ComponentOne

Word for WinForms

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 WinForms Overview

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

Word component uses C1.C1Word.C1WordDocument class, which provides all advanced properties and methods required to generate both Microsoft Word and RTF Documents. The documents generated using Word component can be easily stored in file system and exported in standard Microsoft Word document format.

Help with WinForms Edition

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

Word for WinForms Key Features

The key features of Word for WinForms are as follows:

- **Rich Object Model**
  Word for WinForms 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 WinForms uses advanced methods to add pictures, paragraph, text, font, graphics, curves, and table in Word documents.
- **Draw Text**
  Word for WinForms allows you to draw text in different fonts and use font properties in your Word documents.
- **Add Tables**
  Word for WinForms includes RTFTable object that helps you add data to table cells.
- **Hyperlinks and bookmarks**
  Word for WinForms provides you bookmarks to navigate within the document and hyperlinks to navigate to different URLs.
- **Draw and Play Metafiles**
  Draw and play metafiles in Word document using **C1Word** class. The Word for WinForms exclusively includes DrawMetafile method that takes in metafile images (.wmf, .emf) and adds it to your documents.
- **Read and Write DOCX/RTF files**
  Word for WinForms allows you to read and write DOCX and RTF files. You can also modify these files using the **C1Word** rich API.

Object Model Summary

C1Word provides a rich and powerful object model which is easily programmable. The C1.C1Word.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, Load, Count, Current, Hyperlink, ShapesWord2007Compatible
RTFPageSize Object
A0, A1, A4, A10, B1, B5, HalfLetter, Legal, Letter
Strings
ResourceManager, UICulture
Strings.Errors
ClosedDocument, UnsupportedRotationAngle

Quick Start: Word for WinForms

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

Step 1 of 3: Setting up the Application

In this step, you begin with creating a **New Project** where you create a **Windows Forms Application** in Visual Studio. After creating the application, add **C1Word** reference (dll) to your application.

Perform the following steps to start using the **Word** component:

1. Create a **New Project|Windows Forms Application** in Visual Studio.
2. Add the **C1Word** reference (dll) to your application.
3. Add the **C1WordDocument** component to the form.
4. Switch to the Code view and add the following namespace to the code:

Visual Basic

```
Imports C1.C1Word
```

C#

```
using C1.C1Word;
```

Step 2 of 3: Adding Simple Text

While you are still in code view, add the following lines of code beneath the **InitializeComponent()** method to create a Word document using **C1Word** component:

- **Visual Basic**

```
' create document
Dim c1Word As New C1WordDocument()
c1Word.Info.Title = "Simple Text Sample"

Dim font As New Font("Cambria", 24, FontStyle.Bold)
c1Word.AddParagraph("Hello! This is a C1Word component simple text example.", font, Color.MediumPurple)

c1Word.Save("simple.docx")
Process.Start("simple.docx")
```

- **C#**

```
// create document
C1WordDocument c1Word = new C1WordDocument();
c1Word.Info.Title = "Simple Text Sample";

Font font = new Font("Cambria", 24, FontStyle.Bold);
c1Word.AddParagraph("Hello! This is a C1Word component simple text example.", font, Color.MediumPurple);

c1Word.Save("Simple.docx");
Process.Start("Simple.docx");
```

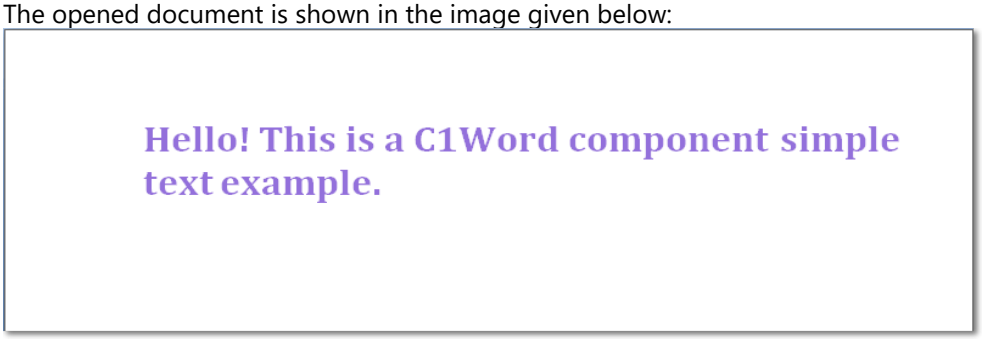
In the above code, a word document with **Simple Text Sample** title is created and some text is added to it using AddParagraph method. Lastly, the document is saved with the name, **Simple.docx**.

