# Controls Extended User Manual

This is a code base DLL that is to be uploaded into IFB (Input Form Builder) forms. The DLL exposes functions that will allow you to create Display/View Only controls from HTML that can be inserted into a panel control on your form or allow you to do styling on EllieMae controls that are not available through the properties page of the control. Most of the available functions are for creating simple tables.

The DLL has been updated to read in Global Custom Data Objects (CDO) that contains a HTML control description. These files can be read in and inserted into a Panel control on the form.

All controls created using this code base can only be inserted into a Panel control, this control is referred to as the parent. If the parent is not a Panel control the DLL will throw an error.

## IFB Form Functions:

**Sub SetControlStyle(control as Control, styleName as string, value as object)**

This function will allow you to set the styling of an IFB form control.

* Parameter Values:
  + control as Control: this can be an EllieMae IFB Form control. IE: TextBox, Label, DropdownBox, etc…
  + styleName as string: this is the name of the style property you want to change or assign a value to. IE: text-decoration-line-through, text-align, etc…
    - The available styles that can be applied to the EllieMae controls are:
      * background-attachment
      * background-image
      * background-position
      * background-position-x
      * background-position-y
      * background-repeat
      * border-bottom
      * border-bottom-color
      * border-bottom-style
      * border-bottom-width
      * border-left
      * border-left-color
      * border-left-style
      * border-left-width
      * border-right
      * border-right-color
      * border-right-style
      * border-right-width
      * border-top
      * border-top-color
      * border-top-style
      * border-top-width
      * clear
      * clip
      * css-text
      * cursor
      * display
      * filter
      * letter-spacing
      * line-height
      * list-style
      * list-style-image
      * list-style-position
      * list-style-type
      * margin
      * margin-bottom
      * margin-left
      * margin-right
      * margin-top
      * overflow
      * padding
      * padding-bottom
      * padding-left
      * padding-right
      * padding-top
      * page-break-after
      * page-break-before
      * pixel-height
      * pixel-left
      * pixel-top
      * pixel-width
      * pos-height
      * pos-left
      * pos-top
      * pos-width
      * style-float
      * text-align
      * text-decoration
      * text-decoration-line-through
      * text-decoration-none
      * text-decoration-overline
      * text-decoration-underline
      * text-indent
      * text-transform
      * vertical-align
      * white-space
      * word-spacing
      * z-index
    - If a style name is not found the DLL will throw an error message:
      * Style '<style name>' was not found and could not be assigned.
    - Not all style can be applied to all controls. I haven’t done an in depth review of what style can be applied to what controls, so it will require some experimentation.
  + value as Object: this is any valid value for the style you are trying to set on the control. To get a complete list of the valid values for the style open google in a browser window and enter “html style <style name>”. One of your first results should be from W3Schools.com.
* Examples:

SetControlStyle(Label1, "text-decoration-line-through", true)

Result: 

SetControlStyle(Label2, "text-align", "center")

Result: 

**Sub CreateControlFromHtml(parent as Panel, html as string)**

This function will allow you to create am Html control for display/view only.

* Parameter Values:
  + Parent as Panel: this must be an EllieMae panel control that has been added to the IFB form.
  + html as string: this is any valid html formatted as a string.
* Example:

dim html as string

html = "<ul style='Max-Height: 50px; overflow-y: scroll;'>"

html &= "<li>First List Item</li>"

html &= "<li>Second List Item</li>"

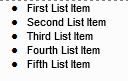
html &= "<li>Third List Item</li>"

html &= "<li>Fourth List Item</li>"

html &= "<li>Fifth List Item</li>"

html &= "</ul>"

CreateControlFromHtml(Panel2, html)

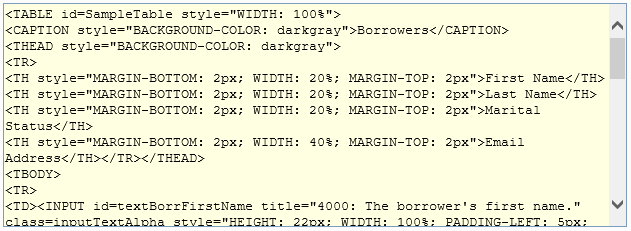
Result: 

**Function GetParentHtml(parent as Panel) as string**

This function is used to assist in trouble shooting Html issues.

* Parameter Values:
  + Parent as Panel: this can be any panel on the IFB form not just the panel you have inserted html into.
* Return Value:
  + This function returns the inner Html of the parent panel as a string. This function is used to assist in trouble shooting Html issues.
* Example:

textHtml.Text = GetParentHtml(Panel1)

Result: 

**Table Factory**

The table factory is what is used to create simple display/view only tables.

