

# **FORMENTRY FOR MAC 2.4.1**

## **The User's Manual**

Last Updated: August 15, 2015



## **Widget Press, Inc.**

Copyright © 2015 Widget Press, Inc. All rights reserved. Widget Press and the Widget Press logo are trademarks of Widget Press, Inc. FormEntry is a trademark of Widget Press, Inc.

## **Acknowledgements**

Portions of this Widget Press Software may utilize the following copyrighted material, the use of which is hereby acknowledged:

Mac Copyright © 2015 Apple, Inc.

iPhone Copyright © 2015 Apple, Inc.

iPod touch Copyright © 2015 Apple, Inc.

iPad Copyright © 2015 Apple, Inc.

DATE	CHANGES
2/15/11	Added information about posting to an internal SSL web server were the iOS device needs the digital certificate installed.
5/10/11	Updated for latest functionality
10/1/11	Updated for FormEntry 2.1, new controls and features.
2/15/12	Added FTP Transmittal Plug-In info, FormEntry FORM (.form) file format into and PHP Archive Info.
3/20/13	Updated for FormEntry 2.3, new controls and features.
10/10/13	Updated with new info on Paid Plug-Ins. New URL Schema.
2/25/14	Updated PHP Archive with additional info. Misc updates.

# CONTENTS



# **Welcome to FormEntry** **13**

## **FormEntry for Mac Features at a Glance** **13**

<i>Sample Forms</i>	14
<i>Wi-Fi Deployment</i>	14
<i>FormEntry Typical Workflow</i>	15
<i>Send Email Post</i>	16
<i>Send HTML Post</i>	21
<i>Send JSON Post</i>	21
<i>Timer Expiration</i>	21
<i>Location Expiration</i>	21

## **Controls at a Glance** **23**

<i>Text Field (§1)</i>	23
<i>On-Off Switch (§2)</i>	23
<i>Checkmark (§3)</i>	23
<i>Text Box (§4)</i>	24
<i>Date &amp; Time Picker (§5)</i>	24
<i>Countdown</i>	24
<i>Date</i>	24
<i>Date and Time</i>	24
<i>Stopwatch</i>	24
<i>Time</i>	25
<i>Photo Picker (§6)</i>	25
<i>Signature (§7)</i>	25
<i>Spin Wheel Picker (§8) and Spin Wheel Item (§9)</i>	25
<i>Location (^0)</i>	26
<i>Hidden (^1)</i>	26
<i>Barcode (^2)</i>	26

<i>Image (^3)</i>	26
<i>Ratings (^4)</i>	26
<i>Web Browser (^5)</i>	27
<i>PIN (^6)</i>	27
<i>Groupings (⌘G)</i>	27
Field-set	27
Drill-Down	27
<i>Icons</i>	27
Form Icons	28
Grouping Icons	28
Section Icons	28
<b>FormEntry Setup</b>	<b>29</b>
<b>Install FormEntry for Mac</b>	<b>29</b>
<b>Install FormEntry Touch</b>	<b>29</b>
<b>Basic Concepts of FormEntry</b>	<b>31</b>
<i>Wi-Fi Syncing</i>	31
<i>FormEntry FORM Format (.FORM)</i>	33
<b>Overview of FormEntry</b>	<b>36</b>
<b>Single User Interface</b>	<b>37</b>
<b>Sections and Groupings</b>	<b>38</b>
<i>Sections</i>	38
<i>Groupings</i>	39
Field-set Grouping	39
Drill-Down Grouping	39
<b>Controls</b>	<b>39</b>
<i>FormEntry - The User's Manual</i>	6

<i>Control Tool Bar Pane</i>	39
Footer Text	40
Validation Rules	41
Initial Keyboard Type for Input/Date Picker for Input	41
Initial Keyboard type for Input	41
Date Picker for Input	41
Web Services	41
Configuration	42
<i>Native iOS and Custom Controls</i>	43
Text (§0)	44
Text Field (§1)	44
On-Off Switch (§2)	45
Checkmark (§3)	46
Text Box (§4)	47
Date & Time (§5)	48
Photo Picker (§6)	48
Signature (§7)	50
Spin Wheel Picker (§8)	50
Spin Wheel Item (§9)	51
Location (^0)	52
Hidden (^1)	53
Barcode (^2)	54
Image (^3)	55
Ratings (^4)	56
Web Browser (^5)	57
PIN Code (^6)	58

## **Settings 59**

<i>General Info</i>	59
---------------------	----

<i>Appearance</i>	61
<i>Sending Form</i>	61
<i>Author Info</i>	62
<i>DSV and PDF Settings</i>	63
<b>Overview of Database</b>	<b>66</b>
<b>Setting up your Database</b>	<b>66</b>
<i>Updating Database via a Web Service</i>	69
Set XML Web URL	69
User Initiated POST Request	70
Web XML URL	70
<i>Column Data Types</i>	73
<i>Standard And Custom Data Types</i>	73
<b>Binding Controls to Database</b>	<b>76</b>
<i>Text Field Bindings</i>	77
Bindings Summary	78
<i>Retrieving Same Table Values</i>	78
<i>Key Path Traversing to Parent Table</i>	80
Update Products Table	81
Add new Location Table	81
Key Path Traversing in Rule Builder	83
Key Path Summary	84
<i>Traversing to Grandparent Table</i>	84
Update Location Table	85
Add New Types Table	86
Key Path Traversing in Rule Builder	87
Key Path Summary	88
<i>Spin Wheel Picker Bindings</i>	89
Switch to Spin Wheel Picker Condition	90
<i>FormEntry - The User's Manual</i>	8

Bind Icons and Image Key Paths	91
Image Summary	97
Filtered Spin Wheel Picker	97
Filtered Summary	100
<i>PIN Code Bindings</i>	<i>100</i>
PIN Code Binding Summary	102
<i>Barcode Bindings</i>	<i>103</i>
Add New Column ID	103
Barcode Key Path	105
Barcode Binding Summary	106
<b>Handling Data &amp; Responding</b>	<b>108</b>
<b>Email Transmittal Plug-In</b>	<b>108</b>
<i>File Names</i>	<i>109</i>
<b>HTML and JSON Post Transmittal Plug-Ins</b>	<b>109</b>
<i>Responding to FormEntry Touch</i>	<i>110</i>
<i>Internal SSL digital Certificates</i>	<i>112</i>
<i>Exporting to PHP Archive</i>	<i>112</i>
Handling Localized Keyword Parameters and MySQL Database Columns	114
<b>Printer (AirPrint) Transmittal Plug-In</b>	<b>117</b>
<b>File Transfer Protocol (FTP) Transmittal Plug-In</b>	<b>118</b>
<i>Directory Paths</i>	<i>121</i>
<i>File Names</i>	<i>121</i>
<b>Open In Transmittal Plug-In</b>	<b>121</b>
<i>File Names</i>	<i>122</i>
<b>WebDAV Transmittal Plug-In</b>	<b>122</b>
<i>Directory Paths</i>	<i>124</i>

<i>File Names</i>	125
<b>Save Transmittal Plug-In</b>	<b>125</b>
<b>Keyword and Reserved Parameter IDs</b>	<b>127</b>
<i>Reserved Parameter Descriptions</i>	128
<i>Accessing Parameter IDs</i>	132
<b>Check Form Version</b>	<b>135</b>
<i>Example of using the Check Form Version</i>	137
<b>Using FormEntry URL Scheme</b>	<b>138</b>
<b>Remotely pre-populate a Form with data</b>	<b>138</b>
<i>FormEntry URL Scheme Example</i>	139
<b>Control configuration through URL Scheme</b>	<b>140</b>
<b>Using PDF Technology</b>	<b>143</b>
<b>Laying out the PDF elements</b>	<b>145</b>
<i>Using PDF Alias</i>	146
<b>Rule Builder</b>	<b>148</b>
<b>Rule and Condition Sequence and Rule actions</b>	<b>149</b>
<i>FormEntry Touch Event Bubbling Stack</i>	149
<i>Rules and Conditions Bubbling Stack Flowchart</i>	151
Rule Alert	152
Controls table	152
Execute a Math Expression on a Control	152
Boolean Values for Checkmarks and Switches	153
Set a String Value on a Control	153
Update the Label Name of a Control	153
<i>FormEntry - The User's Manual</i>	10

Update the Image Control or Icons of a Control	153
<i>Example FormEntry Project with Rules</i>	153
Create a New Project	153
Create First Rule To Enable a Text Field	155
Create Second Rule with a Math Expression and Test Math	158
Create a Third Rule with a Math Expression	161
<b>Math Expressions</b>	<b>163</b>
<b>Built-in Operators</b>	<b>164</b>
<i>Considerations</i>	166
The Degree Operator	166
Logical Values	166
Factorial and Logical Not	167
Parentheses and Associativity	167
Operator Associativity	167
<b>Built-in Functions</b>	<b>167</b>
Functions that take > 1 parameter	167
Functions that take 1 parameter:	168
• The trigonometric functions:	168
Functions that take no parameters ("constant functions"):	169
<b>Implicit Multiplication</b>	<b>170</b>
<i>Handling Operators</i>	170
<i>Functions and Implicit multiplication</i>	171
<i>Argument-less Functions</i>	171
<b>Tokenizing</b>	<b>171</b>
<i>Numbers</i>	171
<i>Functions</i>	172
<i>Variables</i>	172
<i>FormEntry - The User's Manual</i>	11

<i>Operators</i>	172
<i>Regarding Whitespace</i>	173
<b>Term Grouping</b>	<b>173</b>
<i>Group terms</i>	173
<i>Function term</i>	173
<i>Number term</i>	173
<i>Variable term</i>	174
<i>Operator term</i>	174
<b>Publishing to FormEntry Server</b>	<b>175</b>
<b>Resources for More Learning</b>	<b>179</b>
<i>Onscreen Tooltip Help</i>	179
<i>Web Resources</i>	179



# WELCOME TO FORMENTRY

**With FormEntry for Mac, impressive forms are just the beginning. This section provides an overview of FormEntry features.**

FormEntry for Mac is a rapid application development tool for creating professional-quality, native, form-based applications with the power of touch-control input.

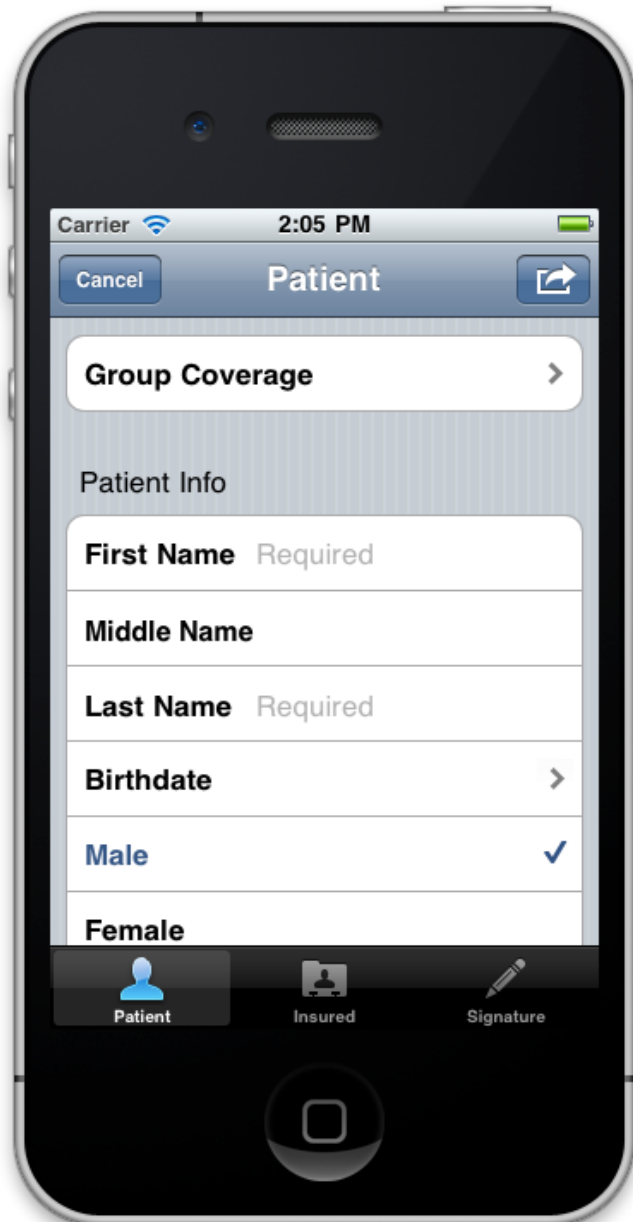
FormEntry Touch is the app installed on a device which accepts forms from FormEntry for Mac and allows users to complete and send form data.

FormEntry for Mac and FormEntry Touch work together seamlessly to develop forms and sync them for use on a device.

Creating a form for a device is now simple and straight forward. Drag and drop controls, add icons, update parameters, and edit each post setting. Never has form development for devices been easier.

## FormEntry for Mac Features at a Glance

The next few pages provide a high level overview of the key *FormEntry* features. The remainder of the manual provides the details of creating and building a form-based application.



**Note 1:** When discussing forms, we tend to refer to the following as having the same meaning: form-based application, FormEntry Touch document, form, form doc, form and form application. From this point forward these documents are referred as a “form”.

**Note 2:** From this point forward an iPhone, iPod Touch, iPad, or iOS device are referred to as a “device”.

## SAMPLE FORMS

Three sample forms are provided in FormEntry for Mac to demonstrate the capabilities of *FormEntry*. The samples are Healthcare, Home Inspection, and Restaurant Menu and are accessible from the FormEntry for Mac menu at *File -> Open Projects Window*.

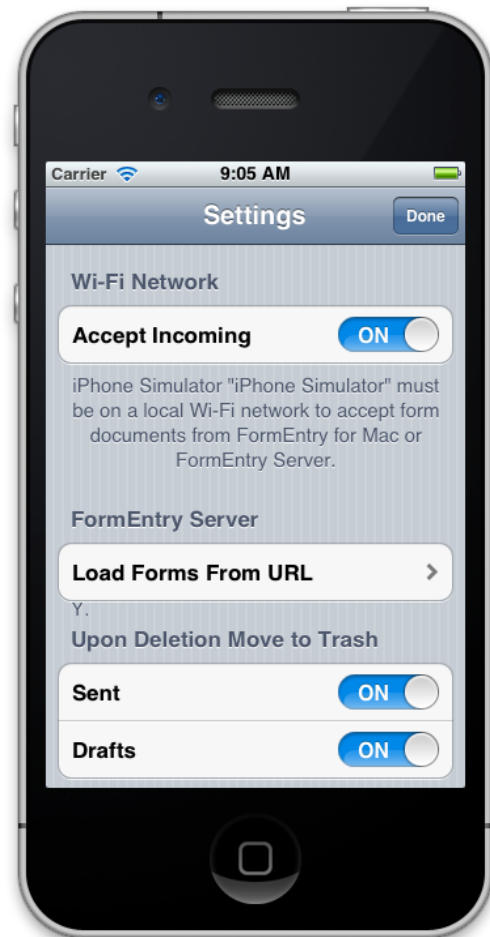
## WI-FI DEPLOYMENT

Transferring a form to a device is easy. In fact, no wires are needed. Simply enable 'Accept Incoming' in FormEntry Touch under the 'Settings' screen and click 'Sync Devices' (⌘⇧) from FormEntry for Mac. The form is sent to any available device.

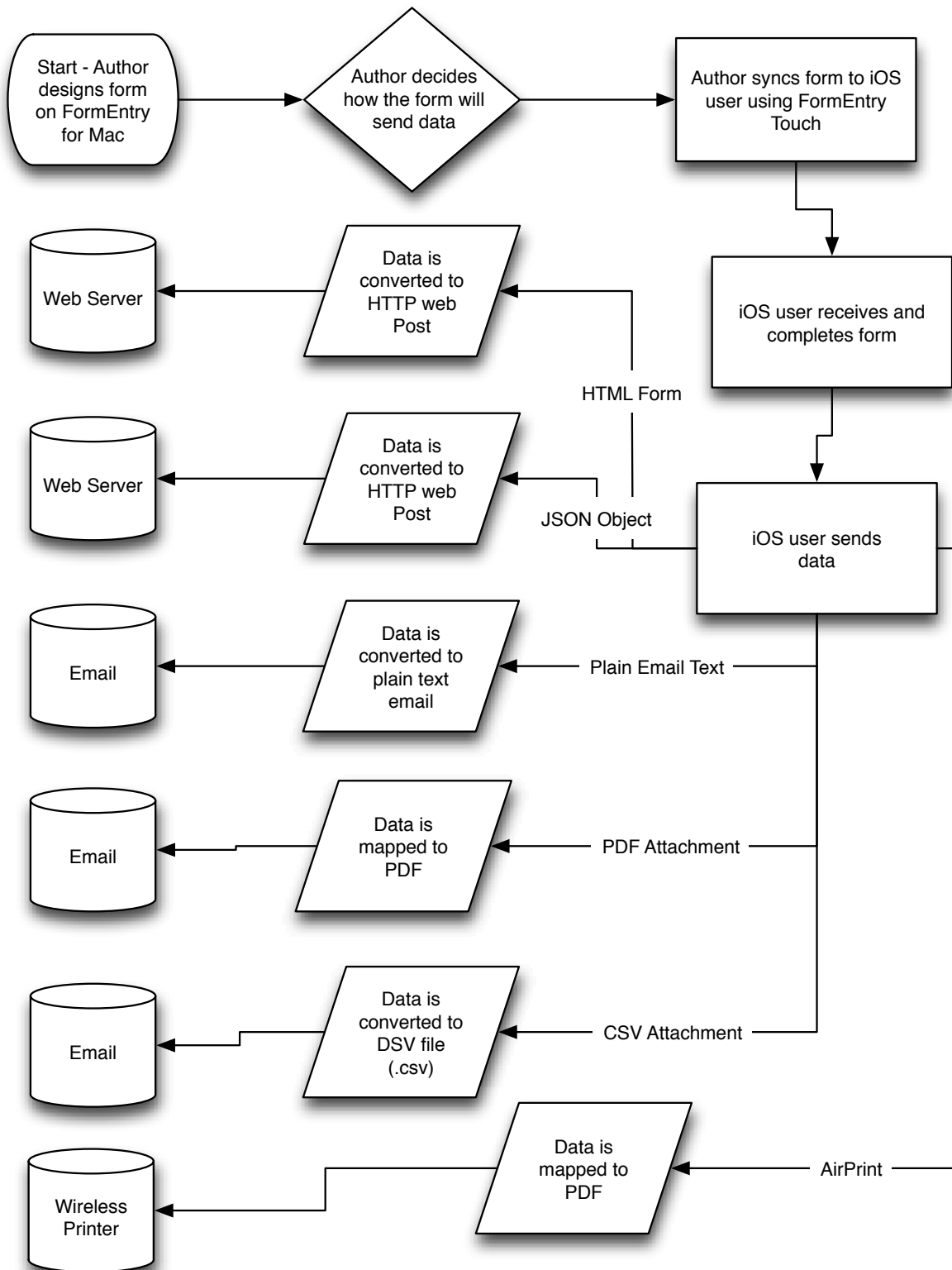
Note that both FormEntry Touch and FormEntry for Mac must be on the same Wi-Fi network to sync forms. An ad hoc Wi-Fi network can be easily created to send forms to any available device. Make sure that Airport, Apple's wireless networking interface, is active on the Mac. Then in the Airport menu, select 'Create Network...'.  
Y.

Enter a name for the network in the 'Name:' field and have the device users join the network just created.

There is also an option to set a form to expire once a user leaves the original paired Wi-Fi network. See [Location Expiration](#) for details.



## FORMENTRY TYPICAL WORKFLOW



## **SEND EMAIL POST**

Need the ability to send an email with form data? No problem. Form data is sent to any email address once the email function is enabled. This capability provides simplicity and power for the mobile user.


Also, photos can be attached and sent via email from the Photo Library or taken with the device's camera.

The FormEntry author can create a default email address, saving users' time and trouble. This feature also prevents users from searching and selecting an email address from a list of contacts, thereby removing any potential error.

When sending an email post, you have the option of sending your data as plain email text, a PDF file or as a DSV file with a .csv file extension, or if installed, in Microsoft Excel .xlsx file (The Microsoft Excel Transmit Format is a paid plug-in).

General Info | Appearance | Sending Form | Author Info | DSV and PDF Settings

### Transmittal Plug-Ins

☒  Email Transmittal Plug-In

Default Email Address:


Button Label:

Default Subject Line:

Display Button: ☒ On New Form & Drafts

Transmit Format:

File Name:

☐  HTML Post Transmittal Plug-In

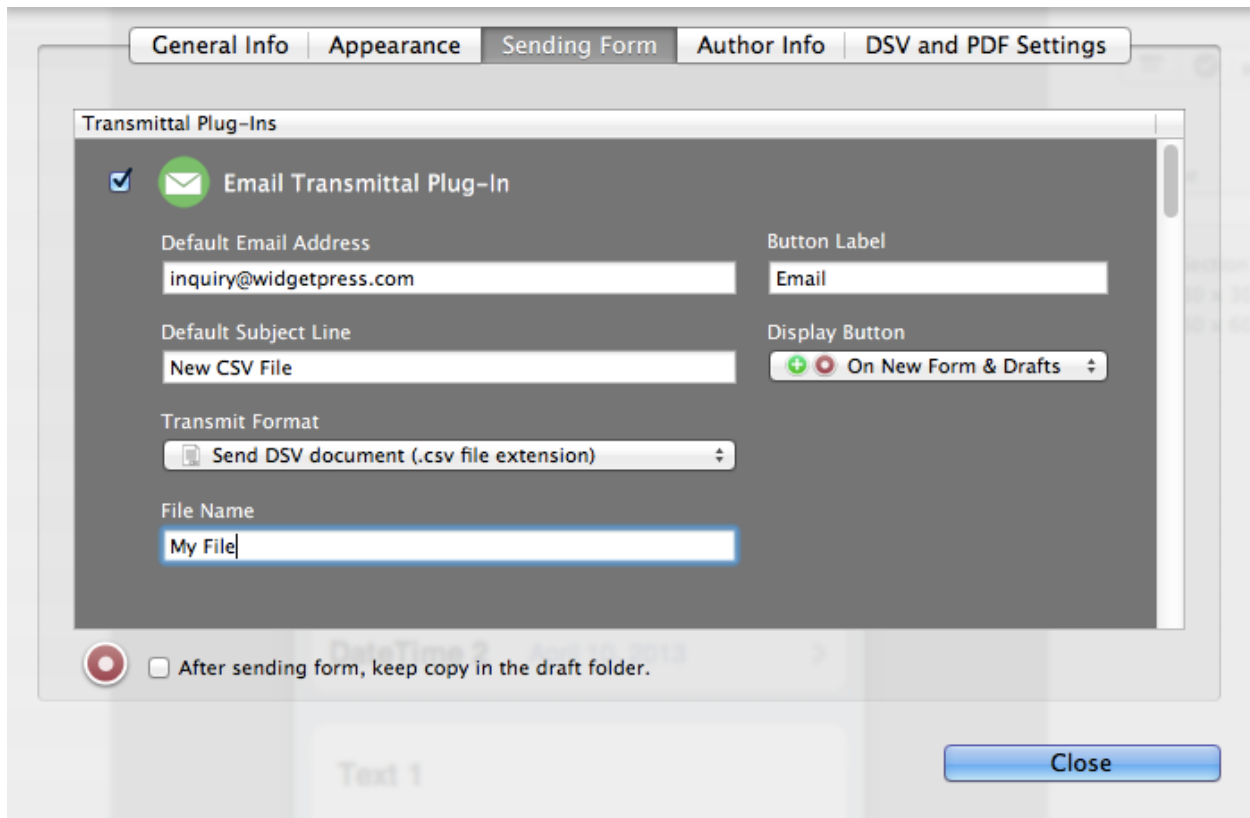
URL to post as HTML form:

Button Label:

☒ ☐ After sending form, keep copy in the draft folder.

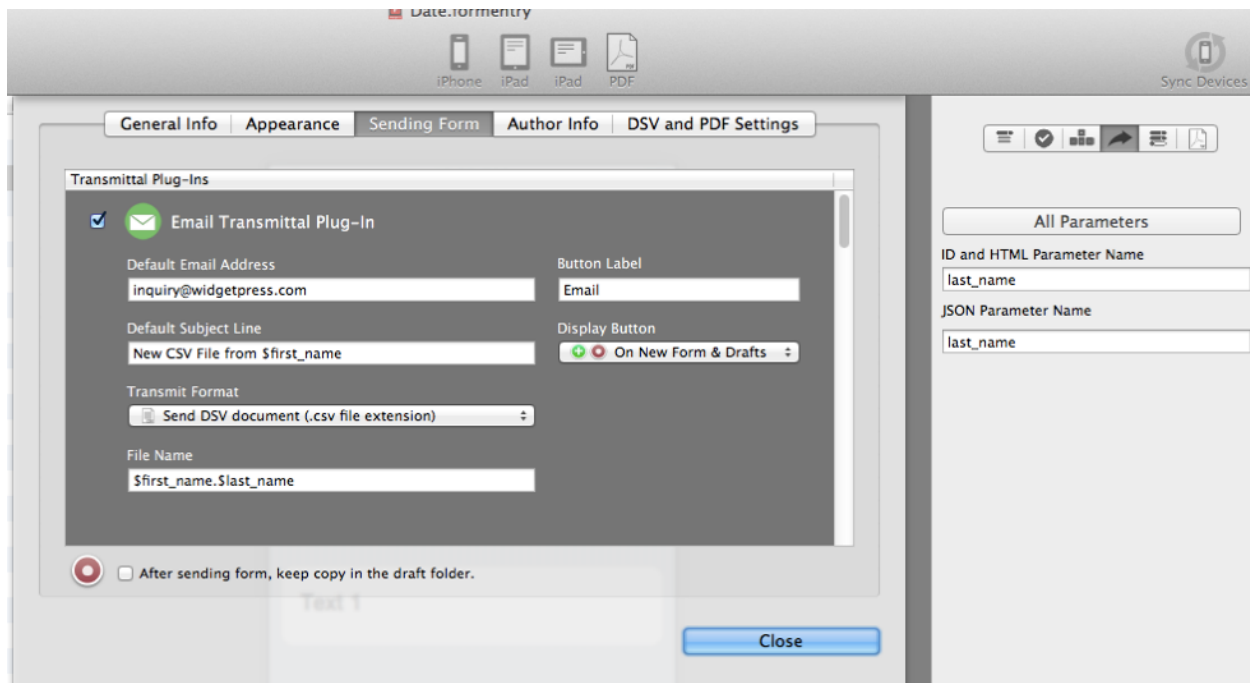
Close

When sending a file attachment from the Email transmittal Plug-In, you will need to specify a Default Subject Line and File Name.



When the user sends the form, the Subject Line and File Name are created and attached within the Mail app.

You can also create dynamic file names and subject lines based upon form values. This type of parameter naming convention for file names also applies to the WebDAV, FTP, Open In Transmittal Plug-Ins.



Dynamic file name and dynamic subject lines are used by the Param ID of controls. For example, if you wanted the values from the form and place them within the subject line and or have the file name be created based upon the values of a control, you can do that.

Simply use the Parameter ID of the control using the “\$” in front. In the above screenshot, the “last\_name” parameter ID is placed within the creation of the file name:

```
$first_name.$last_name
```

Notice you can mix in different parameter IDs together. When the user completes the form with the first name “Jonathan” and last name of “Freeman”, the file name created for the CSV file will be:

```
Jonathan.Freeman.csv
```

AT&T 9:58 AM 100%

Cancel

New CSV File fro...

Send

To: inquiry@widgetpress.com

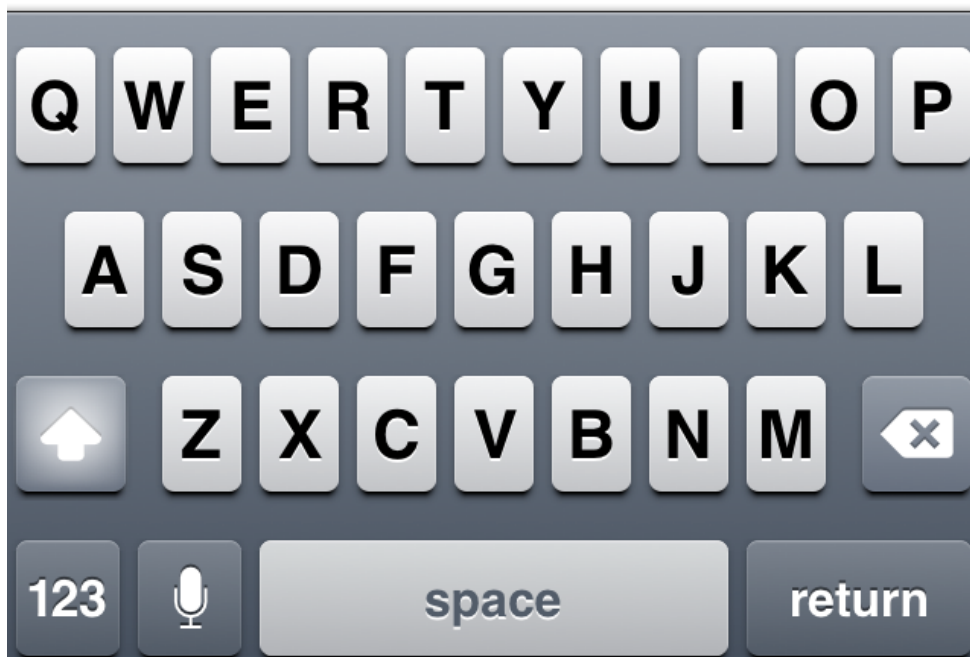
Cc/Bcc, From: support@widgetpress.com

Subject: New CSV File from Jonathan



Jonathan.Freeman.csv

Sent from my iPhone





## **SEND HTML POST**

The form can post data to a web site over HTTP as a standard HTML form *'POST'*. All data is sent from FormEntry Touch using standard web-based technologies.

Each HTML form post uses the *'POST'* method and is multipart assuring that all the data including photos are sent correctly.

## **SEND JSON POST**

The form can easily send all user data as JSON (JavaScript Object Notation) over HTTP.

Each form sent to the web site is a standard HTML form *'POST'* using the single parameter: *'json'*. If photos are being sent with the form, each photo is transformed to BASE64. For more about BASE64 encoding, please refer to this [Wikipedia](http://en.wikipedia.org/wiki/Base64) article (<http://en.wikipedia.org/wiki/Base64>).

Specify the JSON key for a control under the ['Web Services'](#) inspector for that control.

## **TIMER EXPIRATION**


Forms can expire with a timer. This feature is perfect for the classroom during exams, quizzes or lab work. If the timer expiration is enabled, users are not be able to access the form for input, or view any sent items, drafts, or items in trash boxes.


## **LOCATION EXPIRATION**


A form can be set to expire if the users leave the originally paired Wi-Fi network. This is an excellent option when sensitive information is entered, reviewed, and sent from a specific location.


**Important:** Location expiration only works over a shared Wi-Fi network not an “ad hoc” computer-to-computer network.


General Info | Appearance | Sending Form | Author Info | DSV and PDF Settings



 Form Icon  
30 x 30 pixels  
60 x 60 pixels (Retina Display)


 ☐ Set Navigation Bar Color

 ☐ Set Background Color

 ☐ Lock this form in Portrait Mode on an iPad.

 ☒ Display "Send All Drafts" Button on Drafts.

Display "Cancel" Button:   On New Form & Drafts

 ☒ Display "Move to Drafts" Button on Sent forms.

Close

## Controls at a Glance

The section provides a high level overview of the FormEntry for Mac controls used to collect user data. FormEntry for Mac offers seventeen native iOS controls and several organization widgets for the controls. Additionally, each of these controls allows for the display of a multi-line footer below the control or group of controls. The footer serves as an excellent place to add contextual information.

Two of the controls are paid Plug-In controls that can be purchased in the In-App Purchase Plug-In Store within FormEntry for Mac.

### TEXT FIELD (§1)

The Text Field control handles small amounts of editable text. The default keyboard can be set that appears when the Text Field has been tapped. The choices are either the standard keyboard, or the numeric keyboard which allows faster access to the number pad.

Text boxes also have the ability to apply validation rules, such as a *'Required Field'*.

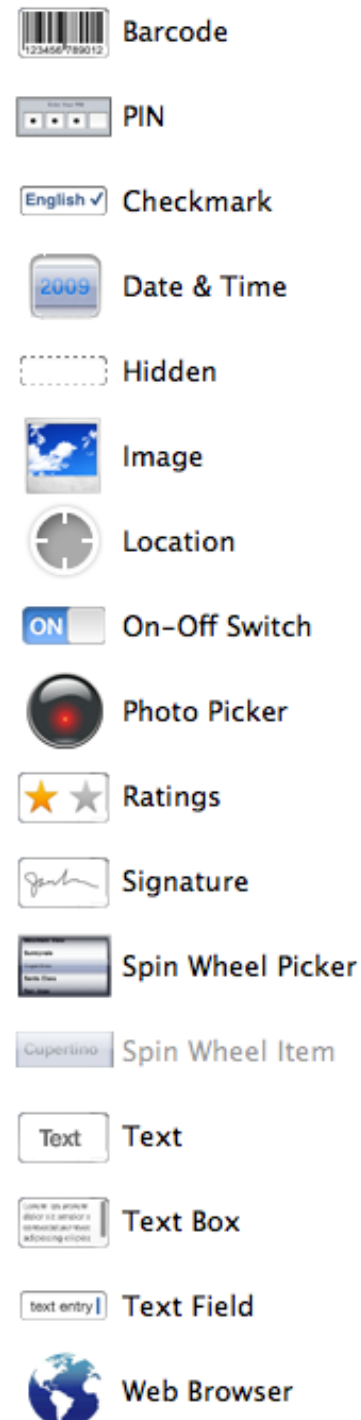
### ON-OFF SWITCH (§2)

The ubiquitous *'On-Off Switch'* is a standard control on the device. Users can touch the switch to enable or disable this item.

This is a paid plug-in and can be purchased within the In-App Purchase Plug-In Store within FormEntry for Mac.

### CHECKMARK (§3)

The Checkmark control allows the user to tap the control to check or uncheck an item.



FormEntry for Mac also handles grouped information. Just place the checkmarks inside a grouping and enable the '*Checkmarks as a single select*' in the group's '*Configuration*' inspector. When enabled, the form allows only one checkbox selection in the grouping.

This is a paid plug-in and can be purchased within the In-App Purchase Plug-In Store within FormEntry for Mac.

## **TEXT BOX (§4)**

The '*Text Box*' control is designed to handle a larger amount of editable text which can not fit in a standard '*Text Field*'. Additionally, the '*Text Box*' allows the user to input text in a multi-line format.

By default, a '*Text Box*' does not show a label. However a label name is still required because the label name is used by the '*Web Services*' inspector to generate HTML and JSON parameter names. For instance, the label name '*Property Condition*' is converted into the HTML and JSON parameter name '*property\_condition*'. The label name for the '*Text Box*' is set in the '*Configuration*' inspector in the '*Label Name*'.

To provide a label for the users, place the text box within a grouping and then use the grouping's label name.

## **DATE & TIME PICKER (§5)**

Need to collect different dates and time from users? FormEntry for Mac offers four '*Date & Time*' options using the standard "spinner" control:

### **COUNTDOWN**

Collects hours and minutes.

### **DATE**

Collects the date in a month, day and year format.

### **DATE AND TIME**

Collects the month, day, hour, minutes and AM or PM.

### **STOPWATCH**

Collects the time from Start to Stop.

## **TIME**

Collects the hours, minutes and AM or PM.

## **PHOTO PICKER (§6)**

The *'Photo Picker'* adds photos to forms by including photos taken with the device's camera or by selecting photos from the device's Photo Library.

When designing the form, the photo choices are the *'Kind:'*, *'JPEG Quality:'* or *'PNG'* and the photo *'Size'*. If JPEG is selected, the image quality can be selected from *'Low'*, *'Medium'*, *'High'* or *'Maximum'*.

When the users send their forms, the photos are sent in the correct format. For email, the users' photos are sent as an email attachment. For HTML posting, photos are sent as a standard web file upload posting. For a JSON object, the photos are transformed into a BASE64 formatted string.

This is a paid plug-in and can be purchased within the In-App Purchase Plug-In Store within FormEntry for Mac.

## **SIGNATURE (§7)**

The *"Signature"* control allows the capture of a signature within the form. The user's signature is captured with just a stroke of an index finger. When the users send their forms, the signature is sent in a PNG format. For email, the users' signatures are sent as an email attachment. For HTML posting, the signature is sent as a standard web file upload posting. For sending as a JSON object, the signatures are transformed into a BASE64 format string.

This control also has a multi-line footer text area which serves as a great place to add contextual information, such as 'I agree' or 'I authorize'.

This is a paid plug-in and can be purchased within the In-App Purchase Plug-In Store within FormEntry for Mac.

## **SPIN WHEEL PICKER (§8) AND SPIN WHEEL ITEM (§9)**

The Spin Wheel Picker provides a single selection of a list of mutually exclusive items. An unlimited number of Spin Wheel Items can be added to the Spin Wheel Picker. Also, each item can include a 30 x 30 pixel icon.

This is a paid plug-in and can be purchased within the In-App Purchase Plug-In Store within FormEntry for Mac.

## **LOCATION (^0)**

With the Location control, you can gather the users GPS coordinates at the latitude and longitude degrees. You can also have the control to be refreshable, meaning the user can keep selecting the control to update their coordinates. If you have refreshable unchecked, the user has only press the location one time to gather their coordinates.

This is a paid plug-in and can be purchased within the In-App Purchase Plug-In Store within FormEntry for Mac.

## **HIDDEN (^1)**

With the Hidden control you can add field values that are not visible to the user. The value can be used to set other input values with the Rule Builder.

## **BARCODE (^2)**

The Barcode control allows your users to scan up to 16 different types of barcodes.

This is a paid plug-in and can be purchased within the In-App Purchase Plug-In Store within FormEntry for Mac.

## **IMAGE (^3)**

The Image control allows you to place full size images within the forms. It also has the ability to allow the user to draw upon the image.

This is a paid plug-in and can be purchased within the In-App Purchase Plug-In Store within FormEntry for Mac.

## **RATINGS (^4)**

The Ratings control is a 5 star ratings control that allows the user to rate.

This is a paid plug-in and can be purchased within the In-App Purchase Plug-In Store within FormEntry for Mac.

## **WEB BROWSER (^5)**

The Web Browser control allows the user to load a remote web page, specified by a URL from the form author.

This is a paid plug-in and can be purchased within the In-App Purchase Plug-In Store within FormEntry for Mac.

## **PIN (^6)**

The PIN Code control allows the form author to lock the entire form or sections of the form by using a 4 digit PIN number.

This is a paid plug-in and can be purchased within the In-App Purchase Plug-In Store within FormEntry for Mac.

## **GROUPINGS (⌘G)**

Groupings organize blocks of form input by breaking up the input into logical sections. Groupings make the user experience more intuitive and pleasant. There are two types of Groupings: Field-set and Drill-down.

### **FIELD-SET**

In a Field-set grouping, the input items are grouped together into a section. A Field-set can have a label name as well as a '*Navigation Icon*' for the grouping.

### **DRILL-DOWN**

A Drill-down group places the input controls on a secondary view. The label name acts as a button. Tapping the label causes the grouped controls to be displayed on the next view.

## **ICONS**

Adding icons to a form helps users identify sections, drill-downs, headers, and forms quickly and easily. These visual indicators enhance the user experience by enabling faster control identification, greater visual appeal, faster workflows, and a finer polish to the forms. All icons are resized to a 30 x 30 pixel height and width, however if you place a 60 x 60 icon, they will be supported on Retina Display iOS devices. Three types of icons are:

## **FORM ICONS**

Adding a 30 x 30 form icon helps the users identify the form on application launch. These icons support 24-bit color channels.

## **GROUPING ICONS**

The location of a grouping icon depends on the grouping type. The Field-set grouping icon is placed to the left of its label. The Drill-down icon is placed to the left of the row. Both types of icons support 30 x 30 icons and 24-bit color channels.

## **SECTION ICONS**

The section icons appear in a tab bar at the bottom of the form if there is more than one section. These icons are referred to as 'Tabs' in iOS lingo. Tabs are a common interface element used in countless iOS applications and help differentiate the sections. These tabs support 30 x 30 images. The images are transformed into a gray icon in the unselected state. Upon selection, the icon color changes to a blue highlight.



# FORMENTRY SETUP

**FormEntry must be installed on both the Mac and the device platforms to make the development environment fully operational.**

## Install FormEntry for Mac

FormEntry for Mac installation is simple. Download FormEntry for Mac from the Mac App Store.

To add FormEntry for Mac in the Dock, drag the FormEntry for Mac icon from the 'Applications' folder to the Dock. This creates a quick access to the FormEntry for Mac application.

## Install FormEntry Touch

Installing FormEntry Touch is straight forward as well. Open the iTunes application, select the 'iTunes Store' and search for 'FormEntry'. Download the application and sync the device. It's that easy!



# BASIC CONCEPTS OF FORMENTRY

## The basic concepts needed to create a functioning form

A Mac with FormEntry for Mac installed and a device with FormEntry Touch installed is required to create a form. If you want to share your form immediately with an iPhone or iPad, a Wi-Fi network is required that is shared by both the Mac and the device. If a Wi-Fi network is not available, an 'ad hoc' Wi-Fi, computer-to-computer network can be created between the Mac and device.

You can also export your FormEntry for Mac project to the FormEntry FORM (.form) file format. You can then host this file on your website or email it to an iOS user to install upon their device.

Eventually, you want the data to come off the FormEntry Touch. There are a number of ways to get the data. Ultimately, the data needs to be sent by user the form after the user has completed data entry.

FormEntry Touch can send the form data in four ways: via email, HTML Post, JSON Post or AirPrint. The settings for sending data are provided in the '*Sending Form*' tab under '*Settings*' in FormEntry for Mac.

If FormEntry for Mac is configured to send emails, an accessible email account is required to receive the users' information. For both HTML Post and JSON Post, access to a web site is needed and the ability to process the incoming form data posted by the users. Typically, processing form data can be achieved using .ASP, PHP, JSP, Ruby, Python or almost any other web friendly language.

### WI-FI SYNCING

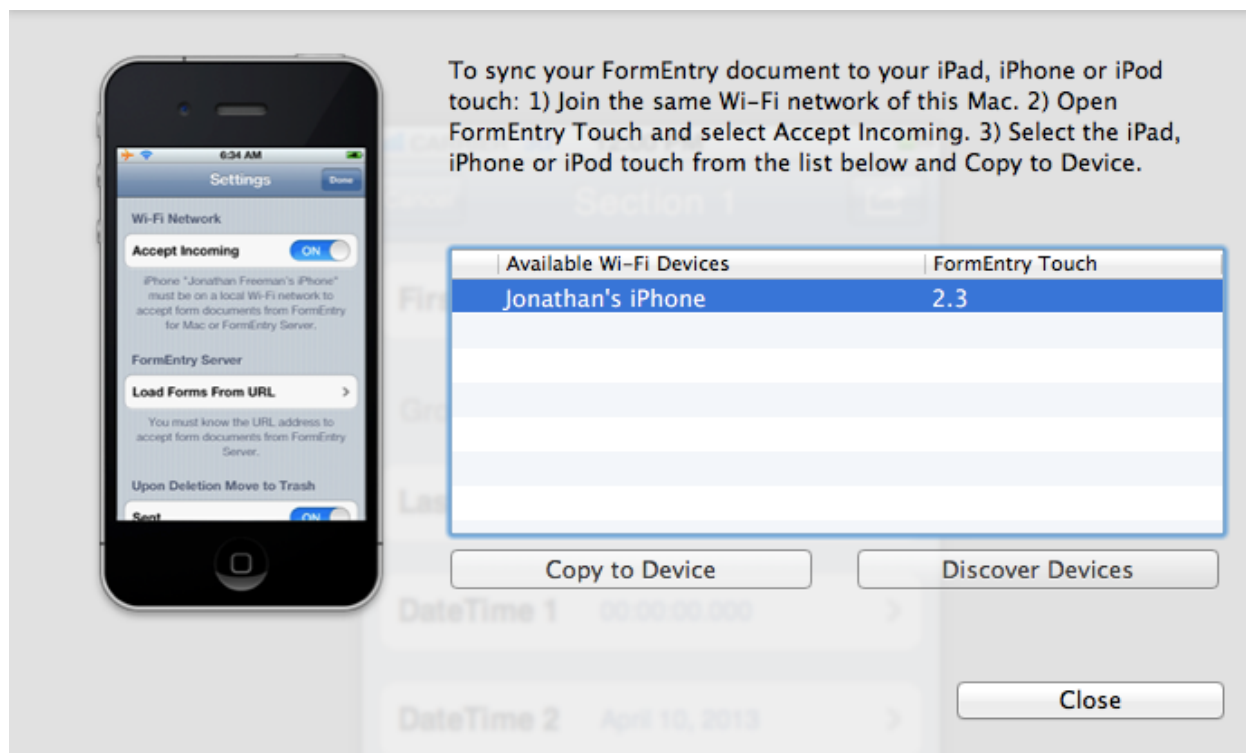
Test it frequently during development by syncing the Mac and device.

