Synexis Rapid Start Add Code

In this walkthrough we'll be adding the code to control an Extron matrix switcher.

Adding the Click Event Handler

To add the click events, we'll start with MainPage.xaml and add the Click property. At the end of the **BtnInput1** tag, start typing Click. Visual Studio will speed your coding with auto complete. Select Click and tab to New EventHandler. The auto-complete process will add "BtnInput1_OnClick".

```
41          <Button
42              x:Name="BtnInput1"
43              Grid.Row="1"
44              Grid.Column="1"
45              Margin="4"
46              FontSize="36"
47              HorizontalAlignment="Stretch"
48              VerticalAlignment="Stretch"
49              Content="1"
50              Click="BtnInput1_OnClick"/>
```

If it's not already open, double-click MainPage.xaml.cs in the Solution Explorer panel. You'll find Visual Studio added an event handler to the code.

```
17
18  namespace QuickStartSwitcher
19  {
20      /// <summary>
21      /// An empty page that can be used on its own or navigated to within a Frame.
22      /// </summary>
23      public sealed partial class MainPage : Page
24      {
25          public MainPage()
26          {
27              this.InitializeComponent();
28          }
29
30          private void BtnInput1_OnClick(object sender, RoutedEventArgs e)
31          {
32              throw new NotImplementedException();
33          }
34      }
35  }
36
```

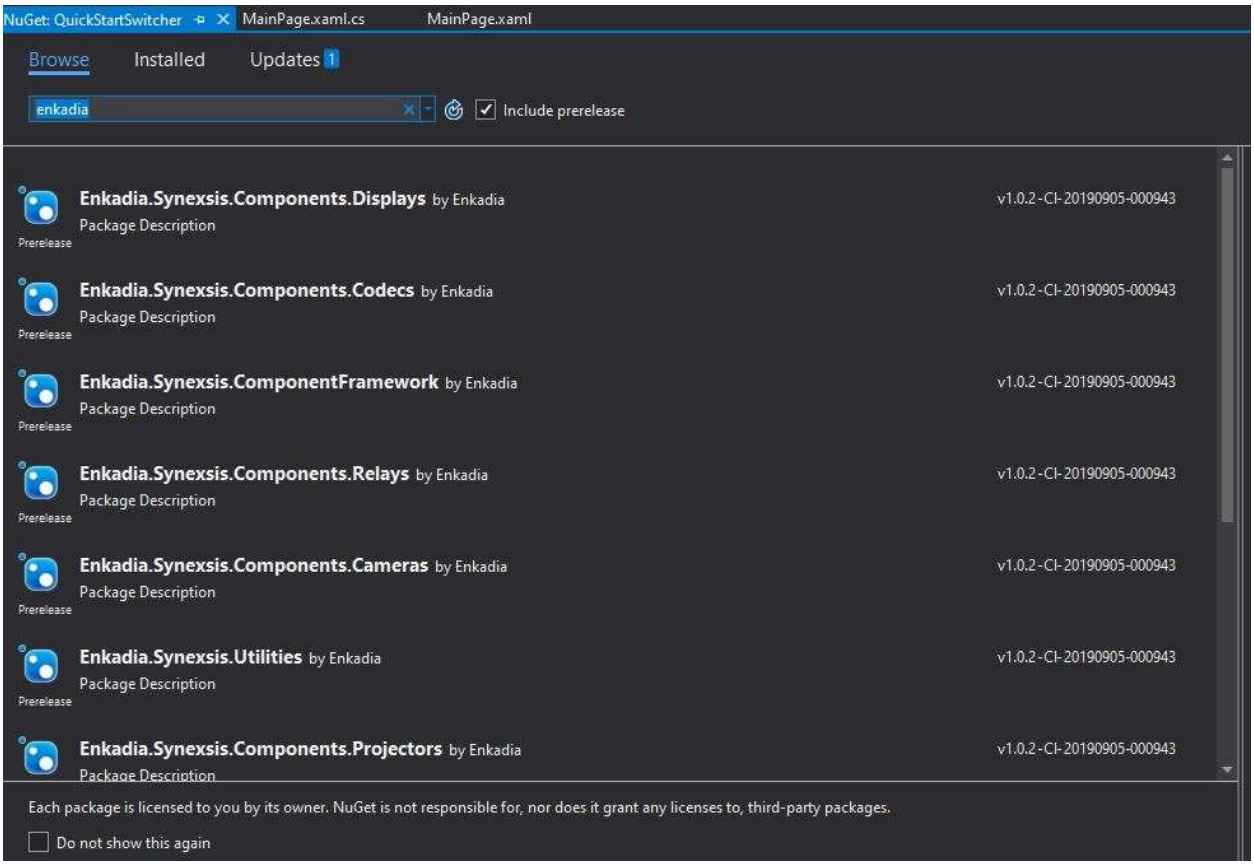
To complete, add click handlers to each of your buttons.

```
private void BtnInput1_OnClick(object sender, RoutedEventArgs e)
{
    throw new NotImplementedException();
}

private void BtnInput2_OnClick(object sender, RoutedEventArgs e)
{
    throw new NotImplementedException();
}

private void BtnInput3_OnClick(object sender, RoutedEventArgs e)
{
    throw new NotImplementedException();
}

private void BtnInput4_OnClick(object sender, RoutedEventArgs e)
{
    throw new NotImplementedException();
}

private void BtnOutput1_OnClick(object sender, RoutedEventArgs e)
{
    throw new NotImplementedException();
}

private void BtnOutput2_OnClick(object sender, RoutedEventArgs e)
{
    throw new NotImplementedException();
}
```

Add NuGet packages

The Enkadia Synexsis components are packaged and available in the NuGet repository. Either select Project | Manage NuGet Packages or right-click on References in the Solution Explorer to open the NuGet manager tab.