**Sub Create(parent as Panel)**

This sub sets everything up to begin creating an Html table. This sub must be called first to begin building the Html table.

* Parameter Values:
  + parent as Panel: This is an IFB form panel.
* Example:

TableFactory.Create(Panel1)

**Sub AddStyle(parent as Panel, styleType as TableStyleTypes, styleName as string, value as string)**

This sub allows you to assign styling to the different parts of the Html table. All styles must be assigned before adding components to the table.

* Parameter Values:
  + parent as Panel: This is an IFB form panel control that will house the table. This should be the same panel control used in the Create sub.
  + styleType as TableStyleTypes: This is an enumeration of the different Html table parts that can have styles assigned to them.
    - TableStyleTypes values are:
      * Caption: The title of the table
      * Table: the html table as a whole
      * TH: the header columns for the table
      * TD: the table cell
  + styleName as string: the name of the style you want to add to the table component.
  + value as string: this is any valid value for the style you are trying to set on the control. To get a complete list of the valid values for the style open google in a browser window and enter “html style <style name>”. One of your first results should be from W3Schools.com.
* Example:

TableFactory.AddStyle(Panel1, TableFactory.TableStyleTypes.TD, "background-color", "LightBlue")

**Sub AddCaption(Panel1, caption as string)**

This sub will create the caption of the Html table.

* Parameter Values:
  + parent as Panel: This is an IFB form panel control that will house the table. This should be the same panel control used in the Create sub.
  + caption as string: Any string value that will be used as the title for the Html table.
* Example:

TableFactory.AddCaption(Panel1, Macro.GetField("11"))

**Sub AddCoumns(parent as Panel, columnHeaders() As String)**

This sub adds the columns and their headers to the Html table. The number of columns created is determined by the size of the columnHeader array.

* Parameter Values:
  + parent as Panel: This is an IFB form panel control that will house the table. This should be the same panel control used in the Create sub.
  + columnHeaders() As String: A string array containing the header for each of the columns to be added to the Html table.
* Example:

dim columHeaders() as string = { "Column1", "Column2", "Column3" }

TableFactory.AddColumns(Panel1, columHeaders)

**Sub AddRow(parent as Panel, row() as object)**

This sub adds each row to the Html table. The number of columns added are determined by the size of the row() array. This array does not have to be the same size as the columHeaders() array, it can contain fewer columns but should not contain more.

* Parameter Values:
  + parent as Panel: This is an IFB form panel control that will house the table. This should be the same panel control used in the Create sub.
  + row() as object: an object array describing the values for each column in the row.
* Example:

dim row1() as object = { Macro.GetField("305"), EncompassApplication.CurrentLoan.Fields("1109").FormattedValue, EncompassApplication.CurrentLoan.Fields("745").FormattedValue }

dim row2() as object = { 5, true}

dim row3() as object = { "Borrower 1", "555-44-3333", "01/01/2001" }

TableFactory.AddRow(Panel1, row1)

TableFactory.AddRow(Panel1, row2)

TableFactory.AddRow(Panel1, row3)

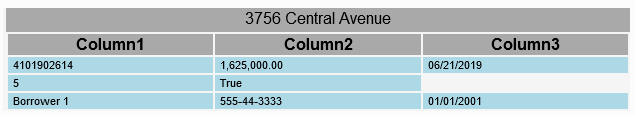
**Sub Show(parent as Panel)**

Once the Html table as finished being constructed, this sub will load it into the parent and display it on the IFB form.

* Parameter Value:
  + parent as Panel: This is an IFB form panel control that will house the table. This should be the same panel control used in the Create sub.
* Example:

TableFactory.Show(Panel1)

The above TableFactory methods will produce this Html table:



## Html Control Loader

The Html Control Loader will automatically load Html controls that are stored in the CDO into a Panel on an IFB form. Because these controls are stored in the CDO the same Html control can be loaded onto multiple IFB forms that have the Controls Extended DLL code base.

This is accomplished by 2 files being loaded into the CDO. The first file is a JSON configuration file that lists the parent control on the IFB form the Html control is to be loaded into and the name of the CDO file that contains the Html controls definition.