Enable '*Accept Incoming*' by tapping the '*Settings*' button on the FormEntry Touch screen and selecting ON.



Detect all devices connected to the Wi-Fi network in FormEntry for Mac by clicking 'Sync Devices' at the top-right of the window.

FormEntry for Mac detects and compares its software version with the versions of FormEntry Touch the running on the devices and determines if the form can be sent to each device.



For example, if the users are running a new version of FormEntry Touch and running an incompatible version of FormEntry for Mac, FormEntry for Mac does not allow the devices to sync.

After clicking the 'Sync Devices' button, choose the applicable devices by selecting them in the table and click the 'Copy to Device' button.

During the process of syncing devices, a green checkmark appears next to the devices which have successfully received the form.

Additionally, the device displays a popup indicating the form has been added to the device.

## FORMENTRY FORM FORMAT (.FORM)

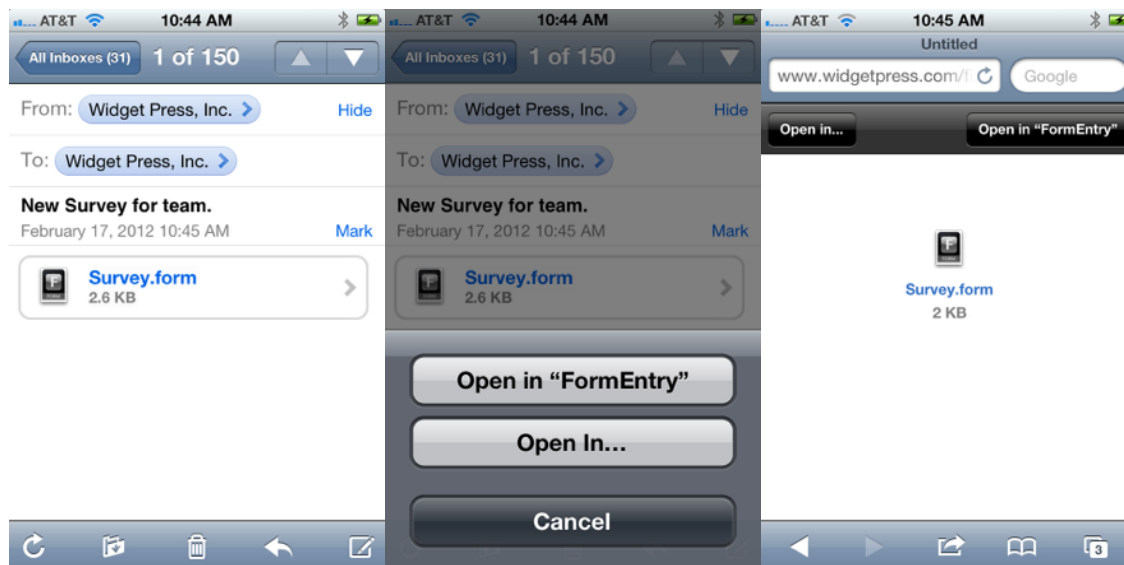
FormEntry for Mac allows you to export your project to the .form file format that can be read by FormEntry Touch. Simple choose File->Export... -> Export FormEntry FORM and save to your hard drive.

You can then place this file on your web site or email the file as an email attachment to an iPad or iPhone user. Your FormEntry Touch user can install your form from a link or from the email attachment by simply touching the icon and select “Open in FormEntry”.

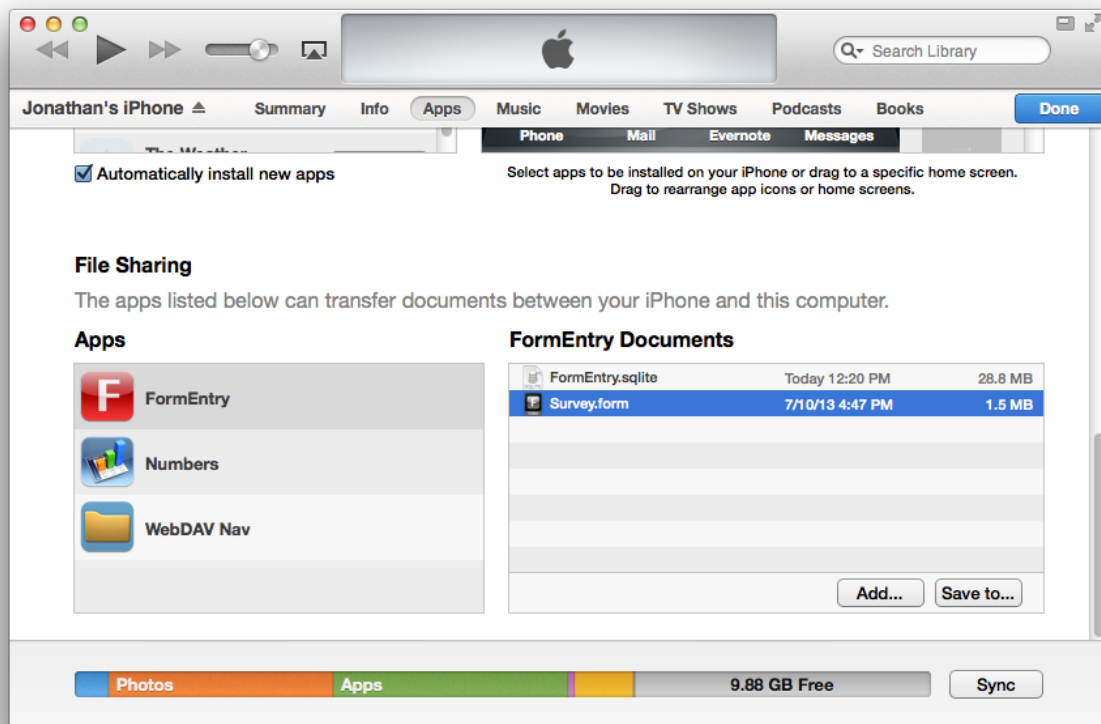
Your users will need to use Mail app, Mobile Safari and FormEntry Touch to install remotely.



Name	Survey.form
Kind	FormEntry Form Document
Size	2 KB
Created	Today 10:41 AM
Modified	Today 10:41 AM
Last opened	Today 10:41 AM



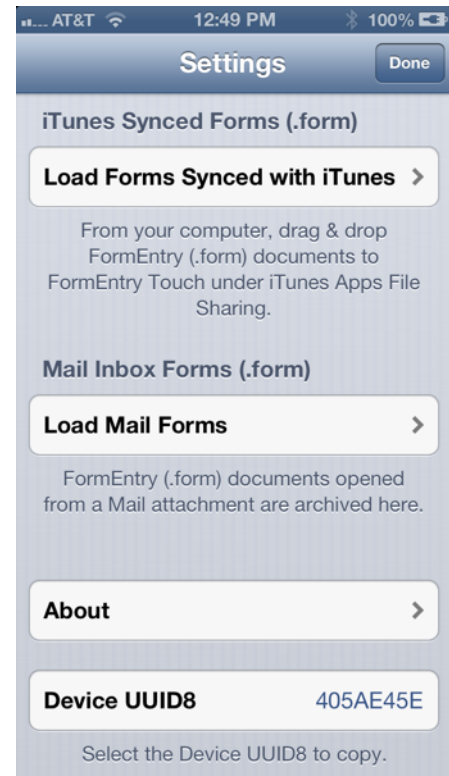
You can even take the .form file and sync it via iTunes through the USB cable. Simply plug in your iOS device into your computer using a USB Cable. Open iTunes on your computer and navigate to the Apps tab and down to FormEntry. Select FormEntry for the list of Apps under File Sharing.



Now select “Add...” button to add your .form file or simply drag and drop the file into the FormEntry Documents window. Select Sync to move the .form file into FormEntry Touch.

From FormEntry Touch, select the Settings button and navigate down to iTunes Synced Forms (.form) section. You will be able to load any .form files that are contained here.

You can also load an emailed .form attachments here as well. All .form files that were emailed and opened by FormEntry Touch are archived in the Mail Inbox Forms folder and can be loaded again.



# **OVERVIEW OF FORMENTRY**

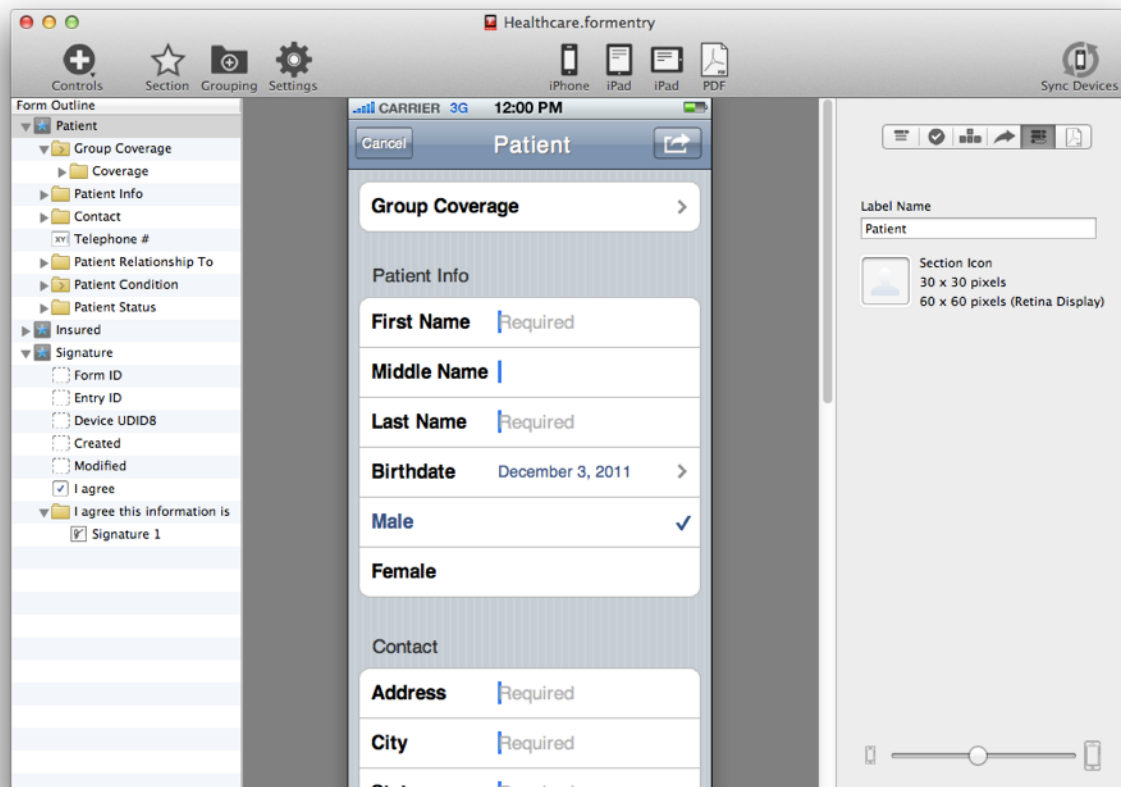
**This section describes the three main areas in the FormEntry User Interface: the Form Outline, the iOS Preview and the Control Tool Bar Pane, and their relationship to Sections, Groupings, or Controls.**

**This section also provides example snippets for each control.**



# Single User Interface

**Note:** Refer to the screenshot below during these descriptions.



The left-side section of the FormEntry for Mac window is the *'Form Outline'*. The *'Form Outline'* is the hierarchical structure of the controls in the form which allow the addition or deletion of the controls, sections or groupings. The items in the outline can be dragged and dropped to rearrange their order.

The *'iOS Preview'* located in the middle of the window shows the currently selected item when adding, deleting or rearranging items in the *'Form Outline'*. The *iOS Preview* is available in three formats selectable from the FormEntry for Mac menu under *View>Switch iOS Preview*.

The *'Control Tool Bar Pane'*, located on the right side of the window, displays the details of the currently selected control, section, or grouping and allows adjustments to their applicable properties.

## Sections and Groupings

Sections and Groupings are logical dividers within the form. Consider sections and groupings the 'branches' for different 'leaves' within the document.

### SECTIONS

Every form must have at least one section. A least a 30 x 30 pixel icon can be added to each section. However, the icon does not appear if there is only one section in the form. It is best to use images that are exactly 30 pixels x 30 pixels or 60 x 60 pixels for Retina Display devices. If the image is smaller or larger than this size, FormEntry for Mac stretches or shrinks the image accordingly. Simply drag an icon from your desktop unto the Section Icon cell and your icon will appear at the bottom as tab bar icons.

The screenshot displays the FormEntry application interface. On the left, a form is shown with several sections: 'Other' with an 'If Other:' field, 'Patient Condition' with a right arrow, 'Patient Status' with a list of 'Single', 'Married', and 'Other', and another 'If Other:' field. At the bottom of the form is a tab bar with three icons: 'Patient' (a person icon), 'Insured' (a folder icon), and 'Signature' (a pen icon). The 'Patient' tab is currently selected. On the right, a configuration panel is visible. It includes a toolbar with icons for undo, redo, save, and other actions. Below the toolbar, there is a 'Label Name' field containing the text 'Patient'. Underneath this is a 'Section Icon' field, which shows a placeholder icon and text indicating supported sizes: '30 x 30 pixels' and '60 x 60 pixels (Retina Display)'. At the bottom of the configuration panel is a zoom slider with a central knob and two smartphone icons at the ends.

A tab bar appears at the bottom of the screen if two or more sections are added. Note that the images for each section is transformed into gray icons. The icons is illuminated with a blue highlight when the icons are touched on the device.

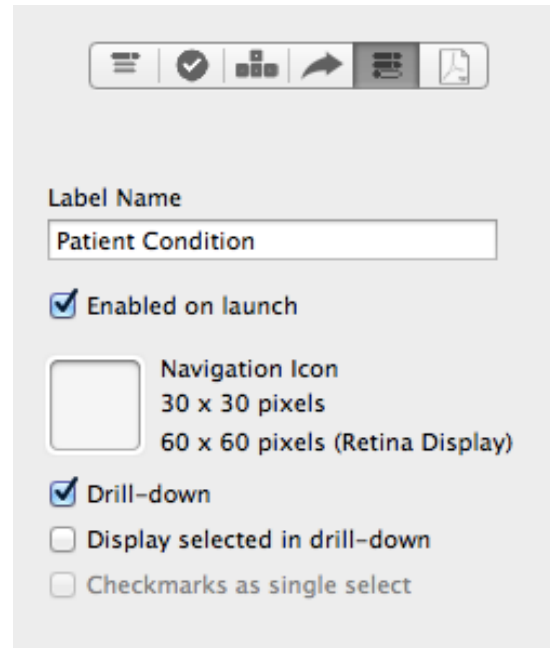
## GROUPINGS

Groupings hold a collection of controls and are either a Field-set grouping or a Drill-down grouping.

### FIELD-SET GROUPING

A Field-set grouping is the default when adding a new grouping. Field-set groupings join its controls together in a standard iOS table view. The label name for each grouping is used as the header. A 30 x 30 icon or 60 x 60 pixel icon can be added to this grouping and appears on the left side of the label name.

Checkmark controls within the grouping can optionally be treated as a unified control allowing only a single selection. Tapping on one checkmark deselects the other checkmarks. To enable single select for checkmarks, click the '*Checkmarks as single select*' checkbox in the '*Control Tool Bar Pane*' for the grouping.



The screenshot shows the 'Control Tool Bar Pane' configuration window. At the top is a toolbar with icons for list, checkmark, grid, arrow, and document. Below the toolbar, the 'Label Name' is 'Patient Condition'. There is a checked checkbox for 'Enabled on launch'. A 'Navigation Icon' section shows a 30 x 30 pixel icon and a 60 x 60 pixel icon (Retina Display). There is a checked checkbox for 'Drill-down', an unchecked checkbox for 'Display selected in drill-down', and an unchecked checkbox for 'Checkmarks as single select'.

### DRILL-DOWN GROUPING

A grouping can appear on another view by changing the grouping type to Drill-down in the '*Configuration*' section of the '*Control Tool Bar Pane*'. Once changed, the Drill-down grouping displays the label name with an arrow (in the table row) indicating it is a drill-down. The grouping displays on the next view when the user selects the row in the table.

## Controls

Controls are used to enable user interaction with *FormEntry Touch*. As controls are added, they are shown in order in the form outline. Also, the type of control is displayed in the '*iOS Preview*' as it would on a device. The controls are configured on the right-side of the FormEntry for Mac. This area is called the '*Control Tool Bar Pane*'.

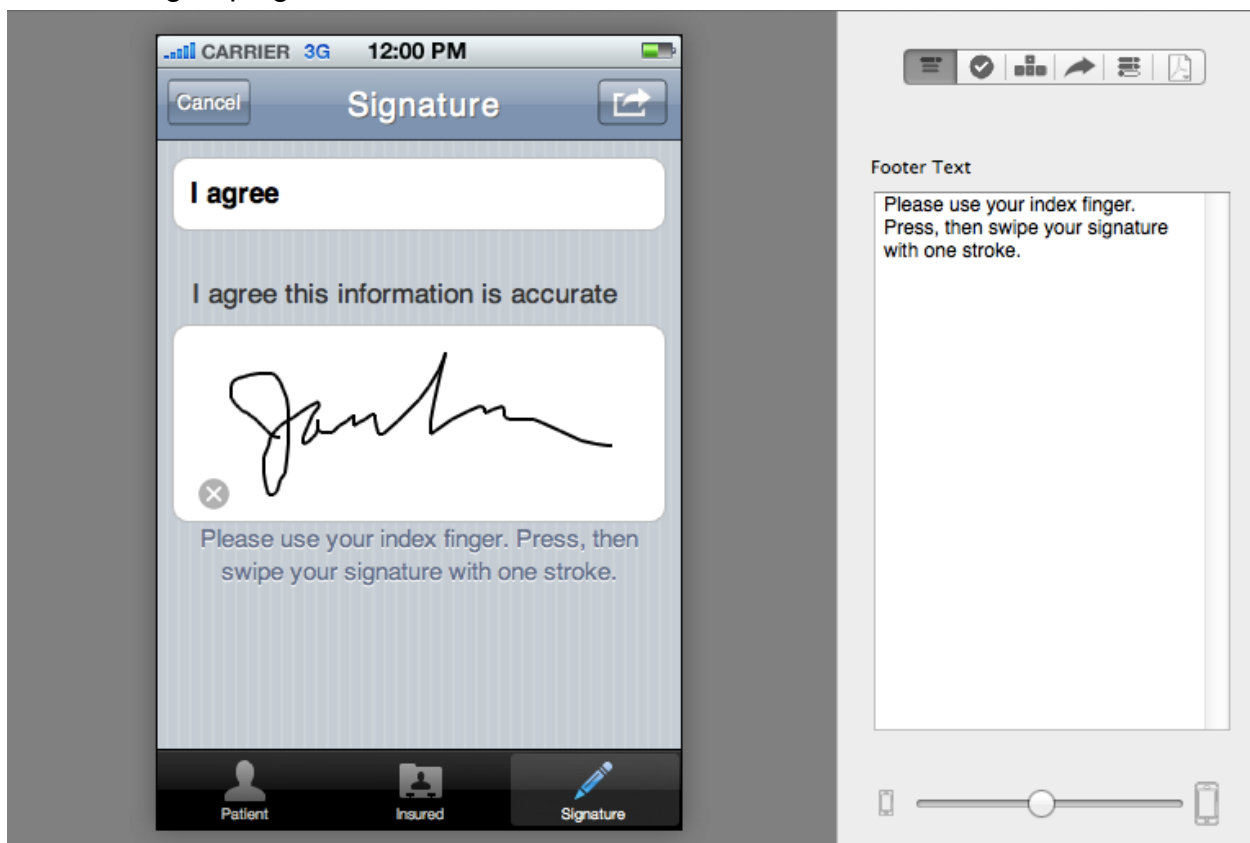
## CONTROL TOOL BAR PANE

The Control Tool Bar Pane can be thought of in Mac terms as the 'Inspectors' for each control, section or grouping in the form.

The six buttons located at the top of the pane representing these inspectors which are: Footer Text, Validation Rules, Initial Keyboard Type or Input/Date Picker for Input, Web Services, Configuration and PDF layout.

## FOOTER TEXT

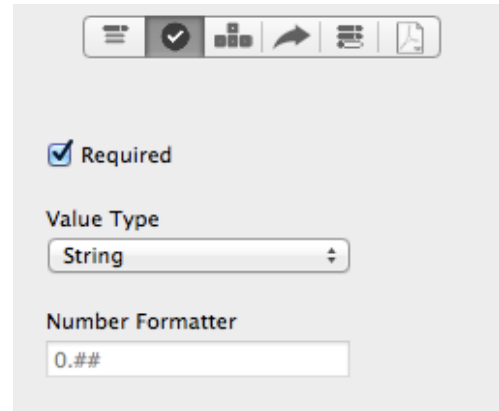
*Footer Text* allows changes to add ready-only text underneath the bottom of a control. They can be under most controls and grouping. Enter text in the Footer Text field to be displayed below a control or group. For controls, no text is displayed if the control is in a Field-set grouping.



## VALIDATION RULES

Select *'Required Field'* in the *'Control Tool Bar Pane'* if a text field is a required entry. *'Not Applicable'* is displayed for some controls.

If the control is a text input style control, you have the options of treating this control as Currency, Integer, Decimal, Percent, Scientific or a String. If its a decimal or integer, you have the options of treating the input value with a number formatter. If you would like to perform any calculations on these controls from the Rule Builder, you will want to set Value Type to integer or decimal.



The screenshot shows a configuration panel for validation rules. At the top is a toolbar with icons for menu, checkmark, grid, arrow, list, and document. Below the toolbar, the 'Required' checkbox is checked. The 'Value Type' dropdown menu is set to 'String'. The 'Number Formatter' text field contains the pattern '0.##'.

## INITIAL KEYBOARD TYPE FOR INPUT/DATE PICKER FOR INPUT

### INITIAL KEYBOARD TYPE FOR INPUT

FormEntry for Mac offers two types of keyboard types for users: *'Keyboard Default'* - the default QWERTY keyboard or *'Keyboard Numbers'* - a keyboard with numbers and symbols.

*'Return Key Type'* changes the text on the keyboard button to allow the user to continue editing the form after completing information in a text field. The choices of text are: *'Done'*, *'Next'*, *'Return'* or *'Send'*.



The screenshot shows the 'Initial Keyboard Type for Input' configuration panel. It features a toolbar at the top. The main section displays a virtual keyboard with two rows of buttons. Below the keyboard, the 'Keyboard Numbers' dropdown menu is selected. At the bottom, the 'Return Key Type' dropdown menu is set to 'Next'.

### DATE PICKER FOR INPUT

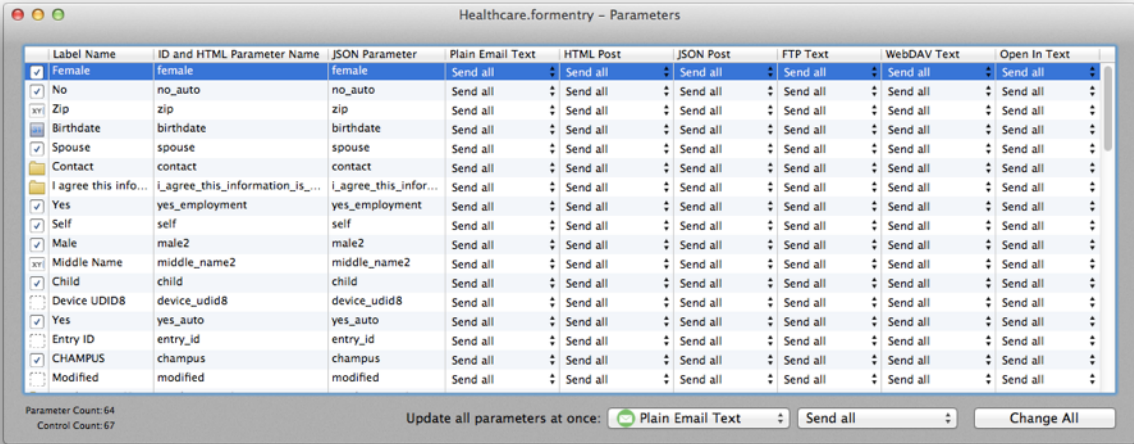
There are four types of configurable input for a Date & Time control: *'Countdown'*, *'Date'*, *'Date & Time'*, *'Stopwatch'* and *'Time'*.

## WEB SERVICES

The parameter name for the HTML or JSON object is entered in this field if *'URL to post as HTML form'* or *'URL to post as JSON form'* is enabled in the *'Settings'* tab.

The 'HTML Parameter Name' or 'JSON Parameter Name' stores the values entered on the form in *FormEntry Touch*. The backend developer then parses the form data using this parameter name and handles the appropriate response back to the users when the users post the form to the web service.

You can also review all the parameters at once by open the Parameters window. Simply select View -> Show All Parameters. You have the option of updating the label name, parameter id, json parameter. You have the option of 'Send all' (send the label every time, plus any data), 'Unless has data' (send the label and data only when the control has data), 'Never send' (never send the label and data) for sending Plain Email Text Email, HTML Post and JSON Post. If you have purchased the FTP, WebDAV and Open In Transmittal Plug-Ins, you have the option of restricting the text (.txt) file as well.



The screenshot shows a window titled "Healthcare.formentry - Parameters". It contains a table with 10 columns: Label Name, ID and HTML Parameter Name, JSON Parameter, Plain Email Text, HTML Post, JSON Post, FTP Text, WebDAV Text, and Open In Text. The table lists various parameters such as Female, No, Zip, Birthdate, Spouse, Contact, I agree this info..., Yes, Self, Male, Middle Name, Child, Device UDID8, Yes, Entry ID, CHAMPUS, and Modified. Each row has a checkbox in the first column and a dropdown menu in the last column. At the bottom, there is a section for "Update all parameters at once:" with a dropdown menu set to "Plain Email Text" and a "Send all" button, followed by a "Change All" button.

Label Name	ID and HTML Parameter Name	JSON Parameter	Plain Email Text	HTML Post	JSON Post	FTP Text	WebDAV Text	Open In Text
<input checked="" type="checkbox"/> Female	female	female	Send all	Send all	Send all	Send all	Send all	Send all
<input checked="" type="checkbox"/> No	no_auto	no_auto	Send all	Send all	Send all	Send all	Send all	Send all
<input checked="" type="checkbox"/> Zip	zip	zip	Send all	Send all	Send all	Send all	Send all	Send all
<input checked="" type="checkbox"/> Birthdate	birthdate	birthdate	Send all	Send all	Send all	Send all	Send all	Send all
<input checked="" type="checkbox"/> Spouse	spouse	spouse	Send all	Send all	Send all	Send all	Send all	Send all
<input checked="" type="checkbox"/> Contact	contact	contact	Send all	Send all	Send all	Send all	Send all	Send all
<input checked="" type="checkbox"/> I agree this info...	i_agree_this_information_is_...	i_agree_this_infor...	Send all	Send all	Send all	Send all	Send all	Send all
<input checked="" type="checkbox"/> Yes	yes_employment	yes_employment	Send all	Send all	Send all	Send all	Send all	Send all
<input checked="" type="checkbox"/> Self	self	self	Send all	Send all	Send all	Send all	Send all	Send all
<input checked="" type="checkbox"/> Male	male2	male2	Send all	Send all	Send all	Send all	Send all	Send all
<input checked="" type="checkbox"/> Middle Name	middle_name2	middle_name2	Send all	Send all	Send all	Send all	Send all	Send all
<input checked="" type="checkbox"/> Child	child	child	Send all	Send all	Send all	Send all	Send all	Send all
<input checked="" type="checkbox"/> Device UDID8	device_udid8	device_udid8	Send all	Send all	Send all	Send all	Send all	Send all
<input checked="" type="checkbox"/> Yes	yes_auto	yes_auto	Send all	Send all	Send all	Send all	Send all	Send all
<input checked="" type="checkbox"/> Entry ID	entry_id	entry_id	Send all	Send all	Send all	Send all	Send all	Send all
<input checked="" type="checkbox"/> CHAMPUS	champus	champus	Send all	Send all	Send all	Send all	Send all	Send all
<input checked="" type="checkbox"/> Modified	modified	modified	Send all	Send all	Send all	Send all	Send all	Send all

Parameter Count: 64  
Control Count: 67

Update all parameters at once: Plain Email Text Send all Change All

## CONFIGURATION

Configuration holds other parameters for controls, sections and groups, navigation icons, label names, placeholder text, and enabling drill-downs. Configuration changes based upon the control selected in the form outline.

## NATIVE IOS AND CUSTOM CONTROLS

FormEntry for Mac offers native iOS controls: Text (⌘0) , Text Field (⌘1), On-Off Switch (⌘2), Checkmarks (⌘3), Text Box (⌘4), Date & Time (⌘5), Photo Picker (⌘6), Signature (⌘7), Spin Wheel Picker (⌘8), Spin Wheel Item (⌘9), Location (^0), Hidden (^1), Barcode (^2), Image (^3), Ratings (^4), Web Browser (^5) and PIN Code (^6).

Once the control is added to the Form Outline, you can access it's capabilities in the Control Toolbar Pane. You can place the control in a different place within the form by dragging and dropping the control in the Form Outline.

For web services, all controls output their respective '*Label Name*', '*HTML Parameter ID Name*' or '*JSON Parameter ID Name*' when the form is sent by the user to the web service.

Form control values can be saved to different types of file formats, such as .csv, .xlsx (paid plug-in), .pdf and for image controls .png or .jpg files.

For plain text email, the data is formatted in an email with photos, images and signatures as attachments.

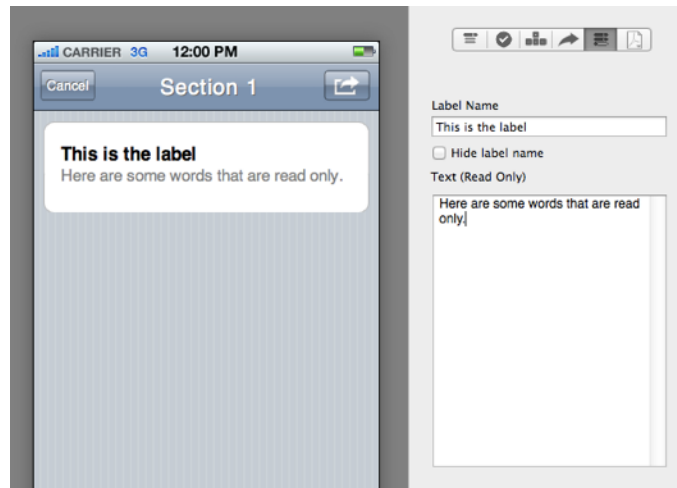
The screenshot shows the 'Control Toolbar Pane' for a text field control. At the top is a toolbar with icons for menu, checkmark, grid, arrow, list, and document. Below the toolbar, the 'Label Name' is set to 'Text Field 2'. There are three checkboxes: 'Enabled on launch' (checked), 'Hide label name' (unchecked), and 'Secure text field' (unchecked). The 'Placeholder Text' is set to 'Required'. There is an unchecked checkbox for 'Bind to Database'. The 'Table ID' is set to '\$table\_id'. The 'Column Sets Value' is set to '\$column\_name'. There is an unchecked checkbox for 'Enable Autocomplete'. At the bottom, there are three circular icons with checkboxes: a blue arrow icon for 'Display in Sent List' (unchecked), a red circle icon for 'Display in Drafts List' (unchecked), and a red trash can icon for 'Display in Trash List' (unchecked).

## TEXT (§0)

The text control allows you to add read only text, such long paragraphs. It can also provide you with a label or or you can hide it.

If you need large blocks of read only text, the Text Control is the one to use.

The label for the text field can be hidden by checking '*Hide label name*' in the '*Configuration*' inspector.

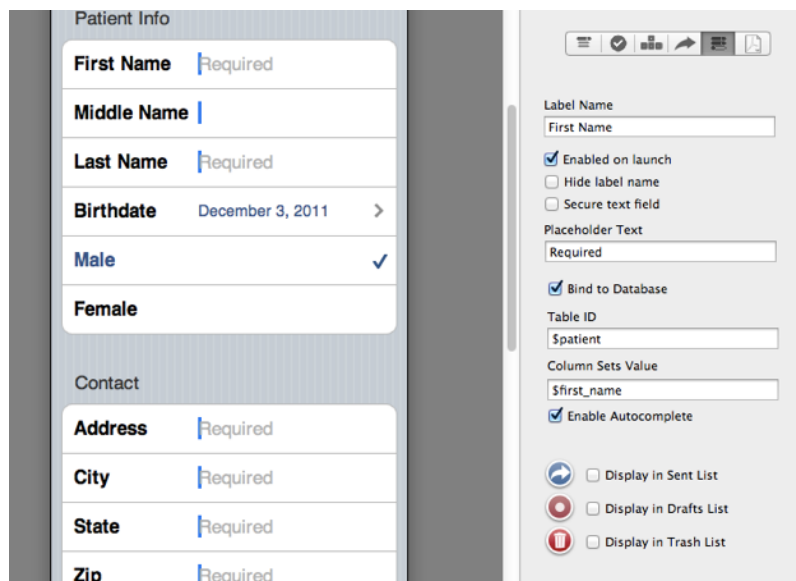


## TEXT FIELD (§1)

The '*Text Field*' control handles small amounts of editable text. It can be configured with different default keyboards that appear when the user taps the control. Select a standard 'QWERTY' keyboard or a numeric keyboard in the '*Initial Keyboard Type for Input*'.

The '*Text Field*' can apply validating rules if the '*Required Field*' in the '*Validation Rules*' is selected in the inspector.

The label for the text field can be hidden by checking '*Hide label name*' in the '*Configuration*' inspector. Another option is to check the '*Secure text field*'. As the characters are typed, they are replaced with dots (e.g. a password field).





The text field can also be bound to a database table's column and can be enabled with autocomplete. Once the user starts typing within the text field, the control will look up within the database the list of items in bound column.

To bind the database to database table's column, select the "Bind to Database" checkmark and place the Table ID and Column ID within the bound fields using the "\$" as the prefix. For example, this text field "First Name" is bound with \$patient table and \$first\_name column.

You will, of course have to have a database table with the Table ID of "patient" and a Column ID of "first\_name".

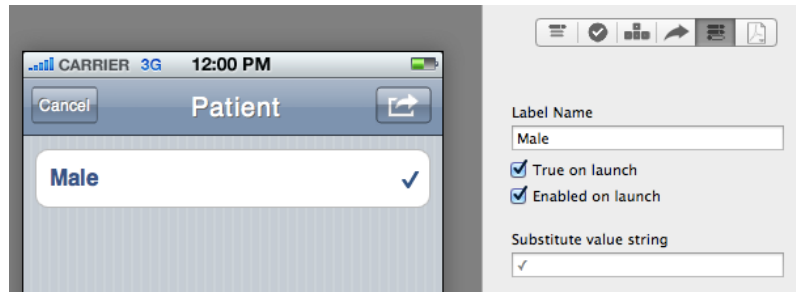
## ON - OFF SWITCH (§2)

The 'On-Off Switch' provides a true/false toggle in the form. It can provide a 'true' or 'false' by sending either an 'On' or 'Off' value. The default state can be set using the 'True on launch' option under 'Configuration'.

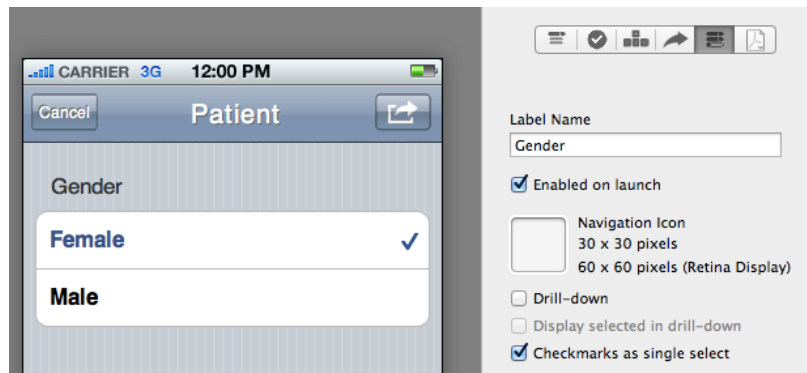
This control also allows you to add custom values and icons. If you would like to add a Yes or No, or even an Up or Down values, simple enable the "Use alternate values & custom icons" checkbox. You can add the values and custom icons to the appropriate tabs.

## CHECKMARK (§3)

Much like an *'On-Off Switch'*, the *'Checkmark'* allows the user to set a true or false value depending on a checkmark's visibility. The default state can be set using the *'True on launch'* option under *'Configuration'*. If *'True on launch'* is selected, a checkmark is visible when the control appears for the first time.



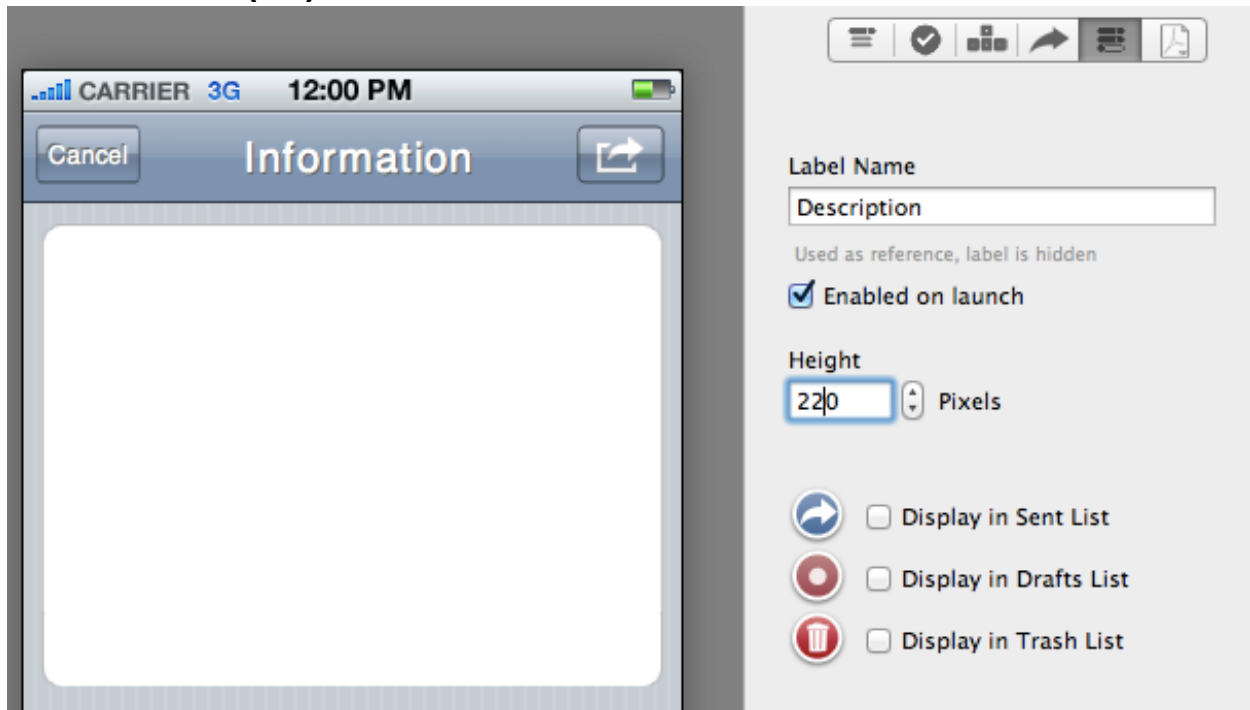
FormEntry for Mac also handles grouped information for checkmarks. Put the checkmarks inside a grouping and enable the *'Checkmarks as a single select'* in the group's *'Configuration'* inspector. The form allows only one checkbox in the grouping to be selected at a time. This behavior mimics a drop-down menu or radio button.



Checkmarks post back to web services or email the default values as *'Checked'* if the control has been enabled. If the checkmark is not enabled, the value is empty.

You can also override the default *"Checked"* value by substituting a text string or character, i.e. symbols, checkmarks, X's, etc. Add the text string or character under *"Substitute value string"* at the input field.

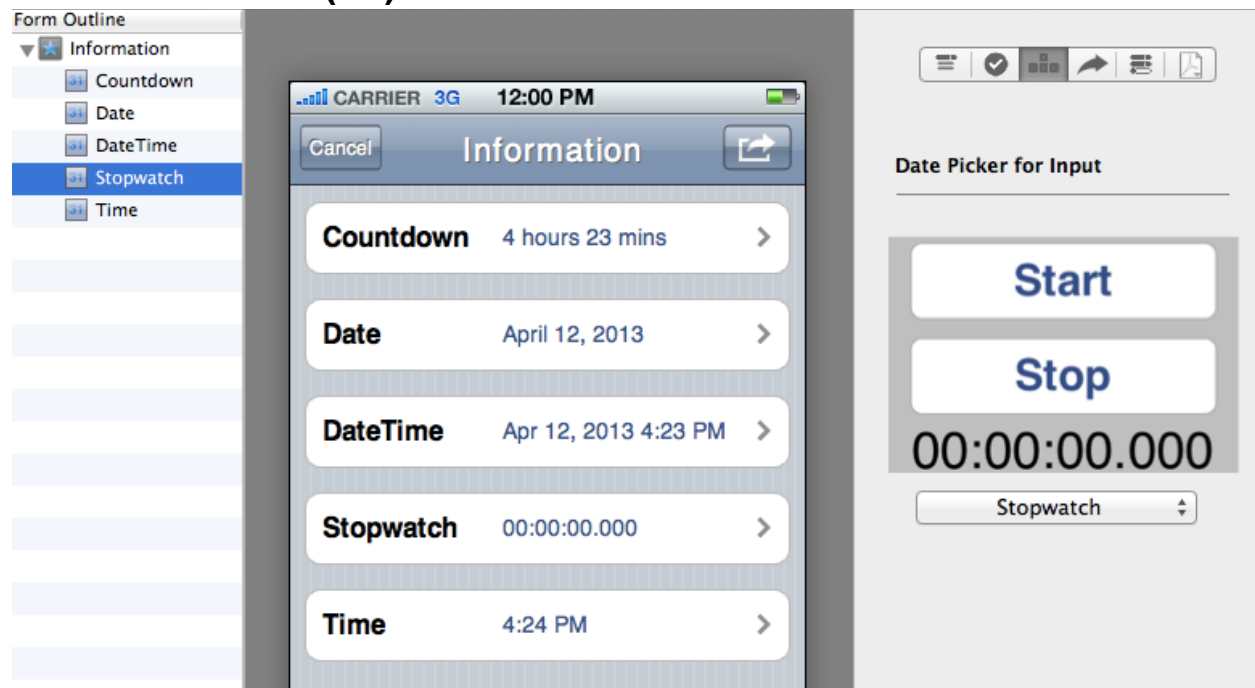
## TEXT BOX (§4)



The '*Text Box*' control hold larger amounts of editable text. The text can be added in a multiline format. By default, the Label name is never displayed. If a label is required for a text box, place the text box in a grouping. The grouping's '*Label Name*' acts as the '*Text Box*' label.

If you would like have more height for text input, you can adjust the height of the text box.

## DATE & TIME (§5)

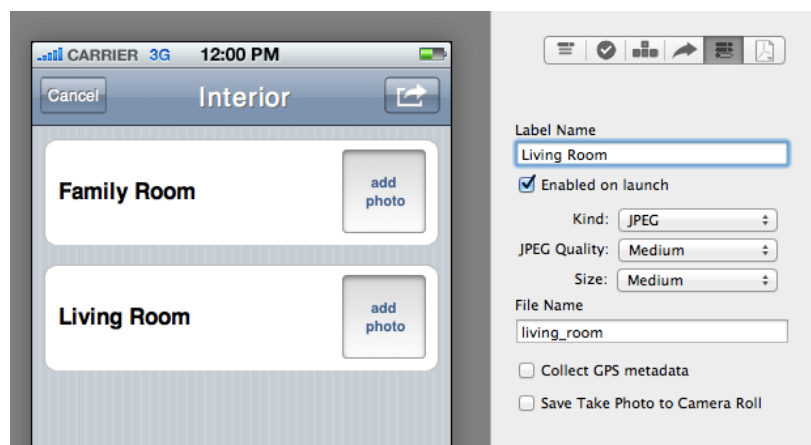


The 'Date & Time' control provides four ways to collect date & time information: 'Countdown', 'Date', 'Date & Time', 'Stopwatch' and 'Time'.

Each of the 'Date and Time' controls sends information differently. The 'Countdown' sends time in hours and minutes; 'Date' sends month, day, year; 'Date & Time' sends month, day, year, hours, minutes, and AM or PM; 'Stopwatch' allows you to record a time from the moment the user starts and stops the watch; 'Time' sends hours, minutes, and AM or PM.

## PHOTO PICKER (§6)

The Photo Picker control allows the user to include photos from device's 'Photo Library' application in a form. If the form is being used on an iPhone or iPad, the user has the option of using the built-in camera to take a photo.



Users can remove any photos they have added by selecting *'edit'* on the thumbnail icon and then selecting *'Delete Photo'*.

Select the type of photo when configuring the photo picker under the *'Kind:'* pull-down menu in the *'Configuration'* section of the *'Control Tool Bar Pane'*. The choices are *'JPEG'* or *'PNG'*. If *JPEG* is selected the image quality under the *'JPEG Quality'* pull-down.

