



Alpha TransForm User's Guide

Chapters 1 - 7

Examples used in this book are fictitious. Names, places, and incidents either are the products of the author's imagination or are used fictitiously. Any resemblance to actual persons, living or dead, events, or locales is entirely coincidental.

Copyright © 2019 by David McCormick and Alpha Software Corporation.

All rights reserved.

Second Electronic Edition June 11, 2019

Last Revision: June 24, 2021

Published by Alpha Software Corporation

www.alphasoftware.com

Table of Contents

Table of Contents	1
Chapter 1: Introduction	5
Alpha TransForm - What is it?	5
Who is it Designed for?	5
Example Application	6
Chapter 2: TransForm Accounts and Licenses	9
Creating an Account	9
Logging In and Out and Confirming Login	10
Resetting Your Password.....	11
Managing Account Users	12
Managing Licenses	14
Switching Between Accounts	16
Naming Your Account.....	17
Assigning Usernames, Passwords, and Roles all at Once	18
Chapter 3: Designing Forms Part 1 - The Basics	19
Forms vs Form Types	19
The Designer Tab	19
Creating a Form Type Using a Template.....	20
Creating a Form Type From Scratch	22
Quickstart Text	24
Getting Quick at Quickstart.....	27
Quickstart Notation	28
Quickstart Text is a One Way Trip.....	29
Form Commands View.....	29
Headings and Pages	32
Data Groups.....	34
Saving Your Design	36
Editing an Existing Form Type.....	37
Deleting a Form Type.....	37
Chapter 4: Using the TransForm App	39
Installing TransForm.....	39
Logging in and Confirming Login.....	40
Resetting a Password	41
Working With Multiple Accounts	41
New Forms	43
Existing Forms	44
Sorting and Grouping Existing Forms.....	46
Filling in Forms.....	47
Navigating Large Forms	50

Finding and Fixing Data Entry Errors	53
Submitting and Uploading Form Data	56
Marking a Form as Submitted, Open, or Another Status.....	56
Uploading Form Data.....	59
Deleting a Form	63
Resetting a Form	66
Getting Help	67
Chapter 5: Designing Forms Part 2 - Form Commands	69
If and Else - Conditional Blocks	69
Syntax for the IF Command.....	73
ELSE IF and ELSE	74
END IF	75
Error Indications	75
Error Checking Example.....	76
Checking for Blank Required Fields	77
Go To	77
Submit Buttons (Change Status Command)	81
Changing the Text on a Submit Button.....	82
Automatically Removing the Form After Upload.....	82
Sharing (Importing/Exporting) a Form Type	83
Exporting a Form Type	83
Importing a Form Type	84
Importing and Exporting Only the Commands.....	86
Chapter 6: Managing Data	87
Management Console	87
Editing Data in a Form	90
Deleting Forms	91
Creating Forms	92
Creating One Form.....	92
Creating Multiple Forms at Once	94
Exporting Data	97
Customizing the Forms List	98
Customizing the Details Field	101
Sorting and Finding Forms	104
Assigning Forms to Users in the Field	104
Adding Comments to a Form	105
Viewing and Editing Form Data in JSON Format (Advanced)	108
Viewing and Editing the JSON Data for a Form	108
Creating a New Form Using JSON Data	109
Chapter 7: TransForm Programming Language	111
TPL Expressions in IF Commands, HTML Blocks, Headers, and Templates	111
The TPL Code Editor	113
Events	115

Event Behavior You Might Not Expect.....	116
Example: Calculation	116
Example: Populating a List Field	119
Displaying One Value and Saving Another.....	121
Example: Populating a List Field From an API.....	122
Example: Populating a List Field from an Onboard Database.....	125
Example: Looking Up Data from a Barcode Scan or List Choice.....	126
Launching Videos, PDFs, Spreadsheets and Other Documents	127
Getting Driving Directions to an Address.....	129
Creating Functions.....	131
Creating a New Function	132
Debugging Your Applications.....	133
Using the Tester: The Basics.....	134
Using the Tester: An Example	137
Using the Debugger.....	139
How to Load, Refresh and Delete On-Device Assets.....	147
Using the Alpha Anywhere On-Device Data Builder.....	148
Setting up an S3 Storage Bucket.....	153
Setting the Download Policy for Assets	158
Whitelisting Files and URLs	159
Whitelisting Image URLs and Folders	160
Whitelisting URLs Used in AJAX (API) Calls	160
Creating a SQLite Database	161
Building a Database Connection String.....	162
To Create a New SQLite Database in Alpha Anywhere	166
Where to Get More Information.....	170

Chapter 1: Introduction

Welcome to the Alpha TransForm User's Guide. This book explains and demonstrates the features and capabilities of Alpha TransForm starting with, "What is it?"

Chapters two and three cover the basics of setting up an account and building simple applications. Chapter four covers how to use TransForm apps (called "forms types") on a mobile phone or tablet. Chapter five shows how to build more sophisticated apps, and chapter six shows you how to manage the data you collect, and more.

Collectively, these six chapters provide a full overview of most of the out-of-the-box features that are included in TransForm.

Alpha TransForm - What is it?

Alpha TransForm consists of two parts: a website called TransForm Central and a mobile application called Alpha TransForm. The website is used to create and deploy applications, as well as to manage the data collected by those applications. The mobile application runs the applications created by the website.

Who is it Designed for?

Alpha TransForm is designed to be used by everyone, while at the same time allowing "power users" (who are more technically inclined) and developers to add more power to their applications with a blend of advanced features.

Anyone can use the applications created in TransForm on their mobile phone or tablet with little or no training.

- ✓ Anyone can create applications, deploy them, use them to collect data and then review that data (and more) with less than 45 minutes of training.
- ✓ Anyone can use the built in Zapier integration to add features like SMS messaging, data integration, workflows, and more.

- ✓ People who are slightly more advanced (“power users”) can use the scripting language (TPL) to further automate their applications.
- ✓ People with coding experience can use the TransForm API to integrate TransForm with virtually any data source or system of record (like SAP, SQL Server, and many more).
- ✓ Alpha Anywhere Developers can use the TransForm functionality in their own Alpha Anywhere applications through a collection of built-in integrations.

Example Application

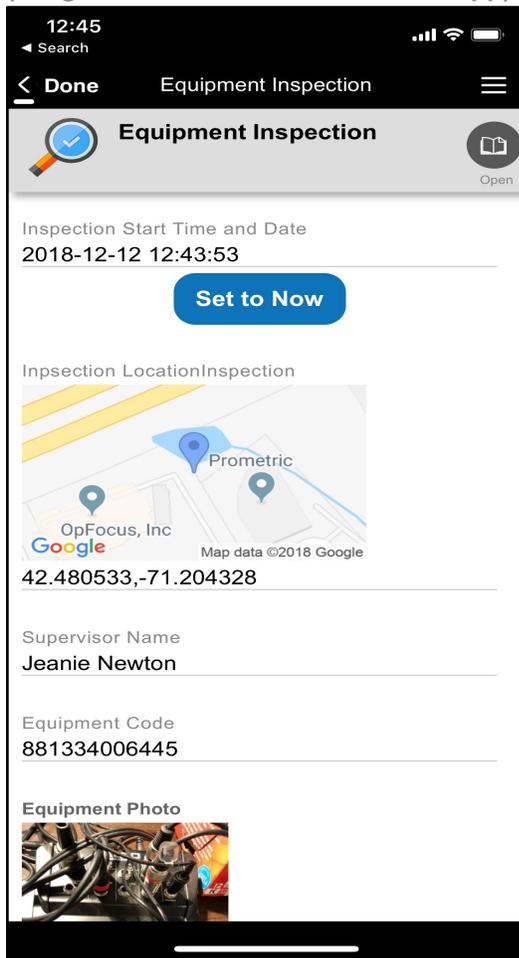
A common use of TransForm is for field service applications. In this scenario, a field worker downloads a series of work tickets called “forms” to their mobile device at the beginning of the day. Then during the day, they perform work, add information to the forms, and finally submit them to their company’s system of record where they can be reviewed and processed.

In this example, workers inspect equipment and report their findings. Because the form is on a mobile device instead of on paper, the worker can take advantage of the features built into the device.

For example, the date and time of the inspection is automatically entered, the location is captured using the device’s built in GPS, and the device’s camera is used to take pictures of the equipment.

You can also use TransForm for barcode and QR code scanning, speech to text (dictation), audio recordings, and more.

(Image 1.1 - The TransForm Mobile App)



Another benefit of TransForm is that it is fully functional offline. Once a form has been downloaded to the device, you no longer need an internet connection to work with the app. You can also create new forms on the device, again without relying on an internet connection. The data you collect is saved on the device until it is later synchronized to the TransForm cloud when an internet connection is available.

Once synchronized, the data becomes available in the TransForm Central website, where it can be viewed, edited, and exported.

(Image 1.2 - TransForm Central - transform.alphasoftware.com)

The screenshot displays the Alpha TransForm Management Console interface. The top navigation bar includes the Alpha TransForm logo, Home, Designer, Permissions, and Management Console tabs. The user is logged in as Dave McCormick, with a login expiration of 2 days (14-Dec-18). The main content area is divided into four panels: Choose Type, Equipment Inspection, Select Fields, and Form Contents. The Equipment Inspection panel shows a table of inspection records.

Status	Created	Changed	User Name	Details
<input type="checkbox"/>	Submitted	Wednesday 10:28am	Dave McCormick	
<input type="checkbox"/>	Open	Wednesday	Dave McCormick	ACME Industries: 10:00 am 18 Industrial Way, Boston
<input type="checkbox"/>	Open	Wednesday	Dave McCormick	Badger Industries: 12:30 PM 12 Forest Ave, Newton

The Form Contents panel shows the details for the selected inspection, including fields for Building Name, Street Address, City, ZIP, Scheduled Time, Inspection Location (with a map showing the location at 42.480533, -71.204328), Supervisor Name (Jeanie Newton), and Equipment Code (881334006445). The Actions panel includes options to prevent form filler from downloading, change status (Currently: Submitted), and revert to Open.

Chapter 2: TransForm Accounts and Licenses

In order to use TransForm, you need an account. You can either sign up for an account yourself or someone can create an account for you (by sending you an email invitation with a signup confirmation link).

Once you have a username and password for your account, you can use them for both the TransForm Central website and the TransForm app.

Creating an Account

To create an account, visit transform.alphasoftware.com and click the “Don’t Have a Username / Password ?” link to get to the registration form.

(Image 2.1 - Registration Page)

alpha TransForm Home Not Logged In Help

Welcome to Alpha TransForm

See how TransForm can benefit your organization by signing up for the 30-day trial account below. Trial account features:

- Short tutorial shows you how to build and deploy an app in minutes
- Add up to 4 other users to your account
- Upgrade to a full account without losing anything
- All features included in the trial
- No credit card required

Register for a Trial Account

First Name (Required)

Last Name (Required)

Email Address (Required for your username and to reset your password)

Password (Required)

Please retype your password (Required)

Company (Optional: If you are willing)

Phone Number (Optional: But, helpful to recover lost account)

It's OK to send periodic emails about new features, tech info, and offers.

I have read and understand the [Privacy Policy](#)

I agree to Alpha TransForm [Terms or Service](#)

REGISTER

Logging In and Out and Confirming Login

Both the TransForm app and the TransForm Central Website allow you to log in in for up to two days at a time. You can log out before that time by clicking Logout in the app and/or TransForm Central.

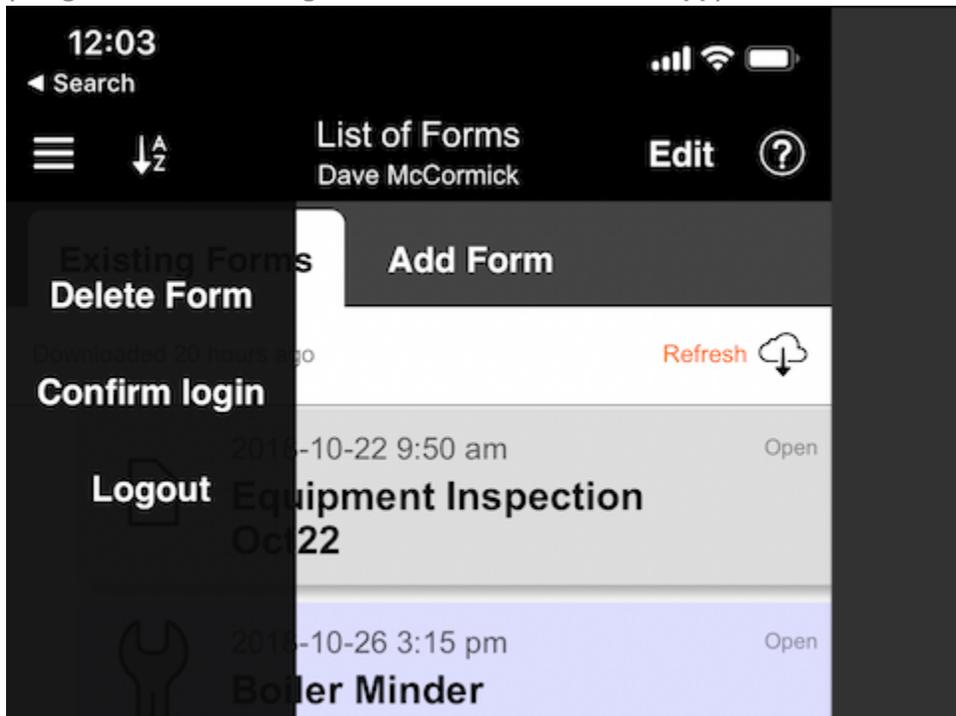
Once two days have passed, TransForm Central will require you to confirm your login by asking you to enter your password again. You can see how much time is remaining by looking at the top right corner of the screen. If you want to confirm ahead of time, you can click the blue text to confirm your login and reset the clock for two days.

Image 2.2 - Confirm Login in the TransForm Central Website



As with TransForm Central, your login for the TransForm app is also valid for two days before you need to confirm login. The app will continue to work after your login has expired, but you will not be able to synchronize data with the TransForm cloud until you confirm your login. To confirm login on the mobile app, tap the hamburger button  at the top left and choose Confirm login.

(Image 2.3 - Confirm Login in the TransForm Mobile App)



Resetting Your Password

If you have forgotten your password or simply want to change it, you can reset your password in TransForm Central. If you have forgotten your password, click the [Forgot Your Password](#) link when you Sign In.

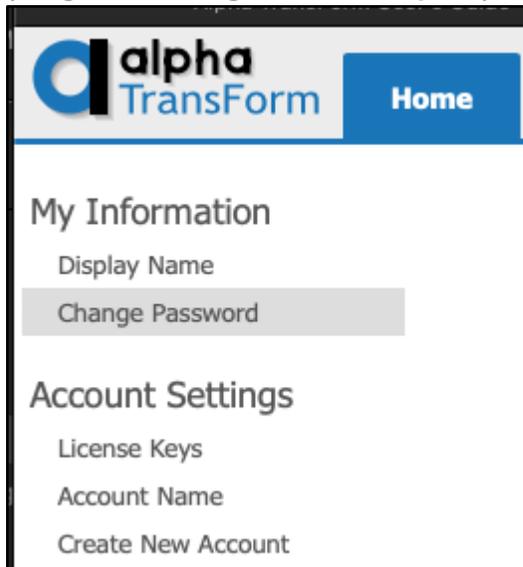
(Image 2.4 - Sign In Box With Forgot Your Password Link)

A screenshot of the Sign In box. The title "Sign in" is in large blue font. Below it are two input fields: "E-mail" and "Password". A blue link "Forgot your password?" is positioned below the password field. A blue "Sign In" button is centered below the input fields. At the bottom, there is a line of text: "By clicking Sign In, you agree to the [Terms of Service](#) and [Privacy Policy](#)." and another blue link "Don't have a username/password?".

TransForm will ask you for your email address, and then email you a password reset link.

If you already know your password, and want to reset it, you can do that on the Home Tab of TransForm Central. Click the Change Password link in the menu on the left.

(Image 2.5 - Change Password Option)



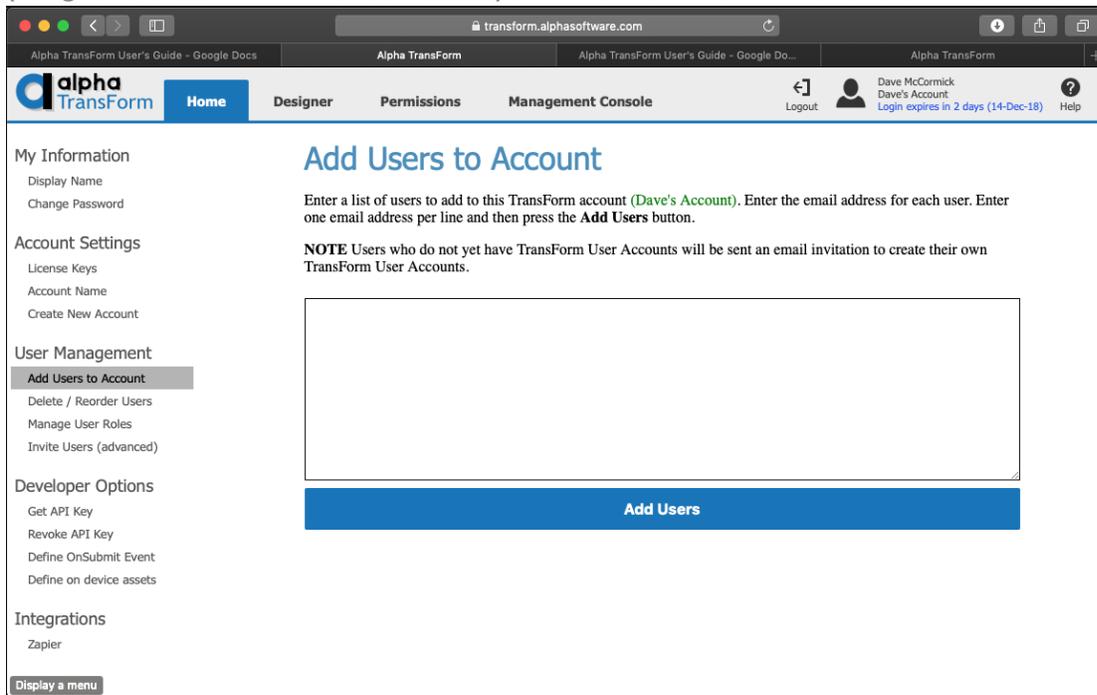
Managing Account Users

Once you've created an account, you can invite other users to your account and assign them roles. The most basic role is simply called a "User." This is a person who is allowed to fill out forms in the TransForm app, but otherwise has no access to the account.

You can also assign one or more users to be Form Designers, which are individuals who are allowed to log in to your TransForm Central account to create new form designs. There are other roles as well, and the system allows you to create new ones (for more information, see Chapter 9).

Inviting users and adding them to your account is done on the Home tab of TransForm Central. To add a user, click the Add Users to Account link in the left menu.

(Image 2.6 - Add Users to Account Box)

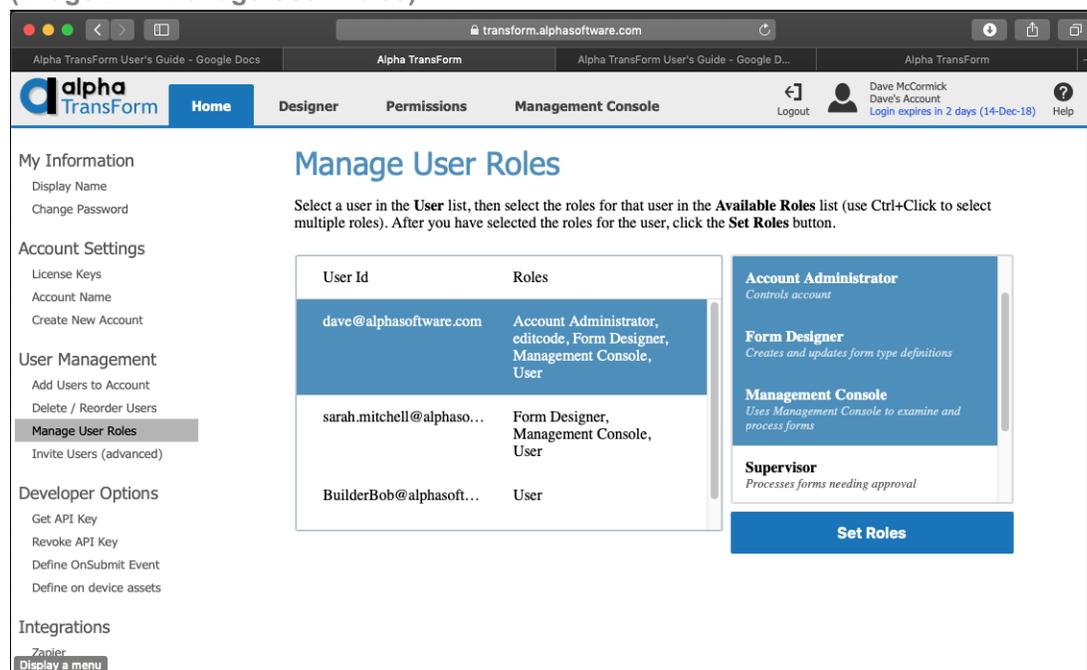


In the box, enter the email address(es), one per line for each person you want to invite.

They will receive an email inviting them to your account, and if they do not already have an account themselves, they will be prompted to create one.

Once user(s) have been added to your account, you can assign them roles by clicking the Manage User Roles link in the left menu.

(Image 2.7 - Manage User Roles)



To assign roles, click on the person to whom you want to assign the role(s), then hold down the Control key on your keyboard and click each of the roles you would like to assign. When you are done, click Set Roles to save your choices. Repeat this for each user.

Managing Licenses

When you first sign up for a TransForm account, you are automatically assigned a 30-day 5-user license trial license. This means that you can invite up to four other people to your account to act as users, form designers, account administrators or any other roles of your choosing.

After 30-days, your trial license expires, and you must enter a license key to keep using your TransForm account. You can either buy a key, or you can request a Sandbox license. A Sandbox license allows 3-users (including you) to use your account for a year.

Sandbox licenses are free and are meant strictly for evaluation and prototyping.

To purchase a license, please contact sales@alphasoftware.com or visit <https://www.alphasoftware.com/alpha-transform-pricing>. For a Sandbox license, you need to contact sales@alphasoftware.com.

Licenses can be added together. For example, you can have a 3-user license and two 10-user licenses for a total of 23 users. To add a license to your account, go to the Home tab of TransForm Central and click the License Keys link in the left menu.

Paste your license key(s) into the box and click Save to save them to your account. To see the user counts and expiration dates of your licenses, click the Show License Information button.

(Image 2.8 - License Keys)

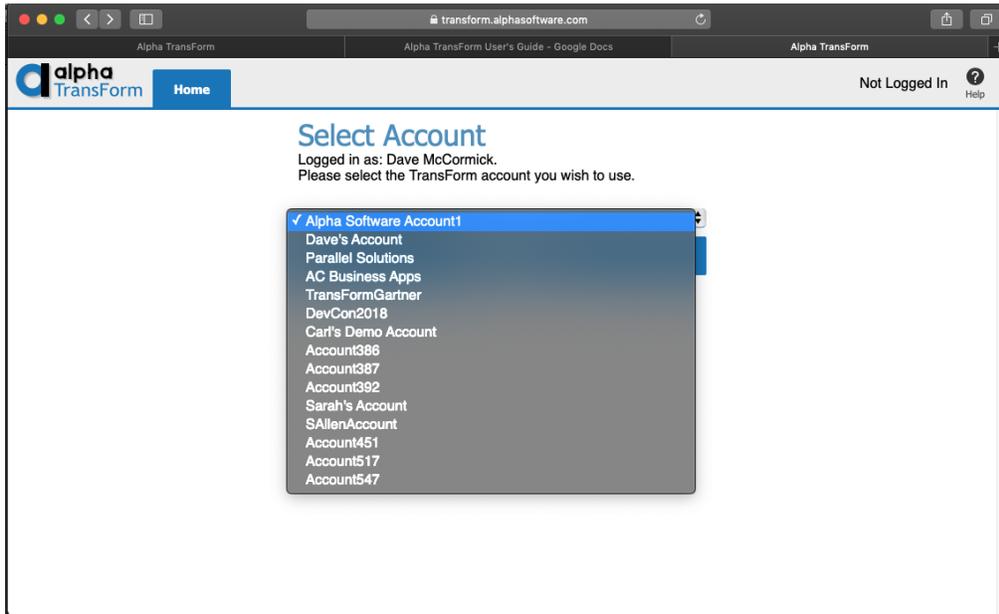
The screenshot shows the Alpha Transform web application interface. The browser address bar displays `transform.alphasoftware.com`. The navigation bar includes the Alpha Transform logo, a 'Home' button, and menu items for 'Designer', 'Permissions', and 'Management Console'. The user profile 'Dave McCormick' is visible with a 'Logout' link and a login expiration notice for '14-Dec-18'. The left sidebar lists various account settings and management options, with 'License Keys' selected. The main content area is titled 'License Keys' and contains the following text: 'The number of users authorized to log-in to this TransForm account is determined by the license key(s) you specify here. If you do not specify any license keys, this TransForm account will operate in a demo mode for a certain period of time. You can enter multiple license keys here. For example, you may have purchased a 5 user license key and then later purchased a 10 user license key. If you enter both license keys here, this TransForm account will allow up to 15 users to log-in. You can purchase license keys in the Alpha Software on-line store, or by contacting an Alpha Software sales person. Enter license key(s) for this account in the text box below. Enter one license key per line.' Below this text is a text input field containing two license keys: `07e02234f38047dbbb6b93732513f445` and `fe9216ff8dcf448990302ecd6193a95d`. There are two buttons: 'Save' and 'Show License Information'. The 'Show License Information' button is active, displaying a summary box with the following data: 'Licensed user count: 13', 'License key: 07e02234f38047dbbb6b93732513f445 - User count: 10 Expires: 2019-01-17', and 'License key: fe9216ff8dcf448990302ecd6193a95d - User count: 3 Expires: 2019-10-02'.

By the way, you don't need to have a license to use someone else's account. When someone invites you to their account, you are covered by their license.

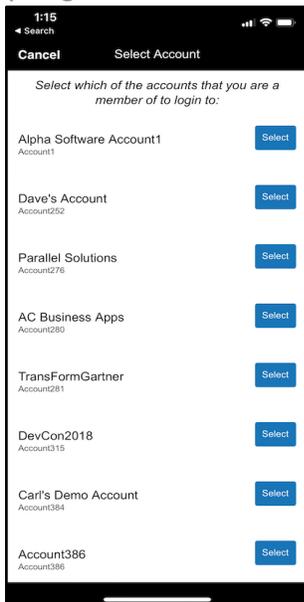
Switching Between Accounts

If someone has invited you to use their account, you will be given an option to select the account to use when you log in to either the TransForm app or TransForm Central.

(Image 2.9 - Select Account in TransForm Central)



(Image 2.10 - Select Account in the TransForm App)



Keep in mind this option is only available when you log in, not when you confirm login. That means if you are already logged in, or your login has expired, you need to select

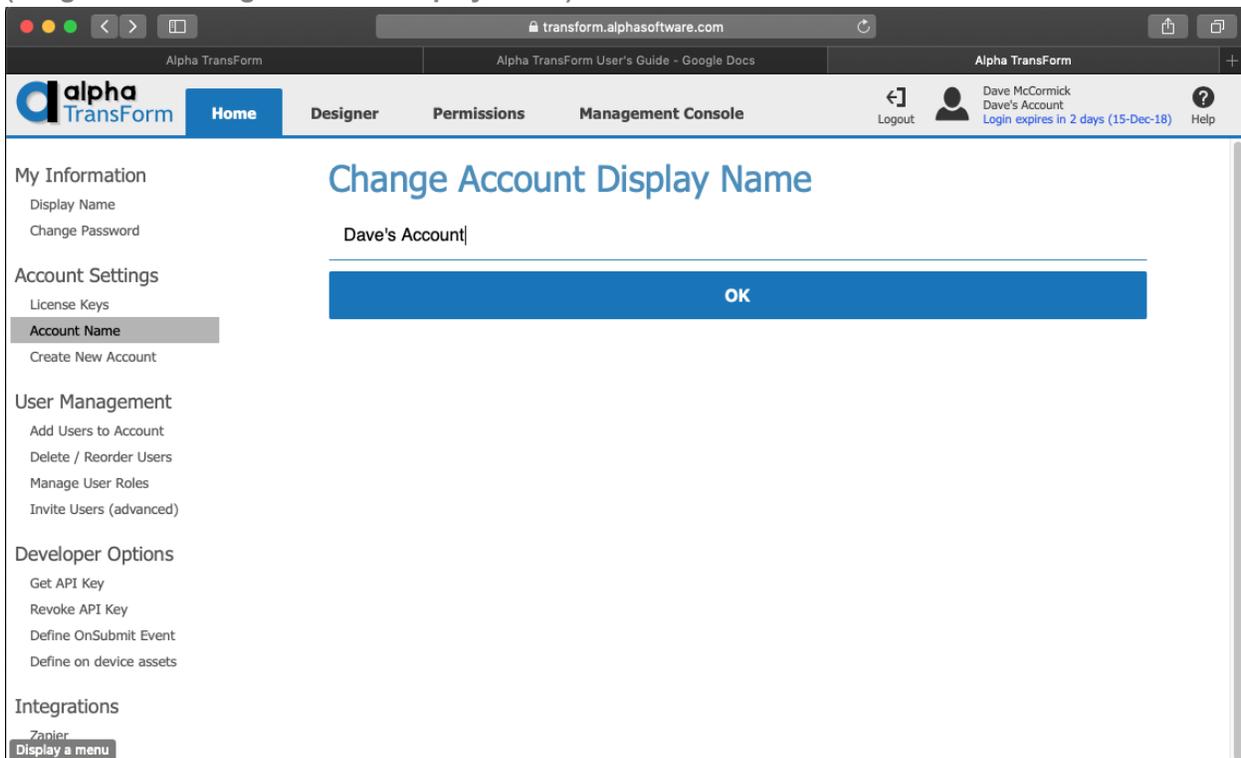
the Logout option in TransForm Central or the TransForm app and then log in again before you are presented with a choice of accounts.

