As New Font("Arial", 14, RtfFontStyle.Bold)
Dim bodyFont As New Font("Times New Roman", 11)
For i As Integer = 0 To 29
        ' create ith header (as a link target and outline entry)
        Dim header As String = String.Format("{0}. {1}", i + 1, BuildRandomTitle())
        rc = WordUtils.RenderParagraph(doc, header, headerFont, rcPage, rc, True, _
                True)

        ' save bookmark to build TOC later
        Dim pageNumber As Integer = doc.CurrentPage() + 1
        bkmk.Add(New String() {pageNumber.ToString(), header})

        ' create some text
        rc.X += 36
        rc.Width -= 36
        For j As Integer = 0 To 3 + (_rnd.[Next](20) - 1)
                Dim text As String = BuildRandomParagraph()
                rc = WordUtils.RenderParagraph(doc, text, bodyFont, rcPage, rc)
                rc.Y += 6
        Next
        rc.X -= 36
        rc.Width += 36
        rc.Y += 20
Next

' start Table of Contents
doc.PageBreak()
' start TOC on a new page
Dim tocPage As Integer = doc.CurrentPage()
' save page index (to move TOC later)
rc = WordUtils.RenderParagraph(doc, Strings.TableOfContentsDocumentTitle, titleFont,
rcPage, rcPage, True)
rc.Y += 12
rc.X += 30
rc.Width -= 40

' render Table of Contents
Dim dottedPen As New Pen(Colors.Gray, 1.5F)
dottedPen.DashStyle = DashStyle.Dot
Dim sfRight As New StringFormat()
sfRight.Alignment = HorizontalAlignment.Right
rc.Height = bodyFont.Size * 1.2
For Each entry As String() In bkmk
        ' get bookmark info
        Dim page As String = entry(0)
        Dim header As String = entry(1)

        ' render header name and page number
        doc.DrawString(header, bodyFont, Colors.Black, rc)
        doc.DrawString(page, bodyFont, Colors.Black, rc, sfRight)

        ' add local hyperlink to entry
        doc.AddLink(Strings.PoundSign + header)

        ' move on to next entry
        rc = WordUtils.Offset(rc, 0, rc.Height)
        If rc.Bottom > rcPage.Bottom Then
                doc.PageBreak()
                rc.Y = rcPage.Y
```