Step 3 of 3: Running the Application

In previous step, you added code 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:

Press **F5** to run the application.

The opened document is shown in the image given below:



Working with Word for WinForms

Word component 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** component in detail in the topics listed below.

Basic Level Working

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

Adding Simple Text

You can add simple text to a word document using C1Word. To add simple text, you need to use AddParagraph method and write your desired text in it. You can also set other properties, such as font style, family, color, and more for the text to be displayed in the word document. The implementation of adding a simple text to a word document is given in the code below:

Use the following code to create an object of C1WordDocument class to use AddParagraph method for adding text:

Visual Basic

```
Dim C1Word As New C1WordDocument()
```

C#

```
C1WordDocument C1Word = new C1WordDocument();
```

Add the following code to add text to word document you want to create:

Visual Basic

```
Dim font As New Font("Tahoma", 24, FontStyle.Italic)
C1Word.AddParagraph("Hello World!", font, Color.Blue)
```

C#

```
Font font = new Font("Tahoma", 24, FontStyle.Italic);
C1Word.AddParagraph("Hello World!", font, Color.Blue);
```

The document will look similar to the image below:

Inserting Pictures

You might need to insert images in your word document along with the text to enhance the overall appearance of your document. To do so, you can use the AddPicture method to insert picture in your word document and align it accordingly. The following code illustrates the use of AddPicture method and RtfHorizontalAlignment enum to set the horizontal alignment of the picture:

Visual Basic

```
Dim img As Image = New Bitmap(dlg.FileName)
C1Word.AddPicture(img, RtfHorizontalAlignment.Left)
```

C#

```
Image img = new Bitmap(dlg.FileName);
C1Word.AddPicture(img, RtfHorizontalAlignment.Left);
```

The document will look similar to the image 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, Bezier, Ellipse, Line, Pie, polygon, PolygonLine, and Rectangle. Use the following code to add graphics such as lines, rectangles, and beziers:

Visual Basic

```
' create document
Dim c1Word As New C1WordDocument()
c1Word.Info.Title = "Graphics primitives sample"

Dim sf As New StringFormat()
sf.Alignment = StringAlignment.Center
sf.LineAlignment = StringAlignment.Center
Dim rc = New RectangleF(250, 100, 150, 20)
Dim font As New Font("Arial", 14, FontStyle.Italic)
c1Word.DrawString(c1Word.Info.Title, font, Color.DeepPink, rc, sf)

c1Word.DrawLine(Pens.Green, 200, 190, 400, 190)

rc = New RectangleF(150, 150, 190, 80)
Using pen As New Pen(Brushes.Blue, 5F)
        c1Word.DrawRectangle(pen, rc)
End Using
c1Word.FillRectangle(Color.Gold, rc)
c1Word.ShapeFillOpacity(50)
c1Word.ShapeRotation(25)

rc = New RectangleF(300, 150, 80, 80)
c1Word.DrawEllipse(Pens.Red, rc)
c1Word.FillEllipse(Color.Pink, rc)
c1Word.ShapeFillOpacity(70)

Dim pts As PointF() = New PointF(3) {}
pts(0) = New PointF(200, 200)
pts(1) = New PointF(250, 300)
pts(2) = New PointF(330, 250)
pts(3) = New PointF(340, 140)
c1Word.DrawPolyline(Pens.BlueViolet, pts)

sf = New StringFormat()
sf.Alignment = StringAlignment.Center
sf.LineAlignment = StringAlignment.Far
sf.FormatFlags = sf.FormatFlags Or StringFormatFlags.DirectionVertical
rc = New RectangleF(450, 150, 25, 75)
font = New Font("Verdana", 12, FontStyle.Bold)
c1Word.DrawString("Vertical", font, Color.Black, rc, sf)

pts = New PointF(3) {}
pts(0) = New PointF(372, 174)
```

```
pts(1) = New PointF(325, 174)
pts(2) = New PointF(325, 281)
pts(3) = New PointF(269, 281)
c1Word.DrawBeziers(Pens.HotPink, pts)

Dim Sdlg As New SaveFileDialog()
Sdlg.FileName = "document"
Sdlg.Filter = "RTF files (*.rtf)|*.rtf|DOCX (*.docx)|*.docx"
Sdlg.ShowDialog()

c1Word.Save(Sdlg.FileName)
MessageBox.Show("Word document is saved successfully.")
```

