Encompass Input Form Builder User's Guide

For use with Encompass Banker Edition

Encompass Input Form Builder User's Guide

Rev. 10/23/13

Table of Contents

Chapter 1

Introduction

Welcome to the Encompass Input Form Builder User's Guide. This guide contains information you need to understand the Input Form Builder environment and quickly begin to create input forms to meet your company's requirements.

Intended Audience

This guide is intended for users responsible for customizing Encompass. A basic knowledge of form design and familiarity with an HTML editing application such as Microsoft Front Page, or a visual development application such as Microsoft Visual Basic is recommended.

Chapter 10, "Creating Custom Event Handlers" explains how to design forms featuring controls that trigger events and similar types of actions. To complete these topics, you should be an experienced Microsoft Visual Basic or .NET developer.

NOTE: *Your access to the Input Form Builder depends on the user settings defined by your Encompass system administrator.*

What is the Input Form Builder?

The Input Form Builder allows you to design and create new input forms, or copy and modify standard Encompass forms.

With the Form Builder you can:

- Create forms to meet your company's specific requirements.
- Build forms using more than 25 different controls, including text boxes, dropdown lists, images, and field locks.
- Add custom forms to your Encompass forms list.
- Publish plugins and custom data objects for use with Encompass to your company's Encompass server.

How This Guide is Organized

- This chapter provides an overview of the Form Builder and discusses the resources available if you need assistance using it.
- Chapter 2, "Creating a Form" on page 4 describes how to create a basic form and save it to your Encompass system. It also discusses the form controls.
- Chapter 3, "Field Controls" on page 6 describes the Field controls which include text boxes, check boxes, and radio buttons.
- Chapter 4, "Field Controls" on page 9 describes the Action controls which include buttons and the Rolodex.
- Chapter 5, "Container Controls" on page 12 describes the Container controls which are used to form logical groups of controls.
- Chapter 6, "User Interface Controls" on page 14 describes the User Interface controls which include labels and images.
- Chapter 7, "Designer and Developer Controls" on page 15 describes the Designer and Developer controls which include zip code lookup and pick lists.
- Chapter 8, "Formatting a Form" on page 17 discusses the various techniques available for formatting and designing your form.
- Chapter 9, "Custom Fields" on page 19 explains how to create new custom fields, including fields that perform calculations.
- Chapter 10, "Creating Custom Event Handlers" on page 20 discusses how to use the Event Editor to create custom events, with or without macros, and assign them to controls on your custom form.
- Chapter 11, "Managing Your Forms, Plugins, and Custom Data Objects" on page 38 discusses how to publish custom forms, plugins, and data objects to an Encompass server, as well as how to work with these files outside of your Encompass system.
- Appendix A, "Control Properties Reference" on page 41 provides detailed descriptions of each Form Builder control and its properties.
- Appendix B, "Button Actions" on page 66 lists the actions available to assign to button controls.
- Appendix C, "Loan Custom Field Calculations" on page 70 describes the types of operations you can use in a custom field calculation, and provides examples of how to use each one.

Getting Help

If you need assistance using the Form Builder, several resources are available.

User's Guide

You can download the Encompass Input Form Builder User's Guide as a .pdf file from the Documentation Library at:

http://help.elliemae.com/DocumentationLibrary/360/DocLibrary.htm

Online Help

The online help contains complete documentation for all Form Builder features.

To Access the Online Help:

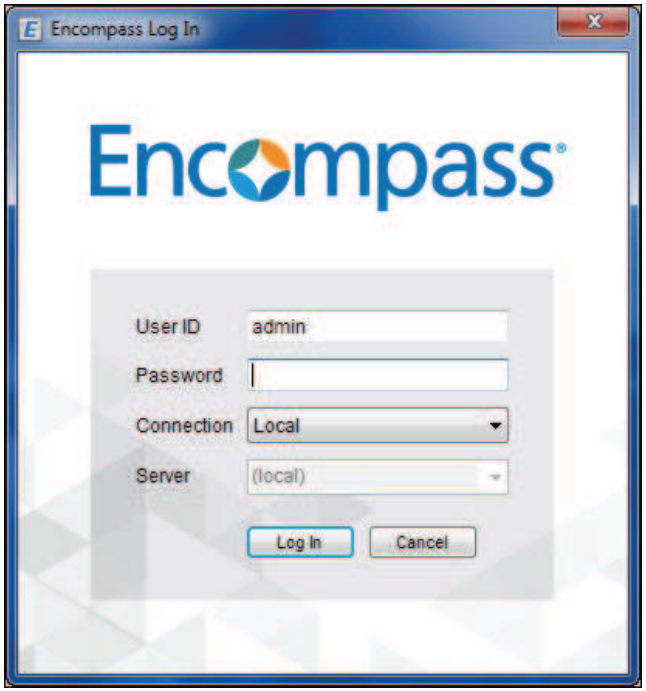
- Click the **Help** button (blue question mark) located above the Controls on your Input Form Builder window or press the **F1** key on your keyboard.

Logging In to the Input Form Builder

The Form Builder is automatically installed with Encompass Banker Edition. Your system administrator will create a user profile for you that includes a User ID and Password.

When you log in to the Form Builder you will type your User ID and Password, and then select **Local** or **Networked** mode. Your system administrator will tell you which mode to use and, if you log in using the **Networked** mode, will provide the name of the server to type in the Server field.

NOTE: *It is strongly recommended that you log in to the Form Builder in **Local** mode while you are building a new form. Or, if you use **Networked** mode, log in to a server that is used for Encompass testing and development. This way your users will not have access to the form while you are making changes to it. Once the form is complete you can publish it to your production server to make it available to your users.*



To Log In to the Input Form Builder:

1 Click **Start**, point to **All Programs**, point to **Ellie Mae Encompass**, and then click **Input Form Builder**.

2 On the Login window, enter the following:

- Login Name: [your assigned User ID]
- Password: [your assigned password]
- Select **Offline** or **Networked**.

NOTE: *If your computer is set up for use only in **Networked** mode, you cannot select **Local** mode.*

- If you are logging in to the **Networked** mode, type the name of the Encompass server provided by your system administrator for development work.

3 Click **Log In**.

The Input Form Builder opens.

Input Form Builder Components

There are three main components on the Form Builder interface, as show in the figure to the right: controls, the form workspace, and the properties window.

Controls

All of the controls used to build a form appear on the left side of the Form Builder window. There are 26 controls to choose from. Add a control to a form by using a simple drag-and-drop operation, or select a control from the Edit menu.

The Form Workspace

The Form Workspace is in the center of the Form Builder window. Add controls to the workspace and then arrange them on the form. You can open multiple workspaces at the same time. Each workspace is represented by a tab at the top of the workspace.

In addition to building a new form, you can build a customized version of any standard input form available in Encompass.

Properties

The Properties window is on the right side of the Form Builder window. When you add a control to the workspace, you define its appearance and behavior by setting values in the Properties window. The available properties are based on the selected controls. (The Form Builder also contains file menus and shortcut keys for the available tasks.)

The Properties window is also where you assign event handlers to controls. Events are triggered when a user performs an action on a form. For example, the user can click a button to trigger a *click event* or click a text box to trigger a *change event*.

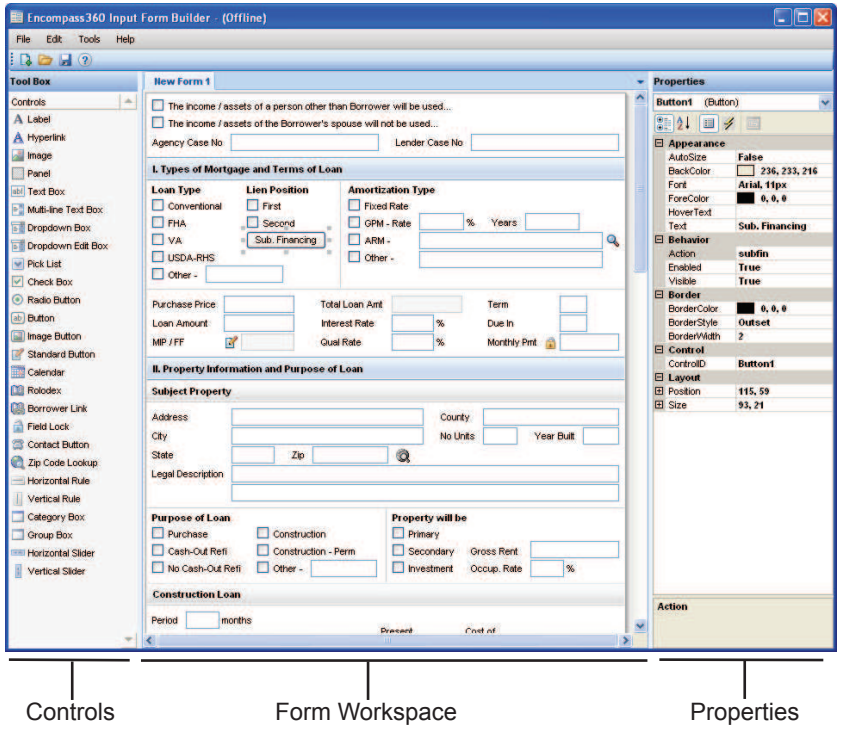
By default the properties are arranged by category. You can also view them alphabetically.

To Change the Properties View:

- Click the **Categorized** button to arrange the properties into categories.

Or, click the **Alphabetic** button to arrange the properties alphabetically.

The Input Form Builder Interface



Chapter 2

Creating a Form

This chapter describes how to start a new custom form and how to save it to your Encompass system. It also discusses form controls, which define how your form appears and behaves.

Starting a New Form

To Start a New (Blank) Form:

- After logging in to the Form Builder, click the **New Form** button.

  Or, on the **File** menu, click **New Form**.

To Start a New Form Using a Template:

Standard Encompass forms (such as a Borrower Summary) can be used as a starting point to decrease the amount of time needed to add controls to your custom form. When using these form templates, the fields are included, but not the logic that controls how the form works within Encompass.

1. After logging in to the Form Builder, click the **Open Form** button.

   Or, on the **File** menu, click **Open From Encompass**.
2. Select the **Standard Input Forms** tab.
3. Select a template, and then click **Open**.

To Open an Existing Custom Form:

1. After logging in to the Form Builder, click the **Open Form** button.

   Or, on the **File** menu, click **Open From Encompass**.
2. Click the **Custom Input Forms** tab, select a form, and then click **Open**.

Understanding Form Controls

An Encompass input form is simply a collection of controls arranged on a blank palette called the Form. Whether you want to add a text field into which a user can enter data, a simple snippet of text such as a field description, or your company's logo, everything you add to the form is part of a control.

Control Properties

Choose from over 25 different control types to create both the behavior and the look and feel you want for your form. These controls are broken down into several categories which are described in upcoming chapters.

Just as a form is made up of controls, each control is defined by its set of properties. Properties are used to customize the appearance and behavior of the control on the form. For example, you can set a label control's ForeColor property to change the color of the text it displays, or set the Action property of a button control to determine what will happen when the user clicks the button.

The ControlID Property

While some controls have more properties than others, every control has the ControlID property. The Control ID is assigned automatically when the control is added to the form. For example, the first text box control placed on the form is assigned the Control ID *TextBox1*, the second text box control is assigned *TextBox2*, and so on.

NOTE: *You can change the Control IDs of the controls. They must be unique, begin with a letter, and contain only the characters A-Z, 0-9, and the underscore character. Chapter 10, "Creating Custom Event Handlers" demonstrates how the control IDs can be used within custom code to modify the attributes of each control at runtime, such as making a control visible or invisible.*

Adding Controls to a Form

Every control can be added to the form using a simple drag-and-drop operation.

Saving a Form

As you build a form, you can easily view and test it in Encompass. When you save your custom form, it is automatically available in the same Encompass system you logged in to to use the Input Form Builder.

For complete information on saving and publishing your custom forms to different Encompass systems, refer to Chapter 11, “Managing Your Forms, Plugins, and Custom Data Objects”

To Save a Form to Your EncompassEncompass System:

1. On the **File** menu, click **Save to Encompass**.
2. On the Save Encompass Form window, type a Form Name, and then click **Save**.
3. Log in to Encompass in the same way that you logged in to the Form Builder (using the same User ID, Password, and Server).
4. Open a loan. click the **Forms** tab, and then select your custom form.

To Save a Form to the File Source:

- On the **File** menu, click **Save Form**.

  Or, click  .

The form is saved to the source location from which it was opened.

Chapter 3

Field Controls

Field controls are used to display data from the underlying loan file, and for the user to input or modify loan data. The following Field controls are available in the Form Builder:

- Calendar
- Check Box
- Dropdown Edit Box
- Dropdown Box
- Multi-line Text Box
- Radio Button
- Text Box

Working With Field Controls

In Encompass, certain fields can only accept one of a set of predefined values. In those cases, it is important that you select a field control type that is appropriate, such as a dropdown box or a set of radio button controls. If you associate a text box with this kind of field, you run the risk of a user entering an invalid value. In this case, Encompass will display an error and clear the value input by the user.

Other fields in Encompass are meant for internal modification only. For example, the MS.START field which represents the start date of the loan. When you associate this field with a Field control, the Form Builder will automatically disable the field to prevent the user from attempting to directly input data. In addition, attempts to enable this field at runtime through code will fail. These types of fields can be used for display purposes only.

The Field Property

When you add a Field control to a form, you must set the control's Field property. This property is the link that joins the visual control to the underlying loan file. Once a control's Field ID is set, the data from that field is automatically populated into the control when the form is loaded in Encompass. Similarly, any changes the user makes within the control will be saved back to this field in the loan.

For example, if a text box's Field property is assigned to field 1109 (the loan amount), that value will automatically display in the field when the user selects the custom form. If the user modifies the value in the field, the loan amount is updated in the loan and all necessary calculations are triggered to ensure the loan's data remains consistent. No additional coding or work is required.

NOTE: *For information about assigning custom field IDs, refer to Chapter 9, "Custom Fields" on page 19.*

Field-level Help

You can enable your form to display field-level help. When the user hovers their mouse pointer over a field, help text associated with the field displays.

The HoverText and HelpKey properties determine what is displayed when users hover their mouse pointer over a field. The HelpKey value is linked to default help text. You can type your own text in the HoverText property field and it will override the default help text.

Calendar 

A calendar allows user to open a calendar pop-up window to select a date for a date field.

To Add a Calendar:

1. Click **Calendar** and drag it to the workspace.
2. To associate the button with a date field, click the **DateField** property.
3. Resize as needed.

Check Box

A check box is an on/off toggle control that allows the user to select one or more options.

To Add a Check Box:

1 Click **Check Box** and drag it to the workspace.

2 Click the **Text** property and type a label title for the check box.

3 Click the **Field** property and then click ... to assign a field ID.

4 Click the **CheckedValue** property and type the value to save into the loan when the check box is selected.

5 Click the **UncheckedValue** property and type the value to save if the check box is not selected.

NOTE: *To save a blank value for a check box that is not selected, remove any values from the UncheckedValue property.*

Check Box Behaviors

Typically, when you add check boxes to a form they are grouped together and designed so users can select more than one related option. However, this is not always the case. For example, if you want the user to specify whether they own or rent their current residence, you can add a "Rent" check box and an "Own" check box, and then restrict the selection to only one. To do this, the check boxes must have the same **Field** property value, but different **CheckedValue** properties.

To Allow Only One Check Box to be Selected:

1 Add two check boxes to the workspace.

2 Click the first check box, click the **Field** property, and then click ... .

3 Select the Field ID **FR0115**. (This is the field ID used to indicate if a borrower rents or owns their current residence.)

4 In the Encompass Input Form Builder window, select the **CheckedValue** for the first check box.

NOTE: *By default, the check box's label matches the selected CheckedValue. Clear the* ***Set control's Text property to selection option's description*** *check box to disable this feature.*

5 If available, select **[Empty] No value assigned** for the UncheckedValue, and then click **OK**.

6 Click the **BehaveAsRadio** property, click the selection arrow, and then select **True**.

NOTE: *When set to True, the field's unchecked value is not saved to the loan when the user clears the check box.*

7 Click the second check box in the workspace.

8 Click the **Field** property, and then click ... .

9 Repeat steps 3-6, being sure to select a different **CheckedValue** than the first check box.

Dropdown Box

Use dropdown boxes to provide a list of fixed options from which the user selects.

To Add a Dropdown Box:

1 Click **Dropdown Box** and drag it to the workspace.

2 Click the **Field** property and then click ... to assign a field ID.

NOTE: *If you assigned a field ID that contains options assigned by Encompass (such as field 52, Borr Marital Status), the options display when you open the Dropdown Options Editor. You can change the Text but not the Value of an option. You also cannot add or delete an option.*

To Create the Options List:

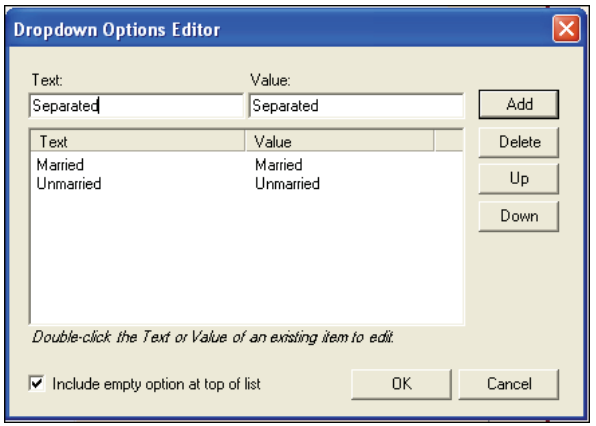
1 Click the **Options** property, and then click ... .

2 On the Dropdown Options Editor window, type the name of an option in the **Text** text box, and then click **Add**.

By default, the same name is added to the **Value** text box.

3 Repeat step 2 to add additional options to the options list.

4 To allow the user to leave the field empty, select **Include empty option at top of list**.



5 Click **OK.**

The options will display below the dropdown box when the user clicks the field.

Dropdown Edit Box

A dropdown edit box displays an options list, but also allows the user to type an alternative custom value.

To Add a Dropdown Edit Box:

1 Click **Dropdown Edit Box** and drag it to the workspace.

2 Click the **Field** property and then click ... to assign a field ID.

NOTE: *If you assign a field ID that contains options assigned by Encompass (such as field 601, Subject Property Building Status), the options display when you open the Dropdown Options Editor. You can change the Text but not the Value of an option. You also cannot add or delete an option.*

To Create the Options List:

1 Click the **Options** property, and then click ... .

2 On the Dropdown Options Editor window, type the name of the option in the **Text** text box, and then click **Add**.

By default, the same name is added to the **Value** text box.

3 Repeat step 2 to add additional options to the options list.

4 To allow the user to leave the field empty, select **Include empty option at top of list**.

5 Click **OK.**

The options will display below the dropdown box when the user selects the field. The user may also type an alternative option in the field.

Multi-line Text Box

A multi-line text box is a multi-line version of the text box control, which provides a field for the user to type extended text, such as comments. When the text reaches the end of a line, it wraps down to the next line.

To Add a Multi-line Text Box:

1 Click **Multi-line Text Box** and drag it to the workspace.

2 Click the **Field** property and assign a field ID.

3 Reposition and resize as needed.

Radio Button

A radio button allows the user to select an option by clicking a button.

To Add a Radio Button:

1 Click **Radio Button** and drag it to the workspace.

2 Click the **Text** property and type a label title.

3 Click the **Field** property and then click ... to assign a field ID.

4 Click the **CheckedValue** property and type the value to save into the loan when the radio button is selected.

NOTE: *If you have a group of radio buttons that you want to work together, but you do not want to assign field IDs, select each radio button and set the same GroupName property. Then you can integrate the group into custom event code.*

Text Box

Text boxes are controls in which users enter and edit single-line Encompass data. The text entered can be both numeric and non-numeric.

To Add a Text Box:

1 Click **Text Box** and drag it to the workspace.

2 Click the **Field** property and then click ... to assign a field ID.

NOTE: *For detailed information about assigning a field ID, refer to "The Field Property" on page 6 .*

Chapter 4

Action Controls

When the user clicks an Action control, it invokes a predefined standard behavior or a custom behavior (defined by the creator of the form). For example, when the user clicks a button control, you can instruct the form to launch a web browser and navigate to a URL. The following Action controls are available in the Form Builder:

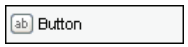
- Borrower Link
- Button
- Contact Button
- Field Lock
- Hyperlink
- Image Button
- Rolodex
- Standard Button

Borrower Link 

Borrower links are used to open the Borrower Contact window, where users can create links between loan files and borrower contacts.

To Add a Borrower Link:

1. Click **Borrower Link** and drag it to the workspace.
2. Select **Borrower** or **CoBorrower** for the **Borrower** property

Button 

Buttons are used to execute a predefined action or a custom event handler. When the user clicks the button, the action is carried out.

To Add a Button:

1. Click **Button** and drag it to the workspace.
2. Click the **Text** property and type a button name.
3. To associate the button with a predefined action, click the **Action** property and type an action.

   For a list and description of the available predefined actions, refer to Appendix B, “Button Actions”

NOTE: *To enable a button to trigger a custom event, add the necessary code using the Event Editor. Refer to Chapter 10, “Creating Custom Event Handlers” for more detailed information.*

Contact Button 

The contact button control opens the Conversation Log worksheet for the contact with which it is associated, and populates it with data from the form. This log tracks interactions with the borrower, service providers, and other contacts related to the loan. Descriptions and comments about conversations with the contact are logged on the worksheet.