```
        End If
Next
```

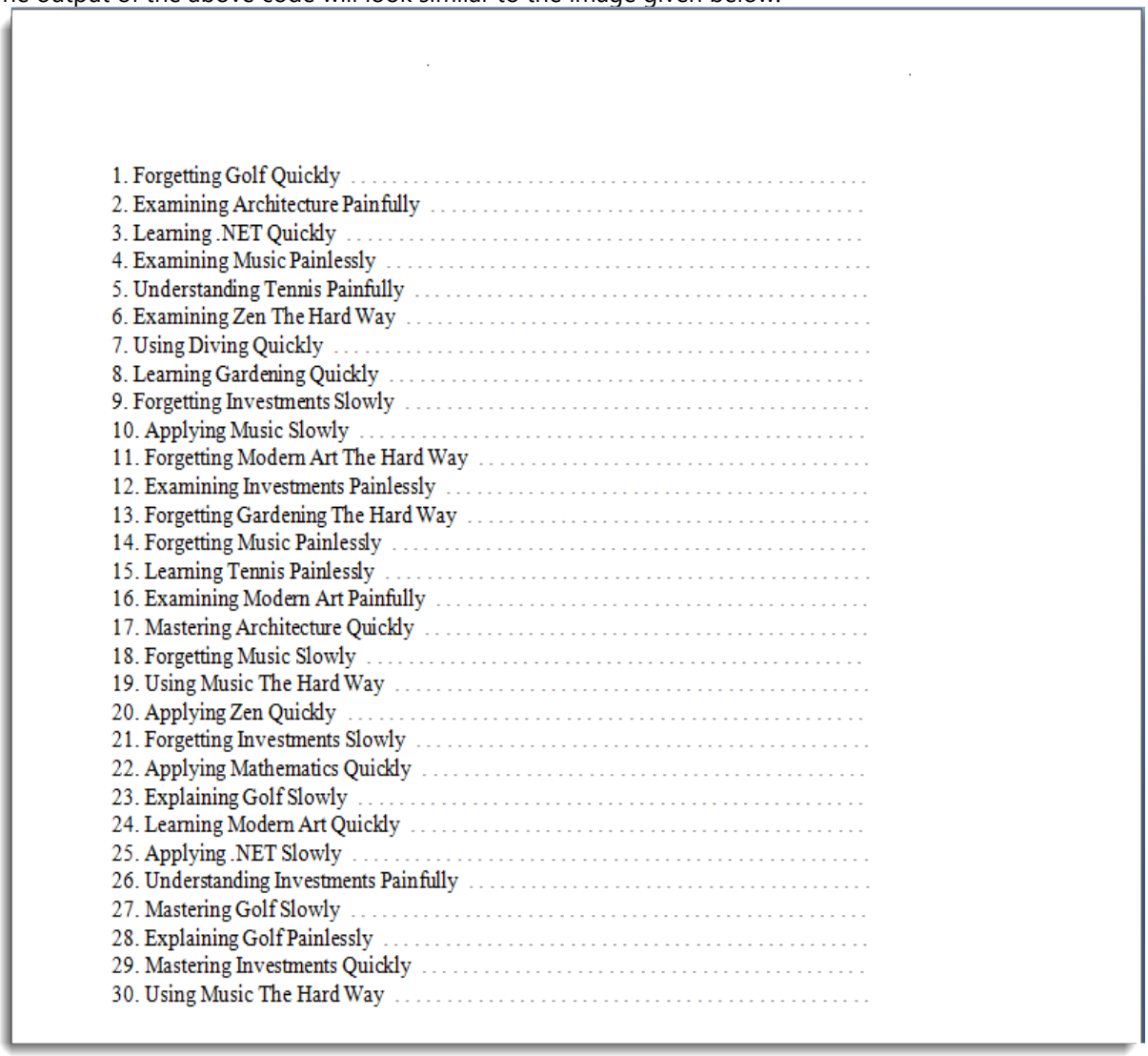
C#

```
C1WordDocument _doc = new C1WordDocument();

// add title
Font titleFont = new Font("Tahoma", 24, RtfFontStyle.Bold);
Rect rcPage = WordUtils.PageRectangle(doc);
Rect rc = WordUtils.RenderParagraph(doc, doc.Info.Title, titleFont, rcPage, rcPage,
false);
rc.Y += 12;

// create nonsense document
var bkmk = new List();
Font headerFont = new Font("Arial", 14, RtfFontStyle.Bold);
Font bodyFont = new Font("Times New Roman", 11);
for (int i = 0; i < 30; i++) {
  // create ith header (as a link target and outline entry)
  string header = string.Format("{0}. {1}", i + 1, BuildRandomTitle());
  rc = WordUtils.RenderParagraph(doc, header, headerFont, rcPage, rc, true, true);

  // save bookmark to build TOC later
  int pageNumber = doc.CurrentPage() + 1;
  bkmk.Add(new string[] {
    pageNumber.ToString(), header
  });

  // create some text
  rc.X += 36;
  rc.Width -= 36;
  for (int j = 0; j < 3 + _rnd.Next(20); j++) {
    string text = BuildRandomParagraph();
    rc = WordUtils.RenderParagraph(doc, text, bodyFont, rcPage, rc);
    rc.Y += 6;
  }
  rc.X -= 36;
  rc.Width += 36;
  rc.Y += 20;
}

// start Table of Contents
doc.PageBreak(); // start TOC on a new page
int tocPage = doc.CurrentPage(); // save page index (to move TOC later)
rc = WordUtils.RenderParagraph(doc, Strings.TableOfContentsDocumentTitle, titleFont,
rcPage, rcPage, true);
rc.Y += 12;
rc.X += 30;
rc.Width -= 40;

// render Table of Contents
Pen dottedPen = new Pen(Colors.Gray, 1.5 f);
dottedPen.DashStyle = DashStyle.Dot;
StringFormat sfRight = new StringFormat();
sfRight.Alignment = HorizontalAlignment.Right;
rc.Height = bodyFont.Size * 1.2;
foreach(string[] entry in bkmk) {
  // get bookmark info
  string page = entry[0];
  string header = entry[1];
```

```
    // render header name and page number
    doc.DrawString(header, bodyFont, Colors.Black, rc);
    doc.DrawString(page, bodyFont, Colors.Black, rc, sfRight);

