Configuring a Custom Panel Based on a Batch Class

(Validation Custom Panel)

Application Note

Date	August 24, 2011
Applies To	Kofax Capture 8.0, 9.0, 10.0
Summary	This application note provides a methodology for programmatically configuring a Custom Panel at runtime.
Revision	1.1

Overview

There are times when a Custom Panel might need to be changed automatically at runtime based on a particular Batch Class. This application note offers a very simple approach to providing this functionality. Although there are solutions that implement a more object-oriented methodology, such as abstract factory methods and the like, this article addresses a very simple concept. It will be up to the developer to implement this concept using a different paradigm.

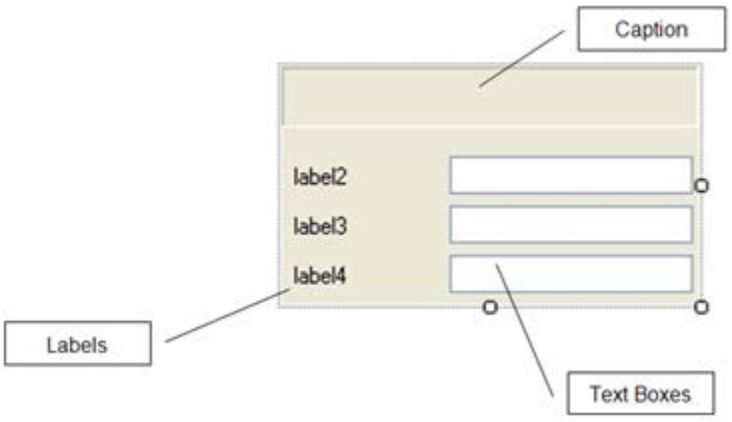
The Problem

A customer is implementing multiple Batch Classes in an environment. Each Batch Class must display its unique configuration in the Validation Module via a Custom Panel. In other words, the Custom Panel must change automatically depending on the current Batch Class.

The Solution

The Kofax Capture Module Type Library API provides a property for obtaining the Batch Class name of the current Batch. The *Application.ActiveBatch.ClassName* can be used to identify the current Batch Class then perform the necessary changes to the Custom Panel.

To illustrate this concept, we have created a User Control and added some labels and text boxes as shown below.



In the code, the procedure follows this process:

1. Trap the *KfxOcxEventBatchOpened* event in the Action method.
2. Check the *Application.ActiveBatch.ClassName* string value.
3. Configure some properties based on the ClassName.