To Add a Contact Button:

1. Click **Contact Button** and drag it to the workspace.
2. Click the **CompanyField** property, and then click  .
3. On the Field Selector window, select a field. This field will populate the Company information in the conversation log worksheet.
4. Click **OK**.
5. Repeat steps 2 through 4 to assign field IDs to the **EmailField**, **FirstNameField**, **LastNameField**, and **PhoneField** properties as required.
6. To enable the contact button to trigger a custom event, add the necessary code using the Event Editor. Refer to Chapter 10, “Creating Custom Event Handlers” for detailed instructions.

NOTE: *You can associate a Rolodex control and a Contact Button control with the same field.*

Field Lock 

The field lock controls the lock state of a calculated field. When the field is unlocked, the value is recalculated if related fields are changed. If locked, the value is not recalculated and the user can enter a value in the field.

To Add a Field Lock:

1 Click **Field Lock** and drag it to the workspace.

2 Click the **ControlToLock** property and select the control ID of the field with which to associate the lock.

NOTE: *To enable a field lock to trigger a custom event, add the necessary code using the Event Editor. Refer to Chapter 10, "Creating Custom Event Handlers" for detailed information.*

Hyperlink 

A hyperlink launches a web browser to a specific URL, or executes a custom event handler.

To Add a Hyperlink:

1 Click **Hyperlink** and drag it to the workspace.

2 Click the **Text** property and then type a URL.

3 Click the **ForeColor** property and then click the arrow to select a color for the hyperlink text.

NOTE: *To enable a hyperlink to trigger a custom event, add the necessary code using the Event Editor. Refer to Chapter 10, "Creating Custom Event Handlers" for detailed information.*

Image Button 

An image button behaves in the same way as a button control, but appears as the custom image you provide. To execute an action, the user clicks the image rather than a labeled button.

The .gif, .jpg, .bmp, .png, and .tiff file formats are supported.

To Add an Image Button:

1 Click **Image Button** and drag it to the workspace.

2 Click the **Source** property and click  .

3 Find or type the location of the image file and then click **Open**.

4 To associate the button with a predefined action, click the **Action** property and type an action.

For a list and description of the available predefined actions, refer to Appendix B, "Button Actions"

NOTE: *To enable an image button to trigger a custom event, add the necessary code using the Event Editor. Refer to Chapter 10, "Creating Custom Event Handlers" for detailed information.*

Rolodex 

The Rolodex is used to access business contacts in Encompass. A Rolodex control launches the Address Book pop-up window and populates one or more Field controls based on the user's selection. When a Rolodex control is associated with a field, the user can right-click the field to open the Address Book and select a company or contact to place in the field.

To Add a Rolodex:

1 Click **Rolodex** and drag it to the workspace.

2 Click the **BusinessCategory** property and select a category.

Since this is the first Rolodex control added to the form, the **Control ID** property is *Rolodex1*.

To Associate a Loan Field with the Rolodex:

1 Click a text box on the workspace.

2 Click the **Field** property and then click  to assign a field ID.

3 Click the **Rolodex** property and then click the arrow to select **Rolodex1** (the control ID for the Rolodex control you just added).

4 Click the **RolodexField** property and then click the arrow to select a field to populate from the Rolodex.

When the user right-clicks the text box, the Address Book will open the business category specified in the Rolodex control. The user can then select a contact, and the contact data will be applied to the field.

NOTE: *You can associate more than one text box control to the same Rolodex control ID. For example, you may want to populate the company, phone, and email fields for a contact from the Rolodex. To do this, reference the same Rolodex control ID, such as Rolodex1, for the* ***Rolodex*** *property on each text box.*

Standard Button 

Standard buttons are used to execute a predefined action or a custom event handler. When the user clicks the button, the action is carried out. Standard buttons have the same behavior as buttons, except that you can select from a set of standard, predefined icons to use for the buttons.

To Add a Standard Button:

1. Click **Standard Button** and drag it to the workspace.
2. Click the button **Type** property and select the icon that will be used for the button.
3. Reposition as needed.
4. Click the **Hover Text** property and type the hover text that will appear when a user moves the cursor over the button.
5. To associate the button with a predefined action, click the **Action** property and type an action.

   For a list and description of the available predefined actions, refer to Appendix B, “Button Actions”

To enable a standard button to trigger a custom event, add the necessary code using the Event Editor. Refer to Chapter 10, “Creating Custom Event Handlers” for more detailed information.

Chapter 5

Container Controls

A Container control is any control into which other controls can be added to form logical groups of controls. Once controls are added to a container, they can be easily moved, copied, deleted, disabled, or hidden as a single group. In addition, Container controls can be nested. For example, a category box control can contain a group box control which, in turn, can contain multiple panel controls.

The top-most control in every custom form is the Form. Any control added to the form is contained, directly or indirectly, within the Form Container control.

In addition to the Form, the following Container controls are available in the Form Builder:

- Category Box
- Group Box
- Panel

You can use the category box and group box Container controls to quickly produce custom forms which match the look and feel of standard Encompass forms.

Working With Container Controls

As you drag a new control onto a form, the Form Builder will indicate the container control into which the control will be placed as you drag the control around the screen. For example, if you drag a text box control into a group box, the Form Builder will select the group box control and display its Control ID and properties in the Properties window. When the container you want is selected, you can drop the control into it.

If you insert a control into the wrong container by mistake, or simply wish to move it later, drag the control from its current container over the new container control. When you drag the control outside of the bounds of its current container, the Form Builder will display a red border around the container over which the control is being dragged. If you release the mouse button, the control will be moved into the highlighted container. To drag a control outside of the container's boundaries without moving it into a new container, press the Control key while dragging.

The Form Builder also provides the following hot keys which you can use to select or view the control's container in the form.

+	Selects the container control of the currently selected control.
Shift-+	Highlights the container control of the currently selected control.
-	Selects the immediate children of the currently selected container control.

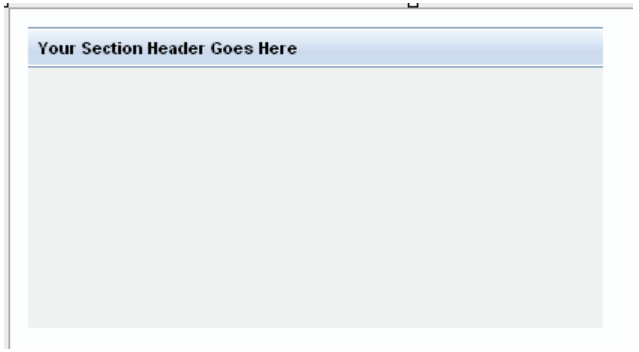
Category Box 

A category box is a container that is typically used on forms to group a section of related fields such as Borrower Information or Credit Information. It includes a top title bar and colored background designed to match the look and feel of the top level containers used on standard Encompass forms. You can customize the look of the category box by changing its values on the Properties window.

When you add controls to a category box they are automatically associated with each other. If you reposition the category box, all of the controls move with it. This makes it easy to organize your layout and keep the controls aligned.

To Add a Category Box:

1 Click **Category Box** and drag it to the workspace.



2 Click the **Title** property and type a title for the category box.

3 To reposition the category box, click it, position the pointer along the edge of the category box to display the four-headed arrow, and then drag the box to the desired location.

4 To resize the category box, position the pointer to display the double-headed arrow, and then drag to the desired size.

Or, expand the **Size** property and then type values for the **Width** and **Height**.

Group Box 

A group box is a container that includes a top title bar designed to match the look and feel of the second level containers used on standard Encompass forms.

When you add controls to a group box they are automatically associated with each other. If you reposition the group box, all of the controls move with it.

To Add a Group Box:

1 Click **Group Box** and drag it to the workspace.

2 Click the **Title** property and type a title for the group box.

3 Reposition and resize as needed.

Panel 

A panel is an “invisible” container used to help organize the controls on your form into easy-to-manage groups. By default, the panel has a visible border which makes it easy to see when placing controls inside it. However, the border can be removed so the container is not visible to users when the form is published to Encompass.

When you insert a group of controls inside a panel, you can move all of them around the form by dragging the panel instead of individually moving each control.

To Add a Panel:

1 Click **Panel** and drag it to the workspace.

2 To remove the border from the panel, click the **BorderStyle** property, click the options arrow, and then select **None**.

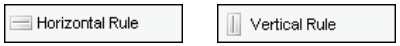
Chapter 6

User Interface Controls

User Interface controls provide visual information to the user but have no bearing on the actual functionality of the form. The following User Interface controls are available in the Form Builder:

- Horizontal Rule
- Label
- Image
- Vertical Rule

Horizontal and Vertical Rules



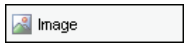
A horizontal rule displays a horizontal divider line on the form that is used to accent the horizontal border of a category box or other section of a form. It can also be used as a separator between sections. By default it is designed to match the look and feel of standard Encompass forms. The vertical rule has the same characteristics as the horizontal rule, except that it displays a vertical divider line.

To Add a Horizontal or Vertical Rule:

1 Click **Horizontal Rule** or **Vertical Rule** and drag it to the workspace.

2 Reposition and resize the control as needed.

Image



Images are graphics that you can insert in a form. The .gif, .jpg, .bmp, .png, and .tiff file formats are supported.

To Add an Image:

1 Click **Image** and drag it to the workspace.

2 Click the **Source** property and click ....

3 Find or type the location of the image file, and then click **Open**.

4 Reposition and resize the image as needed.

Label

A label is typically the most commonly used User Interface control on a form. It is used to place static text on a form. You can then modify the text to describe the control with which it is associated, such as a text box or dropdown box.

To Add a Label:

1 Click **Label** and drag it to the workspace.

2 Click the **Text** property and type a label title.

3 Click the **Font** property and click ....

4 On the Font Editor window, select a font, size, and effect for the label text, and then click **OK**.

5 Click the **ForeColor** property and then click the arrow to select a text color.

Chapter 7

Designer and Developer Controls

Designer Controls

The Form Builder provides Designer controls to assist you in building your form or adding pieces of advanced functionality. What distinguishes a Designer control is that it has no visible display when the form is loaded in Encompass even when it displays within the Form Builder. The following Designer controls are available in the Form Builder:

- Horizontal Slider
- Vertical Slider
- Zip Code Lookup

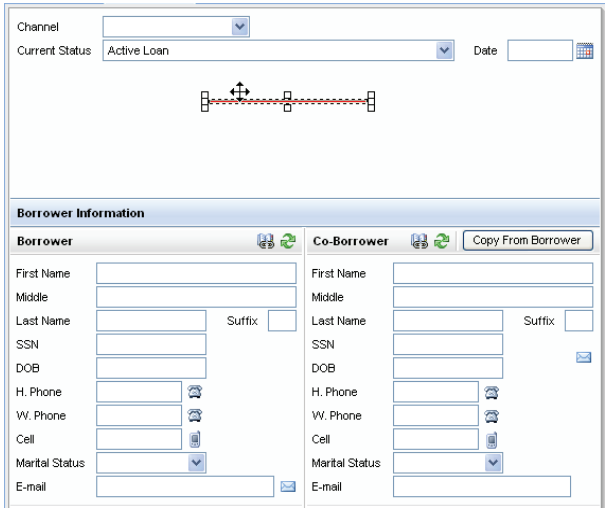
Horizontal and Vertical Slider

Use the horizontal and vertical slider controls to reposition sections of a form. These controls are useful when, for example, you need to add a new text box to the middle of a form. Instead of moving each control individually, use the slider to move all the controls in the area.

Use the horizontal slider to move all of the controls on the right of the slider to the left or right. Use the vertical slider to move all of the controls below the slider up or down.

To Use the Vertical Slider Control:

1. Click **Vertical Slider** and drag it to the workspace.
2. Position the slider above the section you want to move. Every control below the slider will move when the slider moves.
3. Click the **Active** property and then select **True**.
4. Position the mouse pointer on the slider to display the four-headed arrow and then drag the slider to move the controls to the desired location.



5. When you are finished using the slider, you can delete it from the form.

Zip Code Lookup 

The zip code lookup control performs a lookup on a zip code value and populates the corresponding loan fields with the resulting city, county, and state values.

To Add a Zip Code Lookup:

1. Click **Zip Code Lookup** and drag it to the workspace.
2. Click the **CityField** property and click ....
3. Select the field to populate with the city that corresponds to the zip code, and then click **OK**.
4. Click the **CountyField** property and click ....
5. Select the field to populate with the county that corresponds to the zip code, and then click **OK**.
6. Click the **StateField** property and click ....
7. Select the field to populate with the state that corresponds to the zip code, and then click **OK**.
8. Reposition and resize the control as needed. It is common to place the control near the field that will contain the zip code to be resolved.
9. Click the **ZipControl** property and select the field in which the zip code will be entered.

NOTE: *You do not always have to select a CityField, CountyField, and StateField. If you leave one of these properties blank, that data will not be copied to any field.*

Developer Control

A Developer control allows for additional form functionality but requires additional behind-the-scenes coding by the form's author. The following developer control is available in the Form Builder:

- Pick List

Pick List 

The pick list control is used to create a list of dropdown options.

To use the control on your form, you must add code to the *ItemSelected* event handler. For detailed information about adding custom code, refer to Chapter 10, “Creating Custom Event Handlers” on page 20.

To Add a Pick List:

1. Click **Pick List** and drag it to the workspace.
2. Click the **Options** property and then click ....
3. On the Dropdown Options Editor window, type the name of an option in the **Text** text box, and then click **Add**.

   By default, the same name is added to the **Value** text box.
4. Repeat step 3 to add additional options to the options list.
5. To allow the user to leave the field empty, select **Include empty option at top of list**.
6. Click **OK**.
7. Click the **Events** button. 
8. Click **ItemSelected**, and then click ....
9. Type the code, and then click **OK**.

NOTE: *The ItemSelected event handler is not saved until you save the entire form.*

Chapter 8

Formatting a Form

The look and feel of a form are determined by the layout and color.

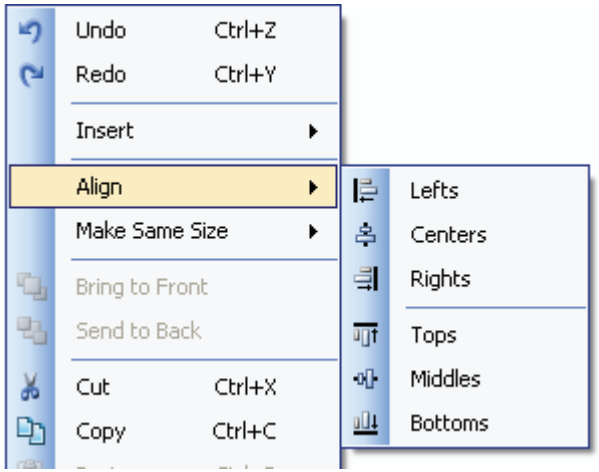
Alignment Options

The alignment options allow you to easily align any selected group of controls. The following example aligns the left sides of label controls.

To Align Controls on a Form:

1 Select three labels by pressing the **Ctrl** key and then clicking each label.

2 Right-click one of the labels, point to **Align**, and then click **Lefts** on the shortcut menu.



The three labels are left-aligned.

NOTE: *When you align multiple controls in this manner, they are always all aligned to the same position as the last control selected.*

Resizing Controls

You can resize individual controls or select multiple controls to make them the same size.

To Resize a Control:

1 Click a control and use the mouse to drag it to the desired width and height. Position the mouse near the edge of the control until the double-sided arrow appears, and then drag the mouse.

Or, expand the **Size** property and type a number in the **Width** and **Height** properties.

To Make Multiple Controls the Same Size:

1 Click a control and resize it to the desired width and height.

2 Select the controls to be changed to this same size by pressing the **Ctrl** key and then clicking each control. Click the control you resized in step 1 last.

NOTE: *When resizing multiple controls, they are all aligned to the size of the last control selected.*

3 Right-click one of the controls, point to **Make Same Size**, and then click **Width** on the shortcut menu.

NOTE: *There are two additional methods to make multiple controls the same size. Refer to the "Resizing Controls" topic in the online help for instructions.*

Color Properties

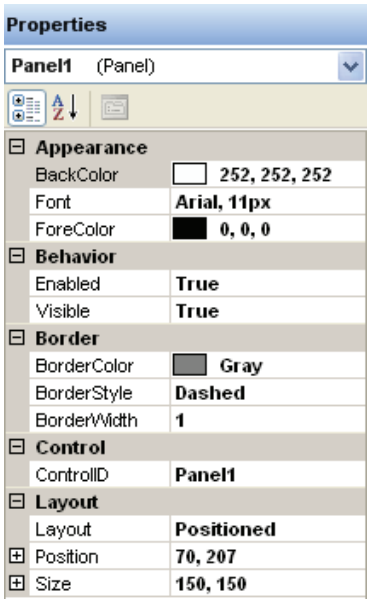
You can change the color of almost every control, including the Form itself, by changing the values on the Properties window.

There are three tabs that display different categories of colors: Web, System, and Custom. There are over 200 colors from which to choose.

- The Web tab contains colors that are compatible with Web sites and most GUI displays.
- The System tab contains colors that are provided by default on all Windows systems. They are the same colors used on all Windows interfaces.
- The Custom tab allows you to create custom colors.

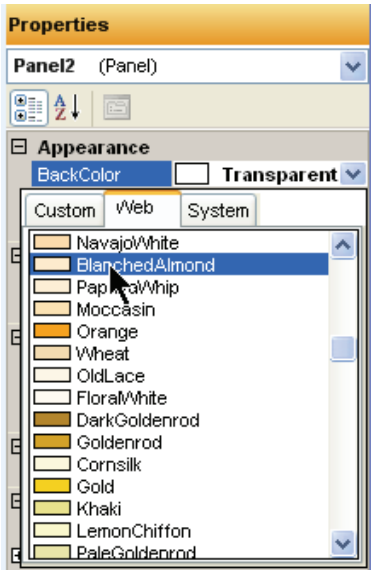
To Set a Background Color:

1 Click a control on the form workspace.



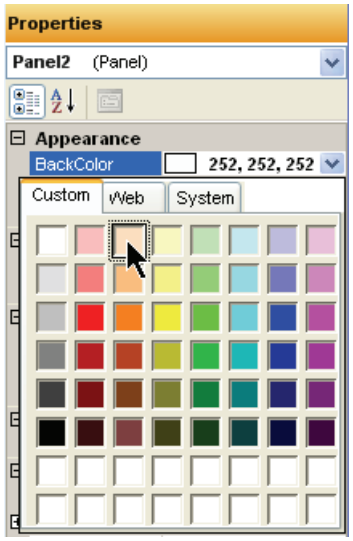
2 Click the **BackColor** property, and then click the arrow.

3 Click a tab and select a color.



To Create a Custom Color:

1 On the **Custom** tab, right click an empty box near the bottom of the color palette.



2 On the Define Color window, type the color values or drag the color selector to create a color, and then click **Add Color**.

Borders

Use the **Border** options on the Properties window to set the color and style of borders.

To Add a Border to a Control:

1 Click a control on the workspace.

2 Click the **BorderWidth** property and then type a number.

3 Click the **BorderColor** property and then click the options arrow.

4 Click a tab and then select a color.

5 Click the **BorderStyle** property and then click the arrow to select a style.

Chapter 9

Custom Fields

Custom Fields

Use the Loan Custom Field Editor to create an unlimited number of loan fields to meet the specific requirements of your business.

There are 18 field types, two of which are dropdown lists of predefined selections that you create.

Predefined Custom Field IDs

There are 100 predefined custom field IDs, in the format "CUST99FV". You can create a description, set the maximum field length, and select a format for these fields. You cannot delete predefined field IDs.

User-Defined Custom Field IDs

You can also create an unlimited number of your own custom field IDs. These custom field IDs always begin with a CX. prefix, followed by an ID of your choice. For example, "CX.TEST.1". You can create the same field information as for the pre-defined field IDs. You can also delete a custom field ID if needed.

Custom Calculations

You can create custom calculations for both pre-defined and user-defined field IDs.

A custom calculation is an expression that returns a number or text value, which is then saved into the associated custom field.

Custom calculations can contain simple arithmetic operations, mathematical operations, text-based operations, date-based operations, values from other fields in the same loan, and branching and logical operations. Refer to Appendix C, “Loan Custom Field Calculations” on page 70 for detailed instructions and examples for creating each type of custom calculation.

To Create Custom Fields:

NOTE: *Your ability to create or modify custom fields using the Custom Field Editor depends on the user settings defined by your system administrator. Users who cannot create new custom fields can add existing custom fields to a form, however they cannot modify them.*

1 Click a control and drag it to the workspace.

NOTE: *Custom field IDs can be assigned to the following controls: dropdown box, dropdown edit box, check box, radio button, multi-line text box, and zip code lookup.*

2 Click the **Field** property, and then click  .

3 Click the Custom Fields tab, and then click the **New/Edit** button.

4 Double-click a field on the Custom Fields list (such as CUST01FV).

5 Type a description of the field.

6 Select a format for the field.

- If you select **String**, type the maximum number of characters in the **Max Length** field. (If you do not set a MaxLength, the number of characters allowed is unlimited.)
- If you select **DROPDOWN or DROPDOWN-Editable**, complete the steps below to add options to the list from which the user will select. The **DROPDOWN-Editable** type allows a user to click from a list or type a value.
- Click the **Add** button to the right of the Options box.
- Type the name of the option and press **Enter**.
- Add additional options as required. To edit an option, click the option and begin typing. Use the **Delete** button to delete entries.
- If you selected **Audit** for the Format, the custom field will contain last-change information for the field ID you specify.
- In the Audit Field field, type the field ID for which to display the last-change information.
- Select one of the three Audit Data options to specify the type of last-change information to display.

