Overview

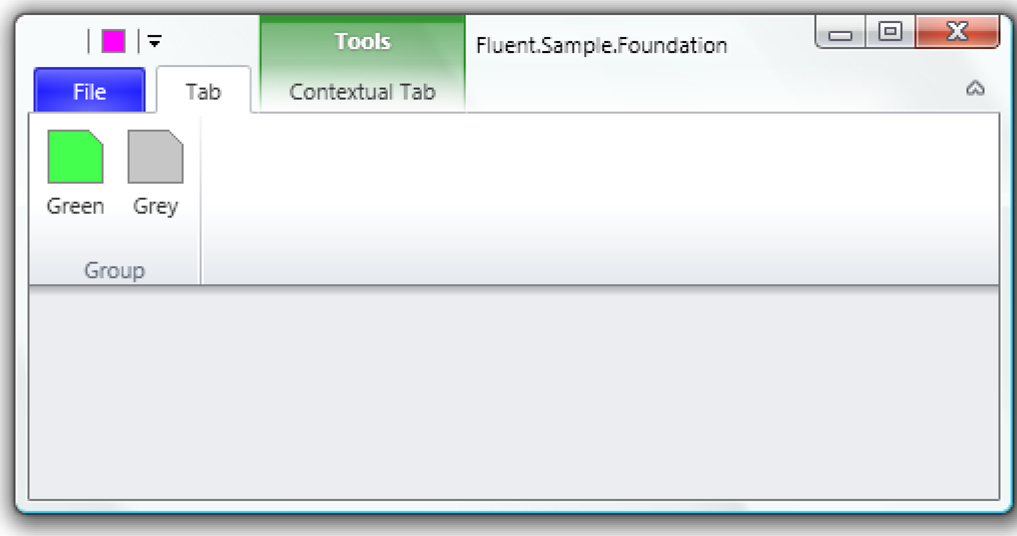
Fluent Ribbon Control Suite is a library that implements Microsoft® Office Fluent™ user interface for the Windows Presentation Foundation (WPF).

This document covers the main features of this framework and highlights how to use it. It is a good document to make **fast inside in ribbon development**. To get more practice you can download **samples** from http://fluent.codeplex.com

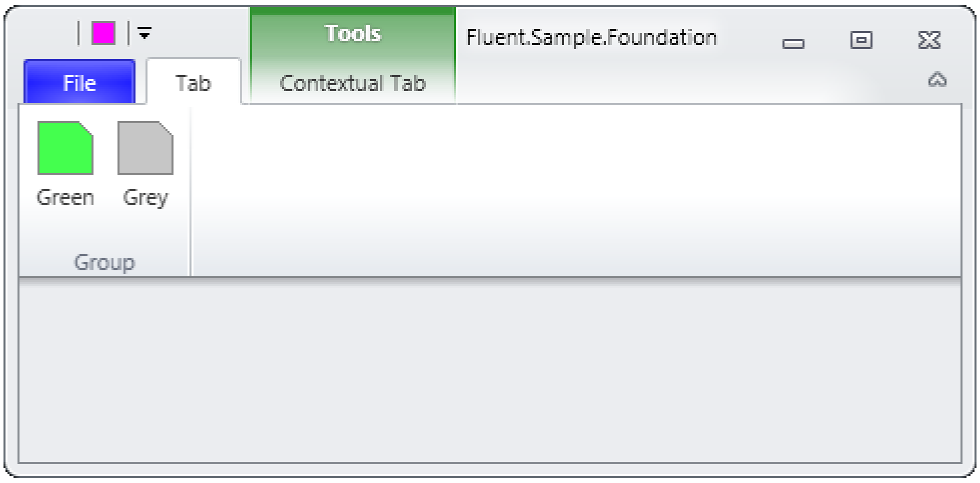
Any portion of this document can be reprinted without restrictions except the reference to original is required.

Foundation

Let's learn how to create a simple window with basic elements, such as RibbonTabItem, RibbonGroupBox, Quick Access Toolbar and so on.



As you can see, the window above is not usual WPF window: it has aero style and some elements located in the title bar area. To achieve this you need use Fluent.RibbonWindow. RibbonWindow is designed to provide proper office-like glass style. RibbonWindow automatically will use special non-DWM style in case of Windows XP or basic Windows 7/Vista theme (see below).



One of the way to create RibbonWindow is to create regular WPF window and change System.Windows.Window to Fluent.RibbonWindow in code and XAML. You can still use regular System.Windows.Window as you need. Your code and XAML will something like this:

```
/// <summary>
/// Represents the main window of the application
/// </summary>
public partial class Window : RibbonWindow
{
    /// <summary>
    /// Default constructor
    /// </summary>
    public Window()
    {
        InitializeComponent();
    }
}
```

```
<Fluent:RibbonWindow x:Class="Fluent.Sample.Foundation.Window"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:Fluent="clr-namespace:Fluent;assembly=Fluent"
    Title="Fluent.Sample.Foundation" Width="500" Height="250" >
    <Grid>
    </Grid>
</Fluent:RibbonWindow>
```

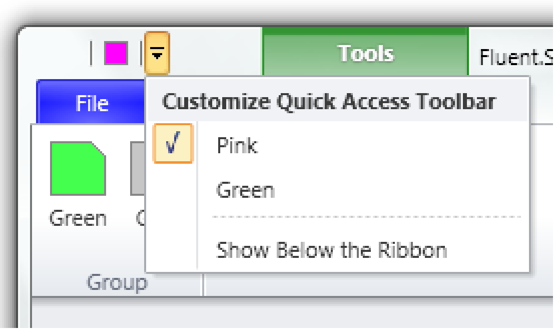
BEAWARE: You need also add one of themes in App.xaml, for example:

```
<Application.Resources>
   <!--Attach Default Fluent Control's Theme-->
   <ResourceDictionary Source="pack://application:,,,/Fluent;
                               Component/Themes/Office2010/Silver.xaml" />
</Application.Resources>
```

Let’s add Ribbon in the Window

```
<Fluent:Ribbon>
    <!--Backstage-->
    <Fluent:Ribbon.Menu>
      <Fluent:Backstage>
      </Fluent:Backstage>
    </Fluent:Ribbon.Menu>

    <!--Tabs-->
    <Fluent:RibbonTabItem Header="Tab">
        <Fluent:RibbonGroupBox Header="Group">
             <Fluent:Button Name="buttonGreen" Header="Green"
                            Icon="Images\Green.png"
                            LargeIcon="Images\GreenLarge.png" />
             <Fluent:Button Name="buttonGray" Header="Grey" Icon="Images\Gray.png"
                            LargeIcon="Images\GrayLarge.png" />
         </Fluent:RibbonGroupBox>
    </Fluent:RibbonTabItem>
</Fluent:Ribbon>
```

The code above produces the ribbon with single tab, group and two buttons. Quick Access Toolbar is a toolbar with shortcuted controls. You can add any Fluent-based control using context menu (a control has to implement IQuickAccessToolbarItem interface). You can also pin controls to Quick Access Toolbar menu.