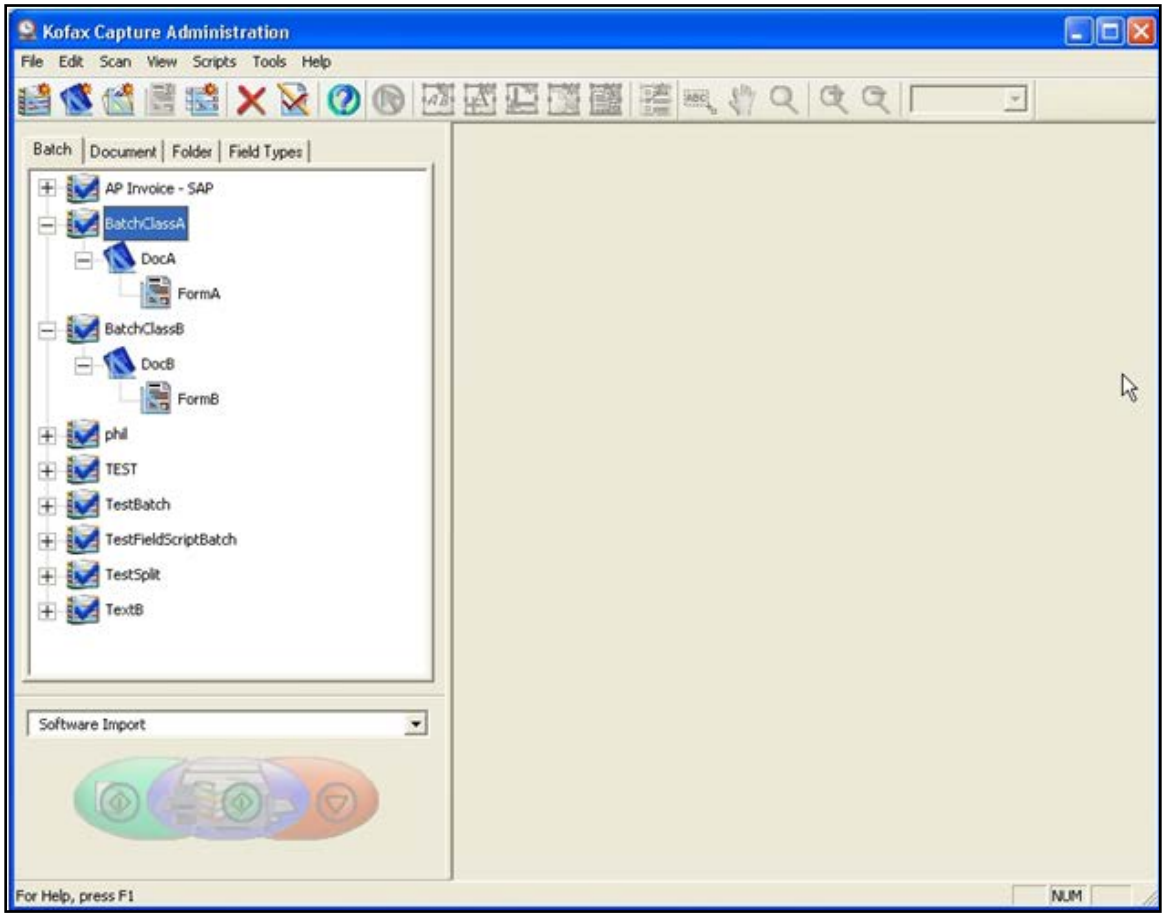
The complete C# code listing is displayed on the following three pages.

```
using System;
using System.Text;
using System.Windows.Forms;
using System.Drawing;
using Kofax.AscentCaptureModule;
using System.Runtime.InteropServices;

namespace MultiPanel
{
    [Guid("1D59962B-54BB-4e59-908D-7CFCE08DD0D6")]
    [InterfaceType(ComInterfaceType.InterfaceIsIDispatch)]
    public interface _Multi
    {
        [DispId(1)]
        Kofax.AscentCaptureModule.Application Application { set; get; }
        [DispId(2)]
        void ActionEvent(int EventNumber, object Argument, out int Cancel);
    }

    [Guid("AD324023-5A19-42f0-A9C9-BFC449587F9C")]
    [ClassInterface(ClassInterfaceType.None)]
    [ProgId("MultiPanel.Multi")]
    public class Multi : UserControl, _Multi
    {
        public TextBox textBox1;
        public TextBox textBox2;
        public TextBox textBox3;
        public Label index1;
        public Label index2;
        public Label index3;
        public Label label1;
        private Kofax.AscentCaptureModule.Application oApp;

        public struct bParams
        {
            public string sIndex1;
            public string sIndex2;
            public string sIndex3;
            public bool bIndex1;
            public bool bIndex2;
            public bool bIndex3;
        }
        bParams pValues;

        private void InitializeComponent()
        {
            this.label1 = new System.Windows.Forms.Label();
            this.textBox1 = new System.Windows.Forms.TextBox();
            this.textBox2 = new System.Windows.Forms.TextBox();
            this.textBox3 = new System.Windows.Forms.TextBox();
            this.index1 = new System.Windows.Forms.Label();
            this.index2 = new System.Windows.Forms.Label();
            this.index3 = new System.Windows.Forms.Label();
            this.SuspendLayout();
            //
            // label1
            //
            this.label1.BorderStyle = System.Windows.Forms.BorderStyle.Fixed3D;
            this.label1.Dock = System.Windows.Forms.DockStyle.Top;
            this.label1.Location = new System.Drawing.Point(0, 0);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(221, 32);
            this.label1.TabIndex = 0;
            this.label1.TextAlign = System.Drawing.ContentAlignment.MiddleCenter;
            //
            // textBox1
            //
            this.textBox1.Location = new System.Drawing.Point(89, 47);
            this.textBox1.Name = "textBox1";
            this.textBox1.Size = new System.Drawing.Size(129, 20);
            this.textBox1.TabIndex = 1;
            this.textBox1.Visible = false;
```

```
            //
            // textBox2
            //
            this.textBox2.Location = new System.Drawing.Point(89, 73);
            this.textBox2.Name = "textBox2";
            this.textBox2.Size = new System.Drawing.Size(129, 20);
            this.textBox2.TabIndex = 2;
            this.textBox2.Visible = false;
            //
            // textBox3
            //
            this.textBox3.Location = new System.Drawing.Point(89, 99);
            this.textBox3.Name = "textBox3";
            this.textBox3.Size = new System.Drawing.Size(129, 20);
            this.textBox3.TabIndex = 3;
            this.textBox3.Visible = false;
            //
            // index1
            //
            this.index1.Location = new System.Drawing.Point(3, 51);
            this.index1.Name = "index1";
            this.index1.Size = new System.Drawing.Size(83, 16);
            this.index1.TabIndex = 4;
            this.index1.Text = "label2";
            this.index1.Visible = false;
            //
            // index2
            //
            this.index2.Location = new System.Drawing.Point(3, 77);
            this.index2.Name = "index2";
            this.index2.Size = new System.Drawing.Size(83, 16);