#### TAKING PHOTOS FROM THE CAMERA

When you select JPEG as an option, you also have the ability to collect GPS metadata, the latitude and longitude coordinates of where the photo was taken from the camera. PNG file format does not have this option. Typically you can parse this information or view it through a photo viewing software.

Also, you as the form author, you can have the photos taken by the user, to be store in the iOS camera roll. If you do not want the user to have a photos of your form within their photo album, uncheck the **“Save Take Photo to Camera Roll”**

#### PHOTO SIZE

The *'Size'* option, which determines the size of the file sent to the web service, is available to either type of file format. Selecting a larger size means a larger the file is sent to the web service. This has an impact on transmission performance while sending forms. **Note:** *Form users do not have the option of changing these settings.*

#### HOW PHOTOS ARE SENT

If the form is configured to return form data in an email, the photo selected by the user is sent as a standard email attachment using the title specified in the *'File Name'* field. There is no need to specify a file extension. FormEntry Touch appends the appropriate extension automatically.

If the form is configured to send form data to the web service in an HTML post, the form is posted as a standard HTTP *'Post'* using a multipart form mechanism. The incoming data must be parsed as a standard form post from a web browser. Specify the *'HTML Parameter Name'* for the photo under the *'Web Services'* tab in the *'Control Tool Bar Pane'*.

If JSON is being to send form data, the form is posted as a standard HTTP post. However, any photo is converted to a BASE64 string. Specify the *'JSON Parameter Name'* for the photo under the *'Web Services'* tab in the *'Control Tool Bar Pane'*.

The following snippets demonstrate the values sent from a 'Photo-Picker' when posting back to an email, or a JSON web service. If the form is being sent as a HTML post, a standard multipart form post is used. Therefore, a snippet example is not shown here.

## SIGNATURE (§7)

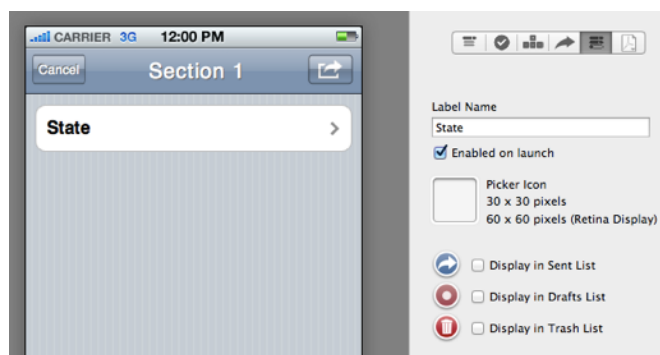


The Signature control allows the user to sign a form by simply drawing their signature with their finger.

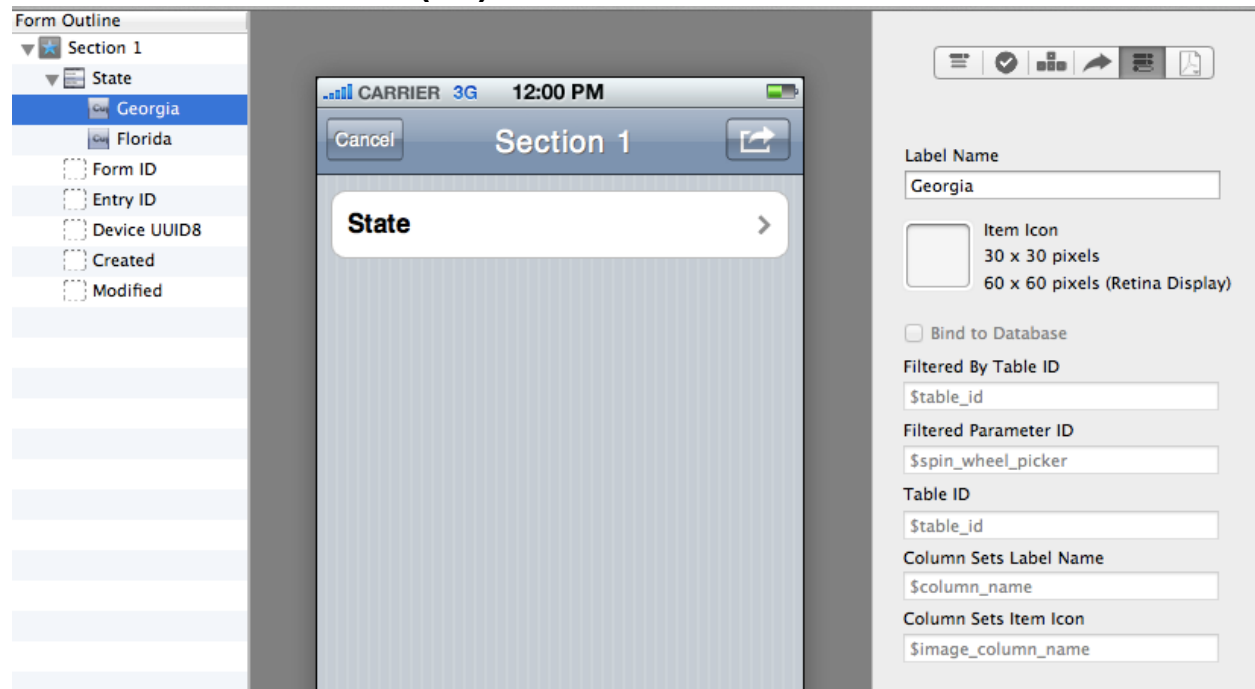
The signature data is stored and sent as a PNG file. The posting of the signature to URL works the same way as posting a photo from the Photo Picker. For example, when emailing, the signature is attached as a PNG file or as an image within the PDF attachment.

## SPIN WHEEL PICKER (§8)

The Spin Wheel Picker control allows the user to select from a list of items. Each control can hold an unlimited number of Spin Wheel Items. Once a user selects the item, the label name value is set to the Spin Wheel Picker.



## SPIN WHEEL ITEM (§9)



Spin wheel items are controls embedded within the Spin Wheel Picker. These items are raw text strings that are assigned to the Spin Wheel Picker when the item is selected by the user. A 30 x 30 or 60 x 60 (Retina Display) pixel icon can also be added for each item.

The selected value will be set to the parent Spin Wheel Picker. If you want to do any Rules against the Spin Wheel Item, you will need to use the param id of the parent Spin Wheel Picker.

The Spin Wheel Item can be bound to the database table's column. To bind the Spin Wheel Item to a database table's column, only 1 Spin Wheel Item is needed. This 1 Spin Wheel Item can then be bound by select "Bind to Database".

The Table ID of the table to "Table ID" and the Column ID of the bound value to "Column Sets Label Name". If you would like to bind the icon of the Spin Wheel Item, you can bind the "Column Sets Item Icon" to the Column ID of a "image" data type column.

## LOCATION (^0)

The image shows the FormEntry interface for the Location control. On the left, a simulated mobile device screen displays a form section titled 'Section 1'. At the top of the screen are status indicators for 'CARRIER 3G' and '12:00 PM'. The form contains a 'Location 3' control showing the coordinates '33.775150, -84.369221' and a 'State' control with a right-pointing arrow. On the right, the configuration panel for the Location control is visible. It includes a toolbar with icons for menu, checkmark, data, share, and print. The configuration options are: 'Label Name' (text field with 'Location 3'), 'Enabled on launch' (checked checkbox), 'Desired Accuracy' (dropdown menu set to 'Best'), 'Refreshable location' (unchecked checkbox), and three unchecked checkboxes for 'Display in Sent List', 'Display in Drafts List', and 'Display in Trash List'.

Location gathers the latitude and longitude coordinates of the user's device. When the user selects the control, the coordinates are collected.

If you want the user to have the ability to refresh their location, you can select the Refreshable Location checkbox. If not, have this unchecked, which will allow the user to only select the control once.

For most locations, your desired accuracy should be set to best.

You can also use the Rule Builder to perform calculations or enable/disable controls using the Location services. For example, you can enable controls if the user is standing within 10 meters of a specific GPS location.

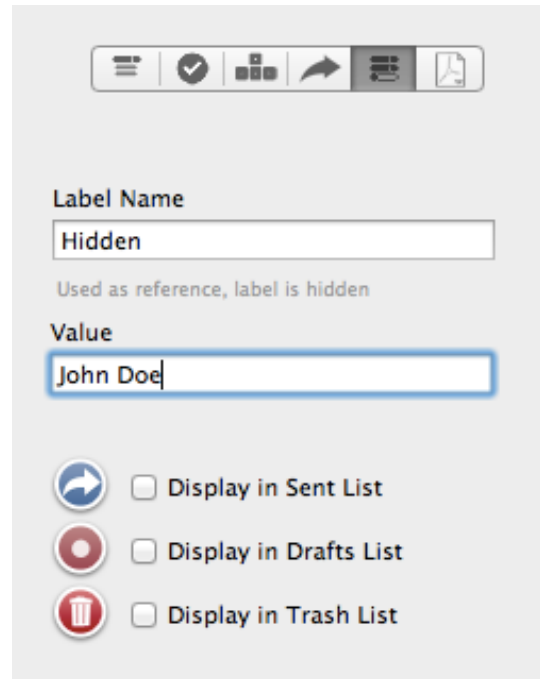


## HIDDEN (^1)

Hidden controls are place holders controls that are invisible to the user. If you like to store values in hidden fields and have that data be sent back or to be used in the Rule Builder, this would be an excellent place to store values.

Note, there are five key parameter values that are stored by default with FormEntry, “form\_id”, “entry\_id”, “device\_uuid8”, “created” and “modified”, which are all hidden fields. These parameters will become populated with the device information.

From the Web Service tab, you can disable the Keyword Parameter Names for the five default key parameters. By disabling these key parameters, you basically will turn them into a normal hidden field.

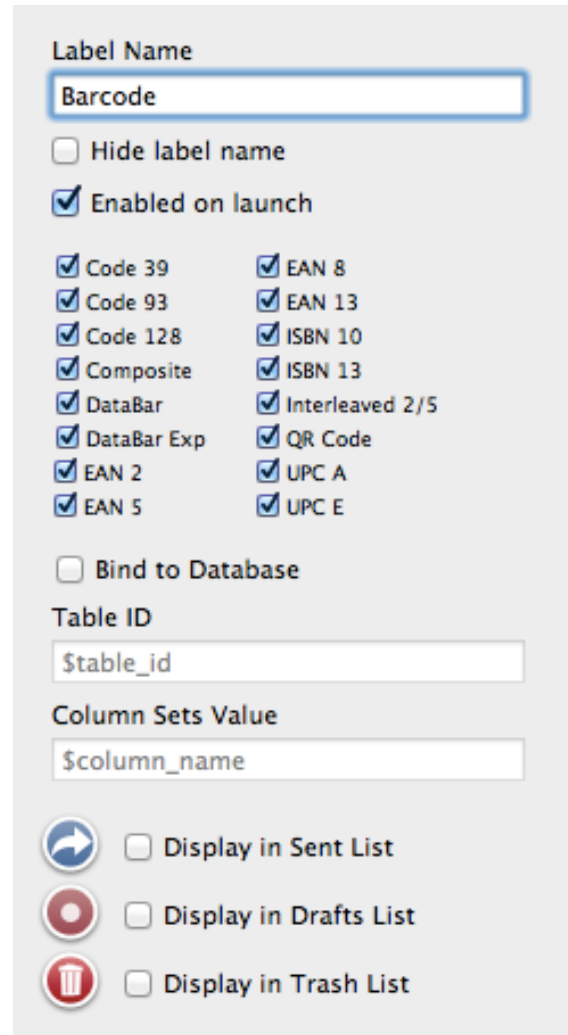


The screenshot shows the FormEntry configuration interface. At the top is a toolbar with icons for menu, checkmark, grid, arrow, and document. Below the toolbar, the 'Label Name' field is set to 'Hidden'. A note below it states 'Used as reference, label is hidden'. The 'Value' field is set to 'John Doe'. At the bottom, there are three options with icons and checkboxes: 'Display in Sent List' (blue arrow icon), 'Display in Drafts List' (red circle icon), and 'Display in Trash List' (red trash icon). All checkboxes are currently unchecked.

## BARCODE (^2)

Once installed you can add barcode scanning to your form apps. Using the iOS device's camera, you will be able to read the following barcodes: Code 39, Code 93, Code 128, Composite, DataBar, DataBar Exp, EAN 2, EAN 5, EAN 8, EAN 13, ISBN 10, ISBN 13, Interleaved 2/5, QR Code, UPC A and UPC E.

You can also bind the values of the scanned barcode to a database table's column. This is useful if you would like to perform a look up on a product and populate additional fields from the database based upon the scanned barcode.



The image shows a configuration interface for a barcode scanner. It includes a text field for 'Label Name' with the value 'Barcode'. Below this are two checkboxes: 'Hide label name' (unchecked) and 'Enabled on launch' (checked). A grid of 16 checkboxes follows, all of which are checked, representing various barcode types: Code 39, Code 93, Code 128, Composite, DataBar, DataBar Exp, EAN 2, EAN 5, EAN 8, EAN 13, ISBN 10, ISBN 13, Interleaved 2/5, QR Code, UPC A, and UPC E. Below the grid is a checkbox for 'Bind to Database' (unchecked). Underneath are two text fields: 'Table ID' with the value '\$table\_id' and 'Column Sets Value' with the value '\$column\_name'. At the bottom, there are three circular icons with corresponding checkboxes: a blue circle with a white arrow (unchecked) for 'Display in Sent List', a red circle with a white dot (unchecked) for 'Display in Drafts List', and a red circle with a white trash can (unchecked) for 'Display in Trash List'.

Label Name  
Barcode




☐ Hide label name  
☒ Enabled on launch

<input checked="" type="checkbox"/> Code 39	<input checked="" type="checkbox"/> EAN 8
<input checked="" type="checkbox"/> Code 93	<input checked="" type="checkbox"/> EAN 13
<input checked="" type="checkbox"/> Code 128	<input checked="" type="checkbox"/> ISBN 10
<input checked="" type="checkbox"/> Composite	<input checked="" type="checkbox"/> ISBN 13
<input checked="" type="checkbox"/> DataBar	<input checked="" type="checkbox"/> Interleaved 2/5
<input checked="" type="checkbox"/> DataBar Exp	<input checked="" type="checkbox"/> QR Code
<input checked="" type="checkbox"/> EAN 2	<input checked="" type="checkbox"/> UPC A
<input checked="" type="checkbox"/> EAN 5	<input checked="" type="checkbox"/> UPC E

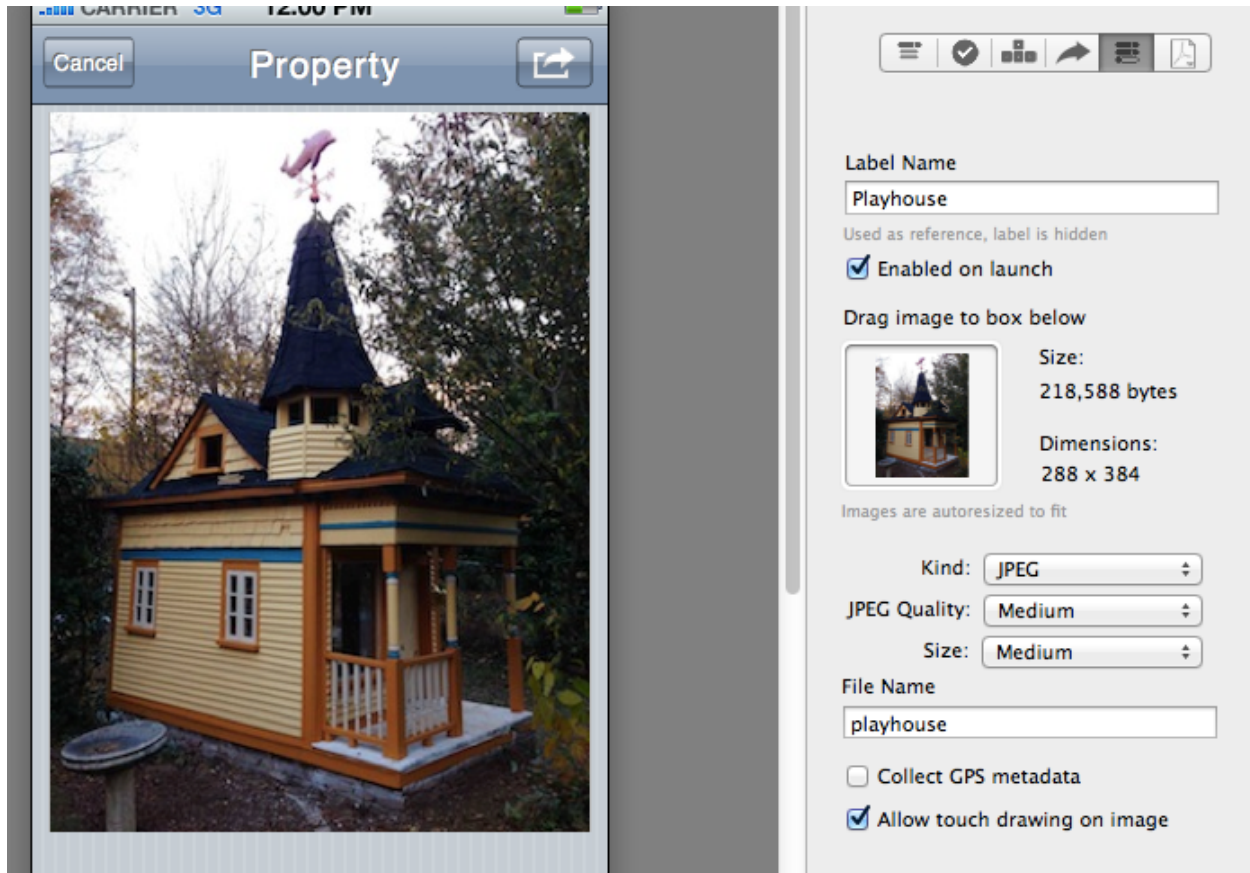
☐ Bind to Database

Table ID  
\$table\_id

Column Sets Value  
\$column\_name

 ☐ Display in Sent List  
 ☐ Display in Drafts List  
 ☐ Display in Trash List

## IMAGE (^3)



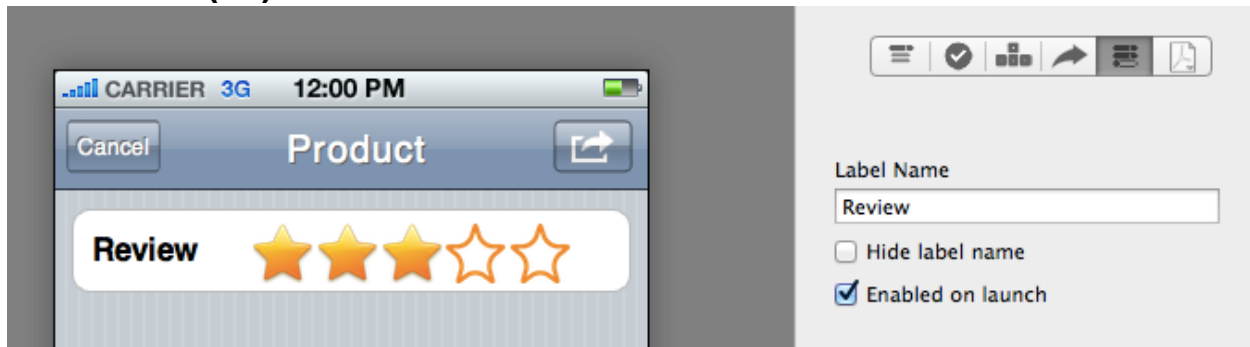
The Image control allows you to place an image within your form app. The image will auto adjust to a full size image to fit on the width under the iPhone or iPad.

You can drag any JPEG or PNG file into the image well. It is HIGHLY recommended you use the smallest file size possible. The above image is roughly 218K in file size and is 288 x 384 in resolution. It could be possible to compress this file even further and the quality would be roughly the same on the iOS devices.

Within the Image control, you have the ability to send the image back to you either as a JPEG or PNG file. If you choose JPEG, you have further options of compressing the image when sending it back as well as collecting GPS and any touch drawings.

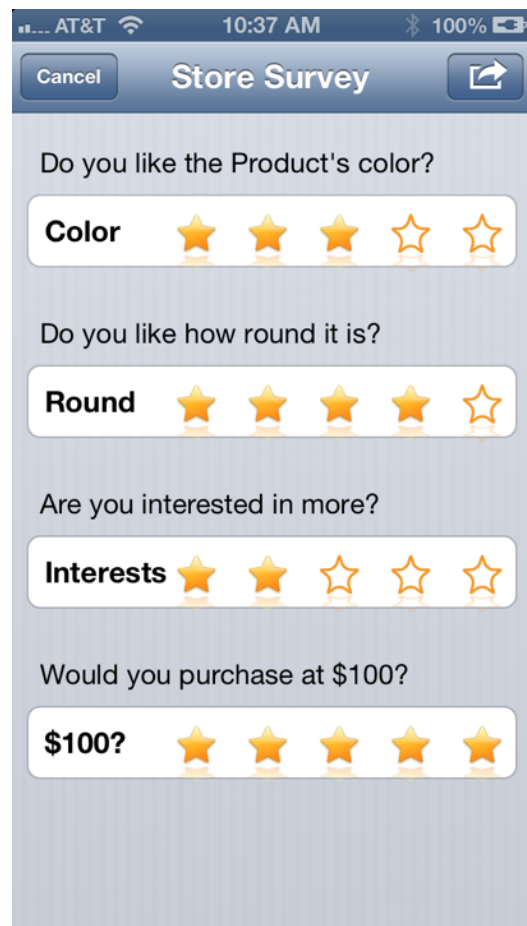
The Image control allows your users to draw with their finger or stylus on your image. When the image is sent back, the drawing is mapped to your image.

## RATINGS (^4)



The Ratings control allows your users to rate from 0 to 5 stars on the iOS device. The iOS user can select a star by simply touching the star. To select no stars, they can simply touch the label or to the right of the first star.

The Ratings control is ideal for ad hoc retail surveys, questionnaires, classroom and reviews.



## WEB BROWSER (^5)

The Web Browser control allows you to add a live web landing page to your form apps. The end user does not have access to change the URL, they can only simple navigate through the URL you have provided.

An Internet connection (Wi-Fi or Cellular) is required for the web page to load.

This plug-in embeds the full functionality of Mobile Safari on the iOS device. The end user can go back and forward, refresh the page and navigate back to your form app.

Ideal for learning resources, links to videos, audio files, PDFs, anything external that you want your users to gain access to.



## PIN Code (^6)

The PIN Code control allows you to lock your forms or different Sections or drill-down groups (Current Screens) with a 4-digit PIN code (passcode).

Your users must successful enter the PIN Code to advance in either loading the form app or getting access to different areas of your form app.

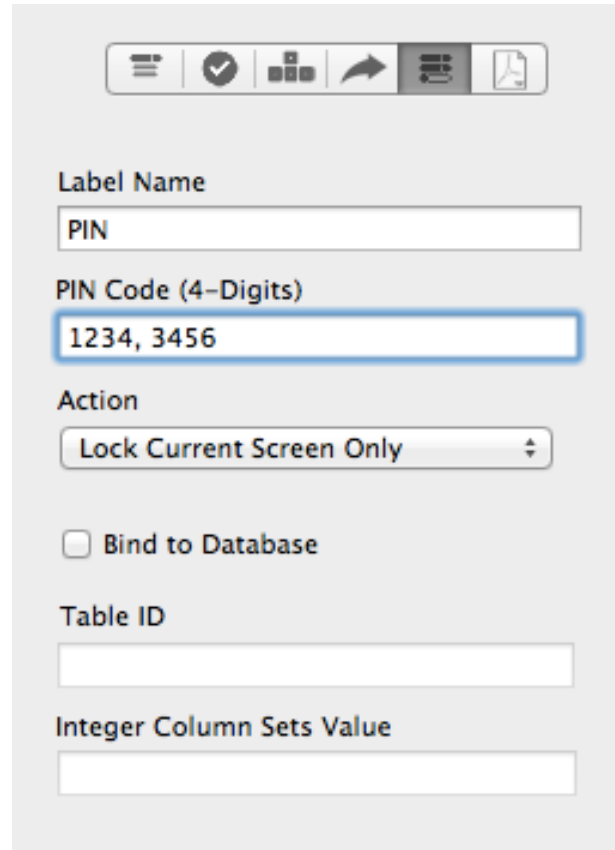
You can capture the user's 4-digit number when the form is sent by selecting either "Lock Current Screen Only" or "Lock Current Screen & Form". If you have selected "Lock Form Only", no value will be collected however, your form cannot be accessible without the correct PIN Code.

If you want to set multiple PIN Codes, you can simply add the 4-digit numbers separated by a comma, i.e. "1234, 3456".

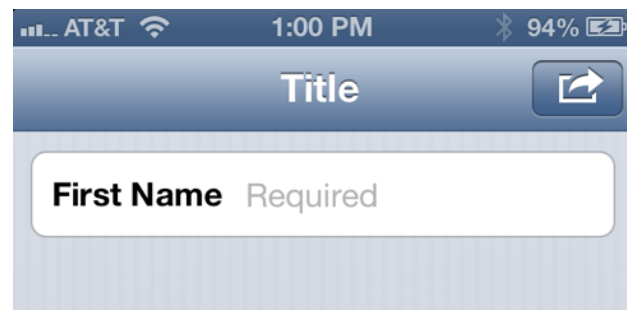
This control can be bound to a database table's column.

*Important configuration to note*, if you had selected to not show the "Cancel" button within your form app and choose to use the "Save" plugin, there would be no way to get back to the current form screen. Your form would constantly be in a New form state. To get around this, the PIN Code control offers the ability to jump back to the current form screen by pressing the Title for 5 seconds.

To enable the 5 second PIN Code recognition the Action must be either "Lock Current Screen & Form"s or "Lock Form Only" selected.



The screenshot shows a configuration window for a PIN Code control. At the top is a toolbar with icons for menu, checkmark, grid, arrow, list, and document. Below the toolbar, the "Label Name" field contains the text "PIN". The "PIN Code (4-Digits)" field contains the text "1234, 3456" and is highlighted with a blue border. The "Action" dropdown menu is set to "Lock Current Screen Only". There is an unchecked checkbox for "Bind to Database". The "Table ID" field is empty. The "Integer Column Sets Value" field is also empty.



The screenshot shows a mobile form app interface. The status bar at the top displays "AT&T", signal strength, Wi-Fi, time "1:00 PM", Bluetooth, and battery level "94%". The app has a blue header bar with the title "Title" and a share icon. Below the header is a form field labeled "First Name" with the text "Required" next to it.

# Settings

The Settings sheet offers the ability to configure your project. From here, you can sent specific information about your project. Configuring your project setting is important part of how your form app will behave once installed on your user's iOS devices.

## GENERAL INFO

From the General Info tab you can configure the Form Name, Version number as well read the Form ID provided by FormEntry for Mac. It also provides the option to expire the form after a time limit when the user leaves the paired Wi-Fi network of the Mac.

The screenshot shows the 'General Info' tab of the FormEntry settings interface. At the top, there are five tabs: 'General Info' (selected), 'Appearance', 'Sending Form', 'Author Info', and 'DSV and PDF Settings'. Below the tabs, the 'Form Name' is 'Home Inspection', 'Version' is '1.0', and 'Form ID' is 'E3F90CA3A3'. There are three checkboxes: 'Automatically check verification Web URL for latest Form Version.' (unchecked), 'Expire this form when user leaves this paired Wi-Fi network of this Mac.' (unchecked), and 'Enable Auto Increment for this form on all New Entries.' (unchecked). The 'Web URL' field contains 'http://www.example.com/CheckVersion.php'. The 'Minutes to expire this form after receiving from paired Wi-Fi network of this Mac.' is set to '1'. The 'Start Number' is '0'. A 'Close' button is at the bottom right.

Form Name	Version	Form ID
Home Inspection	1.0	E3F90CA3A3

☐ Automatically check verification Web URL for latest Form Version.

Web URL:

☐ Prevent user from loading a New Form if a newer Form Version exists.

---

☐ Expire this form when user leaves this paired Wi-Fi network of this Mac.

☐ 1 Minutes to expire this form after receiving from paired Wi-Fi network of this Mac.

---

☐ Enable Auto Increment for this form on all New Entries.

Start Number:  ☐ Reset Auto Increment with this Start Number.

---

☐ Enable Debug Mode on iOS device to show additional Warnings and Messages alerts.

Close

You can also configure whether your form app should check a web service for newer versions, see the Check Form Version section for details.

You can also set an auto increment value for your form app and you can specify the start number. When the form is delivered for the first time to an iOS device, it will start



with the Start Number. If you deliver an update to your form app, it will not reset the number on the device, unless you have “Reset Auto Increment with this Start Number” set.

Lastly, you can also enable a debug mode on the iOS device as you test your form app. This is handy when testing your form app with an attached database.

Under your form app if you navigate to the About screen, you will see Debug Mode - View Log. This log contains the history of your form. The information that is recorded under the log will help you track down issues that come up when doing math, rules, posting form data to websites, etc.

The log file is also accessible under iTunes if you Sync your iOS device to your computer. From the Apps tab under File Sharing, you can save the form’s logs to your computer to be opened in a text editor.

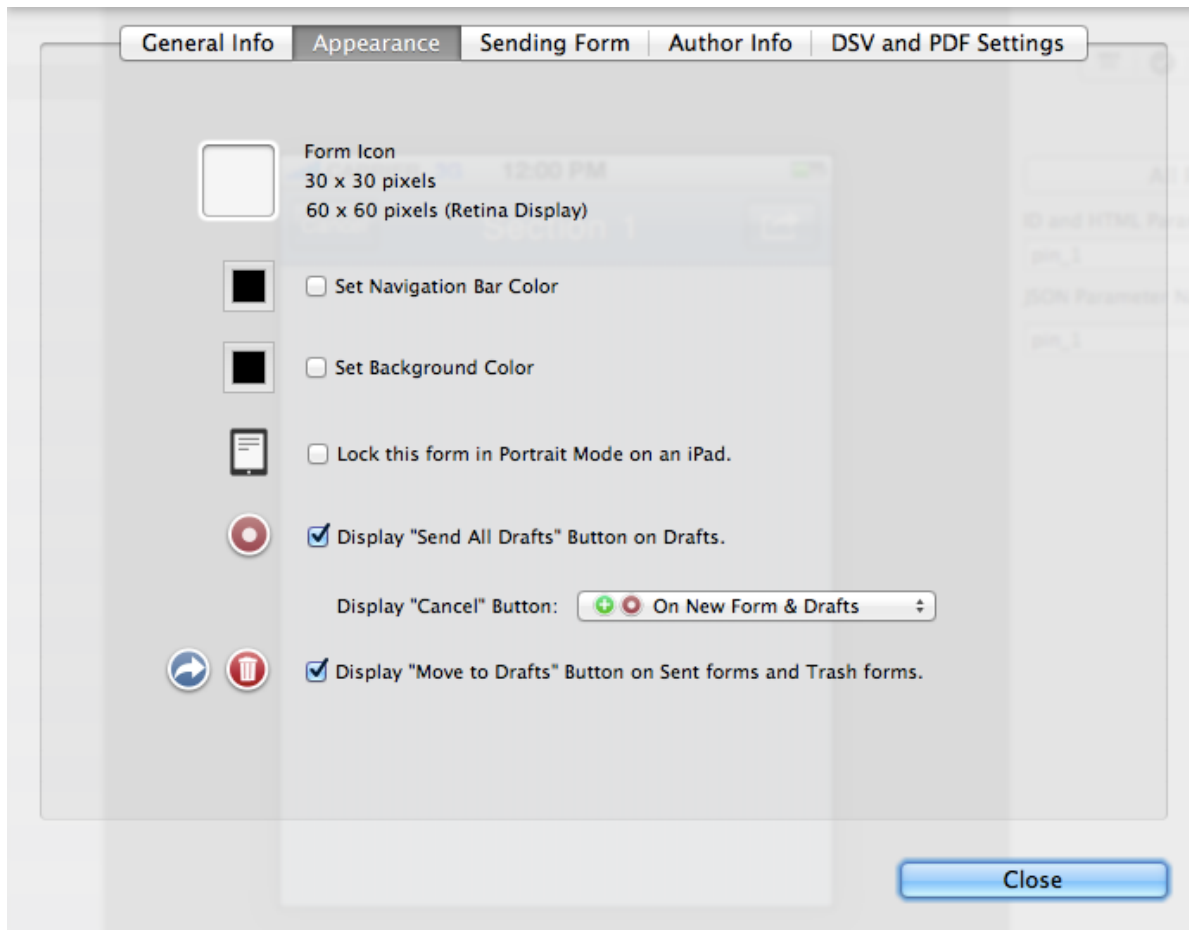




## APPEARANCE

The Appearance tab allows you to configure your Form Icon using a 30 x 30 or 60 x 60 (Retina Display) icons. You can also set the navigation and background colors of your form app.

On iPads, you can lock your form app in Portrait mode. This is handy if you do not want your users to have access to Drafts, New, Sent, Trash folder and this mode works great with the Save and PIN code Plug-ins.



Also, you can decide if you want to have the action buttons displayed on the New Form, Drafts, Sent and Trash screens.

## SENDING FORM

Sending the Form data is probably the end goal of collecting the information on your iOS devices. You want the ability to process, review, perform analytics, collect and store this information. FormEntry provides eight different methods for sending your data:

Email, FTP, HTML Post, JSON Post, Open In, Printer, Save and WebDAV Transmittal Plug-Ins.

The screenshot shows a settings window titled "Transmittal Plug-Ins" with a tabbed interface. The tabs are "General Info", "Appearance", "Sending Form", "Author Info", and "DSV and PDF Settings". The "General Info" tab is active. It contains two main sections: "Email Transmittal Plug-In" and "FTP Transmittal Plug-In".

**Email Transmittal Plug-In:**

- ☒ Email Transmittal Plug-In (with an envelope icon)
- Default Email Address:
- Button Label:
- Default Subject Line:
- Display Button: ☒ On New Form & Drafts (with a dropdown arrow)
- Transmit Format: - File Name:

**FTP Transmittal Plug-In:**

- ☐ FTP Transmittal Plug-In (with an upload icon)
- FTP or SFTP: - Button Label:

At the bottom of the window, there is a radio button (currently unselected) and a checkbox labeled "After sending form, keep copy in the draft folder." A "Close" button is located at the bottom right.

## AUTHOR INFO

The form Author Info tab holds the metadata about your form, such as the Author's first and last name, email, company information and form description.

This information is used within FormEntry Touch under the about section as well as provides further information in the Projects Window if placed within the Projects Folder.

General Info | Appearance | Sending Form | **Author Info** | DSV and PDF Settings

First Name:  Last Name:

Email:

Web Site:

Company Name:

Address:

City:  State:

Zip/Postal:  Country:

Form Description:

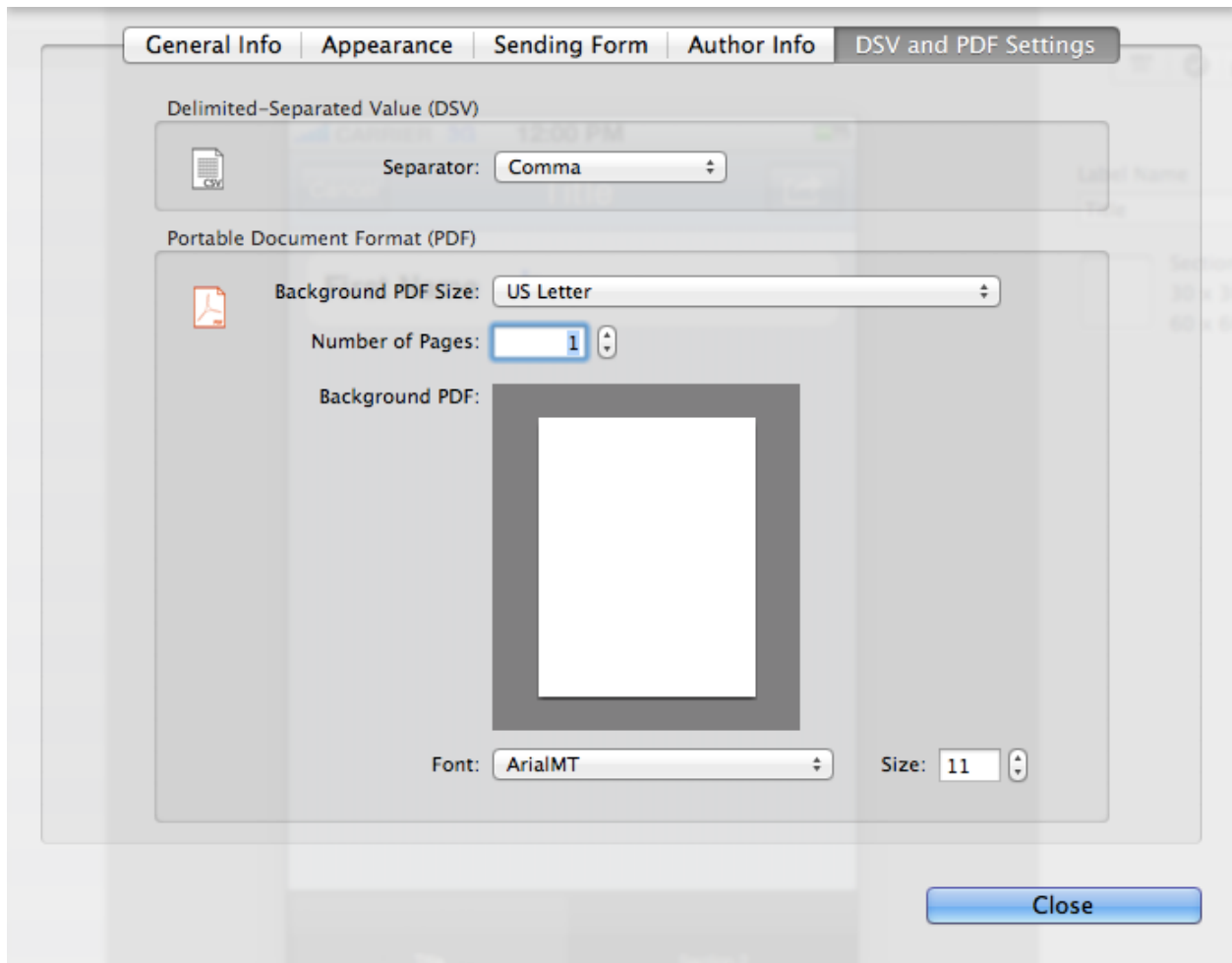
Different parts of the Author Info can be used within your form apps using the reserved form parameter ids, see the section of Reserved Form Parameter IDs.

## **DSV AND PDF SETTINGS**

DSV and PDF Settings tab allows you to configure the .csv and .pdf transmit files.

For Delimited-Separated Value (DSV) files, they will always use the .csv file extension and you can configure it to separate your values either comma, tab or semicolon.

For Portable Document Format (PDF) files, you can configure the file size, number of pages, globally change the font and font size at once. If you want to use your own custom PDF file, you can import that background PDF here. You will need to layout the values on top of your PDF through the PDF editor.



See [Using PDF Technology](#) for further readings on how to layout your values.



# OVERVIEW OF DATABASE

**This section describes how to add a local RDMS database to your form app, using key paths for accessing database values and updating the form's local database remotely from a web service.**

FormEntry for Mac allows you to add a database to your form app. This custom object database that can be used in conjunction with the Rule Builder and with bindings to bindable controls. You can also update this database remotely when the user refreshes the form's database.

There are a few limitations to the database that you will need to know before setting up your database. 1) The database is limited to a Read Only database, we will further explain what this means later in this chapter. 2) You are limited to 9 different data types for your columns. These data types are the typical standard data types as well as some custom types that are designed to work well with FormEntry Touch. 3) You are allowed only a limited number of columns per data type. 4) You do not have SQL access to the database, however, you can update the database via a web service.

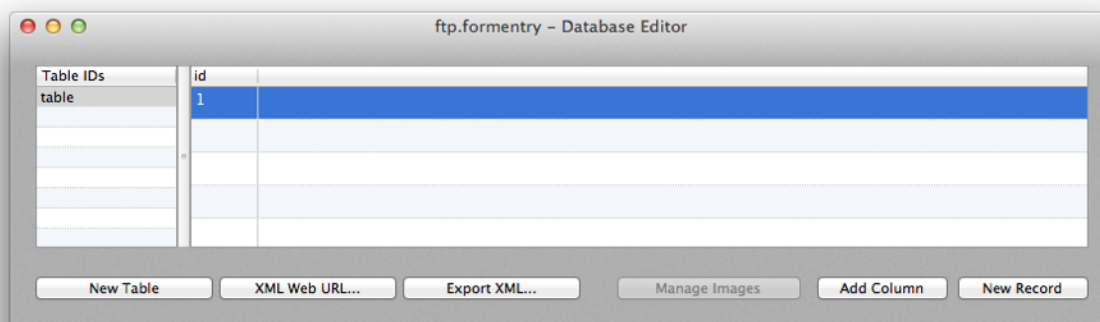
The overall theme for FormEntry Touch's database is to think "small", the smaller the data sets per table the better. And to think of this database as a cached read-only data. Remember that once the form is installed within FormEntry Touch, it can fully operate without an Internet connection. So the database needs to be local and independent of the Internet.

Once the device is on the Internet, there is the opportunity to update the "cache" local database via a web service. You can update one table or a combination of multiples tables at a time, however, you should keep in mind when updating via a web service, you will be "reinstalling" the entire table(s), it's not a true update.

## Setting up your Database

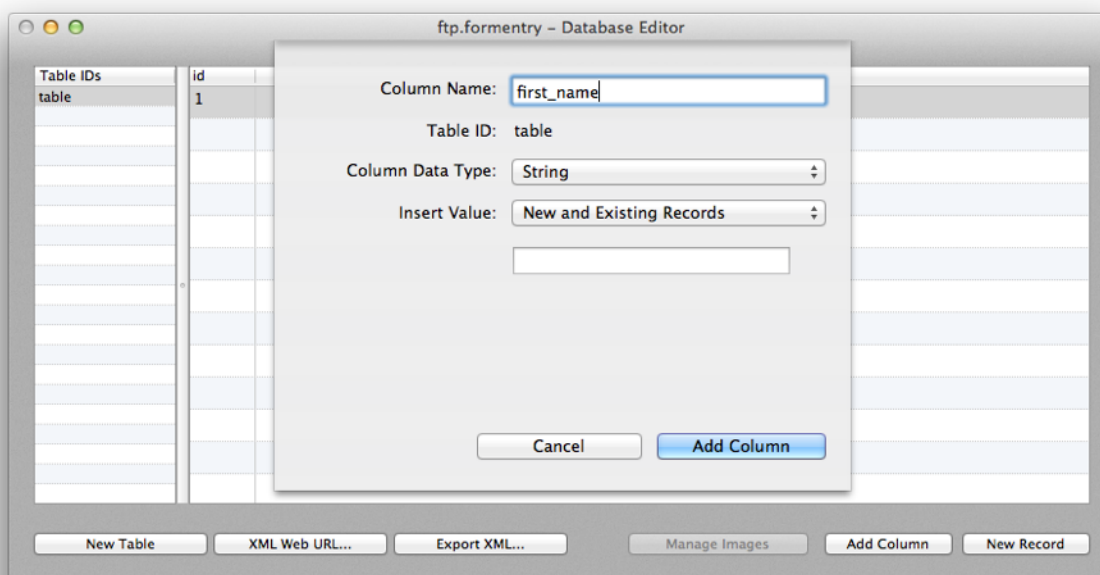
FormEntry for Mac allows you create a database by setting only one table. Once at least one table has been created, your database is set. You can add an unlimited number of tables to your database.

From the Main Menu, select View -> Show Database to bring forward the Database Editor. Select New Table button to add a new table. Rename the table with a new table ID if need be by double-clicking on the 'table' name within the table row. By default 'table' is used and all table IDs must be unique.



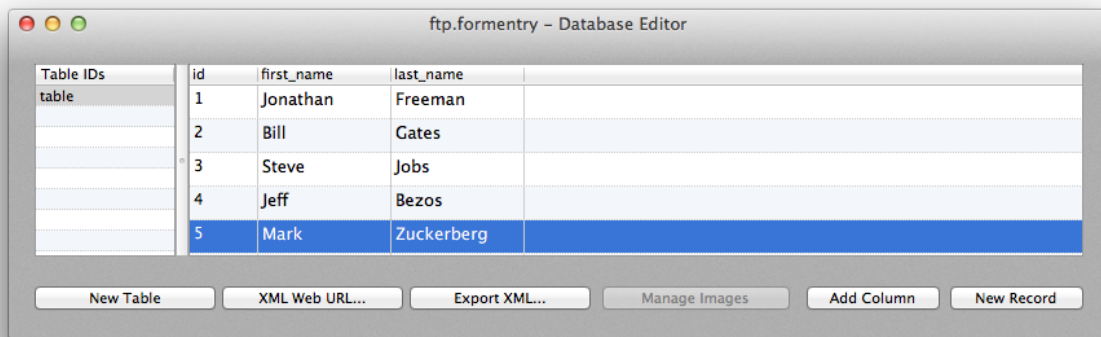
Once you add your first table, you will notice the first record has been created with a record ID of "1" as a default .

