The Color Palette

Each Excel document has a single color palette that manages formatting colors and some drawing layer colors. In Excel binary files (.xls), this palette is limited to 56 colors, and all formatting colors must come from this palette. Excel 2007 and higher (.xlsx files) can have unlimited colors. However, it still has a legacy palette with 56 colors. These colors are the only colors that are displayed when opening the file in Excel 2003 or earlier.

When ExcelWriter opens a workbook that does not contain an explicitly defined palette, or when ExcelWriter creates a new workbook, it uses the same default color palette that Excel uses. ExcelWriter never automatically adds colors to the palette.

Because the behavior of the color palette changed in Excel between the Excel 2003 and 2007 file formats, it will have slightly different behavior in ExcelWriter when dealing with .xls files vs the newer file formats.

The color palette in binary (.xls) files

Workbook.Palette returns a Palette object that represents the workbook's color palette. A Color object represents a single color in the palette.

To replace a color in the palette with a new color, call either of the following methods:

- `Palette.SetColor(Color color, int red, int green, int blue)`

  This method takes the color to replace as a Color object, and the RGB values of the new color.

- `Palette.SetColorAt(int index, int red, int green, int blue)`

  This method takes the 0-based index of the color to replace, and the RGB values of the new color.

If you copy an object or cells with a custom color to another workbook, the custom color is replaced by the color in the corresponding position on the other workbook's color palette. To retain the custom color, either copy the customized color palette to the other workbook or change the corresponding color in the workbook.

If you are not sure that a specific color exists in the palette, and you want to assign the color to an element in the workbook, you can call Palette.GetClosestColor. Pass the method the red, green, and blue values of the ideal color. Palette.GetColor retrieves the color with the exact RGB value specified; if this color is not in the palette, an exception will be thrown.

A workbook's palette contains ten system colors that are fixed and cannot be changed by the user: SystemAutomaticFill, SystemAutomaticLine SystemBlack, SystemBlue, SystemCyan, SystemGreen, SystemMagenta, SystemRed, SystemWhite, and SystemYellow.

Drawing objects (pictures, shapes, comments, etc.) support custom RGB colors that may or may not be present in the palette. If you modify the color of an AutoShape (right-click the shape and select "Format AutoShape"), for example, and select a non-palette custom color, the change will not affect the workbook palette or other elements in the workbook to which the replaced color was assigned. To set custom values for shape fill and line colors, use Shape.SetCustomFillColor and Shape.SetCustomLineColor.

For more information on creating a custom palette for XLS files, refer to Excel's Color Palette Explained.

The color palette in xml (.xlsx, .xlsm, etc) files

Starting with the Excel 2007 file formats, a workbook is no longer limited to 56 colors. However, there is a legacy palette that contains 56 colors, and can be used to specify the colors used when opening the file in Excel 2003.

Unless you want to create files in the 2007 file formats that fully preserve their color information in older versions of Excel, you do not need to worry about the legacy palette. You can use Palette.GetColor or Palette.GetClosestColor to get any color, regardless of whether or not it is in the legacy palette. You may then use this Color object to set the color on any object in the workbook.

If you need to have the same colors when opened in any version of Excel, then you will need to only use colors that are in the palette. You can ensure that all of your colors are palette colors by using the Palette.GetColorAt and Palette.SetColorAt methods. However, you will be limited to 56 colors, due to the limitations in Excel 2003 and earlier.

Example of using the palette with binary (.xls) files

Wherever type Color is referenced in the following example, the type is SoftArtisans.OfficeWriter.ExcelWriter.Color. If you are using ExcelWriter in a class that imports the .NET framework class System.Drawing (added by default by when a new webform is added to a