7 When finished, click **Save**.

NOTE: *You can also open the Custom Field Editor from the Tools menu.*

To Create a User-Defined Custom Field ID:

1 On the upper-right of the Custom Fields list, click the **New** icon.

2 Type a field ID, and then complete the remaining fields as described in the “To Create Custom Fields” steps above.

Chapter 10

Creating Custom Event Handlers

From the time a form is first displayed to the user, to the time the user navigates to another form or closes the loan, Encompass provides opportunities, called "events," for which you can execute custom code. For example, an event occurs whenever the user clicks a button control or modifies the contents of a text box. By adding your own custom functionality to these events, you can add significant functionality to your form beyond the simple fill-in-the-blanks default behavior.

Within the Form Builder, different controls provide different events. As mentioned, the button control provides a click event while a text box control provides a change event. The full list of available events, and the controls for which they are supported, are detailed later in this chapter.

In order to add your custom code to any event you must author an "event handler." An event handler is the code that Encompass will execute at the time the associated event occurs. For example, you can create an event handler for a button's click event that instructs Encompass to copy the value of one loan field into another. This event can also cause a pop-up box to display to gather additional user input.

Custom event handlers are written in either the VB.NET or the C# programming language, as determined by the EventLanguage property on the Form. You do not, however, need to be fluent in either language to write basic event handlers. The Form Builder provides multiple "macros" that you can use to invoke predefined pieces of code to perform common actions, such as copying a value from one field to another. Users who are fluent in one or both programming languages can author far more complex event behavior using the full power of Microsoft's .NET Framework.

To create a new custom event handler, you should first know the control on which the event will occur.

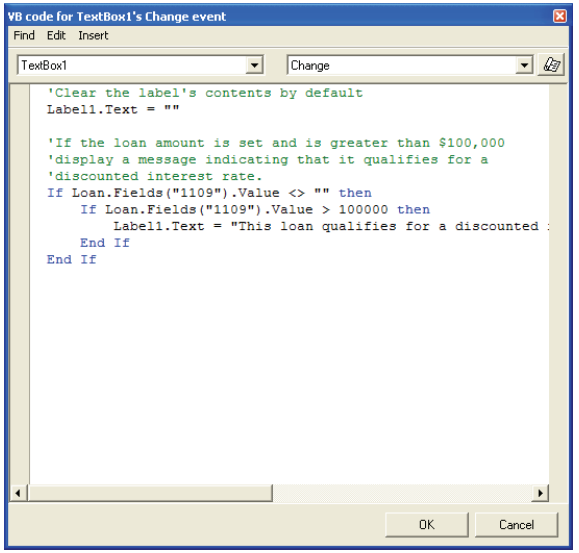
To Write Code To Trigger an Event:

1 Select the control for the desired event in the workspace.

2 On the Properties window, click the **Events** button. 

3 Click the event name, and then click  .

4 Type the code, and then click **OK**.



NOTE: *The new event handler is not saved until you save the entire form.*

Instead of typing the code as you did in step 4, you can use the Macro button to help you author the code. Refer to the “Macros” section on page 25 for more information.

Event Arguments

During the execution of certain events, such as the Format event which is used to perform on-the-fly formatting of user input, data you will need to control the outcome of the event is made available to your event handler.

For example, consider the case where the Format event is being used to force the user to input only characters in the range A through E. With each keystroke, a Format event is triggered on the text box control in which the user is typing. Within your event handler, your code needs to know the text that the user has typed and, if appropriate, needs to reformat it to remove invalid characters.

The following code snippet demonstrates what this code may look like.

```
Dim i as Integer
Dim NewValue as String = " "

For i = 0 to EventArgs.Value.Length - 1
    Dim C as String =
    EventArgs.Value.Substring(i, 1)

    If C >=”A” And C <=”E” then
        NewValue = NewValue & C
    End If
Next
EventArgs.Value=NewValue
```

NOTE: *In this example, the EventArgs.Value property is used to retrieve the new value entered into the text box and to set the updated value which is stripped of all invalid characters.*

Control Events

The following section provides examples of custom event code written for each event type available in the Form Builder.

You can create event handlers for the following events using the Event Editor:

- Click Event
- Change Event
- DataBind Event
- DataCommit Event
- FocusIn Event
- FocusOut Event
- Format Event
- Load Event
- Unload Event

NOTE: *In this section, events supporting EventArgs describe the properties which can be read and/or written in order to affect the outcome of the event.*

Click Event

This event is triggered when the user clicks a control, typically a button.

Click Events are supported by buttons, check boxes, field locks, hyperlinks, image buttons, and radio buttons.

EventArgs Properties:

None

Example:

The following code can be placed in the Click event for a button to open a web page, passing the loan number and loan amount from the current loan as parameters on the URL:

```
'Build the list of arguments to post to the web page
Dim Args as String = " "
Args = "LoanNumber=" & Loan.Fields ("364") .Value
Args = Args & "&Amount=" & Loan.Fields
("1109") .UnformattedValue

'Open the browser to the specified web page
Macro.OpenURL("http://www.mysite.com/DisplayData.asp?" & Args)
```

Change Event

This event is triggered whenever data within the control has changed. It generally occurs when the user leaves the field after making a modification. When this event is called, the data in the control has already been committed to the underlying loan.

NOTE: *Use the DataCommit event to intercept changes to the loan's value before they occur.*

Change events are supported by text boxes, dropdown boxes, and dropdown edit boxes.

EventArgs Properties:

None

Example:

The following code can be placed in the Change event for a text box. The event code below displays a message to the user in a label control (*Label1*) if the loan amount exceeds $100,000.

```
'Clear the label's contents by default
Label1.Text = " "

If Loan.Fields ("1109") .Value <> " " then
    If Loan.Fields("1109").Value > 100000 then
        Label1.Text = "This loan qualifies for a discounted rate!"
    End If
End If
```

DataBind Event

The DataBind event is triggered any time Encompass populates a form control using the data from the loan, a process called "data binding". You can use this event to modify the default behavior of the data binding. For example, you can customize the way in which your form displays the values from the loan.

DataBind events are supported by check boxes, dropdown boxes, dropdown edit boxes, radio buttons, text boxes, and multi-line text boxes.

EventArgs Properties:

Cancel	Boolean	Read/Write	Set this property to True if you have overridden the default data binding behavior or to prevent the default data binding action from occurring.
Value	String	Read/Write	The underlying data value from the loan which will be used to populate the field. Modify this value to change the value to which the control is bound.

Example:

The following code assumes that a text box is placed on the form and associated with Field 981 (line item M1 from the Declarations section on Page 3 of the 1003). The underlying field can take on one of the following values:

- PrimaryResidence
- SecondaryResidence
- Investment

However, we want the text box to display "PR", "SH", or "IP" for these values. To achieve this, we override the DataBind events to translate between the underlying field value and the displayed value. For example, the code in the DataBind event would be:

```
If EventArgs.Value = "PrimaryResidence" then
    EventArgs.Value = "PR"
Else If EventArgs.Value = "SecondaryResidence" then
    EventArgs.Value = "SH"
Else If EventArgs.Value = "Investment" then
    EventArgs.Value = "IP"
Else
    EventArgs.Value = ""
End If
```

DataCommit Event

The DataCommit event is triggered whenever Encompass needs to save modified data from a control into the underlying loan file. You can use this event to modify the default behavior of how data is saved. For example, the value saved into the loan may be derived from the user's input into a text field instead of being the actual value typed into the field.

DataCommit events are supported by check boxes, dropdown boxes, dropdown edit boxes, radio buttons, text boxes, and multi-line text boxes.

EventArgs Properties:

Cancel	Boolean	Read/Write	Set this property to True if you have overridden the default data commit behavior or to prevent the default data commit action from occurring.
Value	String	Read/Write	The underlying data value which will be committed to the loan. Modify this value to change the value that will be committed.

Example:

The following code assumes that a text box is placed on the form and associated with Field 981 (line item M1 from the Declarations section on Page 3 of the 1003). The underlying field can take on one of the following values:

- PrimaryResidence
- SecondaryResidence
- Investment

However, we want the text box to display "PR", "SH", or "IP" for these values.

To achieve this, override the DataCommit events to translate between the underlying field value and the displayed value. For example, you can add the following code to the control's DataCommit event.

```
If EventArgs.Value = "PR" then
    EventArgs.Value = "PrimaryResidence"
Else If EventArgs.Value = "SH" then
    EventArgs.Value = "SecondaryResidence"
Else If EventArgs.Value = "IP" then
    EventArgs.Value = "Investment"
Else
    EventArgs.Value = ""
End If
```

FocusIn Event

The FocusIn event is triggered on a control any time the control gains the input focus. This can happen by virtue of the user clicking the control or by tabbing to the control.

FocusIn events are supported by check boxes, dropdown boxes, dropdown edit boxes, radio buttons, text boxes, and multi-line text boxes.

EventArgs Properties:

None

Example:

The following code can be used to pop-up a selection dialog when the input focus enters a text box named *TextBox2*. The text box is populated based on the user's selection.

```
If TextBox2.Text = " " Then
    If Macro.Confirm ("Is the borrower married?") Then
        TextBox2.Text = "Yes"
    Else
        TextBox2.Text = "No"
    End If
End If
```

FocusOut Event

The FocusOut event is triggered any time the input focus leaves the field. The focus leaves a field when the user clicks another field or tabs out of the field.

FocusOut events are supported by check boxes, dropdown boxes, dropdown edit boxes, radio buttons, text boxes, and multi-line text boxes.

EventArgs Properties:

None

Example:

The following code sets the maximum number of characters allowed in a the text box (*TextBox2*) to a maximum of 25 characters. Any additional characters are truncated.

```
If TextBox2.Text.Length > 25 Then
    TextBox2.Text = TextBox2.Text.Substring (0, 25)
End If
```

Format Event

The format event is used to perform “live” formatting to a field’s value as the user types into a text control.

Format events are supported by Multi-line text boxes, dropdown edit boxes, and text boxes.

EventArgs Properties:

Cancel	Boolean	Read/Write	Set this property to True if you have overridden the default formatting behavior or to prevent the default formatting from occurring.
Value	String	Read/Write	The value to be formatted. Setting this property will cause the field’s default formatting to be applied to the new value.

Example:

The following code demonstrates how to format a 10-digit zip code field by inserting a “-” between the 5th and 6th numbers typed.

```
‘First we need to remove any non-numeric characters
Dim i as Integer
Dim Zip as String = “ “

For i = 0 to EventArgs.Value.Length - 1
        If Char.IsDigit (EventArgs.Value.Substring(i, 1)) Then
        Zip =  Zip & EventArgs.Value.Substring(i, 1)
        End If
Next

' Handle the different formats based on the text length
If Zip.Length <= 5 Then
        EventArgs.Value = Zip
Else If Zip.Length <= 9 Then
EventArgs.Value = Zip.Substring(0, 5) & "-" & Zip.Substring(5)
Else
        EventArgs.Value = Zip.Substring(0, 5) & "-" & Zip.Substring(5, 4)
End If
```

Load Event

The Load event triggers on the Form object when all controls on the form are populated and the form is ready for user input.

Load events are supported by forms only.

EventArgs Properties:

None

Example:

The following code displays or hides a Panel control on the form based on the loan amount of the current loan.

```
If Loan.Fields ("1109") . Value > 100000 Then
    Panel1.Visible = True
Else
    Panel1.Visible = False
End If
```

Unload Event

The Unload event is raised to notify the form that it is about to be closed. The user may have elected to open a new input form or may be closing the loan. This event cannot be used to prevent the form from being unloaded and you should avoid using functionality that displays a user interface.

Unload events are supported by forms only.

EventArgs Properties:

None

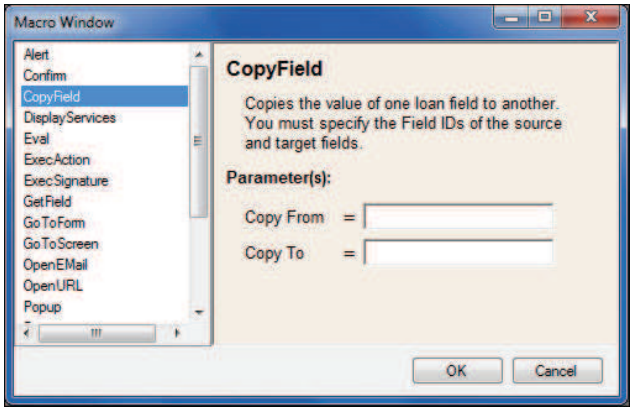
Example:

The following example copies the values from one set of fields to a set of custom fields when the form is unloaded.

```
Macro.CopyField("1109", "CX. LoanAmount")
Macro.CopyField("36", "CX.BorrowerFName")
Macro.CopyField("37", "CX.BorrowerLName")
```

Macros

Macros provide predefined functionality to authors of event handlers to help remove the programming barrier when the desired functionality is simple. Additionally, the Event Editor assists you in using the provided macros by providing a fill-in-the-blanks interface for describing how the macro should behave.



For example, if you wanted a button's click event to cause the value of field 1109 to be copied to the custom field CX.AMOUNT, you could select the CopyField macro from the Macro Assistant and enter the two field IDs in the appropriate spaces. The Macro Assistant will author the appropriate code and place it into the Event Editor window.

As you become more experienced with the macros, you can bypass the Macro Assistant and simply type the macro code directly into the Event Editor, like this:

```
Macro.CopyField("1109", "CX.AMOUNT")
```

Macros can also be combined with more advanced coding techniques for authors who are familiar with VB.NET or C#. This section includes examples in which macros are used within more complex logic.

The following macros are available in the Event Editor:

- Alert
- Confirm
- CopyField
- DisplayServices
- Eval
- ExecAction
- ExecSignature
- GetField
- GoToForm
- GoToScreen
- OpenEmail
- OpenURL
- Popup
- Print
- ResolveZipCode
- Run
- SendKeys
- SetField
- SetFieldEval
- SetFieldNoRules

To Enable a Macro:

1 Select a control on the workspace.

2 On the **Properties** window, click the **Events** button.

3 Click the **Behavior** property and then click

.

4 On the Event Editor window, click the **Macro** button.

5 On the Macro window, select a macro, and then type the appropriate parameters.

6 Click **OK**.

NOTE: *By default the macros are written in VBScript, however you can use different coding languages. For more information, refer to the EventLanguage property on page 50.*

Custom Event Examples Using Macros

The following section provides examples of writing custom event code using each type of macro available in the Form Builder.

Alert

Use the Alert macro to display a pop-up message (with an OK button) to the user.

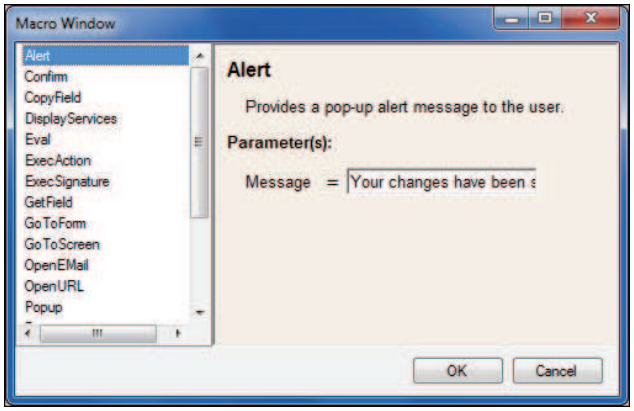
Parameters:

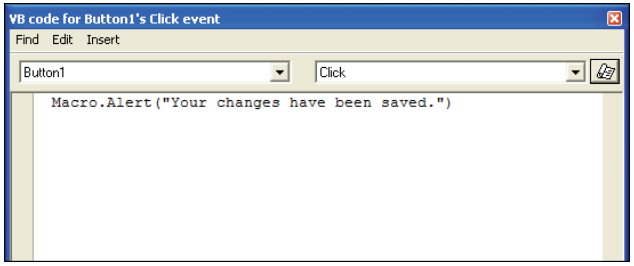
Message: The message to be displayed in the pop-up window.

Return Value:

None

Example:





Confirm

Use the Confirm macro to display a pop-up message to the user with OK and Cancel buttons. The function returns a Boolean to indicate the user's selection.

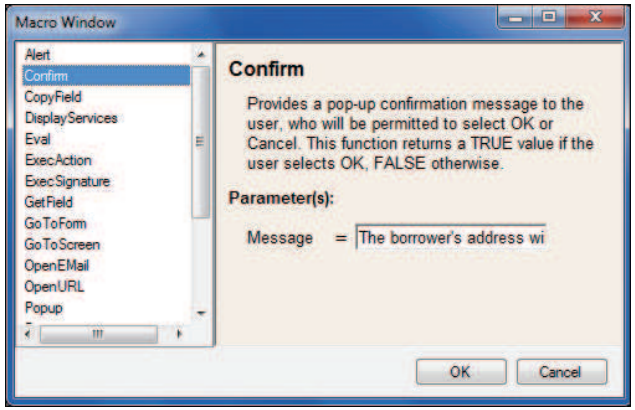
Parameters:

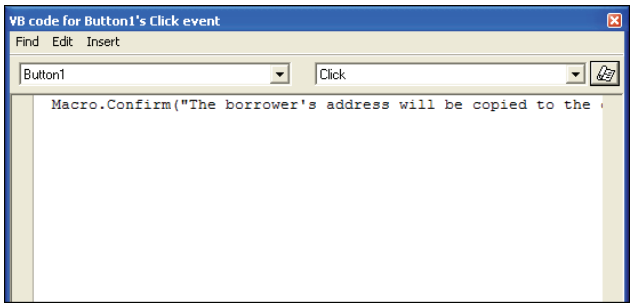
Message: The message to be displayed to the user in the pop-up window.

Return Value:

True, if the user clicks OK; False, if the user clicks Cancel or closes the dialog using the Escape key or the control box.

Example:





CopyField

Use the CopyField macro to copy the value of one loan field into another loan field.

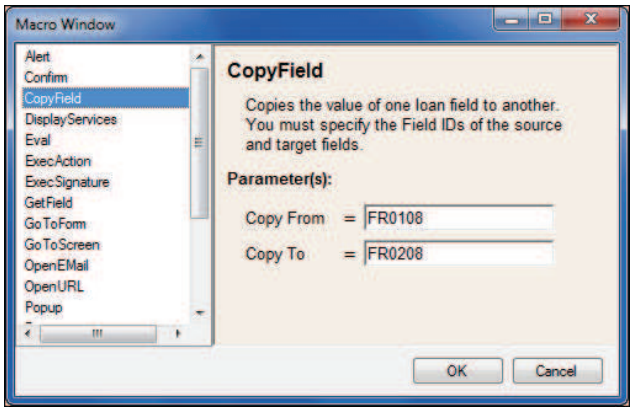
Parameters:

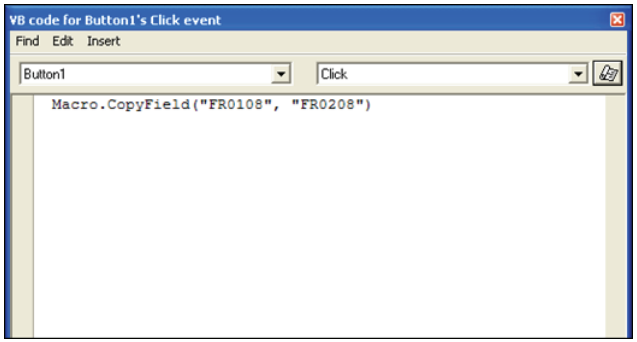
- **Copy From:** The Field ID of the field which is the data source
- **Copy To:** The Field ID which will receive the data

Return Value:

None

Example:





DisplayServices

Use the DisplayServices macro to display the Ellie Mae Network Services winbdow for a particular service type.

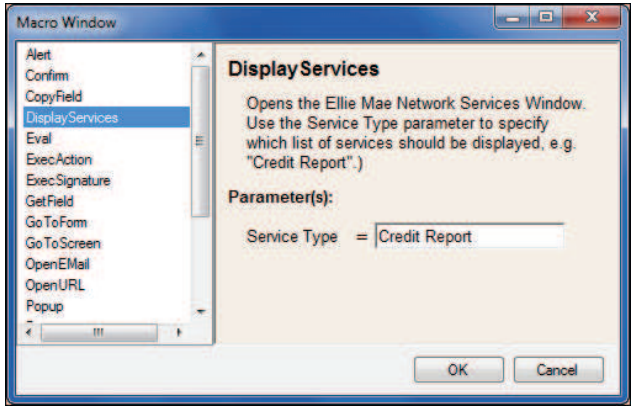
Parameters:

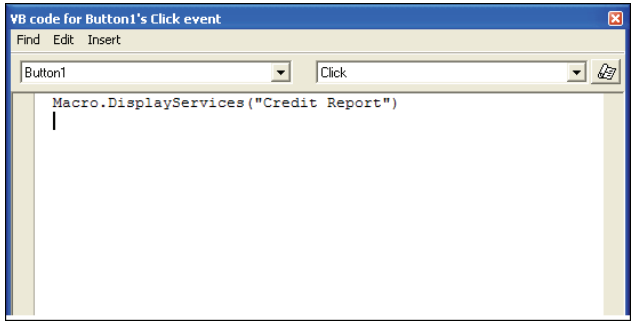
Service Type: The type of Ellie Mae Network service to be used. The set of possible values for this parameter corresponds to the items listed on the Services tab on the loan screen (for example, **Credit Report**, **Lenders**, and **Title & Closing**).

Return Value:

None

Example:





ExecAction

Use the ExecAction macro to execute a predefined action. (Refer to Appendix B, “Button ActionsŽ’ for a list of executable actions.)