            this.index2.TabIndex = 5;
            this.index2.Text = "label3";
            this.index2.Visible = false;
            //
            // index3
            //
            this.index3.Location = new System.Drawing.Point(3, 103);
            this.index3.Name = "index3";
            this.index3.Size = new System.Drawing.Size(83, 16);
            this.index3.TabIndex = 6;
            this.index3.Text = "label4";
            this.index3.Visible = false;
            //
            // Multi
            //
            this.Controls.Add(this.index3);
            this.Controls.Add(this.index2);
            this.Controls.Add(this.index1);
            this.Controls.Add(this.textBox3);
            this.Controls.Add(this.textBox2);
            this.Controls.Add(this.textBox1);
            this.Controls.Add(this.label1);
            this.Name = "Multi";
            this.Size = new System.Drawing.Size(221, 125);
            this.Load += new System.EventHandler(this.Multi_Load);
            this.ResumeLayout(false);
            this.PerformLayout();
        }

        public Multi()
        {
            InitializeComponent();
        }

        private void Multi_Load(object sender, EventArgs e)
        {
            SetDefaults();
        }

        public Kofax.AscentCaptureModule.Application Application
        {
            set {oApp = value;}
            get {return oApp;}
        }
```

```
        public void ActionEvent(int EventNumber, object Argument, out int Cancel)
        {
            // Configure the panel to display the components
            // specified by the current Batch Class.

            switch ((KfxOcxEvent)EventNumber)
            {
                case KfxOcxEvent.KfxOcxEventBatchOpened:
                    switch (oApp.ActiveBatch.ClassName)
                    {
                        case "BatchClassA":
                            pValues.bIndex1 = true;
                            pValues.bIndex2 = true;
                            pValues.bIndex3 = false;
                            pValues.sIndex1 = "Index A";
                            pValues.sIndex2 = "Index B";
                            break;
                        case "BatchClassB":
                            pValues.bIndex1 = true;
                            pValues.bIndex2 = true;
                            pValues.bIndex3 = true;
                            pValues.sIndex1 = "Index A";
                            pValues.sIndex2 = "Index B";
                            pValues.sIndex3 = "Index C";
                             break;
                        default:
                             pValues.bIndex1 = false;
                             pValues.bIndex2 = false;
                             pValues.bIndex3 = false;
                            break;
                    }
                    ConfigureDisplay();
                    break;

                case KfxOcxEvent.KfxOcxEventBatchClosing:
                    SetDefaults();
                    break;
            }
            Cancel = 0;
        }

        private void SetDefaults()
        {
            label1.Text = "";

            index1.Visible = false;
            textBox1.Visible = false;

            index2.Visible = false;
            textBox2.Visible = false;

            index3.Visible = false;
            textBox3.Visible = false;
        }

        private void ConfigureDisplay()
        {
            label1.Text = oApp.ActiveBatch.ClassName;

            index1.Visible = pValues.bIndex1;
            textBox1.Visible = index1.Visible;
            index1.Text = pValues.sIndex1;

            index2.Visible = pValues.bIndex2;
            textBox2.Visible = index2.Visible;
            index2.Text = pValues.sIndex2;

            index3.Visible = pValues.bIndex3;
            textBox3.Visible = index3.Visible;
            index3.Text = pValues.sIndex3;
        }
    }
}
```

