

```

using System;
using System.Windows.Input;
using Xamarin.Forms;

namespace Example.App.CustomControls
{
    public class CustomStack : StackLayout
    {
        public static readonly BindableProperty LongpressCommandProperty =
            BindableProperty.Create(nameof(LongpressCommand), typeof(ICommand), typeof(CustomStack), null);

        public ICommand LongpressCommand
        {
            get { return (ICommand)GetValue(LongpressCommandProperty); }
            set { SetValue(LongpressCommandProperty, value); }
        }

        public static readonly BindableProperty CommandProperty =
            BindableProperty.Create(nameof(Command), typeof(ICommand), typeof(CustomStack), null);

        public ICommand Command
        {
            get { return (ICommand)GetValue(CommandProperty); }
            set { SetValue(CommandProperty, value); }
        }

        public static readonly BindableProperty CommandParameterProperty = BindableProperty.Create("CommandPa
            rameter", typeof(object), typeof(CustomStack), (object)null);

        public object CommandParameter
        {
            get { return GetValue(CommandParameterProperty); }
            set { SetValue(CommandParameterProperty, value); }
        }

        /// <summary>
        /// Long press event.
        /// If the Content or its children have gesture recognizers set, in order to prevent gesture conflicts, it is recomme
        nded to set their InputTransparent property to True.
        /// </summary>
        public event EventHandler LongPressed
        {
            add { LongPressedHandler += value; }
            remove { LongPressedHandler -= value; }
        }
        public EventHandler LongPressedHandler;

        /// <summary>
        /// Tap event.
        /// If the Content or its children have gesture recognizers set, in order to prevent gesture conflicts, it is recomme
        nded to set their InputTransparent property to True.
        /// </summary>

```

```

public event EventHandler Tapped
{
    add { TappedHandler += value; }
    remove { TappedHandler -= value; }
}
public EventHandler TappedHandler;

}
}

```

---

## ----- ANDROID RENDERER -----

---

```

using Android.Content;
using Example.App.CustomControls;
using Example.App.Droid.Renderers;
using Xamarin.Forms;
using Xamarin.Forms.Platform.Android;

```

```

[assembly: ExportRenderer(typeof(CustomStack), typeof(CustomStackRenderer))]
namespace Example.App.Droid.Renderers
{
    class CustomStackRenderer : VisualElementRenderer<StackLayout>
    {
        private CustomStack _view;

        public CustomStackRenderer(Context context) : base(context) { }

        protected override void OnElementChanged(ElementChangedEventArgs<StackLayout> e)
        {
            base.OnElementChanged(e);

            if (e.NewElement != null)
            {
                _view = e.NewElement as CustomStack;
            }

            LongClickable = true;
            Clickable = true;
            LongClick += (s, ea) =>
            {
                if (_view != null)
                {
                    _view.LongPressedHandler?.Invoke(_view, ea);
                    var command = _view.LongpressCommand; // CustomImage.GetCommand(_view);
                    command?.Execute(_view);
                }
            };

            Clickable = true;
            Click += (s, ea) =>
            {
                if (_view != null)

```

```

    {
        _view.TappedHandler?.Invoke(_view, ea);
        var command = _view.Command;// CustomImage.GetCommand(_view);
        command?.Execute(_view);
    }
};
}
}
}
}

```

----- iOS RENDERER -----  
 -----

```

using Foundation;
using Example.App.CustomControls;
using Example.App.iOS.CustomRenderers;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using UIKit;
using Xamarin.Forms;
using Xamarin.Forms.Platform.iOS;

```

```

[assembly: ExportRenderer(typeof(CustomStack), typeof(CustomStackRenderer))]
namespace Example.App.iOS.CustomRenderers

```

```

{
    class CustomStackRenderer : ViewRenderer
    {
        private CustomStack _view;
        private readonly UILongPressGestureRecognizer _longPressRecognizer;
        private readonly UITapGestureRecognizer _tapGestureRecognizer;

        public CustomStackRenderer()
        {
            _longPressRecognizer = new UILongPressGestureRecognizer((s) =>
            {
                if (s.State == UIGestureRecognizerState.Began && _view != null)
                {
                    _view.LongPressedHandler?.Invoke(_view, null);
                    var command = _view.LongpressCommand;// CustomImage.GetCommand(_view);
                    command?.Execute(_view);
                }
            });

            _tapGestureRecognizer = new UITapGestureRecognizer(() =>
            {
                _view.TappedHandler?.Invoke(_view, null);
                var command = _view.Command;// CustomImage.GetCommand(_view);
                command?.Execute(_view);
            });
        }
    }
}

```

```
protected override void OnElementChanged(ElementChangedEventArgs<View> e)
{
    base.OnElementChanged(e);

    if (e.NewElement != null)
    {
        _view = e.NewElement as CustomStack;
    }

    if (Control != null)
    {
        Control.UserInteractionEnabled = true;
        Control.AddGestureRecognizer(_longPressRecognizer);
        Control.AddGestureRecognizer(_tapGestureRecognizer);
    }
}
}
```