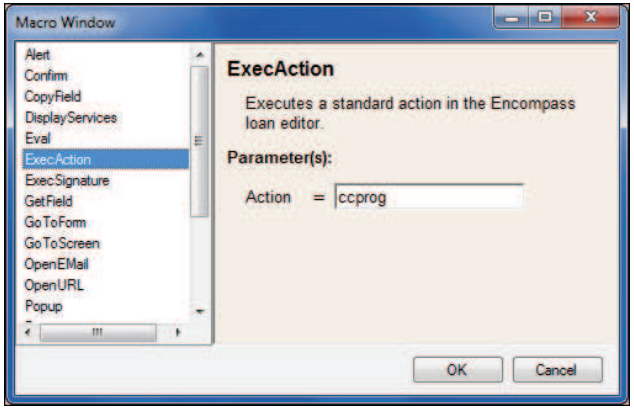
Parameters:

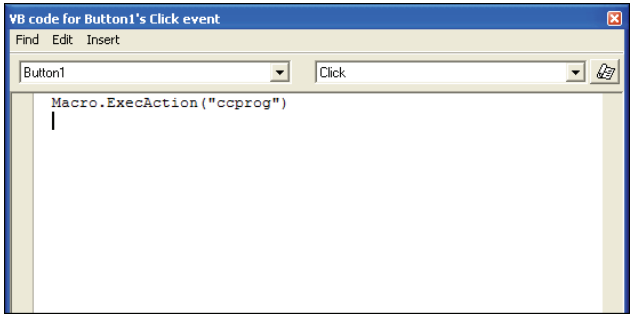
Action: The name of the action to be executed.

Return Value:

None

Example:





ExecSignature

Use the ExecAction macro to launch a service from the Ellie Mae Network.

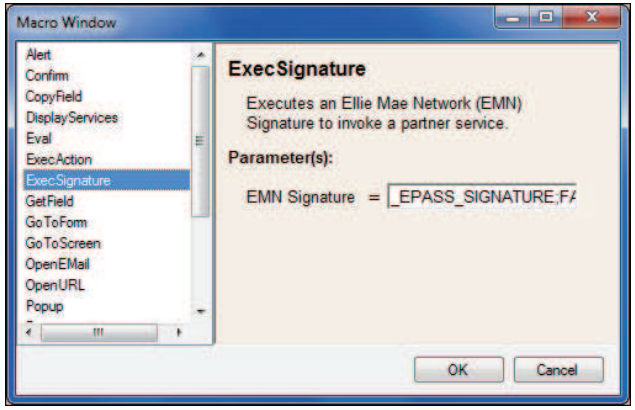
Parameters:

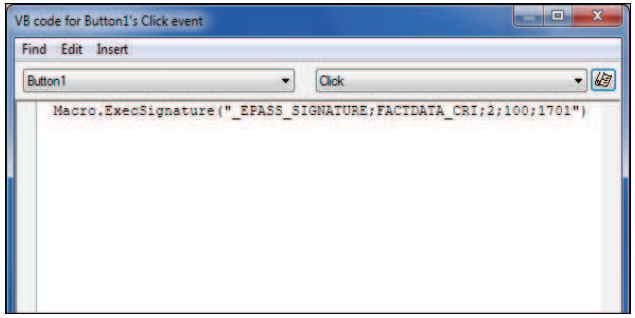
EMNSignature: The “Ellie Mae Network Signature” of the service to be invoked. An EMN Signature will always start with the text “_EPASS_SIGNATURE;”

Return Value:

None

Example:





GoToForm

Use the GoToForm macro to change the input form that is currently displayed to the input form designated in the code.

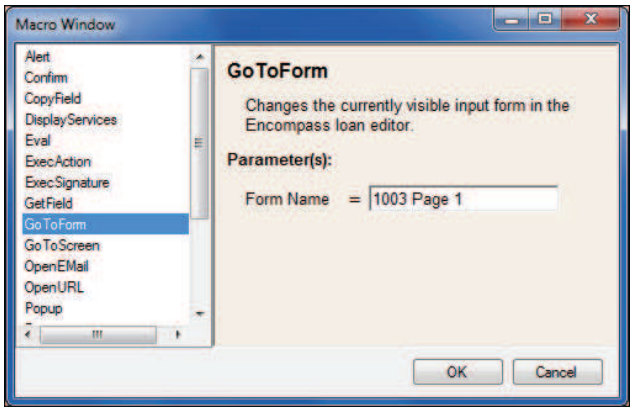
Parameters:

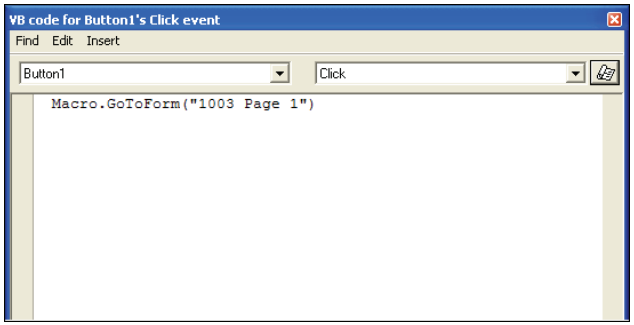
Form Name: The name of the input form to be displayed. This name should match the form name as it appears on the Forms tab in Encompass.

Return Value:

None

Example:





GoToScreen

Use the GoToScreen macro to change the Encompass screen that is currently displayed to the Encompass screen designated in the code.

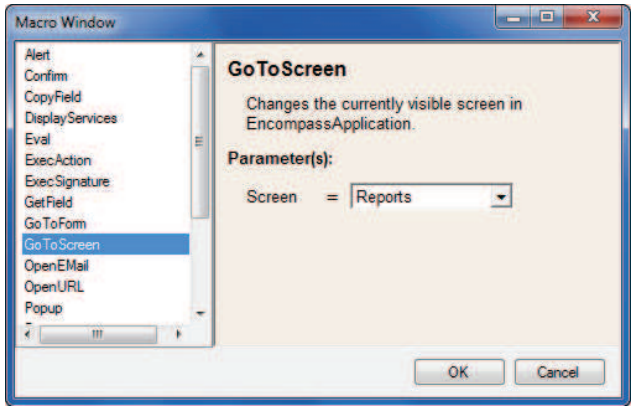
Parameters:

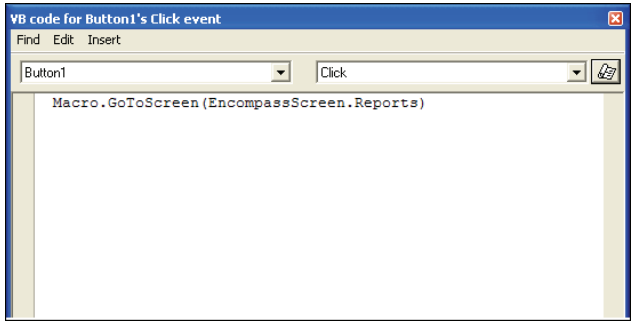
Screen: The name of the screen to be displayed. Select the screen from the **Screen** list on the Macro Window. If the screen you select is not available to the user due to access control restrictions, this macro will not operate.

Return Value:

None

Example:





OpenURL

Use the OpenURL macro to open a browser window that displays the URL designated in the code.

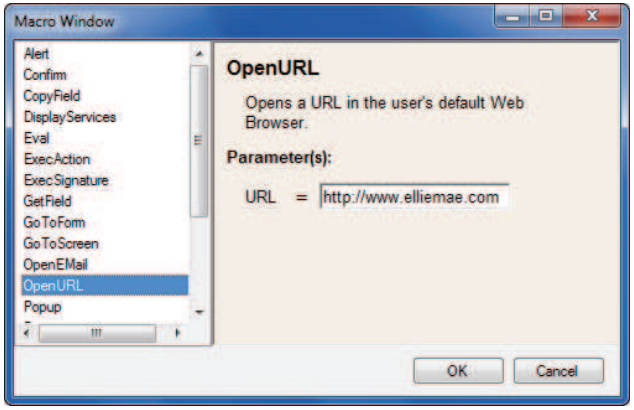
Parameters:

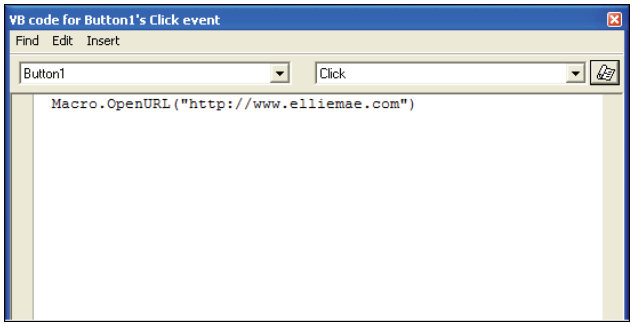
URL: The address of the page to be displayed.

Return Value:

None

Example:





Popup

Use the Popup macro to display an input form in a modal pop-up window.

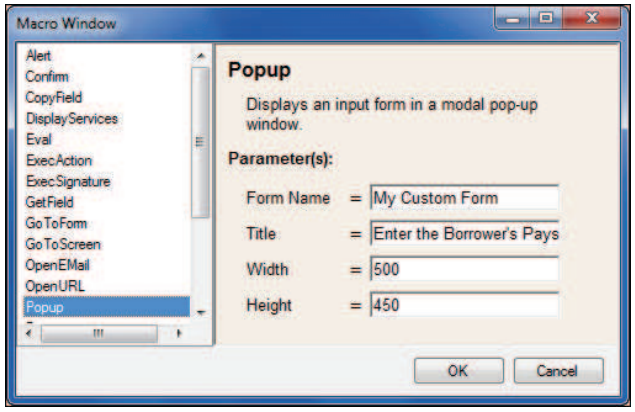
Parameters

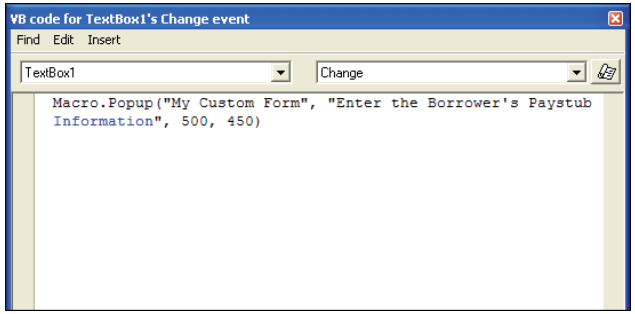
- **Form Name:** The name of the input form to display. This value should match the input form name as it appears on the Forms tab.
- **Title:** The text to display in the title bar of the pop-up window.
- **Width:** The width (in pixels) of the pop-up window.
- **Height:** The height (in pixels) of the pop-up window.

Return Value:

None

Example:





Print

Prints one or more forms for the current loan. For Standard Print Forms, specify the form's name, e.g. "1003 Page 1". For Custom Print Forms, enter the full path to the form, e.g. "public:\My Forms\Appraisal Notification.doc".

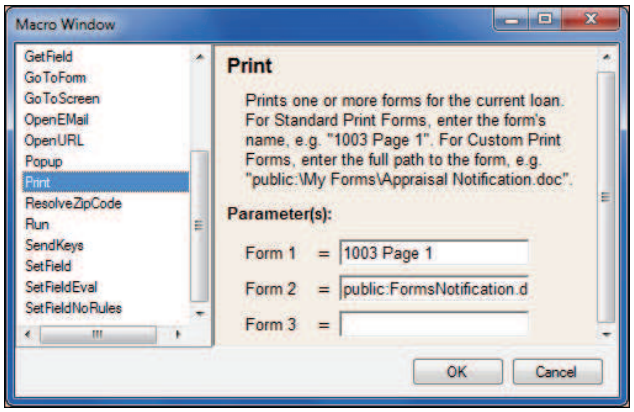
Parameters

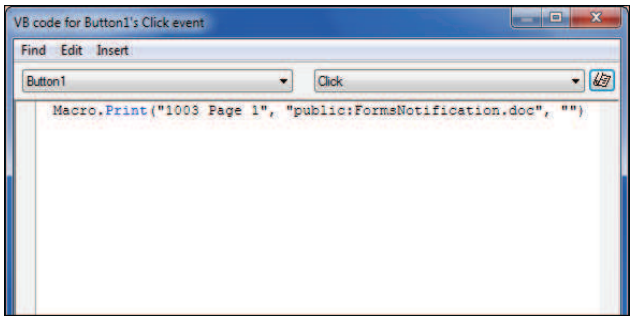
- **Form 1:** The name/path of the first form to be printed.
- **Form 2:** The name/path of the second form to be printed (optional)
- **Form 3:** The name/path of the third form to be printed (optional)

Return Value:

None

Example:





ResolveZipCode

Use the ResolveZipCode macro to perform a lookup on a zip code value and populate the appropriate loan fields with the resulting city, county, and state values.

Parameters:

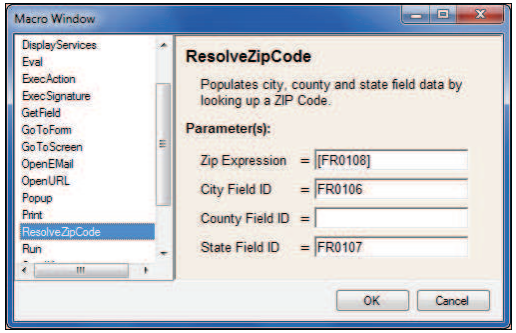
- **Zip Expression:** An expression that is evaluated to determine the zip code to look up.
- **City Field ID:** The field ID of the field to be populated with the city corresponding to the zip code. (This is not a required parameter.)
- **County Field ID:** The field ID of the field be populated with the county corresponding to the zip code. (This parameter is not required.)
- **State Field ID:** The field ID of the field be populated with the state corresponding to the zip code. (This parameter is not required.)

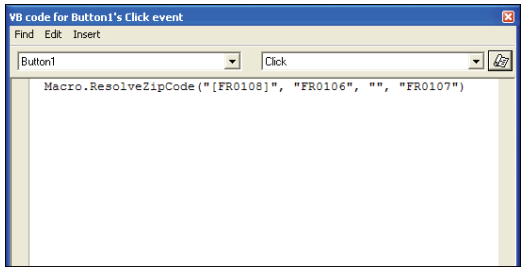
Return Value:

None

Example:

The following macro/code looks up the borrower's current zip code and populates the city and state fields on the form.





Run

Use the Run macro to execute an external program.

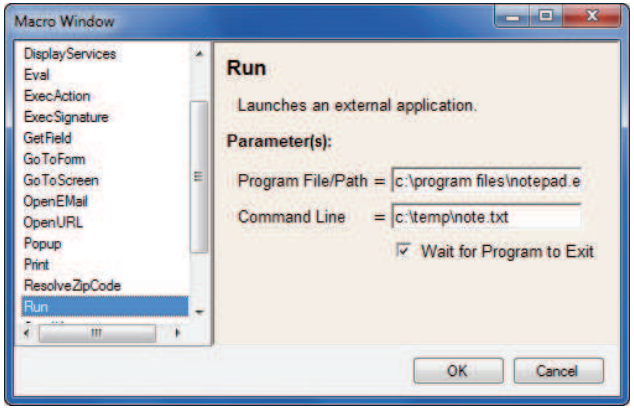
Parameters:

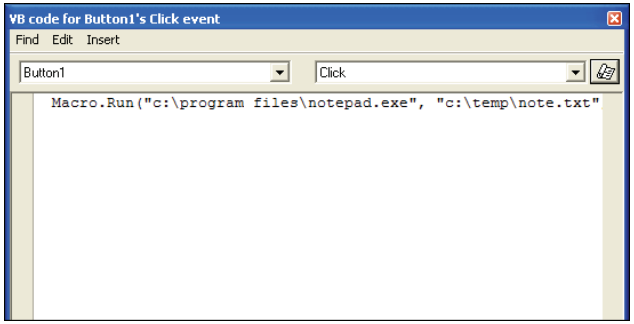
- **Program File/Path:** The path of the executable to be run. The full path must be specified or the executable must reside in a folder included in the user's path environmental variable. You can also specify a file name that has a predefined association to a program (for example, a .pdf document).
- **Command Line:** Any command-line arguments to be passed to the executable.
- **Wait for Program to Exit:** A Boolean indicating if Encompass should suspend itself until the executed program terminates.

Return Value:

None

Example:





SetField

Use the SetField macro to set the value of a single loan field.

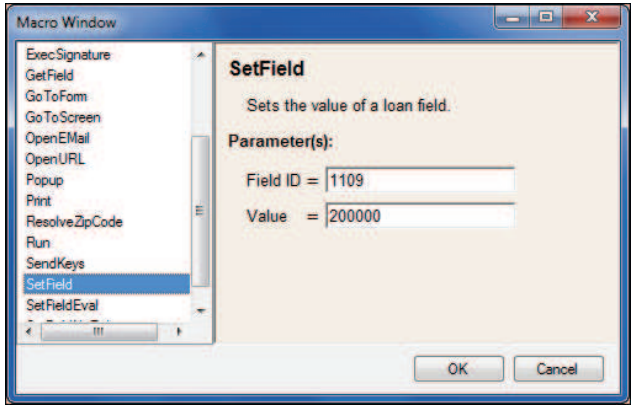
Parameters:

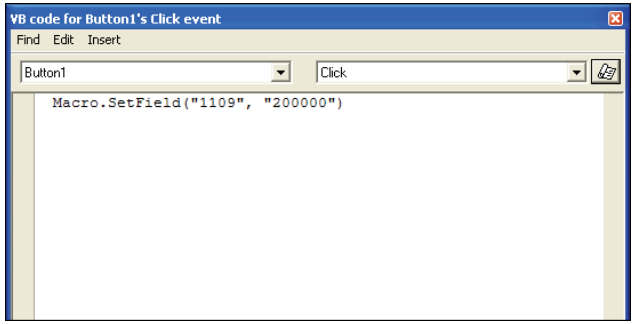
- **Field ID:** The field ID of the loan field to be set.
- **Value:** The value to which to set the field

Return Value:

None

Example:





SetFieldEval

Use the SetFieldEval macro to set the value of a single loan field by evaluating an expression using the Loan Custom Field Calculation syntax.

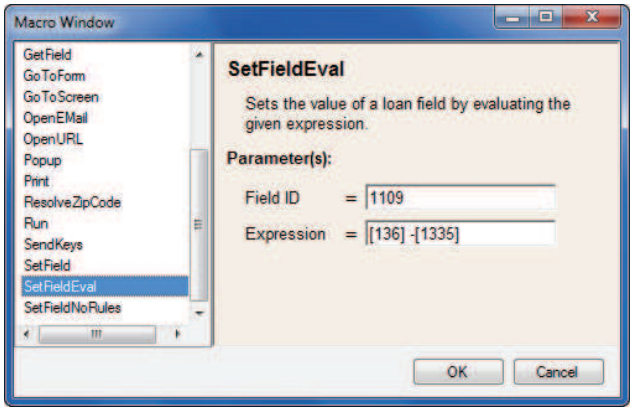
Parameters:

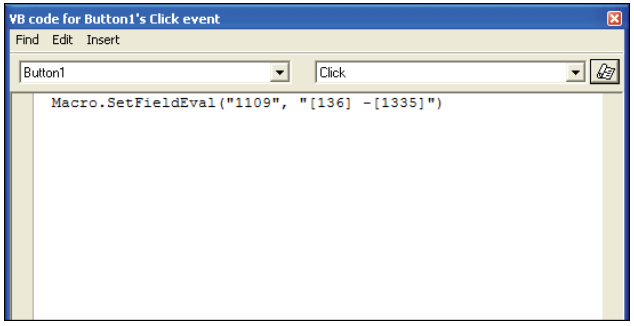
- **Field ID:** The field ID of the loan field to be set.
- **Expression:** The expression to be evaluated, using the same syntax as the Eval macro. For more information, refer to “Eval” on page 34.

Return Value:

None

Example:





SetFieldNoRules

Sets the value of a loan field, ignoring any Field Data or Field Access Rules which would otherwise prevent this action.

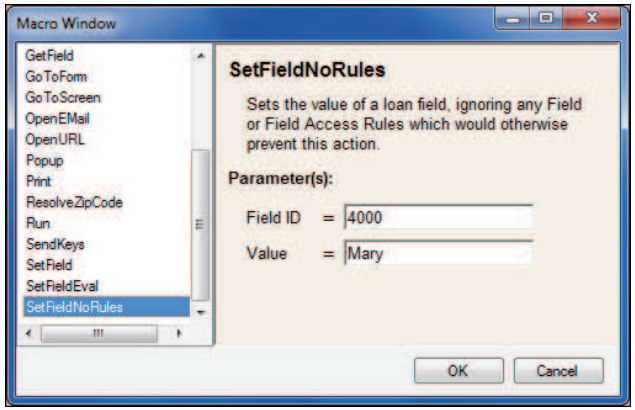
Parameters:

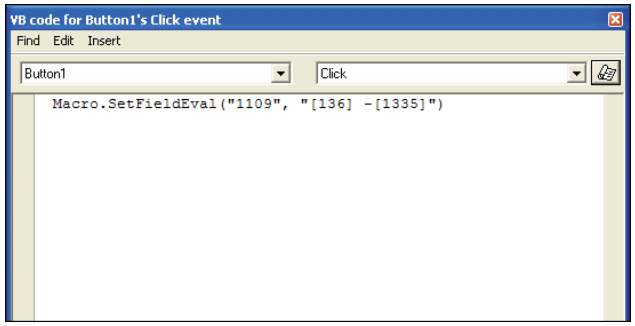
- **Field ID:** The ID of the field to be updated.
- **Value:** The value to be set into the field.

Return Value:

None

Example:





The following examples are more complex and combine multiple macros with custom event code.

Eval

Use the Eval macro to evaluate an expression using the Custom Field Calculation syntax. Refer to Appendix C, “Loan Custom Field Calculations” on page 70 for more information.

Parameters:

- **Expression:** The expression to be evaluated. This expression can contain references to the fields in the current loan (using the square-bracket notation) or employ any of the functions available for use with custom field calculations.