For this example, we created two Batch Classes labeled, “BatchClassA” and “BatchClassB” as shown below.



The Custom Panel configuration will follow this model.

There are two defaults, one representing an untrapped Batch Class name (described below) and the other, a reset state which is set as the Batch closes.

Default:

Caption = Batch Class Name (The reset state sets this string to empty.)
All Labels & Text Boxes Visible = false

BatchClassA:

Caption = Batch Class Name
Label 1 & 2 and Text Box 1 & 2 are visible
Labe1 Text = “Index A”
Label 2 Text = “Index B”

BatchClassB:

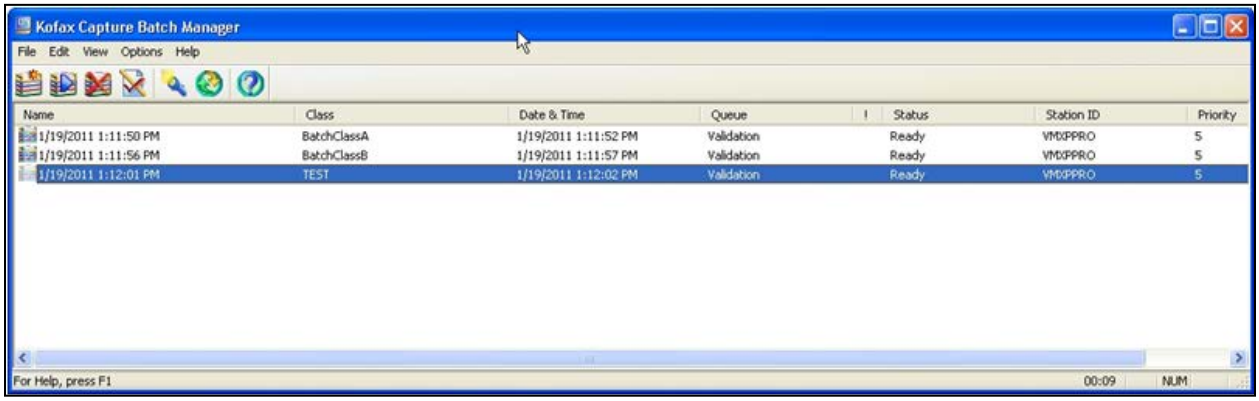
Caption = Batch Class Name
Label 1, 2 & 3 and Text Box 1, 2 & 3 are visible

Labe1 Text = “Index A”