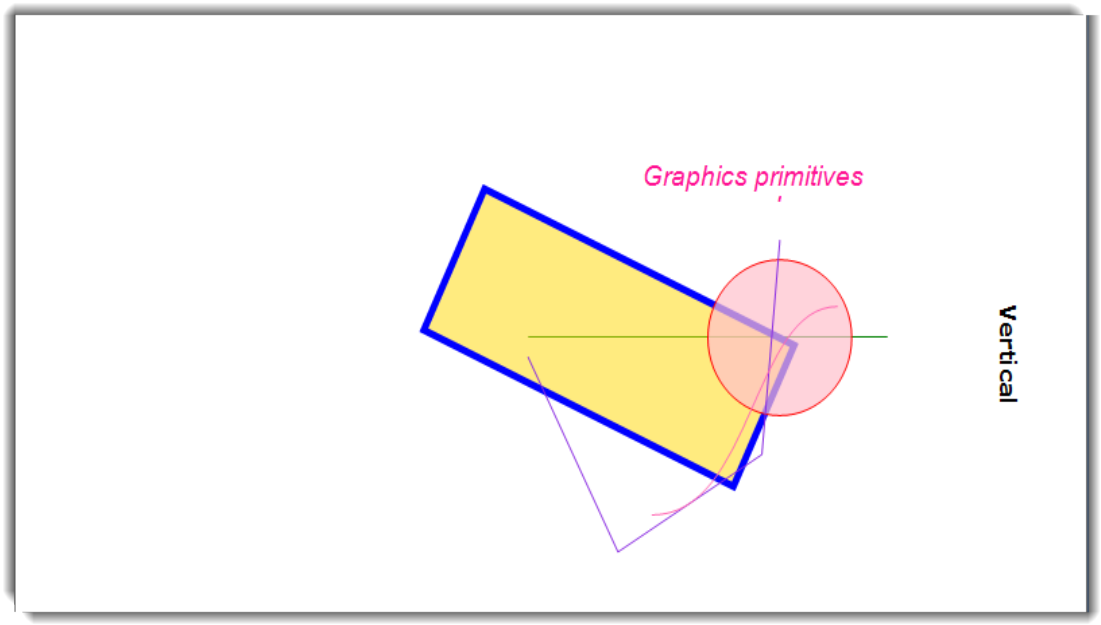
C#

```
// create document
C1WordDocument c1Word = new C1WordDocument();
c1Word.Info.Title = "Graphics primitives sample";

StringFormat sf = new StringFormat();
sf.Alignment = StringAlignment.Center;
sf.LineAlignment = StringAlignment.Center;
var rc = new RectangleF(250, 100, 150, 20);
Font font = new Font("Arial", 14, FontStyle.Italic);
c1Word.DrawString(c1Word.Info.Title, font, Color.DeepPink, rc, sf);

c1Word.DrawLine(Pens.Green, 200, 190, 400, 190);

rc = new RectangleF(150, 150, 190, 80);
using(Pen pen = new Pen(Brushes.Blue, 5.0 f)) {
  c1Word.DrawRectangle(pen, rc);
}
c1Word.FillRectangle(Color.Gold, rc);
c1Word.ShapeFillOpacity(50);
c1Word.ShapeRotation(25);

rc = new RectangleF(300, 150, 80, 80);
c1Word.DrawEllipse(Pens.Red, rc);
c1Word.FillEllipse(Color.Pink, rc);
c1Word.ShapeFillOpacity(70);

PointF[] pts = new PointF[4];
pts[0] = new PointF(200, 200);
pts[1] = new PointF(250, 300);
pts[2] = new PointF(330, 250);
pts[3] = new PointF(340, 140);
c1Word.DrawPolyline(Pens.BlueViolet, pts);

sf = new StringFormat();
sf.Alignment = StringAlignment.Center;
sf.LineAlignment = StringAlignment.Far;
sf.FormatFlags |= StringFormatFlags.DirectionVertical;
```

```
rc = new RectangleF(450, 150, 25, 75);
font = new Font("Verdana", 12, FontStyle.Bold);
c1Word.DrawString("Vertical", font, Color.Black, rc, sf);

pts = new PointF[4];
pts[0] = new PointF(372, 174);
pts[1] = new PointF(325, 174);
pts[2] = new PointF(325, 281);
pts[3] = new PointF(269, 281);
c1Word.DrawBeziers(Pens.HotPink, pts);

SaveFileDialog Sdlg = new SaveFileDialog();
Sdlg.FileName = "document";
Sdlg.Filter = "RTF files (*.rtf)|*.rtf|DOCX (*.docx)|*.docx";
Sdlg.ShowDialog();

c1Word.Save(Sdlg.FileName);
MessageBox.Show("Word document is saved successfully.");
```