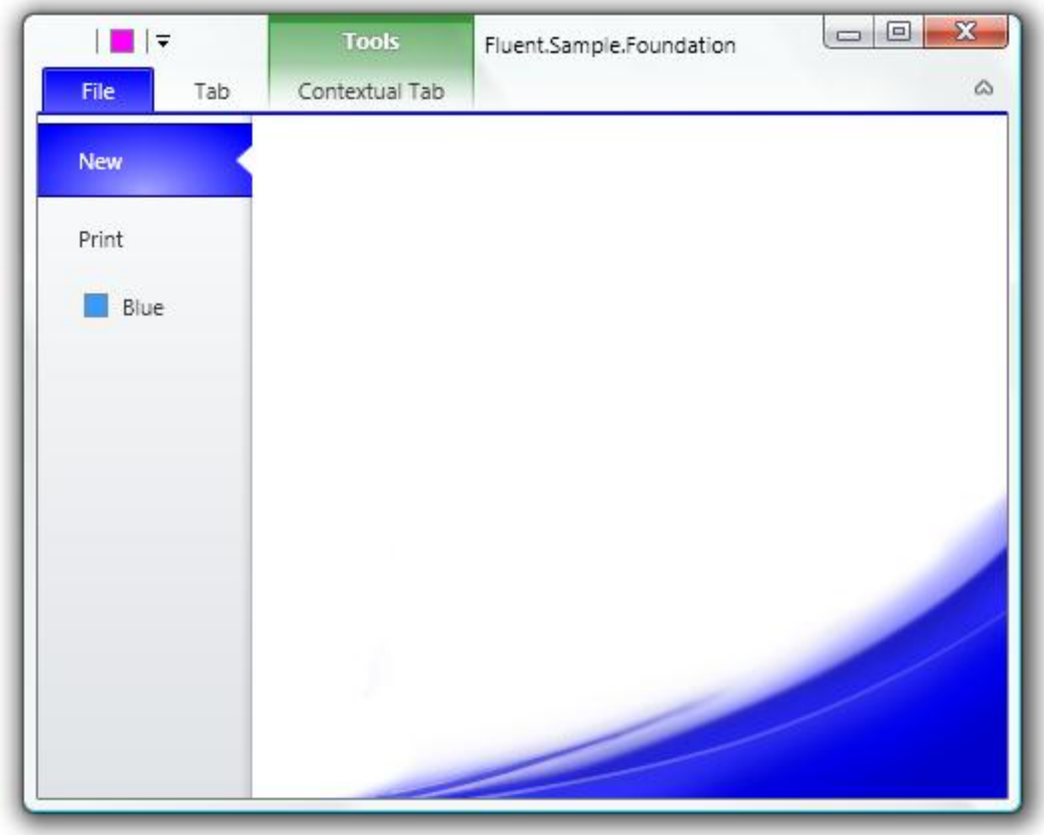


You can add pinned Quick Access Items named QuickAccessMenuItem to collection Ribbon.QuickAccessItems. To associate target element you may bind it to QuickAccessMenuItem.Target property or just set the content.

```
<!--Quick Access Toolbar Items-->
<Fluent:Ribbon.QuickAccessItems>
     <!--Use Content or Target Property to set QAT item-->
     <Fluent:QuickAccessMenuItem IsChecked="true">
       <Fluent:Button Header="Pink" Icon="Images\Pink.png" />
   </Fluent:QuickAccessMenuItem>
   <!--You Can Just Bind with Any Control-->
   <Fluent:QuickAccessMenuItem Target="{Binding ElementName=buttonGreen}"/>
</Fluent:Ribbon.QuickAccessItems>
```

The button “File” in left-top corner opens empty backstage, let’s fill Backstage with items:

```
<!--Backstage-->
<Fluent:Ribbon.Menu>
  <Fluent:Backstage>
    <Fluent:BackstageTabControl>
      <Fluent:BackstageTabItem Header="New"/>
      <Fluent:BackstageTabItem Header="Print"/>
      <Fluent:Button Header="Blue" Icon="Images\Blue.png"/>
    </Fluent:BackstageTabControl>
  </Fluent:Backstage>
</Fluent:Ribbon.Menu>
```



The last step is to add contextual tab. Contextual tab is visible when a particular object in an app selected. Contextual tabs cannot exist out of contextual tab group, so we need to create contextual tab group (RibbonContextualTabGroup) and bind a tab to this group. RibbonContextualTabGroup need to be added to Ribbon.ContextualGroups collection:

```
<!--Contextual Tab Groups-->
<Fluent:Ribbon.ContextualGroups>
    <Fluent:RibbonContextualTabGroup Header="Tools" Visibility="Visible"
            x:Name="toolsGroup" Background="Green" BorderBrush="Green" />
</Fluent:Ribbon.ContextualGroups>
```

And associate a tab to this group:

```
<!--Contextual Tabs-->
<Fluent:RibbonTabItem Header="CT" Group="{Binding ElementName=toolsGroup}"/>
```

RibbonContextualTabGroup is not visible by default. To show or hide contextual tab you must set RibbonContextualTabGroup.Visibility property to Visible or Collapsed.

Resizing

All RibbonGroupBox can be in four states: Large->Middle->Small->Collapsed. By default RibbonGroupBox's are in Large state. When a RibbonGroupBox changes its state it changes size of all its controls.

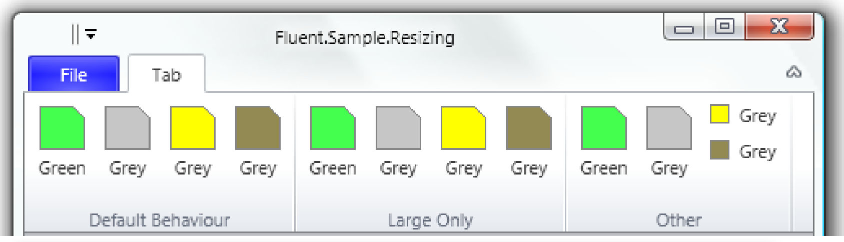
RibbonTabItem has ReduceOrder property. This property defines order of group to reduce. You can enumerate group names from the last to first to reduce.