Naming Your Account

When you first create a TransForm account, the system automatically gives your account a name with numbers at the end, like Account123. When you log in and are choosing from multiple accounts in a list, this default name isn't very helpful, so you should change it to something that is more descriptive.

To change your account name, go to the Home tab of TransForm Central and click the Account Name link in the left menu. Then enter a name for your account into the box and click OK.

(Image 2.11 - Change Account Display Name)



If you want, you can also change your display name, which is your actual name (e.g., Bob Smith) as opposed to your account name (e.g., ACME Manufacturing). This name can appear in the Management Console when reviewing forms that have been submitted.

Assigning Usernames, Passwords, and Roles all at Once

TransForm offers a method for “power users” (i.e., people who are little more technical) to assign usernames, passwords, and roles all at once. This can streamline the process of adding users and assigning roles, and also gives you the option of assigning a password to new users.

To use this feature, go to the Home tab of TransForm Central and click the Invite Users (Advanced) link in the left menu. To see the format you need to enter on each line, click the hyperlinked text to expand the help explanation.

(Image 2.12 - Invite Users Advanced)

The screenshot shows the 'Alpha TransForm' web interface. The top navigation bar includes 'Home', 'Designer', 'Permissions', and 'Management Console'. The user is logged in as 'Dave McCormick' with a login expiration of 2 days (14-Dec-18). The main content area is titled 'Invite Users' and contains the following text:

You can invite other users to join this TransForm account (Dave's Account). Invited users will be able to fill in TransForm forms. If an invited user does not yet have a TransForm User Account they will be able to sign up for a TransForm User Account. If the invited user already has a TransForm User Account they will be added as a member of this TransForm Account.

Enter the email addresses of the users you want to invite to join this TransForm Account. **Enter one e-mail address per line.** Click [here](#) for advanced syntax options.

You can enter just the email address. When the user logs in, their display name will be the same as their email address. For example:
`fred.smith@somecompany.com`
`molly.richter@somecompany.com`

You can enter the email address and the user's display name. When the user logs in, their display name will be as specified here. Use this format: `Display name (email address)`. For example:
`Fred Smith (fred.smith@somecompany.com)`
`Molly Richter (molly.richter@somecompany.com)`

You can enter roles that the user should have. If you don't specify any roles, the user will get a default set of roles. To specify roles end the line with `|comma delimited list of roles`. For example:
`Fred Smith (fred.smith@somecompany.com)`
`Molly Richter (molly.richter@somecompany.com) |User,FormDesigner,MgmtConsole`

Automatically Creating New TransForm User Accounts

You can automatically create TransForm user accounts for users you would like to add to your account without first sending email invitations to these users. In this mode you will specify the password for the user's account. (The user can always change their password later). The syntax for this advanced mode is as follows:

`Display name (email address) |comma delimited roles|noinvite|password.`

For example:
`Molly Richter`
`(molly.richter@somecompany.com) |User,FormDesigner|noinvite|mollyr`

Chapter 3: Designing Forms Part 1 - The Basics

In this chapter you will see how to create your own TransForm applications (called form types) that include the most commonly used TransForm features. In future chapters, we will dive into some of the more sophisticated (and powerful) functionality you can add to your apps.

Forms vs Form Types

First, let's begin with some terminology. In Alpha TransForm, the terms **form type** and **app** are sometimes used interchangeably. That's because applications built in TransForm follow the metaphor of a form in that they contain a list of fields - and many TransForm "applications" are meant for data collection where they replace paper forms. (That said, it is certainly possible to create form designs that are simply meant to display information and not collect it.)

In TransForm however, there is a distinction between a **form** and a **form type**. A form type is the form's format - in other words what fields are included on the form and how are they organized. While a form is a filled in (or partially filled in) instance of a form type.

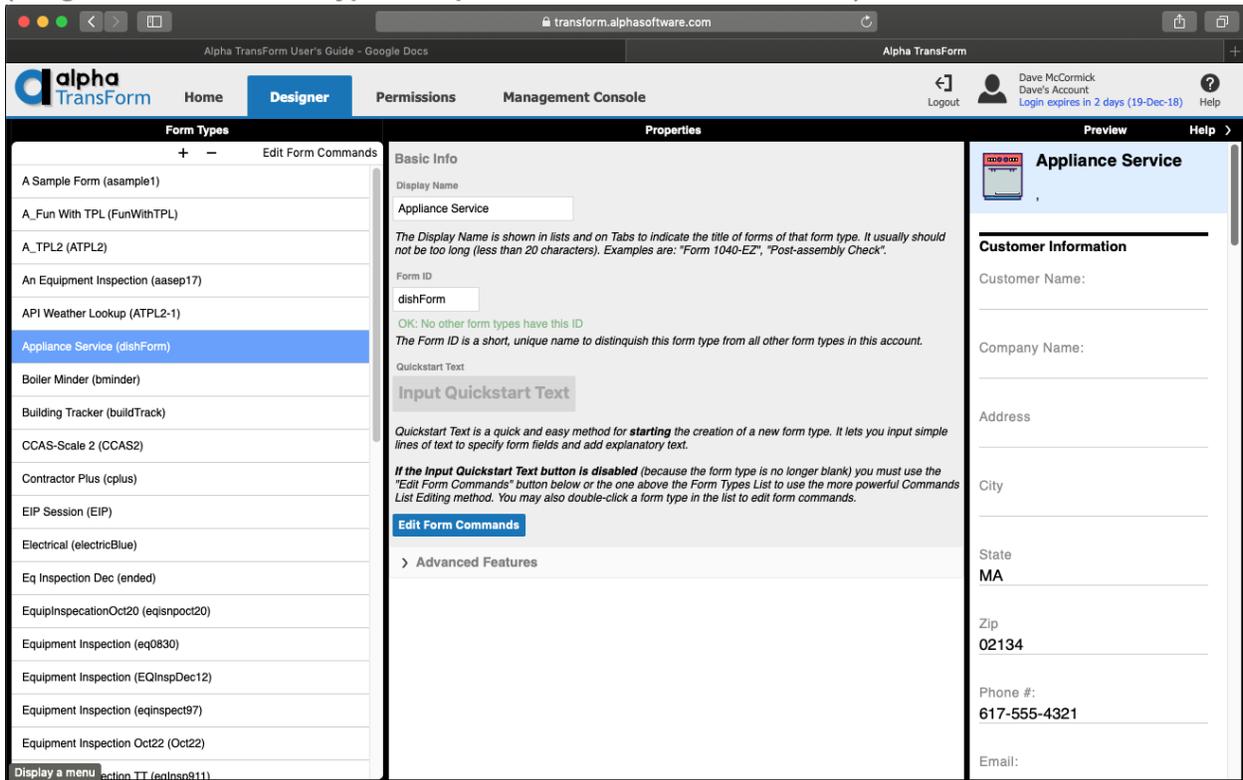
For example, you might have a form type called "Safety Inspection." During the day, you use this form type to perform three safety inspections. At the end of the day, you have three Safety Inspection forms - one for each of the inspections your performed.

The Designer Tab

Form types are created (and edited) on the Designer tab of TransForm Central. When you click the Designer tab, you are presented with a list of form types in your account. (When you first create your account, this list will be blank, since you have not yet created any Form Types.)

When you click on a form type in the list, you'll see properties for that form type (like the form type's name) in the Properties pane (in the middle), and you will see a preview of the form in Preview (on the right).

(Image 3.1 - List of Form Types, Properties Pane, and Preview Pane)

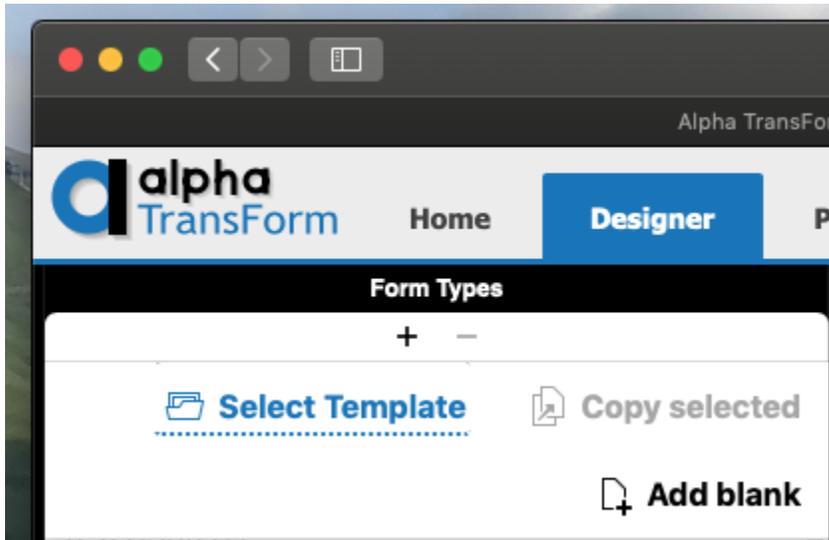


Creating a Form Type Using a Template

Alpha TransForm includes a collection of predesigned form types to provide examples of different ways you can use TransForm. There are form types for equipment inspection, travel expenses, home inspection, and more.

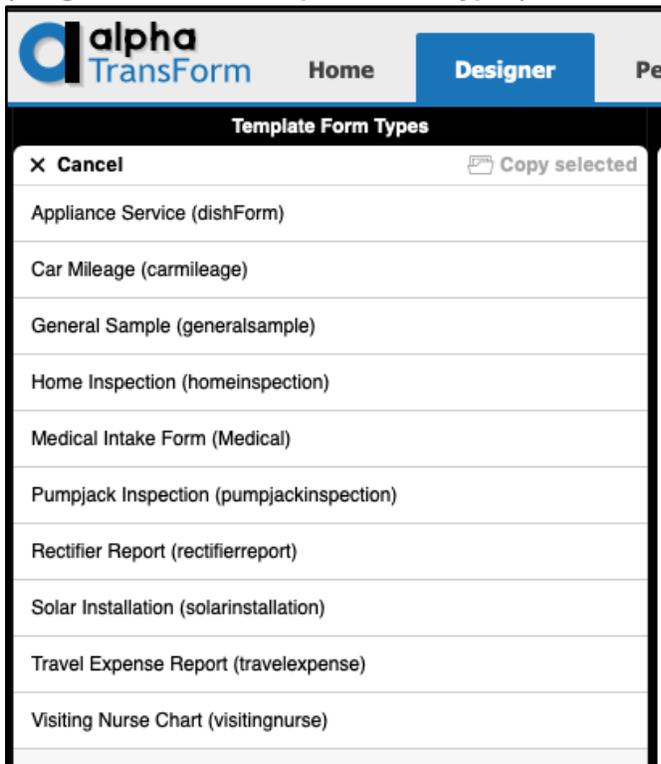
You get to the templates on the Designer tab in TransForm Central by clicking on the + button and choosing the Select Template option.

(Image 3.2 - Select Template Option in the Form Designer)



After you click Select Template, a list of choices appears.

(Image 3.3 - List of Template Form Types)



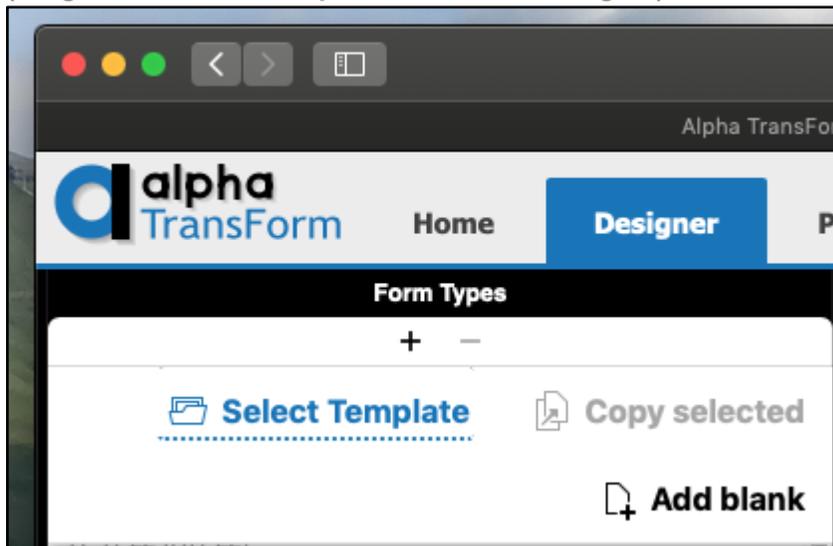
Click on the template you want to use, then click the Copy Selected button. TransForm adds the new design to your list of form types.

Creating a Form Type From Scratch

You can create your own form type on the Designer tab in TransForm Central.

Click the + button then click the Add Blank button.

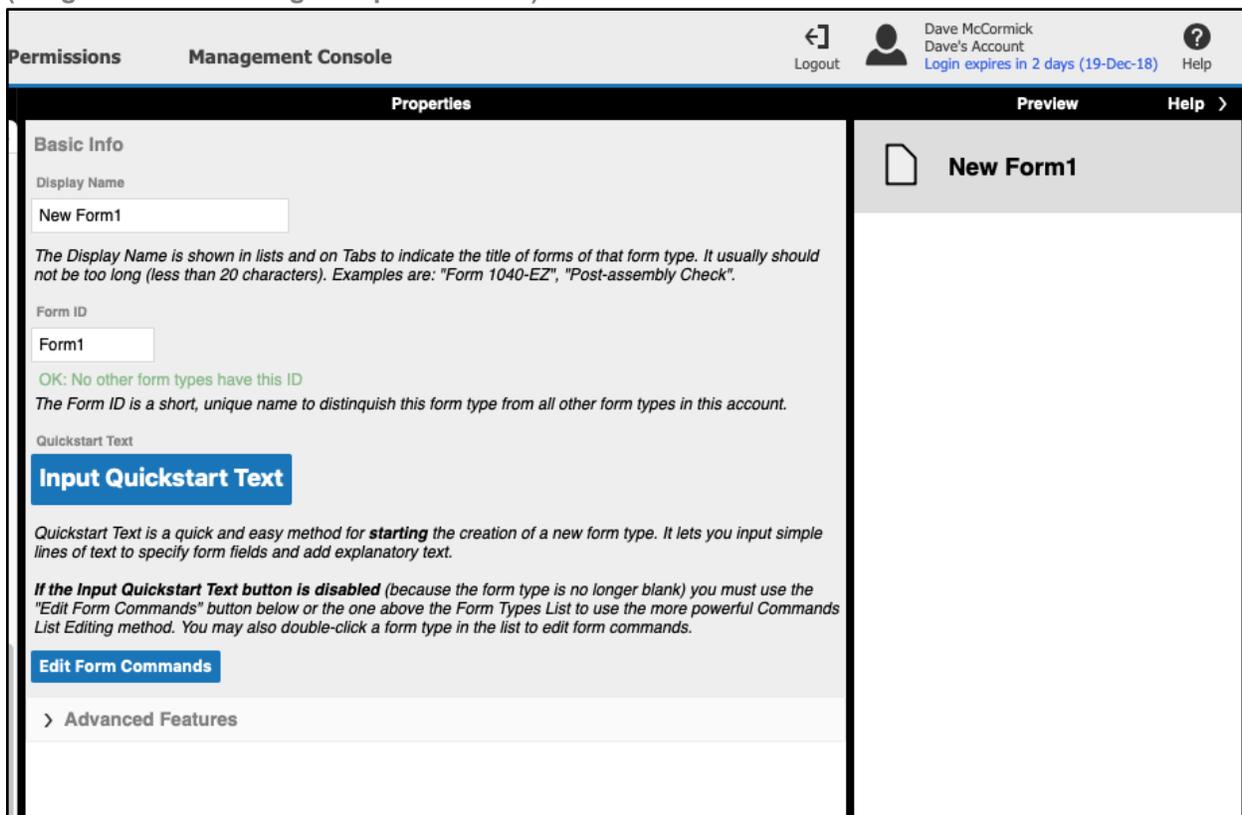
(Image 3.4 - Add Blank Option in the Form Designer)



When you first create a new form, TransForm automatically assigns the form a name called the Display Name and a unique identifier called the Form ID, which you can see in the Properties pane.

NOTE: Avoid confusion later. Change the Display Name and Unique ID to more descriptive names as soon as you create your form type.

(Image 3.5 - Form Design Properties Pane)



The Display Name is used in several places. It appears in TransForm Central in the list of Form Types on both the Designer tab and the Management Console tab. In the TransForm app, the Display Name also appears in the list of form types, as well as at the top of a form as it is being filled out. Because space on a mobile device is limited, you should limit the display name to 20 characters or fewer.

The Form ID, unlike the Display Name is not displayed on the form or in the list or forms. It is used by the system as a unique identifier for the form type. It is also used by developers and power users when working with the TransForm API, Zapier, and Alpha Anywhere. Generally, this name is kept short and does not contain spaces (for the developer's convenience).

After you have created a descriptive Form Name and Form ID, it's time to place fields and other elements onto your form type. There are two ways to do this. You can use the

Quickstart Text method - which is a very fast way of building your application. Or you can use the Form Commands method - which is not as fast but gives you more fine control.

NOTES: After you've created a new form type, the recommended way to begin is to start with the Quickstart Text method and then switch to the Form Commands method. These methods are described separately in the next two sections, starting with Quickstart Text.

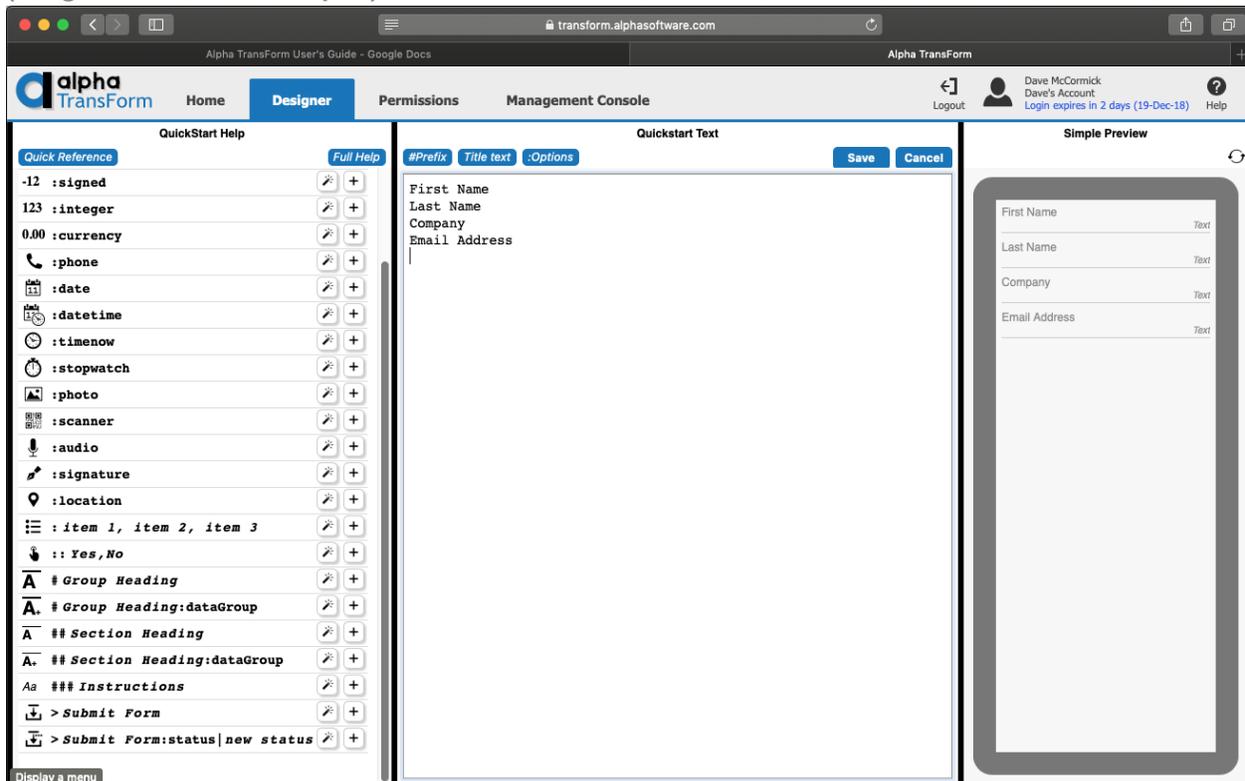
Quickstart Text

The Quickstart Text method follows a word processor metaphor. With Quickstart, you type each element on a separate line and then highlight each element (or multiple elements) to apply formatting. The difference is that instead of applying bold, italics and underlines - like you would in a word processor, you are instead assigning field types like "number," "photo," and "signature."

The Quickstart Method makes it extremely fast to add many fields to your form type. It also allows you to copy lists of fields from another source (like a PDF, a Word document, or a web page) and paste them right into TransForm.

As you enter the fields into the Quickstart Text box, a preview of your form type appears in the Simple Preview pane on the right. If you paste in a list, the fields may not appear in the Preview Window until you click the refresh button at the top right.

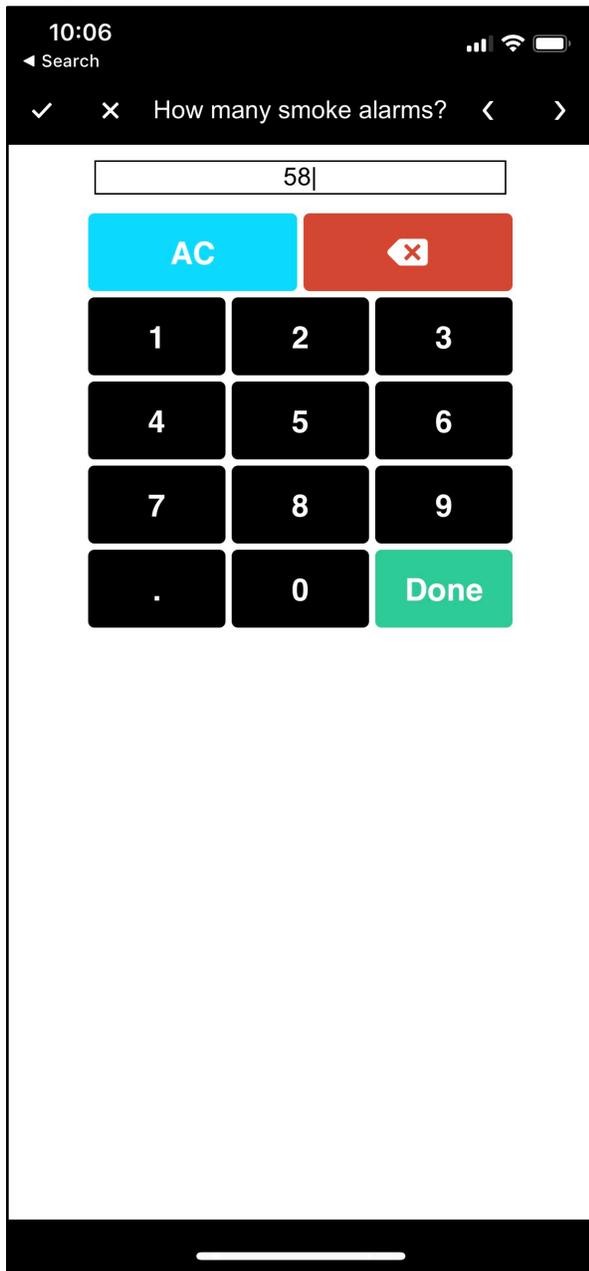
(Image 3.6 - Quickstart Input)



In the previous example (see image 3.6), there are four fields and all of them are text type fields, which means they can contain text (letters, numbers, symbols, spaces). When the user fills in these fields, they are presented with a standard keyboard with which to type including a mic button to use for dictation.

Besides the text type of field, TransForm has dozens of other field types, including numbers, dates, photos, audio recordings, and barcode scans. When the user fills in these kinds of fields, instead of the standard keyboard, they are presented with an interface (called an editor) best suited for that field type.

(Images 3.7, 3.8 - Numeric Field Editor, Photo Field Editor)



While entering text into the Quickstart Text box, there are a few ways you can set the types of your fields. The easiest is to click on the field you want to change (you can also highlight several fields at once by clicking and dragging), then find the field type you want from the Quick Reference list on the left and click its magic wand button.

(Images 3.9, 3.10 - Quick Reference List, Magic Wand Button)



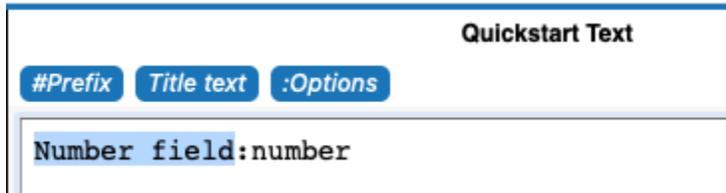
Getting Quick at Quickstart

You may have noticed that next to each of the magic wand buttons in the Quick Reference list is another button, marked with a "+."

If you click one of these + buttons, a new control of that type is inserted into the list. Furthermore, the name portion of the field is automatically highlighted so that when you start

typing in the fieldname, the placeholder name is overwritten.

(Image 3.11 - Highlighted placeholder text in Quick Start)



The benefit is that you can quickly enter text by following the pattern of:

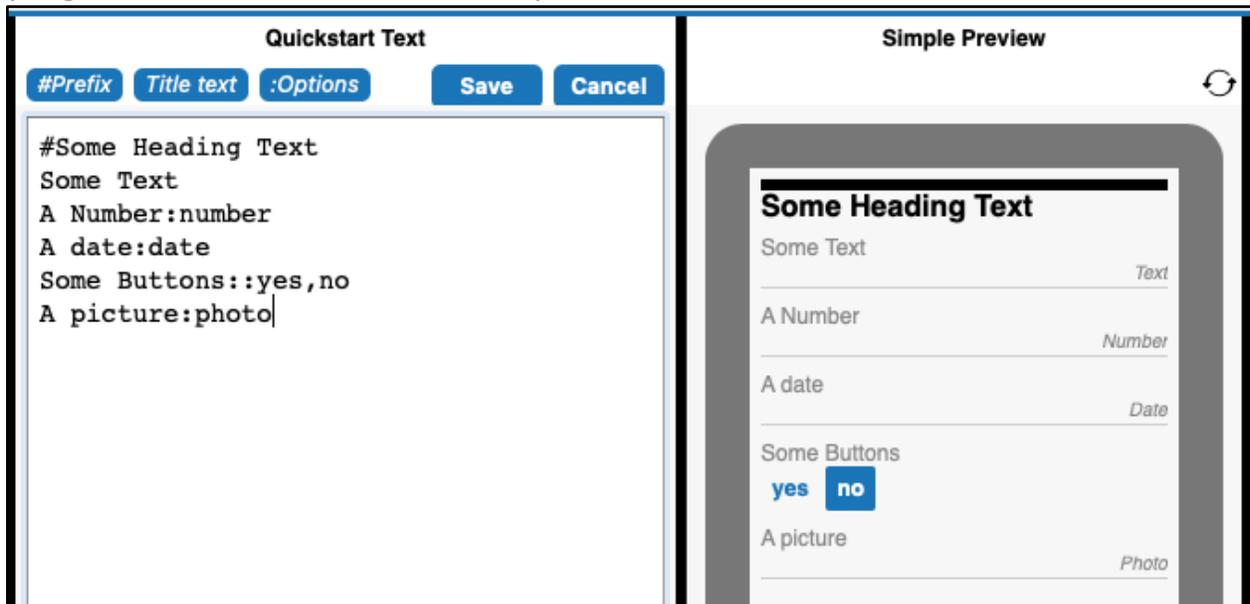
1. Click the desired + button
2. Type the fieldname
3. Repeat

Whether you choose to use this method or enter names first and then set their types with the magic wand buttons at the end is up to your personal preference.

Quickstart Notation

Strictly speaking, it is not necessary to use the + or Magic Wand buttons to enter field types. Instead you can type these in yourself using Quickstart Notation.

(Image 3.12 - Quickstart Text with Preview)



In the Quickstart Text box, notice how “Some Heading Text” begins with a “#” symbol. This indicates that the line should be a heading. Notice also that the “A Number” field ends with “:number,” indicating that it is a number. While the “Some Buttons” field is followed by “:.” and the choices “yes,no” indicating this field should be displayed as buttons and the choices for the buttons should be “yes” and “no.”