In the above code, DrawLine, DrawRectangle, DrawEllipse, DrawPolyline, and DrawBeziers methods are used to draw graphics of different types such as, line, rectangle, bezier, and ellipse.

The document will look similar to the image below:



Adding Curves

You can add curves to your word document using **Word** component. DrawArc method allows you to add a curve to your document. The implementation of DrawArc method is given in the code below:

Visual Basic

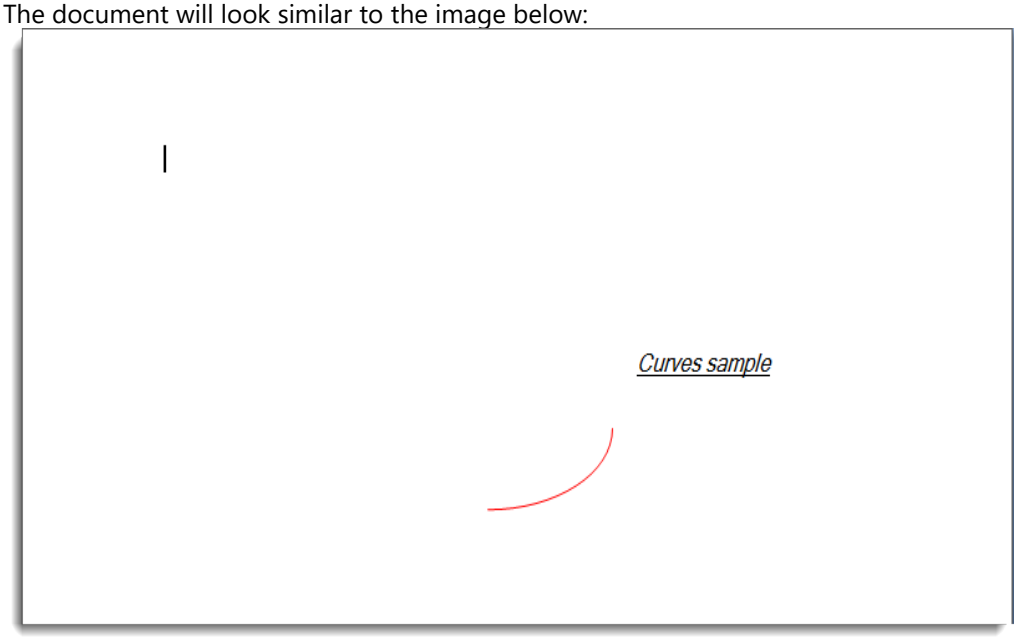
```
rc = New RectangleF(120, 100, 150, 80)
C1Word.DrawArc(Pens.Red, rc, 0, 90)
```

C#

```
rc = new RectangleF(120, 100, 150, 80);
C1Word.DrawArc(Pens.Red, rc, 0, 90);
```

This above code adds a red colors curve to a word document.

The document will look similar to the image 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** component:

Inserting Table

Table is used in word documents to present data in a well formatted form with the help of rows and columns. It is a very common practice to use tables in word documents to represent formatted data in rows and columns. **Word** component enables you to add table to your document using RtfTable class to create a table and RtfParagraph class to insert content in table.