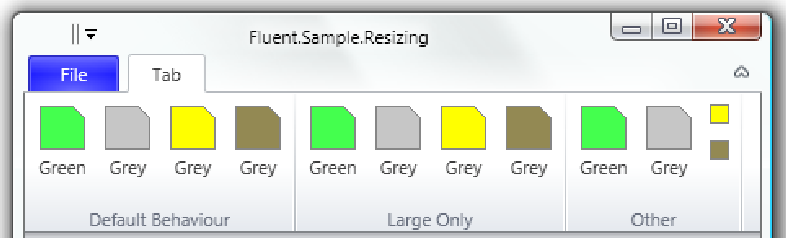
All ribbon contols (Buttons, DropDownButtons, Spinners and so on) have SizeDefinition property. You can define which size will be used when the group will be in the particular state. For example, if you set SizeDefinition = "Middle, Middle, Small", that means:

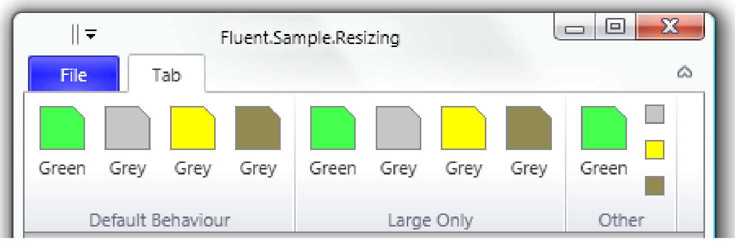
Large State of the group -> Middle size of the control
Middle State of the group -> Middle size of the control
Small State of the group -> Small size of the control

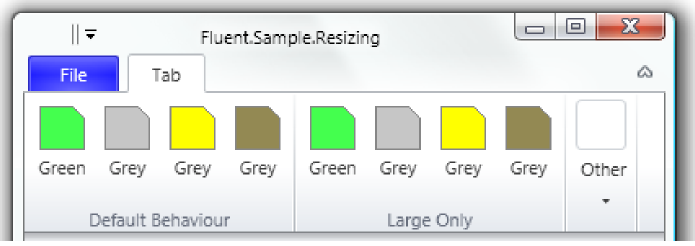
```
<Fluent:RibbonTabItem ReduceOrder="Default,Default,Default,
                                    Large,Large,Large,
                                    Other,Other,Other" ...>

 <!--By default ReduceOrder="Large, Middle, Small"-->
 <Fluent:RibbonGroupBox Name="Default" Header="Default Behaviour">
    <Fluent:Button ... />
    <Fluent:Button ... />
    <Fluent:Button ... />
    <Fluent:Button ... />
 </Fluent:RibbonGroupBox>

 <!--You can use short form
  (for ex, "Middle" is equal "Middle,Middle,Middle")-->
 <Fluent:RibbonGroupBox Name="Large" Header="Large Only">
      <Fluent:Button SizeDefinition="Large" .../>
      <Fluent:Button SizeDefinition="Large" .../>
      <Fluent:Button SizeDefinition="Large" .../>
   <Fluent:Button SizeDefinition="Large" .../>
</Fluent:RibbonGroupBox>

<Fluent:RibbonGroupBox Name="Other" Header="Other">
  <Fluent:Button SizeDefinition="Large, Large, Large" Icon="Images\Green.png" ../>
  <Fluent:Button SizeDefinition="Large, Large, Small" Icon="Images\Gray.png" ../>
  <Fluent:Button SizeDefinition="Middle, Small, Small" Icon="Images\Yellow.png" ..
  <Fluent:Button SizeDefinition="Middle, Small, Small" Icon="Images\Brown.png" ..
</Fluent:RibbonGroupBox>

</Fluent:RibbonTabItem>
```

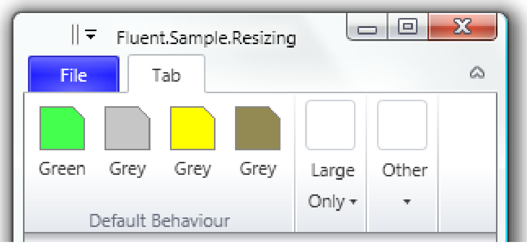
The pseudo code above must produce the following reducing behavior:

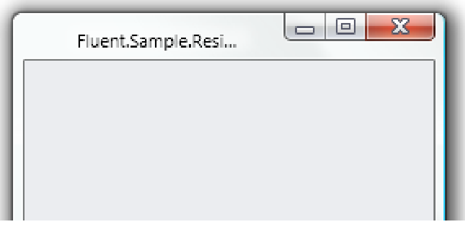

Fluent.Sample.Resizing
File
Tab
Green
Grey
Grey
Grey
Default Behaviour
Green
Grey
Grey
Grey
Large Only
Green
Grey
Grey
Grey
Other


Fluent.Sample.Resizing
File
Tab
Green
Grey
Grey
Grey
Default Behaviour
Green
Grey
Grey
Grey
Large Only
Green
Grey
Other


Fluent.Sample.Resizing
File
Tab
Green
Grey
Grey
Grey
Default Behaviour
Green
Grey
Grey
Grey
Large Only
Green
Other


Fluent.Sample.Resizing
File
Tab
Green
Grey
Grey
Grey
Default Behaviour
Green
Grey
Grey
Grey
Large Only
Other


Fluent.Sample.Resizing
File
Tab
Green
Grey
Grey
Grey
Default Behaviour
Large Only
Other

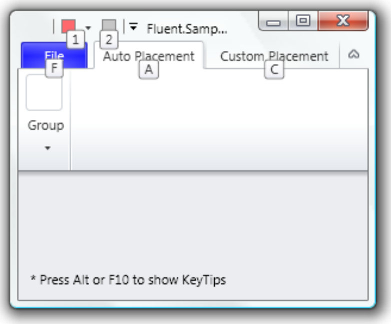

Fluent.Sample.Resi...

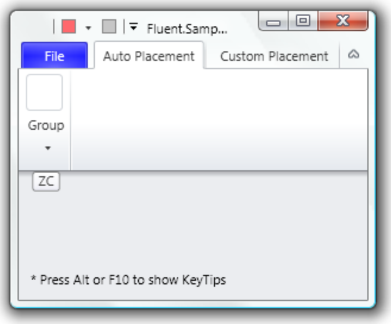
KeyTips

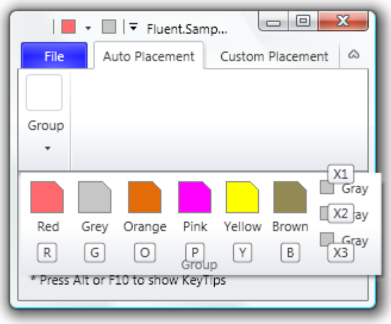
KeyTips provide for users ability to interact with Ribbon using keyboard. To start process user must press Alt or F10. KeyTips are shown upon controls.