    // add local hyperlink to entry
    doc.AddLink(Strings.PoundSign + header);

    // move on to next entry
    rc = WordUtils.Offset(rc, 0, rc.Height);
    if (rc.Bottom > rcPage.Bottom) {
      doc.PageBreak();
      rc.Y = rcPage.Y;
    }
  }
```

The above code adds bookmark to the headers of the text in document. These bookmarks are then used to generate a TOC.

The output of the above code will look similar to the image given below:



Creating Word Document with different paper sizes

Word for UWP gives you the flexibility to create a word document with different paper sizes. You can use **PaperKind** enum to specify any of the available standard paper sizes.

Note that a class named **WordUtils** is used in the code given below. It is available in the product sample located at the following location on your system:
Documents\ComponentOne Samples\UWP\WordSample
You can use these classes in your application from the mentioned location.

The implementation of PaperKind enum is given in the following code:

Visual Basic

```
' create one page with each paper size
Dim firstPage As Boolean = True
For Each pk As PaperKind In [Enum].GetValues(GetType(PaperKind))
        ' Silverlight doesn't have Enum.GetValues
        'PaperKind pk = fi;

        ' skip custom size
        If pk = PaperKind.[Custom] Then
                Continue For
        End If

        ' add new page for every page after the first one
        If Not firstPage Then
                word.PageBreak()
        End If
        firstPage = False

        ' set paper kind and orientation
        'word.PaperKind = pk;
        word.Landscape = Not word.Landscape

        ' draw some content on the page
        rc = WordUtils.PageRectangle(word)
        rc = WordUtils.Inflate(rc, -6, -6)
        'string text = string.Format(Strings.StringFormatTwoArg, word.PaperKind,
word.Landscape);
        Dim text As String = String.Format(Strings.StringFormatTwoArg, pk,
word.Landscape)
        word.DrawString(text, font, Colors.Black, rc, sf)
        word.DrawRectangle(Colors.Black, rc)
Next
```

C#

```
// create one page with each paper size
bool firstPage = true;
foreach(PaperKind pk in Enum.GetValues(typeof(PaperKind))) {
  // Silverlight doesn't have Enum.GetValues
  //PaperKind pk = fi;

  // skip custom size
  if (pk == PaperKind.Custom) continue;

  // add new page for every page after the first one
  if (!firstPage) word.PageBreak();
  firstPage = false;

  // set paper kind and orientation
  //word.PaperKind = pk;
```

```
  word.Landscape = !word.Landscape;

  // draw some content on the page
  rc = WordUtils.PageRectangle(word);
  rc = WordUtils.Inflate(rc, -6, -6);
  //string text = string.Format(Strings.StringFormatTwoArg, word.PaperKind,
word.Landscape);
  string text = string.Format(Strings.StringFormatTwoArg, pk, word.Landscape);
  word.DrawString(text, font, Colors.Black, rc, sf);
  word.DrawRectangle(Colors.Black, rc);
}
```

Adding Text Flow

You can use text flow in a word document. Using **Word for UWP**, you can flow text into columns and pages of a document.

Note that a class named **WordUtils** is used in the code given below. It is available in the product sample located at the following location on your system:
Documents\ComponentOne Samples\UWP\WordSample
You can use these classes in your application from the mentioned location.

The following code shows how the text flow feature can be used in **Word for UWP**:

Visual Basic

```
 ' load long string from resource file
Dim text As String = Strings.ResourceNotFound

Using sr = New StreamReader(GetType(BasicTextPage).GetTypeInfo().
          Assembly.GetManifestResourceStream("WordSamples.Resources.flow.txt"))
        text = sr.ReadToEnd()
End Using
text = text.Replace(vbTab, "    ")

' create word document
word.Info.Title = "Text Flow"

' table
Dim rows As Integer = 4
Dim cols As Integer = 2
Dim table As New RtfTable(rows, cols)
word.Add(table)
table.Rows(0).Cells(0).SetMerged(1, 2)