**JSON Configuration File:**

The configuration file contains a list of the Html controls to be added to the form and their parent control. When creating this file it must be given the same name as the IFB form that will be loading it and have .json as the extension. Be sure to math the spellingand casing of the forms name. For example I have a test form I use to test the functionality of this code base on, it is called TestSetControlStyles. The configuration file loaded into the CDO must be named TestSetControlStyles.json. Otherwise the code base will not find the file and assume there are no Html controls to be loaded.

**Sample JSON Configuration file contents:**

[

{

"Parent": "Panel1",

"ControlFile": "SampleTable.html"

}

]

"Parent": In quotation marks add the name of the parent control the Html is to be inserted into. Be sure to match the spelling and casing of the controls ControlID. If the ControlID is panelHtmlTable then it must be entered here as “panelHtmlTable”.

"ControlFile": In quotation marks add the name of the CDO file that contains the Html controls definition. Be sure to match the spelling and casing of the file name with its extension. I use “.html” to help me identify the Html control files, you can use whatever extension you want.

Multiple controls can be added by listing the json object multiple times and comma separated.

Example of adding multiple HTML controls:

[

{

"Parent": "Panel1",

"ControlFile": "SampleTable.html"

},

{

"Parent": "Panel2",

"ControlFile": "SampleTable2.html"

}

]

This will load the SampleTable.html control in to Panel1 on the IFB form and SampleTable2.html into Panel2 on the IFB form.

**Html Control File:**

The Html Control File describes the Html control that is to be loaded into the parent control. You can use any editor you want to create these files. Notpad or NotePad++ works well if you don’t have Visual Studio or an HTML editor. I built this to create and load tables, but I don’t see why it can’t be used to create other types of controls.

These control files can incorporate a handful of IFB controls. Right now the controls that can be added are the TextBox, DropdownBox, CheckBox, and Label. These controls can be data-bound to Encompass fields, even the Label. The Label control isn’t really data-bound but it is wired so that if a field id is set to it, it will update when the field value updates.

**Note**: Events and custom coding cannot be added to these controls, they can only be data-bound. (For now)

I have added tags that the Controls Extended code base can parse to create the IFB controls. There tags are:

* <TextBox> - creates a TextBox control.
* <DropdownBox> - creates a Dropdown control with predefined options. Can not dynamically add options to this control.
* <CheckBox> - Creates a ChekcBox control.
* <Label> - creates a label control.

These are the attributes for the control tags, although you can add all of these attributes to all of the controls, not every attribute will affect the control it is assigned to.

* name: a unique name that is assigned to the control when it is created.
* field-source: the source of the encompass field that will be bound to the control.
  + Valid values are:
    - CurrentLoan
    - LinkedLoan
* field-id: the encompass field that is to be data bound to the control.
* field-text: the text that is to be displayed in the control. This attribute only applies to the Label and CheckBox controls.
  + If a field-id is assigned to the Label control then the field-text attribute acts as a default text when the Encompass field has no value.
* enabled: sets the enabled property of the control. This attribute does not apply to the Label control. The appearance of the control does not change to the disabled look when this attribute is set to false until the user tries to click into it.
  + Valid values are:
    - true
    - false
* style: used for styling the controls.

**Control Examples:**

**TextBox:**

<TextBox name="textBorrFirstName"

field-source="CurrentLoan"

field-id="4000"

style="height: 22px; width:100%; padding-left:5px; padding-right:5px" />

**DropdownBox:**

<DropdownBox name="ddbBorrStatus"

field-id="52"

style="height: 22px; width:100%; padding-left:5px; padding-right:5px">

<option value=""></option>

<option value="Married">Married</option>

<option value="Unmarried">Unmarried</option>

<option value="Seperated">Seperated</option>

</DropdownBox>

**CheckBox:**

<CheckBox name="checkBorrBackground"

field-source="CurrentLoan"

field-id="3862"

style="height: 22px; width: 22px; padding-left:5px; padding-right:5px" />

**CheckBox with caption:**

<CheckBox name="chkBorrMortgSvc"

field-source="CurrentLoan"

field-id="3860"

field-text="Sample"

style="height: 22px; width: 22px; padding-left:5px; padding-right:5px" />

**Label:**

<Label name="labelCobEmail"

field-source="CurrentLoan"

field-id="1268"

field-text="Co-Borrower Email Missing"