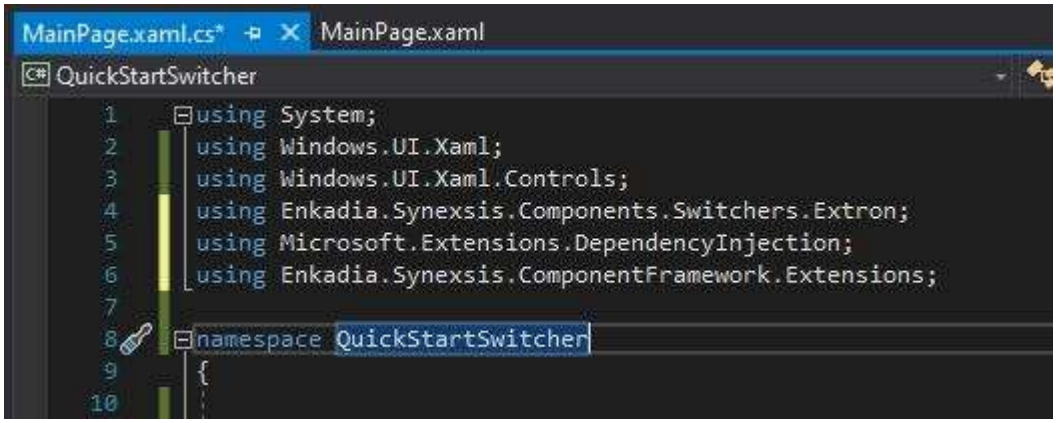
Select Browse and enter Enkadia in the search bar. *If you are a member of the Synexsis beta test program, check the prerelease box*. A list of Synexsis components will be displayed.



Select and install Enkadia.Synexsis.Components.Switchers. A popup will display the contents of the package for that component. Click Ok. A second popup indicates several of the packages, including Synexsis have licensing terms. Click I Accept the to complete the installation.

Using the Switcher component

To start, import the component with the C# using command. You'll need to add using statements for the switcher component, along with Microsoft.Extensions.DependencyInjection and Enkadia.Synexsis.ComponentFramework.Extensions.



For the initial declarations, add a variable for Input and a second for Output to hold the values of the channels to be changed. Add a method known as the ServiceProvider. For now it's good enough to think of the service provider as the connector between the application and the Synexsis components. To complete the initial steps, call out your switcher. When complete, your code block should look something like this:

```
public sealed partial class MainPage : Page
{
    int Input; //create variables for the input and output channels
    int Output;

    IServiceProvider serviceProvider; //Used to provide Synexsis methods to the application
    DXPSwitcher switcher; //Initialize an Extron switcher
```

Initialize the Switcher Component

To get everything setup correctly, add five more lines of code to the MainPage constructor to complete the switcher initialization. This setup code provides a placeholder for the switcher component. The remaining code completes the setup. Check the comments for notes on the role of each line.

```
public MainPage()
{
    this.InitializeComponent();

    //setup the Collection which will contain the Synexsis components
    serviceProvider = new ServiceCollection()
        .AddSynexsis() //add specific Synexsis functions.
        .AddTransient<DXPSwitcher>() //setup the switcher component
        .BuildServiceProvider();

    switcher = serviceProvider.GetService<DXPSwitcher>();
}
```

Add Code to the Buttons

Not much code is needed for this simple program. The channel buttons only require their channel number, while Take/Enter is the hard-working button in this application. In each of the click events for first two input buttons, assign the channel number to the Input variable.

```
private void BtnInput1_OnClick(object sender, RoutedEventArgs e)
{
    Input = 1;
}

0 references
private void BtnInput2_OnClick(object sender, RoutedEventArgs e)
{
    Input = 2;
}
```

Accelerate Your Code with Visual Studio and Synexsis

Because the button methods contain only one-line of code, use an “expression body” to speed up writing methods

```
//For the remaining functions use an expression body as a shortcut and save keystrokes
0 references
private void BtnInput3_OnClick(object sender, RoutedEventArgs e) => Input = 3;

0 references
private void BtnInput4_OnClick(object sender, RoutedEventArgs e) => Input = 4;

0 references
private void BtnOutput1_OnClick(object sender, RoutedEventArgs e) => Output = 1;

0 references
private void BtnOutput2_OnClick(object sender, RoutedEventArgs e) => Output = 2;

0 references
private void BtnOutput3_OnClick(object sender, RoutedEventArgs e) => Output = 3;

0 references
private void BtnOutput4_OnClick(object sender, RoutedEventArgs e) => Output = 4;
```

Wrap it up by adding code to the Take button.

```
private void BtnTake_OnClick(object sender, RoutedEventArgs e) => switcher.AVXPoint(Input, Output);
```

Configuration and licensing

Synexsis builds your components by reading values from an appsettings.json file, located at the root of your program's runtime directory. To run the application, add your specific values to the file.

```
{
        "DXPSwitcher": {
                "IPAddress": "192.168.150",
                "Port": 23,
                “User”:”admin”
                "Password": "default"
        },
        "License": {
                "OfflineActivation": "true",
                "LicenseFileName": "MyLicense.skm"
        }
}
```