For row As Integer = 0 To rows - 1
        If row = 0 Then
                table.Rows(row).Height = 50
        End If
        For col As Integer = 0 To cols - 1
                'RtfParagraph paragraph = new RtfParagraph();
                'paragraph.Alignment = RtfHorizontalAlignment.Undefined;
                'paragraph.Content.Add(new RtfString(string.Format("table cell {0}:
{1}.", row, col)));
                'table.Rows[row].Cells[col].Content.Add(paragraph);
                table.Rows(row).Cells(col).Content.Add(New
RtfString(String.Format("table cell {0}:{1}.", row, col)))
                If row = 0 AndAlso col = 0 Then
                        table.Rows(row).Cells(col).Alignment =
```

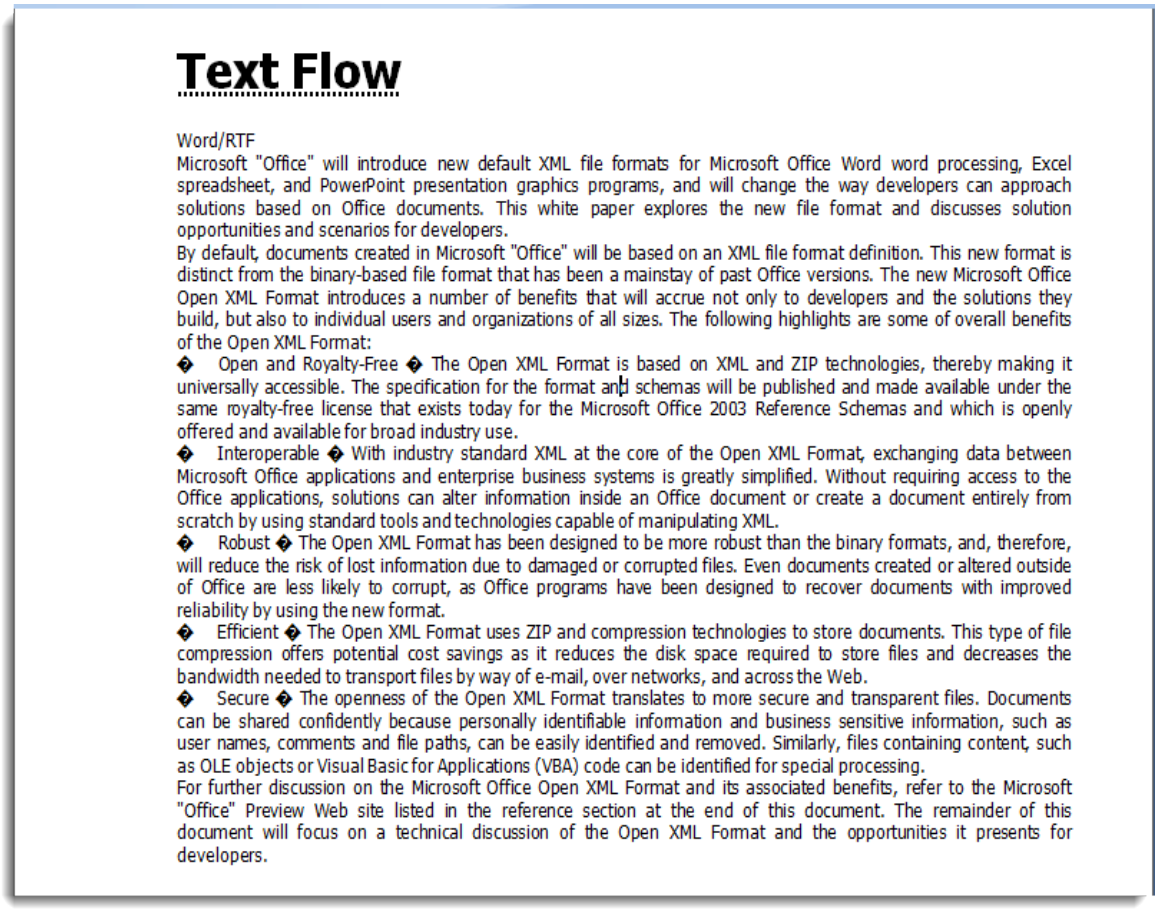
```
ContentAlignment.MiddleCenter
                        table.Rows(row).Cells(col).BackFilling = Colors.LightPink
                Else
                        table.Rows(row).Cells(col).BackFilling = Colors.LightYellow
                End If
                table.Rows(row).Cells(col).BottomBorderWidth = 2
                table.Rows(row).Cells(col).TopBorderWidth = 2
                table.Rows(row).Cells(col).LeftBorderWidth = 2
                table.Rows(row).Cells(col).RightBorderWidth = 2
        Next
Next