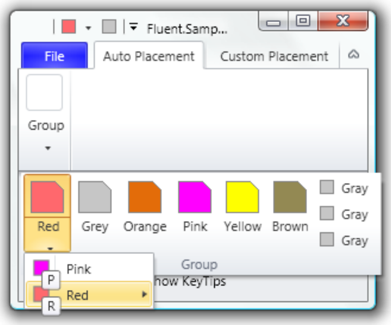
To make KeyTips work it is enough to set attached property Fluent:KeyTip.Keys to the target control and the ribbon will arrange and show the keytips automatically. It is possible to set KeyTips to menu and/or submenu items. Also you need to set KeyTip for groups to open them while they are collapsed.

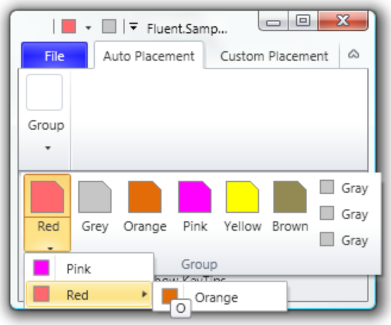
```
<Fluent:RibbonGroupBox Fluent:KeyTip.Keys="ZC" ... >
  <Fluent:SplitButton Fluent:KeyTip.Keys="R"  ... >
     <Fluent:MenuItem Fluent:KeyTip.Keys="P"  ... />
     <Fluent:MenuItem Fluent:KeyTip.Keys="R"  ... >
        <Fluent:MenuItem Fluent:KeyTip.Keys="O"  ... />
     </Fluent:MenuItem>
  </Fluent:SplitButton>
  ...
```







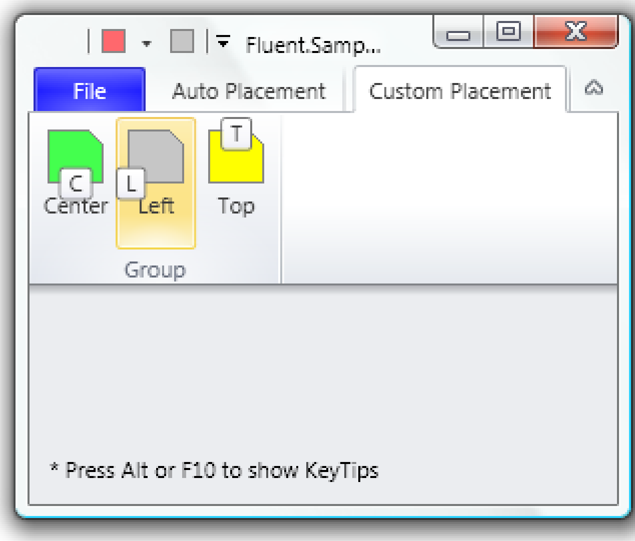




As you can see items in Quick Access Toolbar are keytipped automatically. Also KeyTips are placed well automatically. However, there are cases when custom placement is required. In this case you have to set Fluent:KeyTip.AutoPlacement to false and use additional attached properties:

Property	Description
KeyTip.AutoPlacement	True by default. You need set false to switch on custom placement
KeyTip.HorizontalAlignment	Horizontal alignment relative to keytiped control
KeyTip.VerticalAlignment	Vertical alignment relative to keytiped control
KeyTip.Margin	Margin to offset KeyTip

```
<Fluent:RibbonGroupBox Header="Group">
  <Fluent:Button Text="Center" LargeIcon="Images\GreenLarge.png"
      Fluent:KeyTip.AutoPlacement="False"
      Fluent:KeyTip.HorizontalAlignment="Center"
      Fluent:KeyTip.VerticalAlignment="Center"
      Fluent:KeyTip.Keys="C" />
  <Fluent:Button Text="Left" LargeIcon="Images\GrayLarge.png"
      Fluent:KeyTip.AutoPlacement="False"
      Fluent:KeyTip.HorizontalAlignment="Left"
      Fluent:KeyTip.VerticalAlignment="Center"
      Fluent:KeyTip.Keys="L" />
  <Fluent:Button Text="Top" LargeIcon="Images\YellowLarge.png"
      Fluent:KeyTip.AutoPlacement="False"
      Fluent:KeyTip.HorizontalAlignment="Center"
      Fluent:KeyTip.VerticalAlignment="Top"
      Fluent:KeyTip.Keys="T"/>
</Fluent:RibbonGroupBox>
```

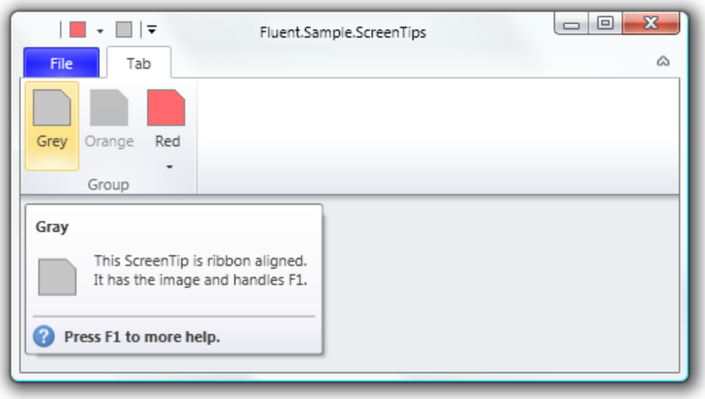


* Press Alt or F10 to show KeyTips

ScreenTips

To use ScreenTip you have to create Fluent.ScreenTip instance and set to ToolTip property:

```
<Fluent:Button ... >
    <Fluent:Button.ToolTip>
         <Fluent:ScreenTip Title="Gray"
              HelpTopic="Help for Gray ScreenTip"
              Image="Images\GrayLarge.png"
              Text="This ScreenTip is ribbon aligned. &#x0a;
                     It has the image and handles F1."/>
    </Fluent:Button.ToolTip>
</Fluent:Button>
```



Gray

This ScreenTip is ribbon aligned.
It has the image and handles F1.