As you get more proficient at building form types, you may find it faster to simply enter some of this notation yourself rather than using the + and magic wand buttons. You can find the proper notation for each field type in the Quick Reference List on the left.

(Image 3.13 - Partial list of the field types in the Quick Reference List)

Quick Reference	
+ Add new line ✨ Change/Set current line(s)	
Aa text	✨ +
1.23 :number	✨ +
-12 :signed	✨ +
123 :integer	✨ +
0.00 :currency	✨ +
☎ :phone	✨ +
📅 :date	✨ +

Quickstart Text is a One Way Trip

Once you have saved your work, you will no longer be able to make any changes to these fields using the Quickstart Text view. Instead changes are done in the Form Commands View (see next section). It is possible to add new fields to an existing form using Quickstart Text from the Form Commands view by clicking the + button in the toolbar and choosing Quickstart Text from the menu.

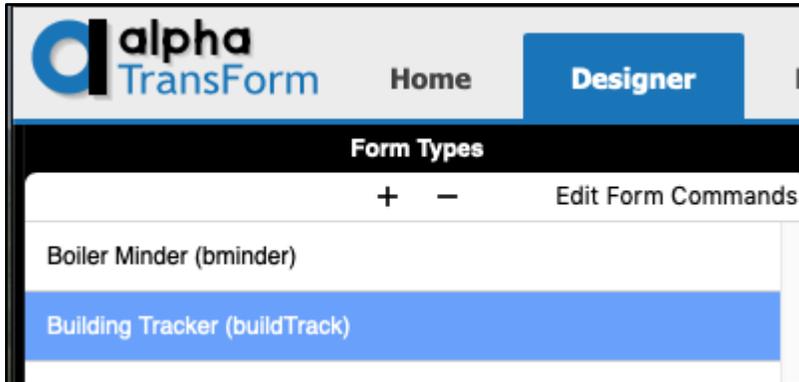
Form Commands View

The Form Commands view is similar to the Quickstart Text view in that it allows you to add fields to your form design and shows you a preview of your form design in the pane

on the right. However, the Form Commands view is much more powerful, as you'll see in this section.

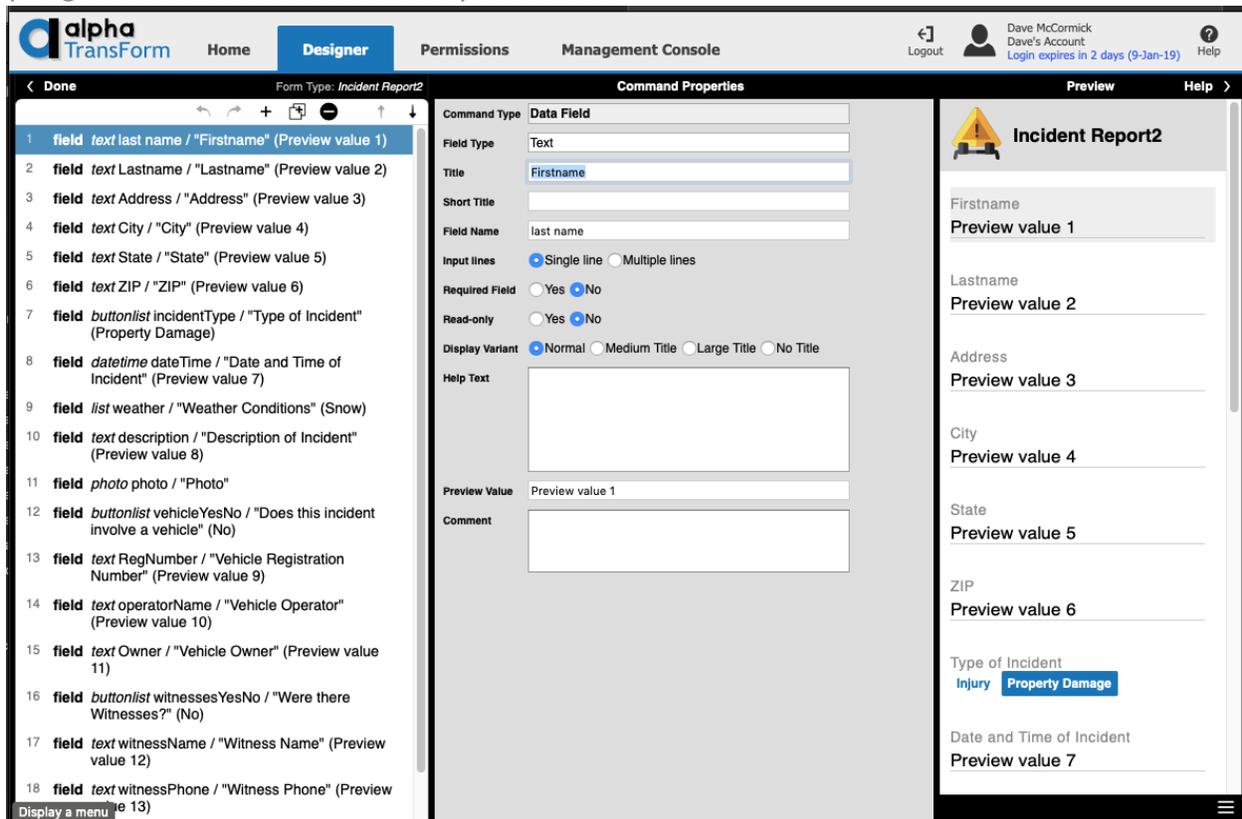
You get to the Form Commands by first clicking on the Designer Tab in TransForm Central.

(Image 3.14 - Edit Form Commands link)



From there, you can either double click on particular form, or you can single click on a form and click the Edit Form Commands link.

(Image 3.15 - Form Commands View)



Perhaps the most important benefit of the Form Commands view is that it allows you to delete, rearrange, and edit the fields that you have already inserted.

(Image 3.16 - Form Commands Tools Bar)



From left to right, these buttons let you:

- ✓ Undo your last action,
- ✓ Redo the last action you undid
- ✓ Add a new field or form command
- ✓ Duplicate the selected field(s)
- ✓ Delete the selected field(s)

- ✓ Move the selected field(s) towards the top of the design.
- ✓ Move the selected field(s) towards the bottom of the design.

The Form Commands view also lets you make fine tuning adjustments to fields you have placed. For example, if you want to change the resolution at which images are saved, or make a text field span multiple lines, you do that in the the Form Commands view. (See the Appendix for a complete listing of all of the form commands and options.)

Plus, the Form Commands view allows you to add other functionality that is not available on in the Quickstart Text view, like the ability to ability to add IF/THEN commands to conditionally show and hide fields (see Chapter 5).

Headings and Pages

When a form design has more than a few fields, it may be helpful to the user to divide the fields into sections. This can be accomplished using headings and pages. Headings are text that appear above a group of fields.

There are three types of headings: group headings, section headings, and instruction headings. Group headings are the highest level.

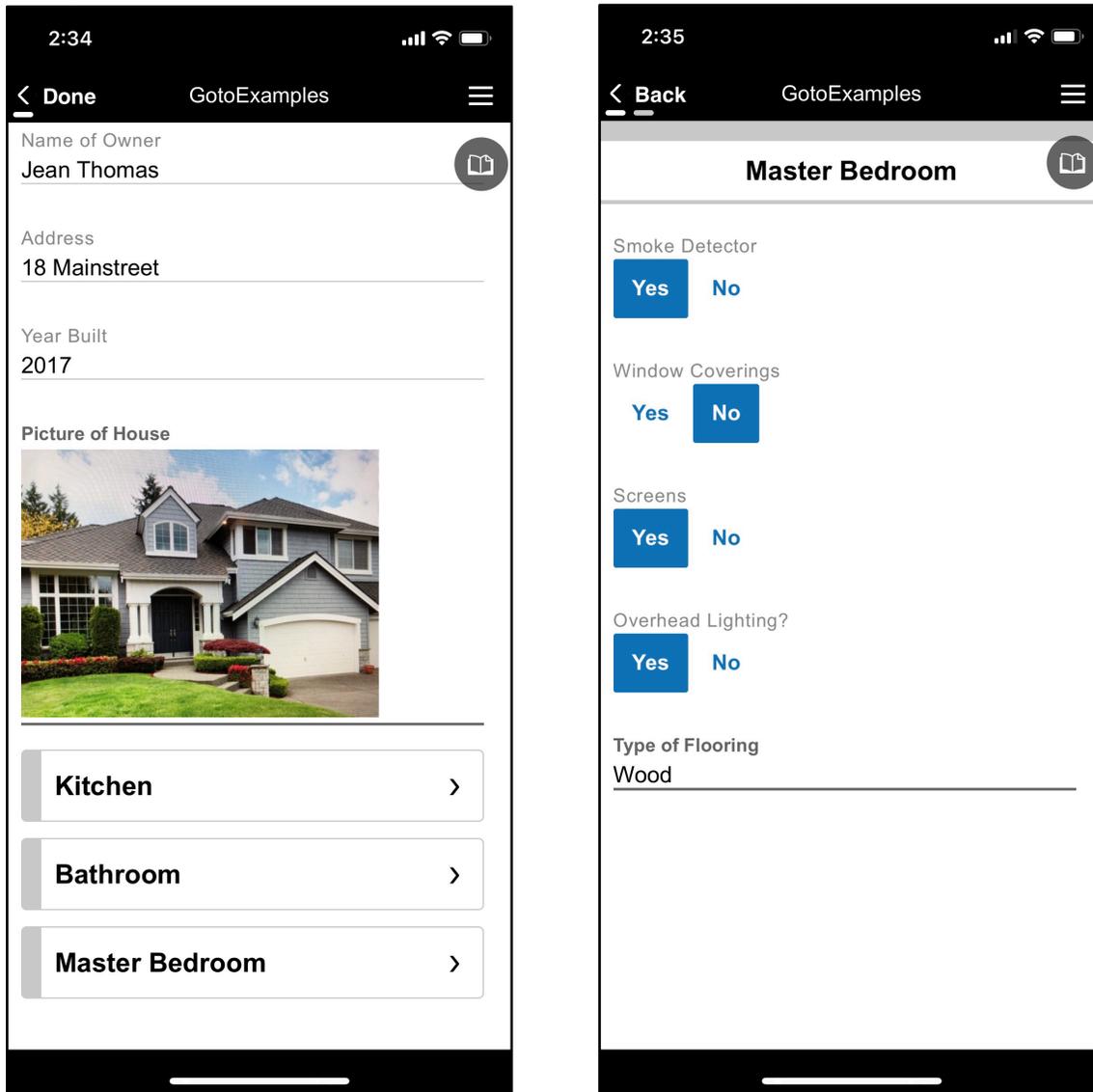
(Image 3.17 - A Form Type With Section Headings, a Group Heading, and an Instruction Heading)

The image shows a mobile application interface for adding a contact. At the top, the status bar shows the time 1:39, signal strength, Wi-Fi, and battery icons. Below the status bar is a navigation bar with a back arrow, the text "Done", the title "Contacts", and a hamburger menu icon. A grey bar below the navigation bar contains a plus icon in a circle. The form is organized into sections separated by thick black horizontal lines. The first section is titled "Name" and contains two input fields: "Firstname" and "Lastname". The second section is titled "Contact Info" and contains one input field: "Email Address". The third section is titled "Phone Numbers" and contains three input fields: "Mobile Phone", "Work Phone", and "Home Phone". At the bottom of the form, there is an instruction: "Leave Home Phone blank if it is the same as the Mobile phone number."

To add a heading, click the + button and select the type of heading you would like from the Form Commands list.

In addition to headings, fields can be organized using pages, as shown in the next image. When the user clicks on the page button, the page is opened.

(Image 3.18 - Master Bedroom Button Opens Master Bedroom Page)



Data Groups

Data Groups allow you to collect (or display) repeating sets of information. For example, suppose you were taking inventory of hundreds of products. For each product, you want to enter the product's ID number and the number in stock. Rather than placing hundreds of fields on your form design, you can instead place two fields: the ID and the quantity. Then you could enclose those fields in a data group. Each time you enter information

about a new product, you click the + button in the app to add a new item to the data group.

(Image 3.19 - A Form With a Data Group)

The screenshot shows a mobile application interface for an inventory system. At the top, the status bar displays the time 1:49, signal strength, Wi-Fi, and battery icons. Below this is a navigation bar with a back arrow, the text 'Done', the title 'Inventory', and a hamburger menu icon. The main content area is a form with the following fields and elements:

- Inventory Date:** 2019-01-29
- Employee:** Jane Smith
- Section Header:** Product Counts
- Product ID:** ABC-001
- Quantity:** 12
- Action:** X Delete Product Count #1
- Product ID:** ABC-002
- Quantity:** 9
- Action:** X Delete Product Count #2
- Footer:** + Add Product Count

A circular icon with a document symbol is located in the top right corner of the form area.

You can include as many data groups as you need in a form design, and you are allowed to place a data group within a data group if you want. For example, you could use TransForm to inspect multiple pieces of equipment in a facility on the same form, and for each piece of equipment you might want to take multiple pictures. You can have a data group for equipment and inside that a data group for pictures.

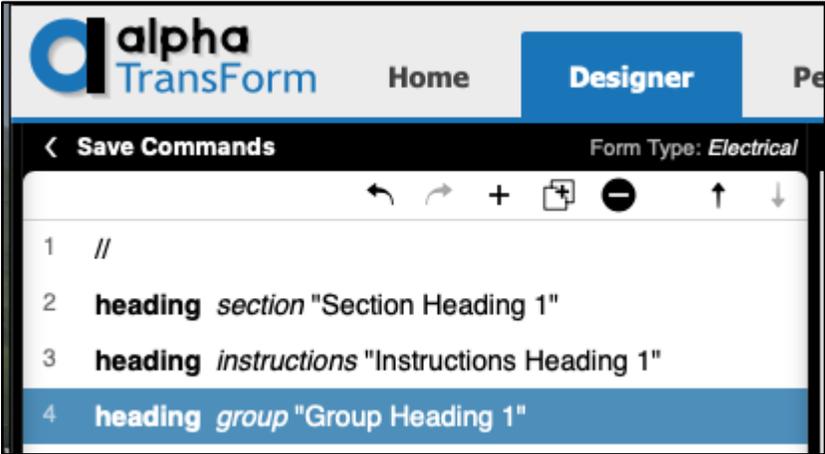
You can add or embed data groups as many layers deep as you need to.

Saving Your Design

When you are done designing your form type, you can make it available to the TransForm

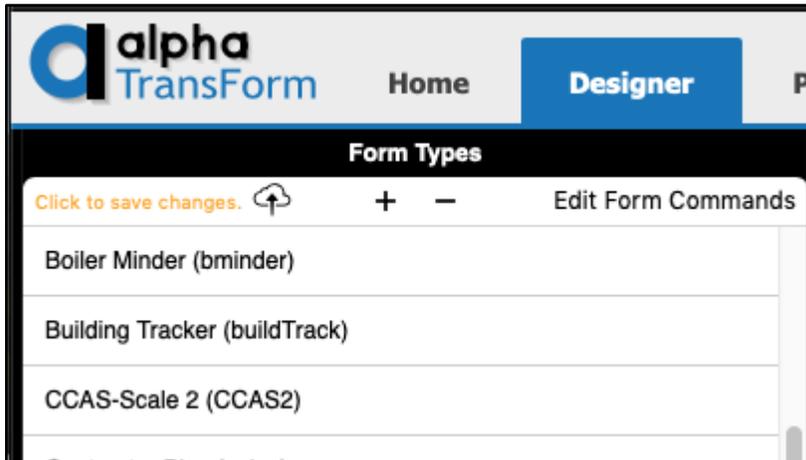
App in two clicks. First, click the < Save Commands link.

(Image 3.20 - Save Commands Link)



This indicates that you want to save the changes you made. TransForm then brings you back to the Form List. Next, click the Click to Save Changes link.

(Image 3.21 - Click to Save Changes Link)



This takes your edits and pushes them live to TransForm. You can now create a new form using this form type. To enter a new form on a mobile device, see the next chapter. You can also enter data into a form in TransForm Central on the Management Console tab. See Chapter 6 for details.

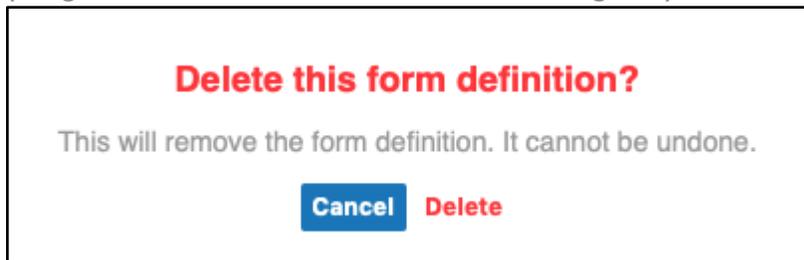
Editing an Existing Form Type

To edit an existing form type, click the Designer tab in TransForm Central. In the list of form types, double click on the form type you want to edit. The Form Commands View opens, and you can edit the form type. Refer back to Form Commands View section for details.

Deleting a Form Type

To delete a form type, click the Designer tab in TransForm Central. In the list of form types, click on the form type you want to delete. The form type is selected. Click the - button in the toolbar to delete the form type. A dialog box appears asking for you to confirm your choice.

(Image 3.22 - Delete This Form Definition Dialog Box)



Click Delete to delete the form type.

NOTES: Even though a form type has been deleted, any forms of that type that were created before you deleted form type are still in the TransForm database. If you want to delete both the forms and the form type, first delete the forms (see chapter 6), then delete the form type.

(Deleting forms may be important for users subject to GDPR, specifically Article 17, "Right to Erasure.")

If you have deleted a form type and want to access the forms again (provided you did not delete the forms), you can create a new, blank form type with the same Form ID as the form type you deleted. Your existing forms will appear in the Management Console (chapter 6), but you won't be able to create a new form of that type because the type has been deleted.

Chapter 4: Using the TransForm App

The TransForm App is a mobile application (for Apple and Android devices) that lets users fill out and review TransForm forms. This chapter shows you how to download and use the application. For some users, this is all they need to do - so this chapter is meant for all users.

Installing TransForm

To install the Alpha TransForm mobile app, you can use the QR codes below.

(Image 4.1 - iOS and Android Installation QR Codes)



Apple IOS



Android



Or you can search the Apple App Store or the Google Play Store for "Alpha TransForm."

(Image 4.2 - Alpha TransForm App Logo)

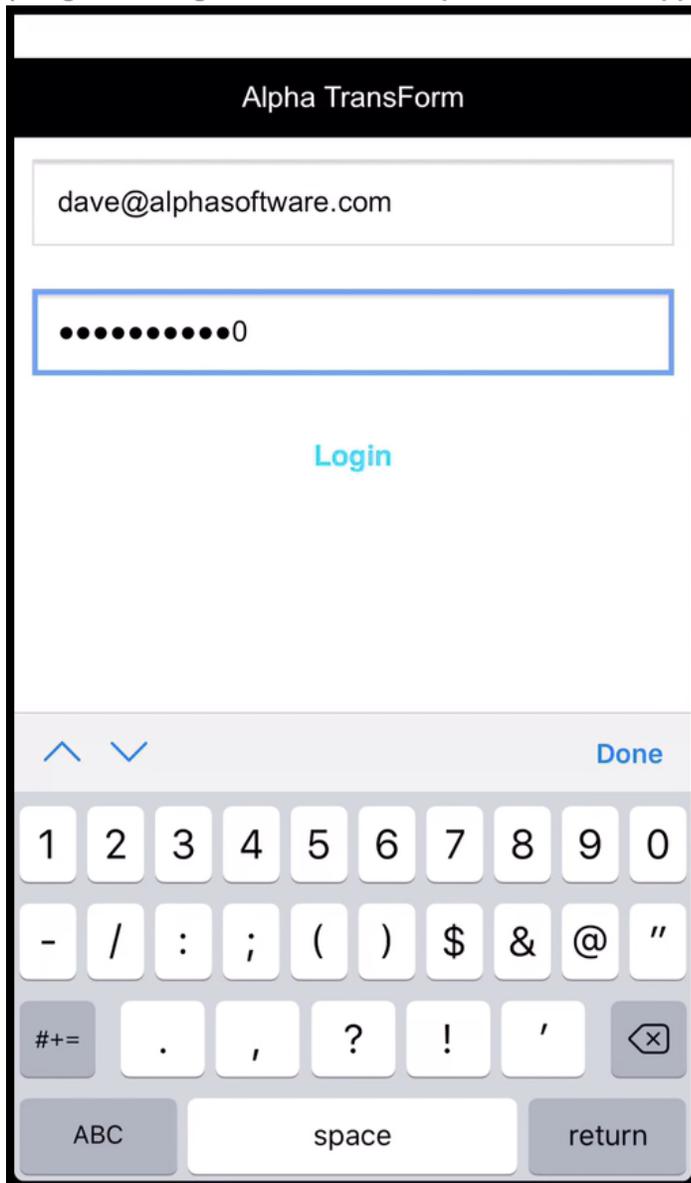


Logging in and Confirming Login

When you launch the TransForm application, you are prompted for a username and password. These are the same credentials you use to log in to TransForm Central.

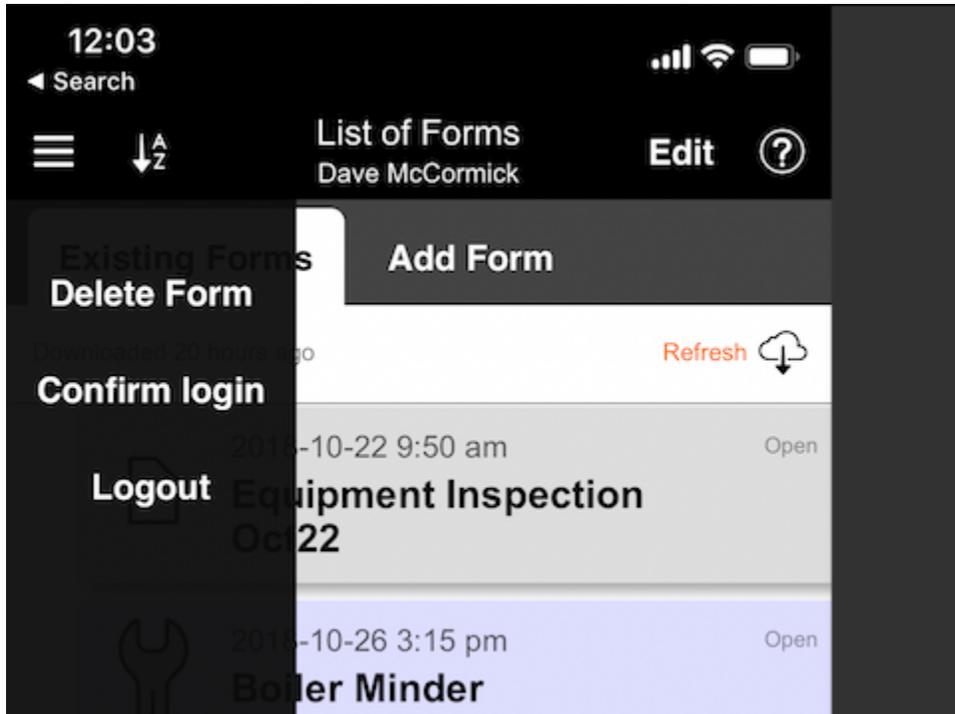
(TransForm uses an email address as the username).

(Image 4.3 - Login Screen of the Alpha TransForm App)



As with TransForm Central, your login for the TransForm app is also valid for two days before you need to confirm login. The app will continue to work after your login has expired, but you will not be able to synchronize data with the TransForm cloud until you confirm your login. To confirm login on the mobile app, tap the hamburger button  at the top left and choose Confirm login.

(Image 4.4 - Confirm Login in the TransForm Mobile App)



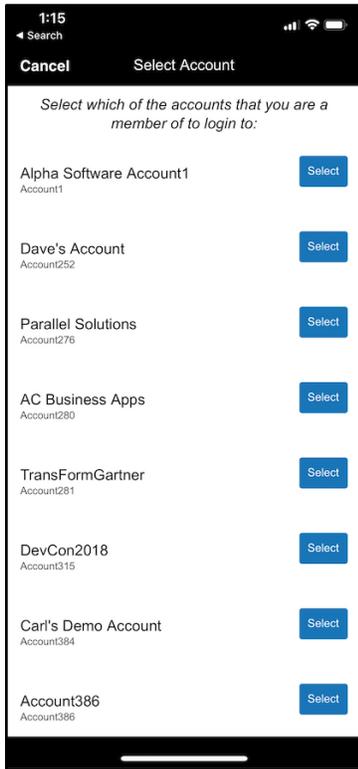
Resetting a Password

If you have forgotten your password or want to change it for security reasons, you can reset your password. This is done in TransForm Central, there is currently no option to reset or recover your password in the TransForm mobile app. See Chapter 3 for details.

Working With Multiple Accounts

If another user has invited you to their TransForm account, you will see a list of accounts after you have entered your password.

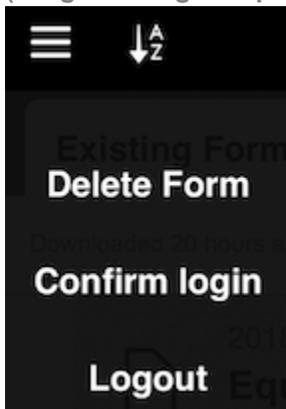
{Image 4.5 - Select Account List}



Tap the corresponding Select button to choose the account to use.

If you want to switch to a different account after you have made a selection, you must first save (upload) any work that has not yet been saved in the current account. Then choose Logout from the hamburger button  at the top left.

(Image 4.6 Logout option in hamburger menu)

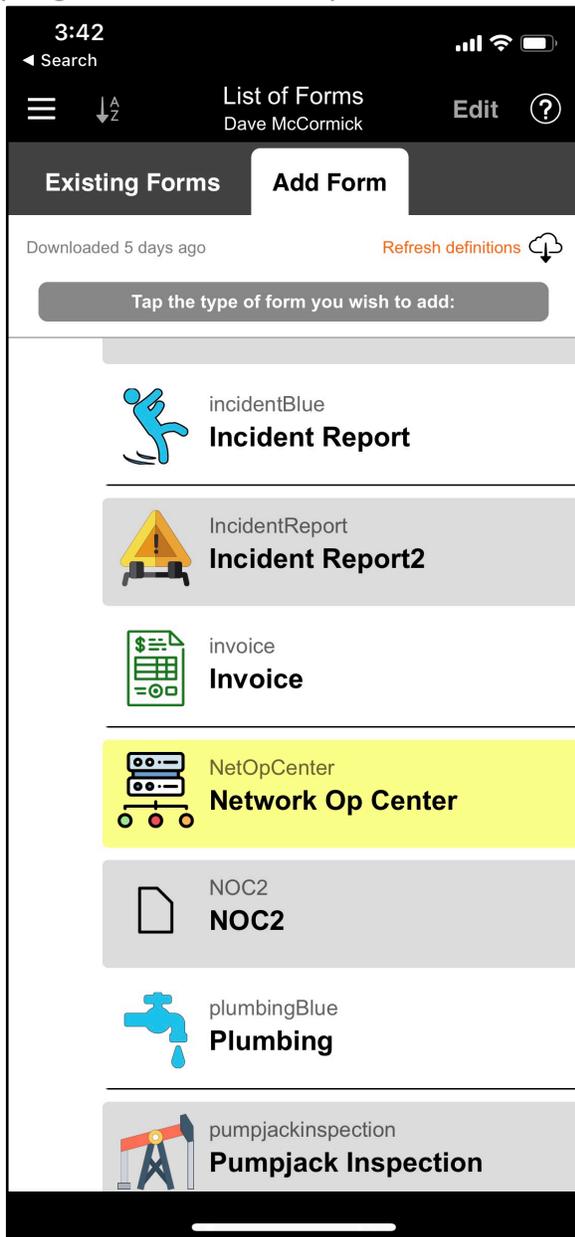


You must choose Logout (not Confirm Login) if you want to switch accounts. Confirm login is for logging back in to the current account.

New Forms

When you open the TransForm App, by default you are shown the Existing Forms tab, which lists forms that have already be filled out (or partially filled out). To fill in a new form, tap the Add Form tab.

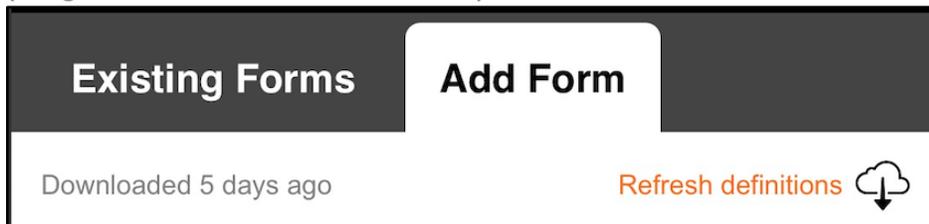
(Image 4.7 - Add Form Tab)



You are presented with a list of form types that you can use. To fill out a new form tap on its form type in the list and TransForm will create and open a new, blank copy of the form.

This list on the Add Form tab is automatically refreshed when you first login. After that, you can refresh the list by tapping the Refresh Definitions link at the top of the tab. Note that TransForm indicates how long it has been since new definitions were downloaded.