To add a new column to your table, select the Add Column button. Choose a Column Name for your table and select its Column Data Type.



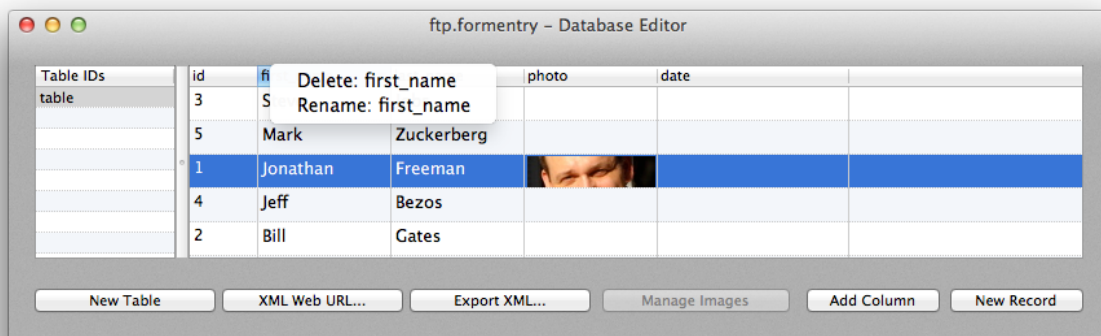
You can also select a default Insert Value when adding New or Existing Records. To add additional records, you will need to select the New Record button and a new record will be added to the selected table.

If you had placed a default Insert Value for New Records, that default value will populate all the records within that column.



To remove a table or record, simple select the row in the table and press the Delete Key on your keyboard.

To remove or rename a Column, right click on the Column ID and select Delete or Rename.



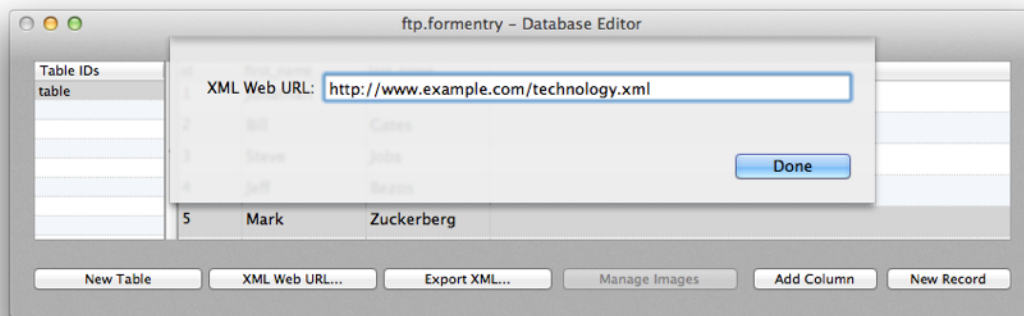


## UPDATING DATABASE VIA A WEB SERVICE

FormEntry for Mac provides you with the ability to update the database remotely. You can update all the tables at once or one table at time once the iOS device is on the Internet. You are responsible for maintaining the web service at your web server or cloud based account.

### SET XML WEB URL

To enable the web service, you must provide a valid Web URL by choosing the XML Web URL button. This fully qualified URL will receive a POST request from the iOS device, if 1) the device is on the Internet and 2) the iOS user requests the database to be updated. The iOS user must initiate the request.



## USER INITIATED POST REQUEST

When the iOS user is at your form app's current form screen (New, Sent, Drafts, Trash and About), they can choose to update the database by pulling down this screen for a refresh.

If the iOS device is connected to the Internet, FormEntry Touch will post to your URL with a standard HTTP Post the following parameters to your web service URL, "device\_uuid8", "form\_version", "form\_id", "table\_id[]", "table\_updated[]" and if installed the "pin\_code" (based upon the parameter id of the PIN Code) parameter id.

The "device\_uuid8" is based upon the Device UUID8 from the device's Settings screen within FormEntry Touch.

The "form\_version" is the version of your form, originally assigned from the Version field on the General Info tab within FormEntry for Mac.

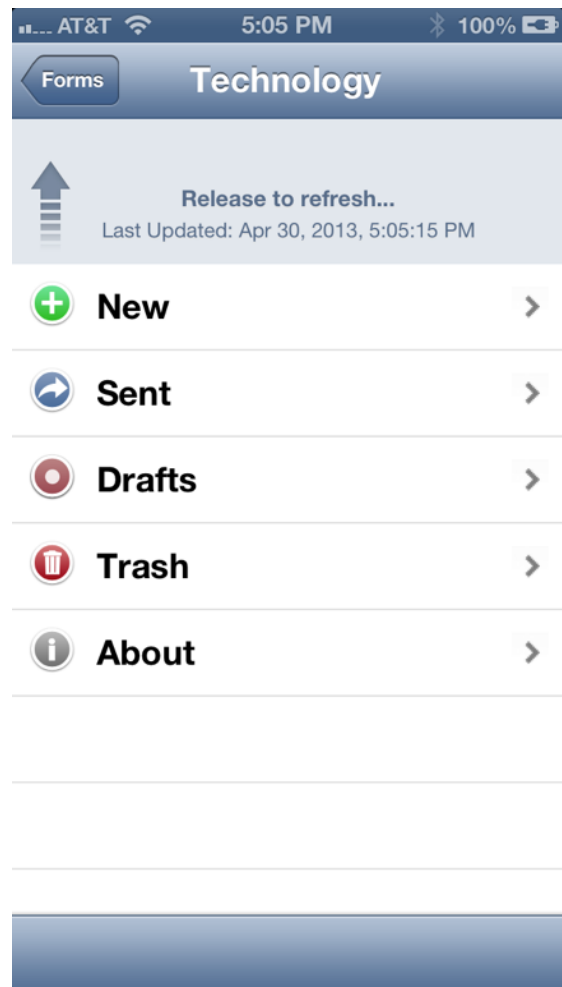
The "form\_id" is the unique identifier that is created for your form, found under the General Info tab within FormEntry for Mac.

The "pin\_code" parameter is sent when the user logs into the form using their 4-digit passcode number. The PIN Code Plug-In must be purchased from within the In-App Purchase Plug-In Store within FormEntry for Mac. This parameter is upon the parameter id of the control and might not be the actually named "pin\_code".

The "table\_id[]" and "table\_updated[]" are parameter arrays which contain a collection of tables IDs of the database along when last updated date in GMT. Based upon the posted parameter values, this is useful if want to update a specific table and not deliver the entire database back to the user.

## WEB XML URL

You are responsible for maintaining the web service that your form app points to the Web XML URL. Your response should always be an XML response. If there are no



tables to update on the iOS device, you should reply back with an XML response of just the empty database and no tables.

```
<?xml version="1.0" encoding="UTF-8"?>
<database>
</database>
```

By looking at the iOS posted 'table\_id[]' and 'table\_updated[]' parameters, you can decide what table needs to be updated. Your response for updating a table or tables should be in the following XML format. Note, that each table within your database will generate a "table" element with an "id" attribute and an "updated" attribute.

The column ids values are placed within the 'th' element and the column type is placed in the 'type' attribute of the header. The record values are placed with a unique 'tr' row and are placed with the appropriate 'td' element.

```
<?xml version="1.0" encoding="UTF-8"?>
<database>
  <table id="table" updated="2013-04-30 21:02:51 +0000">
    <thead>
      <tr>
        <th type="id">id</th>
        <th type="string">first_name</th>
        <th type="string">last_name</th>
      </tr>
    </thead>
    <tbody>
      <tr>
        <td>1</td>
        <td>Jonathan</td>
        <td>Freeman</td>
      </tr>
      <tr>
        <td>2</td>
        <td>Bill</td>
        <td>Gates</td>
      </tr>
      <tr>
        <td>3</td>
        <td>Steve</td>
        <td>Jobs</td>
      </tr>
    </tbody>
  </table>
</database>
```

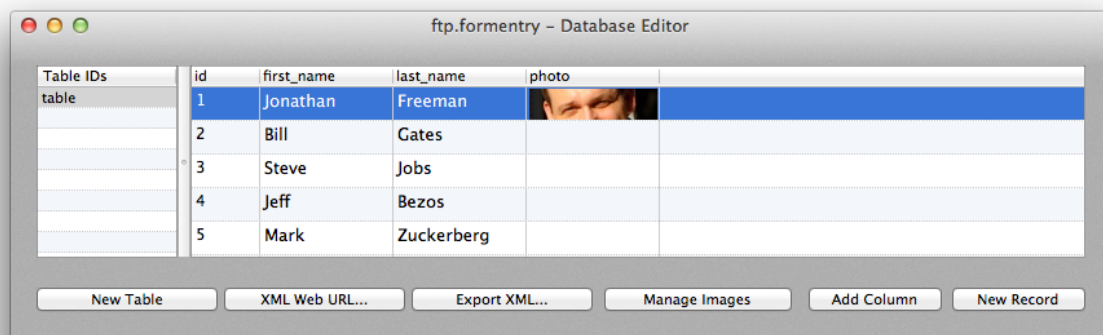
```

        <tr>
            <td>4</td>
            <td>Jeff</td>
            <td>Bezos</td>
        </tr>
        <tr>
            <td>5</td>
            <td>Mark</td>
            <td>Zuckerberg</td>
        </tr>
    </tbody>
</table>
</database>

```

Notice how similar this format is to standard HTML. The only difference is having the extra database element. You could simply change the file extension from .xml to .html and it should render in any modern web browser as standard HTML. This format can be exported from FormEntry for Mac by selecting the 'Export XML...' button on the Database Editor. It will export all the tables from your database into one XML file.

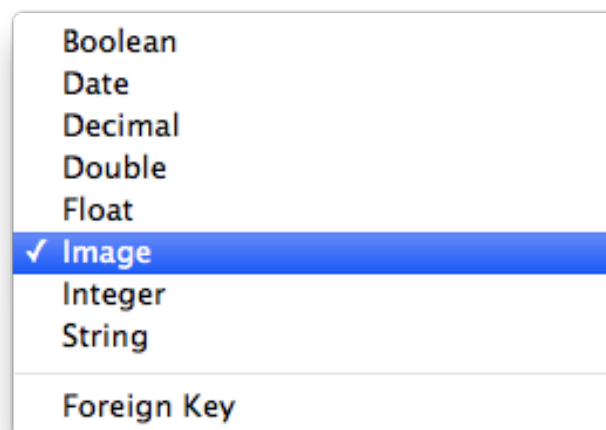
If you add an image column to your table and they will be exported to the 'img' element using inline Base-64 for the 'src' attribute.



## COLUMN DATA TYPES

FormEntry for Mac's Database Editor allows you to add up to nine different data types, Boolean, Date, Decimal, Double, Float, Image, Integer, String and Foreign Key.


The Image and Foreign Key column data types are custom types that help FormEntry Touch query the database faster, while the rest are standard data types.



## STANDARD AND CUSTOM DATA TYPES

The standard data types are really object types that are applied to the object database. The same applies for custom data types, Image and Foreign Key. However, these

custom types are specific to FormEntry and help in querying the database. The data types are explained in the table below:

TYPE	LIMITS	EXAMPLE	DESCRIPTION
Boolean	<ul style="list-style-type: none"> <li>• Sets either TRUE or FALSE</li> <li>• Limited to 10 columns per table</li> </ul>	TRUE	Boolean values are values that are either TRUE or FALSE. You can use these values to set Checkmarks and On/Off Switches from the Rule Builder.
Date	<ul style="list-style-type: none"> <li>• Sets the date and time</li> <li>• Limited to 10 columns per table</li> </ul>	1982-02-12 11:00:00	The Date value must be in this format: YYYY-MM-DD HH:MM:SS
Decimal	<ul style="list-style-type: none"> <li>• Limited to 10 columns per table</li> </ul>	1123.19	Holds a decimal value.
Double	<ul style="list-style-type: none"> <li>• Limited to 10 columns per table</li> </ul>	17919.9391	Holds a double value.
Float	<ul style="list-style-type: none"> <li>• Holds 7 digits of precision</li> <li>• Limited to 10 columns per table</li> </ul>	179.9999	Holds a float value.
Image	<ul style="list-style-type: none"> <li>• It is recommended your keep images to a small file size, small resolution and have them highly compressed.</li> <li>• Limited to 10 columns per table</li> </ul>		Image values can be used to populate icons as well as the Image control by using the Rule Builder.
Integer	<ul style="list-style-type: none"> <li>• Integer values can have a minimum value of -2147483648 to 2147483647</li> <li>• Limited to 10 columns per table</li> </ul>	432	Holds a integer value and values have no decimal places.

TYPE	LIMITS	EXAMPLE	DESCRIPTION
String	<ul style="list-style-type: none"> <li>• Should be kept small</li> <li>• Limited to 20 columns per table</li> </ul>	Jonathan	The String value holds a wide range of different character combinations.
Foreign Key	<p>Should always have a suffix of “_id” in the column id and the prefix should always be a table name.</p> <ul style="list-style-type: none"> <li>• Limited to 10 columns per table</li> </ul>	people_id	<p>Tables that contain foreign key columns are always the child table. The parent table is the suffix of the foreign key column name.</p> <p>“people_id” indicates that there is a parent table called “people” in the database. The row’s value points to the parent’s ID value.</p>

## Binding Controls to Database

Binding a control to the database, simply means that a control’s value should retrieve the values from a specific table at a specific Column ID. Once the value is set to the control’s value, the Record ID “id” of that value is tied or bound to the control. Currently, only four controls can be bound to the database, however, in combination with the Rule Builder, you can set the values to most of the controls with values from the database.

The four bindable controls that can be bound to a database are: Text Field, Spin Wheel Picker Item, Barcode and PIN Code control Plug-Ins.

As we work through the following four bindable controls in this “Binding Controls to Database”, we will be expanding upon a database example.



## TEXT FIELD BINDINGS

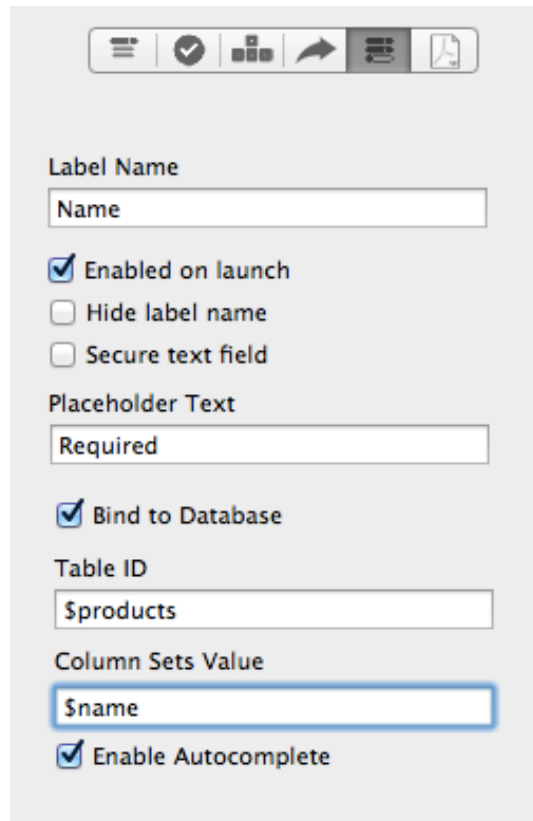
Text Field controls can be bound to a table's column by selecting the configuration tab from the control tab bar pane.

To bind a Text Field control to the database, select "Bind to Database" checkbox. In the Table ID, place the Tables ID you want to bind to, prefixed with the "\$" character. For example, if we are wanting to bind "products" table, you will need to place "\$products" in the Table ID for this Text Field.

Next, you will need to bind the Column ID that you want to set the value for the Text Field at The Column Sets Value input by prefixing "\$" to the Column ID. In this particular image example to the right, we want to bind the "name" Column ID, so we set "\$name" to the Column Sets Value input field.

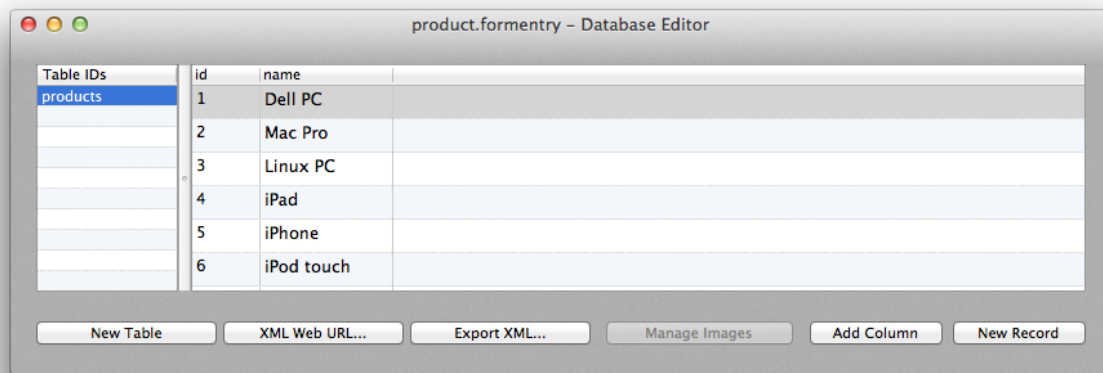
Also, for our example, we want to Enable Autocomplete checkbox. This will showcase an autocomplete drop down on the iOS device when the user starts typing.

From the Database Editor, create a database by adding a new table with the Table ID of "products". Add a "New Column" with the Column ID of "name" and a Column Data Type of "String". Add six new records with the following names in the image below. Your window should look something like the following:



The screenshot shows the configuration pane for a Text Field control. At the top is a tab bar with icons for configuration, validation, data, actions, and a document. The configuration tab is selected. The settings are as follows:

- Label Name:** A text field containing "Name".
- Enabled on launch:** ☒
- Hide label name:** ☐
- Secure text field:** ☐
- Placeholder Text:** A text field containing "Required".
- Bind to Database:** ☒
- Table ID:** A text field containing "\$products".
- Column Sets Value:** A text field containing "\$name", which is highlighted with a blue border.
- Enable Autocomplete:** ☒



Sync the form to FormEntry Touch and select your form. As you proceed to type in the Name Text Field, you will see an auto complete drop-down appear as you type along. You can select the value from the drop-down list and it will populate the the Text Field on the iOS device.

## BINDINGS SUMMARY

In review, adding a bindable autocomplete Text Field to your form app is straight forward.

1. Add a Table ID with a Column ID
2. Add Records to your Table
3. Bind your Text Field to the Table ID and Column ID with prefixes of "\$"
4. Enable Autocomplete



## RETRIEVING SAME TABLE VALUES

If a control is bound to a database, you can retrieve values from the database based upon the Record ID of the bound control. This requires the Rule Builder with a passing condition and an action that sets the value(s) with a key path.

Let's go ahead and add a new column to our "products" table with a Column ID of "sku" with an Column Data Type of "Integer". The database should look something like this:

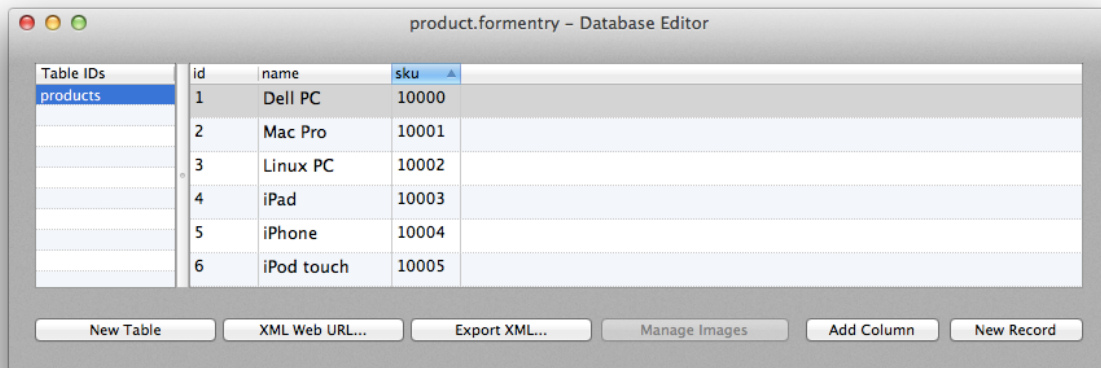


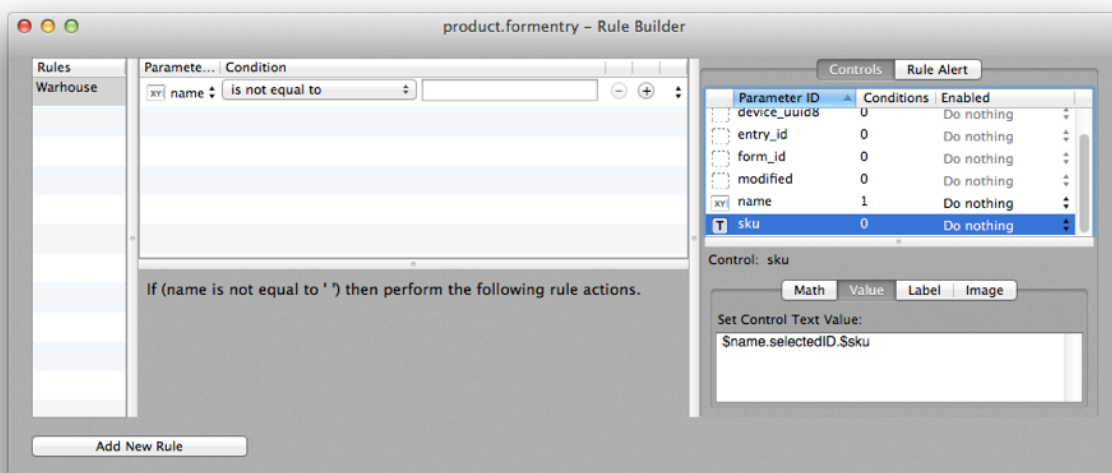
Table IDs	id	name	sku
products	1	Dell PC	10000
	2	Mac Pro	10001
	3	Linux PC	10002
	4	iPad	10003
	5	iPhone	10004
	6	iPod touch	10005

Now, in the Rule Builder, add a new Rule with a condition of "If (name is not equal to ' ') then perform the following rule actions." Note the blank space, if the name is not equal to an empty blank space.

Then for the Rule Action, select 'sku' in the list of Controls and select the Value tab. Set the following key path for the Value of 'sku'.

`"$name.selectedID.$sku"`

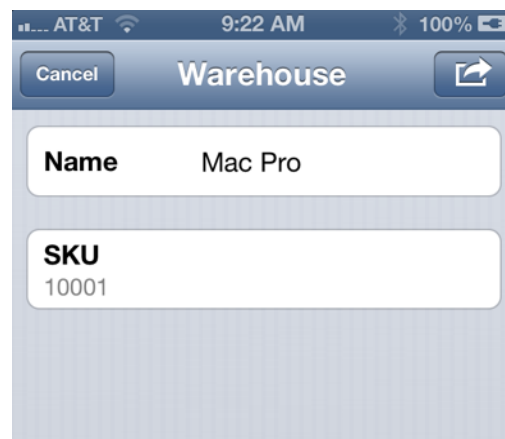
It should look like the following image:



Since both the “name” Column ID and the “sku” Column ID are on the same table “products”, you simply get the “\$name” record (by using the selectedID) and then get the “\$sku” value on the same record.

Sync your form to your device and when you select the Name text field, you will be able to select from the autocomplete list on the “name” Column ID. Once you have selected the name, the text field will be set with the record “id” of that record and the Rule Builder will be triggered.

The key path of “\$name.selectedID.\$sku” will be executed and the value at “sku” on the selected “name” record will be populated at the SKU text control.



## KEY PATH TRAVERSING TO PARENT TABLE

One of the important concepts in gathering data from across multiple tables through the “key path” in combination with the Rule Builder. This allows you to pull data values from other tables and place them into controls on your form app.

To start using key paths to pull values from across tables you will need at least two tables and at least one Foreign Key Column ID in the child table. So, let’s expand upon our database example and add another table. Our data model will be something like the following entity diagram, a many-to-one or one-to-many relationship:



We want to have two tables, “products” and “location”. In this relationship, the “products” table is the child for it has only one location, where as the “location” is the parent for at a given warehouse location, there could can be multiple products in inventory. As such, our Database Editor should look similar to the following with necessary records for the location.

## UPDATE PRODUCTS TABLE

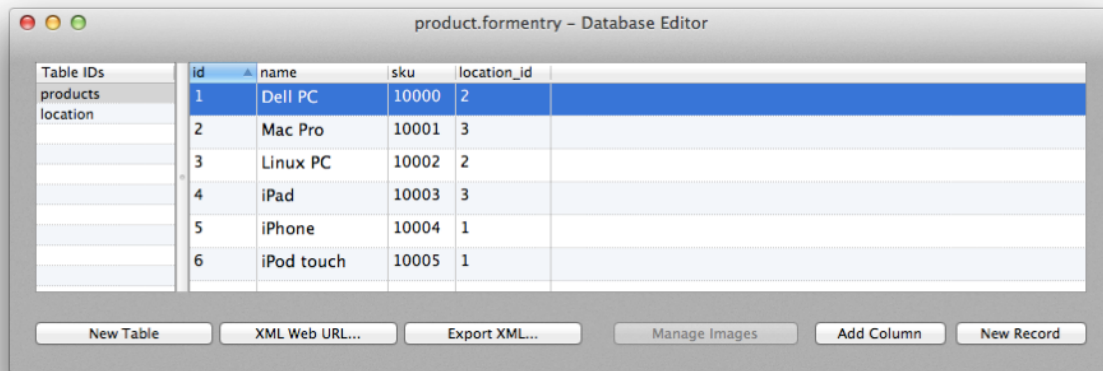


Table IDs	id	name	sku	location_id
products	1	Dell PC	10000	2
location	2	Mac Pro	10001	3
	3	Linux PC	10002	2
	4	iPad	10003	3
	5	iPhone	10004	1
	6	iPod touch	10005	1

Buttons: New Table, XML Web URL..., Export XML..., Manage Images, Add Column, New Record

Add the “location\_id” Foreign Key column to “products”. Note that “location” prefixed to “\_id” tells FormEntry that this table is the child table to the “location” table. All Foreign Key Columns are required to have the suffix of “\_id”.

## ADD NEW LOCATION TABLE

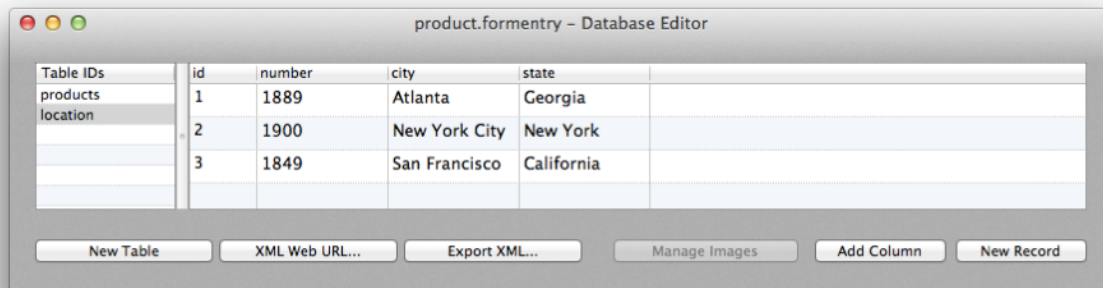
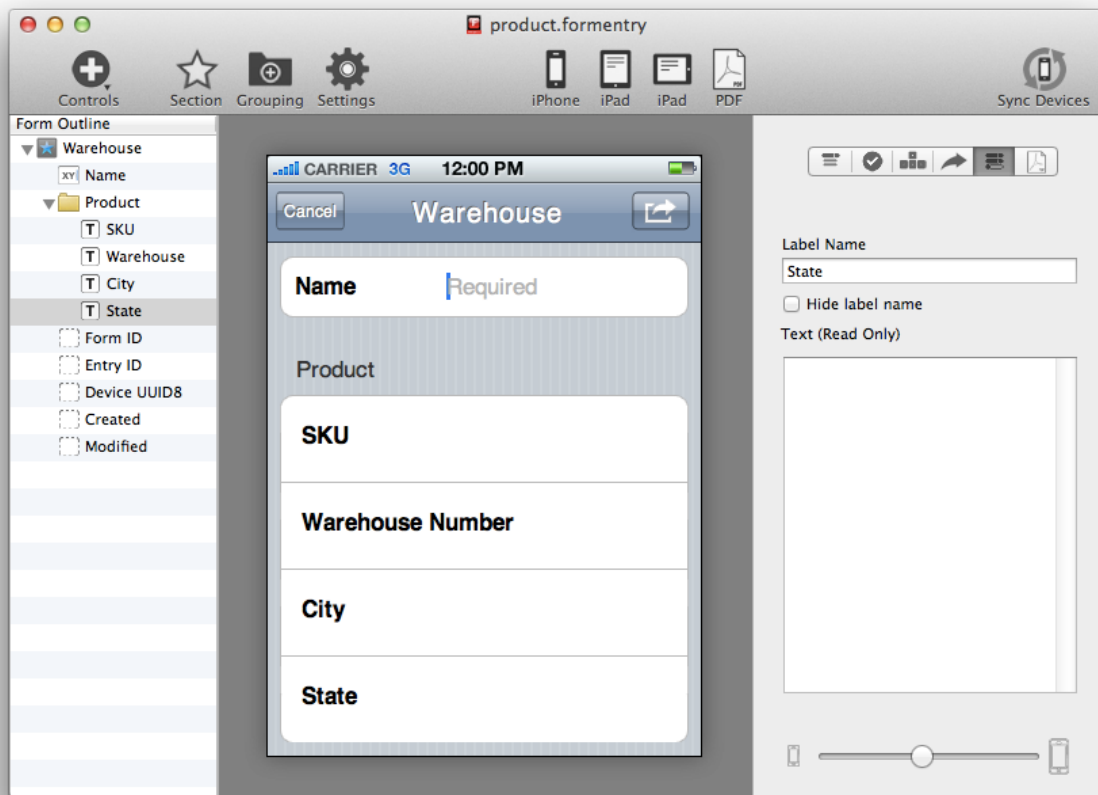


Table IDs	id	number	city	state
products	1	1889	Atlanta	Georgia
location	2	1900	New York City	New York
	3	1849	San Francisco	California

Buttons: New Table, XML Web URL..., Export XML..., Manage Images, Add Column, New Record

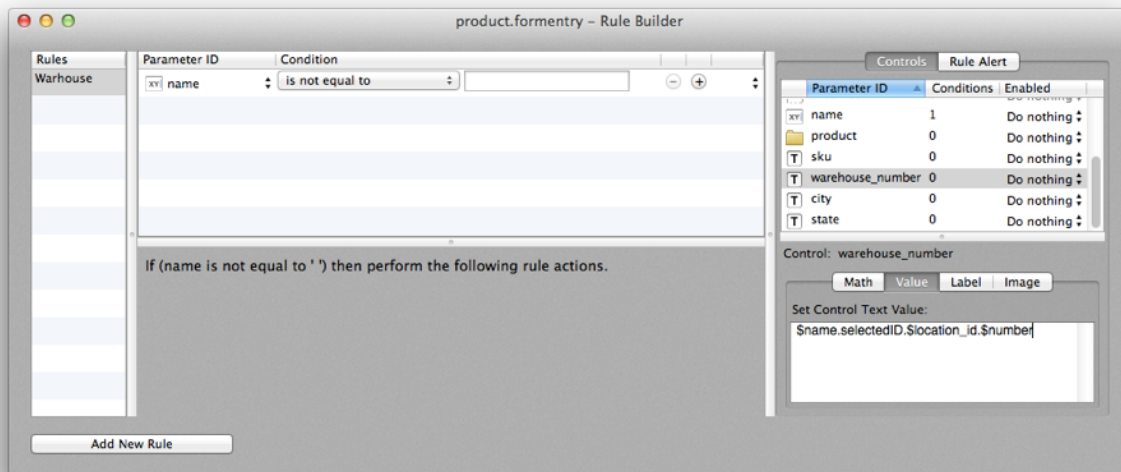
Create the “location” table and add three String Column IDs to the table as shown above.

Now add three new Text Controls (Text Controls are Ready Only) and with Labels “Warehouse Number”, “City” and “State”.



After you have added the Text controls, we need to update the Rule Builder to populate the Warehouse Number, City and State controls.

## KEY PATH TRAVERSING IN RULE BUILDER



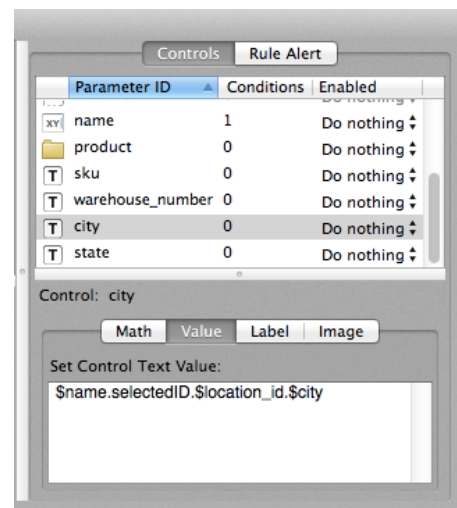
The key value path for retrieving across parents tables is done through the Foreign Key Column ID. In the above example, to set the “Warehouse Number” control, we want to use the following key path in the Rule Builder:

`“$name.selectedID.$location_id.$number”`

What this means is:

`<parameter id>.<record ID>.<parent id>.<column id>`

Select the “warehouse\_number” in the Rule Builder for this Rule and select the Value tab. Basically, when the “name” parameter is set (which is bound to the database), we want to grab the record ID of said “name”, the lookup the Column ID of “location\_id” (FormEntry Touch recognizes the ‘\_id’ in the Column ID name) and



retrieves the “number” Column ID for the “products” “id” at “number” Column ID.

Same methodology applies for both the “City” and “State” controls. We select the “city” and “state” parameter IDs in the Rule Builder and update the their Values with their key path:

```
“$name.selectedID.$location_id.$city”  
“$name.selectedID.$location_id.$state”
```

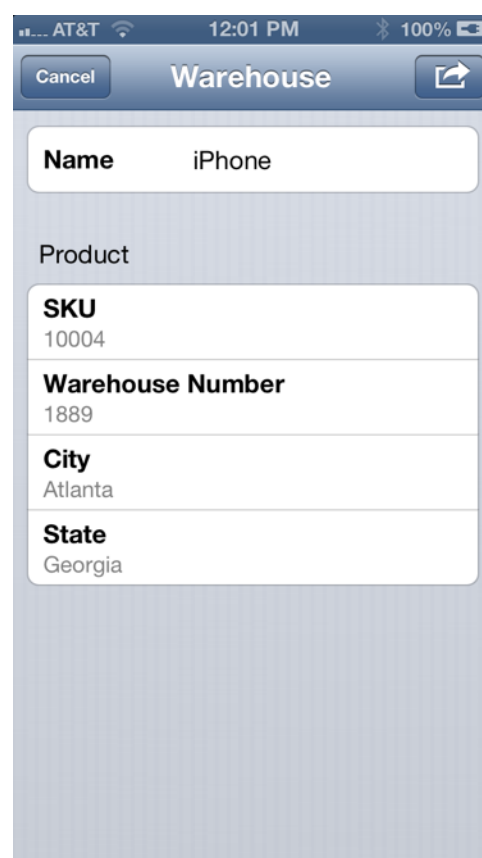
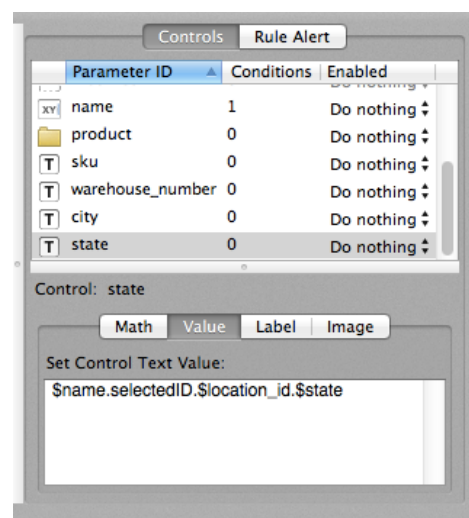
Now update your iOS device with the modified form. Select a Name from the autocomplete and the SKU should be populated from the ‘products’ table at the Column ID of ‘sku’. The Warehouse Number should be populated with ‘number’ Column ID from the ‘location’ table, while the City and State should be populated from their respective Column IDs.

Notice if you select a different product from the autocomplete, the values will looked up and inserted into the four Text controls.

## KEY PATH SUMMARY

In review, traversing from child to parent tables is straight forward.

1. Have at least two tables with a child to parent relationship.
2. The child uses the suffix of “\_id” in a Foreign Key Column Type to the parent.
3. User the Rule Builder for triggering updates and retrieving database values through key paths.

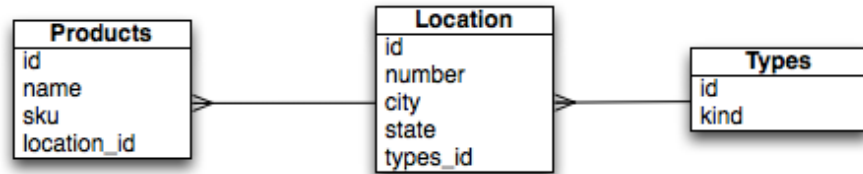


## TRAVERSING TO GRANDPARENT TABLE

You can expand upon the key path convention even further by traversing from child to parent to grandparent table and even further. Let’s see an example of how to transverse to the grandparent table, we will use three tables. Your data model will look like the

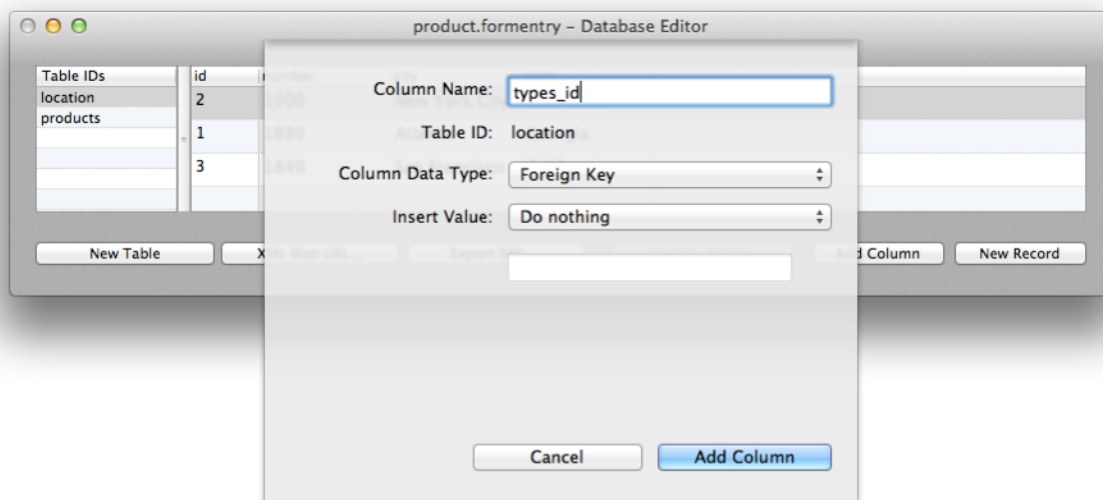


following entity diagram, a many-to-one or one-to-many relationships. We will need a new table “types” that is the parent of “location” and is related with the “types\_id”. Remember, the child table always contains the Foreign Key Column Data Type.

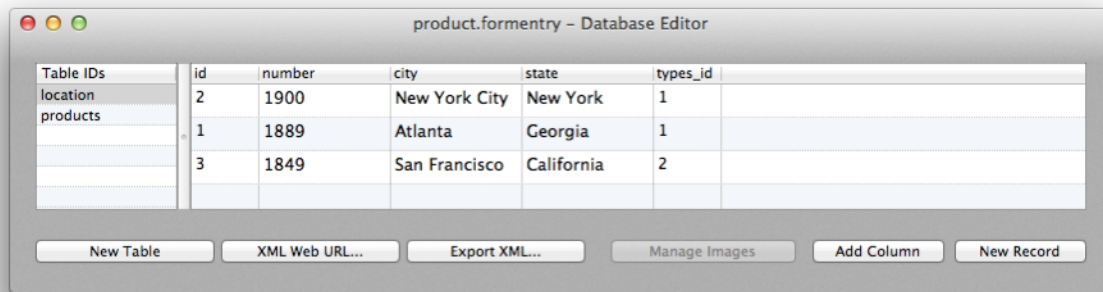


## UPDATE LOCATION TABLE

Update the ‘location’ table by adding a new Foreign Key Column Type called “types\_id” as in the following:

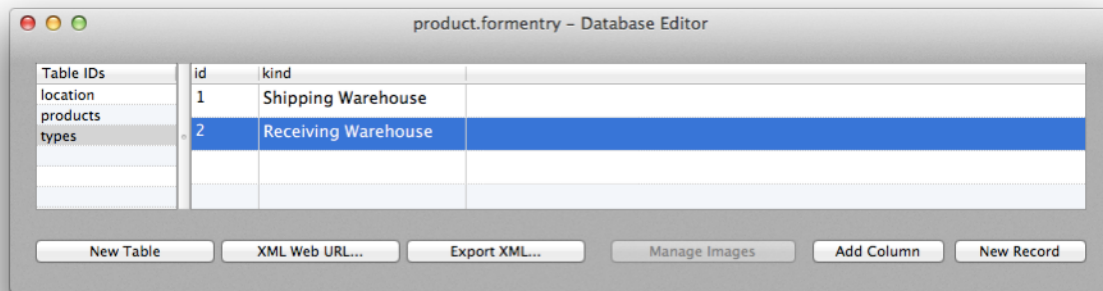


By using the “types\_id”, FormEntry will assume there will be a table called “types” in the database. Let’s add some values to the “types\_id” by adding an either “1” or “2” value for the foreign key.

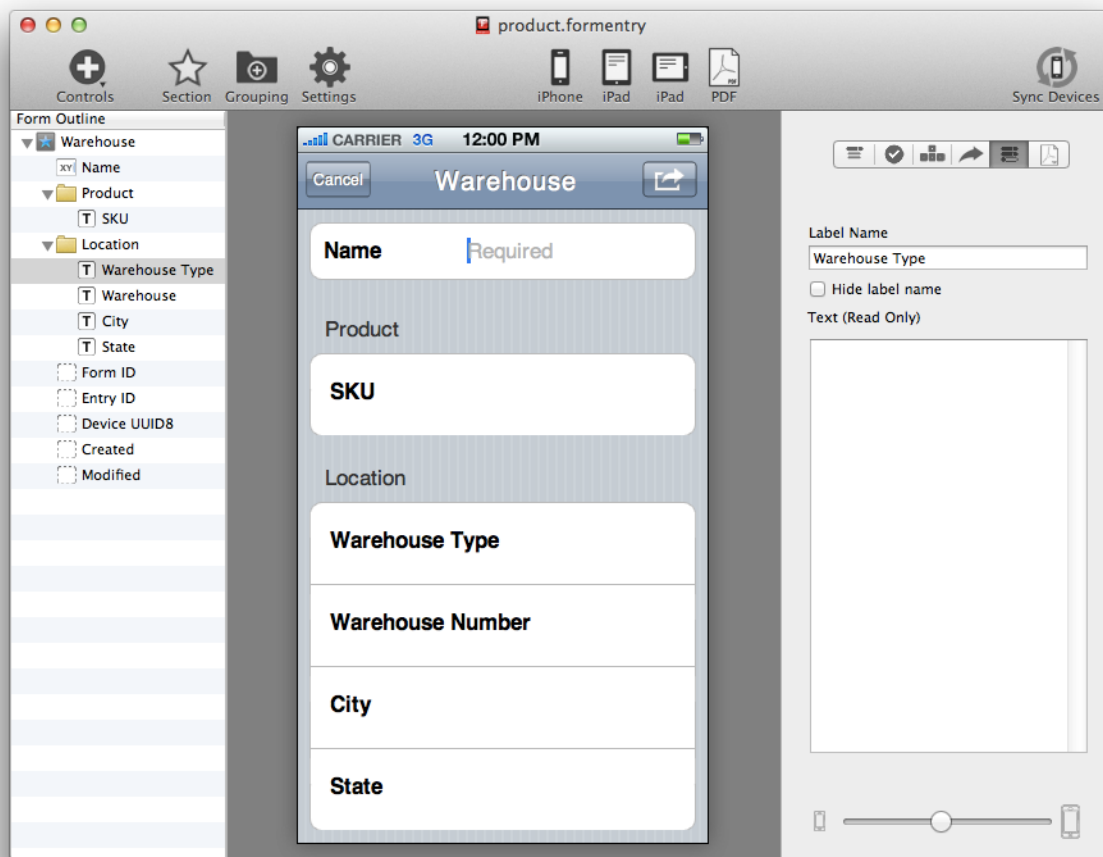


## ADD NEW TYPES TABLE

Now let's add a new table called "types" where this will hold two record values for Shipping Warehouse or Receiving Warehouse. Add a new Column ID with the title of "kind" and set it to a Column Data Type of "String". Your table should look like the following:



Now let's add a new Text Control called "Warehouse Type" and rearrange the form layout to look something like the following:



## KEY PATH TRAVERSING IN RULE BUILDER

From within the Rule Builder, select our previous Rule example and then select the “warehouse\_type” from the Control list. Select the Value tab and add the following key path in the text box:

```
$name.selectedID.$location_id.$types_id.$kind
```

This key path will traverse the 3 tables and retrieve the value of the related “kind” value on the “types” table.