Return

' add title
Dim titleFont As New Font("Tahoma", 24, RtfFontStyle.Bold)
Dim bodyFont As New Font("Tahoma", 9)
Dim rcPage As Rect = WordUtils.PageRectangle(word)
Dim rc As Rect = WordUtils.RenderParagraph(word, word.Info.Title, titleFont, rcPage,
rcPage, False)
rc.Y += titleFont.Size + 6
rc.Height = rcPage.Height - rc.Y

' create two columns for the text
Dim rcLeft As Rect = rc
rcLeft.Width = rcPage.Width / 2 - 12
rcLeft.Height = 300
rcLeft.Y = (rcPage.Y + rcPage.Height - rcLeft.Height) / 2
Dim rcRight As Rect = rcLeft
rcRight.X = rcPage.Right - rcRight.Width

' start with left column
rc = rcLeft

' render string spanning columns and pages
While True
        ' render as much as will fit into the rectangle
        rc = WordUtils.Inflate(rc, -3, -3)
        'int nextChar = word.DrawString(text, bodyFont, Colors.Black, rc);
        word.DrawString(text, bodyFont, Colors.Black, rc)
        rc = WordUtils.Inflate(rc, +3, +3)
        word.DrawRectangle(Colors.LightGray, rc)

        ' break when done
        'if (nextChar >= text.Length)
        If True Then
                Exit While
        End If

        ' get rid of the part that was rendered
        'text = text.Substring(nextChar);

        ' switch to right-side rectangle
        If rc.Left = rcLeft.Left Then
                rc = rcRight
        Else
                ' switch to left-side rectangle on the next page
                word.PageBreak()
                rc = rcLeft
        End If
End While
```

C#

```
// load long string from resource file
string text = Strings.ResourceNotFound;

using(var sr = new StreamReader(typeof(BasicTextPage).GetTypeInfo().

Assembly.GetManifestResourceStream("WordSamples.Resources.flow.txt"))) {
  text = sr.ReadToEnd();
}
text = text.Replace("\t", "    ");

// create word document
word.Info.Title = "Text Flow";

// table
int rows = 4;
int cols = 2;
RtfTable table = new RtfTable(rows, cols);
word.Add(table);
table.Rows[0].Cells[0].SetMerged(1, 2);

for (int row = 0; row < rows; row++) {
  if (row == 0) {
    table.Rows[row].Height = 50;
  }
  for (int col = 0; col < cols; col++) {
    //RtfParagraph paragraph = new RtfParagraph();
    //paragraph.Alignment = RtfHorizontalAlignment.Undefined;
    //paragraph.Content.Add(new RtfString(string.Format("table cell {0}:{1}.", row,
col)));
    //table.Rows[row].Cells[col].Content.Add(paragraph);
    table.Rows[row].Cells[col].Content.Add(new RtfString(string.Format("table cell
{0}:{1}.", row, col)));
    if (row == 0 && col == 0) {
      table.Rows[row].Cells[col].Alignment = ContentAlignment.MiddleCenter;
      table.Rows[row].Cells[col].BackFilling = Colors.LightPink;
    } else {
      table.Rows[row].Cells[col].BackFilling = Colors.LightYellow;
    }
    table.Rows[row].Cells[col].BottomBorderWidth = 2;
    table.Rows[row].Cells[col].TopBorderWidth = 2;
    table.Rows[row].Cells[col].LeftBorderWidth = 2;
    table.Rows[row].Cells[col].RightBorderWidth = 2;
  }
}

return;

// add title
Font titleFont = new Font("Tahoma", 24, RtfFontStyle.Bold);
Font bodyFont = new Font("Tahoma", 9);
Rect rcPage = WordUtils.PageRectangle(word);
Rect rc = WordUtils.RenderParagraph(word, word.Info.Title, titleFont, rcPage,
rcPage, false);
rc.Y += titleFont.Size + 6;
rc.Height = rcPage.Height - rc.Y;

// create two columns for the text
Rect rcLeft = rc;
rcLeft.Width = rcPage.Width / 2 - 12;
```