Following code adds a table to a word document:

Visual Basic

```
Dim rows As Integer = 4
Dim cols As Integer = 2
Dim table As New RtfTable(rows, cols)
C1Word.Add(table)
For row As Integer = 0 To rows - 1
        For col As Integer = 0 To cols - 1
```

```
                Dim paragraph As New RtfParagraph()
                paragraph.Content.Add(New RtfString(String.Format("table
cell {0}:{1}.", row, col)))
                table.Rows(row).Cells(col).Content.Add(paragraph)
        Next
Next
```

C#

```
int rows = 4;
int cols = 2;
RtfTable table = new RtfTable(rows, cols);
C1Word.Add(table);
for (int row = 0; row < rows; row++)
{
        for (int col = 0; col < cols; col++)
        {
        RtfParagraph paragraph = new RtfParagraph();
        paragraph.Content.Add(new RtfString(string.Format("table cell
{0}:{1}.", row, col)));
        table.Rows[row].Cells[col].Content.Add(paragraph);
        }
}
```

Adding Complex Text

Adding text, images, tables and graphics to your document makes it more interactive. Using **Word** component, you can easily add title, images, tables, and graphics to your word document. Till now, you have seen how to add simple text to your document. But your document is actually a composite set of text, images, pictures, graphics etc which actually constitutes a complex text. The code given below adds title, images, tables, and graphics to your word document - all in one code:

Visual Basic

```
' add title
C1Word.AddParagraph(C1Word.Info.Title, New Font("Tahoma", 24,
FontStyle.Italic), Color.BlueViolet)

' add image
C1Word.AddParagraph("picture:", New Font("Courier New", 9,
FontStyle.Regular), Color.Black)
Dim img As New Bitmap(GetManifestResource("picture.jpg"))
C1Word.AddPicture(img, RtfHorizontalAlignment.Center)

' add table
C1Word.LineBreak()
Dim rows As Integer = 7
Dim cols As Integer = 2
Dim table As New RtfTable(rows, cols)
C1Word.Add(table)
```

```
For row As Integer = 0 To rows - 1
        For col As Integer = 0 To cols - 1
                Dim paragraph As New RtfParagraph()
                paragraph.Content.Add(New RtfString(String.Format("table
cell {0}:{1}.", row, col)))
                table.Rows(row).Cells(col).Content.Add(paragraph)
        Next
Next

' add graphics
C1Word.LineBreak()
C1Word.DrawLine(Pens.Green, 200, 90, 400, 90)

Dim rc = New RectangleF(150, 170, 90, 40)
Using pen As New Pen(Brushes.Blue, 5F)
        C1Word.DrawRectangle(pen, rc)
End Using
C1Word.FillRectangle(Color.Gold, rc)
C1Word.ShapeFillOpacity(50)
C1Word.ShapeRotation(25)

rc = New RectangleF(300, 120, 80, 80)
C1Word.DrawEllipse(Pens.Red, rc)
C1Word.FillEllipse(Color.Pink, rc)
C1Word.ShapeFillOpacity(70)
```

C#

```
            // add title
            C1Word.AddParagraph(C1Word.Info.Title, new Font("Tahoma",
24, FontStyle.Italic), Color.BlueViolet);

            // add image
            C1Word.AddParagraph("picture:", new Font("Courier New", 9,
FontStyle.Regular), Color.Black);
            Bitmap img = new Bitmap(GetManifestResource("picture.jpg"));
            C1Word.AddPicture(img, RtfHorizontalAlignment.Center);

            // add table
            C1Word.LineBreak();
            int rows = 7;
            int cols = 2;
            RtfTable table = new RtfTable(rows, cols);
            C1Word.Add(table);
            for (int row = 0; row < rows; row++)
            {
                for (int col = 0; col < cols; col++)
                {
                    RtfParagraph paragraph = new RtfParagraph();
                    paragraph.Content.Add(new
RtfString(string.Format("table cell {0}:{1}.", row, col)));
```

```
                    table.Rows[row].Cells[col].Content.Add(paragraph);
                }
            }

            // add graphics
            C1Word.LineBreak();
            C1Word.DrawLine(Pens.Green, 200, 90, 400, 90);

            var rc = new RectangleF(150, 170, 90, 40);
            using (Pen pen = new Pen(Brushes.Blue, 5.0f))
            {
                C1Word.DrawRectangle(pen, rc);
            }
            C1Word.FillRectangle(Color.Gold, rc);
            C1Word.ShapeFillOpacity(50);
            C1Word.ShapeRotation(25);

            rc = new RectangleF(300, 120, 80, 80);
            C1Word.DrawEllipse(Pens.Red, rc);
            C1Word.FillEllipse(Color.Pink, rc);
            C1Word.ShapeFillOpacity(70);
```