Your Rule Builder should look like the following screenshot to the right.

Now update your iOS device with the new form and select a Name from the autocomplete. You will see either “Shipping Warehouse” or “Receiving Warehouse” in the Warehouse Type Text Control.

You can repeat this process with child to parent relationship, and child to grandparent relationship. It is recommended that when traversing key paths, you should stay with traversing one or two tables. If you try to traverse more tables, be aware of a performance hit depending on your iOS device.

## KEY PATH SUMMARY

In review, traversing from child to grandparent tables is straight forward.

1. Have at least three tables with a child to parent relationship and parent to grandparent relationship.
2. The child and parent uses the suffix of “\_id” in Foreign Key Column Types to the parent and grandparent.
3. User the Rule Builder for triggering updates and retrieving database values through key paths.

The screenshot shows the Rule Builder interface at the top, which includes a table with the following data:

Parameter ID	Conditions	Enabled
name	1	Do noth...
product	0	Do noth...
sku	0	Do noth...
state	0	Do noth...
warehouse_type	0	Do noth...
warehouse_number	0	Do noth...

Below the table, the 'Control' is set to 'warehouse\_type'. The 'Set Control Text Value' field contains the key path: `$name.selectedID.$location_id.$types_id.$kind`.

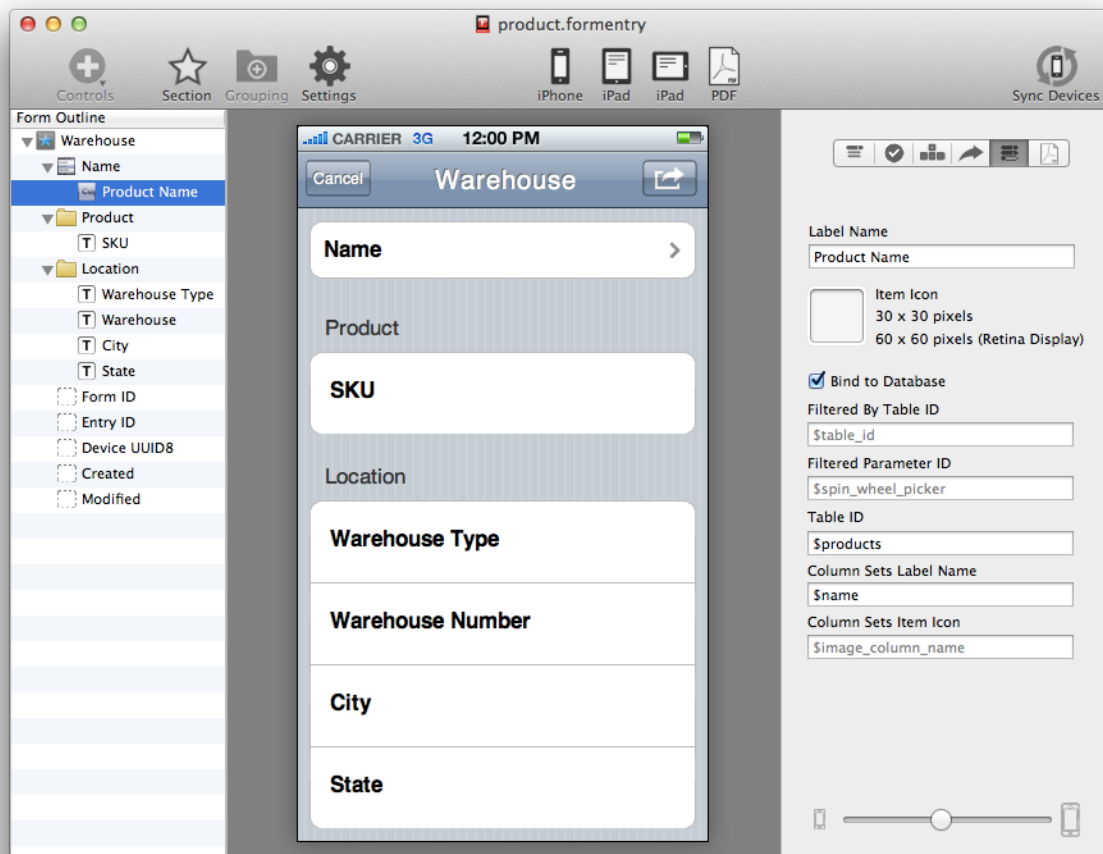
The bottom part of the screenshot shows the 'Warehouse' form on an iOS device. The form has the following fields and values:

- Name:** Linux PC
- Product:**
- SKU:** 10002
- Location:**
- Warehouse Type:** Shipping Warehouse
- Warehouse Number:** 1900
- City:** New York City
- State:** New York

## SPIN WHEEL PICKER BINDINGS

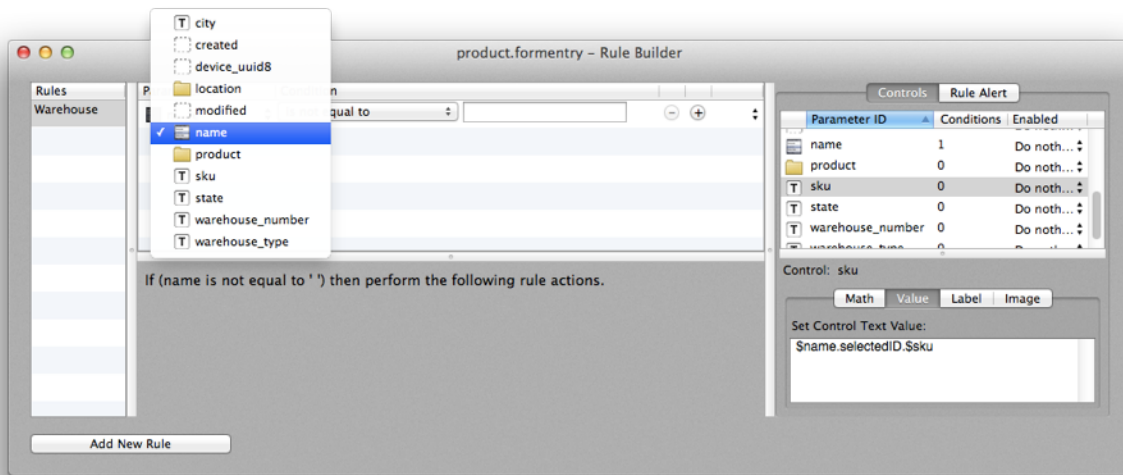
The Spin Wheel Picker Bindings is bound to the database through the Spin Wheel Item. The control can be bound to a table's column by selecting the configuration tab from the control tab bar pane. When binding a Spin Wheel Picker, only one Spin Wheel Item is required.

In our example, let's add a new Spin Wheel Picker. Change the Label Name to "Name". Now add a Spin Wheel Item and check the Bind to Database checkbox. Now add "\$products" to the Table ID and "\$name" to the Column Sets Label Name.



## SWITCH TO SPIN WHEEL PICKER CONDITION

Now in the Rule Builder window, let's make sure we select the "name" of the Spin Wheel Picker and not the "name" for the Text Field.



You will need to change the to Spin Wheel Picker "name", delete the "Name" Text Field and probably rename the Spin Wheel Picker parameter ID from "name1" to "name". Your final Rule Builder window should look like the above screenshot.

None of your rule actions changes, you simply want to switch from using the "Name" Text Field to the "Name" Spin Wheel Picker.

You want to make sure that your condition is basically the same as before : “If (name is not equal to ‘ ‘) then perform the following rule actions.” The same as not equal to a blank space.

Now Sync your form app to your iOS device. You now have a Spin Wheel Picker that is bound to your database instead of the Text Field.

Select the Spin Wheel Picker and you will see the list of products that Spin Wheel Item is bound too. Select an item and press the Warehouse back button, your form app loads all the appropriate values, just like selecting the Text Field autocomplete.

A screenshot of an iPhone screen showing a form titled "Warehouse". At the top, there is a "Cancel" button and a "Warehouse" title with a right arrow. Below the title is a "Name" field with a "Mac Pro" dropdown arrow. Underneath is a "Product" section with an "SKU" field containing "10001". Below that is a "Location" section with a "Warehouse Type" field containing "Receiving Warehouse", a "Warehouse Number" field containing "1849", a "City" field containing "San Francisco", and a "State" field containing "California".

A screenshot of an iPhone screen showing the same "Warehouse" form, but with the "Name" dropdown menu open. The menu lists several options: "Dell PC", "Linux PC", "Mac Pro" (which is highlighted), "iPad", and "iPhone". The background of the form is dark blue.

## **BIND ICONS AND IMAGE KEY PATHS**

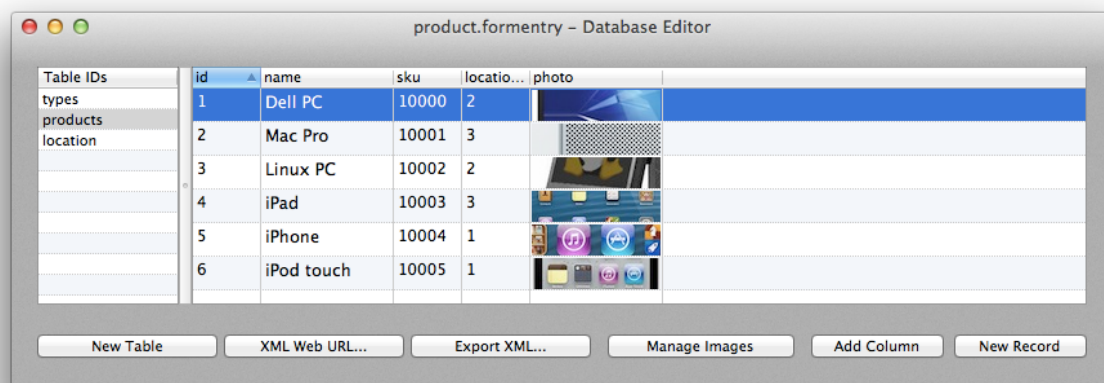
Now we want to populate our Spin Wheel Items, Product grouping and a new Image control with image values from our database.

We need to gather six images and add them to the Database Editor by adding a new Column ID of “photo” with a Column Data Type of Image. Once the Image column of “photo” is added, double click at the cell that of “photo” and the row you are wanting to add an image. Or you can select “Manage Images” button.



From the drop down sheet, drag an image into the Image Well, and in the above example, we added an image of the iPhone from the Desktop. Repeat this process for the remainder of your images.

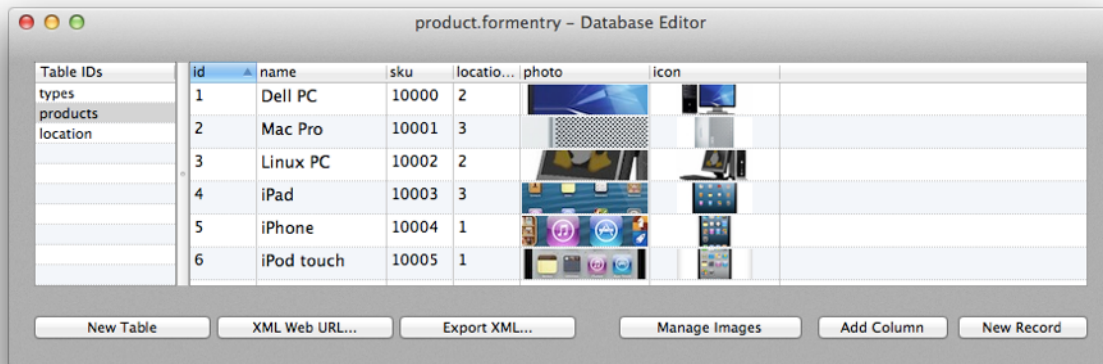
When adding your images, you want to make sure you use the smallest possible size images. Think about using smaller resolution at all times.





Let's repeat the process again for the icons. We to use 60 x 60 icons (Retina Display) of the Spin Wheel Item and Product grouping, so we resized our photos and saved them as icons.

We need to create another Column ID "icon" with an Image Column Data Type. We are repeating the same process as the photos and our Database Editor should look like the following screenshot.



Now, let's go ahead and bind the Spin Wheel Item's Icon to our database. Select the Spin Wheel Item from the Form Outline and add the Column ID of "\$icon" to the Column Sets Item Icon input field.

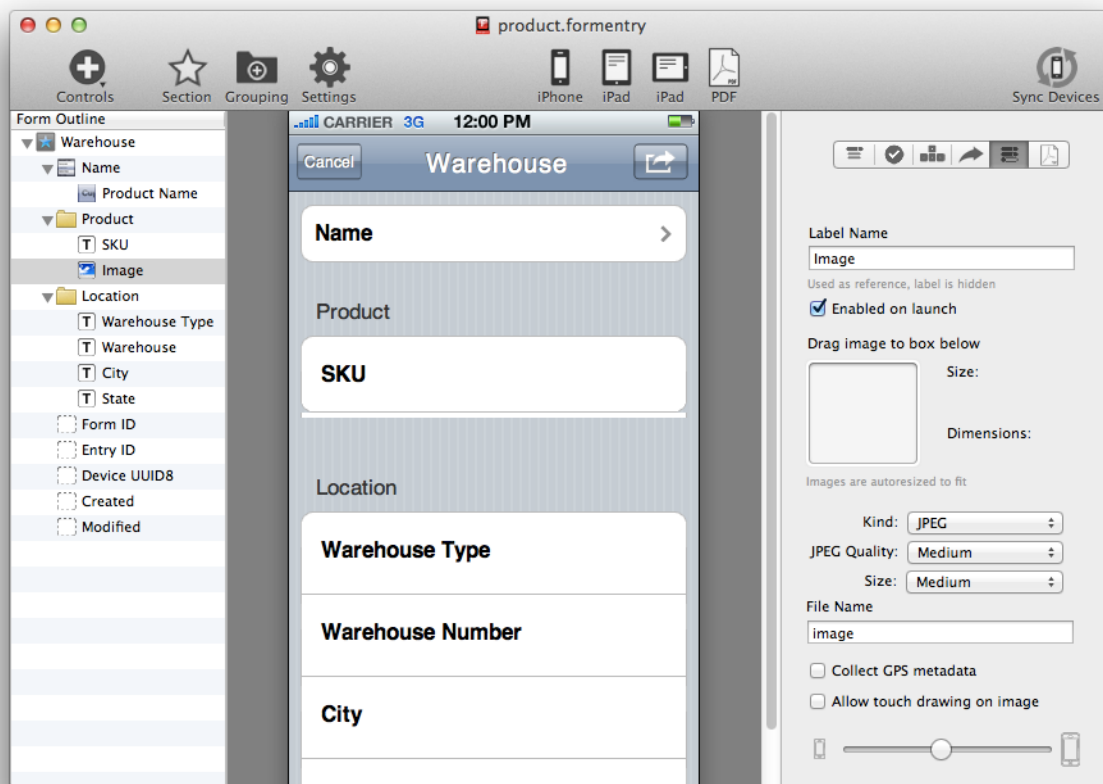
When we bind the Spin Wheel Item's Icon to the Image Column ID, we really want to bind a much smaller icon and not a full size image.

The images we added in our example for the 'photo' column are less than 100K, however they are under 300 pixels in width. You will need to determine the best size and performance expense. Loading photos into icons in your form app can be done, however it will pay a performance penalty.

Now let's add an Image Control to our form app, place the Image Control under the Product grouping. Do not add any image from the Desktop to the Image Well.

The screenshot shows the configuration panel for an 'Item Icon' in the FormEntry application. At the top is a toolbar with icons for menu, checkmark, grid, undo, redo, and save. The configuration fields are as follows:

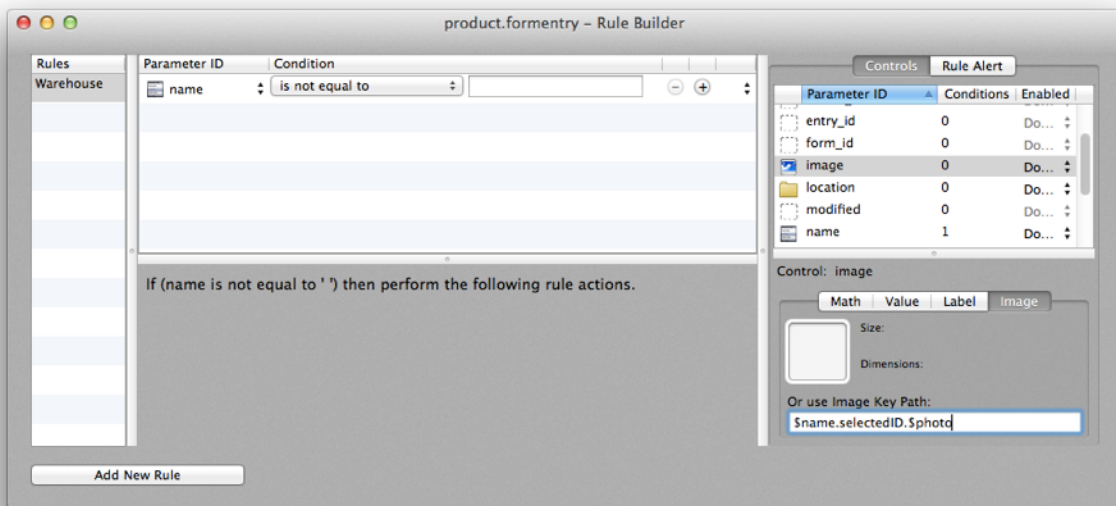
- Label Name:** Product Name
- Item Icon:** A small square icon placeholder. Text indicates sizes: 30 x 30 pixels and 60 x 60 pixels (Retina Display).
- Bind to Database:** A checked checkbox.
- Filtered By Table ID:** \$table\_id
- Filtered Parameter ID:** \$spin\_wheel\_picker
- Table ID:** \$products
- Column Sets Label Name:** \$name
- Column Sets Item Icon:** \$icon (this field is highlighted with a blue border)



Now, within the Rule Builder window, we want to update the Image control with a full size photo and the Product grouping with an icon.

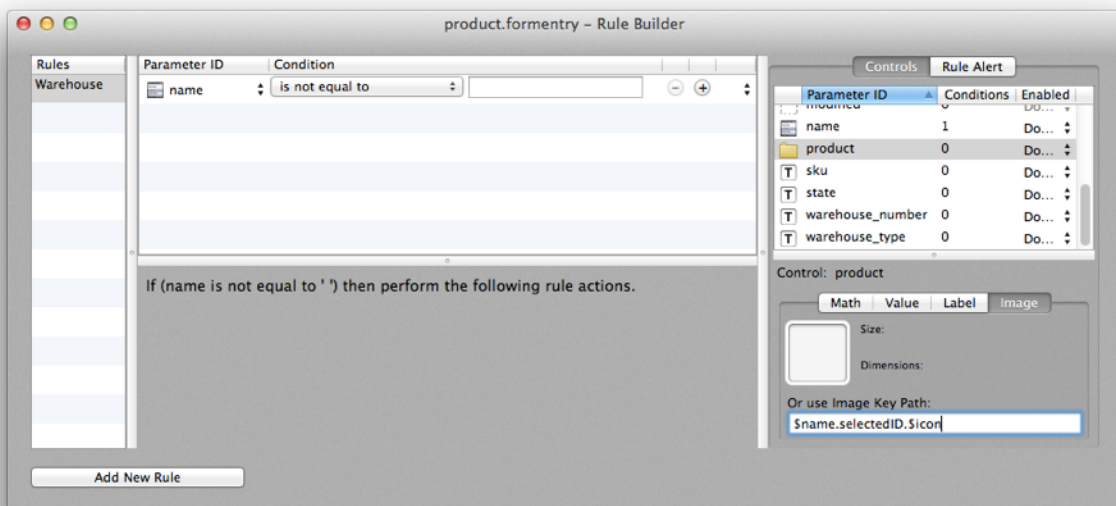
Select the “image” parameter id from the Controls tab and select the Image tab. Add the following Image Key Path to the “image” parameter:

```
$name.selectedID.$photo
```



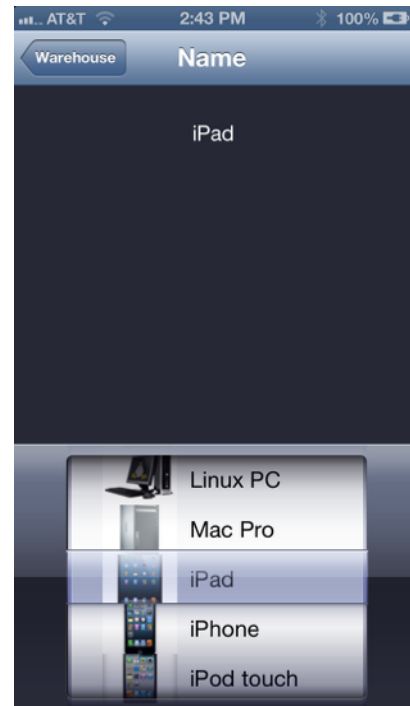
Now select the “product” grouping parameter from the Controls and add the following to the Image tab Image Key Path:

`$name.selectedID.$icon`



Now sync your form app project to your iOS device. Load a new form and select the name of one of the products from the Spin Wheel Picker.

You will notice that the Spin Wheel Picker items now contains the icons next to their names. Once you select the item, you will see the Product grouping has set the icon from the database as well as the Image Control gets set with the photo from the database.



## IMAGE SUMMARY

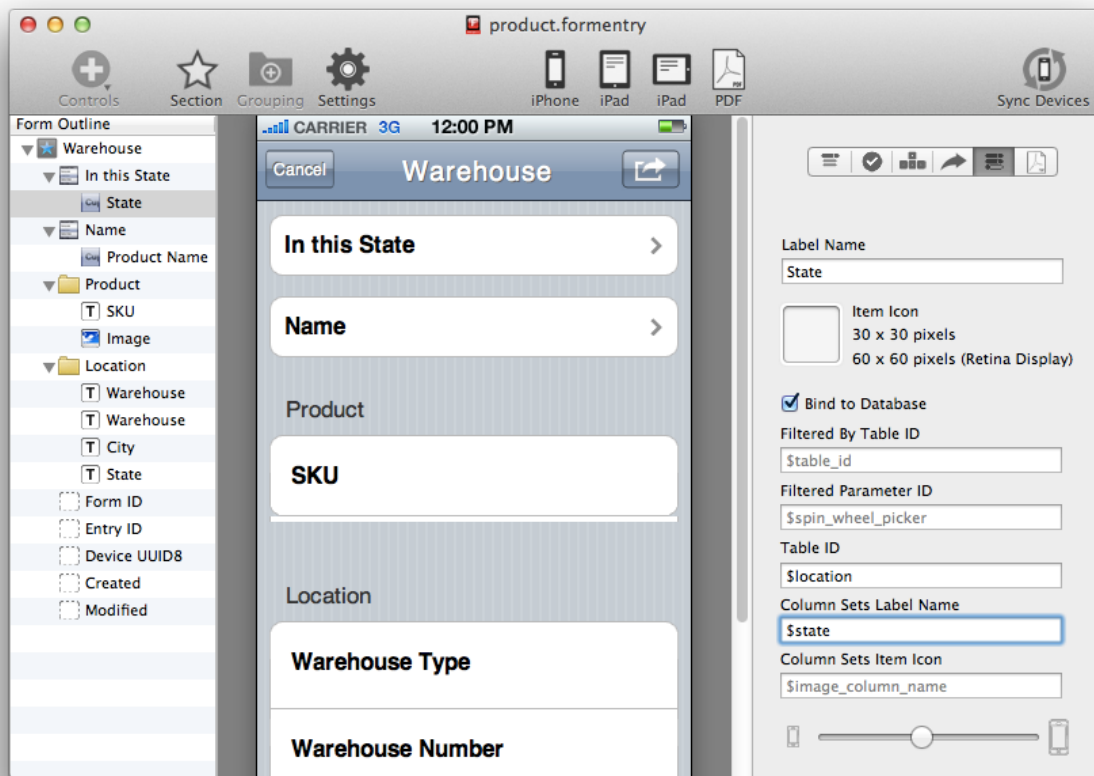
In review, retrieving images from the database is easy.

1. Add an Image Column Data Type to your database and drag the appropriate scaled images into the Image Well for each record.
2. Bind either the Spin Wheel Item icon or use Image Key Path within the Rule Builder to set Grouping Icons and Image Controls.

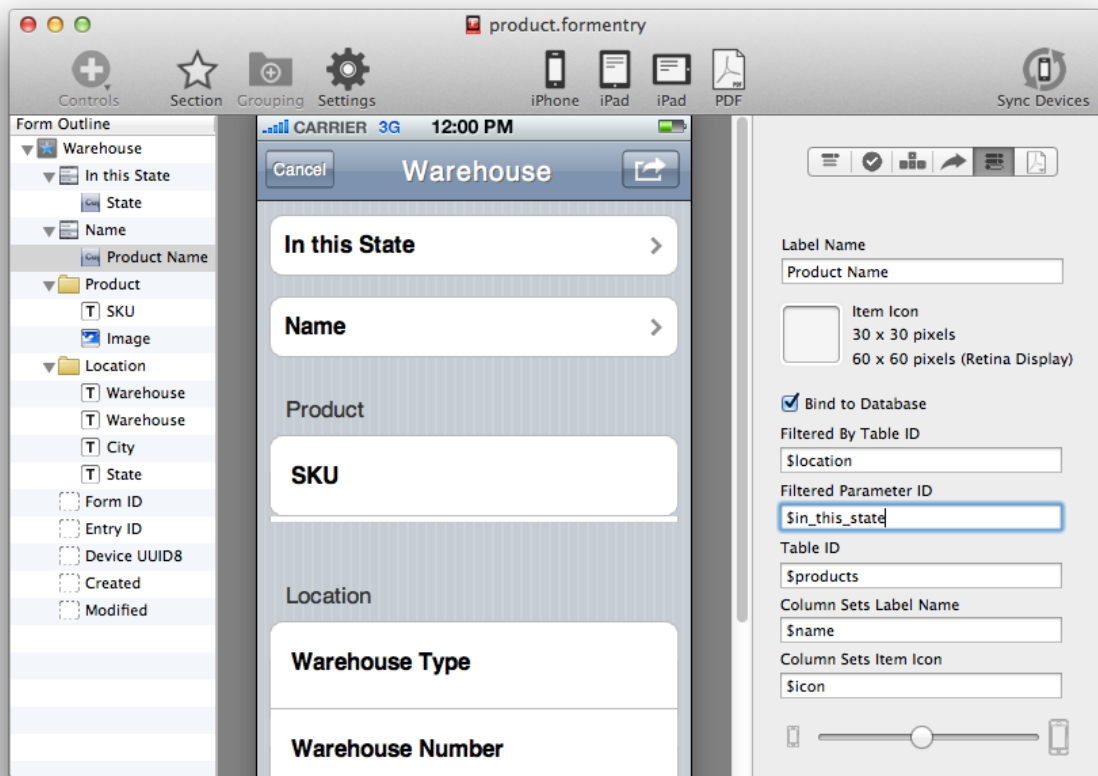
## FILTERED SPIN WHEEL PICKER

Filtering a Spin Wheel Picker is the same as filtering a child based upon parent. In following with our example, we only want to show certain products in the Products Spin Wheel Picker based upon the State from the 'location' table.

Let's add a new Spin Wheel Picker called "In this State" to our form app project. We also want to add a Spin Wheel Item called "State" and bind it to our 'location' database at the 'state' Column ID. Do not forget to add the prefix "\$" to our bindings.

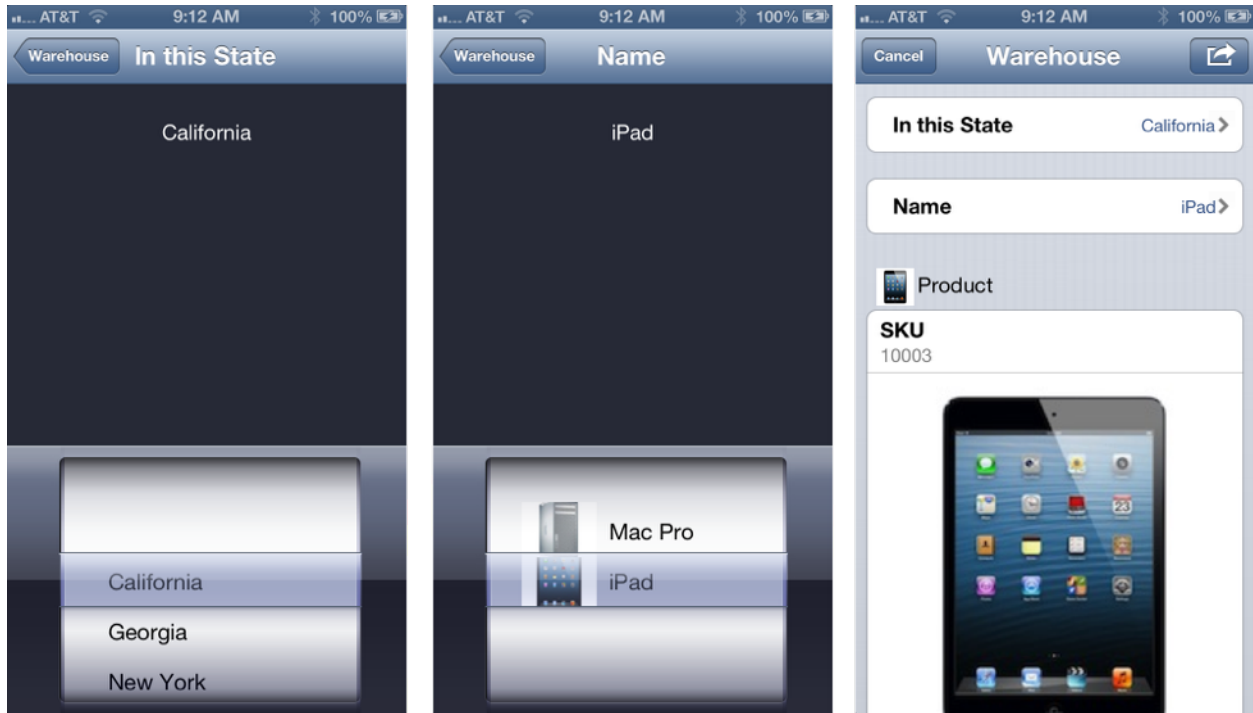


Now that the new “In this State” Spin Wheel Picker is bound to the database through the Spin Wheel Item, let’s filter our “Name” Spin Wheel Picker. Select “Product Name” and enter ‘\$location’ in the “Filtered By Table ID”. Next enter into “Filtered Parameter ID” the parameter ID of the control that will filter this Spin Wheel Picker, which is “\$in\_this\_state”. Your form app project should look like the following:



Now sync your form app project to your iOS device. Select the “In this State” Spin Wheel Picker. You will see the three States, select any state.

Now select the “Name” Spin Whee Picker and you will see that it is filtered by the state.



## **FILTERED SUMMARY**

In review, filtering a Spin Wheel Picker based upon another control is easy.

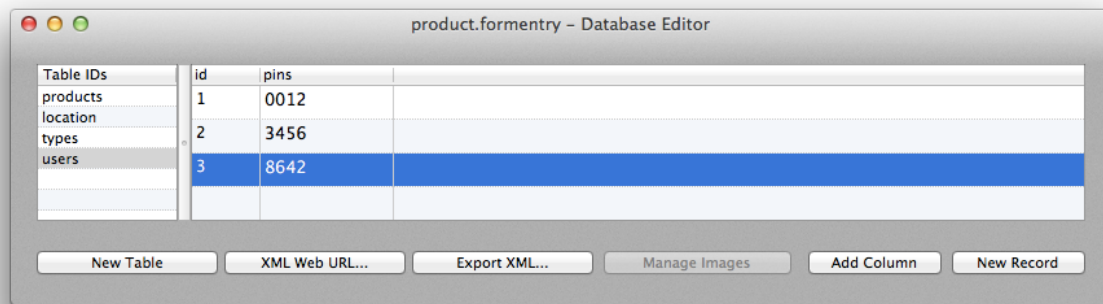
1. Select the Spin Wheel Picker you want to be filtered. Add the table ID by which you want to be filtered by.
2. Add the parameter ID of the control you want to be filtered by.

## **PIN CODE BINDINGS**

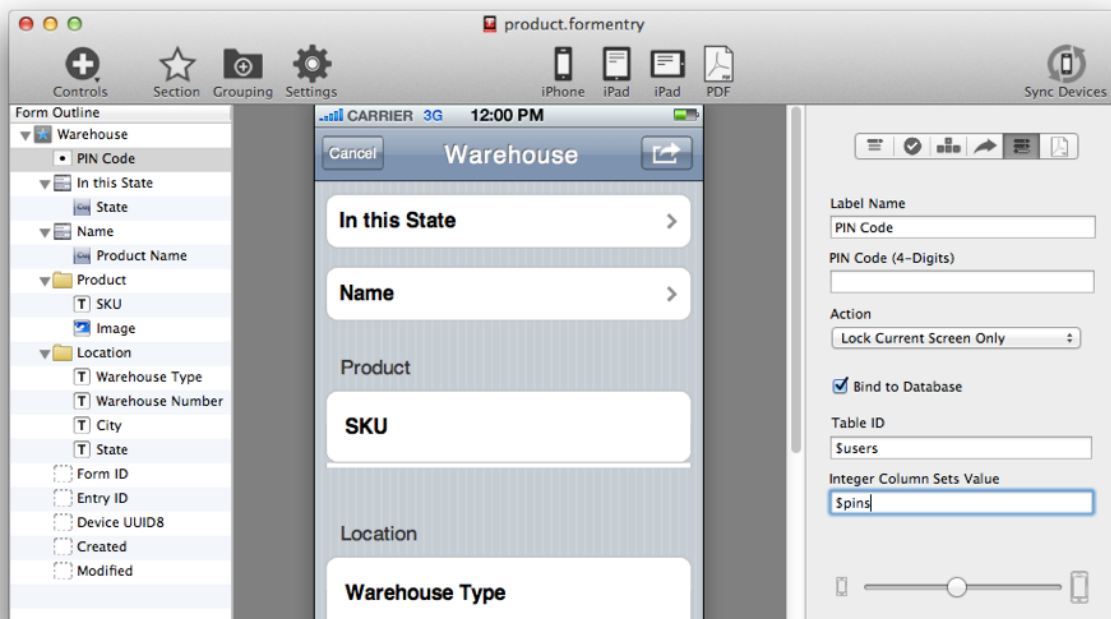
The PIN Code allows you to bind the PIN Code to your database. The PIN Code control requires a 4-digit passcode from your users to gain access to specific screens or the entire form app itself.

To bind the PIN Code to the database, we will need to create a new table. Let's add a new Table ID of 'users' to our example database. Now, let's add a new Column ID called 'pins' with the Column Data Type of either 'String' or 'Integer'\*. Add a few records and your Database Editor window should look like the following screenshot.

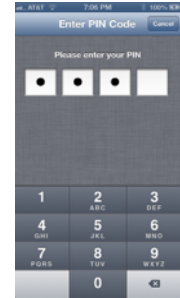




\*Note, that if you bind to an 'Integer' Column Data Type, you will be limited to some PIN numbers and not the full range. For example, in the above screenshot, '0012' is not an Integer, there's no way to add '0's before a true Integer. So, to get around this, your column must be a 'String' type to fully receive the 4-digit range from 0000 to 9999 values.

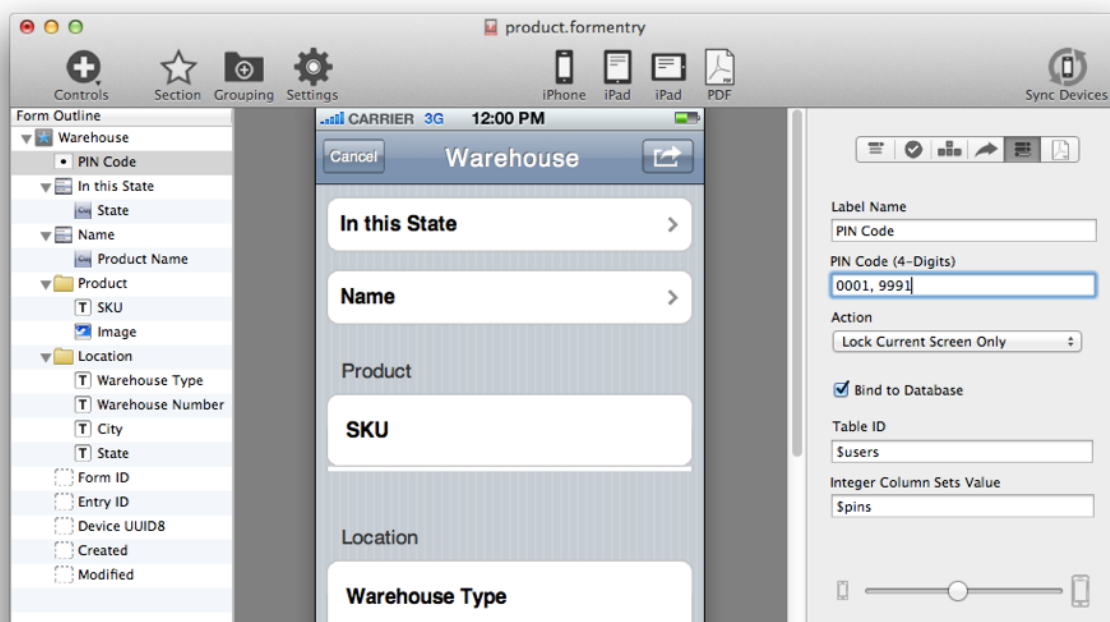


Now, let's add a new PIN Code control and bind to our newly created table. Bind the Table ID to 'users' and the Column ID of 'pins'. Your form app project should like the above screenshot.



Now sync your form app project to your iOS device and you should be able to log into the first screen based upon the values in the 'pins' column.

You can also mix the PIN Code input field with the values in the database. If we placed additional 4-digit PIN Codes at the PIN Code input field, we would have a screenshot like the following.



## PIN CODE BINDING SUMMARY

In review, binding the PIN Code to the database to straight forward.

1. Select the PIN Code control in the Form Outline. Add the table and column IDs to the Table ID and Column Sets Value input fields.
2. Add the Table and Column IDs with records that will be used by the PIN Code control.

## BARCODE BINDINGS

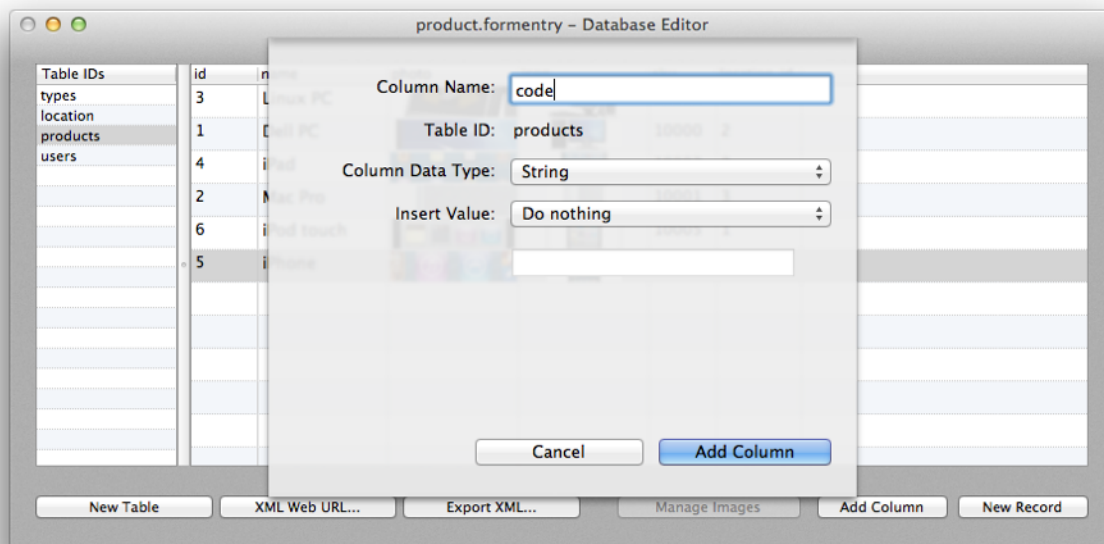
The Barcode Control Plug-In allows you to bind the Barcode control to your database. The Barcode Control allows you to scan up to sixteen different types of barcodes. You will typically bind the Barcode Control to a 'String', 'Double', 'Float' and 'Integer' Column Data Types on your table. Keep in mind, if your barcodes start with '0', you will definitely want to use a 'String' type.

When scanning barcodes, the Barcode value will be set after the camera has identified the barcode and interpreted the value from the barcode. If the Barcode Control is bound to the database, the interpreted barcode value is looked up at the bound Table ID and Column ID. If the interpreted barcode value is in the database, then the Barcode control is marked with the Record ID from the database.

In combination with the Rule Builder (similar to the Text Field bindings in the above section), you can perform actions to set other control values.

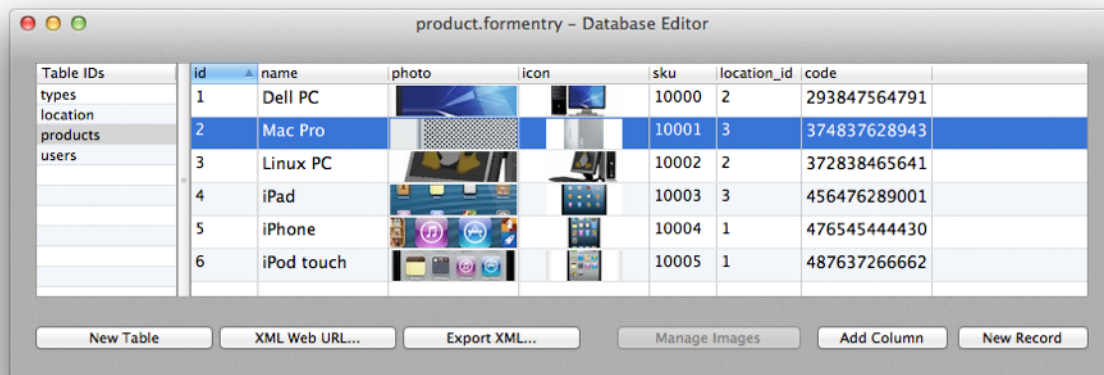
## ADD NEW COLUMN ID

First, let's add a new Column ID called 'code' to our 'products' table and want to make sure we use 'String' as the Column Data Type.



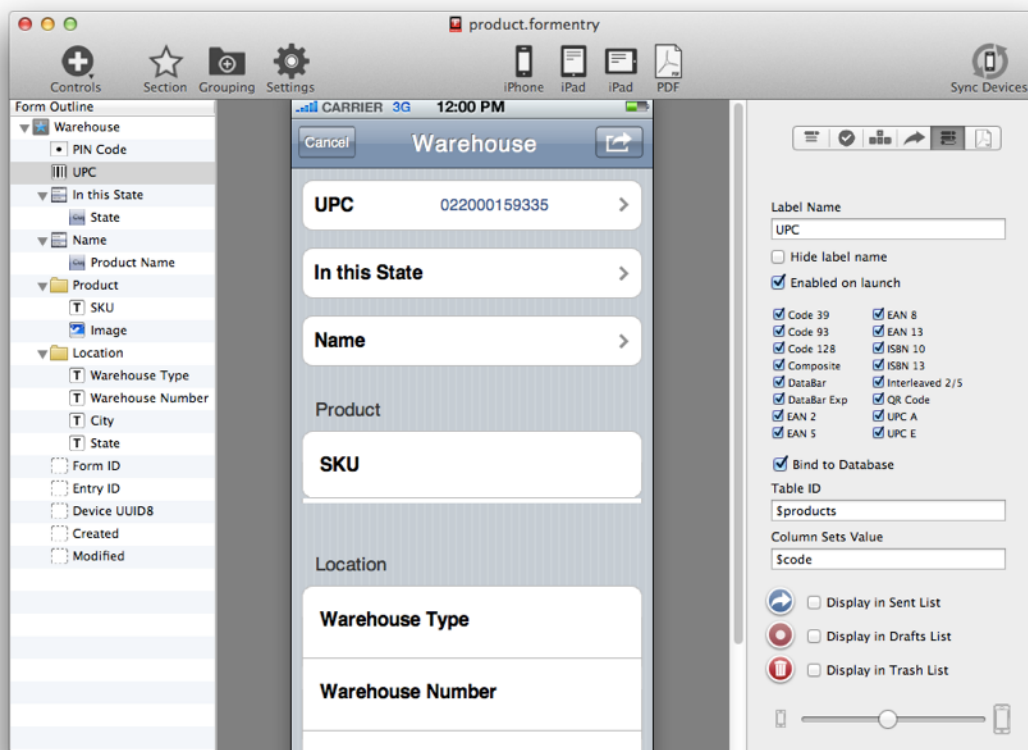


Now enter the barcodes for each of the products. Obtaining, creating and printing barcodes is out of the scope of this manual, however, there are lots of software and online tools that can help you in generating the barcodes necessary for FormEntry Touch to scan. Your Database Editor window should look like the following screenshot.