```
rcLeft.Height = 300;
rcLeft.Y = (rcPage.Y + rcPage.Height - rcLeft.Height) / 2;
Rect rcRight = rcLeft;
rcRight.X = rcPage.Right - rcRight.Width;

// start with left column
rc = rcLeft;

// render string spanning columns and pages
for (;;) {
  // render as much as will fit into the rectangle
  rc = WordUtils.Inflate(rc, -3, -3);
  //int nextChar = word.DrawString(text, bodyFont, Colors.Black, rc);
  word.DrawString(text, bodyFont, Colors.Black, rc);
  rc = WordUtils.Inflate(rc, +3, +3);
  word.DrawRectangle(Colors.LightGray, rc);

  // break when done
  //if (nextChar >= text.Length)
  {
    break;
  }

  // get rid of the part that was rendered
  //text = text.Substring(nextChar);

  // switch to right-side rectangle
  if (rc.Left == rcLeft.Left) {
    rc = rcRight;
  } else // switch to left-side rectangle on the next page
  {
    word.PageBreak();
    rc = rcLeft;
  }
}
```

The output of the above code will look similar to the image given below:



Text Flow

Word/RTF
Microsoft "Office" will introduce new default XML file formats for Microsoft Office Word word processing, Excel spreadsheet, and PowerPoint presentation graphics programs, and will change the way developers can approach solutions based on Office documents. This white paper explores the new file format and discusses solution opportunities and scenarios for developers.
By default, documents created in Microsoft "Office" will be based on an XML file format definition. This new format is distinct from the binary-based file format that has been a mainstay of past Office versions. The new Microsoft Office Open XML Format introduces a number of benefits that will accrue not only to developers and the solutions they build, but also to individual users and organizations of all sizes. The following highlights are some of overall benefits of the Open XML Format:
• Open and Royalty-Free • The Open XML Format is based on XML and ZIP technologies, thereby making it universally accessible. The specification for the format and schemas will be published and made available under the same royalty-free license that exists today for the Microsoft Office 2003 Reference Schemas and which is openly offered and available for broad industry use.
• Interoperable • With industry standard XML at the core of the Open XML Format, exchanging data between Microsoft Office applications and enterprise business systems is greatly simplified. Without requiring access to the Office applications, solutions can alter information inside an Office document or create a document entirely from scratch by using standard tools and technologies capable of manipulating XML.
• Robust • The Open XML Format has been designed to be more robust than the binary formats, and, therefore, will reduce the risk of lost information due to damaged or corrupted files. Even documents created or altered outside of Office are less likely to corrupt, as Office programs have been designed to recover documents with improved reliability by using the new format.
• Efficient • The Open XML Format uses ZIP and compression technologies to store documents. This type of file compression offers potential cost savings as it reduces the disk space required to store files and decreases the bandwidth needed to transport files by way of e-mail, over networks, and across the Web.
• Secure • The openness of the Open XML Format translates to more secure and transparent files. Documents can be shared confidently because personally identifiable information and business sensitive information, such as user names, comments and file paths, can be easily identified and removed. Similarly, files containing content, such as OLE objects or Visual Basic for Applications (VBA) code can be identified for special processing.
For further discussion on the Microsoft Office Open XML Format and its associated benefits, refer to the Microsoft "Office" Preview Web site listed in the reference section at the end of this document. The remainder of this document will focus on a technical discussion of the Open XML Format and the opportunities it presents for developers.

Export UI

Word for UWP enables you to export UI to a document.

Note that a class named **WordUtils** is used in the code given below. It is available in the product sample located at the following location on your system:
Documents\ComponentOne Samples\UWP\WordSample
You can use these classes in your application from the mentioned location.

Using **Word for UWP**, you can export UI as an image which then draws this image in word document. Use the following code to export the UI to a word document:

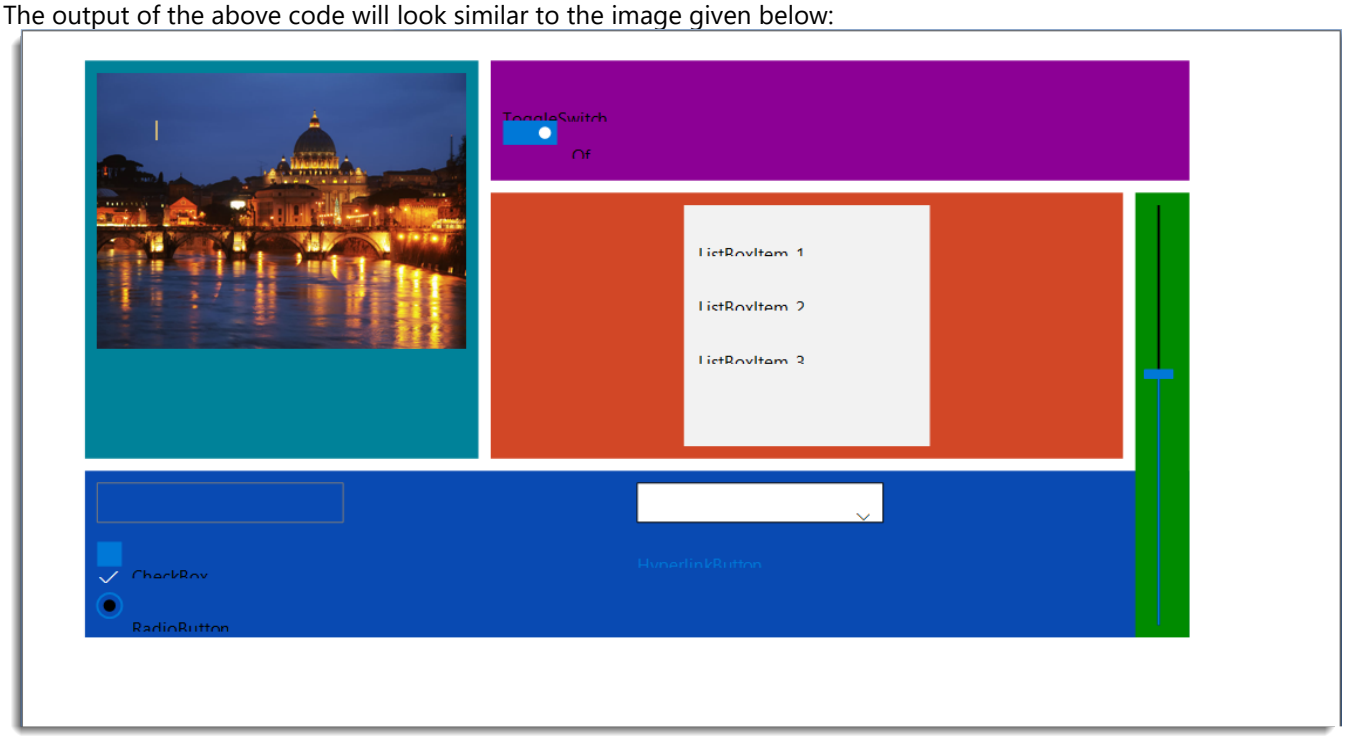
Visual Basic