(Image 4.8 - Refresh Form Definitions)



For information about filling in the fields in a form, see the upcoming section Filling in Fields.

If you find a form definition is missing after you have refreshed the list, it is possible that the form type has been deleted or that you do not have permission to access that form type.

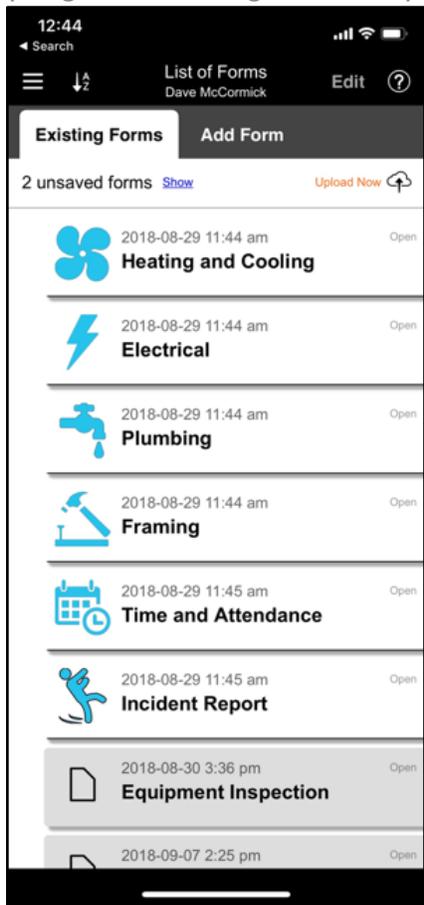
For information on deleting forms types, see Chapter 6. For information on setting permissions, see Chapter 9.

Existing Forms

When you open the TransForm App, you are presented with two tabs: Existing Forms and Add Form. The Existing Forms tab is open by default and shows you both forms you have filled out as well as forms that have been assigned to you.

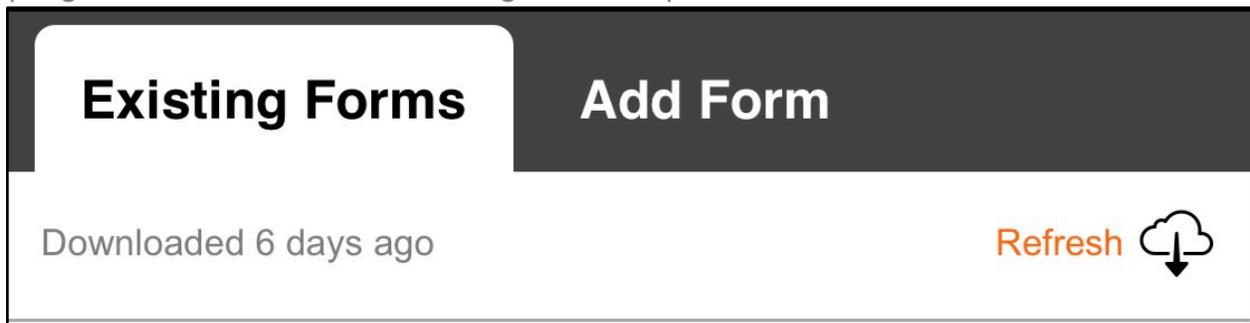
To view or edit an existing form, tap on the Existing Forms tab (if necessary) and then tap on the form in the list.

(Image 4.9 - Existing Forms Tab)



You can refresh this list by tapping the Refresh button at the top of the tab. If the Refresh button is missing and instead you see an Upload Now button, it means that you have data on your mobile device that has not yet been synchronized with the server. Tap the Upload Now button to synchronize. When it completes, the Refresh button will appear.

(Image 4.10 - Refresh Button on Existing Forms Tab.)



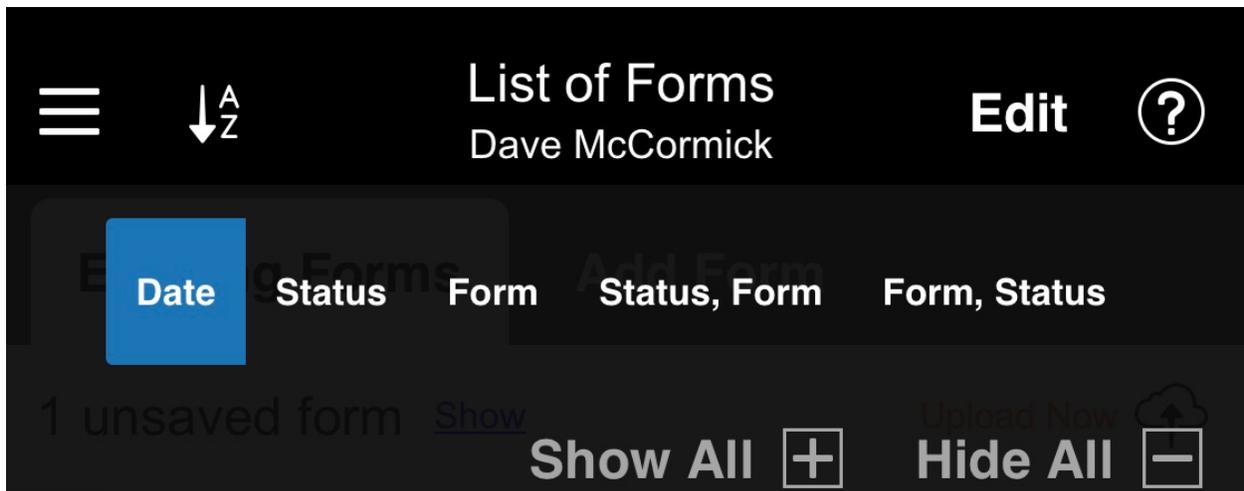
Sorting and Grouping Existing Forms

If you have many forms in the Existing Forms tab, it may be easier to sort them or group them. You can do this by tapping Sort button  at the top of the screen.

(Image 4.11 Sort button)



When you tap the Sort button, a menu drops down giving you options.

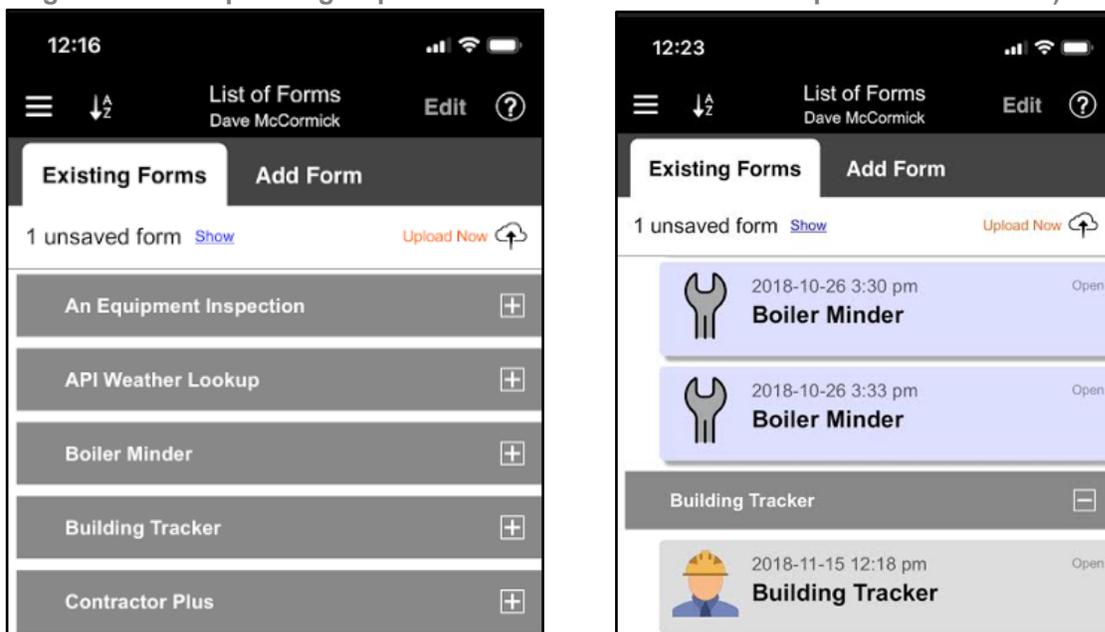


The options are as follows.

- ✓ **Date** - This is the default and sorts each form by the date in which it was created.
- ✓ **Status** - This groups forms by status (submitted, opened, closed, etc). See chapter 9 for a detailed discussion on statuses.
- ✓ **Form** - Forms are grouped by form type.

- ✓ **Status, Form** - Forms are first grouped by status, and then for each status, they are further grouped by form type.
- ✓ **Form, Status** - Forms are first grouped by form type, and then for each form type, they are further grouped by status.
- ✓ **Show All / Hide All** - If you have selected an option other than Date, you can decide whether to show all of the forms in the list, or hide the forms in sections that can be expanded.

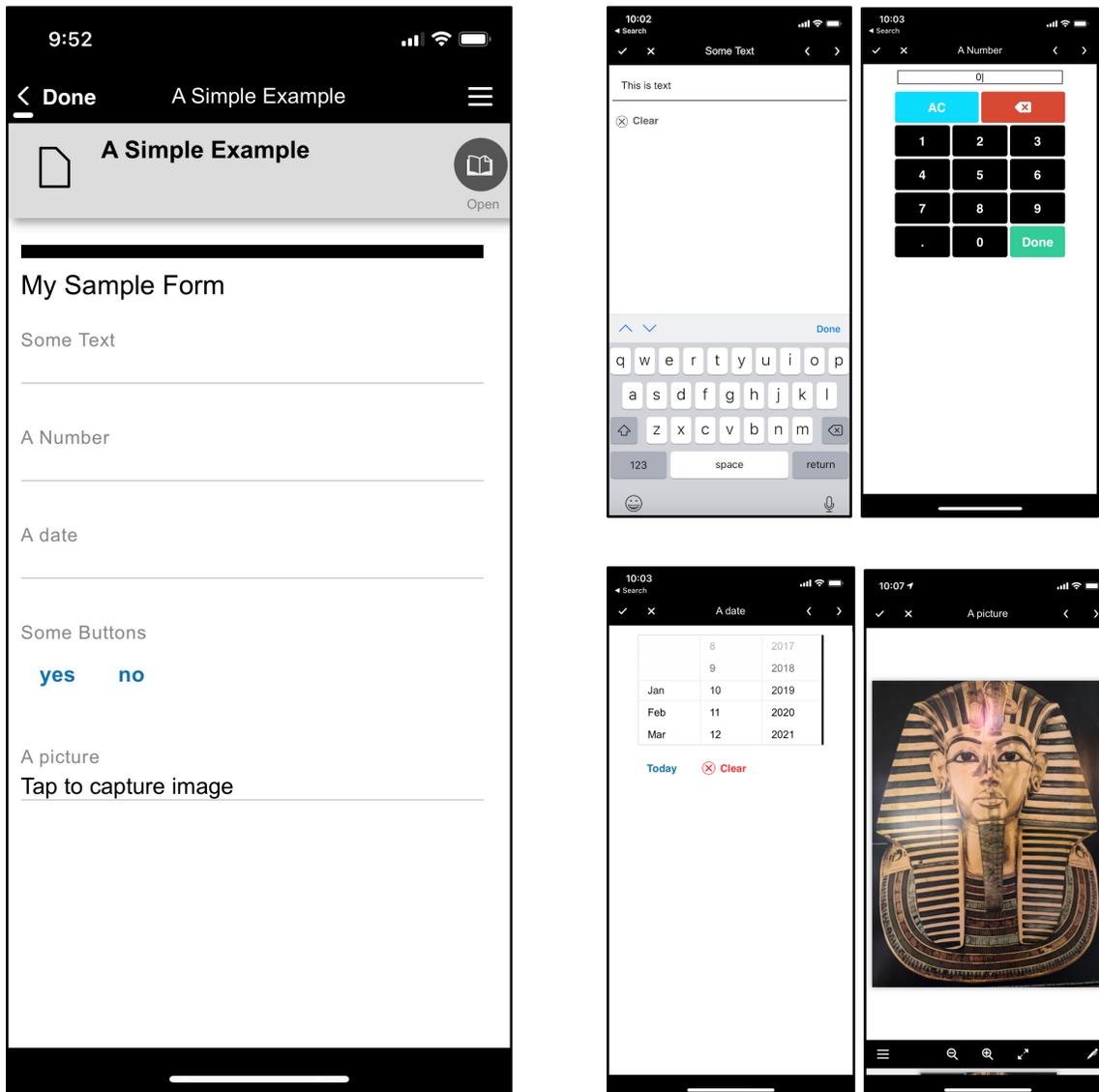
(Images 4.12 Examples of grouped forms hidden and shown in expandable sections.)



Filling in Forms

TransForm provides you special controls and keyboards depending on the type fields you are filling out. For text fields, you are presented with a keyboard. For numbers, you are presented with a numeric keyboard. For dates there is a date picker, and for photos there is a photo editor where you can take photos and annotate them.

(Image 4.13 - A Sample Form With Keyboards and Editors for Different Field Types)



These special keyboards and controls are called “editors,” and most field types have their own special editor that opens when you tap the field. (There are a few exceptions, like the Button List, which does not open an editor). A complete list of all of the field types and their corresponding editors appears later in the appendix.

NOTE: A text field is a little different than the other editors, when its editor appears you need to tap on the field in the editor to make the keyboard appear.

All of the editors have the same top toolbar, which you use to confirm your entry, cancel your entry, or move to the previous or next field.

(Image 4.14 - Editor Toolbar)



 Confirm what you have entered into the field and exit the editor.

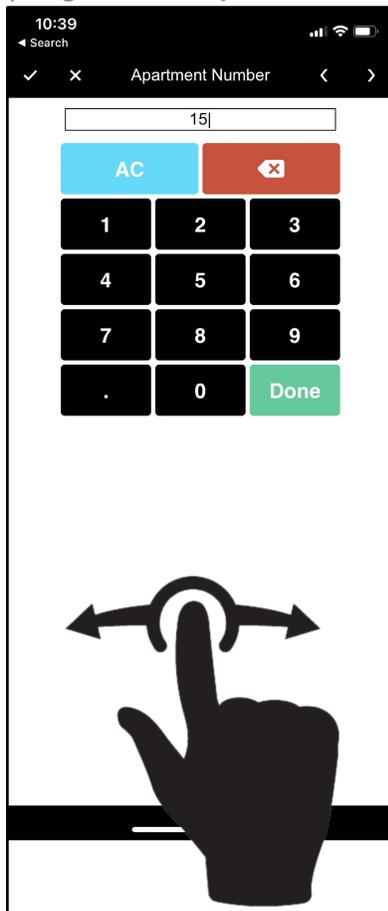
 Cancel what you have entered into the field and exit the editor.

 Confirm your entry and move to the previous field.

 Confirm your entry and move to the next field.

Besides these toolbar buttons, you can also swipe between fields. Swipe navigation is a fast way to enter data into fields and navigate from field to field.

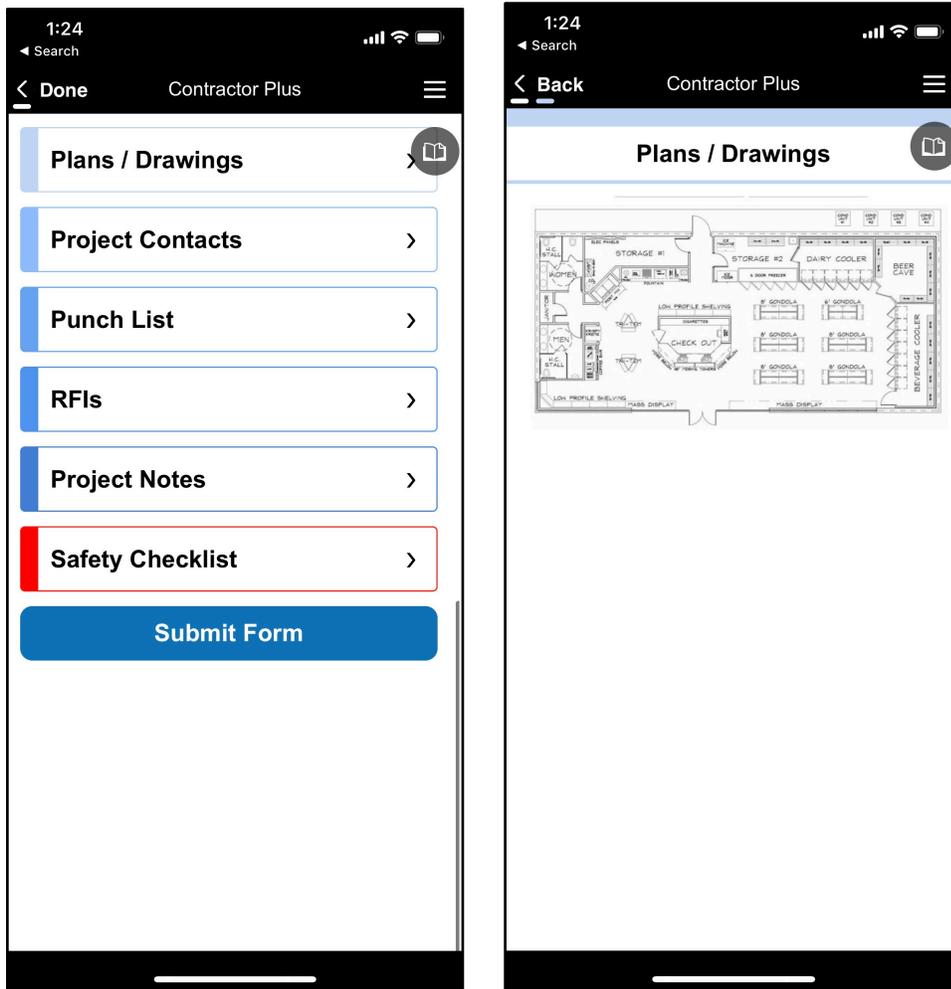
(Image 4.15 - Swipe Left and Right to Navigate Between Fields)



Navigating Large Forms

Forms can easily accommodate hundreds of fields if necessary. Often when forms have many fields, the form is designed so that related fields are grouped together on separate pages, as in the following example.

(Image 4.16 - A Form That Has Been Divided Into Pages)



In this example, when you tap the Plans / Drawings page button, it opens the Plans / Drawings page. Notice at the top of the page, that there is now a Back button to take you back to the previous page. There is also a depth marker to show you how deep you are into the form.

(Image 4.17 - Back Button and Depth Markers)



In this example, there are two lines (depth markers) under the Back button. This indicates that the user is on a page off of the main page. Some form types may have pages within pages, in which case there will be additional depth markers.

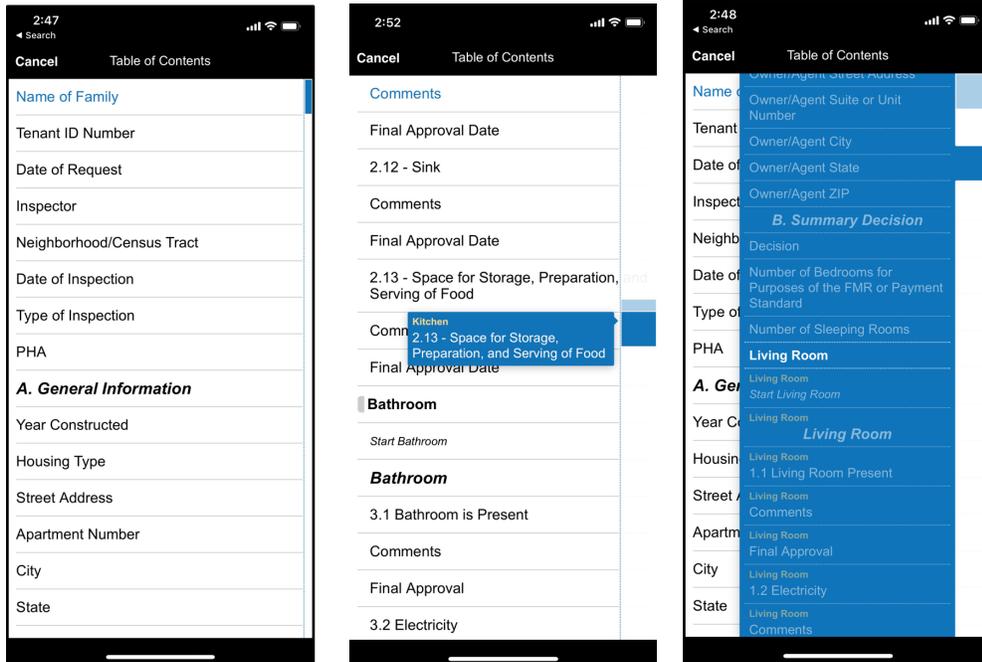
When you have many fields or many pages of fields, sometimes it can be difficult to find a particular field on the form. For these instances, you can use the Table of Contents by tapping the Table of Contents button.

(Image 4.18 - Table of Contents Button)



The table of contents appears showing you a list of all of the fields. You can use the scroll bar on the right to scroll through the fields. By tapping on the scroll bar, and sliding your finger to the center of the screen, you can switch between standard scrolling and fast scrolling.

(Image 4.19 The Form Table of Contents. From left to right, standard Table of Contents, Tables of Contents with Regular Scrolling, Table of Contents with Fast Scrolling.)



Once you find the field you want, tap on it and TransForm will bring you directly to that field on the form.

Finding and Fixing Data Entry Errors

Some form designs include error checking to help make sure information is properly captured. For example, let's say you created a form to record information about people living in an apartment building. One of the questions on the form asks how many children live in each apartment. Error checking could be set up to make sure that the person filling in the field enters a reasonable value, and it can display an error message if they don't.

(Image 4.20 Error Message)

The image shows a mobile application interface for a form titled "Census". At the top, the status bar shows the time 9:48, signal strength, Wi-Fi, and battery icons. Below the status bar is a navigation bar with a search icon, a back arrow, the text "Done", the title "Census" with a red star, and a hamburger menu icon. The form itself has a header with a document icon, the title "Census", and an "Open" button. The form contains three input fields: "First Name" with the value "Jim", "Last Name" with the value "Jenkins", and "Number of Children" with the value "5000". Below the "Number of Children" field, there is a red error message: "You must enter a number between 0 and 20".

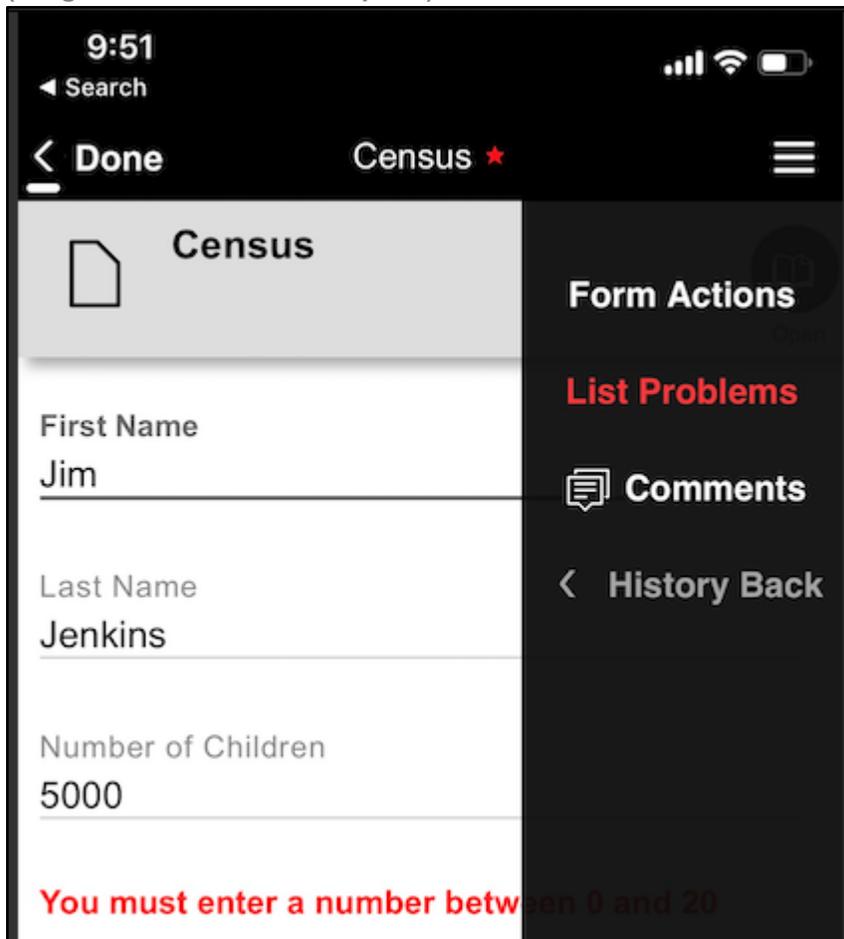
In this example, you can see the error text below the field with an error. There is also a red star next to the title of the form at the top, indicating the form has one or more errors.

This is a very simple form, but suppose you are working with a form that has hundreds of fields and some of those fields had errors. It could be cumbersome scrolling page after page looking for the mistake, so TransForm has a menu option that allows you to

list all of the errors on a single page, and you can tap on a particular error to jump directly to that field.

To get to the error list, open the Form menu by tapping on the hamburger button , then choose the List Problems option.

(Image 4.21 List Problems Option)



For information on adding error validation and error checking to forms, see Chapter 5.

Submitting and Uploading Form Data

In TransForm, there is a distinction between the terms Submitting and Uploading form data, even though they may sound like synonyms.

In TransForm, submitting a form means changing the status field in the form from Open to Submit. This generally indicates that data entry is complete and that the form is now locked from future editing.

Uploading, on the other hand, means taking the form data, which is saved locally on the device, and transferring (uploading) it to the TransForm cloud, where it can be accessed.

It is possible to upload a form that has not yet been marked as submitted. It is also possible to mark a form as submitted without uploading it.

Marking a Form as Submitted, Open, or Another Status.

Generally, a form is submitted by tapping a Submit button, which is part of the form's design.

(Image 4.22 - Inspection form with Submit button.)

11:49
◀ Search

◀ Done Equipment Inspection

Equipment Photo

Equipment Checks

Pass Fail

Sign

2019-01-14 11:48:38

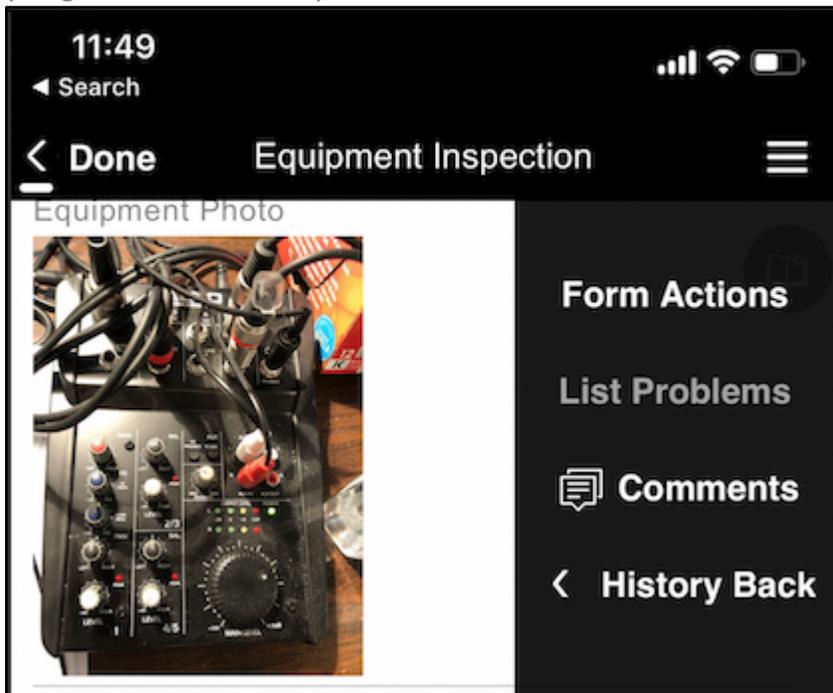
× Delete #1

+ New Inspection

Submit Form

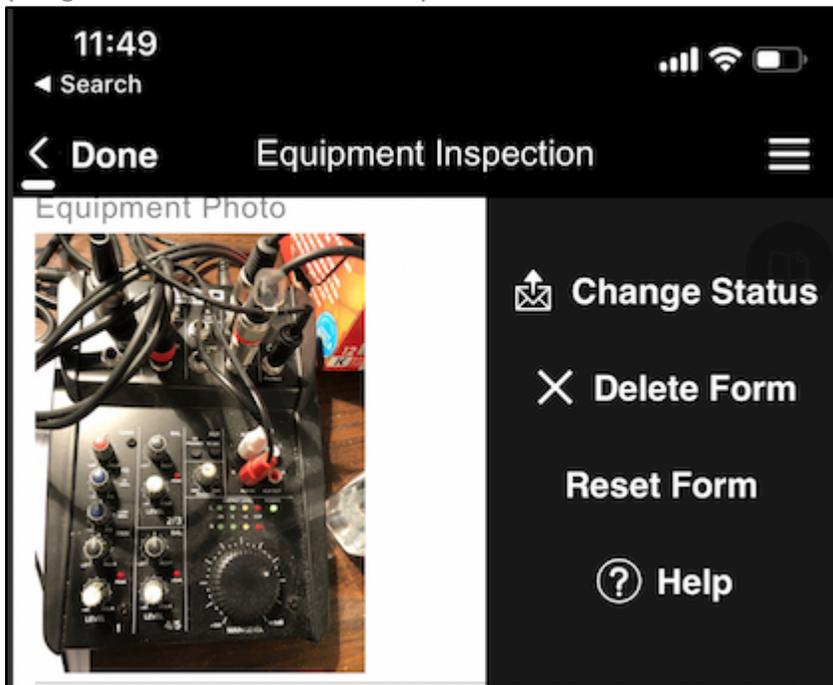
You can also change the form's status using an option found in the form menu, which you open by clicking the hamburger button .