Adding Fonts

Using the right font makes the document stand out. You may want to create a document with a variety of font styles to display every text written in different font style appear distinct. You can use different fonts to your word document using the following code:

Visual Basic

```
' draw text in many fonts
Dim font As New Font("Tahoma", 9)
Dim ifc As New InstalledFontCollection()
For Each ff As FontFamily In ifc.Families
        ' create font
        Dim sample As Font = Nothing
        For Each fs As FontStyle In [Enum].GetValues(GetType(FontStyle))
                If ff.IsStyleAvailable(fs) Then
                        sample = New Font(ff.Name, 9, fs)
                        Exit For
                End If
        Next
        If sample Is Nothing Then
                Continue For
        End If

        ' show font
        C1Word.AddParagraph(ff.Name, font, Color.Black)
        C1Word.AddParagraph("The quick brown fox jumped over the lazy
dog. 1234567890!", sample, Color.Black)
```

```
        sample.Dispose()
Next
```

C#

```
// draw text in many fonts
Font font = new Font("Tahoma", 9);
InstalledFontCollection ifc = new InstalledFontCollection();
foreach (FontFamily ff in ifc.Families)
{
        // create font
        Font sample = null;
        foreach (FontStyle fs in Enum.GetValues(typeof(FontStyle)))
        {
                if (ff.IsStyleAvailable(fs))
                {
                        sample = new Font(ff.Name, 9, fs);
                        break;
                }
        }
        if (sample == null) continue;

        // show font
        C1Word.AddParagraph(ff.Name, font, Color.Black);
        C1Word.AddParagraph("The quick brown fox jumped over the lazy
dog. 1234567890!", sample, Color.Black);
        sample.Dispose();
}
```

Playing Metafiles

A Metafile is a type of file that can store variety of data. You can add .emf and .wmf images to your word documents. Following code shows the use of DrawMetafile method to draw a metafile image in a Word document:

Visual Basic

```
Dim img As Image = Metafile.FromFile(dlg.FileName)
C1Word.DrawMetafile(DirectCast(img, Metafile))
```

C#

```
Image img = Metafile.FromFile(dlg.FileName);
C1Word.DrawMetafile((Metafile)img);
```

Reading and Writing RTF Files

Word component allows reading and writing word document with RTF Documents with Rich Text format (*.rtf) extension. You can use Load method to read a document and Save method to write a document as illustrated in the following code:

Visual Basic

```
' load Word/RTF document
Dim C1Word As New C1WordDocument()
C1Word.Load(dlg.FileName)

' save RTF document
C1Word.Save("sample.rtf")
```

C#

```
// load Word/RTF document
C1WordDocument C1Word = new C1WordDocument();
C1Word.Load(dlg.FileName);

// save RTF document
C1Word.Save("sample.rtf");
```

Reading and Writing Docx Files

Word component allows reading and writing word document with Microsoft Word Open XML format (*.docx) extension. You can use the Load method to read a document and Save method to write a document as illustrated in the following code:

Visual Basic

```
' load Word/RTF document
Dim C1Word As New C1WordDocument()
C1Word.Load(dlg.FileName)

' save Word (docx) document
C1Word.Save("sample.docx")
```

C#

```
// load Word/RTF document
C1WordDocument C1Word = new C1WordDocument();
C1Word.Load(dlg.FileName);

// save Word (docx) document
C1Word.Save("sample.docx");
```

Word for WinForms 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\WinForms

The list of samples available for Word component is as follows:

Sample	Description
Word Samples	This sample demonstrates different tasks that can be accomplished using the Word component., such as adding Simple text, pictures, graphics, curves, table, complex text, fonts, metafiles, and Reading and Writing RTF and DOCX Files. It also shows how to switch between DOCX and RTF formats, and make the DOCX document compatible with Microsoft Word 2007.

API Reference

This section provides the API reference for Word for WinForms.