```
Dim _doc As New C1WordDocument()
_doc.Clear()
progressRing.IsActive = True
_doc.Landscape = True
panel.Arrange(_doc.PageRectangle())

' Draw every UI elements inside the panel in word document.
Await _doc.DrawElement(panel, _doc.PageRectangle())
WordUtils.SetDocumentInfo(_doc, Strings.RenderUIDocumentTitle)
WordUtils.Save(_doc)
progressRing.IsActive = False
```

C#

```
C1WordDocument _doc = new C1WordDocument();
_doc.Clear();
progressRing.IsActive = true;
_doc.Landscape = true;
panel.Arrange(_doc.PageRectangle());

// Draw every UI elements inside the panel in word document.
await _doc.DrawElement(panel, _doc.PageRectangle());
WordUtils.SetDocumentInfo(_doc, Strings.RenderUIDocumentTitle);
WordUtils.Save(_doc);
progressRing.IsActive = false;
```

The output of the above code will look similar to the image given below:



Word for UWP Samples

Please be advised that this ComponentOne software tool is accompanied by various sample projects and/or demos which may make use of other development tools included with the ComponentOne Studio. Please refer to the pre-installed product samples through the following path:
Documents\ComponentOne Samples\UWP

The list of samples available for **Word for UWP** is as follows:

Sample	Description
WordSamples	This sample showcases main features of the C1.UWP.Word, such as adding text, images, graphics, quotes, table, TOC, text flow, creating a document with different paper sizes, and export UI to word document.