(Image 4.23 - Form Menu)



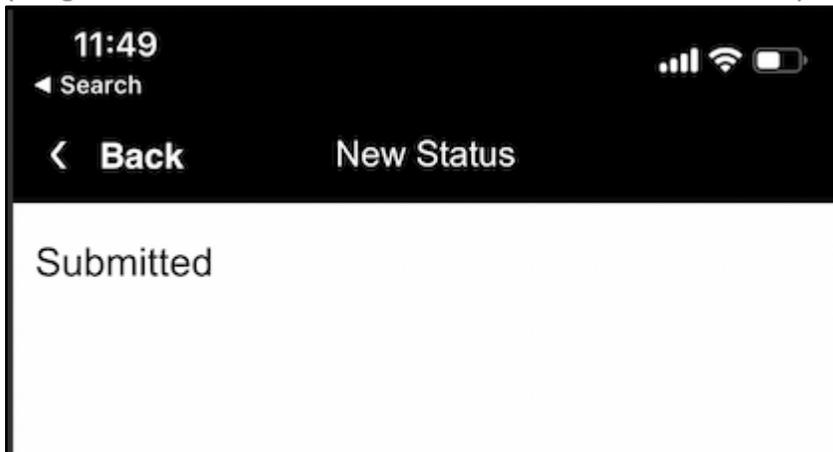
In the Form Menu, tap on Form Actions to open the Form Actions menu.

(Image 4.24 - Form Actions Menu)



Tap Change Status. Another screen appears showing you the statuses that you can select. Depending on how the form is designed, there may just be one choice (or even no choices).

(Image 4.25 - List of Statuses With Just One Choice, Submitted)



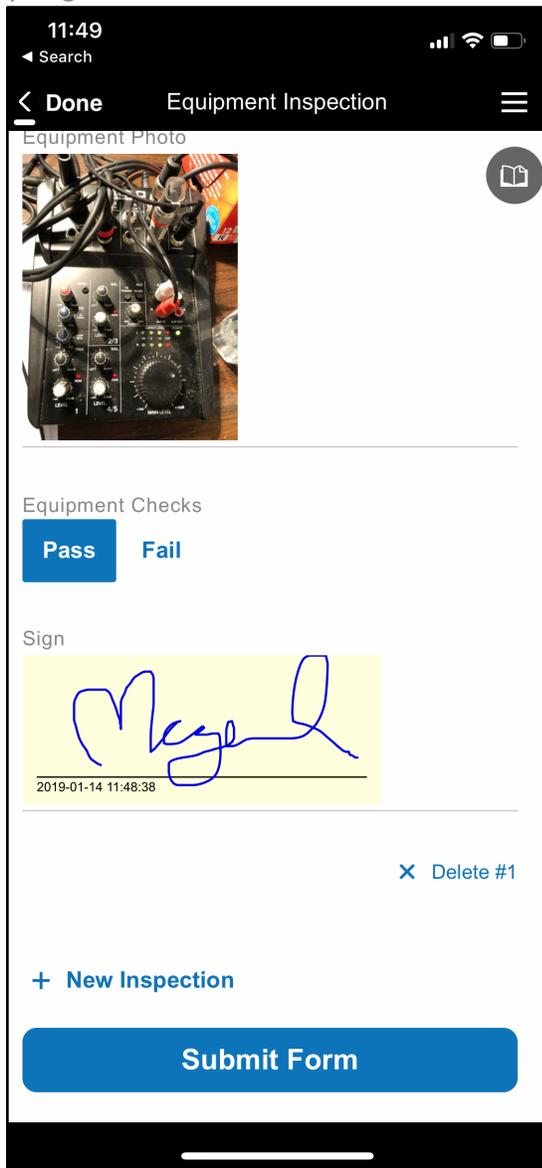
Tap on the status you want or tap Back to cancel out. Depending on how the form is designed, you may be able to go back into the menu and change the form's status back to Open or some other status.

For a detailed discussion on how to use statuses (and even create new ones), see chapter 9.

Uploading Form Data

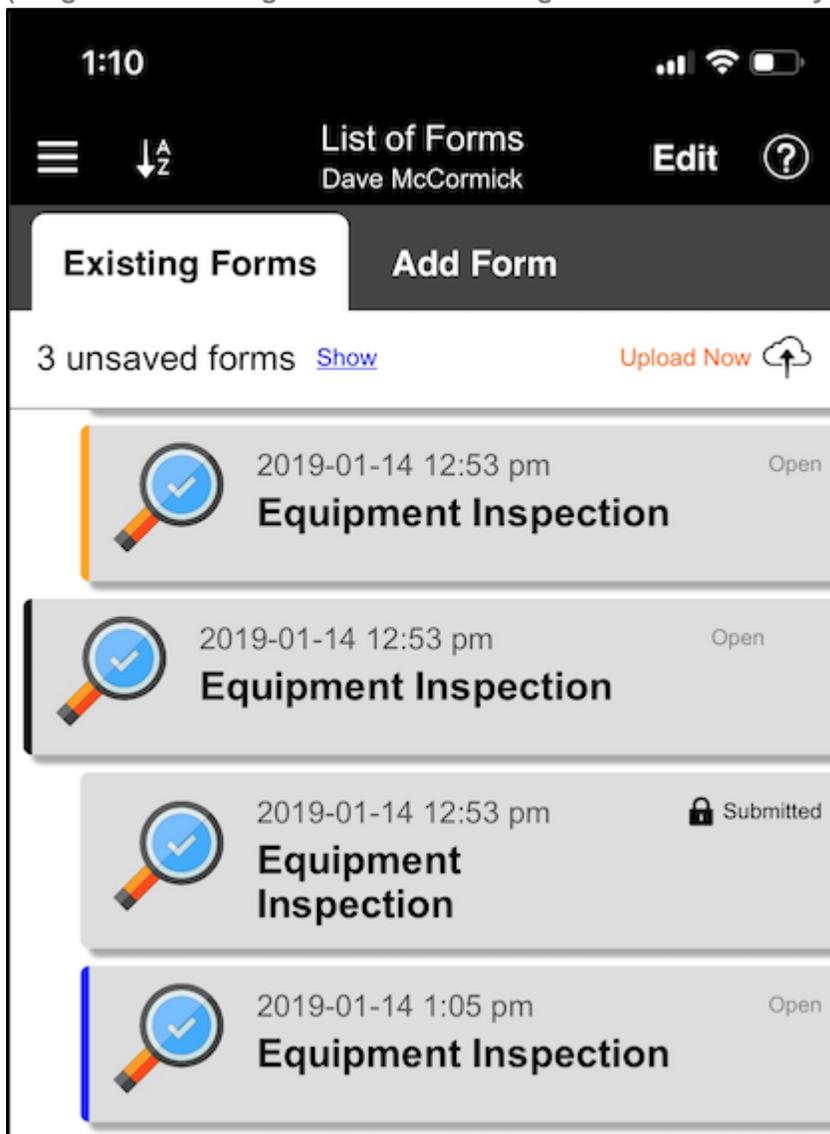
After you have finished filling in form data, you can upload the data to TransForm. This can be done from the Existing Forms tab. You get to this tab either by tapping a Submit button or by tapping the Done button.

(Image 4.26 - Form With Submit Button at the Bottom, Done Button at the Top)



Done takes you back to the Existing Forms tab, and from there you can upload the form to TransForm. The form's Submit button also returns you to the Existing Forms tab, but first it prompts you to upload the form to TransForm.

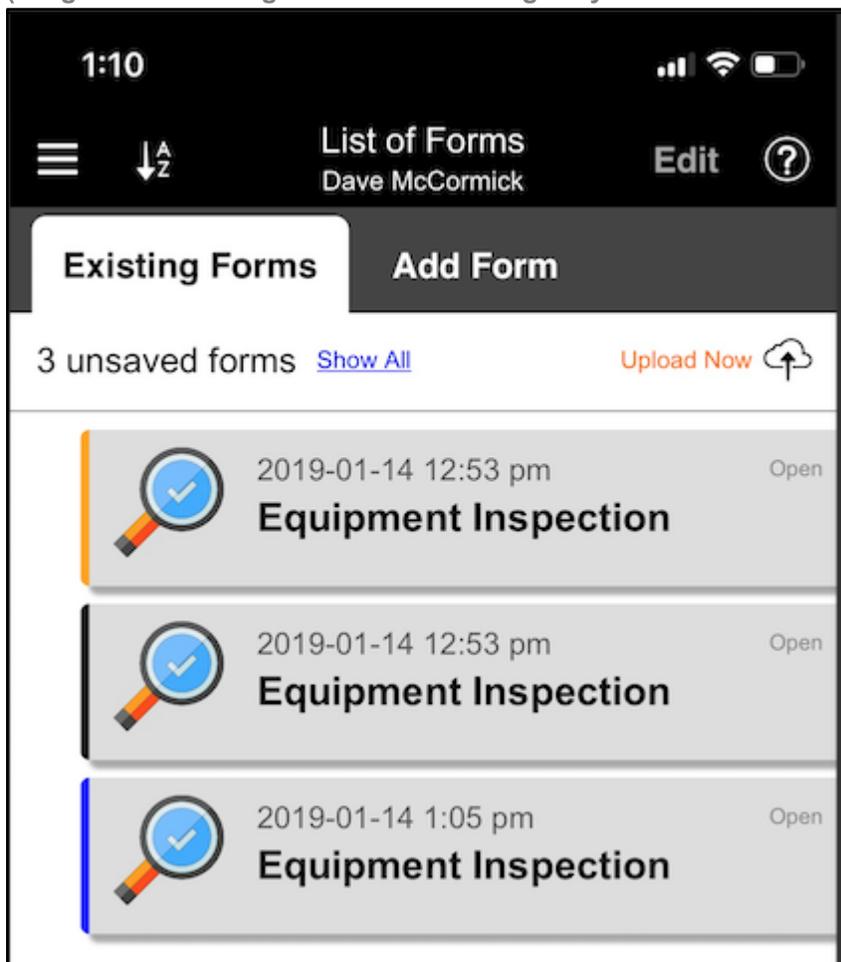
(Image 4.27 - Existing Forms Tab Indicating Forms Need to Be Synchronized)



Near the top of the screen is text indicating the number of forms that have not yet been uploaded to TransForm. In this case there are 3 unsaved forms. If you have many forms on your device, it may be easier to see a list of just the ones that need to be uploaded. You can do that by tapping the Show link next to the text indicating the number of unsaved forms.

You can return to showing all of the forms by tapping the Show All link.

(Image 4.28 - Existing Forms Tab Showing Only Forms That Need to Be Uploaded)



On the left of each of the forms is a thin vertical line. These lines are color coded to indicate whether the record is new (blue line), the record is not new but has been modified (orange line), or the record has been deleted (black line).

To upload the forms to TransForm, tap the Upload Now link.

(Image 4.29 Upload Now Link)



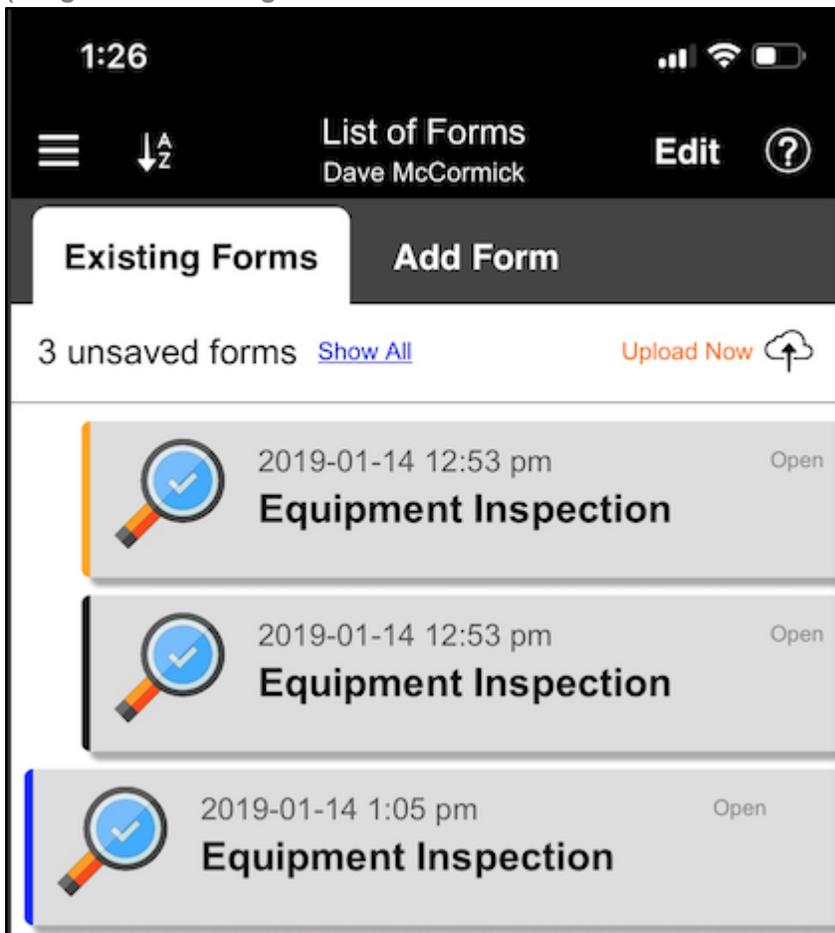
The TransForm app uploads the new data.

Deleting a Form

There are a few different ways to delete a form. However, in some cases you will not be allowed to delete a form, because of the way the form is set up. For example, if a form is marked as submitted you typically are not allowed to delete it. This is up to the person who designed the form.

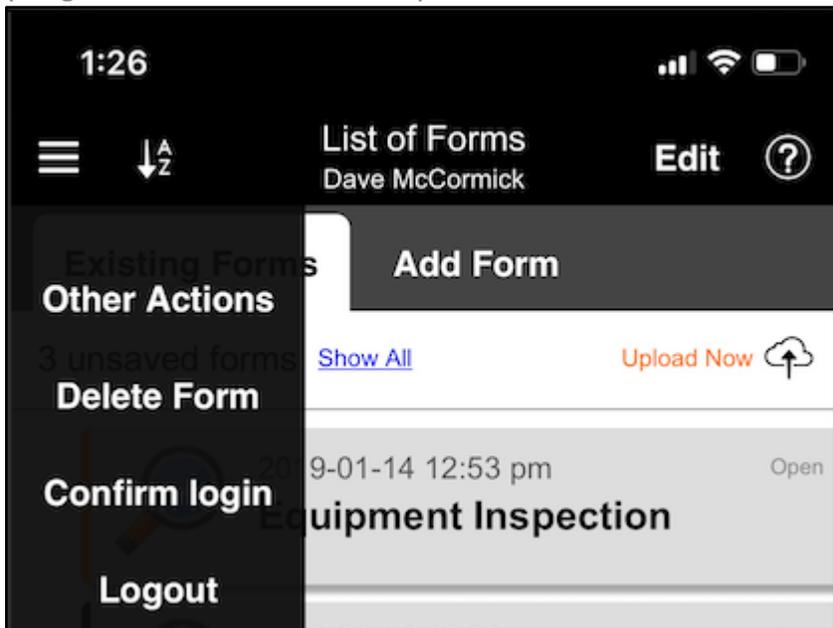
You can delete the last form that you used from the Existing Forms tab. The form you last used is outdented in the list.

(Image 4.30 - Existing Forms Tab - Form at the Bottom is Outdented)



To delete the outdented form, tap the hamburger button  to open the menu.

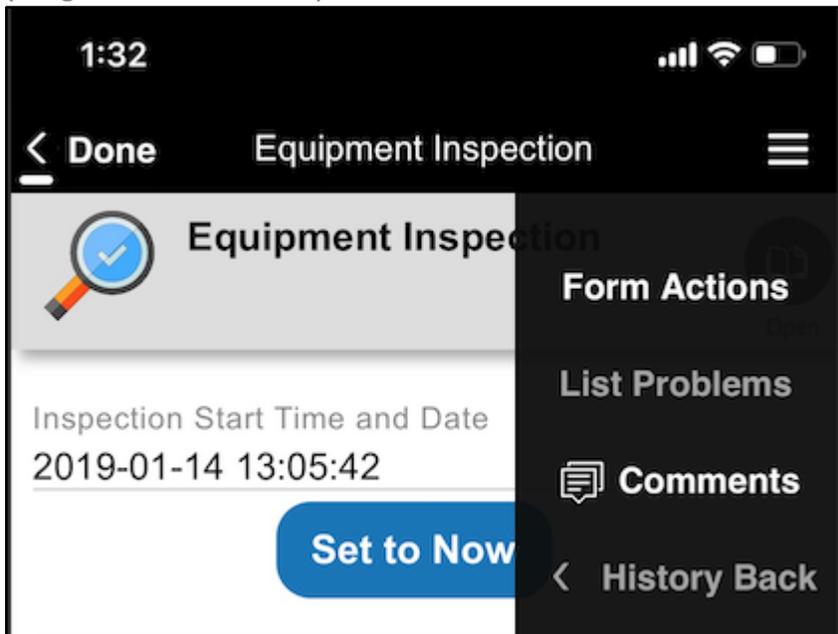
(Image 4.31 - List of Forms Menu)



Choose Delete Form from the list to delete it. After you confirm your choice, the form will be marked with a thin, black vertical line on its left edge.

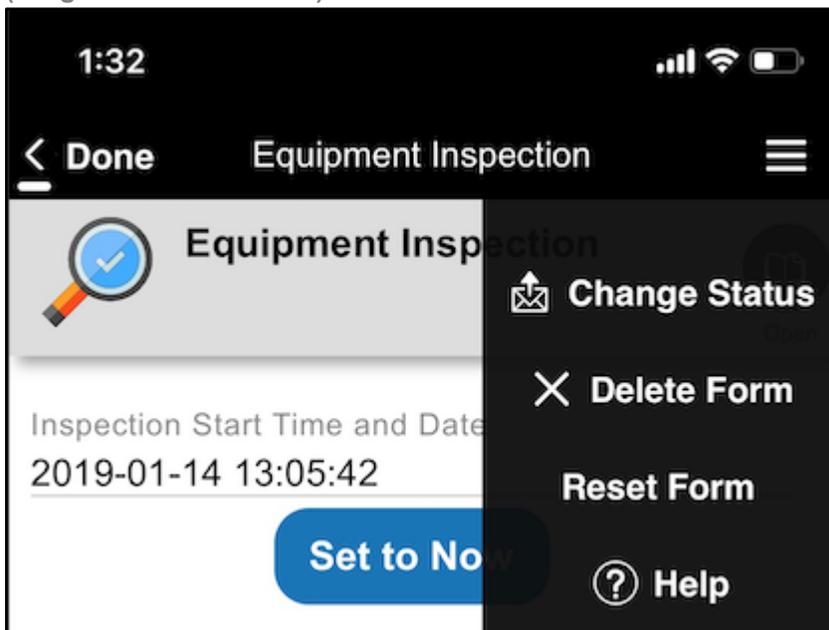
Alternatively, you can delete a form from within the form itself. From the Existing Forms tab, tap on the form you want to delete to open it. Then tap on the hamburger button  to open the menu.

(Image 4.32 - Form Menu)



From the menu, choose Form Actions. You are presented with a list of form actions.

(Image 4.33 - Form Menu)



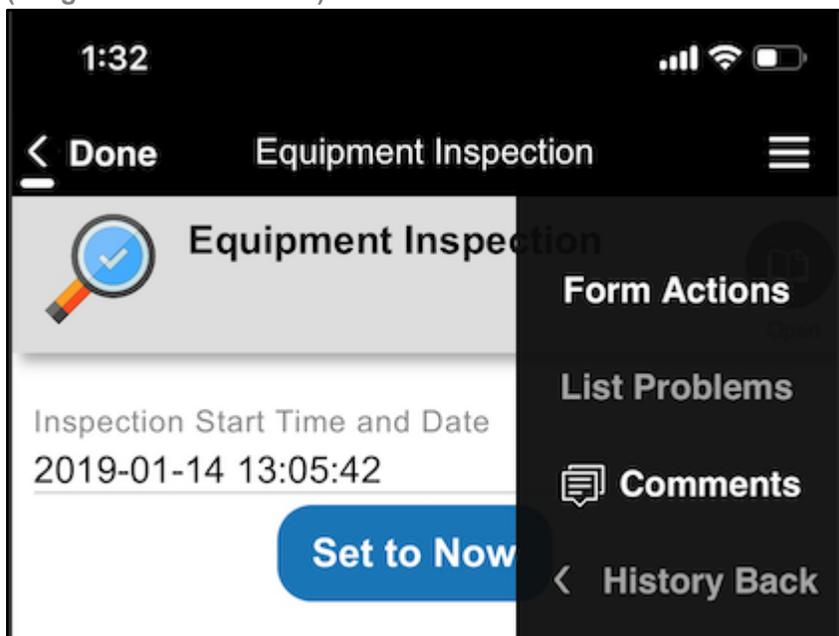
Choose Delete Form from the list.

Resetting a Form

When you are editing an existing form that has already been saved to TransForm, you may find that you have made a mistake and either entered wrong information into the correct form or correct information into the wrong form. In these cases, it is possible to undo what you have done and revert back to the form's original data. This is called Resetting the Form.

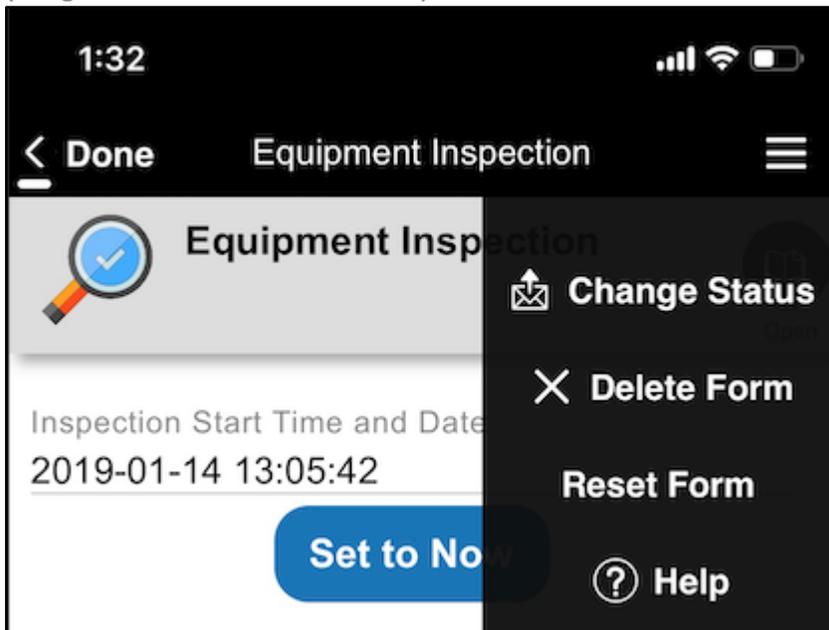
To reset a form, tap on the form's hamburger button  to open the menu.

(Image 4.34 - Form Menu)



From the menu, choose Form Actions. You are presented with a list of form actions.

(Image 4.35 - Form Actions Menu)



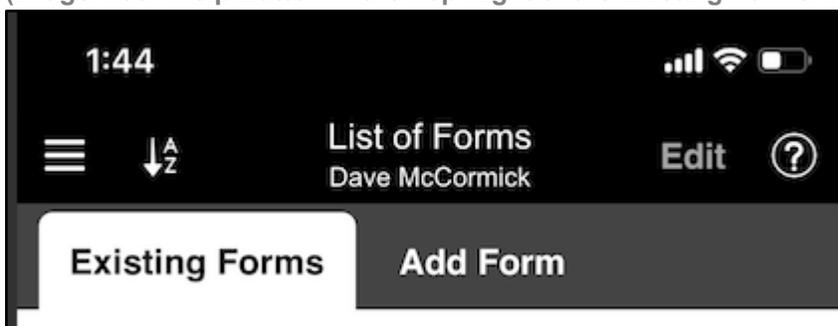
Choose Reset Form from the list.

Getting Help

The Alpha TransForm app has extensive documentation built into its help system. The help system is stored on the device and can be accessed even when there is no internet connection. You can get to the help system from either the Existing Forms or Add Form tab by tapping the ? button.

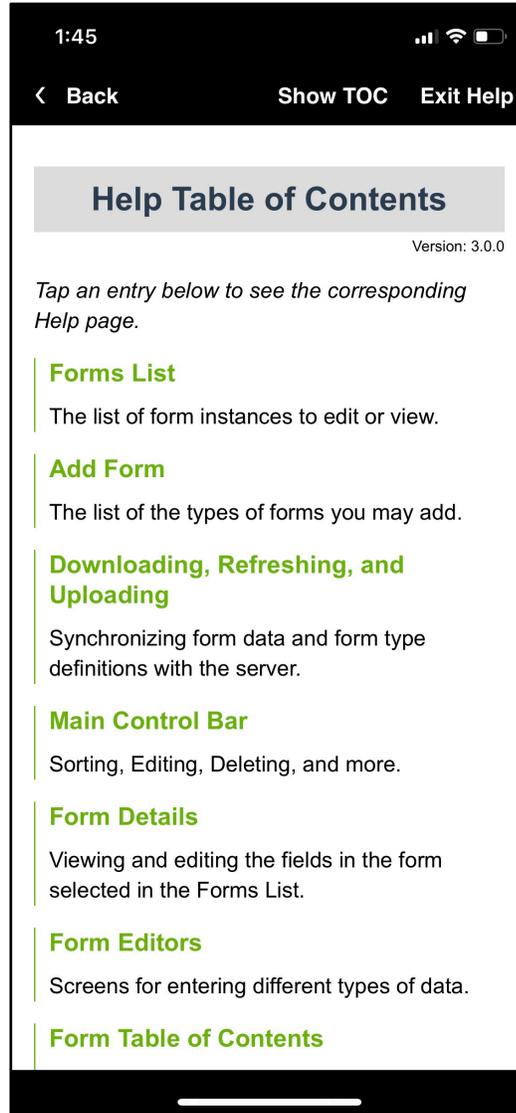
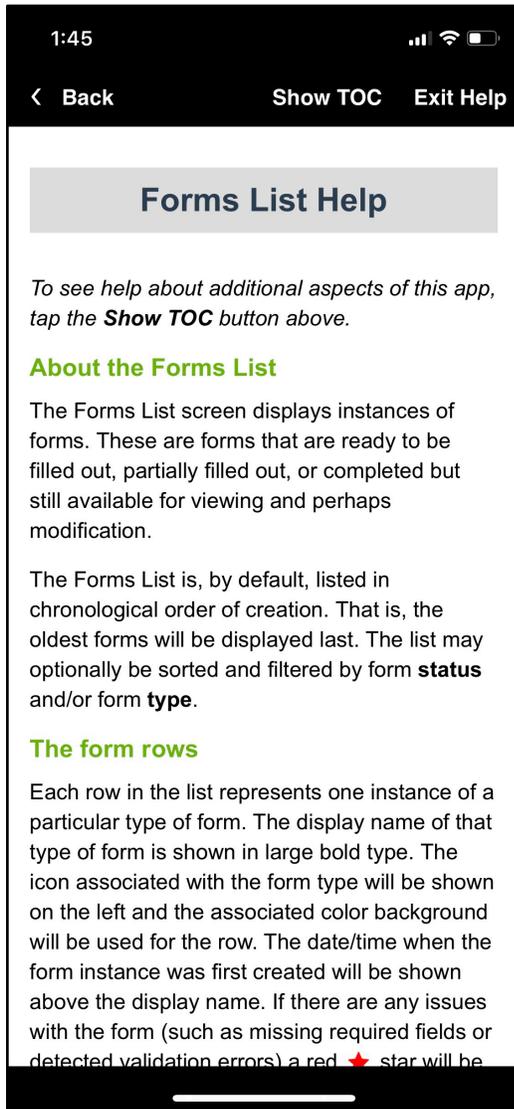


(Image 4.36 - Help Button in the Top Right of the Existing Forms Tab)



When the help system opens, you can get to the help system's table of contents by tapping the Show Toc link at the top.

(Images 4.37 - The Help System and The Help System Table of Contents)



Chapter 5: Designing Forms Part 2 - Form Commands

Back in Chapter 3, you saw the basics of putting together a TransForm app (which we call a form type). In this chapter and future chapters, you will build on what you learned to make your form types more intelligent using conditional logic, error checking and more. You'll also see how to share your form type with other TransForm users who may want to build on one of your designs or examine it to see how it works.