## BARCODE BINDINGS

Now add a Barcode Control and give it a Label Name of 'Barcode'. Bind it to the database checking the "Bind to Database" checkbox and by providing the '\$products' for the Table ID and '\$code' for the Column Sets Value input field. Your project window should look like the following:

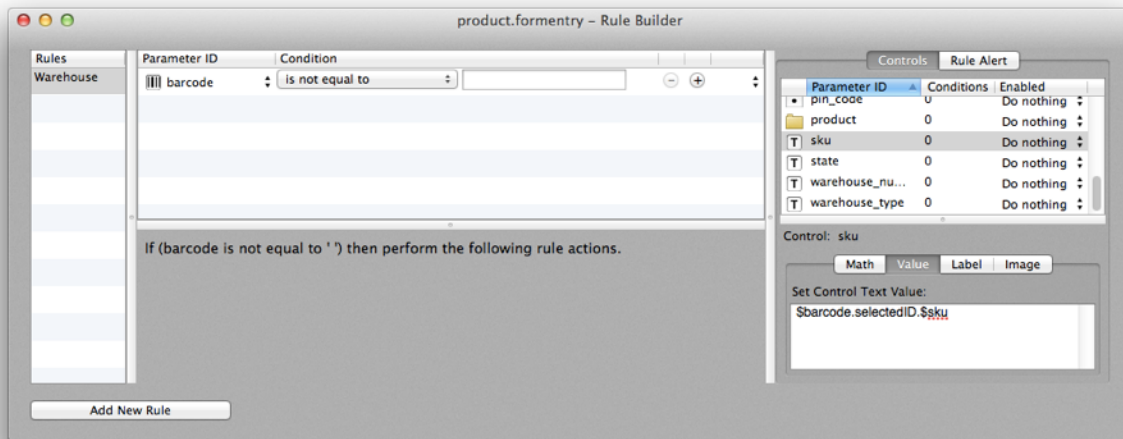


## BARCODE KEY PATH

Now, let's make sure we can update the form with values, so we need to add a new condition to the Rule Builder at the 'Warehouse' Rule. Select 'barcode' and the condition should be: "If (name is not equal to ' ') AND (barcode is not equal to ' ') then perform the following rule actions."

Now delete the "name" condition and change all the Values and Image of the Controls from "\$name.selectedID.\$sku" to "\$barcode.selectedID.\$sku"

You Rule Builder window should look like the following screenshot:



Now sync your form app project to your iOS device and load a new form. Select the Barcode control and point the iOS device over the barcode that matches the number in your database. The camera on the iOS device will automatically detect the barcode and load the value in the Barcode Control.

If the Barcode Control is bound to the database, it will look up the values and set the Record ID to the Barcode Control.



The Rule Builder Rules will execute and the rest of the form should be populated with values from the database. Your FormEntry Touch should look like the above screenshots.

## BARCODE BINDING SUMMARY

In review, binding the Barcode to the database is straight forward.

1. Select the Barcode control in the Form Outline. Add the table and column IDs to the Table ID and Column Sets Value input fields.
2. Add the Table and Column IDs with records that will be used by the Barcode control.

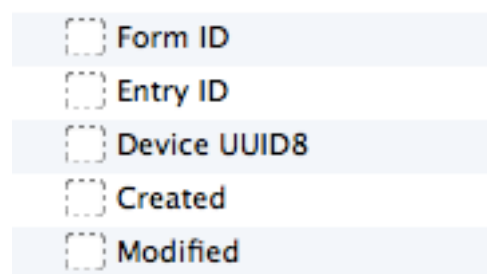


# HANDLING DATA & RESPONDING

**This section describes the ways of handling data coming from users and responding back to the devices:**

**Email, HTML Post and JSON Post, Open In, AirPrint, FTP, Save, WebDAV.**

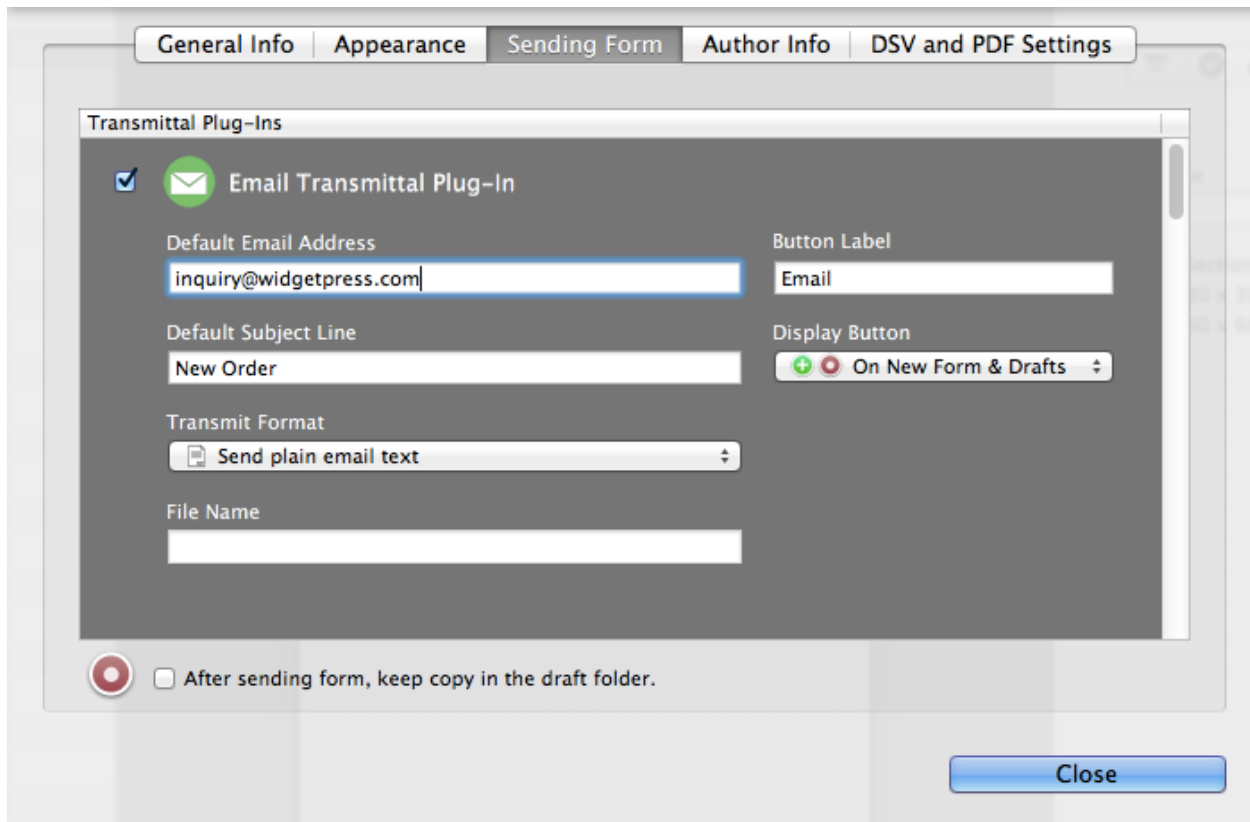
FormEntry Touch, by default when creating a new project, we add five hidden attributes that are key parameters. You can disable these fields, however, it is best to always send the Form ID, Entry ID, Device UUID (first 8 characters only), and the Created and Modified dates. If the form sends an HTML or JSON post, the following parameters are sent as: 'form\_id', 'entry\_id', 'device\_uuid8' ('device\_udid8' was deprecated in 2.3, it is no longer used), 'created' and 'modified' parameters. The 'created' attribute reflects when the entry was initially created by the user. The 'modified' attribute indicates when the entry was last modified by the user. Do not use these reserved key parameter names for any other control's HTML or JSON parameter names.



## Email Transmittal Plug-In

FormEntry for Mac sends four different types of formats in an email: 1) Plain email text 2) Attached DSV (Delimited Separated File, a.k.a. a CSV file) or 3) Attached PDF or if installed 4) Microsoft Excel (.xlsx file) It is up to the backend developer how an incoming email is processed. All emails use standard email technology and when sending plain text emails, all forms are sent as inline text emails with photos as email attachments. Emailing provides the simplest approach to collecting data.





The 'Mail' app on the device must be configured with a valid email account. FormEntry Touch hands off the task of emailing the form to the 'Mail' app which then sends the data.

FormEntry Touch assumes the form was successfully sent even if there's a network outage. All sent forms are archived under the 'Sent' folder on *FormEntry Touch*.

## FILE NAMES

File names follow the same style of naming convention. They can be based upon standard file names and/or mixed in with parameter IDs.

## HTML and JSON Post Transmittal Plug-Ins

The HTML and JSON Transmittal Plug-In, are paid plug-ins that first must be purchased from within FormEntry for Mac under the FormEntry -> Buy Plug-Ins... menu under the Main Menu. These are In-App purchases within FormEntry for Mac and you must use your Apple ID to make the transaction.

FormEntry for Mac can be configured to send data as a HTML post and/or as a JSON post. For an HTML posting, the form send is the data as a standard HTTP 'Post'. If there

are any photos, the post is sent as a multipart form 'Post'. If the form is sent via a JSON posting, it uses the standard HTTP 'Post' method, however any photos, images or signatures are converted to BASE64 strings.

The screenshot shows the 'Transmittal Plug-Ins' dialog box. It has five tabs: 'General Info', 'Appearance', 'Sending Form', 'Author Info', and 'DSV and PDF Settings'. The 'Sending Form' tab is selected. Inside the dialog, there are two sections for plug-ins. The first section is for 'HTML Post Transmittal Plug-In', which is checked. It has a text field for 'URL to post as HTML form' containing 'http://www.example.com/form.php', a text field for 'Button Label' containing 'Post to Web Service', and a 'Display Button' dropdown menu set to 'On New Form & Drafts'. The second section is for 'JSON Post Transmittal Plug-In', which is also checked. It has a text field for 'URL to post JSON object' containing 'http://www.example.com/form.php', a text field for 'Button Label' containing 'Post to Web Service', and a 'Display Button' dropdown menu set to 'On New Form & Drafts'. At the bottom of the dialog, there is a checkbox labeled 'After sending form, keep copy in the draft folder.' which is unchecked. A 'Close' button is located at the bottom right of the dialog.

The value entered under '*HTML Parameter Name*' or '*JSON Parameter Name*' for each control under '*Web Services*' tab contains the data. For the JSON posting, there is only one parameter called '*json*' that is sent. It contains the JSON object of the form's data.

Processing the data returned by the form is up to the backend developer, however how a response is sent back to the users is structured.

## RESPONDING TO FORMENTRY TOUCH

If the web site parses the data and responds with only an HTTP 200 header and nothing else, FormEntry Touch assumes everything was successfully posted. Consequently, no message is sent to the user on the device.

If the web site is down or responds with a non HTTP 200 header, FormEntry Touch assumes an error has occurred and posts a message to the user stating there was an error during posting. In this case, the form data is stored in the '*Drafts*' folder.

To send a success or failure reply to *FormEntry Touch*, send an HTTP 200 header in a structured UTF-8 XML format. The XML response is parsed by *FormEntry Touch*. Here is an example:

```
<?xml version="1.0" encoding="UTF-8"?>
<response>
  <success>true</success>
  <show_alert>true</show_alert>
  <title>Successful Post</title>
  <message>The form survey has been successful posted.</message>
  <button_label>Ok</button_label>
</response>
```



The `<success/>` element contains a string of 'true' or 'false'. You can also use '1' or '0'. If the element is 'true', then FormEntry Touch moves the form from 'Drafts' to the 'Sent' folder. If the element is 'false', FormEntry Touch keeps the form entry in the 'Drafts' folder.

The `<show_alert/>` element contains a string of 'true' or 'false'. You can also use '1' or '0'. If element is 'true', FormEntry Touch shows a pop-up message to the users. If the element is 'false', it does not show the pop up message.

The `<title/>` element, contains a string for the 'Title' of the pop up message.

The body of the pop up message string is contained in the `<message/>` element.

The button on the pop up message simply closes the message and does nothing else. The button text is contained in the `<button_label/>` element string.

All the elements described previously must to be contained in the parent `<response/>` element.

## INTERNAL SSL DIGITAL CERTIFICATES

A server certificate can be installed on each device if the data is being posted to a secure socket (https://) that is **not** public facing. The device is not be able to post to the *secure internal web server* until the certificate is installed on the device.

The certificate can be installed in a number of ways. A simple approach is to first save the digital certificate from a web browser (typically double clicking the lock icon). Then email the certificate as an attachment to an account on the device. Install the certificate by selecting it from the Mail app on the device.

## EXPORTING TO PHP ARCHIVE

Within FormEntry for Mac, each project has the capability of exporting the entire project to a PHP Archive by selecting File -> Export... -> Export to PHP Archive from the Main Menu. The export plug-in allows you to generate all the necessary web server side code that can collect the iOS user's data.

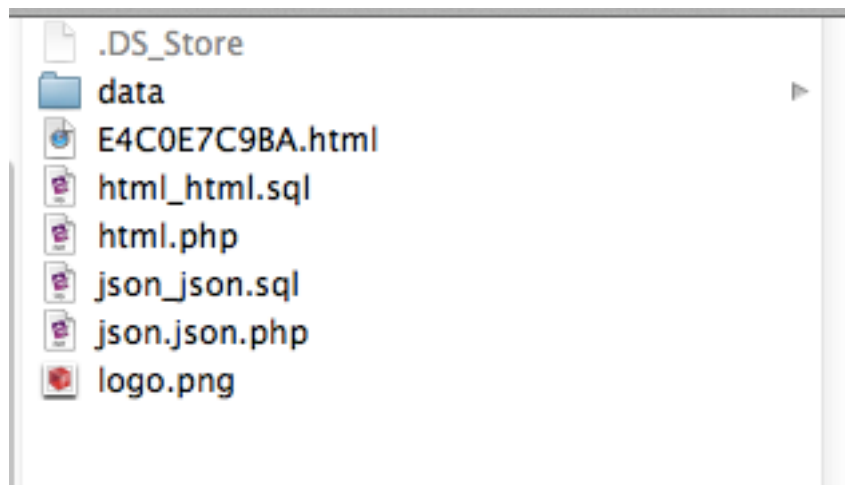
The exported code is treated as a “Reference Implementation” and can be modified to meet your business requirements.

The Linux Apache MySQL PHP (LAMP) generated code includes a MySQL script to create your database, a PHP file to

handle the data and place the data into the MySQL database. The generated PHP file can also create a CSV file on the server as well as insert into SQLite. You can also have the data be sent via email from the server. You should open the .php and .sql files in text editor to configurable.

There is a plain .html file that gets generated that can be used for testing. The structure within this HTML is exactly how FormEntry Touch will post the information via the HTML post.

Also, the “data” folder gets generated that holds your uploaded form assets, which can be your photos and signatures. There is also a “csv” and “sqlite” folder that holds your data if you choose to enable this from within the PHP script.



To configure your web server is out of the scope of this manual, however, the basic steps are outlined below:

- 1) When you are satisfied with your form you need to export with PHP Archive.
- 2) You will need to create a MySQL database on your web server with a user name and password. Consult with your web hosting company on the proper steps. If you are not using MySQL to store the data, you can skip to step 6.
- 3) You will to edit the .php script under the MySQL section with the host name, database name, user name and password and you will need to enable MySQL.
- 4) You will need to edit the .php script and double check the database table name matches the database table name within the .sql file.
- 5) You will need to install the .sql script on your web server instance of MySQL, consult with your web hosting provider.
- 6) Upload all files to your web server, including the directories and you will need to maintain the directory hierarchy. If you are uploading photos and signature, make sure the directory “media” is writable.
- 7) Open a web browser and go to the URL were you placed files and point to the .php file. For example, it might be “<http://www.example.com/formentry.php>” (This is the URL where your iOS users will be posting). You should see a dialog box asking for authentication, use the default “admin” for user name and “admin” for password. You can change these if you so desire. As your users post data, you should be able to see the information posted here.

If you have the Automatically check verification enabled, the PHP Archive will export two files for you to be used in your testing.

The screenshot shows the 'General Info' tab of the FormEntry application. It contains the following fields and settings:

- Form Name:** Versioning
- Version:** 1.0
- Form ID:** E4C0E7C9BA
- ☒ Automatically check verification Web URL for latest Form Version.
- Web URL:** <http://10.0.1.25:8888/version/CheckiOSVersion.php>
- ☐ Prevent user from loading a New Form if a newer Form Version exists.

These two generated files are 1) PHP versioning check and the other 2) is a HTML test file on how FormEntry Touch will post the data to your web service. In the image to the side, the are called “CheckiOSVersion.html” and “CheckiOSVersion.php” respectively.

The versioning checking PHP file can be placed upon your web server. Open the PHP file in a text editor for further details and configuration.



## **HANDLING LOCALIZED KEYWORD PARAMETERS AND MYSQL DATABASE COLUMNS**

PHP can handle all the UTF-8 parameter names and values coming from FormEntry Touch, however MySQL has limitations on column name conventions. As such, it is critical that you review all your parameters and all your MySQL table columns and make sure they are the same. For example, the French column name of “entrée\_id” cannot be used in MySQL.

By default, if the five Hidden controls for keyword parameters (Form ID, Entry ID, Device UUID8, Creation and Modified) are enabled, FormEntry Touch will post these parameters in the localized language. For convenience, it is recommended that these five keywords be changed to “English” ASCII text strings.

***Recommendation: It is recommended that all parameters have a matching MySQL table column name in ASCII text.***

The exported .php script will try and place the localized keyword parameters into the appropriate table column. Note that the “created” and “modified” parameters should be placed in the “entry\_created” and “entry\_modified” table columns as it relates when the entry was created and modified on the iOS device. The table columns of “created” and “modified” should be applied to when that table’s record was created and when that record was modified.

See the follow chart on how these parameters are posted via the HTML Post Transmittal Plugin with a localized language:

LANGUAGE	ENGLISH KEYWORD PARAMETER	LOCALIZED KEYWORD PARAMETER	RECOMMEND MYSQL COLUMN NAME
English	entry_id	entry_id	entry_id
French	entry_id	entrée_id	entry_id
Dutch	entry_id	id-invoer	entry_id
German	entry_id	eintrags-id	entry_id
Spanish	entry_id	id._de_la_entrada	entry_id
Italian	entry_id	id_entrata	entry_id
Portuguese	entry_id	id_da_entrada	entry_id
Chinese	entry_id	项id	entry_id
Korean	entry_id	항목_id	entry_id
Japanese	entry_id	エントリー_id	entry_id
English	form_id	form_id	form_id
French	form_id	identification_formulaire	form_id
Dutch	form_id	id-formulier	form_id
German	form_id	form-id	form_id
Spanish	form_id	id._del_formulario	form_id
Italian	form_id	id_modulo	form_id
Portuguese	form_id	id_do_formulário	form_id
Chinese	form_id	表格id	form_id
Korean	form_id	양식_id	form_id
Japanese	form_id	フォームid	form_id
English	device_uuid8	device_uuid8	device_uuid8
French	device_uuid8	appareil_uuid8	device_uuid8

LANGUAGE	ENGLISH KEYWORD PARAMETER	LOCALIZED KEYWORD PARAMETER	RECOMMEND MYSQL COLUMN NAME
Dutch	device_uuid8	apparaat_uuid8	device_uuid8
German	device_uuid8	geräte-uuid8	device_uuid8
Spanish	device_uuid8	dispositivo_uuid8	device_uuid8
Italian	device_uuid8	dispositivo_uuid8	device_uuid8
Portuguese	device_uuid8	dispositivo_uuid8	device_uuid8
Chinese	device_uuid8	设备uuid8	device_uuid8
Korean	device_uuid8	장치_uuid8	device_uuid8
Japanese	device_uuid8	デバイス_uuid8	device_uuid8
English	created	created	entry_created
French	created	créé	entry_created
Dutch	created	gemaakt	entry_created
German	created	erstellt	entry_created
Spanish	created	creado	entry_created
Italian	created	creato	entry_created
Portuguese	created	criado	entry_created
Chinese	created	已创建	entry_created
Korean	created	생성됨	entry_created
Japanese	created	作成済み	entry_created
English	modified	modified	entry_modified
French	modified	modifié	entry_modified
Dutch	modified	aangepast	entry_modified
German	modified	geändert	entry_modified



LANGUAGE	ENGLISH KEYWORD PARAMETER	LOCALIZED KEYWORD PARAMETER	RECOMMEND MYSQL COLUMN NAME
Spanish	modified	modificado	entry_modified
Italian	modified	modificato	entry_modified
Portuguese	modified	modificado	entry_modified
Chinese	modified	已修改	entry_modified
Korean	modified	수정됨	entry_modified
Japanese	modified	修正済み	entry_modified

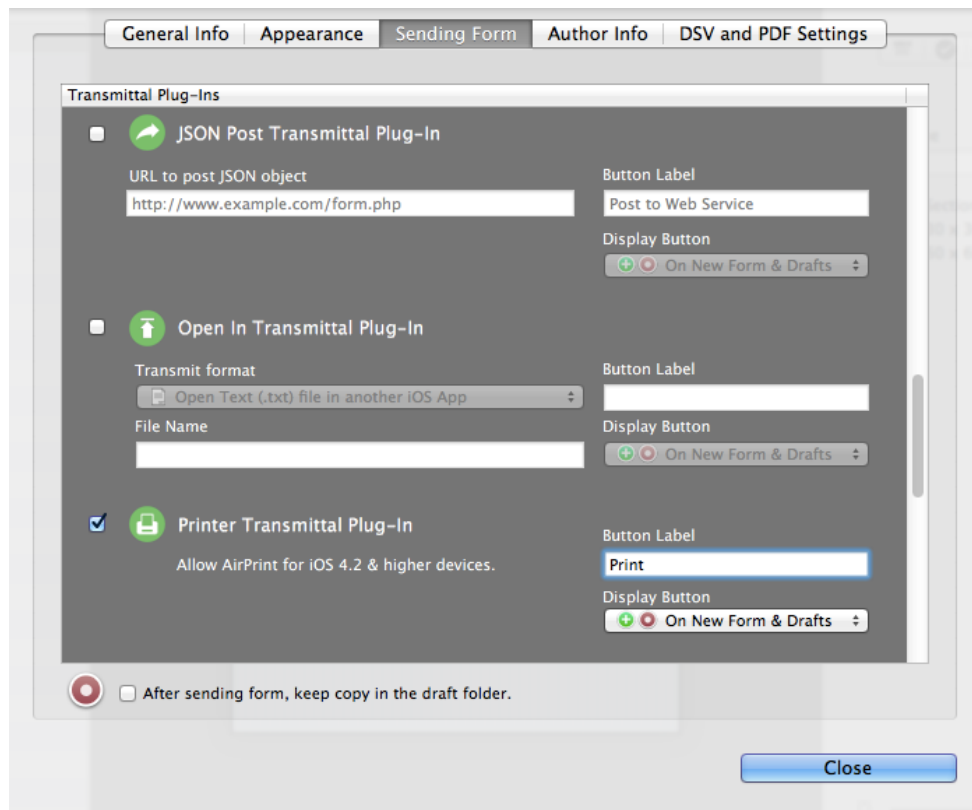
## Printer (AirPrint) Transmittal Plug-In

The Printer Transmittal Plug-In, is a paid plug-in that first must be purchased from within FormEntry for Mac under the FormEntry -> Buy Plug-Ins... menu under the Main Menu. This is an In-App purchase within FormEntry for Mac and you must use your Apple ID to make the transaction.

iOS users can print the form directly to a Wi-Fi printer that is compatible with AirPrint.

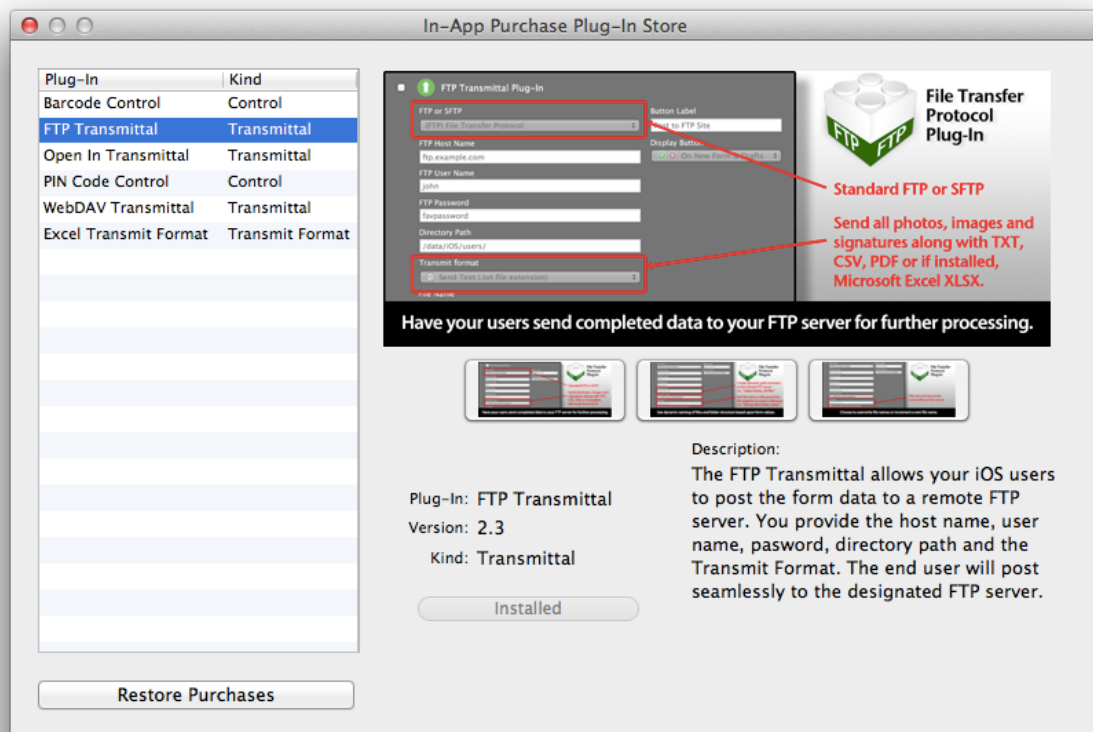
To enable AirPrint printing, click the Settings icon on the 'Sending Form' tab. Select the Printer Transmittal Plug-In in the list and enter a label for the button.

You will need to set your printing layout with the PDF layout editor. Select the DSV and PDF Settings from the Settings screen. See the Using PDF Technology for laying out and updating your PDF files.



## **File Transfer Protocol (FTP) Transmittal Plug-In**

The FTP Transmittal Plug-In, is a paid plug-in that first must be purchased from within FormEntry for Mac under the FormEntry -> Buy Plug-Ins... menu under the Main Menu. This is an In-App purchase within FormEntry for Mac and you must use your Apple ID to make the transaction.



FormEntry for Mac sends four different types of formats in a FTP or SFTP upload: 1) Plain text, .txt file 2) Attached DSV (Delimited Separated File, a.k.a. a CSV file) or 3) PDF and if installed 4) Microsoft Excel (.xlsx). All photos, signatures and images are uploaded separately in their respected designated .jpg or .png file format.

The screenshot shows the 'Transmittal Plug-Ins' dialog box within the 'Sending Form' tab. The 'FTP Transmittal Plug-In' is selected. The settings are as follows:

Field	Value
FTP or SFTP	(FTP) File Transfer Protocol
FTP Host Name	ftp.example.com
FTP User Name	john
FTP Password	favpassword
Directory Path	/data/iOS/users/
Transmit format	Send Text (.txt file extension)
File Name	sample_filename
Overwrite files with identical file names.	<input type="checkbox"/>
Button Label	Post to FTP Site
Display Button	On New Form & Drafts

At the bottom, there is a checkbox labeled 'After sending form, keep copy in the draft folder.' which is currently unchecked. A 'Close' button is located at the bottom right of the dialog.

From within the Sending Form screen, you can provide the FTP Host Name, FTP User Name, FTP Password, Directory Path and Button Label.

For the FTP Host Name value, it can be a domain name or IP address. The FTP User Name is the account name for the FTP server and the FTP Password is the account's password.

Note: When setting the User Name and Password, refrain from using special characters within the name, such as "@" or "\$". These characters tend to cause some issues when logging in successfully.

The Directory Path is the file path to place the uploading data from FormEntry Touch. FormEntry Touch will create the Directory Path if there is no path that exists.

## DIRECTORY PATHS

**FTP** Directory paths are *relative* paths, taken from when the user logs into the directory. **SFTP** are taken from the *absolute* path, taken from the full path of the server.

Both the FTP and SFTP can deliver create the directory paths if the paths do not exists. You can also have the paths dynamically created based upon user's input as well as use reserved and keyword parameter IDs.



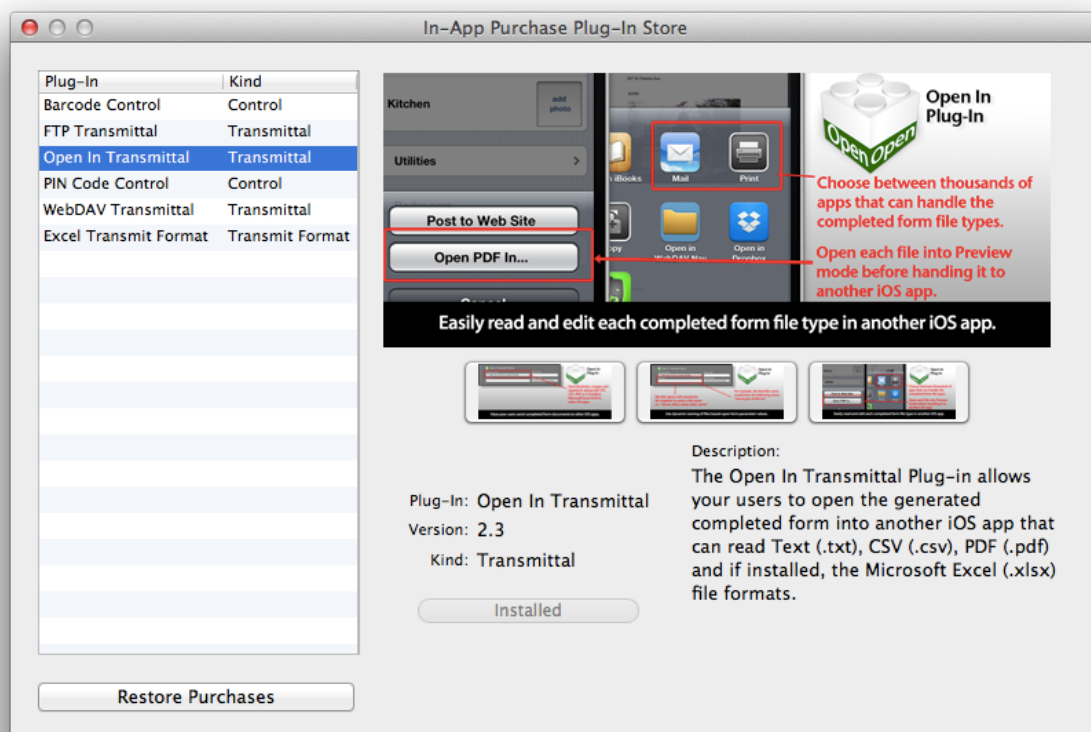
For example, the 'last\_name' and 'entry\_id' parameter IDs are used in the Directory Path image above. If these folders do not exists they will be created on the server.

## FILE NAMES

File names follow the same stye of naming convention. They can be based upon standard file names and/or mixed in with parameter IDs. They will be uploaded to the Directory Path.

## Open In Transmittal Plug-In

The Open In Transmittal Plug-In, is a paid plug-in that first must be purchased from within FormEntry for Mac under the FormEntry -> Buy Plug-Ins... menu under the Main Menu. This is an In-App purchase within FormEntry for Mac and you must use your Apple ID to make the transaction.



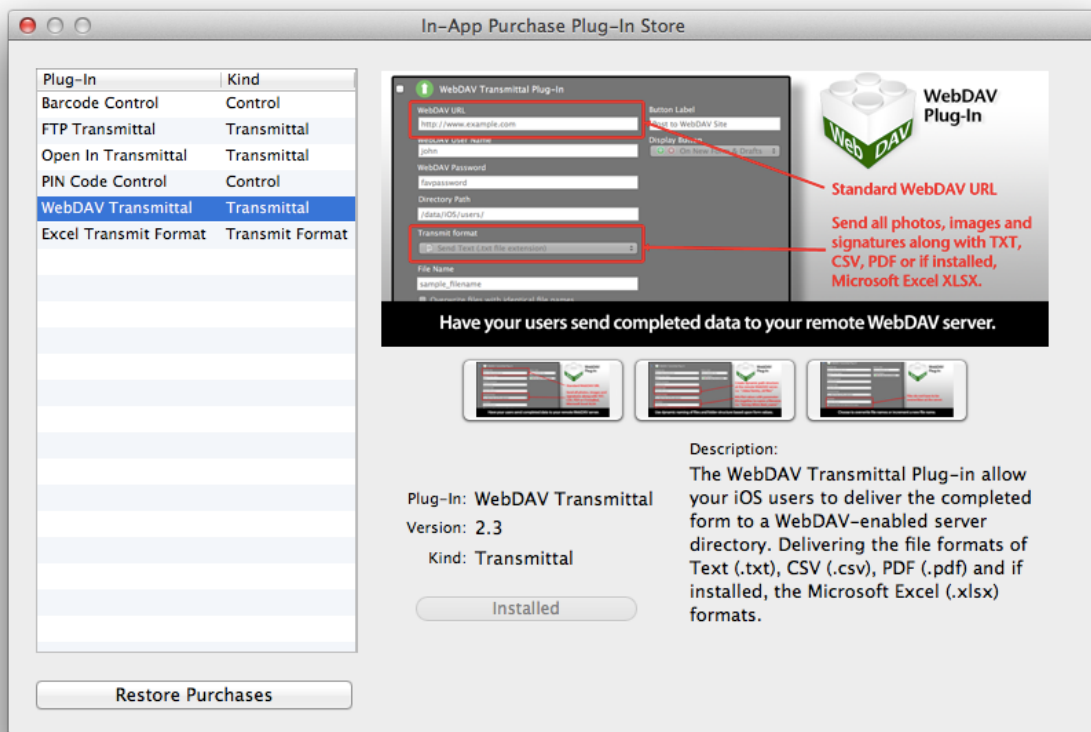
FormEntry for Mac sends four different types of formats to be opened in another iOS application installed on the device: 1) Plain text, .txt file 2) Attached DSV (Delimited Separated File, a.k.a. a CSV file) or 3) PDF and if installed 4) Microsoft Excel (.xlsx). All photos, signatures and images can be opened in separately in their respected designated .jpg or .png file format.

## FILE NAMES

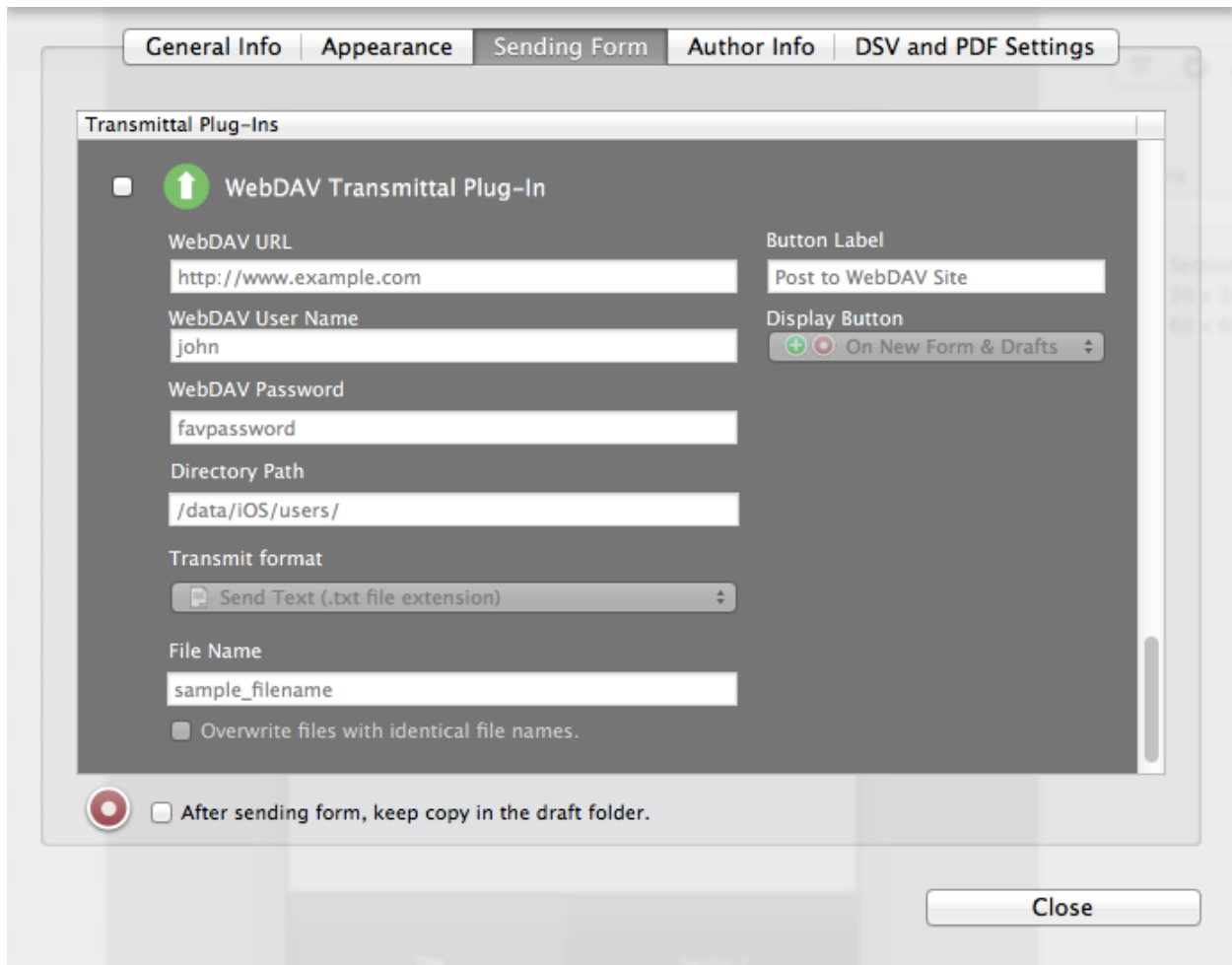
File names follow the same stye of naming convention. They can be based upon standard file names and/or mixed in with parameter IDs.

## WebDAV Transmittal Plug-In

The WebDAV Transmittal Plug-In, is a paid plug-in that first must be purchased from within FormEntry for Mac under the FormEntry -> Buy Plug-Ins... menu under the Main Menu. This is an In-App purchase within FormEntry for Mac and you must use your Apple ID to make the transaction.



FormEntry for Mac sends four different types of formats in a WebDAV upload: 1) Plain text, .txt file 2) Attached DSV (Delimited Separated File, a.k.a. a CSV file) or 3) PDF and if installed 4) Microsoft Excel (.xlsx). All photos, signatures and images are uploaded separately in their respected designated .jpg or .png file format.



From within the Sending Form screen, you can provide the WebDAV URL, WebDAV User Name, WebDAV Password, Directory Path and Button Label.

For the WebDAV URL value, it can be a domain name or IP address. The WebDAV User Name is the account name for the WebDAV server and the WebDAV Password is the account's password.

The Directory Path is the file path to place the uploading data from FormEntry Touch. FormEntry Touch will create the Directory Path if there is no path that exists.

## DIRECTORY PATHS

WebDAV Transmittal Plug-In can deliver create the directory paths if the paths do not exists. You can also have the paths dynamically created based upon user's input as well as use reserved and keyword parameter IDs.



Directory Path

/data/\$last\_name/\$entry\_id/

For example, the 'last\_name' and 'entry\_id' parameter IDs are used in the Directory Path image above. If these folders do not exist they will be created on the server.

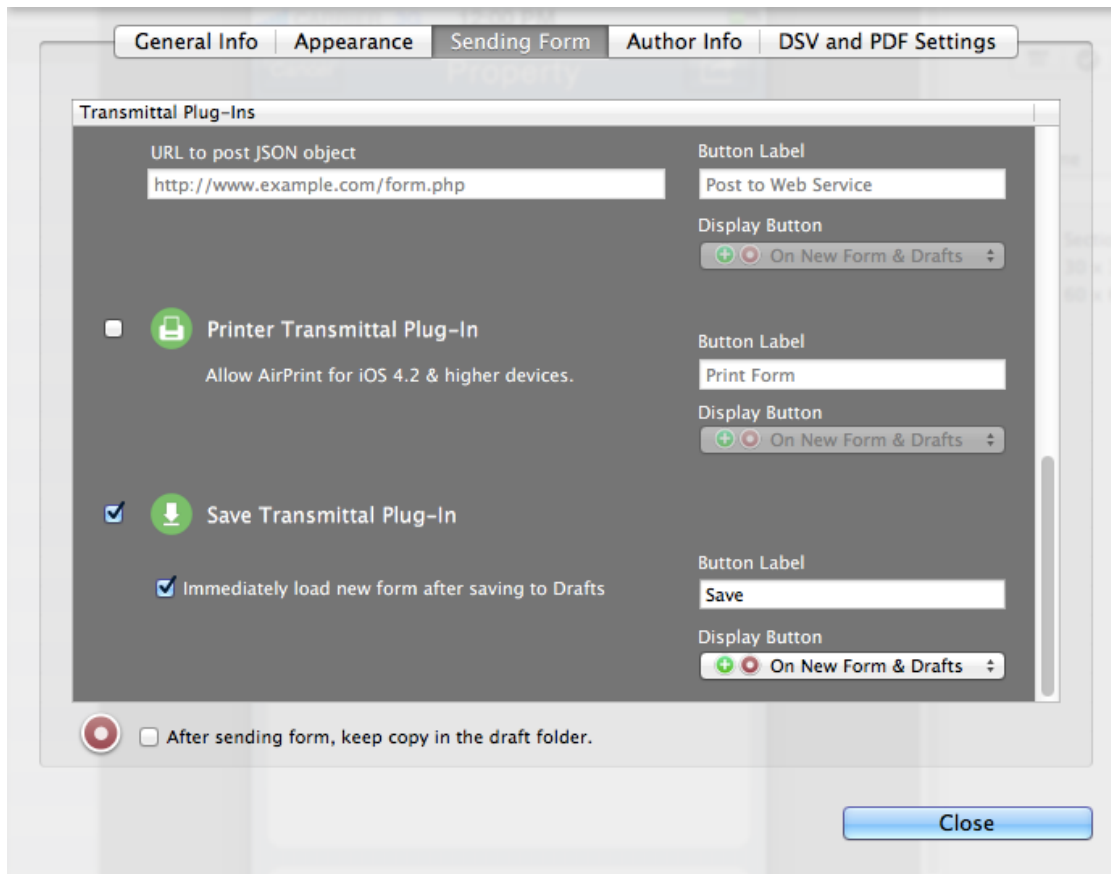
## **FILE NAMES**

File names follow the same style of naming convention. They can be based upon standard file names and/or mixed in with parameter IDs. They will be uploaded to the Directory Path.

## **Save Transmittal Plug-In**

The Save Transmittal Plug-In, is a paid plug-in that first must be purchased from within FormEntry for Mac under the FormEntry -> Buy Plug-Ins... menu under the Main Menu. This is an In-App purchase within FormEntry for Mac and you must use your Apple ID to make the transaction.

The Save Transmittal Plug-In allows your users to directly save to the drafts folder. This is useful if you are collecting multiple sets of consistent information, such as inventory tracking warehouse with the Barcode Plug-In.



The Save Transmittal Plug-In also always you to immediately load a new form after selecting your Save action.

# KEYWORD AND RESERVED PARAMETER IDS

**This section discusses the different reserved form parameter IDs that you can use within form app.**

FormEntry Touch allows you to access values from within your form app and from the device during runtime. Also, you can use the five keyword parameter names in combination with the reserved parameter IDs. Keyword parameters are based upon the five hidden controls and have the ability to become optional and not pull the values from the device. It is recommended to keep the five keyword parameters on your form app.

You can use these parameters in your the Rule Builder, naming files and folder paths by appending the “\$” in the prefix. To use parameters, see the different sections in this manual for further details.

The following tables discusses the parameter along with descriptions that are available during runtime of your form app.