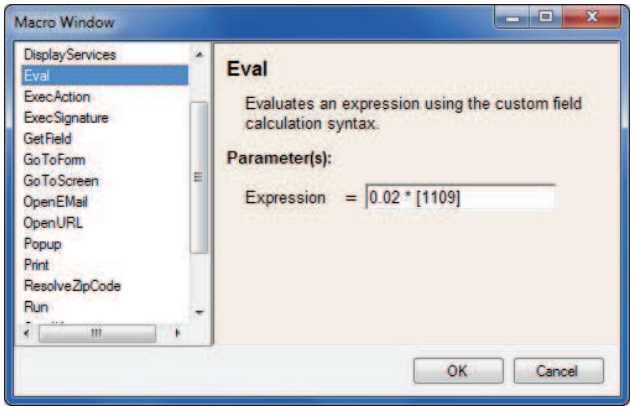
Return Value:

The result of the expression. This value may be numeric, string, or Boolean depending on the expression being evaluated.

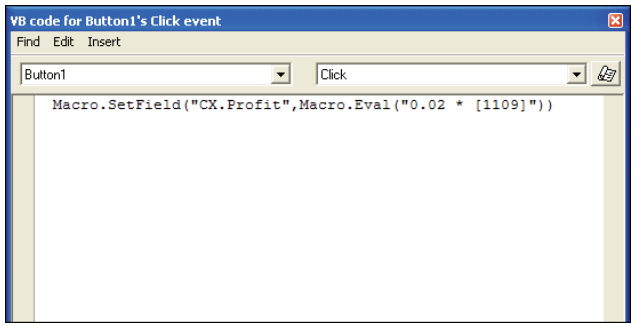
Example 1:

The following example uses two macros, SetField and Eval, to create an event that saves 2% of the loan amount in a custom field.

Add the SetField macro to the Event Editor first, then add the Eval macro/code.

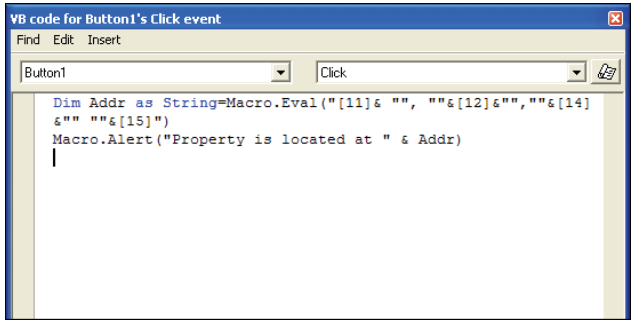


NOTE: *When adding a new macro to the Event Editor, the macro is inserted according to the cursor’s position.*



The following example uses two macros, Eval and Alert, to create an event that displays the full address of the subject property.

Example 2:



GetField

Use the GetField macro to obtain the value of a loan field. You must specify the field ID of the source.

Parameters:

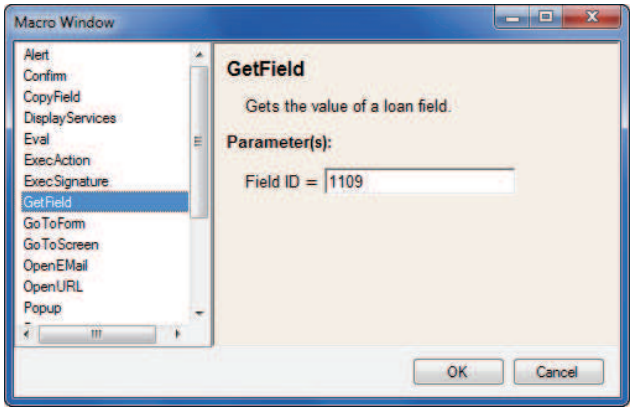
Field ID: The field ID of the loan field that contains the value to be obtained.

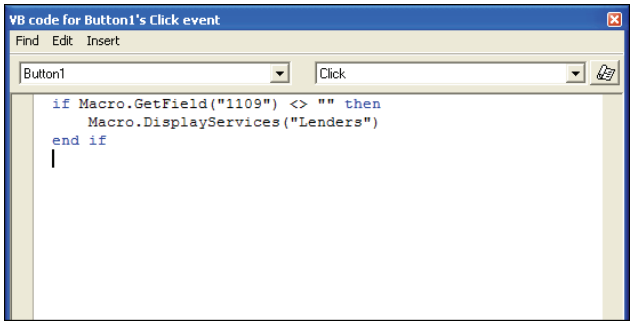
Return Value:

The value of the specified field. This value is displayed as a formatted string. For example, “200,000.00”.

Example:

The following example uses two macros, GetField and DisplayServices. The macro checks loan field 1109. If it contains a value, the macro displays the Lenders Ellie Mae Network Service.





OpenEmail

Use the Open Email macro to open the user’s email client and populate the recipient and subject of the email message.

Parameters:

- **Addresses:** A semicolon or comma-delimited list of email addresses to be placed in the **To** field of the email message.
- **Subject:** The subject of the email message.

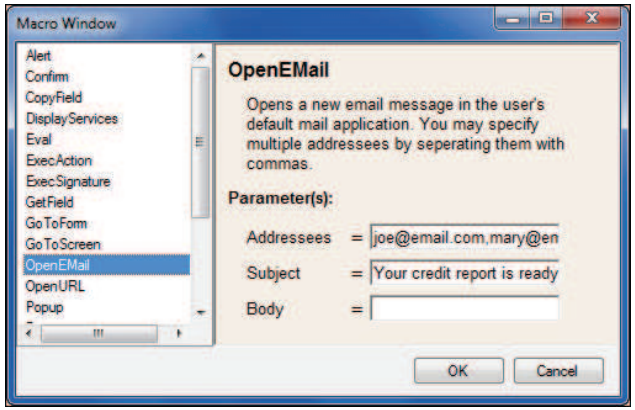
Return Value:

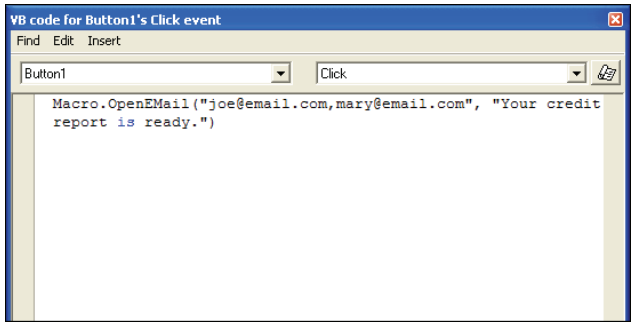
None

Example 1:

The following example uses two macros, Eval and OpenEmail, to create an event that evaluates and pulls addresses from borrower and co-borrower email address fields and opens an email message with the subject “Your credit report is ready.”

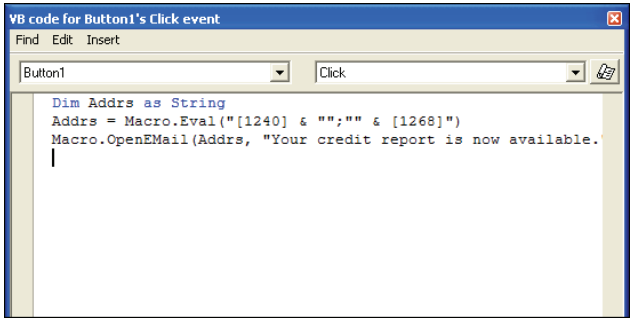
You need to type additional code in addition to adding the macros.





Example 2:

This is an example of creating a custom event by combining macros and custom code. The code below evaluates and pulls addresses from the borrower and co-borrower email address fields on the Borrower Summary. The event is created by typing custom code and then adding the Eval and OpenEmail macros.



SendKeys

Use the SendKeys macro to send keystrokes to the Encompass application.

Parameters:

Key Sequence: The series of keystrokes to send to the Encompass application. Refer to the “Additional Information” section below.

Additional Information:

Each key is represented by one or more characters. To specify a single keyboard character, use the character itself. For example, to represent the letter A, pass in the string *A* to the method. To represent more than one character, append each additional character to the one preceding it. To represent the letters A, B, and C, specify the parameter as *ABC*.

The plus sign (+), caret (^), percent sign (%), tilde (~), and parentheses (()) have special meanings to the SendKeys macro. To specify one of these characters, enclose it within braces ({ }). For example, to specify the plus sign, use *{+}*. To specify brace characters, use *{ {}* and *{ }}*. Brackets have no special meaning to the SendKeys macro, but you must enclose them in braces. In other applications, brackets do have a special meaning that might be significant when dynamic data exchange (DDE) occurs.

To specify characters that are not displayed when you press a key, (ENTER or TAB, for example), and keys that represent actions rather than characters, use the codes in the following table:

Key	Code
BACKSPACE	{BACKSPACE}, {BS}, or {BKSP}
BREAK	{BREAK}
CAPS LOCK	{CAPSLOCK}
DEL or DELETE	{DELETE} or {DEL}
DOWN ARROW	{DOWN}
END	{END}
ENTER	{ENTER}
ESC	{ESC}
HELP	{HELP}
HOME	{HOME}
INS or INSERT	{INS} or {INSERT}
LEFT ARROW	{LEFT}
NUM LOCK	{NUMLOCK}
PAGE DOWN	{PGDN}
PAGE UP	{PGUP}
PRINT SCREEN	{PRTSC}
RIGHT ARROW	{RIGHT}
SCROLL LOCK	{SCROLLLOCK}
TAB	{TAB}
UP ARROW	{UP}
F1	{F1}
F2	{F2}
F3	{F3}

Key	Code
F4	{F4}
F5	{F5}
F6	{F6}
F7	{F7}
F8	{F8}
F9	{F9}
F10	{F10}
F11	{F11}
F12	{F12}
F13	{F13}
F14	{F14}
F15	{F15}
F16	{F16}
Keypad add	{ADD}
Keypad subtract	{SUBTRACT}
Keypad multiply	{MULTIPLY}
Keyboard divide	{DIVIDE}

To specify keys combined with any combination of the SHIFT, CTRL, and ALT keys, preceded the key code with one or more of the following codes.

Key	Code
SHIFT	+
CTRL	^
ALT	%

To specify that any combination of SHIFT, CTRL, and ALT should be held down while several other keys are pressed, enclose the code for these keys in parentheses. For example, to specify to hold down SHIFT while E and C are pressed, use *+(EC)*. To specify to hold down SHIFT while E is pressed, followed by C without SHIFT, use *+EC*.

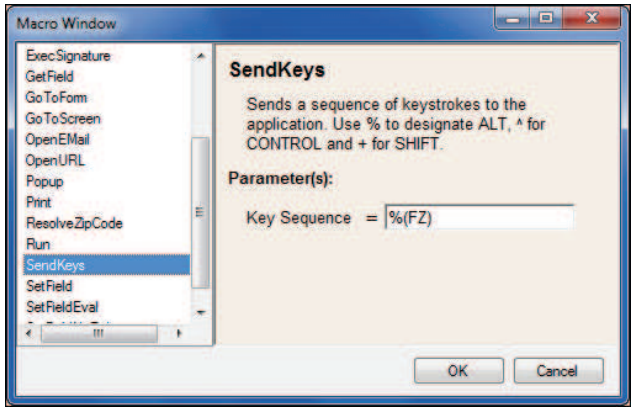
To specify repeating keys, use the form {key number}. You must put a space between the key and number. For example, *{LEFT 42}* means press the LEFT ARROW key 42 times; *{h 10}* means press H 10 times.

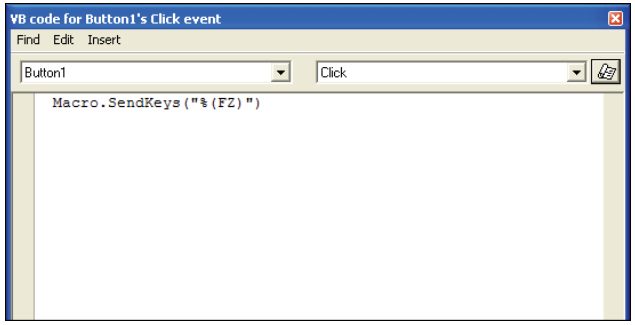
Return Value:

None

Example:

The following sample code opens the REGZ-TIL form by sending an ALT-F, ALT-Z key combination.





Chapter 11

Managing Your Forms, Plugins, and Custom Data Objects

When you create a new form and save it, that form immediately becomes available for use within the Encompass system to which the Form Builder is connected. If you open an existing form, modify it and save it, any user who has access to that form will see those changes the next time they open it in Encompass. If you have configured a plugin or custom data object, you can also make those items available for use immediately within the Encompass system to which the Form Builder is connected.

Because of the immediate nature of this process, it is strongly recommended that you first create and test your custom input forms, plugins, and data objects on a non-production system. In this way you can create and validate your forms locally before pushing them to your production system. Alternatively, you could configure a “mirror” Encompass system that matches your production system and use it specifically to validate and review new custom forms before moving them to production.

In order to allow you to effectively move forms, plugins, and custom data objects between Encompass systems, the Form Builder provides four different options:

- Import or Export a Form
- Upload a Plugin or Custom Data Object
- Import or Export a Package
- Publish a Package

Form Import/Export

Individual forms can be directly exported to the local system’s hard drive as a single file (with the extension .emfrm). These forms can then be copied, emailed, or otherwise transferred to any other computer running the Form Builder. The recipient can import the form from the file and, if desired, save it to their Encompass system. Alternatively, they could simply modify the import form and export it again without ever saving it to their local Encompass system.

NOTE: *Exporting a form exports only the form itself. Any custom field definitions or codebase assemblies used by the form are not included. To include the custom field definitions and codebase assemblies you need to export a package (detailed on page 39).*

To Import a Form From Outside of Your Encompass System:

1. Log in to the Form Builder.
2. Click the **Import Form** button.

   Or, on the **File** menu, click **Import**.
3. Locate and select the desired .emfrm file, and then click **Open**.

NOTE: *All Encompass forms have an .emfrm file extension.*

To Export a Form Outside of Your Encompass System:

1. Log in to the Form Builder.
2. On the **File** menu, click **Export**.
3. Find the desired location for the file, type a new File Name for the form, and then click **Save**.

The form is saved as an .emfrm file.

NOTE: *.emfrm files can be attached to an email message and sent to other recipients. However, these form files can only be opened using the Input Form Builder.*

Uploading Plugins and Custom Data Objects

Plugins and custom data objects created by your company for use with Encompass can be directly uploaded to the Encompass system to which the Form Builder is connected. Once uploaded, the plugin or data object is available to publish, as part of a forms package, to your company’s Encompass server. For more information about publishing plugins or custom data objects to the Encompass server, view “Package Publishing” on page 40.

To Upload a Plugin From Outside of Your Encompass System:

1. Log in to the Form Builder.
2. On the **Tools** menu, click **Manage Customizations**.
3. On the Manage Customizations window, click the **Plugins** tab.
4. Click the **Upload** button.
5. Type or browse to the location of the plugin file, and then click **Open**.
6. Click **OK** to acknowledge the notification message informing you that the plugin will become active within Encompass immediately.
7. Repeat steps 4-6 to import additional plugins.
8. When finished, click **Close**.

NOTE: *The Encompass plugins are saved as DLL files. The file type is classified as an* Application Extension.

To Upload a Custom Data Object From Outside of Your Encompass System:

1. Log in to the Form Builder.
2. On the **Tools** menu, click **Manage Customizations**.
3. On the Manage Customizations window, click the **Custom Data Objects** tab.
4. Click the **Upload** button.
5. Type or browse to the location of the data object file, and then click **Open**.
6. Click **OK** to acknowledge the notification message informing you that the data object will become active within Encompass immediately.
7. Repeat steps 4-6 to import additional data objects.
8. When finished, click **Close**.

Downloading Plugins and Custom Data Objects

Once you successfully upload a plugin or custom data object file to the Encompass system to which the Form Builder is connected, you can download a copy of the file to your system's local hard drive as needed.

To Download a Copy of a Plugin or Custom Data Object File to your Local Hard Drive:

1. Follow the steps provided in this guide to upload a plugin or custom data object file.
2. On the Manage Customizations window, click the **Plugins** or **Custom Data Objects** tab.
3. Click the file to download, and then click the **Download** button.
4. Type or browse to the location where you want to download a copy of the file, and then click **Save**.
5. Repeat steps 2-4 to download additional files.
6. When finished, click **Close**.

Package Import/Export

Similar to the Form Import/Export option, the Form Builder's packaging capabilities allow for the creation of a single package file (with the .empkg extension) which can contain multiple custom forms as well as all of the supporting codebase assemblies and custom field definitions that they utilize, along with plugins and custom data objects.

Packaging allows a set of forms and all of their dependencies to be bundled together into a single unit of distribution, ensuring consistency between the source and target Encompass systems. Once you have created a package on the source system, it can be physically transferred to another Encompass system and imported using the Form Builder's Package Import Wizard. Using this technique, third-party or contract service providers can author custom forms without requiring direct access to your Encompass system. By delivering a package file to you, you can be assured that you are receiving a consistent, up-to-date version of all aspects of the forms.

To Create and Export a Package:

1. On the **Tools** menu, click **Package Export Wizard**.
2. On the Package Export Wizard window, select the forms and any additional files to be included in the package, and then click **Next**.
3. Select the custom field definitions to include in the package, and then click **Next**.
4. Select the plugins to include in the package, and then click **Next**.
5. Select the custom data objects to include in the package, and then click **Next**.
6. Click **Export**.
7. On the Save the Export Package window, find the desired location for the package file, type a new File name for the package, and then click **Save**.
8. Click **OK** in response to the completion message.

The selected items are saved as an .empkg file.

To Import a Package:

1. On the **Tools** menu, click **Package Import Wizard**.
2. On the Package Import Wizard window, type or browse to the location of the package file, click **Open**, and then click **Next**.
3. Click **Import**.

4 Click **OK** in response to the completion message.

5 If prompted, click **Yes** to open each form included in the package in the Form Builder's Form Workspace.

Package Publishing

Package publishing is essentially exporting a package from one Encompass system, and then importing it into another in a single set of steps. To publish a package from one system to another, you must have the ability to log into both systems from the computer on which the Form Builder is running.

In such an environment, you must first log into the Form Builder using the login information for the "source" Encompass system (the one from which the package will be drawn) and then publish to the "target" Encompass system using the Package Publishing Wizard.

Package publishing is ideal for companies using split development and production Encompass systems for form creation. In this type of environment, both Encompass systems are generally accessible on the same network, making package publishing the easiest way to ensure that all features of the package are transferred as a single unit.

To Publish a Form, Plugin or Data Object to an Encompass System Outside of Your Encompass System:

1 On the **Tools** menu, click **Package Publishing Wizard**.

2 On the Package Publishing Wizard window, select the form, and then click **Next**.

3 Select the custom field definitions to publish, and then click **Next**.

4 Select the plugins to publish, and then click **Next**.

5 Select the custom data objects to publish, and then click **Next**.

6 Click **Next** to confirm publishing.

7 Provide your login information for the Encompass server where you want the form published.

8 Click **Publish**.

9 Click **OK** in response to the completion message.

Your form is saved to the specified server and listed on the **Forms** tab.

NOTE: *You can include one or more custom input forms in the package you are publishing. All forms, custom fields, plugins, and data objects included in the package are saved to the Encompass server you specify. Custom code base assemblies can also be included.*

Appendix A

Control Properties Reference

Every Form Builder control has specific properties that define its behavior, appearance, or value. Use the Properties window to modify a control's properties.

The best way to familiarize yourself with the control properties is to experiment. Change the properties of different controls in the Properties window, save the form, and then view the form in Encompass.

This appendix lists the properties for each control. The tables match the default Categorized view of the Properties window.

To Change the Properties View to Categorized:

- Click the **Categorized** button  to arrange the property types into categories.

Borrower links are used to open the Borrower Contact window, where users can create links between loan files and borrower contacts.

Behavior	
Enabled	Sets the initial state of the pick list as enabled (True) or disabled (False). If set to False the pick list is dimmed.
Visible	If set to True, the pick list is visible. If False, the pick list is not visible.
Control	
ControlID	Used to identify the control. For example, the first link on the form will be assigned the control ID BorrowerLink1 while the next link is assigned BorrowerLink2. You can change the control ID as needed. The ID can contain numerals or letters, but it must begin with a letter.
Layout	
Position	Controls the relative position (in pixels) of the link on the form. The X value determines left to right, the Y value determines top to bottom position. Click the Position property and type values or expand the row to view the X and Y properties and make your changes.
Size	Controls the width and height of the link (in pixels). Click the Size property and type values or expand the row to view the Width and Height properties and make your changes.

A button is used to execute a predefined action or a custom event handler.

Appearance	
AutoSize	If set to True, the button automatically adjusts its size to match the length of the Text. If set to False, the button does not automatically adjust its size.
BackColor	The background color of the button. Choose from web, system, or custom colors.
Font	Refers to a type family. Controls the appearance of the text that displays on the button including size, bold, italic, and underline.
ForeColor	The color of the text that displays on the button. Choose from web, system, or custom colors.
HoverText	The text that displays when the user hovers the mouse pointer over the button.
Text	The text that displays on the button.
Behavior	
Action	The action the button performs when clicked. For example, the *copybrw* action copies data from the Borrower section of the Borrower Summary Form to the Co-Borrower section.
Enabled	Sets the initial state of the button as enabled (True) or disabled (False). If set to False, the button is dimmed.
Visible	If set to True, the button is visible. If set to False, the button is not visible.
Border	
BorderColor	The color of the border appearing around the button. Choose from web, system, or custom colors.
BorderStyle	Controls the style of the border. Typically the Outset border style is preferred.
BorderWidth	Controls the width of the border (in pixels).
Control	
ControlID	Used to identify the control. For example, the first button on the form will be assigned the control ID *Button1* and the next button is assigned *Button2*. You can change the control ID as needed. The ID can contain numerals or letters, but it must begin with a letter.
Layout	
Position	Controls the relative position (in pixels) of the button on the form. The X value determines left to right, the Y value determines top to bottom position. Click the Position property and type values or expand the row to view the X and Y properties and make your changes.
Size	Controls the width and height of the button (in pixels). Click the Size property and type values or expand the row to view the Width and Height properties and make your changes.