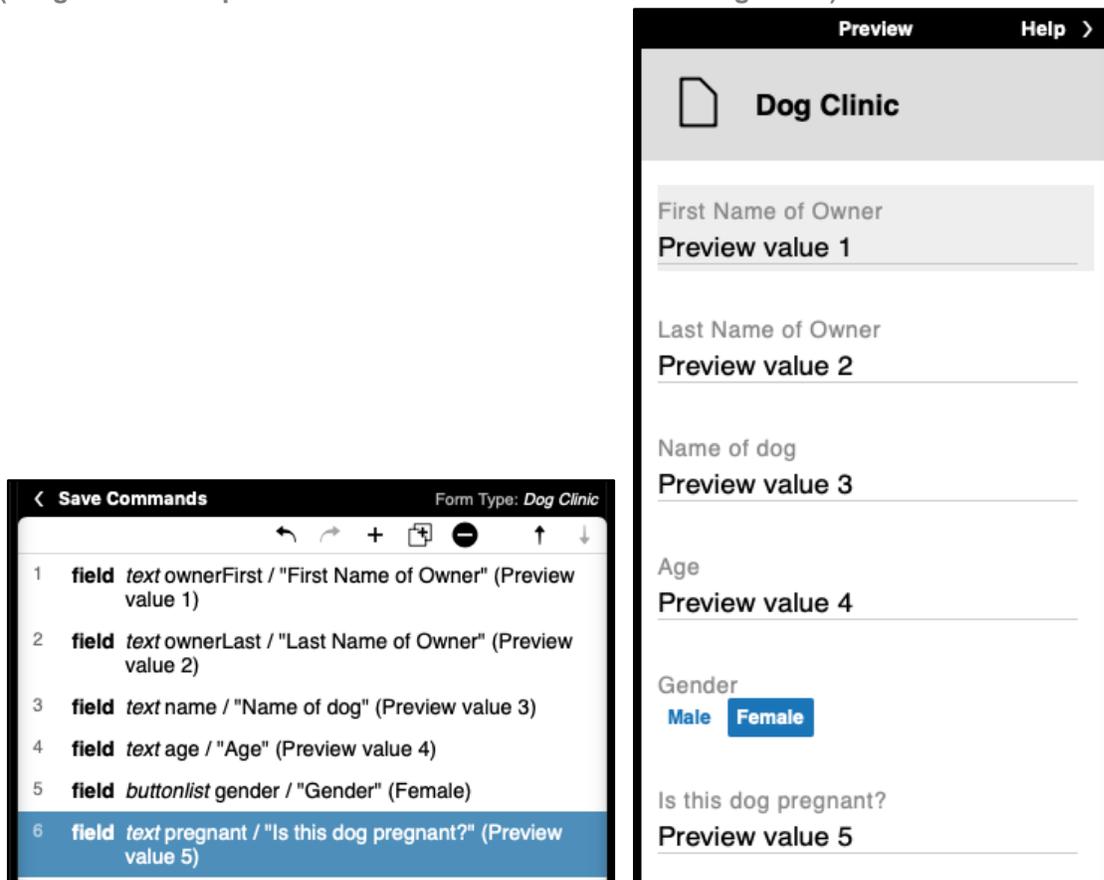
If and Else - Conditional Blocks

One powerful feature of TransForm is its ability to perform actions based on data that have been entered or captured. A common use for this feature is to ask follow-up questions based on the answer to a previous question. This lets you hide fields unless they are needed, making your forms easier to navigate and more efficient to fill out.

For example, suppose you were designing a form type to collect medical information about dogs. Along with the weight and age of the dog, you might also ask the dog's gender. If the dog is female, follow up questions can appear asking, for example, if the dog is pregnant.

To create a form like that, you first create a form with all of the fields, like the following example.

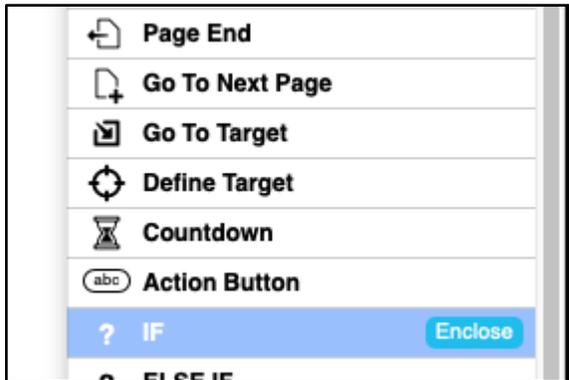
(Image 5.1 - Example Form Definition and Preview for a Dog Clinic)



In the previous example, the question about whether the dog is pregnant is always asked. However, you can change this so that the question is only asked if the dog is female. To do this, you first hold down the SHIFT key and click on the fields that you want to hide. In this case, there is just one field, the last one in the list.

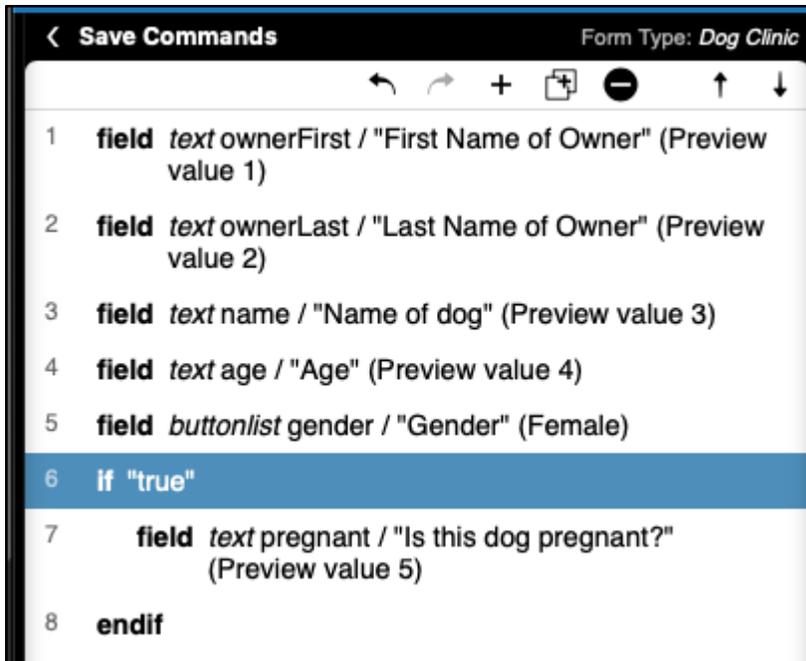
Next, click the + button in the toolbar to open the list of form commands and scroll down to the list until you come to the IF command.

(Image 5.2 - Commands List)



On the line for the IF command, click the Enclose button. Transform inserts an IF command above the field you selected and an ENDIF command below the field you selected (enclosing it). It then highlights the IF command.

(Image 5.3 - An IF and ENDIF Commands Enclose a Field)



While the IF statement is highlighted, the Command Properties pane changes to let you set up the IF command.

(Image 5.4 - Command Properties for IF Command)

Command Properties

Command Type

Test

Insert Field Name

This must be a TPL expression. In TPL, a non-blank value is true and a blank value ("") is false. Comparisons like "==" (equals) result in "1" for true and "" for false.

Examples of TPL tests:

```
#field1==#field2
$#status!="open"
#switch=="Yes" && ##field2>14
```

See the Help for more information about TPL expressions.

Test in Preview by using the buttons below to set True or False. Note: If Default, Preview will use Preview Data for the whole form, not the field's individual Preview Values. If no Preview Data for whole form, then Default will preview as "true".

Preview Value True False Default

Comment

You set the expression of the IF command by typing it into the Test box. When you first create the IF command, the Test box is set to true. Which means that anything enclosed in the IF block will always be displayed (unless it is enclosed in an ELSE block, but we'll cover that later). So in this example, the question about whether or not the dog is pregnant is still always asked.

To change the Test box, first delete its default value of True. Next select the field you want to examine by clicking the Insert Field Name button. A list of field slides in.

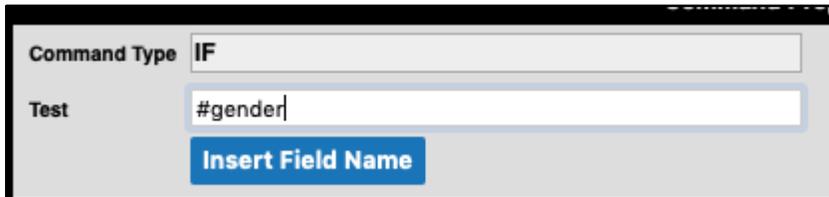
(Image 5.5 - List of Fields For Configuring an IF Command)

Cancel

- First Name of Owner** (#ownerFirst)
- Last Name of Owner** (#ownerLast)
- Name of dog** (#name)
- Age** (#age)
- Gender** (#gender)
- Is this dog pregnant?** (#pregnant)

To select a field, click on it. In this example, Gender is the field we want to use. TransForm inserts the field into the Test box.

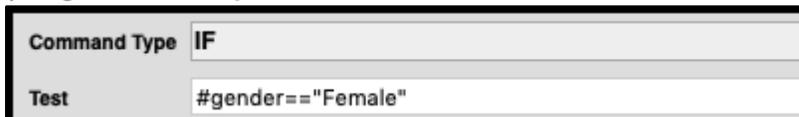
(Image 5.6 - A Field Inserted Into the Test Box of an IF Command)



Command Type	IF
Test	#gender
<input type="button" value="Insert Field Name"/>	

Next, type in the rest of the expression. In this case we want to show the follow up question about whether the dog is pregnant only when gender is female. So the full Test expression would read `#gender=="Female"`.

(Image 5.7 - An Expression Inserted Into the Test Box of an IF Command)



Command Type	IF
Test	#gender=="Female"

Once the expression has been entered, the form now only displays the pregnancy question if the user has first marked the Gender field as female.

Syntax for the IF Command

TransForm uses TPL syntax (TransForm Programming Language) to define the Test expression. TPL is similar (but more secure) than JavaScript. So if you are familiar with JavaScript, you'll probably recognize the `"=="` syntax in the previous example.

Here are some other examples of test expressions.

#Field1 !=17	Field1 is not equal to 17
#Field1 > #Field2	Field1 is greater than Field2
#Field1 > 17 && #Field2 > 17	Field1 is greater than 17 AND Field2 is greater than 17.
#Field1 > 17 #Field2 > 17	Field1 is greater than 17 OR Field2 is greater than 17.

TPL expressions can be very powerful and can reference system fields, perform calculations and more. For more details on TPL expressions, see the online help..

ELSE IF and ELSE

If you have taken a course in computer programming (of any kind), you are probably already familiar with the idea behind ELSE IF and ELSE. Basically, whenever you have an IF, you have the option of adding another IF inside it, in case you want to see if a second condition is met instead.

For example, suppose you are collecting information about animals for an animal clinic, you could ask questions based on the type of animal the user entered. In the example below, the user is asked for an animal species. If they choose Bird, another field appears asking what type of bird.

(Image 5.8 - Example of IF, ELSEIF, ELSE, and ENDIF)

```
1 field text name / "Name of Pet" (Preview value 1)
2 field buttonlist petType / "Type of Pet" (Bird)
3 if "#petType=="Bird""
4     field buttonlist birdType / "Type of Bird" (Parrot)
5 elseif "#petType=="Mammal""
6     field buttonlist mammalType / "Type of Mammal"
7     (Dog)
8 elseif "#petType=="Reptile""
9     field buttonlist field5 / "Type of Reptile" (Snake)
10 else
11     field buttonlist fishType / "Type of Fish" (value1)
12 endif
```

If they do not answer bird, TransForm looks at the first ELSE IF command. If they selected Mammal, a question appears asking what type of mammal. If it is not a mammal, TransForm goes to the next ELSE IF command to see if it is a reptile. If it is not a reptile, TransForm looks for the next ELSE IF command, but there isn't one. So instead it looks to see if an ELSE command exists. If there is an ELSE command, TransForm displays what is enclosed in the ELSE command.

END IF

The END IF command is required at the end of an IF, after all of the ELSE IF commands (if present) and the ELSE command (if present). This lets TransForm know that the IF command and its associated commands are complete.

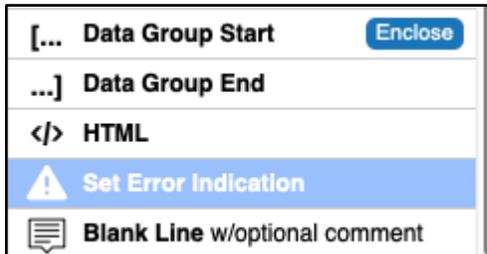
Error Indications

The Error Indication command allows the form to alert users when they enter data that appears to be invalid. An Error Indication is used in conjunction with the IF command. The IF command determines if there is an error, and the Error Indication command displays the error to the user and sets an error indicator in the TransForm app.

For information on how errors are displayed and managed in the TransForm App, refer back to Chapter 3.

To add an Error Indication, click the plus button in the toolbar to open the list of form commands and scroll down to almost the very bottom of the list until you come to the Set Error Indication command.

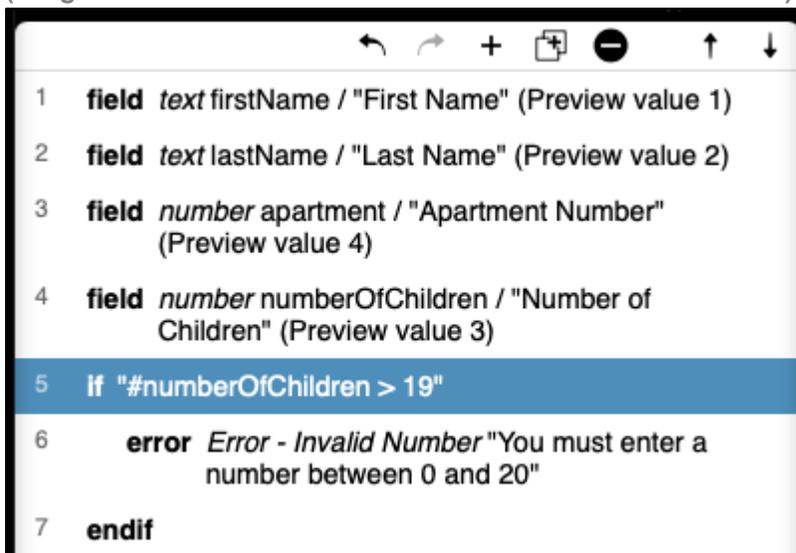
(Image 5.9 - Error Indication Command)



Error Checking Example

For example, suppose you have an application for collecting information about the occupants of an apartment building. For each apartment, you would like to know the number of children living in the apartment. To prevent data entry errors, you could set up an IF command and an Error Indication command to make sure that the number that was entered is 19 or fewer.

(Image 5.10 - IF Command with Set Error Indication Command)

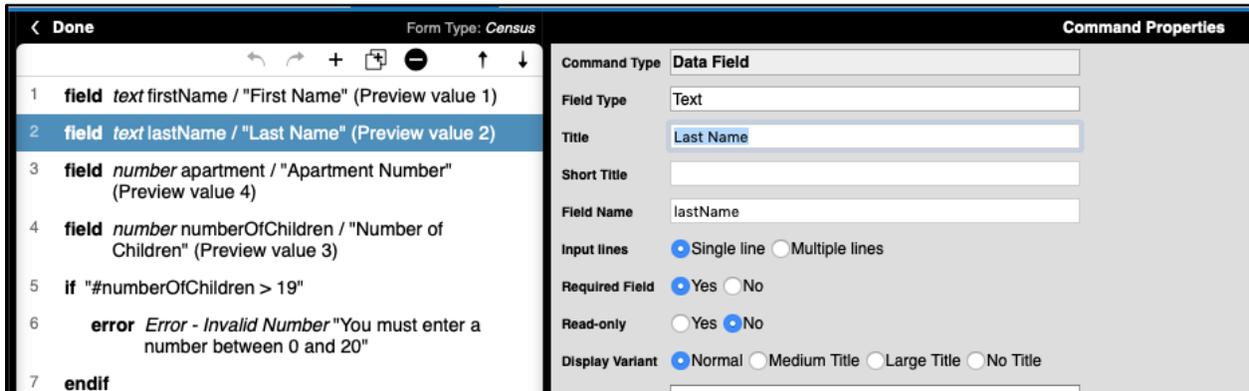


If the Number of Children field is greater than 19, the Error Indication appears.

Checking for Blank Required Fields

If you want to make one or more fields on your form required, you could use an IF command and an Error Indicator command, but there is an easier way. Click on the field in the Form Commands view. Then in the Command Properties pane, set the Required Field button to Yes.

(Image 5.11 - The Required Field Button in the Command Properties Pane)



The field is marked with a red star in the TransForm app to indicate it is required. If the field is left blank, an error appears in the list of errors for the form (as described in Chapter 4).

Go To

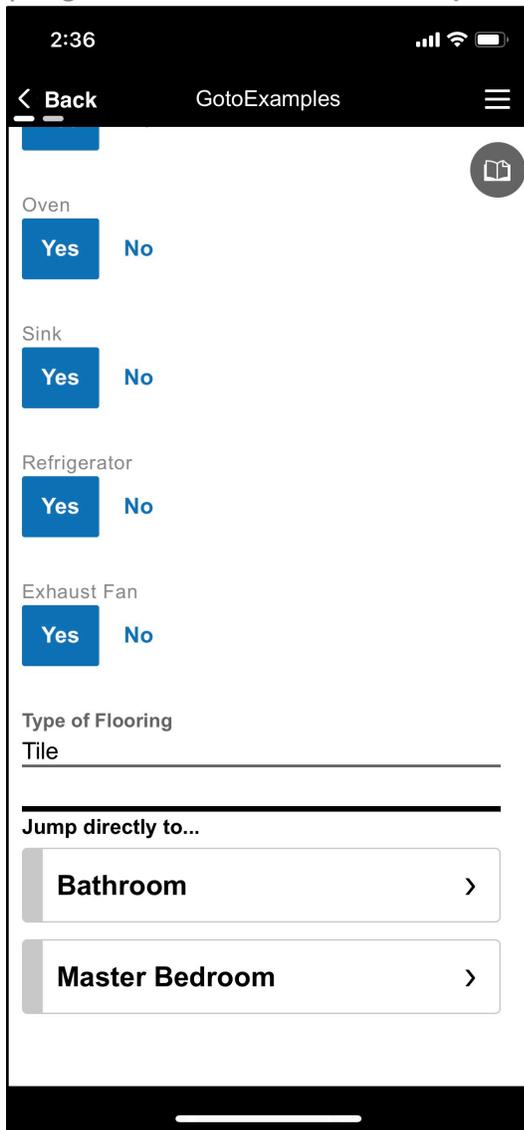
Chapter 3 discussed breaking large forms into pages, so that related fields can be grouped together. When you create a page, a button is automatically created. This is a type of "Go To" button, because when you click the button, you go to that page.

(Image 5.12 - Example of Form Buttons to Go To Different Pages)

The image shows a mobile application interface with a dark header. The time is 2:34, and there are icons for signal strength, Wi-Fi, and battery. The header contains a back arrow, the text 'Done', the title 'GotoExamples', and a menu icon. The form has four main sections: 1. 'Name of Owner' with the text 'Jean Thomas' and a circular icon with a document symbol. 2. 'Address' with the text '18 Mainstreet'. 3. 'Year Built' with the text '2017'. 4. 'Picture of House' with a photograph of a two-story house. Below the photo are three buttons: 'Kitchen', 'Bathroom', and 'Master Bedroom', each with a right-pointing chevron.

When you create a page, this Go To button is created for you automatically. However, you can create your own Go To buttons to make navigating your form type easier, like in the example below.

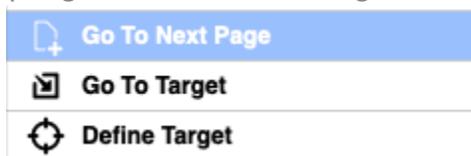
(Image 5.13 - Go To Buttons to Jump Directly to Parts of the Form)



Go To buttons can be found by clicking the + button in the Form Commands toolbar.

There are two types, Go to Next Page and Go To Target.

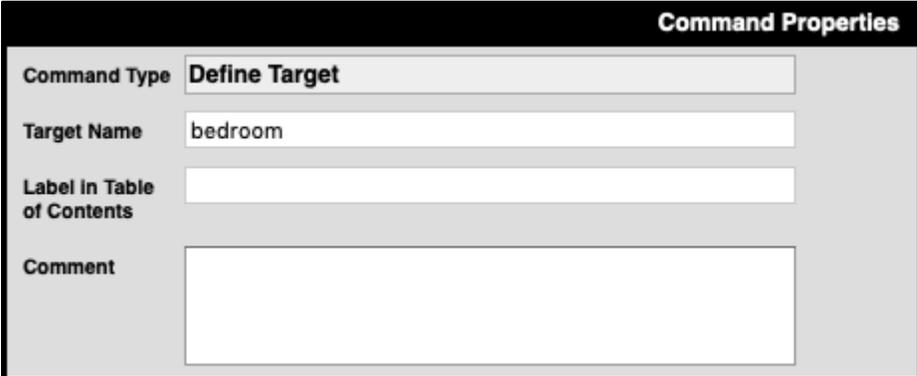
(Image 5.14 - Go To and Target Commands in the New Command menu)



Go To Next Page, as you may expect, creates a button that takes the user to the next page. While, Go To Target buttons can take the user to any page, section, or field in your form. The Go To Target command works in conjunction with the Define Target command.

To create a Go To button that goes to a target, in the Form Commands List click on the page, section, or field that you would like to target. Then click the + button in the Toolbar and choose the Define Target command. In the Command Properties pane, type in a name for the target. You will use this name when you set up the Go To button.

(Image 5.15 - Command Properties for a Define Target Command)



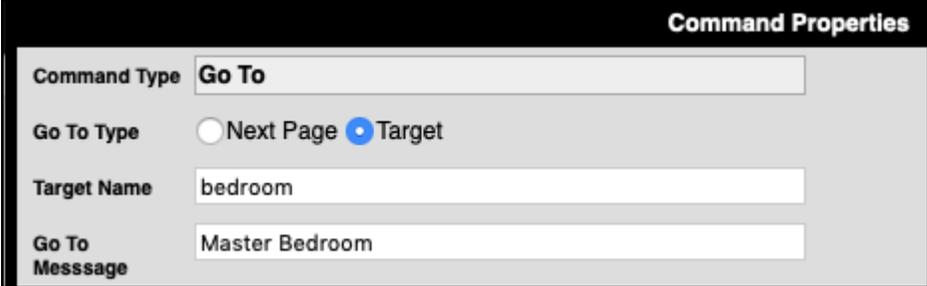
The screenshot shows a 'Command Properties' dialog box with the following fields:

- Command Type: Define Target
- Target Name: bedroom
- Label in Table of Contents: (empty)
- Comment: (empty)

You can also add a label that will appear in the form's table of contents.

Next, click the + button in the toolbar again and select Go To Target from the commands list. In the Command Properties pane, type in the name of the target you just created.

(Image 5.16 Command Properties for a Go To Target Command)



The screenshot shows a 'Command Properties' dialog box with the following fields:

- Command Type: Go To
- Go To Type: Next Page Target
- Target Name: bedroom
- Go To Message: Master Bedroom

In the Go To Message box, type in the text you would like to appear on the Go To button.

Submit Buttons (Change Status Command)

When a user is done filling out a form, they can tap the Done button at the top of the page to exit the form and then tap the Upload button to upload the form's data. You can make this process easier for them by adding a Submit button that both exits the form and uploads the data. The button can also change the status of the form from Open to Submit (or some other status). You can even set up the button so that form will disappear from the device the next time the application is synchronized.

Submit buttons can be added using either the Quickstart view or the Form Command view.

In Quickstart View, there are two Submit commands at the bottom of the list.

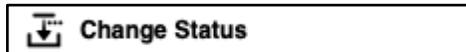
(Image 5.17 Submit Commands in Quickstart Text)



The first option creates a button that exits the form and marks the forms status as submitted. The second option also exits the form, but allows you to decide what the status should be set to.

In Form Command View, there is no Submit button option in the Command menu. Instead you create submit buttons using the Change Status command.

(Image 5.18 Change Status Command in the Form Commands List)



When you add a Change Status command, you are given several options for how you would like it to operate.

(Image 5.19 - Command Properties for Change Status Command)

Command Properties

Command Type **Change Status**

Status Change Message

Set Status To

- Open
- Submitted**
- Approval
- Closed

Upload-Only Message

Change form instance so that it will not download to Filler Yes No

The Set Status To list lets you choose what form status should be assigned when the button is tapped. The form status can be used to control whether a record can be edited and for other reasons. See Chapter 9 for Details.

Changing the Text on a Submit Button

There are two boxes that control the text of the button that appears on the form. The Status Change Message is usually the text that appears on the button. By default this text reads “Close and Upload,” but you can change it something else.

In some cases, however, you may not wish to change the status. For example, the record may have already been marked as submitted and you have gone back to make edits. In that case the button only uploads the form data without changing the status. In this case, the Upload Only Message appears instead. By default this text reads “Upload Only,” but you can also change this to something else.

Automatically Removing the Form After Upload

In some cases, once a form is uploaded there is no longer a need for it to be on the mobile device. Removing these forms clears up space on the Existing Forms tab and frees up storage space on the device. To automatically remove a form after the user

taps the upload button, set the Change Form Instance So That It Will Not Download To Filler button to Yes.

Sharing (Importing/Exporting) a Form Type

There may be times when you want to take a form design and send it to someone else via email. Perhaps you want to share your form design with a colleague or associate, but you do not have access to their account as a designer. Or perhaps, you would simply like a backup copy of the form for safe keeping.

With TransForm, you can save, share, and import form types. The TransForm system stores form types as JSON text, which makes them easy to save and email.

Exporting a Form Type

To export a form type, in the Designer tab, click on the form type you want to export. In the Form Properties pane, click the Advanced Features link to display more options.

(Image 5.20 - Edit Entire Definition as Raw JSON Link)

Properties

▼ **Advanced Features**

*For information about these advanced features, click the **Help** button above the Preview Screen to the right of this Properties Screen and go to the Form Properties Screen section.*

Set Color and Icon

Enable custom code for this form type
 Yes No

Edit Custom Code

Status/Role Permissions Group

Initial Status

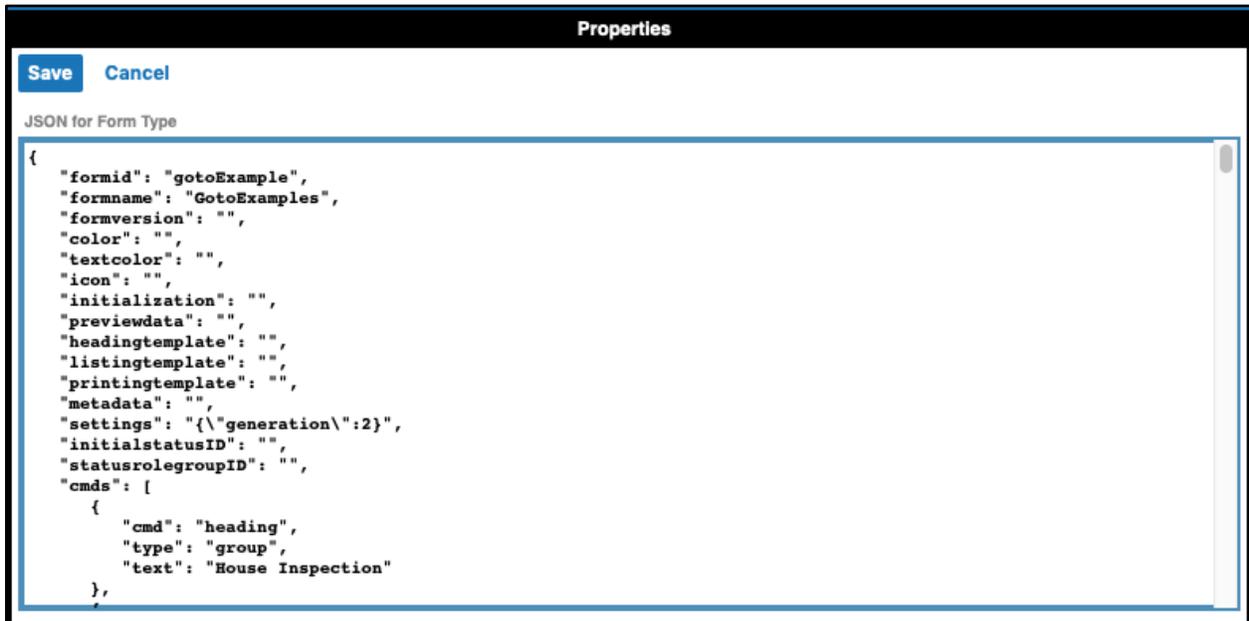
Allow status change only through buttons
 Yes No

[Edit Commands As Raw JSON](#)

[Edit Entire Definition As Raw JSON](#)

Click the Edit Entire Definition as Raw JSON link. A text box slides in showing the JSON text for the form.

(Image 5.21 - Raw JSON for a Form Type)

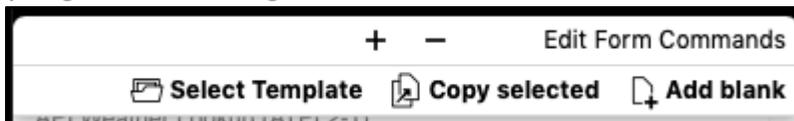


Select all of the text in the box and copy it to the clipboard. The text in the clipboard can now be pasted into a text document (Notepad, Google Docs, Word, etc.) or it can be pasted and sent in an email.