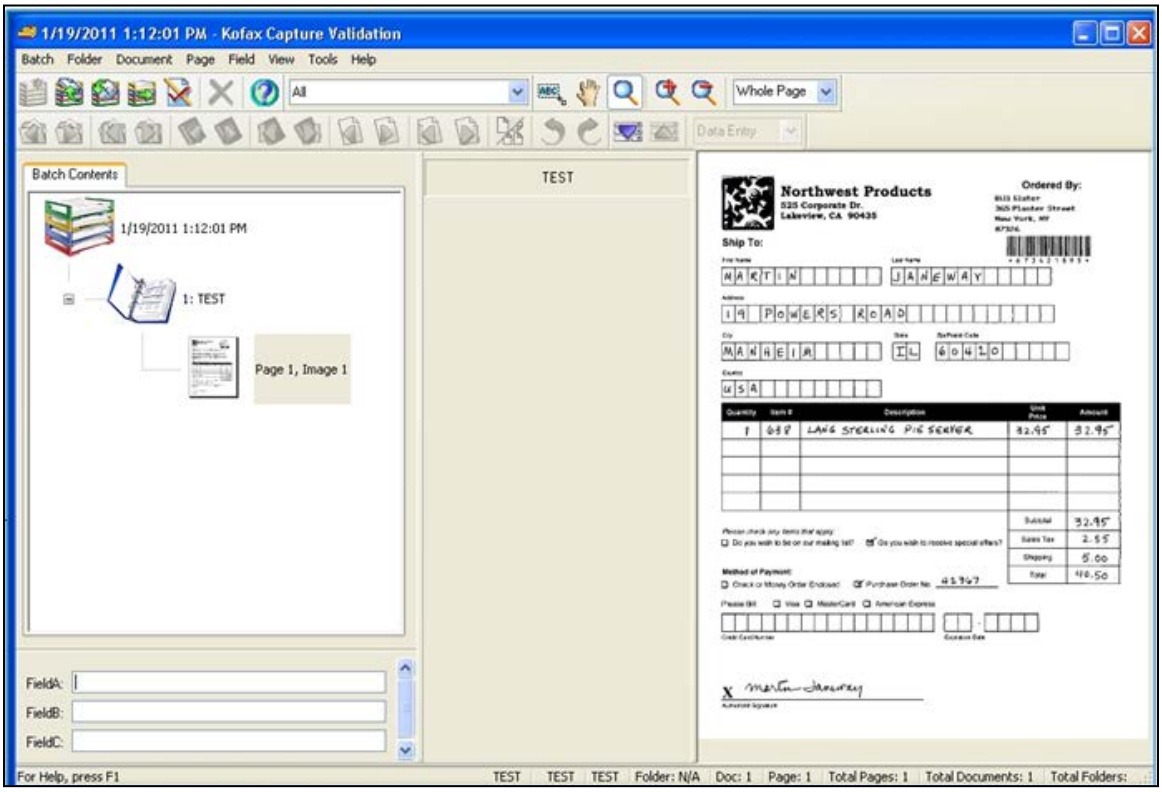
Label 2 Text = “Index B”

Label 3 Text = “Index C”

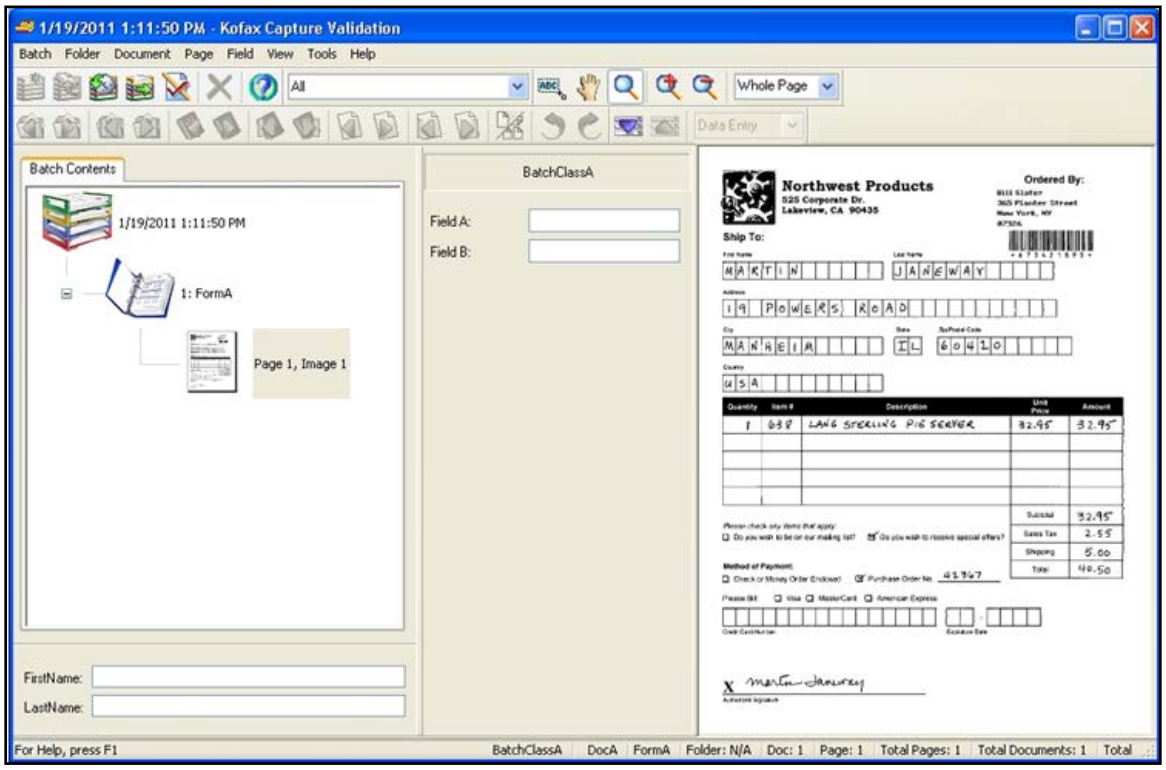
We then created three Batches, one of which will not be trapped, and cause the Custom Panel to be set to the default configuration.