Calendars allows user to open a calendar pop-up window to select a date for a date field.

Appearance	
HoverText	The text that displays when the user hovers the mouse pointer over the calendar.
Behavior	
Enabled	Sets the initial state of the contact button as enabled (True) or disabled (False). If set to False, the calendar is dimmed.
Visible	If set to True, the calendar is visible. If False, the calendar is not visible.
Control	
ControlID	Used to identify the control. For example, the first calendar on the form will be assigned the control ID Calendar1 while the next calendar is assigned Calendar2. You can change the control ID as needed. The ID can contain numerals or letters, but it must begin with a letter.
Data	
DateField	The Encompass date field associated with the calendar. ***NOTE:*** *Custom field IDs created from this property are assigned to the selected control.*
FieldSource	If the loan has a piggyback loan associated with it, determines whether the source for the field is the current loan file or the piggyback loan file.
Layout	
Position	Controls the relative position (in pixels) of the calendar on the form. The X value determines left to right, the Y value determines top to bottom position. Click the Position property and type values or expand the row to view the X and Y properties and make your changes.
Size	Controls the width and height of the calendar (in pixels). Click the Size property and type values or expand the row to view the Width and Height properties and make your changes.

A category box is typically used on forms to designate a section of related fields such as Borrower Information or Credit Information. By default, it is designed to match the look and feel of the standard Encompass forms. When you add controls to a category box they are automatically associated with each other. If you reposition the category box, all of the controls move with it.

Appearance	
BackColor	The background color of the category box. Choose from web, system, or custom colors.
Font	Refers to a type family. Controls the appearance of the text including size, bold, italic, and underline. Every control inside the category box adopts this font property unless otherwise specified.
ForeColor	The color of the text. Choose from web, system, or custom colors. Every control inside the category box adopts this color property unless otherwise specified.

Behavior	
Enabled	Sets the initial state of the field as enabled (True) or disabled (False). If set to False, all of the controls within the category box are disabled.
Visible	If set to True, the category box is visible. If set to False, the category box is not visible. If set to False, none of the controls within the category box are visible.
Border	
BorderColor	The color of the border that displays around the category box. Choose from web, system, or custom colors.
BorderStyle	Controls the style of the border. The default value is None, meaning no border displays around the category box.
BorderWidth	Controls the width of the border (in pixels).
Control	
ControlID	Used to identify the control. For example, the first category box on the form will be assigned the control ID *CategoryBox1* and the next category box is assigned *CategoryBox2*. You can change the control ID as needed. The ID can contain numerals or letters, but it must begin with a letter.
Header	
HeaderBackColor	Controls the color of the category box's header (also called the section header).
HeaderFont	Refers to a type family. Controls the appearance of the section header's text including size, bold, italic, and underline.
HeaderForeColor	The color of the section header's text. Choose from web, system, or custom colors.
Title	The title of the category box.
Layout	
Layout	Determines the location of the control on the form, relative to other controls (Flow) or in an absolute X and Y location (Positioned). The Flow option is most often used with category box or group box controls. For example, if the Layout of two controls is set to Flow, the first control is placed at the top-left of the form and the second is placed directly to the right or below it. If you increase or decrease the size of the control, the second control automatically moves to maintain its position next to the first. (Flow is similar to traditional HTML page layouts.)
Position	Controls the relative position (in pixels) of the category box on the form. The X value determines left to right, the Y value determines top to bottom position. Click the Position property and type values or expand the row to view the X and Y properties and make your changes.
Size	Controls the width and height of the category box (in pixels). Click the Size property and type values or expand the row to view the Width and Height properties and make your changes.

A check box that allows the user to select one or more options allows by selecting the box.

Appearance	
BackColor	The background color that displays inside the check box. Choose from web, system, or custom colors.
Font	Refers to a type family. Controls the appearance of the text displayed in the check box including size, bold, italic, and underline.
ForeColor	The color of the text displayed in the check box. Choose from web, system, or custom colors.
HelpKey	An index into the Help system to provide default hover text for the field. This property is typically the same as the field ID. Controls on standard Encompass forms are all assigned with HelpKey values. Based on this property, appropriate hover text is associated with the control. Custom fields need an original HelpKey.
HoverText	The text that appears when the user hovers the mouse pointer over the field. This text overrides the default text referenced by the HelpKey property.
Text	The text that displays next to the check box on the form.
Behavior	
BehaveAsRadio	If set to True, the field's unchecked value is not saved to a database when the user clears the check box. If set to False, the field's unchecked value is saved to a database when the user clears the check box.
Enabled	Sets the initial state of the field as enabled (True) or disabled (False). If set to False, the check box is dimmed.
Visible	If set to True, the check box is visible. If set to False, the check box is not visible.
Border	
BorderColor	The color of the border appearing around the check box and its label. Choose from web, system or custom colors.
BorderStyle	Controls the style of the border. The default property is None meaning no border appears around the check box.
BorderWidth	Controls the width of the border (in pixels).
Control	
ControlID	Used to identify the control. For example, the first check box on the form will be assigned the control ID *CheckBox1* and the next check box is assigned *CheckBox2*. You can change the control ID as needed. The ID can contain numerals or letters, but it must begin with a letter.
Data	
CheckedValue	The value stored in the loan field when the check box is selected.
Field	The Encompass field ID associated with the check box field. ***NOTE:*** *Custom field IDs created from this property are assigned to the selected control.*
UncheckedValue	The value stored in the loan field when the field is cleared. (See the **BehaveAsRadio** property description.)

Layout	
Position	Controls the relative position (in pixels) of the check box on the form. The X value determines left to right, the Y value determines top to bottom position. Click the Position property and type values or expand the row to view the X and Y properties and make your changes.
Size	Controls the width and height of the check box (in pixels). Click the Size property and type values or expand the row to view the Width and Height properties and make your changes.
TabIndex	Determines the tab order of the check boxes on the form.

The contact button control opens the Conversation Log worksheet for the contact with which it is associated, and populates it with data from the form. This log tracks interactions with the borrower, service providers, and other contacts related to the loan. Descriptions and comments about conversations with the contact are logged on the worksheet.

Appearance	
HoverText	The text that displays when the user hovers the mouse pointer over the contact button.
Behavior	
ContactMethod	Select Phone or Email. When the user clicks a contact button, the contact method property determines if a Conversation Log worksheet and Microsoft Outlook are opened (Email) or if a Conversation Log worksheet is opened (Phone).
Enabled	Sets the initial state of the contact button as enabled (True) or disabled (False).
Visible	If set to True, the contact button is visible. If set to False, the contact button is not visible.
Control	
ControlID	Used to identify the control. For example, the first Contact Button on the form will be assigned the control ID *ContactButton1* and the next Contact Button is assigned *ContactButton2*. You can change the control ID as needed. The ID can contain numerals or letters, but it must begin with a letter.
Data	
CompanyField	The field ID that will populate the contact's company name on the Conversation Log worksheet.
EmailField	The field ID that will populate the contact's email address on the Conversation Log worksheet.
FirstNameField	The field ID that will populate the contact's first name on the Conversation Log worksheet.
LastNameField	The field ID that will populate the contact's last name on the Conversation Log worksheet.
PhoneField	The field ID that will populate the contact's telephone number the Conversation Log worksheet.

Layout	
Position	Controls the relative position (in pixels) of the Contact Button on the form. The X value determines left to right, the Y value determines top to bottom position. Click the Position property and type values or expand the row to view the X and Y properties and make your changes.
Size	Controls the width and height of the Contact Button (in pixels). Click the Size property and type values or expand the row to view the Width and Height properties and make your changes.

Dropdown boxes provide a list of options for the user to select from.

Appearance	
BackColor	The background color that displays inside the dropdown box. Choose from web, system, or custom colors.
Font	Refers to a type family. Controls the appearance of the selection list text including size, bold, italic, and underline.
ForeColor	The color of the selection list text. Choose from web, system, or custom colors.
Behavior	
Enabled	Sets the initial state of the dropdown box as enabled (True) or disabled (False). If set to False, the dropdown box is dimmed.
Visible	If set to True, the dropdown box is visible. If False, the dropdown box is not visible.
Control	
ControlID	Used to identify the control. For example, the first dropdown box on the form will be assigned the control ID *DropdownBox1* and the next dropdown box is assigned *DropdownBox2*. Change the control ID as needed. The ID can contain numerals or letters, but must begin with a letter.
Data	
Field	The Encompass field ID associated with the dropdown box field. ***NOTE:*** *Custom field IDs created from this property are assigned to the selected control.*
Options	The list of options that displays when the user clicks the dropdown box. Click  to view and edit the list. If you assign a field ID that contains options assigned by Encompass (such as field 52, Borr Marital Status), the options display when you open the Dropdown Options Editor. You can change the Text but not the Value of an option. You also cannot add or delete an option.
Layout	
Position	Controls the relative position (in pixels) of the dropdown box on the form. The X value determines left to right, the Y value determines top to bottom position. Click the Position property and type values or expand the row to view the X and Y properties and make your changes.
Size	Controls the width and height of the dropdown box (in pixels). Click the Size property and type values or expand the row to view the Width and Height properties and make your changes.
TabIndex	Determines the tab order of the dropdown boxes on the form.

Dropdown Edit Box

A dropdown edit box, when clicked, displays a list of options. Users can select an option or type an alternative value in the field.

Behavior	
Enabled	Sets the initial state of the dropdown edit box as enabled (True) or disabled (False). If set to False, the dropdown edit box is dimmed.
Visible	If set to True, the dropdown edit box is visible, If False, the dropdown edit box is not visible.
Control	
ControlID	Used to identify the control. For example, the first dropdown edit box on the form will be assigned the control ID *DropdownEditBox1* and the next dropdown edit box is assigned *DropdownEditBox2*. You can change the control ID as needed. The ID can contain numerals or letters, but it must begin with a letter.
Data	
Field	The Encompass field ID associated with the dropdown edit box field. ***NOTE:*** *Custom field IDs created from this property are assigned to the selected control.*
Options	The list of options that displays when the user clicks the dropdown edit box. Click  to view and edit the list. If you assign a field ID that contains options assigned by Encompass (such as field 601, Subject Property Building Status), the options display when you open the Dropdown Options Editor. You can change the Text but not the Value of an option. You also cannot add or delete an option.
Layout	
Position	Controls the relative position (in pixels) of the dropdown edit box on the form. The X value determines left to right, the Y value determines top to bottom position. Click the Position property and type values or expand the row to view the X and Y properties and make your changes.
Size	Controls the width and height of the dropdown edit box (in pixels). Click the Size property and type values or expand the row to view the Width and Height properties and make your changes.
TabIndex	Determines the tab order of the dropdown edit boxes on the form.

The field lock controls the lock state of a calculated field. When the field is unlocked the value is recalculated if related fields are changed. If locked, the value is not recalculated and the user can enter a value in the field.

Appearance	
HoverText	The text that displays when the user hovers the mouse pointer over the field lock.
Behavior	
ControlToLock	The control ID of the field associated with the field lock control.
Enabled	Sets the initial state of the field lock as enabled (True) or disabled (False).
Visible	If set to True, the field lock icon is visible. If set to False, the field lock icon is not visible.
Control	
ControlID	Used to identify the control. For example, the first field lock on the form will be assigned the control ID *FieldLock1* and the next field lock is assigned *FieldLock2*. You can change the control ID as needed. The ID can contain numerals or letters, but it must begin with a letter.
Layout	
Position	Controls the relative position (in pixels) of the field lock on the form. The X value determines left to right, the Y value determines top to bottom position. Click the Position property and type values or expand the row to view the X and Y properties and make your changes.
Size	Controls the width and height of the field lock (in pixels). Click the Size property and type values or expand the row to view the Width and Height properties and make your changes.

The Form is the top-most control in every custom form. It contains the default form panel that displays when you open a new workspace.

Appearance	
BackColor	The color of the form background. Choose from web, system, or custom colors.
Font	Refers to a type family. Controls the appearance of the text including size, bold, italic, underline. Every control on the form adopts this font property unless otherwise specified.
ForeColor	The color of the text. Choose from web, system, or custom colors. Every control on the form adopts this color property unless otherwise specified.
Border	
BorderColor	The color of the border that displays around the control. Choose from web, system, or custom colors.
BorderStyle	Controls the style of the border. The default property is None meaning no border is displayed.
BorderWidth	Controls the width of the border (in pixels).
Control	
ControlID	Used to identify the control. By default, the control ID is *Form1*. You can change the control ID as needed. The ID can contain numerals or letters, but it must begin with a letter.
Implementation	
CodeBase	Designates the .NET assembly and form class from which the control will inherit functionality. Use this property to link to a custom code base assembly that is written using Encompass SDK. For more information, refer to the *Encompass SDK Programmer's Guide* (for Banker Edition).
EventLanguage	The programming language, Visual Basic or C#, used to write form event code. Use only one programming language when building a form. Using both languages one the same form causes conflicts.

A group box is a second level header that typically appears inside a category box. Like the category box, it is designed to match the look and feel of standard Encompass forms. When you add controls to a group box they are automatically associated with each other. If you reposition the group box, all of the controls move with it.

Appearance	
BackColor	The background color of the group box. Choose from web, system, or custom colors.
Font	Refers to a type family. Controls the appearance of the text including size, bold, italic, and underline. Every control inside the group box adopts this font property unless otherwise specified.
ForeColor	The color of the text. Choose from web, system, or custom colors. Every control inside the group box adopts the color property unless otherwise specified.
Behavior	
Enabled	Sets the initial state of the field as enabled (True) or disabled (False). If set to False, all of the controls within the group box are disabled.
Visible	If set to True, the group box is visible. If set to False, the group box is not visible and none of the controls within the group box are visible.
Border	
BorderColor	The color of the border that displays around the group box. Choose from web, system, or custom colors.
BorderStyle	Controls the style of the border. The default property is None meaning no border is displayed.
BorderWidth	Controls the width of the border (in pixels).
Control	
ControlID	Used to identify the control. For example, the first group box on the form will be assigned the control ID *GroupBox1* and the next group box is assigned *GroupBox2*. You can change the control ID as needed. The ID can contain numerals or letters, but it must begin with a letter.
Header	
HeaderFont	Refers to a type family. Controls the appearance of the group header's text including size, bold, italic, and underline.
HeaderForeColor	The color of the group header's text. Choose from web, system, or custom colors.
Title	The title of the group box.
Layout	
Layout	Determines the location of the control on the form, relative to other controls (Flow) or in an absolute X and Y location (Positioned). The Flow option is most often used with category box or group box controls. For example, if the Layout of two controls is set to Flow, the first control is placed at the top-left of the form and the second is placed directly to the right or below it. If you increase or decrease the size of the control, the second control automatically moves to maintain its position next to the first. (Flow is similar to traditional HTML page layouts.)

Position	Controls the relative position (in pixels) of the group box on the form. The X value determines left to right, the Y value determines top to bottom position. Click the Position property and type values or expand the row to view the X and Y properties and make your changes.
Size	Controls the width and height of the group box (in pixels). Click the Size property and type values or expand the row to view the Width and Height properties and make your changes.

A horizontal rule control displays a horizontal divider and is used to accent the horizontal border of a category box or other section of a form. It can also be used as a separator between sections. By default, it is designed to match the look and feel of standard Encompass forms.

Behavior	
Visible	If set to True, the horizontal rule is visible. If set to False, the horizontal rule is not visible.
Control	
ControlID	Used to identify the control. For example, the first horizontal rule on the form will be assigned the control ID *HorizontalRule1* and the next horizontal rule is assigned *HorizontalRule2*. You can change the control ID as needed. The ID can contain numerals or letters, but it must begin with a letter.
Layout	
Position	Controls the relative position (in pixels) of the horizontal rule on the form. The X value determines left to right, the Y value determines top to bottom position. Click the Position property and type values or expand the row to view the X and Y properties and make your changes.
Size	Controls the width and height of the horizontal rule (in pixels). Click the Size property and type values or expand the row to view the Width and Height properties and make your changes.

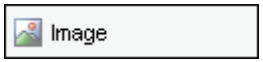
A horizontal slider is a Design control that moves controls up or down within the form. Any control that intersects with the slider moves as the slider moves. You can use the slider to reposition form sections or groups of controls on a form.

Behavior	
Active	Activates the slider control. Select True to activate or False to deactivate. The slider is green when activated and red when deactivated.
Control	
ControlID	Used to identify the control. For example, the first horizontal slider on the form will be assigned the control ID *HorizontalSlider1* and the next horizontal slider is assigned *HorizontalSlider2*. You can change the control ID as needed. The ID can contain numerals or letters, but it must begin with a letter.
Layout	
Position	Controls the relative position (in pixels) of the horizontal slider on the form. The X value determines left to right, the Y value determines top to bottom position. Click the Position property and type values or expand the row to view the X and Y properties and make your changes.
Size	Controls the width and height of the horizontal slider (in pixels). Click the Size property and type values or expand the row to view the Width and Height properties and make your changes.

A hyperlink, when clicked, launches a web browser to a specific URL, or executes a custom event handler.

Appearance	
AutoSize	If set to True, the hyperlink control automatically adjusts its size to match the length of the Text value. If set to False, the hyperlink control does not automatically adjust its size.
BackColor	The background color appearing behind the text. Choose from web, system, or custom colors.
Font	Refers to a type family. Controls the appearance of the text including size, bold, italic, and underline.
ForeColor	The color of the text. Choose from web, system or custom colors.
Text	The text of the hyperlink that displays on the form.
Behavior	
Enabled	Sets the initial state of the hyperlink as enabled (True) or disabled (False). If set to False, the hyperlink is dimmed.
URL	The destination of the hyperlink. For example, http://www.elliemae.com.
Visible	If set to True, the hyperlink is visible on the form. If False, the hyperlink is not visible.

Border	
BorderColor	The color of the border that displays around the hyperlink. Choose from web, system, or custom colors.
BorderStyle	Controls the appearance of the border. The default value is None, meaning no border displays around the hyperlink.
BorderWidth	Controls the width of the border (in pixels).
Control	
ControlID	Used to identify the control. For example, the first hyperlink on the form will be assigned the control ID *Hyperlink1* and the next hyperlink is assigned control ID *Hyperlink2*. You can change the control ID as needed. The ID can contain numerals or letters, but it must begin with a letter.
Layout	
Position	Controls the relative position (in pixels) of the hyperlink on the form. The X value determines left to right, the Y value determines top to bottom position. Click the Position property and type values or expand the row to view the X and Y properties and make your changes.
Size	Controls the width and height of the hyperlink (in pixels). Click the Size property and type values or expand the row to view the Width and Height properties and make your changes.



An image is a graphic that can be displayed on a form.

Appearance	
Source	The file path that maps to the graphic file. For example, *C:\Art\Logos\smalllogo.gif*. Only graphics in the following file formats are permitted: • .bmp • .gif • .jpeg • .png • .tiff
Stretch	Controls the display of the loaded image. When set to False, the image displays at its true dimensions within the image control box. If you expand or shrink the size of the image control box, the image inside does not change. If set to True (default), when you expand or shrink the image control box, the image expands or shrinks also.
Behavior	
Visible	If set to True, the image is visible on the form. If False, the image is not visible.

Layout	
Position	Controls the relative position (in pixels) of the image on the form. The X value determines left to right, the Y value determines top to bottom position. Click the Position property and type values or expand the row to view the X and Y properties and make your changes.
Size	Controls the width and height of the image (in pixels). Click the Size property and type values or expand the row to view the Width and Height properties and make your changes.

An image button is used to execute a predefined action or a custom event handler. To do this, the user clicks an image rather than a button.

Appearance	
HoverText	The text that displays when the user hovers the mouse pointer over the image button.
Source	The file path that maps to the graphic file. For example, *C:\Art\Shapes\redtriangle.gif*. Only graphics in the following file formats are permitted: • .bmp • .gif • .jpeg • .png • .tiff
Behavior	
Action	The action the image button performs when clicked. For example, the *copybrw* action copies data from the Borrower section of the Borrower Summary Form to the Co-Borrower section.
Enabled	Sets the initial state of the image button as enabled (True) or disabled (False).
Visible	If set to True, the image button is visible. If set to False, the image button is not visible.
Control	
ControlID	Used to identify the control. For example, the first image button on the form is assigned the control ID *ImageButton1* and the next image button is assigned *ImageButton2*. You can change the control ID as needed. The ID can contain numerals or letters, but it must begin with a letter.
Layout	
Position	Controls the relative position (in pixels) of the image button on the form. The X value determines left to right, the Y value determines top to bottom position. Click the Position property and type values or expand the row to view the X and Y properties and make your changes.
Size	Controls the width and height of the image button (in pixels). Click the Size property and type values or expand the row to view the Width and Height properties and make your changes.

A label is identifying or descriptive text that is associated with a control, such as a text box or dropdown box.