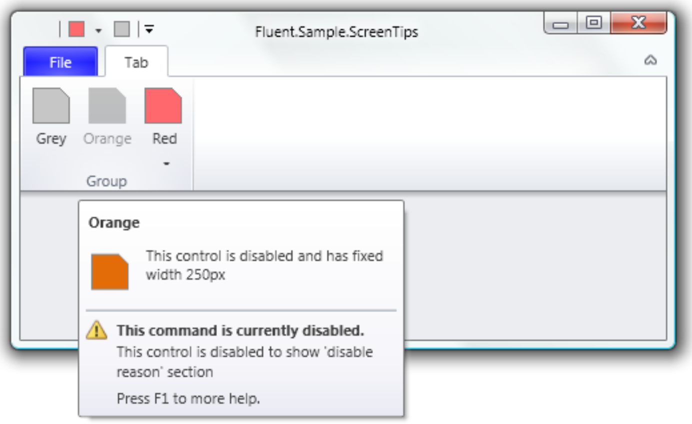
Press F1 to more help.

ScreenTip has a unique feature to invoke contextual help when it is in open state. To handle contextual help you must set ScreenTip.HelpTopic in XAML and subscribe to ScreenTip.HelpPressed event. Be aware, that the event is static, so you must manually unsubscribe from it when it is appropriate to avoid memory leaks or subscribe application's lifetime object as shown below:

```
public partial class Application : System.Windows.Application
{
   void OnStartup(object sender, StartupEventArgs e)
   {
       ScreenTip.HelpPressed += OnScreenTipHelpPressed;
   }

   /// <summary>
   /// Handles F1 pressed on ScreenTip with help capability
   /// </summary>
   /// <param name="sender">Sender</param>
   /// <param name="e">Arguments</param>
   static void OnScreenTipHelpPressed(object sender, ScreenTipHelpEventArgs e)
   {
       // Show help according the given help topic
       // (here just show help topic as string)
       MessageBox.Show(e.HelpTopic.ToString());
   }
}
```

ScreenTip shows on disabled Fluent-based controls automatically. Moreover you can set text which should be shown when the target control is disabled.

```
<Fluent:Button IsEnabled="False" ... >
    <Fluent:Button.ToolTip>
        <Fluent:ScreenTip Title="Orange" Width ="250"
            Image="Images\OrangeLarge.png"
            Text="This control is disabled and has fixed width 250px"
            HelpTopic="Help for Orange ScreenTip"
            DisableReason="This control is disabled
                           to show 'disable reason' section"/>
    </Fluent:Button.ToolTip>
</Fluent:Button>
```

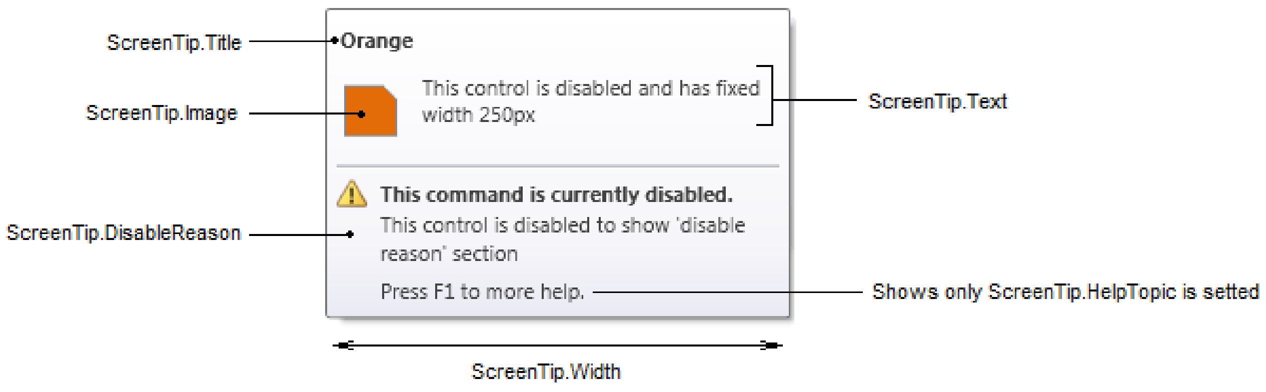


Orange

This control is disabled and has fixed width 250px

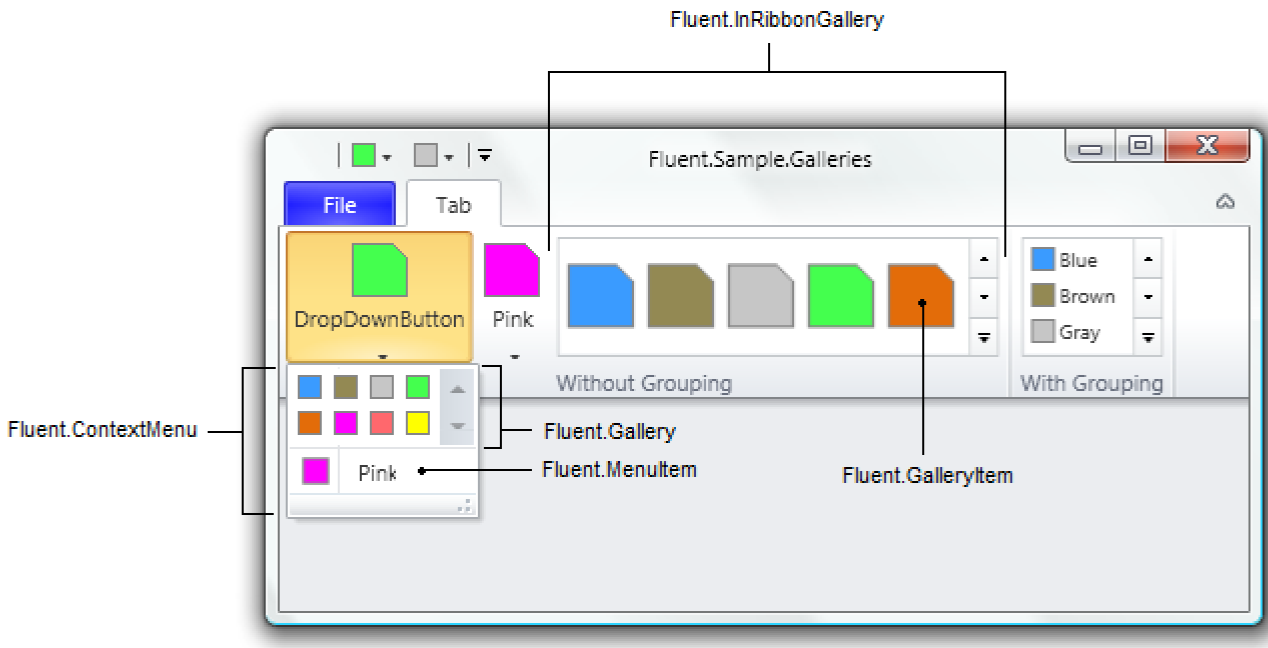
This command is currently disabled.
This control is disabled to show 'disable reason' section
Press F1 to more help.