Importing a Form Type

Importing a previously exported form type is a two step process. First you create a new, empty form in the Designer tab by clicking the + button on the Toolbar and choosing Add Blank.

(Image 5.22 - Creating A Blank Form to Paste the JSON Definition)



The second step is to paste the JSON definition into the new form. In the Form Properties pane, click the Advanced Features link to display more options (if necessary).

(Image 5.23 - Edit Entire Definition as Raw JSON Link)

The screenshot shows the 'Properties' dialog box with the 'Advanced Features' section expanded. It contains a 'Set Color and Icon' button, a checkbox for 'Enable custom code for this form type' (set to 'No'), an 'Edit Custom Code' button, two text input fields for 'Status/Role Permissions Group' and 'Initial Status', another checkbox for 'Allow status change only through buttons' (set to 'No'), and two links: 'Edit Commands As Raw JSON' and 'Edit Entire Definition As Raw JSON'.

Click the Edit Entire Definition as Raw JSON link. A text box slides in showing the JSON text for the form.

(Image 5.24 Raw JSON for a New Form Type)

The screenshot shows a dialog box titled 'JSON for Form Type' with 'Save' and 'Cancel' buttons. It contains a text area with the following JSON code:

```
{
  "formid": "Form1",
  "formname": "New Form1",
  "formversion": "",
  "color": "",
  "textcolor": "",
  "icon": "",
  "initialization": "",
  "previewdata": "",
  "headingtemplate": "",
  "listingtemplate": "",
  "printingtemplate": "",
  "metadata": "",
  "settings": "",
  "initialstatusID": "",
  "statusrolegroupID": "",
  "cmds": [
    {}
  ]
}
```

Select all of the text in this box and overwrite it by pasting the new JSON text. Click the Save button to save changes to your form. You are returned to the Form Properties

screen. At this point, you may want to change the Display Name and Form ID, since these were also imported with the JSON definition.

Importing and Exporting Only the Commands

The last two sections showed how to import and export a form type's entire definition. A form's definition includes not only the commands you added to the form, but also its name, heading template, listing template, colors, icons, and more.

In some cases, this may be more than you want, and instead you would prefer to just export and import the commands in the form without all of the other information and settings. You can do this by using the Edit Commands as Raw JSON option instead of the Edit Entire Definition as raw JSON.

Chapter 6: Managing Data

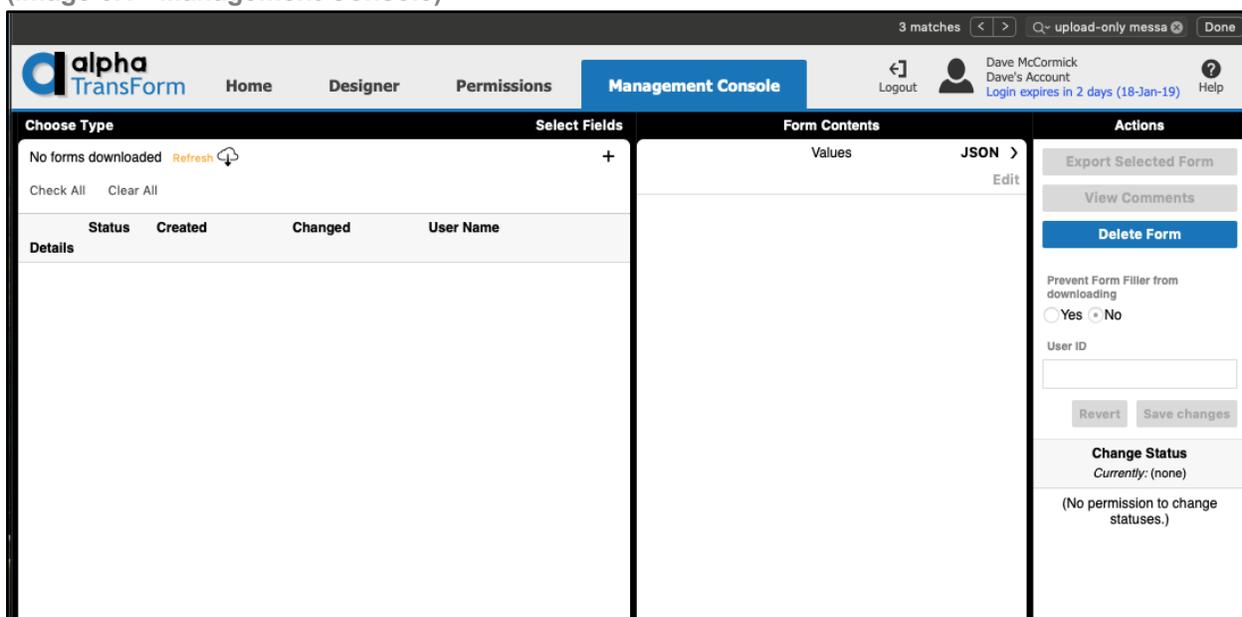
Up until now, this book has focused on creating form types and filling in forms. In this chapter, you will see how to view, edit, and export the data you have collected. You'll also see how to create partially (or fully) filled in forms that you can assign to users in the field.

You do all of this in the Management Console tab of the TransForm Central website.

Management Console

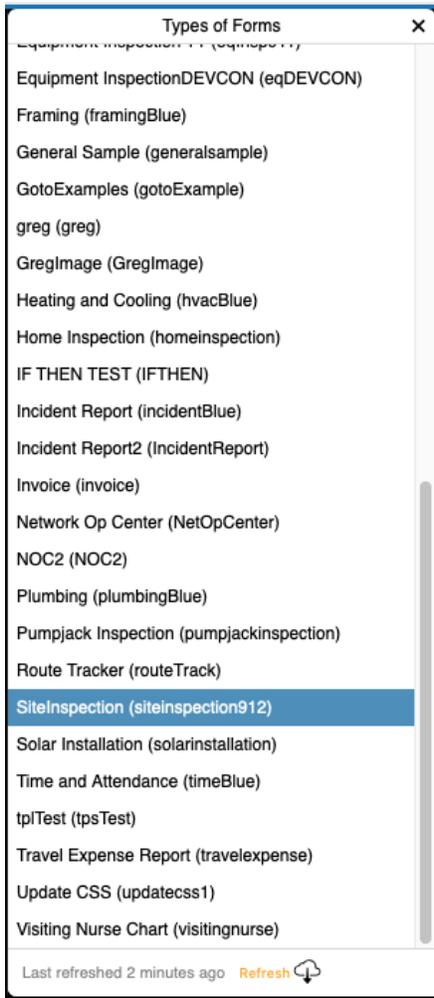
Unlike the Designer tab, where you create form types, the Management Console tab is where you work with the forms themselves. The tab is divided into three panes. The pane on the left allows you to choose a form type, and then displays a list of all of the forms of that type.

(Image 6.1 - Management Console)



To choose a form type, click the Choose Type link. A Types of Forms list appears.

(Image 6.2 - Types of Forms List)



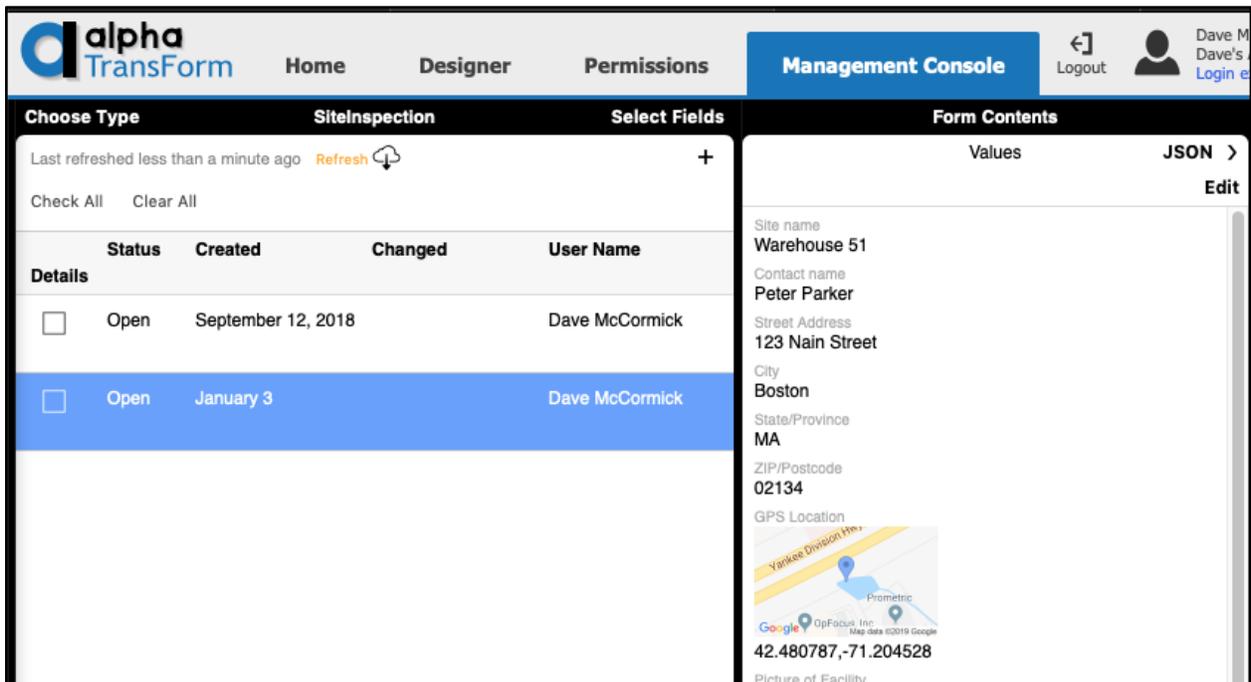
Click the Refresh button at the bottom to get the most recent form types, then click on the form type you want. The form type list closes, and all of the forms of that type are loaded into the list on the left.

(Image 6.3 - List of Two SiteInspection Forms in the Management Console)

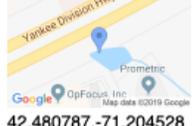
Choose Type	SiteInspection	Select Fields		
Last refreshed less than a minute ago Refresh 				
Check All Clear All				
Details	Status	Created	Changed	User Name
<input type="checkbox"/>	Open	September 12, 2018		Dave McCormick
<input type="checkbox"/>	Open	January 3		Dave McCormick

When you click on a form in this list, the data in the form is displayed in the Form Contents pane. (Note you can select a row by clicking any part of an entry on the list, except the checkbox. Clicking the checkbox does not select the row.)

(6.4 - The Form Contents Pane Displays the Form's Data)



The screenshot shows the alpha TransForm Management Console interface. The top navigation bar includes 'Home', 'Designer', 'Permissions', and 'Management Console'. The 'Management Console' pane is active, displaying a list of SiteInspection forms. The second row is selected, showing details for a form created on January 3 by Dave McCormick. The 'Form Contents' pane on the right displays the data for this form, including site name, contact name, address, city, state, and ZIP code. A map shows the GPS location in Boston, MA.

Choose Type	SiteInspection	Select Fields	Form Contents		
Last refreshed less than a minute ago Refresh 			Values JSON 		
Check All Clear All			Edit		
Details	Status	Created	Changed	User Name	Site name Warehouse 51
<input type="checkbox"/>	Open	September 12, 2018		Dave McCormick	Contact name Peter Parker
<input type="checkbox"/>	Open	January 3		Dave McCormick	Street Address 123 Nain Street
					City Boston
					State/Province MA
					ZIP/Postcode 02134
					GPS Location 
					42.480787,-71.204528
					Picture of Facility

Editing Data in a Form

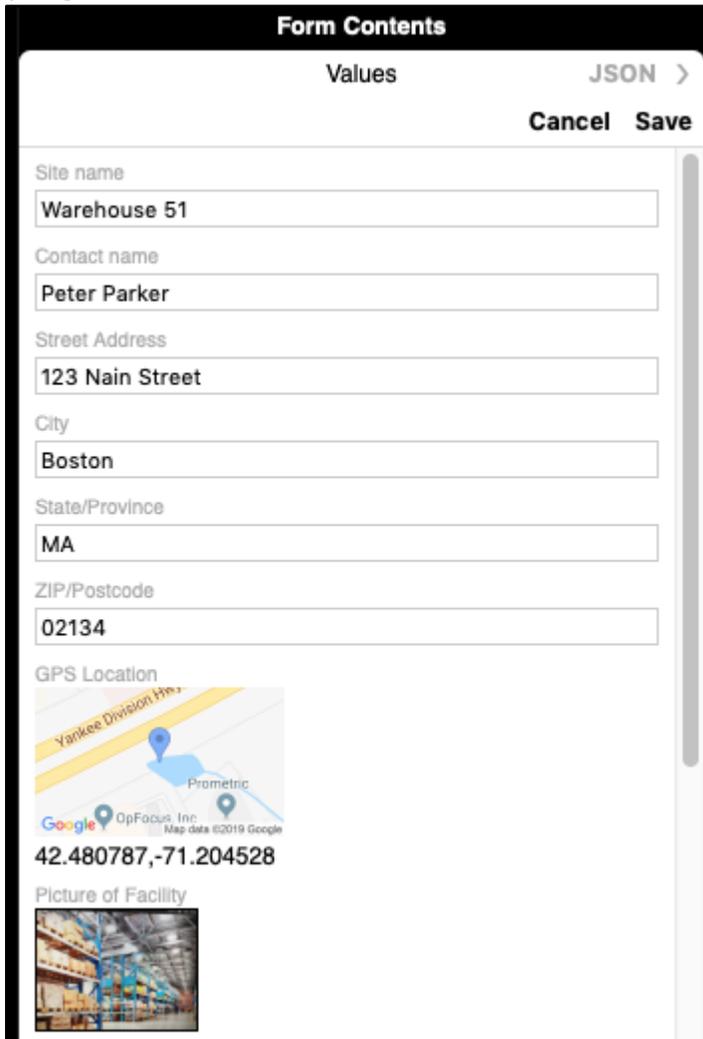
The Form Contents pane displays the data that was entered into the currently selected form. To edit the data in this form, click the Edit link in the toolbar.

(Image 6.5 - Edit Link in Form Contents Pane)



When you click Edit, the contents of the form become editable.

(Image 6.6 - Form Contents Pane With Editable Contents)

A screenshot of the 'Form Contents' pane with the 'Edit' button clicked. The pane is titled 'Form Contents' and has 'Values' and 'JSON >' tabs. Below the tabs are 'Cancel' and 'Save' buttons. The form contains several input fields with the following values: 'Site name' (Warehouse 51), 'Contact name' (Peter Parker), 'Street Address' (123 Nain Street), 'City' (Boston), 'State/Province' (MA), and 'ZIP/Postcode' (02134). Below the text fields is a 'GPS Location' section with a map showing a location in Boston, with the coordinates '42.480787,-71.204528' displayed. At the bottom is a 'Picture of Facility' section with a small image of a warehouse interior.

You can edit the data in all of the text fields, number fields, and lists. However, you cannot enter a GPS location, take a picture, or perform a barcode scan because these field types are currently only supported when the form is on a mobile device.

When you have finished making edits, click the Save button in the toolbar. Or to cancel and changes you have made, click Cancel.

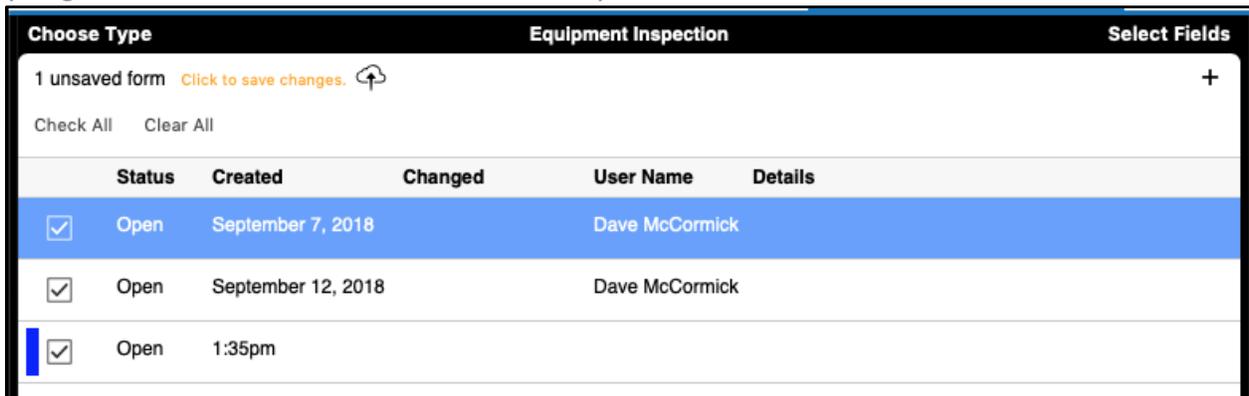
Deleting Forms

TransForm lets you delete the selected record or delete multiple records that you have marked with a checkbox. To delete the selected record, click the Delete Selected Form button in the Actions pane.

Delete Selected Form

Alternatively, you can delete multiple forms at once by clicking the checkboxes next to each of the forms you want to delete. You can use the Check All link to check all of the boxes and the Clear All link to clear all of the boxes.

(Image 6.7 -Form List With 3 Records Checked)



The screenshot shows a mobile application interface for 'Equipment Inspection'. At the top, there are three tabs: 'Choose Type', 'Equipment Inspection', and 'Select Fields'. Below the tabs, it says '1 unsaved form' with a 'Click to save changes.' link and a refresh icon. There are 'Check All' and 'Clear All' links. Below this is a table with three columns: 'Status', 'Created', and 'User Name'. The first two columns are highlighted in blue. The first row is selected, with a blue background. The second row is also selected, with a blue background. The third row is not selected. The 'Details' column is visible on the right side of the table.

Status	Created	Changed	User Name	Details
<input checked="" type="checkbox"/>	Open	September 7, 2018	Dave McCormick	
<input checked="" type="checkbox"/>	Open	September 12, 2018	Dave McCormick	
<input checked="" type="checkbox"/>	Open	1:35pm		

Once you have checked one or more forms, click the Delete Selected Forms button.

Delete Selected Form

Be careful, these forms are permanently deleted.

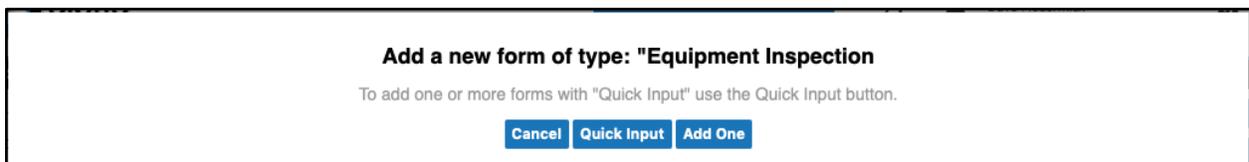
Creating Forms

Chapter 4 showed how to create new forms in the TransForm app. However, it is possible to create and fill in a new form using TransForm Central.

One common example for this is for dispatching workers to a site, where an office worker creates a new form and fills in some of the fields and then assigns the form to another user who is out in the field who will finish filling in the form using a mobile device. We refer to this as a “dispatch” scenario and it is covered in the upcoming section “Assigning Forms to Users.”

To create a form, in the management console tab, click the Choose Type link to select the type of form you would like to create. If your form isn’t listed, you may need to click the Refresh button at the bottom of the list.

After choosing a form, click the + button in the toolbar. A new dialog box appears asking if you want to create just one form or multiple forms all at once.

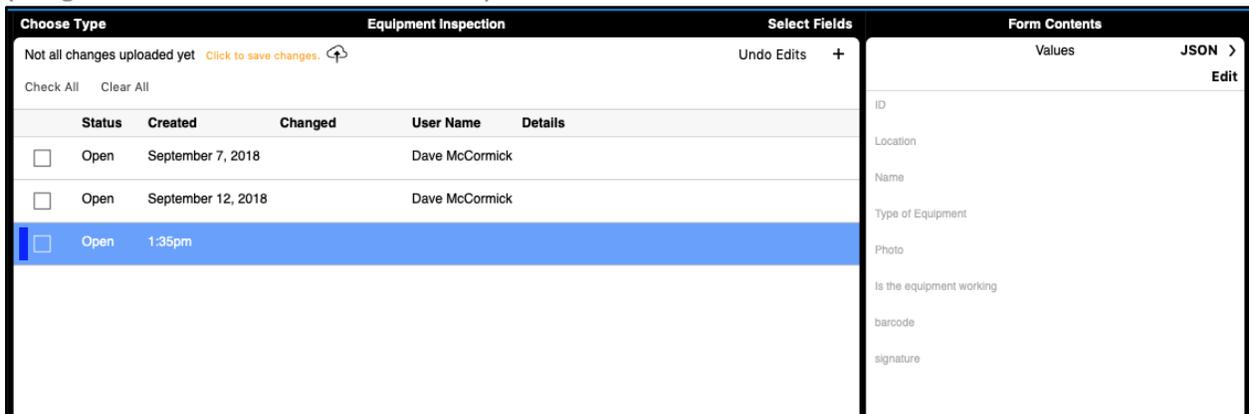


To create just one form, click the Add One button. To create multiple forms at once, click Quick Input. These two methods of creating forms are discussed in the next two sections.

Creating One Form

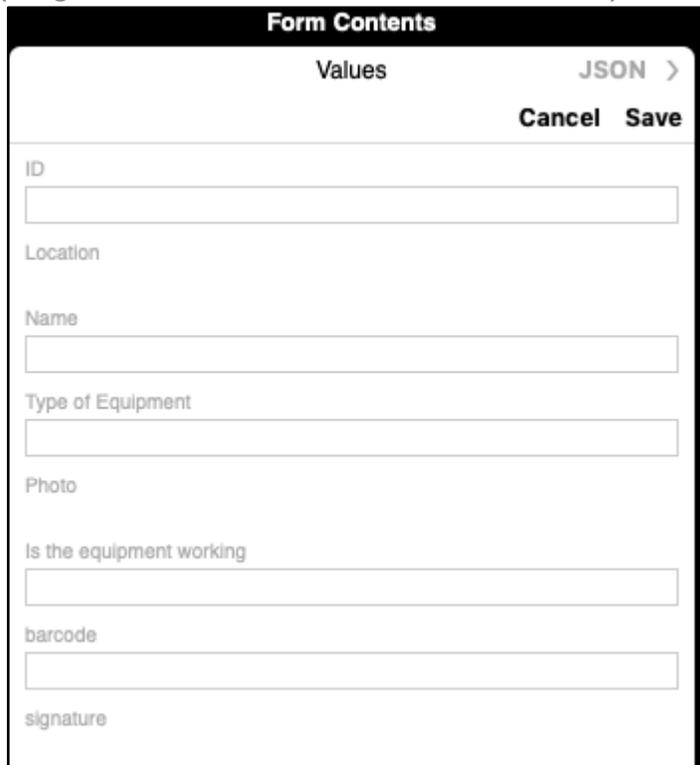
When you click the Add One button, TransForm inserts a new, blank form.

(Image 6.8 - Add One New Blank Form)



The new, blank form is automatically selected in the list, and the Form Contents pane shows you the fields in the form. You can add data to the form by clicking the Edit link in the toolbar.

(Image 6.9 - New Form. Form Contents are Blank)



Fields that you can type data into have boxes under them. Click on the boxes to type in the values. You cannot add a signature or photo, because that is not supported in the

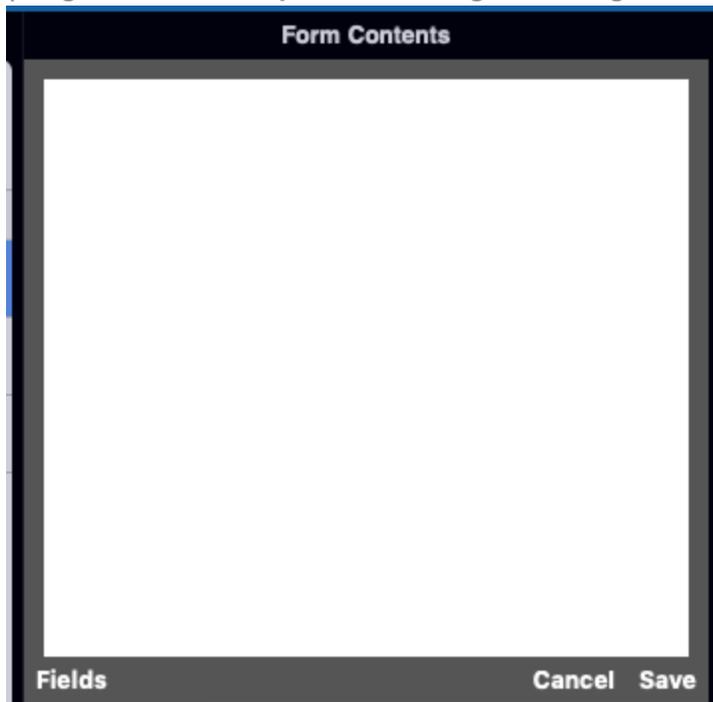
TransForm Central, only in the TransForm app. You cannot scan a barcode either, however you can still type a value into the barcode box.

When you are done, click the Save link to save your changes.

Creating Multiple Forms at Once

TransForm provides a “Quick Input” method of adding and filling in new forms. Quick Input is similar to the Quick Start Text method of creating a form type (see Chapter 3), except that instead of creating form types, you are creating the forms themselves. When you click the Quick Input button in the dialog, a blank box appears.

(Image 6.10 - Quick Input For Creating and Filling In Forms)



In the box, you can enter form data one line at a time using the format.

(fieldname):field value

The fieldname is the name of the field and field value is the text or number you would like to put in that field. After adding your field assignments, one per line, you can add a blank line and add another grouping of field assignments to create a second record.

Remembering the fieldnames, however is cumbersome, so instead of typing them in, click the Fields link to insert all of the fields into the box.

(Image 6.11 - Quick Input Box After Clicking the Fields Link Box)



The image shows a dialog box titled "Form Contents". Inside the dialog, there is a list of four fields: "First Name (firstName):", "Last Name (lastName):", "Apartment Number (apartment):", and "Number of Children (numberOfChildren):". Each field name is followed by a colon. The dialog box has a dark border and a dark header bar with the title "Form Contents". At the bottom of the dialog, there are three buttons: "Fields", "Cancel", and "Save".

To add data to the form, type in the information you would like to add to the end of each line.

(Image 6.12 - Quick Input Box Filled in To Create a New Record)

The screenshot shows a dialog box titled "Form Contents" with a dark header. The main area is white and contains the following text:
First Name (firstName): Frank
Last Name (lastName): Jennings
Apartment Number (apartment): 21
Number of Children (numberOfChildren): 2
At the bottom of the dialog, there is a dark bar with the text "Fields" on the left and "Cancel Save" on the right.

You can now add a second form, by clicking the Fields button again and filling in the information for the second form.

(Image 6.13 - Quick Input Box Filled in To Create Multiple New Records)

The screenshot shows a dialog box titled "Form Contents" with a dark header. The main area is white and contains two sets of text, one above the other:
First Name (firstName): Frank
Last Name (lastName): Jennings
Apartment Number (apartment): 21
Number of Children (numberOfChildren): 2

First Name (firstName): Lisa
Last Name (lastName): Wilton
Apartment Number (apartment): 7
Number of Children (numberOfChildren): 1
At the bottom of the dialog, there is a dark bar with the text "Fields" on the left and "Cancel Save" on the right.

To create the forms with the data you entered, click the Save button.

Exporting Data

Besides viewing and entering forms In the Management Console, you can export your forms to a variety of file formats. There is also an option for sending forms by email. To export forms, first click the Choose Type link on the Management Console tab to select the type of form.

Check the checkboxes of the forms you want to export, then click the Export Checked Forms button in the Actions pane.

**Export Checked Forms
Only**

TransForm opens an the Export screen with the forms you selected.

(Image 6.14 - Export Screen in the Management Console)

The screenshot shows the 'Export' screen in the Management Console. The top bar has a back arrow and 'Done' on the left, and 'Export' on the right. The main content is divided into three columns:

- Forms to Export:** A list of two entries, each with the name 'Dave McCormick' and the time '3:12pm'. There are checkboxes next to each entry.
- Export Format:** A section with four radio button options: 'Basic', 'Simple' (which is selected), 'JSON', and 'Tables'. Below these is an 'Email Recipient' input field and an 'Email Data' button. At the bottom of this section are three buttons: 'Download as File', 'Download as PDF', and 'Download for Excel'.
- Census:** A section displaying two census records. Each record is shown in a table format with columns for date/time, status, and email. Below each table is a summary of personal information: 'First Name', 'Last Name', 'Apartment Number', and 'Number of Children'.

To select one of the four types of export formats, click on the one you want. Of these formats, only the Tables format can be exported as a Microsoft Excel (XLSX) file. Here are details about each format type:

Basic - The basic format is a plain text listing with each field listed on a separate line and an extra blank line in between records. If you choose this format you can use the buttons below to Email Data, Download as File (a text file), or Download as a PDF file.

Simple - The simple format places the data from each form in a separate box. Data about the form is displayed horizontally on top. Data in the form is placed inside the box. If you choose this format, you can use the buttons below to Email Data, Download as File (an HTML file), or Download as PDF.