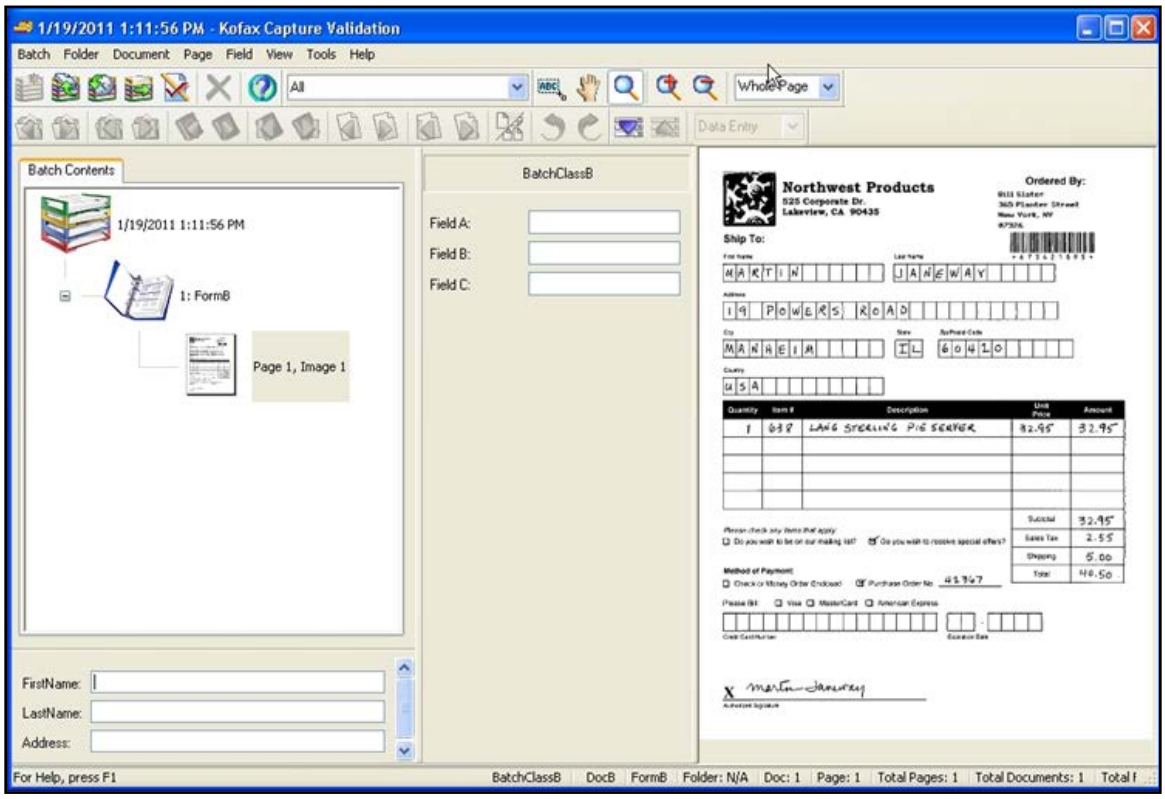
The default state when an untrapped Batch is detected.



The BatchClassA state.



The BatchClassB state.



Summary

This is a very simplified solution intended to show how one might implement a Custom Panel that automatically changes based on the current Batch Class name. Other ideas might include adding multiple Panel controls with associated text Boxes, Labels, etc. and enabling/disabling those Panels, as required. Again, a more object-oriented approach could also be implemented using an abstract factory method.