You can find main properties of ScreenTip in the table below.


ScreenTip.Title
ScreenTip.Image
ScreenTip.DisableReason
Orange
This control is disabled and has fixed width 250px
ScreenTip.Text
This command is currently disabled.
This control is disabled to show 'disable reason' section
Press F1 to more help.
Shows only ScreenTip.HelpTopic is setted
ScreenTip.Width

Property	Description
ScreenTip.Title	The title of the ScreenTip
ScreenTip.Text	The text of the ScreenTip
ScreenTip.Image	Image
ScreenTip.Width	Set this property if you want to make fixed sized ScreenTip
ScreenTip.DisableReason	If target control is disabled this text will be shown to user
ScreenTip.HelpTopic	Set this property and subscribe to ScreenTip.HelpPressed to execute your contextual help.

Galleries



Fluent.Gallery is designed to be in ContextMenu. Actually, Gallery control is similar to ListBox. Below we insert Gallery in DropDownButton's context menu as well as one MenuItem.

```
<Fluent:DropDownButton Header="DropDownButton" ... >
    <Fluent:Gallery ItemsSource ="{Binding DataItems}"
                    ItemTemplate="{DynamicResource middleDataItemTemplate}" />
    <Fluent:MenuItem Icon="Images\Pink.png" Header="Pink"/>
</Fluent:DropDownButton>
```

In the last example we have set ItemsSource and ItemTemplate to fill gallery with items. Though, you may add children of the Gallery explicitly:

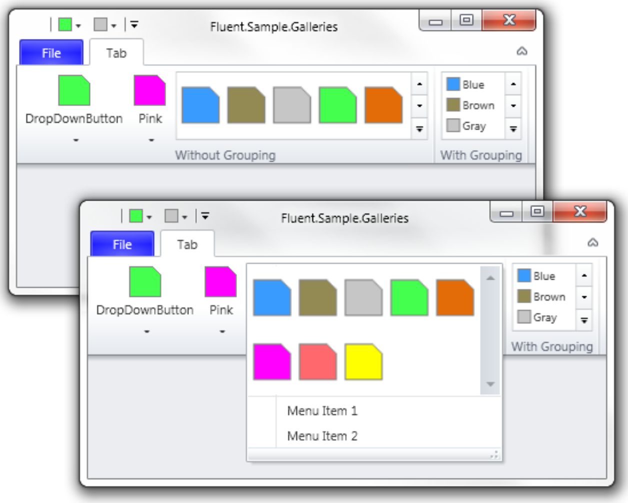
```
<Fluent:DropDownButton Text="Pink" ... >
    <Fluent:Gallery>
        <Image Source="Images\RedLarge.png" Stretch="None"/>
        <Image Source="Images\GreenLarge.png" Stretch="None"/>
        <Image Source="Images\BlueLarge.png" Stretch="None"/>
    </Fluent:Gallery>
</Fluent:DropDownButton>
```

Some of important properties of Gallery (and InRibbonGallery) are listed below:

Property	Description
GroupBy	Provides simplified way to group items. Read more about grouping feature below
Orientation	Defines whether items must be arranged in horizontal or vertical direction
Filters	Represents collection of filters. Read more about filtering feature below
SelectedFilter	Is the currently selected filter. Read more about filtering feature below
Selectable	Determines whether the gallery has ability to select an item
SelectedIndex	Is the index of the selected item or -1 in case of no selection presents
SelectedItem	Is the selected item or null in case of no selection is presented
ItemWidth	Defines the fixed width of GalleryItem in the Gallery (Double.NaN by default)
ItemHeight	Defines the fixed height of GalleryItem in the Gallery (Double.NaN by default)

Fluent.InRibbonGallery is similar to use:

```
<Fluent:InRibbonGallery MinItemsInRow="3" MaxItemsInRow="8"
                        ItemWidth="40" ItemHeight="56"
                        ItemsSource ="{Binding DataItems}"
                        ItemTemplate="{DynamicResource largeDataItemTemplate}"
                        ResizeMode="Both"/>
```

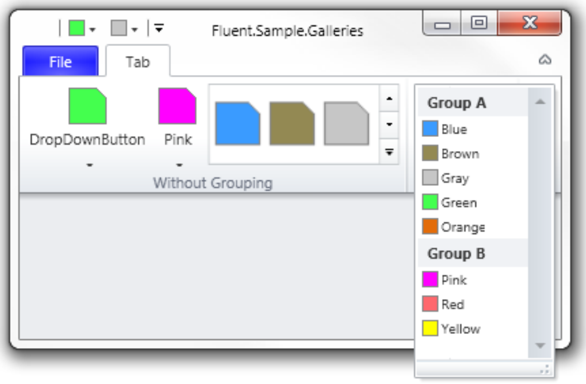


Set MinItemsInRow and MaxItemsInRow to define minimal and maximal width of a InRibbonGallery. By default InRibbonGallery appears in maximal width. Use special syntax in RibbonTabItem.ReduceOrder to reduce InRibbonGallery. To reduce all InRibbonGalleries in a particular group you have to write name of this group in brackets:

```
<Fluent:RibbonTabItem ReduceOrder="(A),(A),(A),(A),(A),(B),(B),(B),(B)" ... >
    <Fluent:RibbonGroupBox Header="Without Grouping" Name="A">
      ...
```

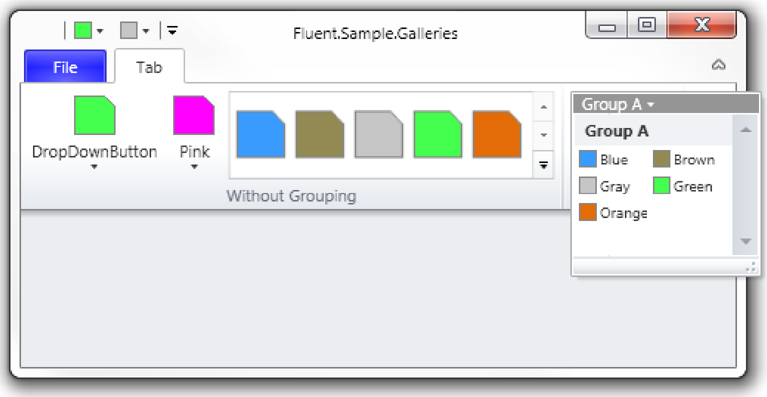
Gallery items can be grouped. To group items in Gallery (or InRibbonGallery) you may use GroupBy for simplified grouping method or use GroupByAdvanced to introduce your own method. Note in the example below each data item has Group property. You can use Tag property if you add controls in Items.

```
<Fluent:InRibbonGallery MinItemsInRow="1" MaxItemsInRow="5"
                        ItemWidth="50" ItemHeight="18"
                        ItemsSource ="{Binding DataItems}"
                        ItemTemplate="{DynamicResource middleDataItemTemplate}"
                        GroupBy="Group" ResizeMode="Both"/>
```