style="height: 22px; width:100%; padding-left:5px; padding-right:5px" />

**Sample Html Control:**

<table id="SampleTable" style="width: 100%;">

<caption style="background-color: darkgray;">Borrowers</caption>

<thead style="background-color: darkgray;">

<tr>

<th style="margin-top: 2px; margin-bottom: 2px; width: 20%;">First Name</th>

<th style="margin-top: 2px; margin-bottom: 2px; width: 20%;">Last Name</th>

<th style="margin-top: 2px; margin-bottom: 2px; width: 20%">Marital Status</th>

<th style="margin-top: 2px; margin-bottom: 2px; width: 40%">Email Address</th>

</tr>

</thead>

<tbody>

<tr>

<td>

<TextBox name="textBorrFirstName"

field-source="CurrentLoan"

field-id="4000"

style="height: 22px; width:100%; padding-left:5px; padding-right:5px" />

</td>

<td>

<TextBox name="textBorrLastName"

field-source="CurrentLoan"

field-id="4002"

style="height: 22px; width:100%; padding-left:5px; padding-right:5px" />

</td>

<td style="align-content: center">

<DropdownBox name="ddbBorrStatus"

field-id="52"

style="height: 22px; width:100%; padding-left:5px; padding-right:5px">

<option value=""></option>

<option value="Married">Married</option>

<option value="Unmarried">Unmarried</option>

<option value="Seperated">Seperated</option>

</DropdownBox>

</td>

<td>

<Label name="labelBorrEmail"

field-source="CurrentLoan"

field-id="1240"

field-text="Borrower Email Missing"

style="height: 22px; width:100%; padding-left:5px; padding-right:5px" />

</td>

</tr>

<tr style="background-color: lightgrey;">

<td>

<TextBox name="textCobFirstName"

field-source="CurrentLoan"

field-id="4004"

style="height: 22px; width:100%; padding-left:5px; padding-right:5px" />

</td>

<td>

<TextBox name="textCobLastName"

field-source="CurrentLoan"

field-id="4006"

style="height: 22px; width:100%; padding-left:5px; padding-right:5px" />

</td>

<td style="align-content: center">

<DropdownBox name="ddbCobStatus"

field-id="84"

enabled="false"

style="height: 22px; width:100%; padding-left:5px; padding-right:5px">

<option value=""></option>

<option value="Married">Married</option>

<option value="Unmarried">Unmarried</option>

<option value="Seperated">Seperated</option>

</DropdownBox>

</td>

<td>

<Label name="labelCobEmail"

field-source="CurrentLoan"

field-id="1268"

field-text="Co-Borrower Email Missing"