VS.NET project), you will get an "ambiguous reference" error at compile-time because System.Drawing also contains a type Color. The solution is to either remove the System.Drawing import statement, or use the fully qualified name of ExcelWriter's Color object.

```
private void SetupColors(Workbook wb)
{

     //--- Workbook.Palette returns the Palette object
     Palette pal = wb.Palette;

     //--- GetClosestColor finds the color in the palette
     //--- that most closely matches the specified Red,
     //--- Green, and Blue values.
     Color clrDarkBlue = pal.GetClosestColor(0, 0, 255);
     Color clrTitleCell = pal.GetClosestColor(100, 100, 255);

     try
     {
          //--- GetColor will find the exact color you request.
          //--- If the color does not exist in the palette
          //--- it will throw IllegalArgumentException.
          Color clrDarkGreen = pal.GetColor(51, 153, 102);
     }
     catch
     {
          //--- The color you requested was not found in the palette
     }

     //--- GetColorAt gets the color from the palette at
     //--- the specified index.  This retrieves the second
     //--- color from the palette.
     Color clrComment = pal.GetColorAt(1);

     //--- SetColor and SetColorAt are for changing
     //--- the colors in the palette. Specify a Color to
     //--- change for SetColor, the index of the color for
     //--- SetColorAt, and then the RGB values for the desired
     //--- appearance.
     pal.SetColor(clrDarkBlue, 0, 0, 200);
     pal.SetColorAt(0, 0, 255, 0);

     //--- The Color object has built-in colors
     Color clrSysGreen = Color.SystemColor.Green;
     Color clrSysBlack = Color.SystemColor.Black;
}

private void StylizeHeaderCharacters(Workbook wb, Worksheet ws)
{
     Palette pal = wb.Palette;

     //--- Get a reference to cell A1
     Cell cellHeader = ws[0, 0];

     Color clrDarkBlue = pal.GetClosestColor(0, 0, 255);
     Color clrDarkGreen = pal.GetColor(51, 153, 102);

     //--- Stylize the first word, starting from the
     //--- first character, ending at the 10th
     cellHeader.GetCharacters(0, 10).Font.Size = 12;
```

```
//--- Format the font for the next two words in the cell
cellHeader.GetCharacters(11, 12).Font.Color = clrDarkBlue;
```

```
    cellHeader.GetCharacters(24, 12).Font.Color = clrDarkGreen;
}
```

Example of using colors with Excel 2007 file formats

This example shows how to use colors with 2007 file format workbooks. Because the 2007 file formats support unlimited colors, we do not need to do any work to set up the palette. We can use any colors we want without any additional overhead.

```
private void StylizeHeaderCharacters(Workbook wb, Worksheet ws)
{
    Palette pal = wb.Palette;

    //--- Get a reference to cell A1
    Cell cellHeader = ws[0, 0];

    //--- In the 2007 file formats, you are guaranteed to get the
    //--- color you ask for, even if it is not in the legacy palette.
    //--- Because you are getting the color directly, and not using
    //--- Palette.GetColorAt or Palette.SetColorAt, the color returned
    //--- will be a custom color, and will not be a palette color.
    Color clrDarkBlue = pal.GetClosestColor(0, 0, 255);
    Color clrDarkGreen = pal.GetColor(51, 153, 102);

    //--- Stylize the first word, starting from the
    //--- first character, ending at the 10th
    cellHeader.GetCharacters(0, 10).Font.Size = 12;

    //--- Format the font for the next two words in the cell
    cellHeader.GetCharacters(11, 12).Font.Color = clrDarkBlue;
    cellHeader.GetCharacters(24, 12).Font.Color = clrDarkGreen;
}
```

Example of using 2007 file formats with the legacy palette

This example shows how to use the legacy palette in 2007 file formats to ensure that if the file is opened in Excel 2003, the colors will remain the same. Note that in this example, we always use GetColorAt or SetColorAt to get color objects. This ensures that the colors will be in the legacy palette.

```
private void StylizeHeaderCharacters(Workbook wb, Worksheet ws)
{
    Palette pal = wb.Palette;

    //--- Get a reference to cell A1
    Cell cellHeader = ws[0, 0];

    //--- By using SetColorAt, we ensure that the colors that
    //--- are returned are part of the legacy palette. We need
    //--- to use separate palette indices for each color we want
    //--- in the workbook, limiting us to 56 colors.
    Color clrDarkBlue = pal.SetColorAt(0, 0, 0, 255);
    Color clrDarkGreen = pal.SetColorAt(1, 51, 153, 102);

    //--- Stylize the first word, starting from the
    //--- first character, ending at the 10th
    cellHeader.GetCharacters(0, 10).Font.Size = 12;

    //--- Format the font for the next two words in the cell
    cellHeader.GetCharacters(11, 12).Font.Color = clrDarkBlue;
    cellHeader.GetCharacters(24, 12).Font.Color = clrDarkGreen;
}
```