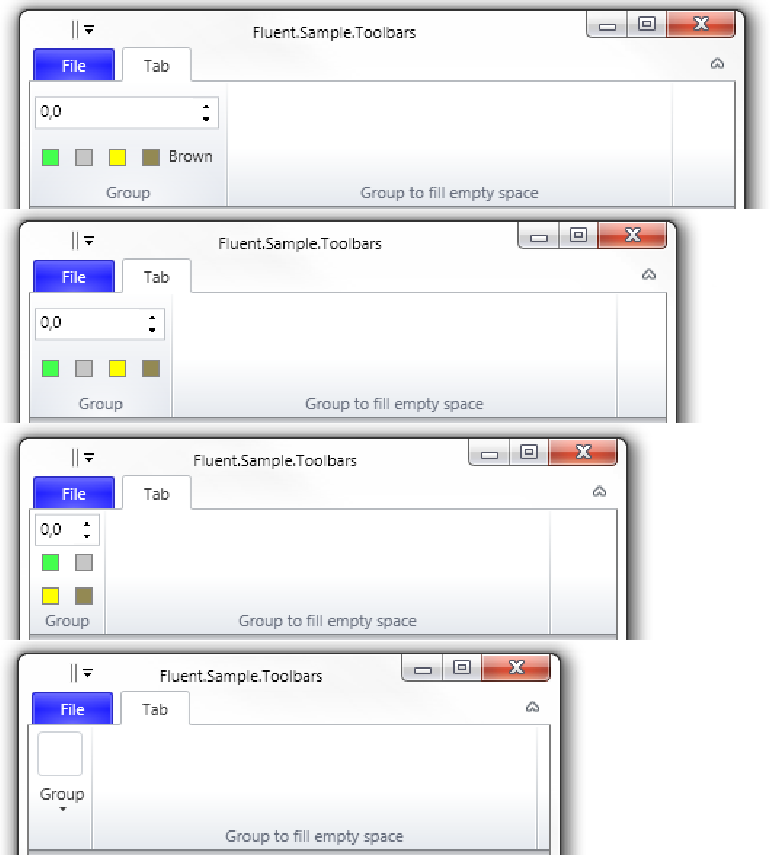
Other unique feature of Gallery (and InRibbonGallery) is filtering. You can add filters and user will be able to filter items using them. Filter is one or more group. To enable grouping just add the following lines:

```
<Fluent:InRibbonGallery MinItemsInRow="1" MaxItemsInRow="5"
                        ItemWidth="50" ItemHeight="18"
                        ItemsSource ="{Binding DataItems}"
                        ItemTemplate="{DynamicResource middleDataItemTemplate}"
                        GroupBy="Group" ResizeMode="Both">

      <!--Filters-->
      <Fluent:InRibbonGallery.Filters>
           <Fluent:GalleryGroupFilter Title="All" Groups="Group A,Group B"/>
           <Fluent:GalleryGroupFilter Title="Group A" Groups="Group A"/>
           <Fluent:GalleryGroupFilter Title="Group B" Groups="Group B"/>
      </Fluent:InRibbonGallery.Filters>

</Fluent:InRibbonGallery>
```



Toolbars

Fluent Ribbon Control Suite provides capability to make toolbar-like layout. Usually this is used in legacy-style controls such as font group in Microsoft Word. This layout is more complex and difficult to code it. Let’s try to make something like the following example:



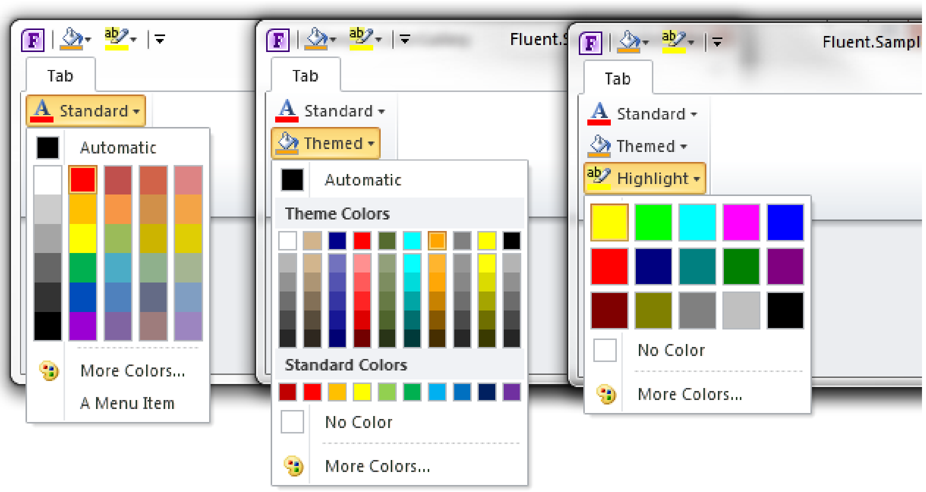
We see here spinner and four buttons. In the large size controls are placed in two rows, brown button is in middle size. In middle size brown button becomes small and in small size controls are relocated in three rows.