Appearance	
AutoSize	If set to True, the label control automatically adjusts its size to match the length of the Text. If set to False, the label control does not automatically adjust its size.
BackColor	The background color of the label. Choose from web, system, or custom colors.
Font	Refers to a type family. Controls the appearance of the label text including size, bold, italic, and underline.
ForeColor	The color of the label text. Choose from web, system, or custom colors.
Text	The text of the label that displays on the form.
Behavior	
Enabled	Sets the initial state of the field as enabled (True) or disabled (False). If set to False, the label is dimmed.
Visible	If set to True, the label is visible on the form. If False, the label is not visible.
Border	
BorderColor	The color of the border that displays around the label. Choose from web, system, or custom colors.
BorderStyle	Controls the appearance of the border. The default value is None, meaning no border displays around the label.
BorderWidth	Controls the width of the border (in pixels).
Control	
ControlID	Used to identify the control. For example, the first label on the form will be assigned the control ID *Label1* and the next label is assigned *Label2*. You can change the control ID as needed. The ID can contain numerals or letters, but it must begin with a letter.
Layout	
Position	Controls the relative position (in pixels) of the label on the form. The X value determines left to right, the Y value determines top to bottom position. Click the Position property and type values or expand the row to view the X and Y properties and make your changes.
Size	Controls the width and height of the label (in pixels). Click the Size property and type values or expand the row to view the Width and Height properties and make your changes.

A multi-line text box provides a field for the user to input multi-line text, such as comments. When the text reaches the end of a line it wraps down to the next line.

Appearance	
Alignment	This property determines the positioning of the text in the multi-line text box: • Auto - text is aligned based on the field type of the assigned field ID. • Left - text is left-aligned. • Right - text is right-aligned.
BackColor	The background color that displays inside the multi-line text box. Choose from web, system, or custom colors.
Font	Refers to a type family. Controls the appearance of the text displayed in the multi-line text box including size, bold, italic, and underline.
ForeColor	The color of the text displayed in the multi-line text box. Choose from web, system, or custom colors.
HelpKey	An index into the Help system to provide default hover text for the field. This property is typically the same as the field ID. Controls on standard Encompass forms are all assigned with HelpKey values. Based on this property, appropriate hover text is associated with the control. Custom fields need an original HelpKey.
HoverText	The text that appears when the user hovers the mouse pointer over the field. This text overrides the default text referenced by the HelpKey property.
Behavior	
Enabled	Sets the initial state of the multi-line text box as enabled (True) or disabled (False). If set to False, the multi-line text box is dimmed.
Visible	If set to True, the multi-line text box is visible on the form. If False, the multi-line text box is not visible.
Border	
BorderColor	The color of the border that displays around the multi-line text box. Choose from web, system, or custom colors.
BorderStyle	Controls the style of the border around the multi-line text box. Typically, the Inset style is preferred.
BorderWidth	Controls the width of the border (in pixels).
Control	
ControlID	Used to identify the control. For example, the first multi-line text box on the form will be assigned the control ID *MultilineTextBox1* and the next multi-line text box is assigned *MultilineTextBox2*. You can change the control ID as needed. The ID can contain numerals or letters, but it must begin with a letter.
Data	
Field	The Encompass field ID associated with the multi-line text box field. ***NOTE:*** *Custom field IDs created from this property are assigned to the selected control.*

Layout	
Position	Controls the relative position (in pixels) of the multi-line text box on the form. The X value determines left to right, the Y value determines top to bottom position. Click the Position property and type values or expand the row to view the X and Y properties and make your changes.
Size	Controls the width and height of the multi-line text box (in pixels). Click the Size property and type values or expand the row to view the Width and Height properties and make your changes.
TabIndex	Determines the tab order of the multi-line text boxes on the form.
Rolodex	
Rolodex	The control ID of the Rolodex control associated with the multi-line text box.
RolodexField	The field populated from the Rolodex.

A panel is used to group various controls together. All the controls placed inside the panel are grouped together. If you reposition the panel, all of the controls move with it.
NOTE: *When you open a new workspace, a new panel displays by default.*

Appearance	
BackColor	The background color of the panel. Choose from web, system, or custom colors.
Font	Refers to a type family. Controls the appearance of the text including size, bold, italic, and underline. Every control inside the panel adopts the font property unless otherwise specified.
ForeColor	The color of the text. Choose from web, system or custom colors. Every control inside the panel adopts the color property unless otherwise specified.
Behavior	
Enabled	Sets the initial state of the panel as enabled (True) or disabled (False). If set to False, all of the controls within the panel are disabled.
Visible	If set to True, the panel is visible on the form. If False, the panel is not visible. If set to False, none of the controls within the panel are visible.
Border	
BorderColor	The color of the border that displays around the panel. Choose from web, system, or custom colors.
BorderStyle	Controls the appearance of the border. The default value is None, meaning no border displays around the panel.
BorderWidth	Controls the width of the border (in pixels).

Control	
ControlID	Used to identify the control. For example, the first panel on the form will be assigned the control ID *Panel1* and the next panel is assigned *Panel2*. You can change the control ID as needed. The ID can contain numerals or letters, but it must begin with a letter. ***NOTE:*** *The control ID of the default panel that displays when you open a new workspace is pnlForm.*
Layout	
Layout	Determines the location of the control on the form, relative to other controls (Flow) or in an absolute X and Y location (Positioned). For example, if the Layout of two controls is set to Flow, the first control is placed at the top-left of the form and the second is placed directly to the right or below it. If you increase or decrease the size of the control, the second control automatically moves to maintain its position next to the first. (Flow is similar to traditional HTML page layouts.)
Position	Controls the relative position (in pixels) of the panel on the form. The X value determines left to right, the Y value determines top to bottom position. Click the Position property and type values or expand the row to view the X and Y properties and make your changes.
Size	Controls the width and height of the panel (in pixels). Click the Size property and type values or expand the row to view the Width and Height properties and make your changes.

The Pick List control is used to create a list of dropdown options. To use the control on your form you must add code to the *ItemSelected* event handler.

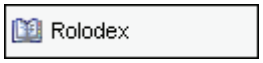
Appearance	
Title	The title that displays above the list of options.
Behavior	
Enabled	Sets the initial state of the pick list as enabled (True) or disabled (False). If set to False the pick list is dimmed.
Visible	If set to True, the pick list is visible. If False, the pick list is not visible.
Control	
ControlID	Used to identify the control. For example, the first pick list on the form will be assigned the control ID *PickList1* and the next pick list is assigned *PickList2*. You can change the control ID as needed. The ID can contain numerals or letters, but it must begin with a letter.
Data	
Options	The list of options that displays when the user clicks the pick list control. Click  to view and edit the list.

Layout	
Position	Controls the relative position (in pixels) of the pick list on the form. The X value determines left to right, the Y value determines top to bottom position. Click the Position property and type values or expand the row to view the X and Y properties and make your changes.
Size	Controls the width and height of the pick list (in pixels). Click the Size property and type values or expand the row to view the Width and Height properties and make your changes.

A radio button allows the user to select an option by clicking a button. Radio buttons are grouped in a set and only one button can be selected.

Appearance	
BackColor	The background color that displays behind the radio button label. Choose from web, system, or custom colors.
Font	Refers to a type family. Controls the appearance of the text that displays next to the radio button including size, bold, italic, and underline.
ForeColor	The color of the text that displays next to the radio button. Choose from web, system, or custom colors.
HelpKey	An index into the Help system to provide default hover text for the field. This property is typically the same as the field ID. Controls on standard Encompass forms are all assigned with HelpKey values. Based on this property, appropriate hover text is associated with the control. Custom fields need an original HelpKey.
HoverText	Text that displays when the user hovers the mouse pointer over the field. This text overrides the default text referenced by the HelpKey property.
Text	The text that displays next to the radio button on the form.
Behavior	
Enabled	Sets the initial state of the field as enabled (True) or disabled (False). If set to False, the radio button is dimmed.
Visible	If set to True, the radio button is visible. If set to False, the radio button is not visible.
Border	
BorderColor	The color of the border appearing around the radio button and its label. Choose from web, system, or custom colors.
BorderStyle	Controls the style of the border. The default property is None meaning no border appears around the radio button.
BorderWidth	Controls the width of the border (in pixels).
Control	
ControlID	Used to identify the control. For example, the first radio button on the form will be assigned the control ID *RadioButton1* and the next radio button is assigned *RadioButton2*. You can change the control ID as needed. The ID can contain numerals or letters, but it must begin with a letter.

Data	
CheckedValue	The value stored in the loan field when the radio button is selected.
Field	The Encompass field ID associated with the radio button field. ***NOTE:*** *Custom field IDs created from this property are assigned to the selected control.*
GroupName	A name assigned to each member of a group of radio buttons. Only one radio button in the group can be selected.
Layout	
Position	Controls the relative position (in pixels) of the radio button on the form. The X value determines left to right, the Y value determines top to bottom position. Click the Position property and type values or expand the row to view the X and Y properties and make your changes.
Size	Controls the width and height of the radio button (in pixels). Click the Size property and type values or expand the row to view the Width and Height properties and make your changes.
TabIndex	Determines the tab order of the radio buttons on the form.



A Rolodex control launches the Address Book pop-up window and populates one or more Field controls based on the user's selection.

Behavior	
Enabled	Sets the initial state of the control as enabled (True) or disabled (False).
Visible	If set to True, the Rolodex control is visible. If set to False, the Rolodex control is not visible.
Control	
ControlID	Used to identify the control. For example, the first Rolodex control on the form will be assigned the control ID *Rolodex1* and the next Rolodex control is assigned *Rolodex2*. You can change the control ID as needed. The ID can contain numerals or letters, but it must begin with a letter.
Data	
BusinessCategory	The business category that will display when the Address Book is opened. For example, credit or underwriter.
Layout	
Position	Controls the relative position (in pixels) of the Rolodex on the form. The X value determines left to right, the Y value determines top to bottom position. Click the Position property and type values or expand the row to view the X and Y properties and make your changes.
Size	Controls the width and height of the Rolodex (in pixels). Click the Size property and type values or expand the row to view the Width and Height properties and make your changes.

Text boxes allow users to enter and edit single-line Encompass data. The text entered can be both numeric and non-numeric.

Appearance	
Alignment	This property determines the positioning of the text in the text box: • Auto - text is aligned based on the field type of the assigned field ID. • Left - text is left-aligned. • Right - text is right-aligned.
BackColor	The background color that displays inside the text box. Choose from web, system, or custom colors.
Font	Refers to a type family. Controls the appearance of the text typed in the text box including size, bold, italic, and underline.
ForeColor	The color of the text displayed in the text box. Choose from web, system, or custom colors.
HelpKey	An index into the Help system to provide default hover text for the field. This property is typically the same as the field ID. Controls on standard Encompass forms are all assigned HelpKey values. Based on this property, appropriate hover text is associated with the control.
HoverText	The text that appears when the user hovers the mouse pointer over the field. This text overrides the default text referenced by the HelpKey property.
Behavior	
Enabled	Sets the initial state of the text box as enabled (True) or disabled (False). If set to False, the text box is dimmed.
MaxLength	The maximum length of the text that can be typed in the field.
Visible	If set to True, the text box is visible on the form. If False, the text box is not visible.
Border	
BorderColor	The color of the border that displays around the text box. Choose from web, system, or custom colors.
BorderStyle	Controls the style of the border. Typically, the Inset style is preferred.
BorderWidth	Controls the width of the border (in pixels).
Control	
ControlID	Used to identify the control. For example, the first text box on the form will be assigned the control ID *TextBox1* and the next text box is assigned *TextBox2*. You can change the control ID as needed. The ID can contain numerals or letters, but it must begin with a letter.
Data	
Field	The Encompass field ID associated with the text box field. ***NOTE:*** *Custom field IDs created from this property are assigned to the selected control.*

Layout	
Position	Controls the relative position (in pixels) of the text box on the form. The X value determines left to right, the Y value determines top to bottom position. Click the Position property and type values or expand the row to view the X and Y properties and make your changes.
Size	Controls the width and height of the text box (in pixels). Click the Size property and type values or expand the row to view the Width and Height properties and make your changes.
TabIndex	Determines the tab order of the text boxes on the form.
Rolodex	
Rolodex	The control ID of the Rolodex control associated with the text box.
RolodexField	The field populated from the Rolodex.

A button is used to execute a predefined action or a custom event handler. Standard buttons have the same behavior as buttons, except that you can select from a set of standard, predefined icons to use for the buttons.

Appearance	
Button Type	Determines the predefined icon used to for the button: Edit, Lookup, Clear, Refresh, Help, or Calendar.
HoverText	The text that displays when the user hovers the mouse pointer over the button.
Behavior	
Action	The action the button performs when clicked. For example, the copybrw action copies data from the Borrower section of the Borrower Summary Form to the Co-Borrower section.
Enabled	Sets the initial state of the button as enabled (True) or disabled (False). If set to False, the button is dimmed.
Visible	If set to True, the button is visible. If set to False, the button is not visible.
Control	
ControlID	Used to identify the control. For example, the first button on the form will be assigned the control ID StandardButton1 while the next button is assigned StandardButton2. You can change the control ID as needed. The ID can contain numerals or letters, but it must begin with a letter.
Layout	
Position	Controls the relative position (in pixels) of the button on the form. The X value determines left to right, the Y value determines top to bottom position. Click the Position property and type values or expand the row to view the X and Y properties and make your changes.
Size	Controls the width and height of the button (in pixels). Click the Size property and type values or expand the row to view the Width and Height properties and make your changes.

A vertical rule control displays a vertical divider and is used to accent the vertical border of a category box or other section of a form. It can also be used as a separator between sections. By default, it is designed to match the look and feel of standard Encompass forms.

Behavior	
Visible	If set to True, the vertical rule is visible. If set to False, the vertical rule is not visible.
Control	
ControlID	Used to identify the control. For example, the first vertical rule on the form will be assigned the control ID *VerticalRule1* and the next vertical rule is assigned *VerticalRule2*. You can change the control ID as needed. The ID can contain numerals or letters, but it must begin with a letter.
Layout	
Position	Controls the relative position (in pixels) of the vertical rule on the form. The X value determines left to right, the Y value determines top to bottom position. Click the Position property and type values or expand the row to view the X and Y properties and make your changes.
Size	Controls the width and height of the vertical rule (in pixels). Click the Size property and type values or expand the row to view the Width and Height properties and make your changes.

A vertical slider is a Design control that moves controls left or right within the form. Any control that intersects with the slider moves as the slider moves. You can use the slider to reposition form sections or groups of controls on a form.

Behavior	
Active	Activates the slider control. Select True to activate or False to deactivate. The slider is green when activated and red when deactivated.
Control	
ControlID	Used to identify the control. For example, the first vertical slider on the form will be assigned the control ID *VerticalSlider1* and the next vertical slider is assigned *VerticalSlider2*. You can change the control ID as needed. The ID can contain numerals or letters, but it must begin with a letter.
Layout	
Position	Controls the relative position (in pixels) of the vertical slider on the form. The X value determines left to right, the Y value determines top to bottom position. Click the Position property and type values or expand the row to view the X and Y properties and make your changes.
Size	Controls the width and height of the vertical slider (in pixels). Click the Size property and type values or expand the row to view the Width and Height properties and make your changes.

A zip code lookup is a Design control that performs a lookup on a zip code value and populates the city, county, and state loan fields accordingly.

Control	
ControlID	Used to identify the control. For example, the first zip code lookup control on the form will be assigned the control ID *ZipCodeLookup1* and the next zip code lookup control is assigned *ZipCodeLookup2*. You can change the control ID as needed. The ID can contain numerals or letters, but it must begin with a letter.
Data	
CityField	The field ID of the field to be populated with the city corresponding to the zip code.
CountyField	The field ID of the field to be populated with the county corresponding to the zip code.
StateField	The field ID of the field to be populated with the state corresponding to the zip code.
Layout	
Position	Controls the relative position (in pixels) of the zip code lookup control on the form. The X value determines left to right, the Y value determines top to bottom position. Click the Position property and type values or expand the row to view the X and Y properties and make your changes.
Size	Controls the width and height of the zip code lookup control (in pixels). Click the Size property and type values or expand the row to view the Width and Height properties and make your changes.
Source	
ZipControl	The control that will contain the zip code value to be resolved.

Appendix B

Button Actions

The following is a list of actions that can be assigned to a button control. Type an action in the button's **Action** property to enable the button to carry out the predefined behavior.

Action	Behavior
1stmor	Brings up the monthly payment calculator
addregistration	Opens the Investor Registration dialog
altlender	Brings up the Ellie Mae Network Alternate Lender list for closing
baseincome	Brings up the borrower base income worksheet
ccprog	Show the Closing Cost selection dialog
cityfee	Allows the selection of a city fee from the defined fee tables
cityfeeca	Allows the selection of a city fee from the defined fee tables, but populates the MLDS instead of the GFE
condlist	Brings up the Document Tracking worksheet
copyaddr	Copies the subject property address information into the borrower's mailing address fields
cashflow	Displays the Pre-qual Cash Flow dialog
detailrequest	Opens the detailed Lock Request form
editcheck	Launches the Ellie Mae Network PCI Edit Check interface
fhamaxloan	Performs the FMA loan limit calculation and updates field 1720
geocode	Launches the Ellie Mae Network Geocoding service
hazins	Displays the hazard insurance calculation dialog
hudsetup	Displays the HUD-1 Initial Escrow Account Setup dialog

Action	Behavior
hud1a	Displays the HUD-1A Disbursement dialog
importliab	Imports liabilities from a previously retrieved credit report
income	Displays the Pre-qual Monthly Income dialog
investor	Opens the Investor selection window
loanprog	Displays the Loan Program selection dialog
loanprog2	Opens the Loan Program selection dialog for the Piggyback/Linked loan
manageborrowers	Displays the Co-Mortgagers dialog
mersmin	Generates a new MERS/MIN number for the loan
mipff	Displays the MIP/PMI Calculation dialog
mortg	Displays the Mortgage Insurance payment dialog
mpiffs	Displays the MIP/PMI Calculation dialog for the Piggyback/Linked loan
mtginsprem	Displays the Mortgage Insurance premium dialog
mtginspremca	Displays the MLDS Mortgage Insurance premium dialog
mtginsreserv	Displays the Mortgage Insurance reserves dialog
orderappraisal	Launches the Ellie Mae Network Appraisal services window
ordercredit	Launches the Ellie Mae Network Credit services window
orderflood	Launches the Ellie Mae Network Flood Insurance services window
ordertitle	Launches the Ellie Mae Network Title Insurance services window
other	Displays the Other Housing Expense dialog
otherf	Displays the Other Financing dialog
plancode	Displays the Closing Documents Plan Code window
presenthe	Displays the Present Housing Expense dialog

Action	Behavior
productandpricing	Displays the Ellie Mae Network Product & Pricing services window
ratespread	Displays the PCI Rate Spread window
resetrequestform	Re-populates the lock request form with data from the loan
retaxes	Displays the Real Estate tax dialog
selecttemplate	Displays the Purchase Advice Template selection dialog
statefee	Displays the State Fee selection dialog
statefeeca	Displays the State Feed selection dialog for the MLDS
subfin	Displays the Subordinate Financing dialog
subfin2	Displays the Subordinate Financing dialog for the Piggyback/Linked loan
syncbrw	Displays the Borrower Contact synchronization dialog
taxes	Displays the Tax computation dialog
taxesreserv	Displays the Taxes Reserved computation dialog
transferto	Opens the Closing Document Transfer service window
trusteedb	Displays the Trustee selection dialog
userfee1	Displays the User Fee 1 computation dialog
userfee1ca	Displays the User Fee 1 computation dialog for the MLDS
userfee2	Displays the User Fee 2 computation dialog
userfee2ca	Displays the User Fee 2 computation dialog for the MLDS
userfee3	Displays the User Fee 3 computation dialog
userfee3ca	Displays the User Fee 1 computation dialog for the MLDS
viewcredit	Displays the Credit Report information for the loan
vor	Displays the VOR list
voe	Displays the VOE list

Action	Behavior
vol	Displays the VOL list
vod	Displays the VOD list
vom	Displays the VOM list
zoomarm	Displays the ARM Type Details dialog

Appendix C

Loan Custom Field Calculations

The Loan Custom Fields tool provides the ability to create calculations for both predefined and user-defined custom field IDs. A custom calculation is an expression that returns a number or text value, which is then saved into the associated custom field. This document describes the types of operations you can use in a custom field calculation, and provides examples of how to use each one.

NOTE: *Loan custom field calculations can also be used in advanced coding for business rules. See the "Advanced Coding for Business Rules" document.*

To Define a Custom Field with a Calculation:

1 On the menu bar, click **Encompass**, and then click **Settings**.

2 On the left panel, click **Loan Setup**, and then click **Loan Custom Fields**.

3 Double-click to open a pre-defined custom Field (such as CUST01FV).

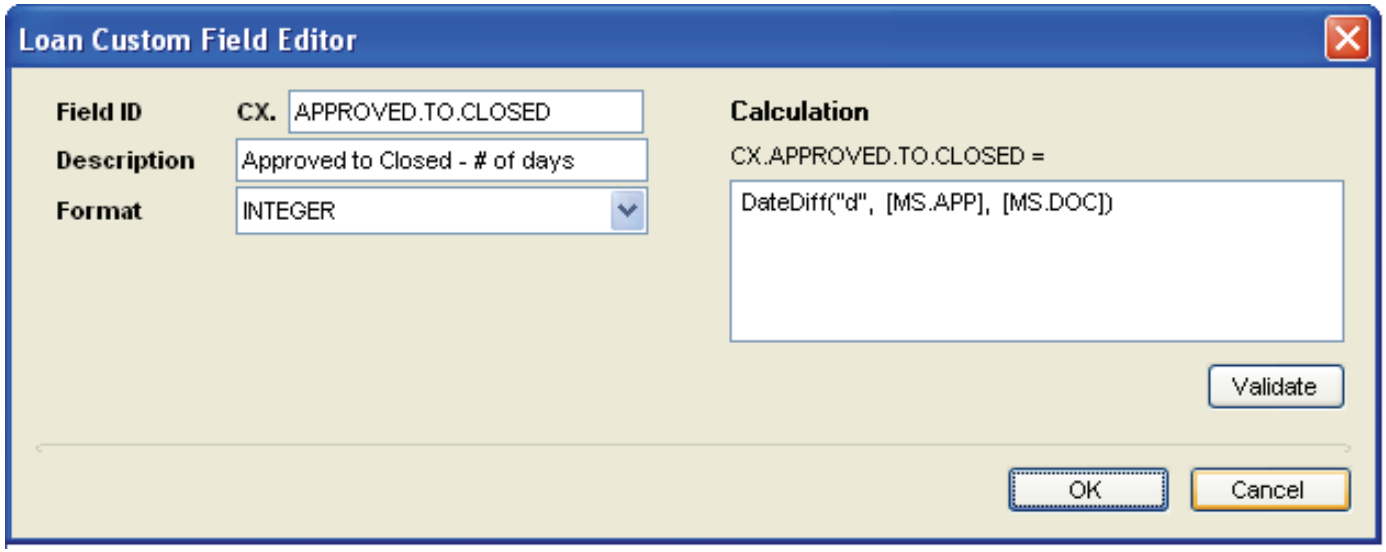
- To create a user-defined custom field ID, click the **New** icon and type the Field ID.