## **RESERVED PARAMETER DESCRIPTIONS**

PARAMETER	TYPE	DESCRIPTION
form_id	Keyword Parameter	This parameter provides the 10-digit Form ID value found under the Settings screen under the General Info tab. This is a unique identifier. <b>Example: 27AA552FB3</b>
entry_id	Keyword Parameter	This parameter provides the 8-digit Entry ID value. Each form entry in your form app generates a unique identifier. <b>Example: 1D284A3C</b>
device_uuid8	Keyword Parameter	This parameter pulls the 8-digit unique identifier from the Settings within FormEntry Touch. This ID is unique per device. Note, if the user were to reinstall FormEntry Touch, a new Device UUID 8 would be generated. <b>Example: 6289A693</b>
created	Keyword Parameter	This parameter is provides when the entry was created at GMT time. <b>Example: 2013-04-27 10:28:01 +0000</b>
modified	Keyword Parameter	This parameter is provides when the entry was modified at GMT time. <b>Example: 2013-04-27 10:29:21 +0000</b>
form_name	Reserved Parameter	This parameter provides the Form Name value found under the Settings screen under the General Info tab. <b>Example: Property Inspection</b>

PARAMETER	TYPE	DESCRIPTION
form_version	Reserved Parameter	This parameter provides the Form Version value found under the Settings screen under the General Info tab. <b>Example: 1.0</b>
form_auto_increment	Reserved Parameter	This parameter provides the Form Auto Increment number which is based upon the start value found under the Settings screen under the General Info tab. <b>Example: 1001</b>
form_unix_timestamp	Reserved Parameter	This parameter provides the 10-digit UNIX timestamp at runtime. <b>Example: 1367061394</b>
form_total_sent_count	Reserved Parameter	This parameter provides the total number of forms that have been sent for the form. <b>Example: 12</b>
form_sent_count	Reserved Parameter	This parameter provides the total number of forms that are in the Sent Folder. <b>Example: 3</b>
form_drafts_count	Reserved Parameter	This parameter provides the total number of forms that are in the Drafts Folder. <b>Example: 10</b>

PARAMETER	TYPE	DESCRIPTION
form_trash_count	Reserved Parameter	This parameter provides the total number of forms that are in the Trash Folder. <b>Example: 4</b>
form_author	Reserved Parameter	This parameter provides the First Name and Last Name values found under the Settings screen under the Author Info tab. <b>Example: Jonathan Freeman</b>
form_email	Reserved Parameter	This parameter provides the Email value found under the Settings screen under the Author Info tab. <b>Example:</b> <a href="mailto:inquiry@widgetpress.com">inquiry@widgetpress.com</a>
form_website	Reserved Parameter	This parameter provides the Web Site value found under the Settings screen under the Author Info tab. <b>Example: http://www.widgetpress.com</b>
form_company	Reserved Parameter	This parameter provides the Email value found under the Settings screen under the Author Info tab. <b>Example: Widget Press, Inc.</b>
form_address1	Reserved Parameter	This parameter provides the Address 1 value found under the Settings screen under the Author Info tab. <b>Example: 537 St. Charles Ave, NE</b>

PARAMETER	TYPE	DESCRIPTION
form_address2	Reserved Parameter	This parameter provides the Address 2 value found under the Settings screen under the Author Info tab. <b>Example: Suite 1010</b>
form_city	Reserved Parameter	This parameter provides the City value found under the Settings screen under the Author Info tab. <b>Example: Atlanta</b>
form_state	Reserved Parameter	This parameter provides the State value found under the Settings screen under the Author Info tab. <b>Example: Georgia</b>
form_zip	Reserved Parameter	This parameter provides the Zip/Postal value found under the Settings screen under the Author Info tab. <b>Example: 30308</b>
form_country	Reserved Parameter	This parameter provides the Country value found under the Settings screen under the Author Info tab. <b>Example: USA</b>

## ACCESSING PARAMETER IDS

Within FormEntry for Mac, you have the ability to access these parameter values set by FormEntry Touch at different locations in your form app. You can access them at file names, Rule Builder, emails and folder paths.

In the image below, you can set different parameters for the Email Transmittal Plug-In. This convention is applied to all other Transmittal Plug-Ins.



For the Default Email Address, there two addresses, first being a hardcoded email and the other will pull the value from the Text Field that contains the 'email\_address' parameter id. The prefix contains "\$" must be used in front of the parameter id. This tells FormEntry Touch this is a parameter id and go get the value, if any.


The screenshot shows the 'Transmittal Plug-Ins' settings window. The 'Email Transmittal Plug-In' is enabled. The 'Default Email Address' field contains 'inquiry@widgetpress.com, \$email\_address'. The 'Button Label' is 'Email'. The 'Default Subject Line' is 'New Order from \$first\_name \$last\_name'. The 'Display Button' is set to 'On New Form & Drafts'. The 'Transmit Format' is 'Send PDF document (.pdf file extension)'. The 'File Name' is '\$form\_id.\$form\_unix\_timestamp'. There is an unchecked checkbox for 'After sending form, keep copy in the draft folder.' and a 'Close' button at the bottom right.

You can also mix in regular words with parameter ids as in the Default Subject Line for the Email Plug-In. For the File Name, you can create file names based upon parameters that have been filled out by the user or retrieve the reserved and keyword parameter IDs.

With the FTP or WebDAV Transmittal Plug-Ins, you can also dynamically create the folder path based upon values from parameter IDs. The following image below show how a directory folder can be created from the Text Field of "last\_name".

General Info | Appearance | Sending Form | Author Info | DSV and PDF Settings

Transmittal Plug-Ins

☒  WebDAV Transmittal Plug-In

WebDAV URL

WebDAV User Name

WebDAV Password

Directory Path

Transmit format

File Name

☐ Overwrite files with identical file names.

Button Label

Display Button  
☒ ☐ On New Form & Drafts

☐ After sending form, keep copy in the draft folder.

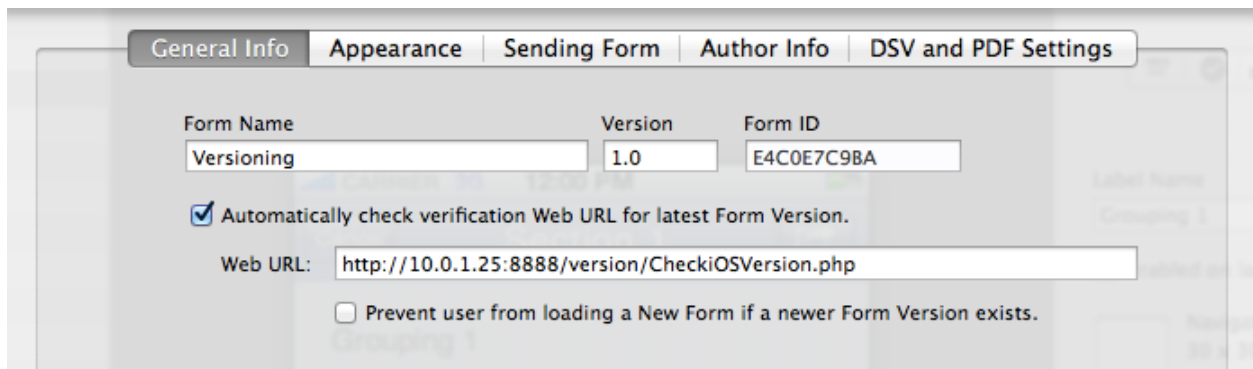
Close

# CHECK FORM VERSION

**This section describes how to handle the checking Form Version of your forms with users, responding to your users and even preventing them from loading a new form.**

FormEntry Touch provides you with a web service *hook* into your form app before your form loads to the current form screen. Meta data from the end user's device will be posted from FormEntry Touch to a specified URL, provided by you. You can analyze this information and provide some actions with a message back to the user.

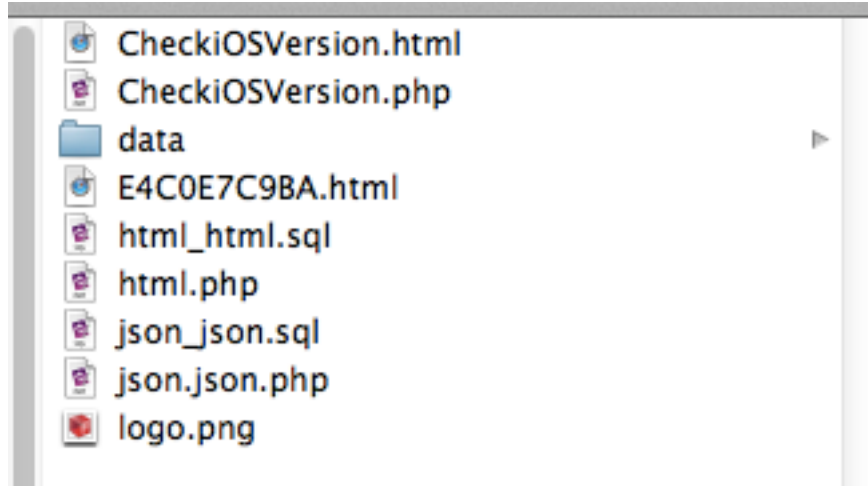
Under General Info tab from the Settings sheet, you can place a Version, i.e. 1.0, 2.0, 2.2, etc at the Version Text Field.

The screenshot shows the 'General Info' tab of the FormEntry Touch settings. It features three input fields: 'Form Name' with the value 'Versioning', 'Version' with the value '1.0', and 'Form ID' with the value 'E4C0E7C9BA'. Below these fields is a checked checkbox labeled 'Automatically check verification Web URL for latest Form Version.' followed by a 'Web URL:' label and a text field containing 'http://10.0.1.25:8888/version/CheckiOSVersion.php'. At the bottom, there is an unchecked checkbox labeled 'Prevent user from loading a New Form if a newer Form Version exists.'.

To enable automatic verification of your form version, you will need to enable the checkbox at “Automatically check verification Web URL for latest Form Version” and you will need to provide a fully qualified web URL that will respond back with an XML response.

When the iOS user selects your form on FormEntry Touch, a web POST will be posted at your Web URL address. This is standard web HTTP POST that will have specific parameters for you to check.

If you were to Export the project to a PHP Archive and had the “Automatically check verification Web URL for latest Form Version” checked, you will see two files generated. One being a PHP file that contains a working reference



implementation of how

to check the incoming parameters. The other, a HTML file that you can use to test simulate how FormEntry Touch will post the parameters to your web server.

FormEntry Touch will post the following parameters to your web service URL, “device\_uuid8”, “form\_version”, “form\_id”, “forms\_sent[]”, “forms\_drafts[]”, “forms\_trash[]” and if installed the “pin\_code” (based upon the parameter id of the PIN Code) parameter id.

The “device\_uuid8” is based upon the Device UUID8 from the device’s Settings screen within FormEntry Touch.

The “form\_version” is the version of your form, originally assigned from the Version field on the General Info tab within FormEntry for Mac.

The “form\_id” is the unique identifier that is created for your form, found under the General Info tab within FormEntry for Mac.

The “pin\_code” parameter is sent when the user logs into the form using their 4-digit passcode number. The PIN Code Plug-In must be purchased from within the In-App Purchase Plug-In Store within FormEntry for Mac. This parameter is upon the parameter id of the control and might not be the actually named “pin\_code”.

The “forms\_sent[]”, “forms\_drafts[]” and “forms\_trash[]” are parameter arrays which contain a collection of Entry IDs within each of their respected folder. This is useful if you want provide a message to the user about a specific Entry ID that has been already Sent, in Drafts state or has been Trashed.

## **EXAMPLE OF USING THE CHECK FORM VERSION**

Let's say a field agent had completed and sent a form to your database via the HTTP Post transmittal. The field agent's form would be archived in their Sent folder. A day later, your manager notices that there an error on the form for some Customer Information is not completed. The field agent must update the form with the corrections and must have the form resigned by the customer.

By using this form check version mechanism, you have a chance to inform the field agent about the correction required. When the field agent attempts to load the form remotely, they will post to your web service URL all the information you need to determine the required message.

By inspecting the POST parameters, you can determine the "device\_uuid8" is the field agent's device. By inspecting the "form\_id", you can determine that this specific form is needing attention. By inspecting the Entry IDs within the "forms\_sent[]", you can determine that the field agent has a specific Entry ID within their sent form that needs correction.

You can then send a Pop-up alert back to the user with a custom message stating "Corrections required to Entry ID: 1234ABCD. Please move this Entry from the Sent folder back into Drafts and complete Customer Info again, resign and submit."

# USING FORMENTRY URL SCHEME

**This chapter explains how to remotely alter the form based upon a defined URL schema API for FormEntry Touch 2.3.5 and higher.**

The FormEntry URL Scheme was added to FormEntry Touch 2.3.5 as a way to remotely interact with forms installed within FormEntry Touch. By passing the iOS user a URL (via website or email or even as a local hyperlink in a local DropBox file), you can pre-populate data into a new form. One iOS devices running FormEntry Touch 2.3.4 and less, this URL will be ignored.

For online use, this process could even be dynamically created at a web server and delivered to users via a web site. However, the URL scheme is not required to be online. By clicking on a link within archived email or a flat HTML file can open FormEntry Touch and add a new form with values.

If you are an iOS developer, you can even add the FormEntry URL Scheme into your app and deliver data into a new form.

This API will be expanded upon in future releases and that new functionality will be documented here.

## Remotely pre-populate a Form with data

Here's an example of a FormEntry URL Scheme that we will use for the remainder of this chapter:

```
formentry://C52802BB5A/new?checkmark_4=1&switch_2=1&ratings_8=3&ratings_8.labelName=Rating&text_field_1.isEnabled=1&text_field_1=Jonathan Freeman
```

You could place the above URL within a regular HTML anchor link:

```
<a href="formentry://C52802BB5A/new?checkmark_4=1&switch_2=1&ratings_8=3&ratings_8.labelName=Rating&text_field_1.isEnabled=1&text_field_1=Jonathan Freeman">New Form Entry</a>
```

## FORMENTRY URL SCHEME EXAMPLE

The FormEntry URL Scheme name starts off with the protocol. By simply calling this protocol with nothing else, you can open FormEntry Touch app:

```
formentry://
```

After the scheme name, we place the Form ID as the host name. This tells FormEntry Touch what form we want to interact with:

```
formentry://C52802BB5A
```

After the Form ID, we pass the path of 'new' separated by a '/'. This tells FormEntry Touch that we want to create a new form entry using this Form ID:

```
formentry://C52802BB5A/new
```

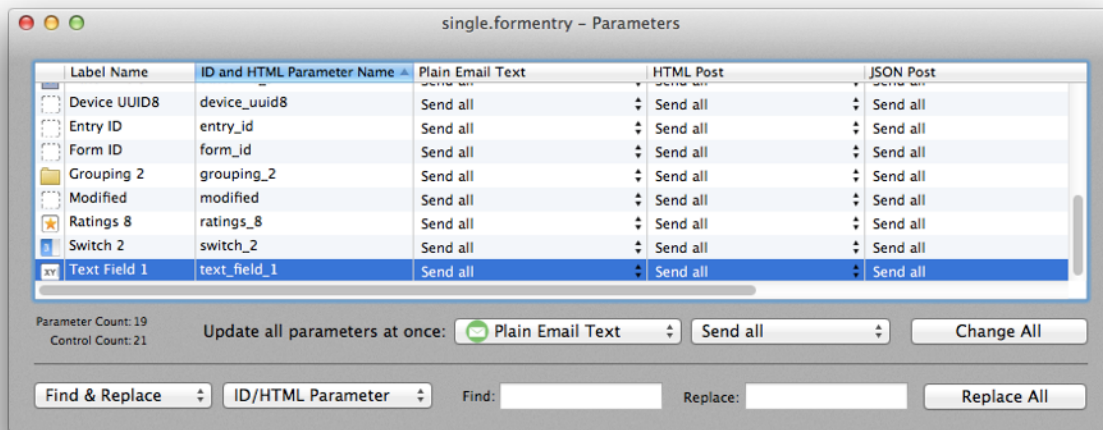
After the new action path, the rest is an optional query "?" that is used to inject data into the new form:

```
formentry://C52802BB5A/new?checkmark_4=1&switch_2=1&ratings_8=3&ratings_8.labelName=Rating&text_field_1.isEnabled=1&text_field_1=Jonathan Freeman
```

The query uses the standard URL queries that is traditional used in HTTP. Each parameter is separated by a "&" and each parameter contains an "=" to the value that you would like to populate that control with.

Notice the ".labelName" and ".isEnabled" dot syntax after the parameter names. This tells FormEntry Touch that you want to dynamically change the Label Name and to enable or disable the control. The Label Name takes a string and latter takes string boolean of "0" to disable or "1" to enable.

All URL query parameters are based upon the ID/HTML Parameter Name within your FormEntry for Mac project:



## Control configuration through URL Scheme

There are some limitations on what can be done with the URL Scheme and the controls within your form. This control table provides a breakdown what can be done from adding values, changing labels and enabling/disabling the control on new form entry creation.



CONTROL	CHANGE VALUE	URL VALUE TYPE	CHANGE LABEL	CHANGE ENABLED
Barcode	No	String "123456789"	Yes	Yes
PIN	No	N/A	Yes	N/A
Checkmark	Yes Changes State	Boolean "0" or "1"	Yes	Yes
Date & Time	Yes	GMT Date String YYYY-DD-MM HH:mm:ss +0000 "1982-02-12 16:00:00 +0000"	Yes	Yes
Hidden	Yes	String	Yes	N/A
Image	No	N/A	Yes	Yes
Location	No	N/A	Yes	Yes
On-Off Switch	Yes Changes State	Boolean ("0" or "1")	Yes	Yes
Photo Picker	No	N/A	Yes	Yes
Ratings	Yes	Integer "0", "1", "2", "3", "4", "5"	Yes	Yes
Signature	No	N/A	Yes	Yes
Spin Wheel Picker	Yes	String	Yes	Yes
Spin Wheel Item*	Yes	String	N/A	N/A
Text	Yes	String	Yes	Yes
Text Box	Yes	String	Yes	Yes
Text Field	Yes	String	Yes	Yes
Web Browser	N/A	N/A	Yes	Yes

\*Note: That Spin Wheel Items do not have Parameter IDs. To set a new list of Spin Wheel Items to a Spin Wheel Picker, use the Parameter ID of the Spin Wheel Picker appending by each parameter ID with "[ ]".

For example, the following FormEntry URL Scheme would populate a Spin Wheel Picker with the following items. Icons cannot be used in the URL.

```
formentry://C52802BB5A/new?  
spin_wheel_picker_1[ ]=Apple&spin_wheel_picker_1[ ]=Banana&spin_wheel_picker_1[ ]=Orange&spin_wheel_picker_1[ ]=Watermelon
```

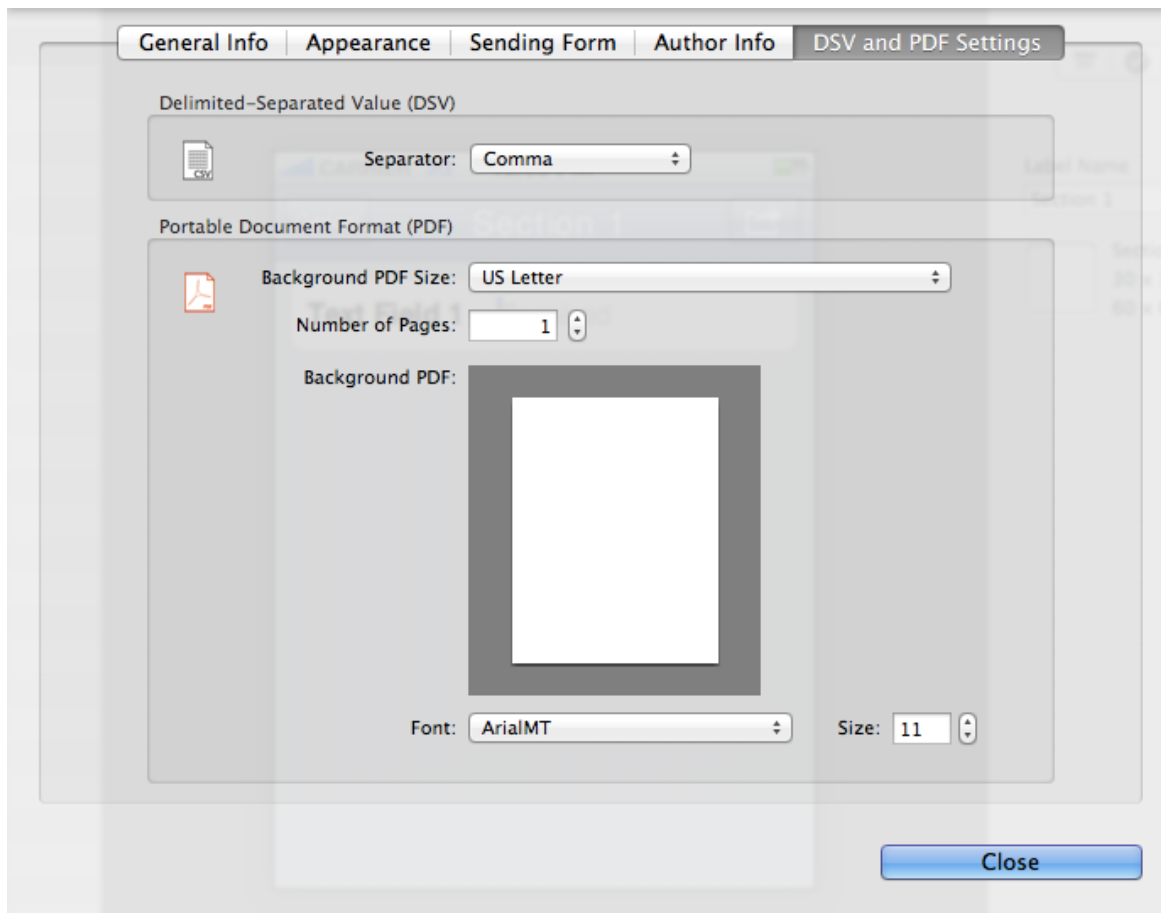
The Spin Wheel Items from the FormEntry URL will be attached to the Spin Wheel Picker even after saving it to drafts and when the device goes offline from the Internet.

# USING PDF TECHNOLOGY

**This chapter explains how to use existing PDFs or generate your own PDFs to be used when the form is submitted via email or when printing to AirPrint.**

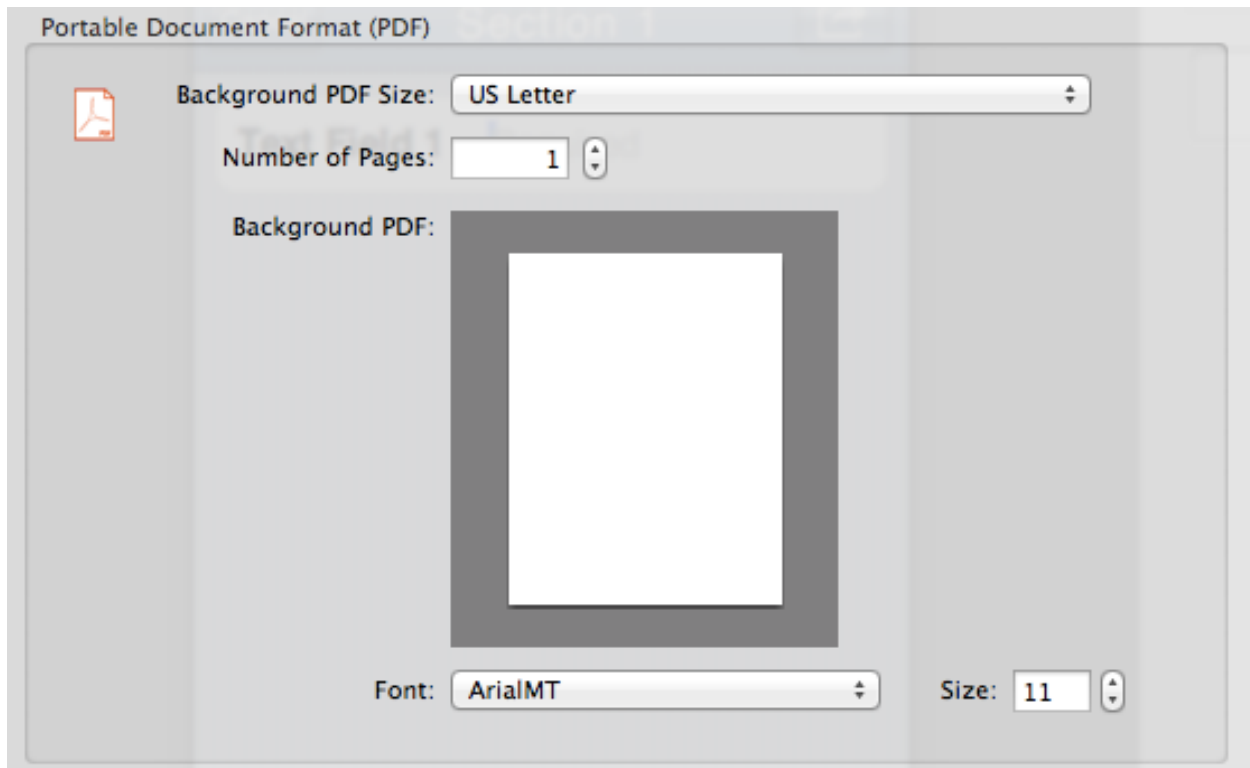
Using PDFs in your form application are a great way to deliver your information in the exact placement on a page. Whether sending a PDF attachment from email or printing to an AirPrint printer, both methods use the FormEntry's PDF layout framework. Using PDFs in FormEntry is simple, whether you want to use a PDF you've already created or have FormEntry generate a blank PDF for you.

From the Settings screen, under the DSV and PDF Settings tab, you can choose US Letter, US Letter, A4 or A5 and you can also select the number of pages you want for your PDF. There's only one PDF per form. By default, FormEntry for Mac using Arial MT at 11 points.



By changing the font and font size from the Settings screen, you will change all the fonts of your Label Names, Values and Footer on your PDF.

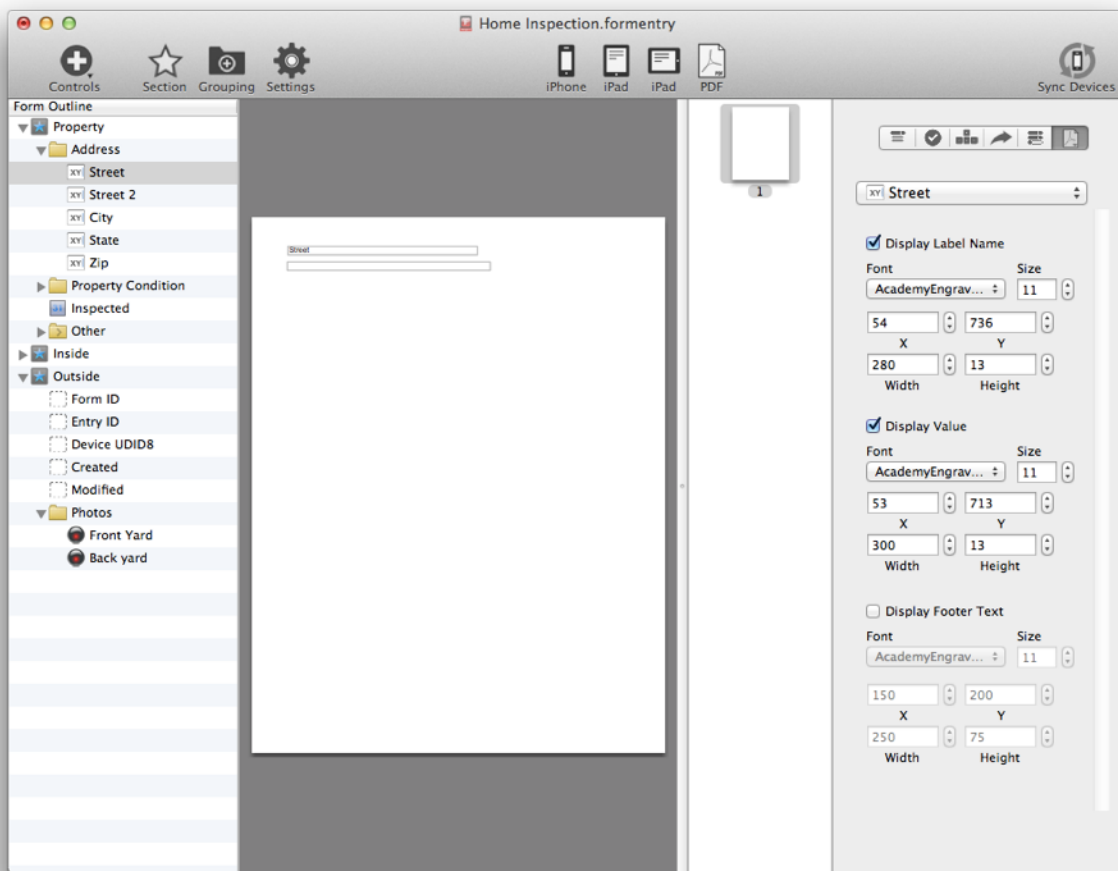
If you would like to add a custom PDF as the background PDF of your form, select the Background PDF Size from the drop down and choose Other (Add Background PDF...). A open dialog box will appear and you can then choose your PDF file from your file system. Find your PDF, select it and click Choose PDF File. You should now see your PDF file in the preview of the Background PDF.



## Laying out the PDF elements

From the PDF tab, you can review your PDF document and now have the chance to layout your visually how you want the PDF to look. From the PDF button in the Control Bar Pane, you can have a control's Label Name, Value and Footer displayed.

Select the control you want displayed on a page, then select the Display Label Name, Display Value or Display Footer Text. Once the control annotation is displayed, you can resize the annotation, move it around to your desired location, change the font and font size.

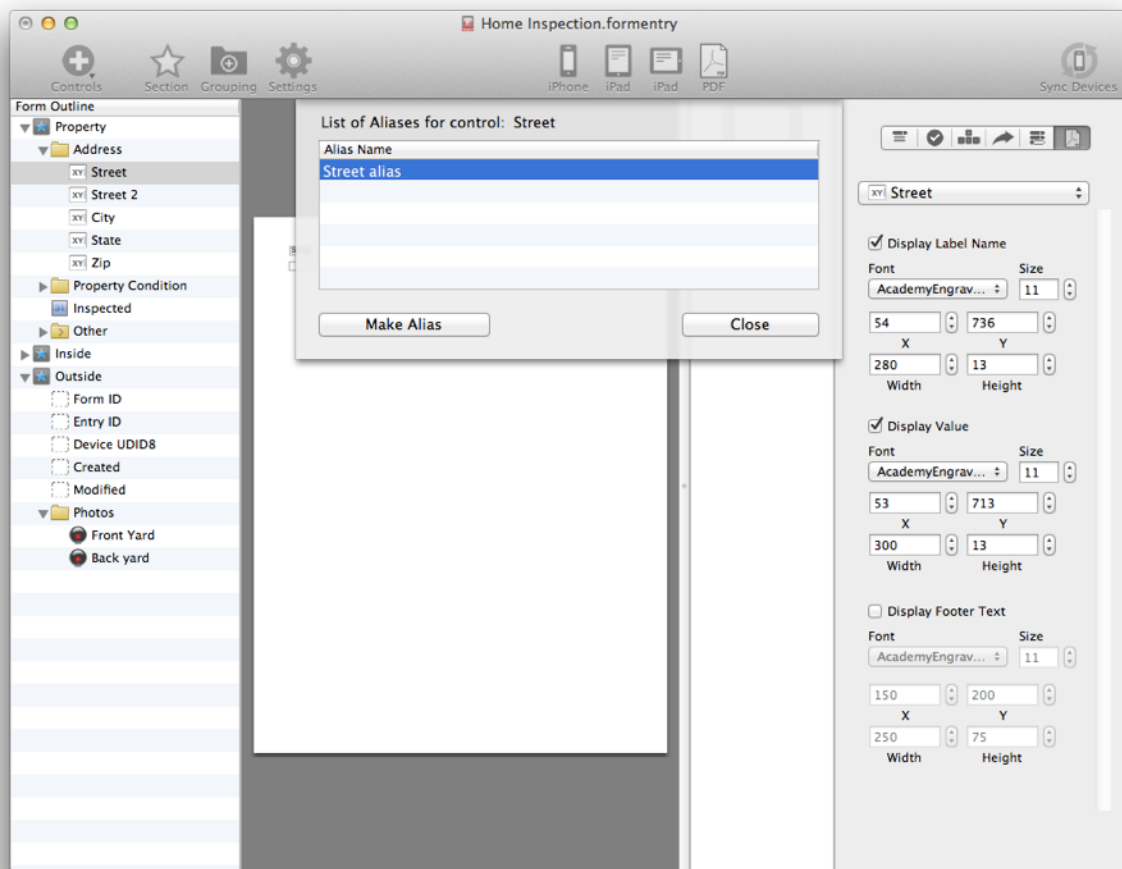


To move an control annotation from one page to another, you will have to deselect from being displayed on the current page, select the new page in the PDF thumbnails and then select the Display checkmark within the Control Pane.

The end user on the iOS device never sees the PDF you are designing. They will always interact with the native iOS controls. When they have completed your form, the PDF will be generated and used for AirPrint and by sending as a PDF attachment.

## USING PDF ALIAS

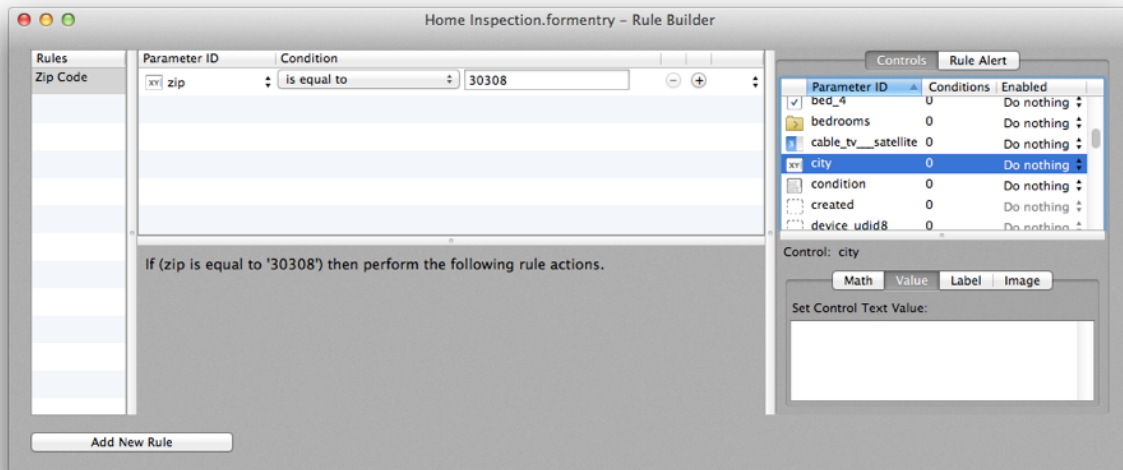
If you wanted to have the same labels, values or footer text a control on multiple PDF pages, you can create a PDF Alias by selecting the Manage Aliases... under the Control's Pop-Up menu. This is ideal for creating footers and headers through your document.



# RULE BUILDER

**This chapter explains how to create Rules and Conditional logic for your controls. Learning how to master the Rule Builder will allow you to create sophisticated form applications.**

With FormEntry for Mac, you can create as many rules as you want. With each rule, there has to be at least one condition. To add a new rule, simply select the Add New Rule from the Rule Builder window.



The new rule will automatically set the first condition. As you add controls to your form, the list of controls will appear in the drop down within the condition showing the parameter ID. Each control has a unique parameter ID that assigned to each control. You can rename these parameter ID as you see fit, however they have to be unique.

When you select the different types of parameter IDs from the drop down, you will toggle the through the options that are available for that control's condition. If you select a switch, you will see that the condition options are "is true" or "is false".

For example, if you select a location control, you will see you have five inputs for your condition, "is within" or "is not within"; the distance in meters or kilometers; selection of



meters or kilometers; the latitude and longitude. Depending on the type of control you selected, the type and range of conditions to choose will change.

You can add as many conditions as you want with the options of breaking up your conditions with inclusive ANDs or optional ORs. This is important to understand these principals for you can create complex logic within FormEntry.

As you create your conditional statements, a sentence statement about the conditions will be created in the window pane below the conditions. This will help you think through the Rule's conditions on whether it will pass or fail when performed.

If (state is equal to 'Georgia') OR (state is equal to 'California') then perform the following rule actions.

## **Rule and Condition Sequence and Rule actions**

The ordering of the rules are important to how your form application will behave to user input. Once a user completes “data” input for a control, the event bubbling order will execute.

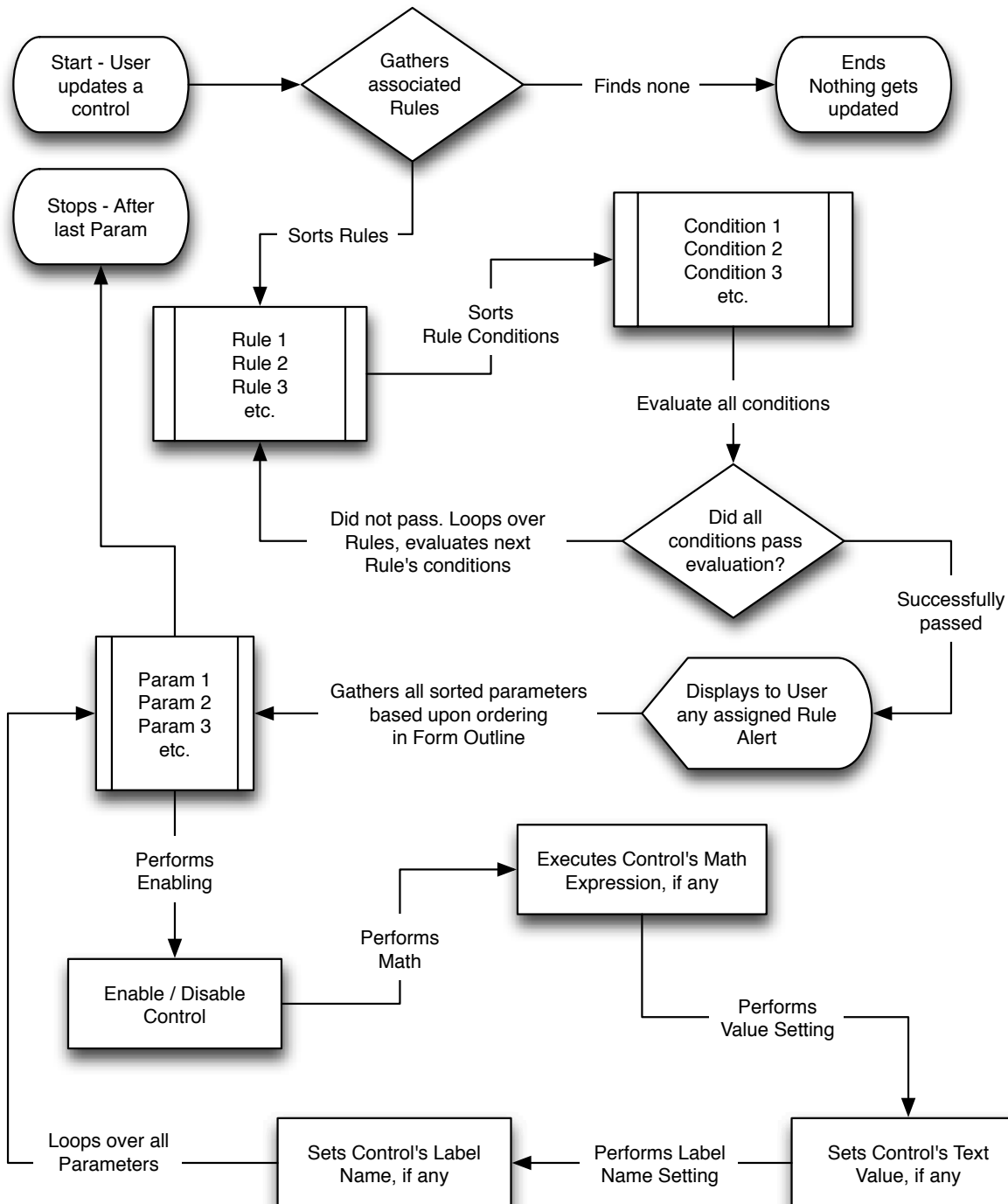
### **FORMENTRY TOUCH EVENT BUBBLING STACK**

The execution of the event bubbling is performed when there are two event criteria. First, there has to be at least one rule for the form. The second, the iOS user has to update the “data” of a control that has its parameter ID assigned to a condition. The following list illustrates the event bubbling stack within FormEntry Touch.

1. An iOS user completes an add, edit or deletion of data for a control. Simple viewing information has no effect, there must be a user driven touch upon a control.
2. FormEntry Touch will gather all the rules that have the control's parameter id assigned in a condition. The selected rules will be sorted based upon the ordering within Rule Builder rule listing. If you need to rearrange the ordering of Rules, simply select a Rule and drag it before and below another Rule.
3. FormEntry will evaluate all the conditions of the first rule. It will continue to loop through all the sorted rules and evaluate all the conditions.

4. During the evaluation of all the rule's conditions, if all the rule conditions passes, the rule's actions will be performed.
5. If a rule alert is selected, an alert message will be shown to the user, displaying the information with title, message and button label name.
6. FormEntry Touch will gather all the controls for the form and sort them based upon their ordering stack within FormEntry Touch. The ordering stack is the order of the control within FormEntry for Mac's Form Outline tree.
7. FormEntry Touch will perform the enabling or disabling of the control, the math of the control, the text value of the control and then the label update of the control, in that order:
  - First, the enabling or disabling of the controls. All the visible controls will either be enabled or disabled based upon the enabled checkbox in the Controls table. If it is a hidden control, then it will not available for enabling or disabling.
  - If there is a math expression to be performed for that control, it will be evaluated and the resulting value will be set to the performing control's value. See the Math section on how to create math expressions.
  - If there is any string value for control, it will set that value to the performing control's value. If there was any math that was previously been set, it will overwrite that value. So, it's probably best to either choose to set the control's value with a math expression or just a value.
  - Lastly, if there is any label values for the control, it will set the label value for that control.

## RULES AND CONDITIONS BUBBLING STACK FLOWCHART



## RULE ALERT

If you want to alert the user with a message, you can add a rule alert when all the conditions are met. You can display the title, message body and even the button text.

## CONTROLS TABLE

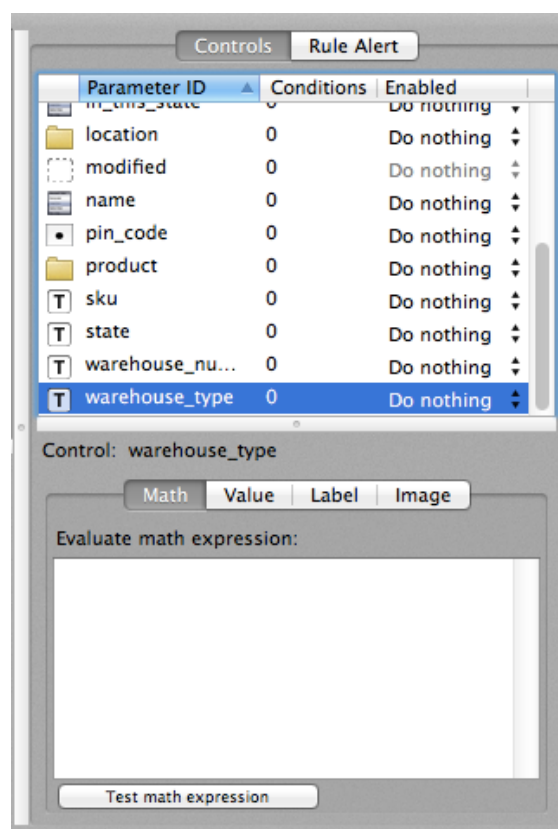
Within the Controls table lists the Parameter ID column and enabled column. When all the conditions have been met for a rule, you can enable or disable visible controls to the user. For example, if the condition for a switch is true, you can then enable textfields. If false, you can disable the textfields. Not all controls will be available for enabling or disabling, for example the Hidden and Spin Wheel Item controls.

Upon selection of the Parameter ID within the Controls table, you have the ability to execute a math expression, update the value or change the label for that selected control.

In the image to the right, notice when you select the Parameter ID of 'warehouse\_type', you see just below the Control Table, the Control label has been updated to 'warehouse\_type' and the Math, Value, Label and Image tabs are representative of the selected 'warehouse\_type'.

## EXECUTE A MATH EXPRESSION ON A CONTROL

FormEntry for Mac allows you to create a math expression that will be evaluated and set the selected control from the Controls table. You can even gather the values from other controls and use them as part of your



expression. FormEntry for Mac also allows you to quickly experiment with test expression using the “Test Math Expression” window.

## **BOOLEAN VALUES FOR CHECKMARKS AND SWITCHES**

Something to note about Checkmarks and On-Off Switches in using them to calculate in Math Expression. It is possible to take the BOOLEAN value of whether a checkmark is checked or a switch is toggled and add up those values to set on a different control.

For example, if we had a Text Field called “Totals” and three checkmarks, we can total up the checkmarks that are checked with the following Math example. So if 2 out of the 3 checkmarks were checked, the value being set for “Totals” would be 2.

```
$checkmark_1@boolValue + $checkmark_2@boolValue + $checkmark_3@boolValue
```

## **SET A STRING VALUE ON A CONTROL**

If you want to update the control’s text value, you can select the Value tab and add the value in the input field. If you had any math expressions on a control and you had the value text being set, the value text will overwrite any values being set by the math expression.

## **UPDATE THE LABEL NAME OF A CONTROL**

If you want to update the control’s label name, you can select the Label tab and add the new label name in the input field.