The main concept of RibbonToolBar control is you add controls and add layout definitions to each size of the toolbar. Layout definition (RibbonToolBarLayoutDefinition) is definition where controls are located. Each layout definition has one or more rows (RibbonToolBarRow) with control definitions or control group definitions (RibbonToolBarControlDefinition, RibbonToolBarControlGroup). Control group must have one or more control definitions inside, it will be separated with special separator or have other visual representation which is defined in the particular style. It’s required to set Target property of RibbonToolBarControlDefinition to name of one of toolbar controls. Also you can define Size (Small, Middle or Large) and Width of control. By default size and width is Small and Auto (Double.NaN).

```
<Fluent:RibbonToolBar>
<!--ToolBar Layout Definitions-->
<Fluent:RibbonToolBar.LayoutDefinitions>

  <!--Large Size of the RibbonToolBar-->
  <Fluent:RibbonToolBarLayoutDefinition Size="Large">
    <Fluent:RibbonToolBarRow>
      <Fluent:RibbonToolBarControlDefinition Target="spinner" Width="127"/>
    </Fluent:RibbonToolBarRow>
    <Fluent:RibbonToolBarRow>
      <Fluent:RibbonToolBarControlDefinition Target="buttonGreen" />
      <Fluent:RibbonToolBarControlDefinition Target="buttonGray" />
      <Fluent:RibbonToolBarControlDefinition Target="buttonYellow" />
      <Fluent:RibbonToolBarControlDefinition Target="buttonBrown" Size="Middle"/>
    </Fluent:RibbonToolBarRow>
  </Fluent:RibbonToolBarLayoutDefinition>


  <!--Large Size of the RibbonToolBar-->
  <Fluent:RibbonToolBarLayoutDefinition Size="Middle">
    <Fluent:RibbonToolBarRow>
      <Fluent:RibbonToolBarControlDefinition Target="spinner" Width="90"/>
    </Fluent:RibbonToolBarRow>
    <Fluent:RibbonToolBarRow>
      <Fluent:RibbonToolBarControlDefinition Target="buttonGreen" />
      <Fluent:RibbonToolBarControlDefinition Target="buttonGray" />
      <Fluent:RibbonToolBarControlDefinition Target="buttonYellow" />
      <Fluent:RibbonToolBarControlDefinition Target="buttonBrown"/>
    </Fluent:RibbonToolBarRow>
  </Fluent:RibbonToolBarLayoutDefinition>

  <!--Middle Size of the RibbonToolBar-->
  <Fluent:RibbonToolBarLayoutDefinition Size="Small">
    <Fluent:RibbonToolBarRow>
      <Fluent:RibbonToolBarControlDefinition Target="spinner" Width="45"/>
    </Fluent:RibbonToolBarRow>
    <Fluent:RibbonToolBarRow>
      <Fluent:RibbonToolBarControlDefinition Target="buttonGreen" />
      <Fluent:RibbonToolBarControlDefinition Target="buttonGray" />
    </Fluent:RibbonToolBarRow>
    <Fluent:RibbonToolBarRow>
      <Fluent:RibbonToolBarControlDefinition Target="buttonYellow" />
      <Fluent:RibbonToolBarControlDefinition Target="buttonBrown" />
    </Fluent:RibbonToolBarRow>
  </Fluent:RibbonToolBarLayoutDefinition>

</Fluent:RibbonToolBar.LayoutDefinitions>

<!--ToolBar Controls-->
<Fluent:Spinner x:Name="spinner" />
<Fluent:Button x:Name="buttonGreen" Header="Green" Icon="Images\Green.png” ... />
<Fluent:Button x:Name="buttonGray" Header="Gray" Icon="Images\Gray.png"  ... />
<Fluent:Button x:Name="buttonYellow" Header="Yellow" Icon="Images\Yellow.png" ../>
<Fluent:Button x:Name="buttonBrown" Header="Brown" Icon="Images\Brown.png" ... />
</Fluent:RibbonToolBar>
```

ColorGallery

Fluent Ribbon Control Suite provides capability to make color picker controls like color picker buttons or split buttons or even insert in context menu.

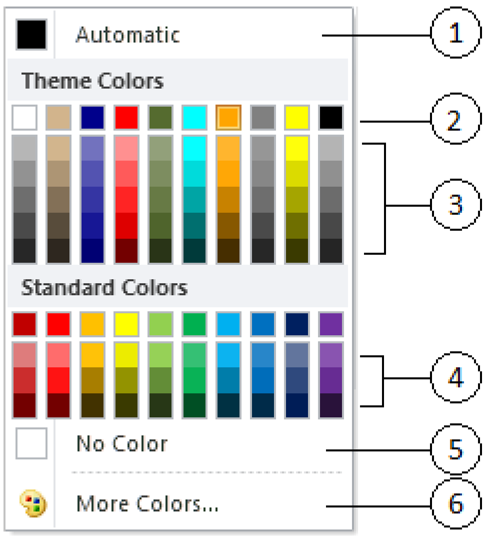


As shown above the ColorGallery control have three modes of appearance (to set mode use ColorGallery.Mode property):

- ColorGalleryMode.HighlightColors (color gallery displays only fixed highlight colors);
- ColorGalleryMode.StandardColors (color gallery displays only fixed standard colors);
- ColorGalleryMode.ThemeColors (color gallery displays theme colors and fixed standard colors).

ColorGallery can be placed anywhere in your application. For example, we can place it in DropDownButton as shown below:

```
<!-- The following code shows standard mode for color gallery -->
<Fluent:DropDownButton Name="colorPickerStandard"
 SizeDefinition="Middle" Header="Standard" >

  <!-- It's possible to create custom icon to present selected color -->
  <Fluent:DropDownButton.Icon>
    <Grid Width="16" Height="16">
      <Image Source="Images\FontColor.png"/>
      <Border BorderThickness="0" VerticalAlignment="Bottom" Height="4">
        <Border.Background>
          <SolidColorBrush Color="{Binding ElementName=colorGalleryStandard,
                                   Path=SelectedColor, FallbackValue=Black}" />
        </Border.Background>
      </Border>
    </Grid>
  </Fluent:DropDownButton.Icon>

  <Fluent:ColorGallery x:Name="colorGalleryStandard"
          SelectedColor="Red" IsNoColorButtonVisible="False" />

  <Fluent:MenuItem Icon="Images\Pink.png" Header="A Menu Item"/>
</Fluent:DropDownButton>
```



Some of important properties of ColorGallery are listed below:

Property	Description
SelectedColor	Use this property to set or get currently selected color. Default value is null
Mode	You can set or get one of the three modes of appearance
ChipWidth	You can set or get size of chips (boxes where color presented)
ChipHeight	You can set or get size of chips (boxes where color presented)
IsAutomaticColorButtonVisible	You can hide or show the button (1) (see the picture above)
IsNoColorButtonVisible	You can hide or show the button (2) (see the picture above)
IsMoreColorsButtonVisible	You can hide or show the button (6) (see the picture above). Additionally you can subscribe MoreColorsExecuting event to use you own ‘color choose’ dialog (otherwise the default dialog will be used).
IsRecentColorsVisible	You can hide or show the bar where colors picked with MoreColors appear
Columns	You can manage number of color gallery columns. It works only when Mode is ThemeColors
StandardColorGridRows	You can set or get how many rows will be in standard colors area (4) (see the picture above)
ThemeColorGridRows	You can set or get how many rows will be in theme colors area (3) (see the picture above)
ThemeColors ThemeColorsSource	Theme colors must be setted to show it in the gallery. Use one of these properties

IMPORTANT NOTES:

- SelectedColor property is Nullable (Color?).
- No Color value is Colors.Transparent.
- Automatic color value is null.