4 Enter a Description, and then select a Format for the field. Refer to the online help for more information regarding the field formats.

5 Enter the calculation.

6 To validate the calculation syntax, click the **Validate** button and then click **OK** to the message.

7 When finished, click **OK**.



Using Loan Field Values

Most custom calculations need to use the value of loan fields in order to calculate the value for the associated custom field. You can insert a reference to any loan field which has a defined field ID by placing the field's ID within square-brackets ([]). For example, [136] references the purchase price. Any of the more than 7000 standard fields shipped as part of Encompass, or the loan custom fields you have already defined, can be referenced in this manner.

By default, when referencing a field, the type of value returned (numeric or text) is based on the type of the underlying field. For example:

- [1109] (loan amount) returns a numeric value
- [4000] (borrower first name) returns a string value

There is one exception to this rule: when a numeric field has not yet been populated, a reference to that field will return an empty string. As a result, you must use care when writing calculations to consider what will happen when the fields used in the
calculation are empty (refer to “Calculation Errors” on page 79).

Operations and Features

A custom calculation can contain any combination of the following operations and features:

- Simple arithmetic operations, such as addition and subtraction
- Mathematical operations, such as exponentiation and absolute value
- Text-based operations, such as concatenation, truncation, and extraction of characters
- Date-based operations, such as adding a fixed number of days or months to a date
- Values of other fields from the same loan
- Branching and logical operations, such as if...then, AND, and OR

Arithmetic Operations

Arithmetic operations form the basis for most custom calculations and consist of addition (+), subtraction (-), multiplication (*), and division (/).

For example, the following custom calculation returns 2% of the value stored in field ID 1109 (loan amount).

```
0.02 * [1109]
```

Encompass custom calculations use standard arithmetic order-of-operations rules and allow for use of parentheses to force a specific execution order.

```
0.02 * (([1109] + [44])
```

NOTE: *Refer to “Using Loan Field Values” on page 71 for details on using the square bracket syntax.*

An important consideration when authoring custom calculations is whether or not the specified arithmetic operations can be performed based on the values in the fields. In the example above, the calculation assumes that both fields 1109 and 44 contain valid numeric data. If either field is blank or contains non-numeric data, the calculation will fail and the custom field's value will be cleared (refer to “Calculation Errors” on page 79).

Safe Operations

When you are unsure of the potential validity of an arithmetic expression within a calculation, you can use one of the following built-in functions in place of the standard operator.

Function	Description	Example
Sum(x, y, z, ...)	Adds the specified values. If any operand cannot be converted to a number, the entire expression returns an empty value.	Sum([1109], [44])
SumAny(x, y, z, ...)	Adds the specified values, ignoring those that cannot be converted to numbers. If none of the values can be converted to a number, the function returns a blank value.	SumAny([1109], [44])
Diff(x, y)	Evaluates to x – y if both values are numeric. If either value is non-numeric, the function returns a blank value.	Diff([1109], [44])
Mult(x, y, z, ...)	Returns the product of the operands. If any of the operands cannot be converted to a number, the function returns a blank value.	Mult([1109], [44])
MultAny(x, y, z, ...)	Returns the product of all operands which can be converted to numbers. If none of the values can be converted to a number, returns a blank value.	MultAny([1109], [44])
Div(x,y)	Returns the value of x / y if both operands can be converted to numeric values. Otherwise, returns a blank value.	Div([1109], [44])

Using these operators, it is possible to write a “safe” version.

Example:

```
0.02 * ([1109] + [44])
```

Rewritten using the safe operators:

```
Mult(0.02, Sum([1109], [44]))
```

If either field 1109 or field 44 is empty or non-numeric, this expression will evaluate to an empty value. Expressions of this form are most commonly needed when using the branching function IIF. Refer to “Branching and Logic Operations” on page 76.

Mathematical Operations

Within custom calculations you can employ any of a number of pre-defined mathematical functions to calculate the resulting value. The following table lists the supported functions.

Function	Description	Example
Abs(x)	Returns the absolute value of the specified argument.	Abs([101] - [102])
Min(x, y, z, ...)	Returns the smallest of a set of values.	Min([102], [103], [104])
Max(x, y, z, ...)	Returns the largest of a set of values.	Max([102], [103], [104])
LMedian(x, y, z, …)	Returns the median value of a set of numbers. If an even number of values is specified, the lower of the two middle values will be returned.	LMedian([#67], [#1450], [#1414])
UMedian(x, y, z, …)	Returns the median value of a set of numbers. If an even number of values is specified, the higher of the two middle values will be returned.	UMedian([#67], [#1450], [#1414])
Sqrt(x)	Returns the square root of a value.	Sqrt(0.02 * [1109])
Log(x)	Returns the natural logarithm (base-*e*) of the value.	Log(1000 + [910])
Log10(x)	Returns the base-10 logarithm of the value.	Log10(1000 + [910])
Exp(x)	Returns the value of e^x.	Exp([1171] / 100)
Pow(x, y)	Returns the value of x^y.	Pow([1171], 5)
Sgn(x)	Returns 1 if x > 0, 0 if x = 0, or -1 if x < 0.	Sgn([1093])
Round(x, precision)	Rounds the value x to a number of decimal places specified by *precision*.	Round([1109], 2)
Trunc(x, precision)	Truncates the value x to a number of decimal places specified by *precision*.	Trunc([1109], 2)
XInt(x, *default*)	Converts the value x to an integer. If x is a string, the integer value it represents is returned. Non-integral values are rounded to the nearest integer. If the value cannot be converted successfully, the optional *default* value is returned. If no default is provided, the value 0 is returned.	XInt(“230”) or XInt([1109], 1)
XDec(x, *default*)	Converts the value x to a decimal. If x is a string, the numeric value it represents is returned. If the value cannot be converted successfully, the optional *default* value is returned. If no default is provided, the value 0 is returned.	XDec(“245.112”) or XDec([1109], -1)

Text-based Operations

Depending on the type of field with which the calculation is associated, your calculation may need to return a text-based value instead of a numeric value. For example, if your field type is defined to be “Y/N”, then your custom calculation should evaluate to either "Y" or "N", since these are the only two valid values for this field type.

The most basic string operation that can be performed is concatenation, which is done by using the ampersand (&) and operator. For example:

Builds the borrower’s full name from their first and last names with a space separating them.

```
[4000] & " " & [4002]
```

Returns the value of field ID 1109 (loan amount) preceded by the dollar sign and followed by a space and USD, such as $200,000.00 USD.

```
"$" & [+1109] & " USD"
```

Note that literal strings, such as " " (space), "$", and the " USD" are denoted by enclosing the text in double quotes. If the need arises to include the double-quote character itself in a string literal, it should occur twice.

For example, assuming that field 4000 has the value "Joe", the expression:

```
[4000] & " is the ""primary"" borrower for the loan."
```

would evaluate to:

```
Joe is the "primary" borrower for the loan.
```

In addition to concatenation, Encompass provides the following string-based functions to assist you in authoring custom calculations.

Function	Description	Example
Trim(x)	Removes any white space characters from the beginning and end of the value x.	Trim([4000] & " " & [4002])
Left(x, n)	Returns the left-most n characters from the value x. If x is shorter than n characters, the whole value is returned.	Left([4002], 5)
Right(x, n)	Returns the right-most n characters from the value x. If x is shorter than n characters, the whole value is returned.	Right([4002], 5)
Mid(x, start, *length*)	Returns the substring of the value x that starts at the specified locations and has the specified length. If *length* is omitted, the entire string after *start* is returned. The start position is 0-based.	Mid([4002], 3, 2)
InStr(x, y)	Returns the first position of the substring y within the string x. The comparison is case-sensitive.	InStr([4002], "mith")
Int2Text(x)	Converts an integer value to its spelled out representation, e.g. “Five Hundred Thirty-Four.”	Int2Text([4])
Dec2Text(x)	Converts a decimal value to its spelled out representation, e.g. “Sixty-Five and Fifty-Three Hundredths.”	Dec2Text([3])
Money2Text(x)	Converts a decimal value to its spelled out representation using ‘dollars’ and ‘cents’ notation, e.g. “Sixty-Five Dollars and Fifty-ThreeCents.”	Money2Text([1109])
LCase(x)	Returns the value of x with all letters converted to lower case.	LCase([4002])

Function	Description	Example
UCase(x)	Returns the value of x with all letters converted to lower case.	UCase([4002])
Replace(x, y, z)	Replaces all instances of the substring y within the string x with the replacement value z, and returns the resulting string.	Replace([1264], "Investor", "Lender")

Date-Based Operations

Just as a custom calculation can yield a numeric or text result, it can also yield a date result when the underlying field type is of type DATE. Note that fields that have the MONTHDAY format do not contain full date values and cannot be used with the functions below.

Function	Description	Example
Day(x)	Returns the day portion of the date value x. The parameter x must be a valid date or the calculation will fail.	Day([1402])
Month(x)	Returns the month portion of the date value x. The parameter x must be a valid date or the calculation will fail.	Month([1402])
Year(x)	Returns the year portion the of date value x. The parameter x must be a valid date or the calculation will fail.	Year([1402])
DateAdd(period, count, x)	Add a fixed number of days, months, or years to the date value x. The period should be one of "d" (days), "m" (months), or "yyyy" (years). The count is the number of days/months/years to add. If any parameter is invalid, the entire calculation will result in a blank value.	DateAdd("yyyy", 1, [1402])
XDateAdd(period, count, x)	Add a fixed number of days, months, or years to the date value x. The period should be one of "d" (days), "m" (months), or "yyyy" (years). The count is the number of days/months/years to add. If any parameter is invalid, an empty value is returned from the function and the calculation will proceed.	XDateAdd("yyyy", 1, [1402])
DateDiff(period, x, y)	Computes the difference between the dates x and y in days, months or years. The period should be one of "d" (days), "m" (months) or "yyyy" (years). If any parameter is invalid, the entire calculation will result in a blank value.	DateDiff("d", [1402], [1403])
XDateDiff(period, x, y)	Computes the difference between the dates x and y in days, months or years. The period should be one of "d" (days), "m" (months) or "yyyy" (years). If any parameter is invalid, an empty value is returned from the function and the calculation will proceed.	XDateDiff("d", [1402], [1403])
XDate(x, *default*)	Converts the value x to a date. If x is a string, the date value it represents is returned. If the value cannot be converted successfully, the optional *default* value is returned. If no default is provided, the date 1/1/1 is returned.	XDate([1402], "11/30/2010")

Function	Description	Example
XMonthDay(x, *default*)	Converts the value x to a “month-day” value. A month-day is represented as a date within the year 2000. For example, XMonthDay(“3/15”) would return the date 3/15/2000. If the value cannot be converted to a month-day value, the optional default is provided.	XMonthDay(“3/15”)
Today	Returns today’s date.	DateAdd("d", 7, Today)

Calendar-Based Operations

If your custom field calculation requires access to your company’s compliance calendar, the functions below provide the necessary operations to add or adjust dates based on either the Postal or Business calendar.

Function	Description	Example
Calendar.AddBusinessDays (date, count, moveToNext)	Adds the specified number of days from your company’s Business Calendar to the date provided. The moveToNext parameter is a Boolean which indicates if the date should first be advanced to the next business day if the date specified is not a business day.	Calendar.AddBusinessDays ([763], 5, true)
Calendar.AddPostalDays (date, count, moveToNext)	Adds the specified number of days from the US Postal Calendar to the date provided. The moveToNext parameter is a Boolean which indicates if the date should first be advanced to the next business day if the date specified is not a business day.	Calendar.AddPostalDays ([763], 5, true)

Branching and Logic Operations

Many custom calculations require complex branching (if…then) logic in order to arrive at the desired value. To accommodate this need, the custom calculations provide the IIF() function, which can be used to express if…then…else logic. The basic syntax of this function is as follows:

```
IIF(Boolean expression, True value, False value)
```

The first argument to IIF() is an expression that evaluates to either TRUE or FALSE. If that expression evaluates as TRUE, then the IIF function returns the value specified in the second parameter (the "True value"). Otherwise, the IIF function returns the third parameter, the "False value."

For example, the following custom calculation returns 2% of the loan amount for loan amounts above $100,000 and 5% of the loan amount for loans below $100,000:

```
IIF([1109] > 100000, 0.02 * [1109], 0.05 * [1109])
```

The Boolean expression of an IIF statement can use the AND and OR operations to perform more complex logic, for example:

```
IIF([1109] > 100000 AND [1335] < 20000, 0.02 * [1109], 0.05 * [1109])
```

Often, you may encounter the need to handle more than just two cases (a TRUE case and a FALSE case). In these scenarios, you can use nested IIF statements. For example, the following code demonstrates performing a calculation based on the purpose of the loan.

```
IIF([1811] = "PrimaryResidence", 0.05 * [1109],
     IIF([1811] = "SecondHome", 0.02 * [1109],
     IIF([1811] = "Investor", 0.01 * [1109], 0)))
```

The expression above evaluates to 5% of the loan amount if the loan is for the borrower's primary residence, 2% if it's a second home, 1% for an investment property and return the value "0" if the loan purpose has not yet been specified.

IMPORTANT: *When the IIF function is invoked, the calculation engine first evaluates both the second and third parameters of the function before evaluating the Boolean expression. Therefore, both parameters need to evaluate to valid values even though only one value will be used. For example, the following IIF statement will fail.*

```
IIF(5 > 0, 1, 2 * [4000])
```

The expression 2 * [4000] is typically invalid since field 4000 contains the borrower's first name (which is typically not numeric). Even though the logical expression 5 > 0 will always evaluate to TRUE (and thus the value returned by IIF will always be 1), the failure of the third parameter to evaluate successfully will cause the entire calculation to fail.

To work around this issue, you can use any of the following techniques:

- Use the safe arithmetic operations, e.g. IIF(5 > 0, 1, Mult(2, [4000])). These operations ensure that no errors will occur.
- Use the field modifiers, e.g. IIF(5 > 0, 1, 2 * [#4000]). The numeric conversion modifier will convert non-numeric values to 0, allowing the calculation to be carried out successfully (refer to “Calculation Errors” on page 79).
- Use a combination of two custom fields, for example:

  ```
  CX.FIELD1 = 2 * [4000]
  CX.FIELD2 = IIF(5 > 0, 1, [CX.FIELD1])
  ```

  When field 4000 is non-numeric, the calculation for field CX.Field1 will be invalid and, as a result, the field will be blank. However, the expression for CX.Field2 can now be evaluated successfully since the substitution of the value CX.FIELD1 will always work (even if the value happens to be blank).

As a second example, the expression above that branches based on the loan purpose would be rewritten as follows to ensure safe evaluation regardless of the value in field 1109:

```
IIF([1811] = "PrimaryResidence", 0.05 * [#1109],
     IIF([1811] = "SecondHome", 0.02 * [#1109],
     IIF([1811] = "Investor", 0.01 * [#1109], 0)))
```

In addition, the custom calculation engine provides a few functions which can be used to determine the state of a field so you can add condition logic that behaves appropriately.

Function	Description	Example
IIF(x, truepart, falsepart)	Returns the truepart if x is true, the falsepart otherwise.	IIF([#1109] > 100000, 10, 20)
IsEmpty(x)	Returns a boolean indicating if the value x is the empty string.	IsEmpty([1109])
IfEmpty(x, val)	Returns the value *x* if it is non-empty, otherwise returns the value *val*.	IfEmpty([1109], 0)
IsNumeric(x)	Returns a boolean indicating if the value x can be converted into a numeric value.	IsNumeric([1109])
IsDate(x)	Returns a boolean indicating if the value x can be converted to a valid date.	IsDate([1402])

List-Based Operations

Evaluating a custom calculation will frequently require logic which involves looking up a value in list of possible values or within a range of values. These operations could be carried out using one or more nested IIF() expressions as demonstrated above, but the calculation engine offers several functions to simplify this task.

Function	Description	Example
Match(x, value0, value1, …)	Returns the index of the first value in the list that matches x. If not match is found, the value -1 is returned.	Match([608], "Fixed", "GraduatedPaymentMortgage", "AdjustableRate", "OtherAmortizationType")
Range(x, value0, value1, …)	Returns the index of the first value which is greater than x.	Range([1109], 100000, 200000, 500000)
RangeLow(x, value0, value1, …)	Returns the index of the first value which is greater than or equal to x.	RangeLow([1109], 100000, 200000, 500000)
Pick(x, value0, value1, …)	Returns the value whose index is x.	Pick([#16] - 1, "1 unit", "2 units", "3 units", "4 units")
Count(value0, value1, …)	Returns the number of parameters that are non-empty.	Count([4000], [4002], [98], [99])

Using these functions, this calculation:

```
IIF([1811] = "PrimaryResidence", 0.05 * [#1109],
     IIF([1811] = "SecondHome", 0.02 * [#1109],
     IIF([1811] = "Investor", 0.01 * [#1109], 0)))
```

could be rewritten as:

```
Pick(Match([1811], "PrimaryResidence", "SecondHome", "Investor", ""),
     0.05 * [#1109], 0.02 * [#1109], 0.01 * [#1109], 0)
```

Advanced Functions

The Encompass Custom Calculation Engine leverages the Visual Basic.NET programming language when evaluating your calculation. As a result, you may make use of any function which is provided as part of the VB.NET programming language. For a complete reference of these functions, visit Microsoft's VB.NET Reference.

Keep in mind that your calculation should not invoke any method which can trigger the display of a user interface of any kind as this can cause Encompass or the Encompass Server to fail.

Calculation Errors

There are two types of errors that can occur when authoring custom field calculations:

- Syntax errors, which are detected when you validate or save the custom calculation
- Runtime errors, which occur during the evaluation of the calculation which Encompass is running

When a runtime error occurs during the evaluation of a custom field calculation, Encompass will clear the field. For example, consider the field calculation:

```
[1109] * 2
```

If field 1109 is blank, this calculation will fail and the custom field will also be blank. Referencing an unpopulated numeric field, returns an empty string. When field 1109 takes on a numeric value, the calculation will succeed and the custom field's value will be populated appropriately.

In general, if an error occurs in any part of the calculation, the entire calculation will fail. To avoid this, use the build-in functions (Sum(), Diff(), etc.) or use the field modifiers described below. Either technique can be used to avoid errors that might otherwise short circuit your calculation.

Consider the following expression:

```
[101] + [102] + [103] + [104] + [105]
```

If all five referenced fields contain numeric data, this expression will evaluate to the sum of these values and the result saved into the custom field. However, if any one or more of these fields is unpopulated, the expression will fail since a string is not valid within a summation. As a result, the custom field's value will be cleared.

There are two possible resolutions to this issue:

1 Use the "safe" arithmetic operators. The expression above could be rewritten as:

```
SumAny([101], [102], [103], [104], [105])
```

By definition, the SumAny() function ignores any values which are non-numeric and returns the sum of those that are. If none of the values are numeric, the SumAny() function returns an empty string.

2 Use the numeric field modifier. Field modifiers allow you to provide additional instructions on the format in which the field should be inserted into the code. Modifiers are inserted within the square brackets but before the field ID, such as [#101]. The following modifiers are available.

Modifier	Description	Example
#	Forces a numeric value to be returned. If the field cannot be converted to a number, the value 0 is returned.	[#1109]
-	Forces the field value to be returned as an unformatted string. For example, a typical numeric value might return "200000.00".	[-1109]
+	Forces the field value to be returned as a formatted string. For example, a typical numeric value might return "200,000.00".	[+1109]
@	Forces the field value to be returned as a date. If the field cannot be converted to a date, the date 1/1/1 is returned.	[@1109]

Using the “#”modifier, the expression above could be rewritten as:

```
[#101] + [#102] + [#103] + [#104] + [#105]
```

Note that unlike the SumAny() function that will return a blank value if all of the referenced fields are empty, this expression would evaluate to 0 in that scenario (since each operand will evaluate to 0).

When you place a reference to another loan field within a custom field calculation, that field will automatically be updated whenever the value of the referenced field is modified. As a result, you must avoid circular dependencies in your custom field calculations. For example, say you define the following custom field calculations:

```
CX.FIELD1 = [1109] + [CX.FIELD2]
CX.FIELD2 = [CX.FIELD3] * 2
CX.FIELD3 = [CX.FIELD1] + 20000
```

Individually, each calculation is valid, but a circular dependency exists between the three fields. The Custom Field Editor will notify you of circular dependencies when they exist.

Index

G

H

I

L

M

O

P

R

S

T

U

V

W

Z