## **UPDATE THE IMAGE CONTROL OR ICONS OF A CONTROL**

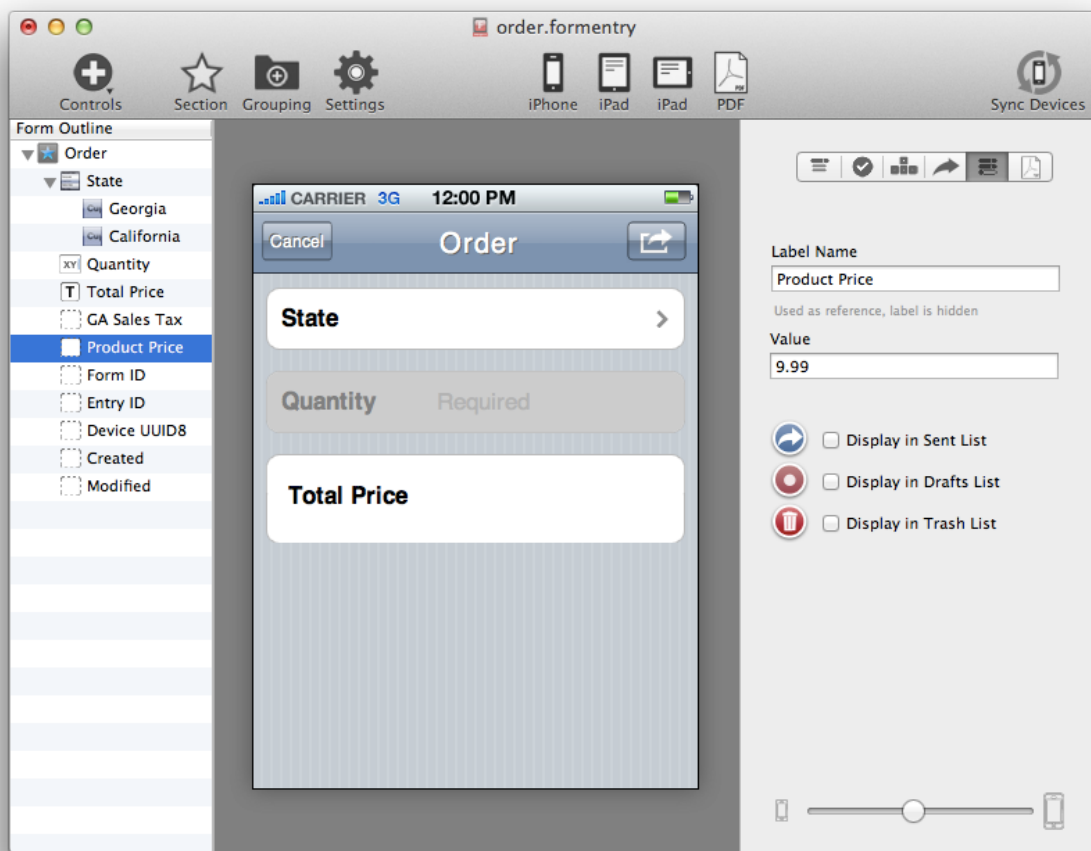
You can also update the Image Control or set the Icons of a Control either with a image or an image within the Database using the Image Key Path.

## **EXAMPLE FORMENTRY PROJECT WITH RULES**

### **CREATE A NEW PROJECT**

Let’s create an example project to illustrate how the math expression works and such. Let’s calculate if we make an order in the state of Georgia, we will add 8% sales tax. We want to multiple the quantity of products being purchased with the product price of \$9.99 and we want to show the grand total in a Total Price text control.

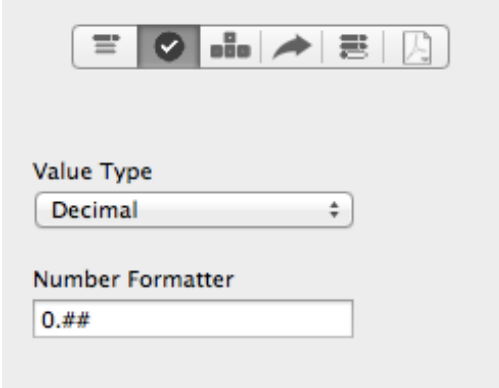
First, we will create a project called ‘Order’ and add some controls to it. The following screenshot shows our completed FormEntry project with different controls.



Controls added to our math project are:

1. Spin Wheel Picker with a label name of “State” containing two Spin Wheel Items, “Georgia” and “California”.
2. Let’s add one Text Field control and provide a label name of “Quantity”. From the Control Pane, let’s make sure the “Enabled on launch” is unchecked on the Configuration tab. Basically we want this Text Field to be disabled when the user first launches the form.

3. We will add a Text control with the label name of “Total Price” on the Configuration tab. On the Validation tab, let’s make sure we select ‘Decimal’ for the Value Type and add ‘0.##’ for the Number Formatter.
4. We will also add a Hidden control with the label name of “GA Sales Tax”. For the value, we will add “0.08”, representing the 8% sales tax in the state of Georgia. On the Validation tab, let’s make sure we select ‘Decimal’ for the Value Type and add ‘0.##’ for the Number Formatter.
5. And lastly, we added a Hidden control with the label name of “Product Price” and a value of “9.99”, representing the price of the product in US dollars. On the Validation tab, let’s make sure we select ‘Decimal’ for the Value Type and add ‘0.##’ for the Number Formatter.



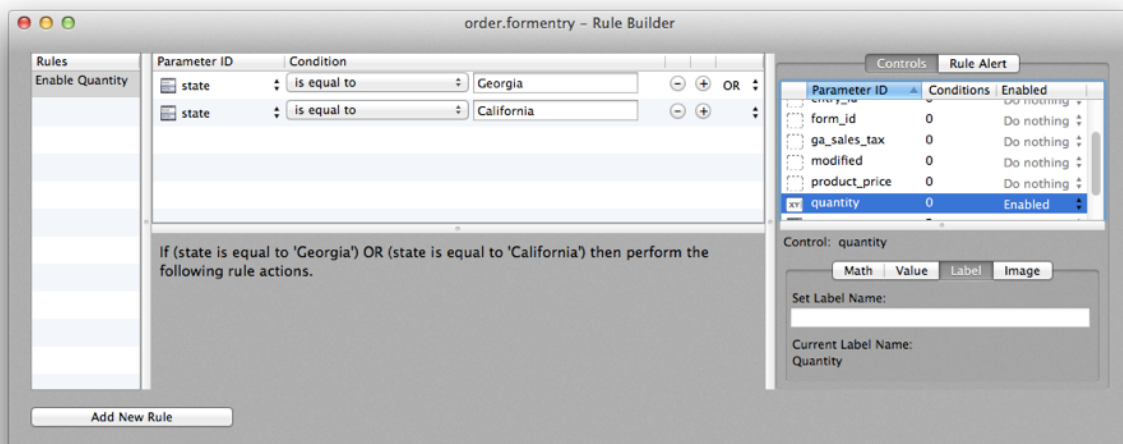
The screenshot shows a configuration window for a control's validation. At the top is a toolbar with icons for menu, checkmark, grid, arrow, list, and document. Below the toolbar, the 'Value Type' is set to 'Decimal' in a dropdown menu. The 'Number Formatter' is set to '0.##' in a text input field.

### **CREATE FIRST RULE TO ENABLE A TEXT FIELD**

Now let’s open the Rule Builder by selecting View -> Show Rule Builder from the Main Menu. Let’s add our first Rule and re-name it to “Enable Quantity”. For the first condition, change the Parameter ID to “state” and select “is equal to” on the drop down. Then place “Georgia” within the text field.

Add another condition and change the first condition’s logical operator to “OR”.

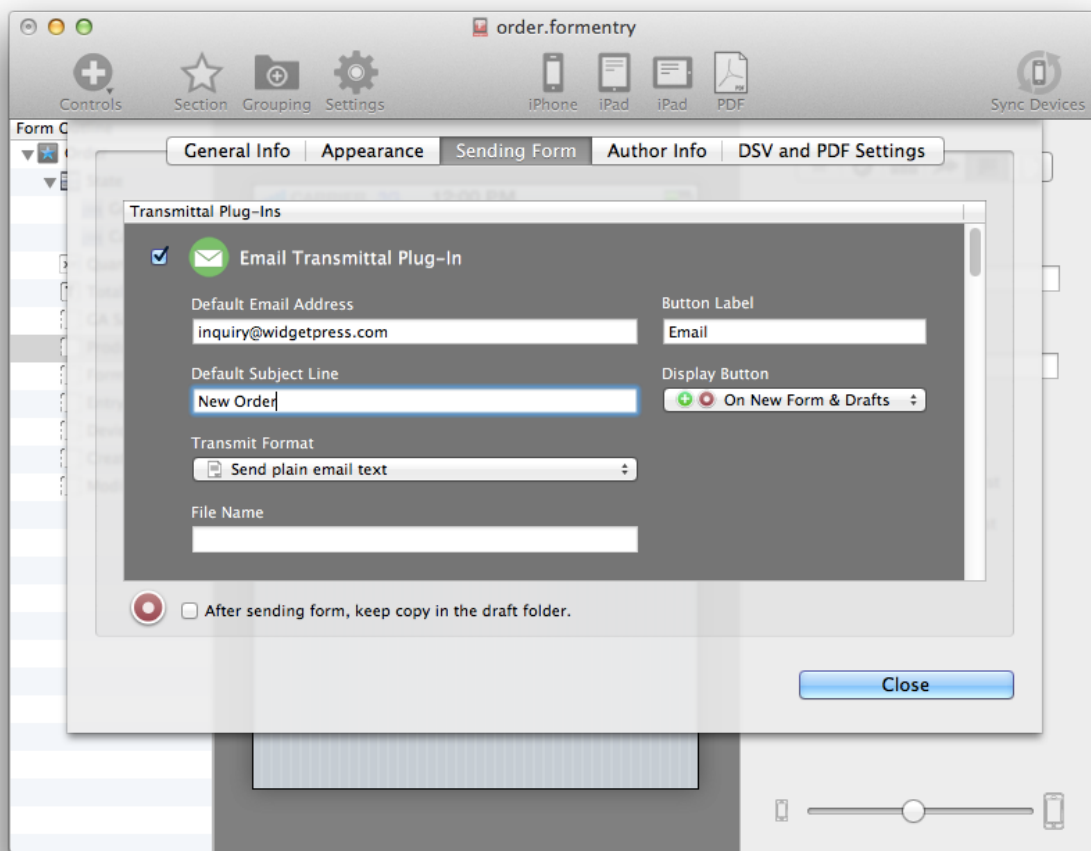
Update the second’s Parameter ID to “state” and choose “is equal to” in the drop down. Add “California” to the text field. You should have a Rule Builder screen that looks like the following screenshot:



Let's read the condition sentence statement: "If (state is equal to 'Georgia') OR (state is equal to 'California') then perform the following rule actions". At the moment, we have the "quantity" being "Enabled" from the Controls table. Remember, when the user loads the form for the first time, the "Quantity" control will be disabled and upon an update to the Spin Wheel Picker with our given states, FormEntry Touch will then enable the control of Quantity.

Let's give this a quick test on our iPhone. First, let's save our form project and give our form a 'Form Name' of "Orders" and set the "Sending Form" to email under the "Settings" tab as such:





Now make sure you are running FormEntry Touch 2.3 and now sync this form to your

iOS device by pressing the Sync Devices button. You should have the following screenshot on your iPhone (or iPad if you are using an one).

You see that the running iOS Device looks our FormEntry for Mac preview. If you select the state, you should be able to see the Quantity text field to become enabled.

## **CREATE SECOND RULE WITH A MATH EXPRESSION AND TEST MATH**

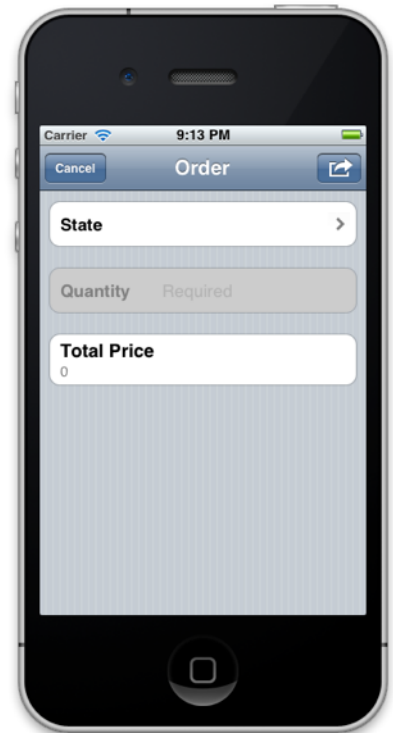
Now we want to calculate the total price with the quantity, product price and any Georgia taxes. So when the user updates the quantity, we should run another rule to perform some math on the Total Price control. Let's add another rule and name it "Updating GA Tax". Basically we want to perform some actions when the Quantity has a value of 0 or more and that the State has a value of "Georgia".

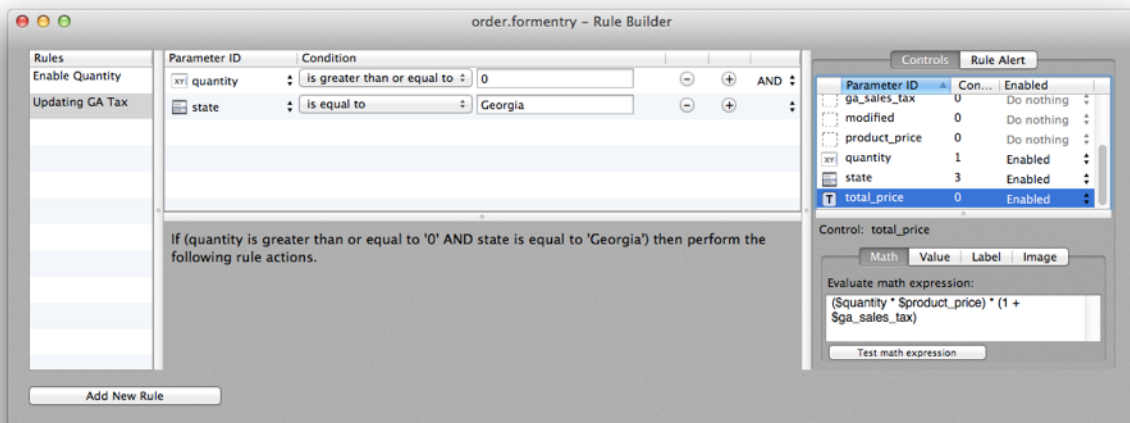
For the first condition, we select 'quantity' from the Parameter ID and choose "is greater than or equal to" in the drop down, with "0" in the text field. We add another condition with 'state' selected for the Parameter ID and change the drop down to 'is equal to' with a value of 'Georgia' in the text field.

Now if the quantity is greater than or equal to zero and the state is equal to Georgia, we should do some math on the total price. Now let's add some math. Select the "total\_price" from the Control table and select the Math tab. Enter the following math expression:

```
($quantity * $product_price) * (1 + $ga_sales_tax)
```

Your screen should look like the following screenshot:

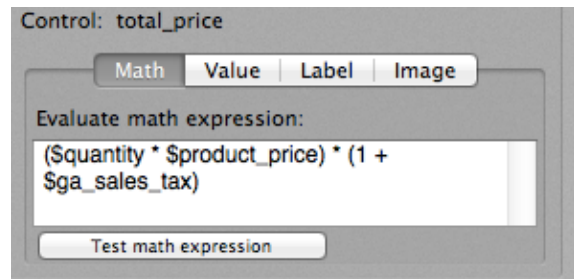




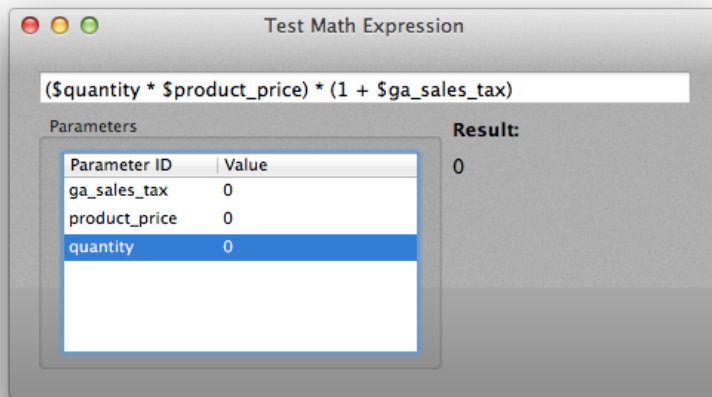
It basically says we want to multiply the price of the product with the quantity and then multiply this result with the sum of 1 plus the tax.

What we doing here is setting the value of the “total\_price” with results of this math expression. Notice how use the “\$” prefix to all parameter IDs. If you are planning using values from other controls, you must use this convention and place all parameter IDs with the prefix of “\$”.

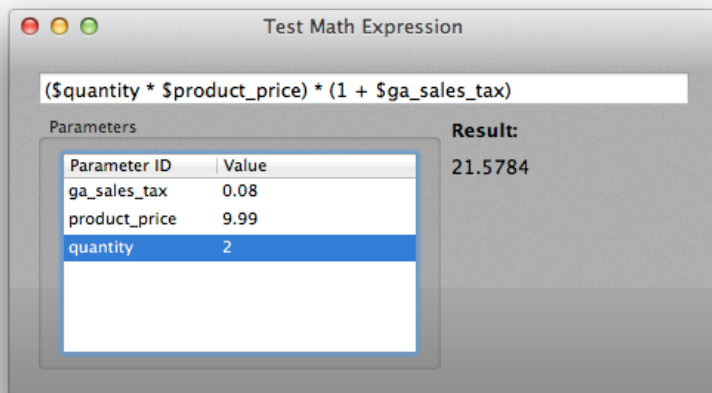
You’re math expression should look like the screenshot to the right.



Now that we have our math expression in place, we can experiment with our formula by testing with our Testing Math Expression window. Choose the “Test math expression” button and you should see the following window:

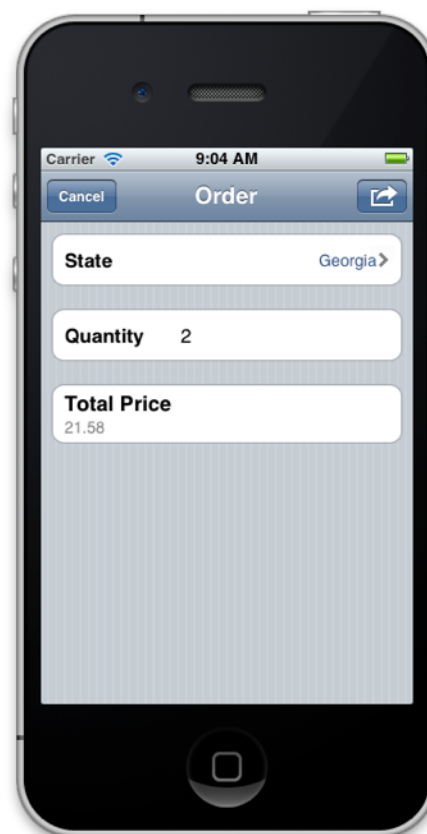


Let's go ahead and add our known values to our tax and product price. Recall that our Georgia tax was .08 and that our product price was 9.99. Add these values next to the appropriate parameter ID. Go ahead and experiment with the quantity values as if the user were updating the values on the iOS Device. Enter in 2 for the 'quantity'. You should have 21.5784 as the amount as in the screenshot below:



Now let's test this on our iPhone by syncing our form to it by pressing the Sync Devices button. Once updated our our device, test the Quantity control by adding a number of "2". You should see the 'Total Price' text control be updated to show '21.58' as in the following screenshot. You will notice that we did not receive our '21.5784', instead we are displaying '21.58'. This is the result of us adding a Number Formatter to this control.

Also, note that we are not finished yet. Try changing the state to 'California', do you notice that the total price is not being updated? Let's create a third rule to address this.



## CREATE A THIRD RULE WITH A MATH EXPRESSION

Let's create another rule for all non-Georgia states, so that we do the same math but do not add any sales tax.

Add a new Rule and name it "Updating Non-GA".

Update the first condition to 'quantity' in the Parameter ID drop down and change the condition to 'is greater than or equal to' with '0' in the next text field. Now add another condition and update the Parameter ID to 'state', change the condition drop down to 'is not equal to' with 'Georgia' in the next field. You should have something like the following:

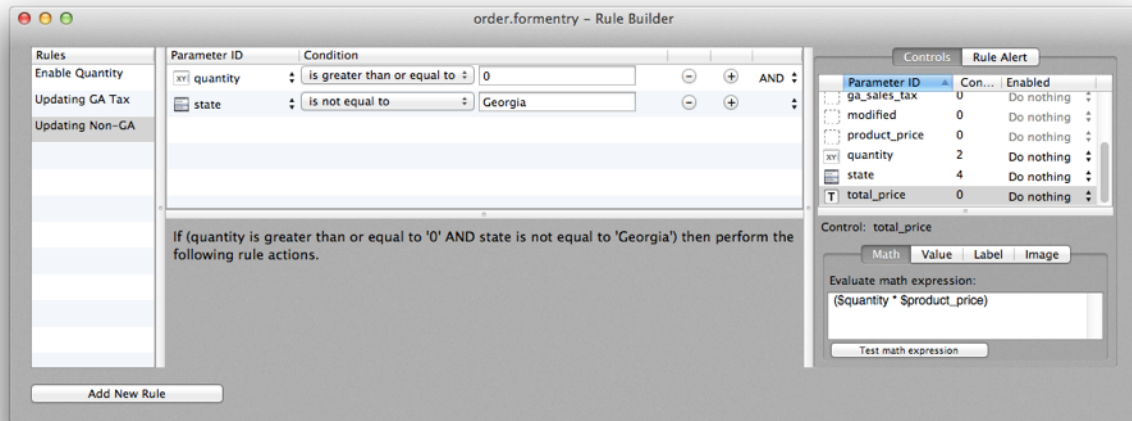
Rules	Parameter ID	Condition			
Enable Quantity	xy quantity	is greater than or equal to	0	-	+
Updating GA Tax	state	is not equal to	Georgia	-	+
Updating Non-GA					

If (quantity is greater than or equal to '0' AND state is not equal to 'Georgia') then perform the following rule actions.

So now that we have our Rule set, the only Rule Action we need is to set the total price to calculated with the quantity by the product price. Add the following math expression to the 'total\_price' Parameter ID under the Math tab:

```
($quantity * $product_price)
```

You should have the following screen shot for your Rule Builder:



Now let's test our form on an iOS device and see how it works. If you select 'California' as the state and add a '2' for the quantity, you should receive the total price of '19.98'. Your iPhone or iPad should look like the following screenshot.

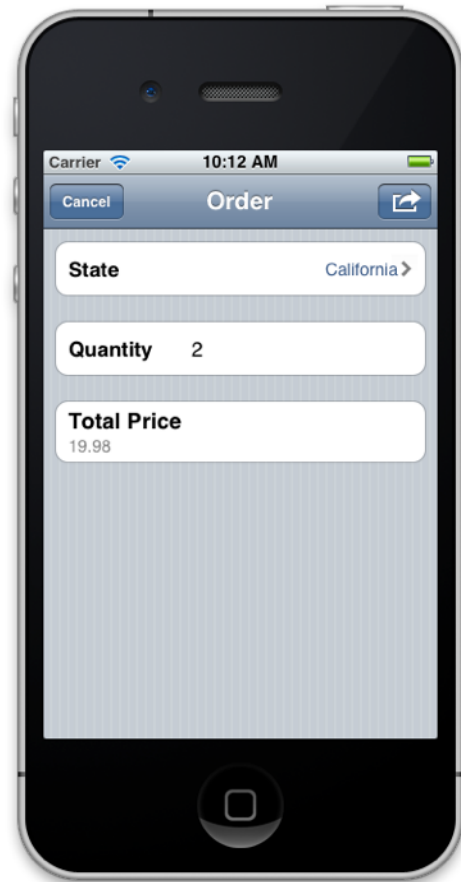
#### WHAT DID WE JUST DO?

Let's recap what we did with our project. We have three rules in order 'Enable Quantity', 'Updating GA Tax' and 'Updating Non-GA'.

We have three rules that will be performed when there is an update to 'State' Spin Wheel Picker and we will enable the quantity control.

When the user enters an amount of 0 or more within the quantity, we want to perform some math and set the results to the 'Total Price' control values.

And if the user has selected 'Georgia' we want to perform the math with the Georgia sales tax. For all others, run the math with no sales tax.



## MATH EXPRESSIONS

**Math expressions within FormEntry can be extremely powerful. The basic structure of a math expression uses operators, functions and variables.**

**FormEntry comes with built-in operators and functions that you can use in your form applications.**

## **Built-in Operators**



OPERATOR	FUNCTION	DESCRIPTION
----------	----------	-------------

#### Standard Operators

+	add	addition and unary positive
-	subtract and negate	subtraction and negation
* and ×	multiply	multiplication
/ and ÷	divide	division
%	mod	modulus
!	factorial	factorial
**	pow	exponentiation
° or °	dtor	converts the value to radians

#### Bitwise Operators

&	and	bitwise and
	or	bitwise or
^	xor	bitwise xor
~	not	bitwise not
<<	lshift	bitwise left shift
>>	rshift	bitwise right shift

#### Comparison Operators

OPERATOR	FUNCTION	DESCRIPTION
<code>==</code> or <code>=</code>	<code>l_eq</code>	equal
<code>!=</code>	<code>l_neq</code>	not equal
<code>&lt;</code>	<code>l_lt</code>	less than
<code>&gt;</code>	<code>l_gt</code>	greater than
<code>&lt;=</code> or <code>≤</code>	<code>l_ltoe</code>	less than or equal
<code>&gt;=</code> or <code>≥</code>	<code>l_gtoe</code>	greater than or equal
Logical Operators		
<code>&amp;&amp;</code> or <code>∧</code>	<code>l_and</code>	logical and
<code>  </code> or <code>∨</code>	<code>l_or</code>	logical or
<code>!</code> or <code>¬</code>	<code>l_not</code>	logical not

## CONSIDERATIONS

### THE DEGREE OPERATOR

The degree operator (`°`) is interesting. Because all of the trigonometric functions require their parameters to be passed in radians, the degree operator will convert its operand into radians. Thus, `45°` is equivalent to `dtor(45)`.

### LOGICAL VALUES

The comparison and logical operators both return a boolean value. During evaluation, that boolean value is strictly interpreted as either 0 (false) or 1 (true). For example, `41 + (1 && 1)` is literally `41 + true`, but is evaluated as `42`.

## **FACTORIAL AND LOGICAL NOT**

Differentiating between factorial (!) and a logical not (!) is difficult. A ! is interpreted as a logical not if:

- it is the first token
- it is preceded by a binary operator
- it is preceded by a right associative unary operator

Otherwise it is treated as a factorial. A  $\neg$  token is always treated as a logical not (for obvious reasons).

## **PARENTHESES AND ASSOCIATIVITY**

For simplification in dealing with implicit multiplication, an opening parenthesis is considered a right associative unary operator, and a closing parenthesis is considered a left associative unary operator.

## **OPERATOR ASSOCIATIVITY**

By default, all binary operators are left associative. That means if you give a string, such as  $1 - 2 - 3$ , it will be parsed as  $(1 - 2) - 3$ .

## **Built-in Functions**

The following functions are available:

### **FUNCTIONS THAT TAKE > 1 PARAMETER**

- `sum()` - returns a sum of the passed parameters
- `count()` - returns the number of passed parameters
- `min()` - returns the minimum of the passed parameters
- `max()` - returns the maximum of the passed parameters
- `median()` - returns the median of the passed parameters
- `stddev()` - returns the standard deviation of the passed parameters
- `average()` - returns the average of the passed parameters

- `random( )` - returns a random integer. Can take 0, 1, or 2 parameters. The first parameter (if given) is the lower bound of the random integer. The second parameter (if given) is the upper bound of the random integer.
- `nthroot( )` - returns the  $n^{\text{th}}$  root of a number. For example, `nthroot(27,3)` returns the cube root of 27, or 3.

### **FUNCTIONS THAT TAKE 1 PARAMETER:**

- `sqrt( )` - returns the square root of the passed parameter
- `log( )` - returns the base 10 log of the passed parameter
- `ln( )` - returns the base e log of the passed parameter
- `log2( )` - returns the base 2 log of the passed parameter
- `exp( )` - returns e raised to the power of the passed parameter
- `ceil( )` - returns the passed parameter rounded up
- `floor( )` - returns the passed parameter rounded down

### **• THE TRIGONOMETRIC FUNCTIONS:**

- `sin()`, `cos()`, `tan()`
- Their inverses (`asin`, `acos`, `atan`)
- Their reciprocals (`csc`, `sec`, `cotan`)
- The reciprocals of the inverses (`acsc`, `asec`, `acotan`)
- The hyperbolic variations of all the above functions (`sinh`, `cosh`, `tanh`, `asinh`, `acosh`, `atanh`, `csch`, `sech`, `cotanh`, `acsch`, `asech`, `acotanh`)
- The versine functions (`versin`, `vercosin`, `coversin`, `covercosin`, `haversin`, `havercosin`, `hacoversin`, `hacovercosin`, `exsec`, `excsc`, `crd`)
- `dtor( )` - converts the passed parameter from degrees to radians

- `rtod()` - converts the passed parameter from radians to degrees

### **FUNCTIONS THAT TAKE NO PARAMETERS ("CONSTANT FUNCTIONS"):**

- `phi()` - returns the value of  $\phi$  (the Golden Ratio). Also recognized as  $\phi()$
- `pi()` - returns the value of  $\pi$ . Also recognized as  $\pi()$
- `pi_2()` - returns the value of  $\pi/2$
- `pi_4()` - returns the value of  $\pi/4$
- `tau()` - returns the value of  $\tau$ . Also recognized as  $\tau()$
- `sqrt2()` - returns the value of the square root of 2
- `e()` - returns the value of  $e$
- `log2e()` - returns the value of the log base 2 of  $e$
- `log10e()` - returns the value of the log base 10 of  $e$
- `ln2()` - returns the value of the log base  $e$  of 2
- `ln10()` - returns the value of the log base  $e$  of 10

## Implicit Multiplication

The parser recognizes implicit multiplication. For example, we can write  $3(4)$  and understand that the answer should be 12. Implicit multiplication is applied when a number, variable, or closing parenthesis are followed by either a number, variable, function, or opening parenthesis.

Implicit multiplication is quite simple to recognize. We only need 2 tokens in order to recognize it:

		Current Token			
		Number	Operator	Variable	Function
Previous Token	Number	Yes	Maybe	Yes	Yes
	Operator	Maybe	Maybe	Maybe	Maybe
	Variable	Yes	Maybe	Yes	Yes
	Function	No	Maybe	No	No

## HANDLING OPERATORS

Handling implicit multiplication when operator tokens are involved is tricky. Under normal circumstances, an operator followed by an operator should not result in implicit multiplication. For example, if you have  $1 + + 3$ , this should be  $1 + 3$  (the second  $+$  being the unary positive, and is thus discarded). It would be incorrect to insert a multiplier between the two  $+$  signs.

However, under other circumstances, you *do* want to insert a multiplier. The common case of  $(2)(3)$  should result in a multiplier being inserted in between to the inner two parentheses (which are both considered operators by the tokenizer).

Thus, a multiplier is inserted in conjunction with operators if:

the previous token is a number, a variable, or a left associative unary operator (factorial, closing parenthesis, etc) and

the current token is a number, a variable, or a right associative unary operator (logical not, opening parenthesis, etc)

## **FUNCTIONS AND IMPLICIT MULTIPLICATION**

A multiplier token is never inserted after a function token, because the rules of argument-less function make that it impossible for a function token to be followed by anything other than an opening parenthesis (an operator).

## **ARGUMENT-LESS FUNCTIONS**

Normally, a function is entered in the form `function(parameter, parameter)`. However, the tokenizer can also recognize functions as simply `function`. In this case, the opening and closing parentheses are injected into the token stream. This is only useful when entering constants, since these are the only functions which do not accept parameters. The upshot of this is that you can pass the string " $\pi + e$ " and will be correctly parsed as if you had passed " $\pi ( ) + e ( )$ ".

If you attempt to use this with other functions, an error will be generated and evaluation will fail. This makes sense since (for example) `sin ( )` cannot be evaluated. The `sin` function requires a parameter.

## **Tokenizing**

Tokenization is the first phase of parsing. It's the point where characters are extracted from the source string and grouped. There are four kinds of tokens:

1. numbers
2. functions
3. variables
4. operators

## **NUMBERS**

All numbers extracted from the source string are positive, and are defined as anything that matches the following regular expression:

```
\d*\.\d*([eE][+-]?\d+)?
```

It should be noted here that "." is a valid number and is interpreted as 0.

No effort is made to allow for locale-sensitive numbers. Allowing things like thousands groupings or the comma as the decimal separator can introduce ambiguity in the parser. For example, if "," were recognized as the decimal separator, then this is ambiguous:

```
max(1,2,3)
```

Should that be parsed as the maximum of three numbers (1, 2, and 3), or the maximum of two (1.2 and 3, or 1 and 2.3)? Similar problems arise when dealing with thousands groupings. As such, numbers are not locale-sensitive.

## FUNCTIONS

Function tokens are strictly the *name* of a function. For example, given the string "sin(0)", the extracted function token is "sin".

Functions can contain letters (upper and lowercase), decimal digits, and underscores.

The exception to this are three special functions that are special cased in their recognition: " $\pi$ ", " $\Phi$ ", and " $\tau$ ". These correspond to the mathematical constants of pi, phi, and tau, respectively.

## VARIABLES

Variables follow the same rules as functions, except that they must be prefixed with a "\$" character. Thus, the following are all legal variable names:

- \$a
- \$\_
- \$0xdeadbeef

Variables are denoted in the source string as beginning with \$ and can contain numbers or letters. They are case **sensitive**. (\$a is not the same as \$A)

## OPERATORS

Operators are all other characters in the string. With three exceptions, they are all single characters. The three exceptions are "\*\*\*", "<<<", and ">>>".



Parentheses are parsed as operator tokens, even though they are not listed as part of the built-in operators. Parentheses used to denote order of operations and functions arguments are eliminated during term grouping.

## REGARDING WHITESPACE

Whitespace is seen as a *logical break* in the token stream. Now, "3 4" will be parsed as the 3 token followed by the 4 token. And because of the logic in recognizing implicit multiplication, a multiplication operator will be injected into the stream. Now, "3 4" is recognized as "3\*4", or 12.

## Term Grouping

Grouping happens following tokenization and is when tokens are organized hierarchically by parenthetical level.

This is accomplished by a simple LL(1) recursive descent parser. Tokens are organized into *terms*, of which there are several kinds.

## GROUP TERMS

Group terms are the representation of a parenthetical group and all of the terms inside of it. For example, given the string "1 + ( 2 - ( 3 / 4 ) )", there are three group terms:

The term containing the 3 term, the / term, and the 4 term

The term containing the 2 term, the - term, and group term #1

The overall "root" group term containing the 1 term, the + term, and group term #2.

## FUNCTION TERM

A function term is a specialized group term. Like the group term, it can have "subterms", but it also has a property denoting the name of the function. For example, if we have the string "sin(\$x) + \$x", the function term will be the term containing the first \$x variable, plus the name of the function ("sin").

## NUMBER TERM

A number term is a term representing a numeric literal.

## **VARIABLE TERM**

A variable term is a term representing a variable.

## **OPERATOR TERM**

An operator term is a term representing one of the built-in operators.

During tokenization, we created operator tokens for opening and closing parentheses. An operator term does *not* represent those particular tokens, since parentheses are represented with group terms and function terms.

# **PUBLISHING TO FORMENTRY SERVER**

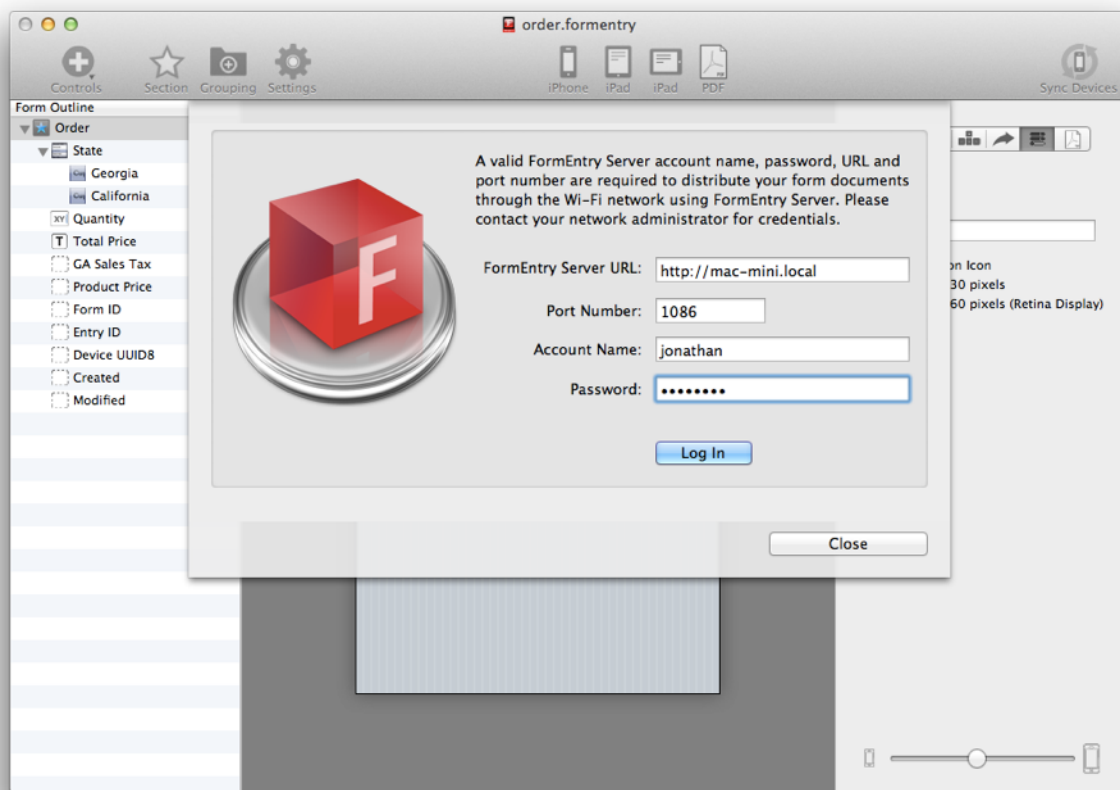
**This chapter explains how to publish and manage the forms on FormEntry Server.**

FormEntry Server enables the sending of forms over the Internet to FormEntry Touch users. Users can also receive forms within the local Wi-Fi hotspot of FormEntry Server if Wi-Fi is set on the server. Users of FormEntry Touch can also “retrieve” forms from FormEntry Server over the Internet. In this case, the URL and Port number of the assigned FormEntry Server is required.

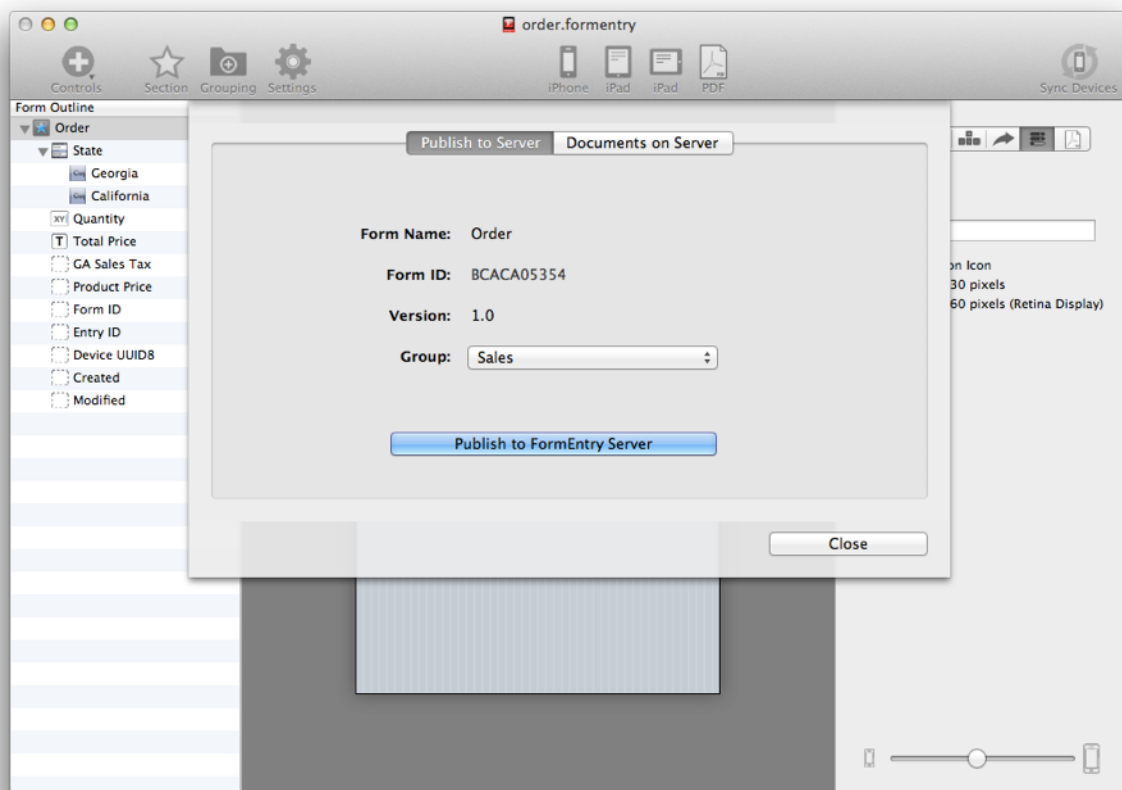
If you would like to store the FormEntry Server URL and Port number, you can save this information within the Preferences under the FormEntry Server tab.

After you have completed your form and would like to publish it to FormEntry Server, you will need to save your project, then select File -> Publish to FormEntry Server from the main menu.

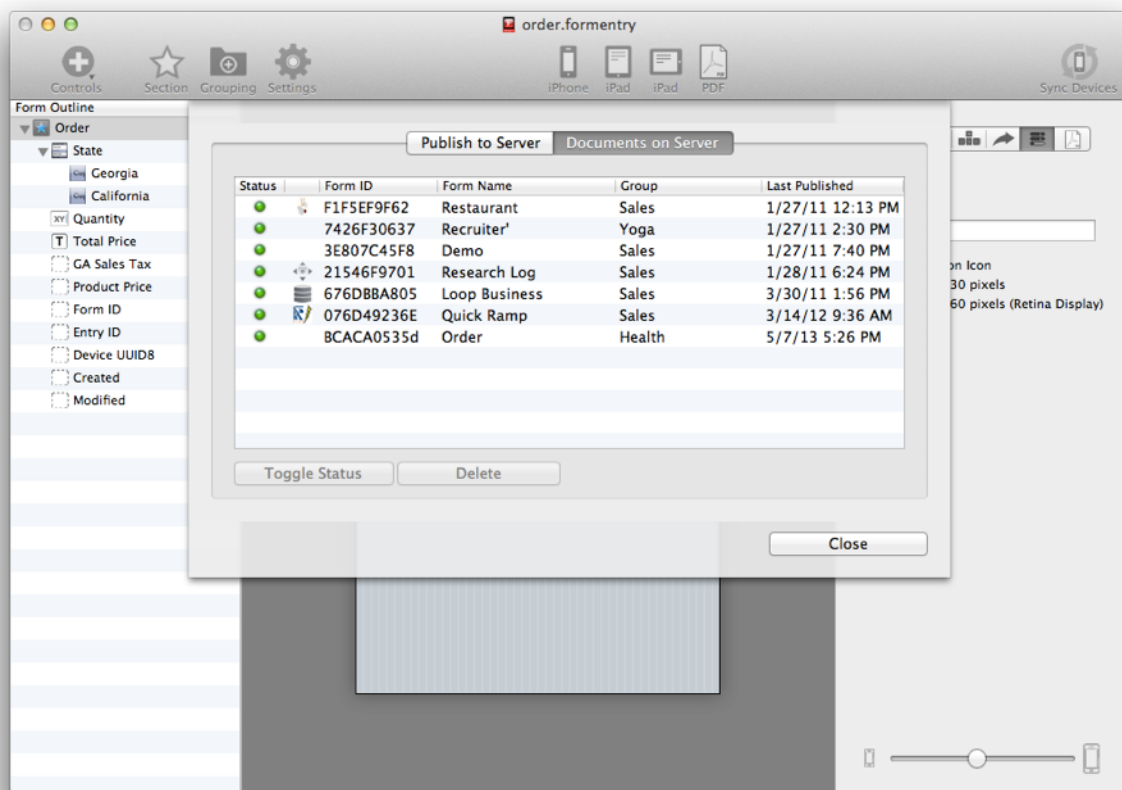
Within the window sheet, you will need to enter the FormEntry Server URL, Port Number, Account Name and Password. Select the Log In button and you will be authenticated against FormEntry Server.



After you have logged in, you can publish your form to any of the groups that have been assigned to you. Select the drop down Group and press Publish to FormEntry Server button to send your form document to the server.



From the Documents on Server tab, you can manage your documents if you are granted permissions to remotely administer your documents. You can have the ability to toggle the availability of your form to your users.



# RESOURCES FOR MORE LEARNING

Please consult the following resources to get the most out of FormEntry:

## **ONSCREEN TOOLTIP HELP**

To see the help, open FormEntry and create a new project. Hover over sections and labels to receive information in a tooltip format.

## **WEB RESOURCES**

Go to [www.widgetpress.com/support](http://www.widgetpress.com/support) to get the latest information about updates, articles, forums and how to guides. A FormEntry license key can also be purchased on this website.