style="height: 22px; width:100%; padding-left:5px; padding-right:5px" />

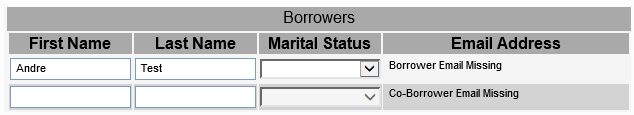
</td>

</tr>

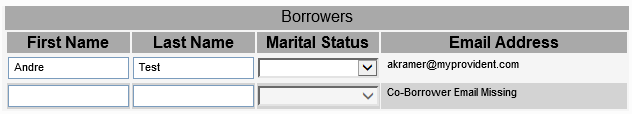
</tbody>

</table>

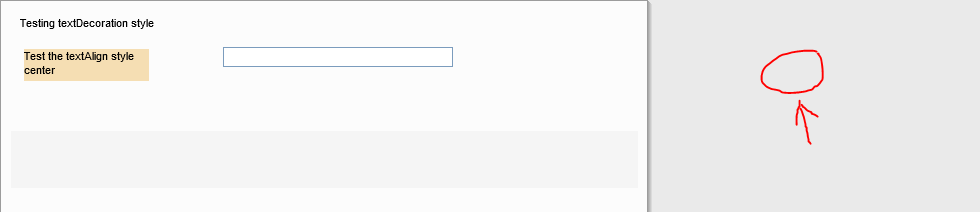
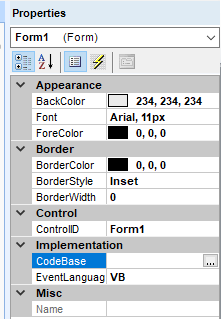
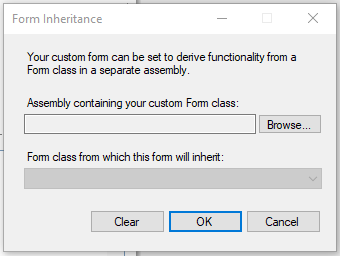
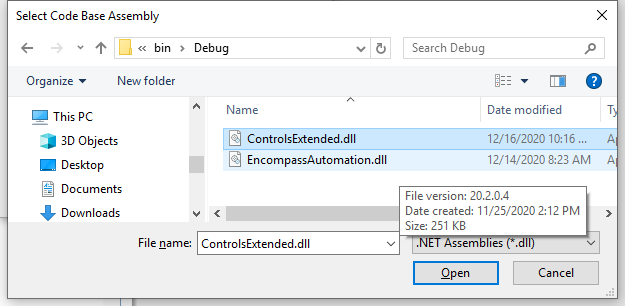
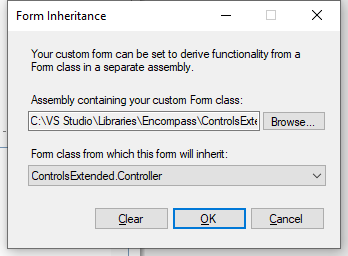
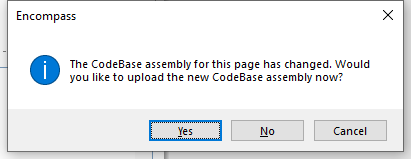
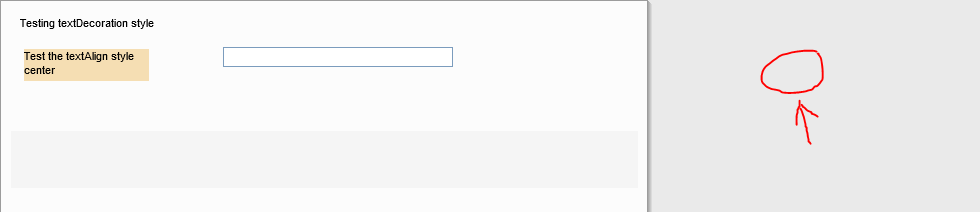
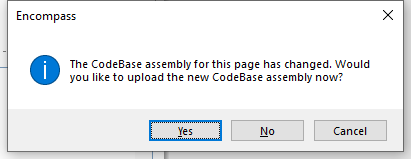
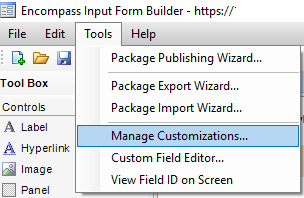
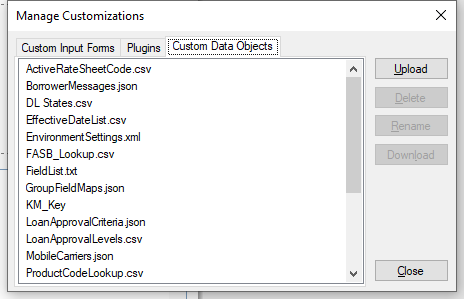
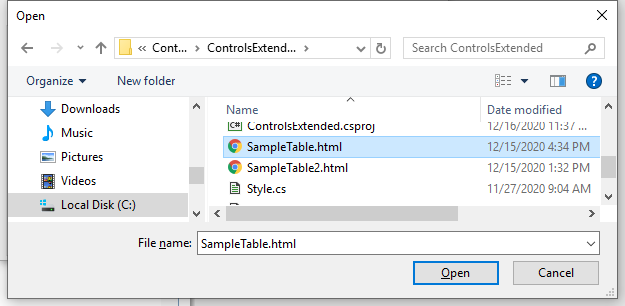
The above Sample Html control will produce this table:



After adding the borrowers email address on the same form:



### FAQ

* How do I load a code base DLL into an IFB form?
  + In the IFB with your form open, click into the gray empty area of the form designer. This will bring up the form properties window.
  + 
  + In the forms properties window click into the CodeBase property:
  + 
  + Click on the ellipsis button to open the Form Inheritance form.
  + 
  + Click on the browse button and navigate out to where the CodeBase DLL is. Select the DLL and press the Open button.
  + 
  + In the “Form class from which this form will inherit:” drop down select the class that will be the code base for the form.
  + 
  + Press the Ok button.
  + When you save the form you will get the following popup message:
  + 
  + Press the Yes button.
* How do I re-load a code base DLL after it has been updated?
  + In the IFB with your form open, click into the gray empty area of the form designer. This will bring up the form properties window.
  + 
  + When you save the form you will get the following popup message:
  + 
  + Press the Yes button.
* How do I upload a file to the CDO?
  + In the IFB click on the Tools menu and then the Manage Customizations..
  + 
  + In the manage customization form select the Custom Data Objects tab.
  + 
  + Click the Upload button and navigate to the file you want to add to the CDO
  + 
  + Select the file and press the open button. Press ok on the popup after the file is loaded.