JSON - JSON format is for power users and developers. The data from all of the forms are placed into a JSON data array. This format is sometimes used for development work and for troubleshooting. If you choose this format, you can use the buttons below to Email Data, Download as File (a JSON file), or Download as PDF.

Tables - The Tables format places form data into table rows. Some columns are prefixed with the word “Wrapper_” if they contain information about the form. Other columns are prefixed with “Formdata_” to indicate they contain the actual data in the form. If you choose this format, you can use the buttons below to Email Data, Download as a File (an HTML file), or Download as an Excel File.

Customizing the Forms List

When viewing forms in the Management Console, TransForm shows you the form’s status, when it was created and last edited, and the username to whom the form is assigned.

(Image 6.15 - List of Forms in Management Console)

The screenshot shows the 'alpha Transform' Management Console. The navigation bar includes 'Home', 'Designer', 'Permissions', and 'Management Console'. The main content area is titled 'Building Census' and features a 'Choose Type' dropdown and a 'Select Fields' link. The data is presented in a table with columns for 'Status', 'Created', 'Changed', 'User Name', and 'Details'. The table contains seven rows of form entries, all submitted by 'Dave McCormick' at various times between 11:27am and 11:33am. The third row is highlighted in blue.

Status	Created	Changed	User Name	Details
<input type="checkbox"/> Submitted	11:27am	11:29am	Dave McCormick	
<input type="checkbox"/> Submitted	11:29am	11:29am	Dave McCormick	
<input type="checkbox"/> Submitted	11:30am	11:30am	Dave McCormick	
<input type="checkbox"/> Submitted	11:30am	11:31am	Dave McCormick	
<input type="checkbox"/> Submitted	11:31am	11:31am	Dave McCormick	
<input type="checkbox"/> Submitted	11:32am	11:32am	Dave McCormick	
<input type="checkbox"/> Submitted	11:33am	11:33am	Dave McCormick	

Suppose this form is used for keeping track of people in an apartment building, and you wanted to find the form for a particular apartment. You would need to click on each form individually to find the form you want. Or, you could customize the forms list to include the apartment field (and other fields if you want).

To display other fields in the list, click the Select Fields link. A fields list appears.

(Image 6.16 Fields List in Management Console)

Fields		✓
Row fields { } wrapper		
Form ID	forminstanceid	<input type="checkbox"/>
Form Type ID	formid	<input type="checkbox"/>
Created	created	<input type="checkbox"/>
Completed	completed	<input type="checkbox"/>
Person	person	<input type="checkbox"/>
Person Display Name	displayname	<input type="checkbox"/>
Status	status	<input type="checkbox"/>
haserrors	haserrors	<input type="checkbox"/>
missingrequired	missingrequired	<input type="checkbox"/>
Form Data { } formdata		
Apartment Number	apartment	<input type="checkbox"/>
First Name	firstName	<input type="checkbox"/>
Last Name	lastName	<input type="checkbox"/>
Occupants	occupants	<input type="checkbox"/>

Click on the fields that you want to include, then click the checkmark ✓ at the top right to close the list. The fields now appear in the list.

(Image 6.17 - List of Forms With Custom Fields in Management Console)

Status	Created	Changed	User Name	Details
<input type="checkbox"/> Submitted	11:27am	11:29am	Dave McCormick	Apartment Number: 1, Last Name: August
<input type="checkbox"/> Submitted	11:29am	11:29am	Dave McCormick	Apartment Number: 2, Last Name: Burns
<input type="checkbox"/> Submitted	11:30am	11:30am	Dave McCormick	Apartment Number: 3, Last Name: Chiswick
<input type="checkbox"/> Submitted	11:30am	11:31am	Dave McCormick	Apartment Number: 4, Last Name: Duncan
<input type="checkbox"/> Submitted	11:31am	11:31am	Dave McCormick	Apartment Number: 5, Last Name: Edwards
<input type="checkbox"/> Submitted	11:32am	11:32am	Dave McCormick	Apartment Number: 6, Last Name: Fuccoli
<input type="checkbox"/> Submitted	11:33am	11:33am	Dave McCormick	Apartment Number: 7, Last Name: Giordonno

NOTE: Your selection of fields disappears when you refresh the list of forms. Click Select Fields to choose the fields again.

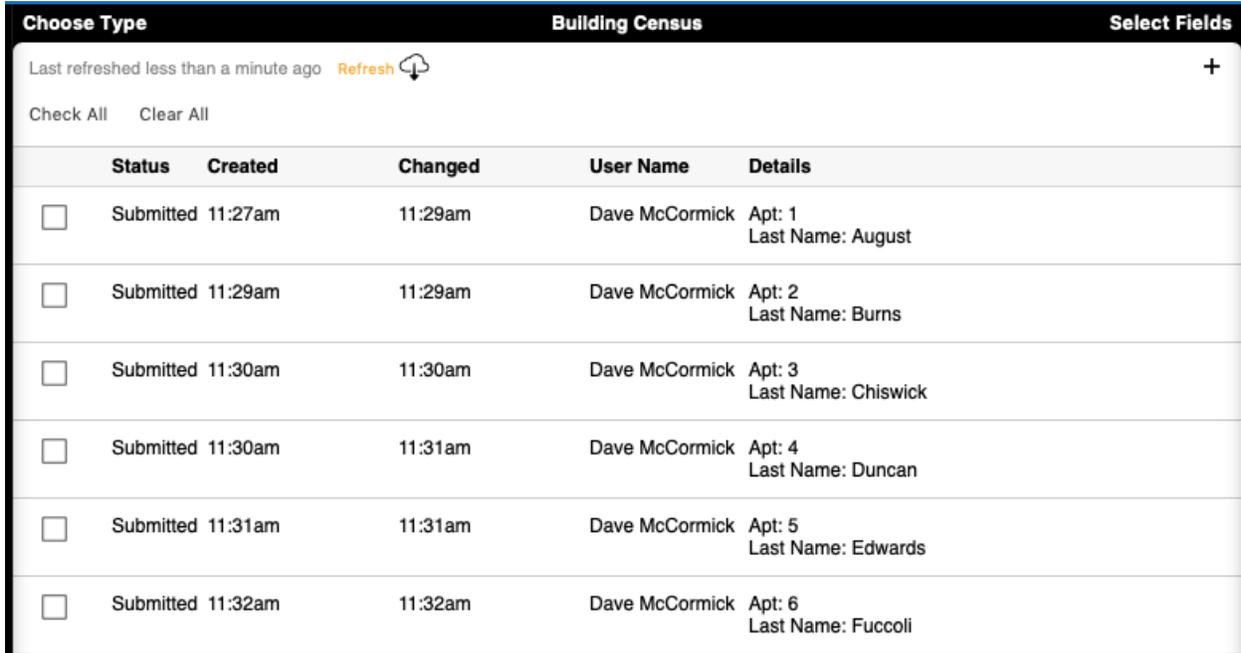
Customizing the Details Field

In the previous example, you selected fields to display in the Form List in the Management Console. The fields appeared below on a separate line. You may have also noticed the list has a column called Details, which is blank. As an alternative to

listing custom fields below on a separate line, you can also customize the details column to insert fields.

Unlike the Select Fields option, the fields you place in the Details column remain in the Details column even after the list is refreshed.

(Image 6.18 - Custom Details Column Showing Apartment Number and Last Name)

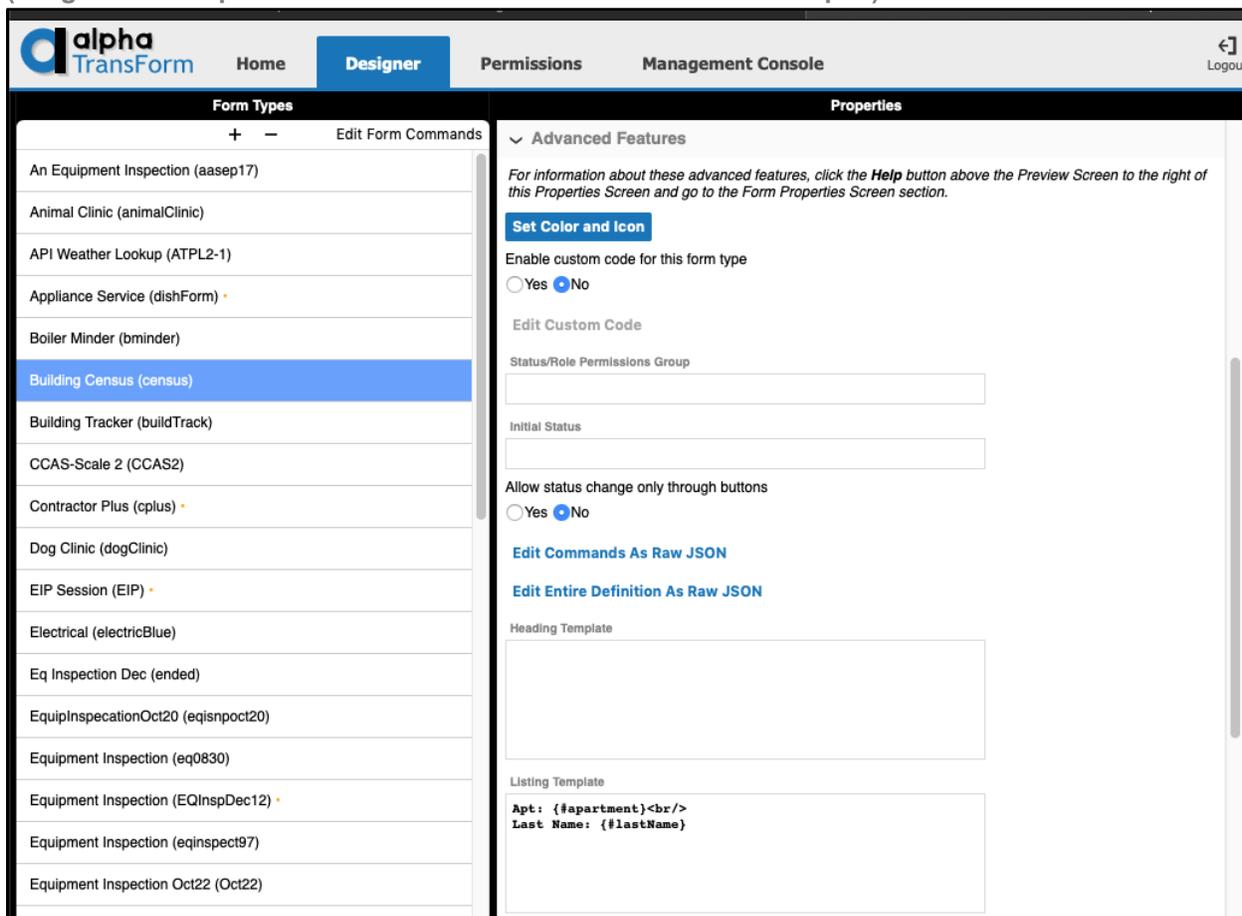


The screenshot shows a table titled "Building Census" with a header bar containing "Choose Type", "Building Census", and "Select Fields". Below the header, there is a status bar indicating "Last refreshed less than a minute ago" with a "Refresh" button and a plus sign. Below that are "Check All" and "Clear All" options. The table has five columns: "Status", "Created", "Changed", "User Name", and "Details". Each row represents a record with a checkbox, a status of "Submitted", and timestamps for "Created" and "Changed". The "User Name" for all records is "Dave McCormick". The "Details" column is customized to show "Apt: [number]" and "Last Name: [name]".

	Status	Created	Changed	User Name	Details
<input type="checkbox"/>	Submitted	11:27am	11:29am	Dave McCormick	Apt: 1 Last Name: August
<input type="checkbox"/>	Submitted	11:29am	11:29am	Dave McCormick	Apt: 2 Last Name: Burns
<input type="checkbox"/>	Submitted	11:30am	11:30am	Dave McCormick	Apt: 3 Last Name: Chiswick
<input type="checkbox"/>	Submitted	11:30am	11:31am	Dave McCormick	Apt: 4 Last Name: Duncan
<input type="checkbox"/>	Submitted	11:31am	11:31am	Dave McCormick	Apt: 5 Last Name: Edwards
<input type="checkbox"/>	Submitted	11:32am	11:32am	Dave McCormick	Apt: 6 Last Name: Fuccoli

In this example, the apartment and lastname fields were added to the Details column. You can customize the Details view for each form type. To do this, click the Designer tab and select the form type you want to customize. Then click the Advanced Features link to open the Advanced Features section.

(Image 6.19 - Properties Pane With Advanced Features Section Open)



Scroll down to the Listing Template box. In this box, enter the HTML and TPL code to determine how the Details column should appear.

(Image 6.20 - Listing Template Example)

Listing Template

```
Apt: {#apartment}<br/>
Last Name: {#lastName}
```

In this example, the apartment field is placed on the top line and the lastName field is placed on the second line. The
 HTML tag inserts a line break after the apartment field so that the lastName field appears on the next line.

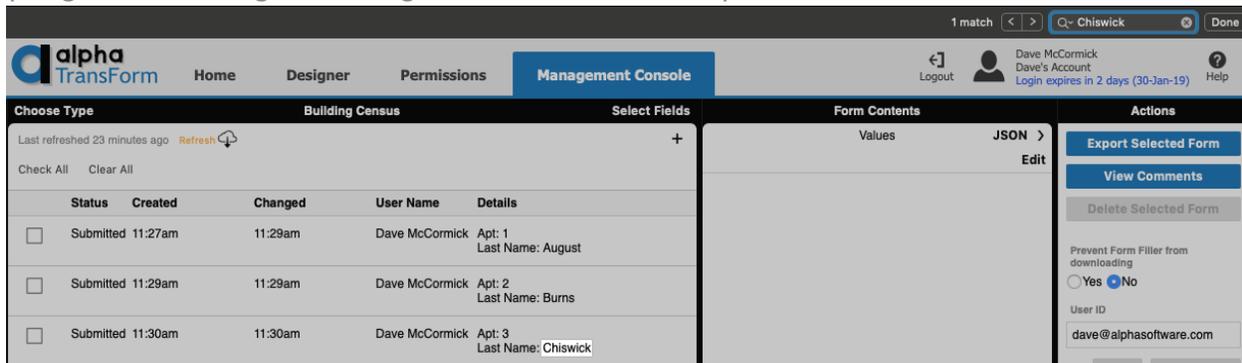
The {#apartment} and {#lastName} are placeholders used in TPL templating to display the contents of the apartment field and the lastName field.

NOTE: For security, TransForm restricts some HTML tags. These include script <SCRIPT> tags and tags that call external URLs like the image and anchor <A> tags.

Sorting and Finding Forms

Forms in the Management Console Form List always appear in the order in which they were created. You cannot change this order. Furthermore, there is no built in Find or Filter commands to search for records. However, you can use the browser's own search capability by pressing CTRL+F or CMD+F on the keyboard and entering a key term.

(Image 6.21 - Finding Data Using Built In Browser Search)

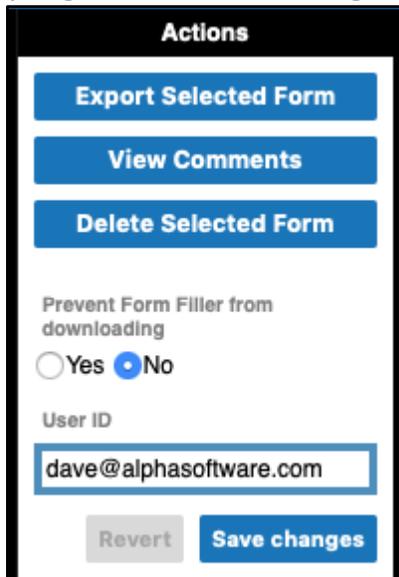


Assigning Forms to Users in the Field

One common use case for TransForm is that of dispatch, where an employee at an office creates a new form and assigns it to a worker in the field. The Creating Forms section, earlier in this chapter covers how to create one (or more) forms, but by default, these forms are unassigned, which means they only appear in the Management Console.

By assigning a user to a form, you can make the form appear when the user next refreshes the data in the Transform app. To assign a user to a form, click on the form you want to assign to select it. Then in the UserID box of the Actions pane, enter the email address of the person to whom you want to assign the form and click Save Changes.

(Image 6.22 - User ID Setting in Actions Pane of the Management Console)



The screenshot shows a mobile interface titled "Actions". It contains three blue buttons: "Export Selected Form", "View Comments", and "Delete Selected Form". Below these is a section titled "Prevent Form Filler from downloading" with two radio buttons: "Yes" (unselected) and "No" (selected). Underneath is a "User ID" label and a text input field containing "dave@alphasoftware.com". At the bottom are two buttons: "Revert" (grey) and "Save changes" (blue).

After you have saved changes, click the Click to Save Changes link to save the assignment to the TransForm cloud. (Make sure the Prevent Form Filler From Downloading button is set to No - otherwise the form will not download to the user's device.)

Click to save changes. 

When the user refreshes the Existing Forms list in the TransForm app, the form will appear.

Adding Comments to a Form

The Management Console allows you to add comments to a form. These comments can be read by users when working with the form in the TransForm app. Comments can be used in variety of ways. For a field service worker, you might include comments

about how to get into a building. Or, you might add a comment that asks the field worker to double check something they entered.

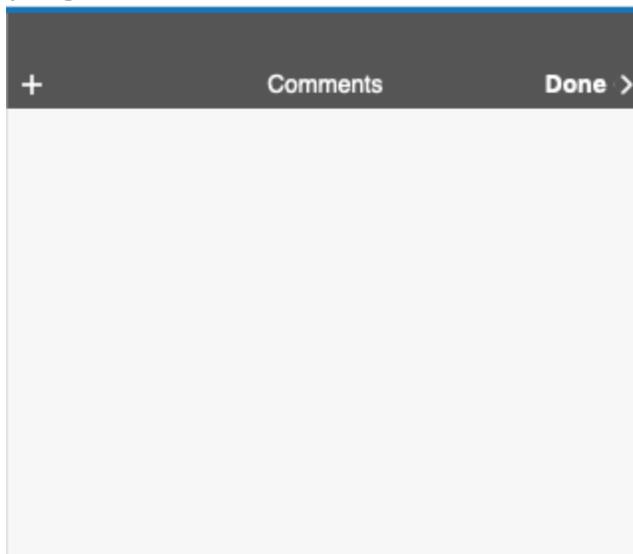
To add a comment to a form, click on the form in the forms list to select it. Then click the View Comments button in the Actions pane.

(Image 6.23 - View Comments Button in Actions Pane)



When you click View Comments, a window slides in showing you the comments that have been previously added to the form.

(Image 6.24 - Comments Part of a Form. No comments have been entered).



To add a comment, click the + button in the toolbar to open a box in which to type.. 

(Image 6.25 - Add Comments Box)



+ Comments Done >

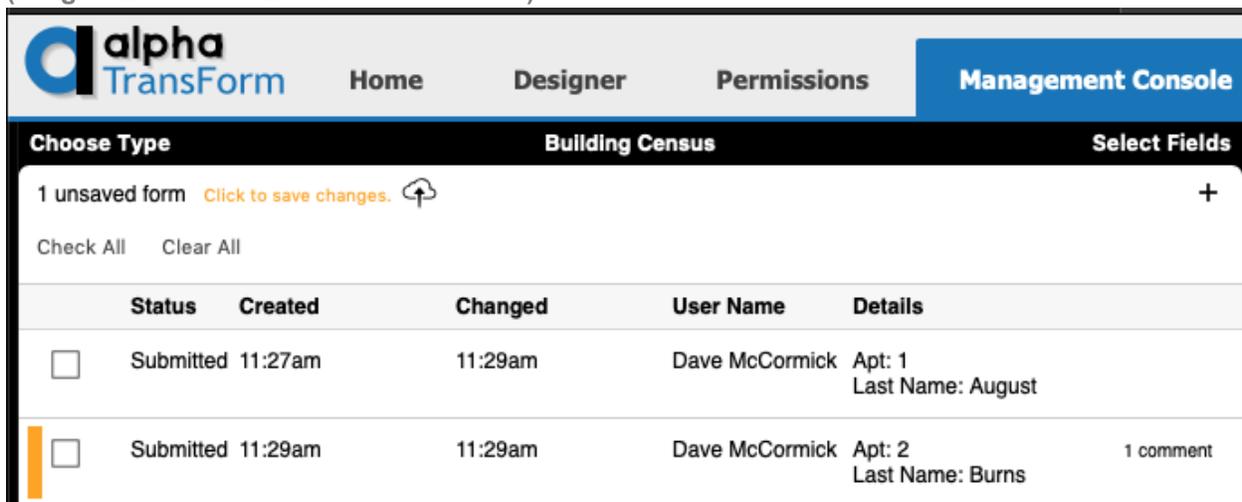
Please check this form. Apartment 2 is supposed to be vacant.

Cancel Save

Type in a comment and click Save to save the comment, or Cancel to close the dialog box without saving.

When a form has a comment, a note appears the Forms list displaying the number of comments that are associated with the form.

(Image 6.26 - Forms List With 1 Comment)



alpha TransForm Home Designer Permissions Management Console

Choose Type Building Census Select Fields

1 unsaved form Click to save changes. ↻

Check All Clear All

Status	Created	Changed	User Name	Details
<input type="checkbox"/>	Submitted 11:27am	11:29am	Dave McCormick	Apt: 1 Last Name: August
<input type="checkbox"/>	Submitted 11:29am	11:29am	Dave McCormick	Apt: 2 Last Name: Burns 1 comment

To view a comment in a form, click on the form to select it, then click the View Comments button in the Actions pane.

NOTES: Once entered, comments cannot be edited or deleted. Also note that while you can view comments both in the TransForm App and in the Management Console of TransForm Central, you can only enter comments in the Management Console of TransForm Central.

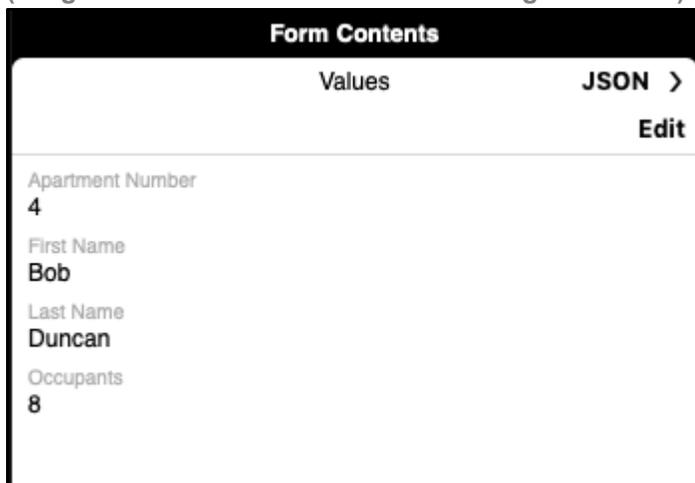
Viewing and Editing Form Data in JSON Format (Advanced)

The data in the forms can be viewed and edited in JSON (JavaScript Object Notation). This advanced feature provides a convenient method for API developers to perform quick tests in an interactive setting. For example, if you have created a script to create new forms, you can paste the JSON output of the script into TransForm to see if your syntax is correct. Likewise, you can copy JSON and paste it into a system you plan to use for reading TransForm data.

Viewing and Editing the JSON Data for a Form

To view the JSON data for a form, click on the form in the list to select it. The contents of the form appears in the Form Contents box.

(Image 6.27 - Form Contents Box Showing Form Data)

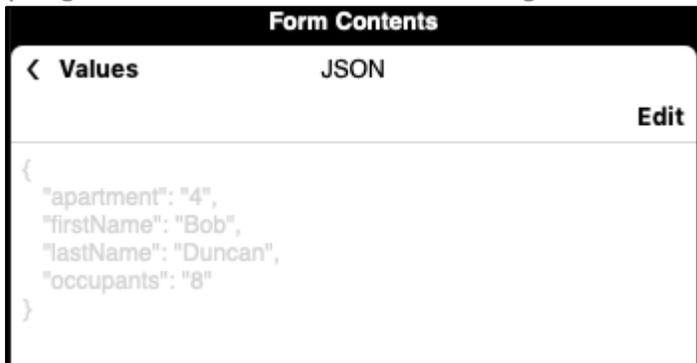


The screenshot shows a window titled "Form Contents" with a toolbar at the top containing "Values", "JSON >", and "Edit". The main area displays the following form data:

Apartment Number	4
First Name	Bob
Last Name	Duncan
Occupants	8

Next, click the JSON > link in the toolbar. The form data is displayed in JSON format. From there, you can select it and copy it to the clipboard.

(Image 6.28 - Form Contents Box Showing Form Data in JSON Format)



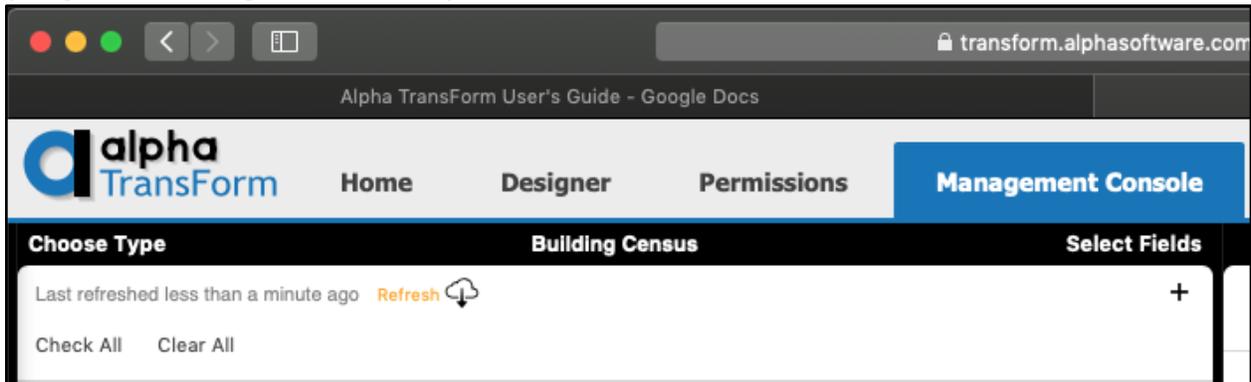
You can also edit the JSON data by clicking the Edit link in the Form Contents toolbar.

You can also export multiple records to JSON format, refer back to the Exporting Data section in this chapter for instructions.

Creating a New Form Using JSON Data

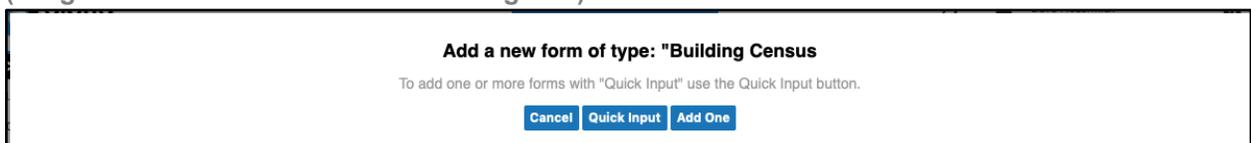
To create a new form using JSON data, click the Choose Type link in the toolbar to select the type of form you would like to create. Then click the + button in the toolbar to create a new form.

(Image 6.29 - Management Console)



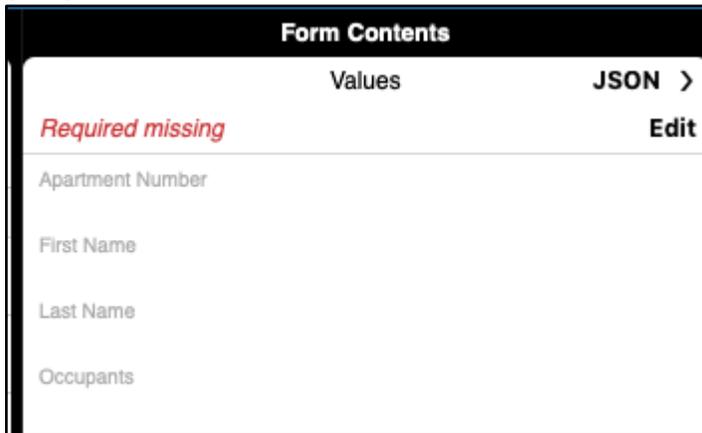
A dialog box appears asking you the method you would like to use. Click Add One.

(Image 6.30 - Create Form Method Dialog Box)



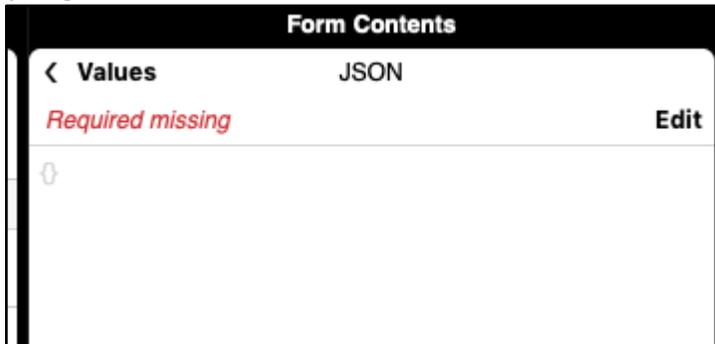
A blank form is created.

(Image 6.31 - Form Contents in Standard View - Form is Blank)



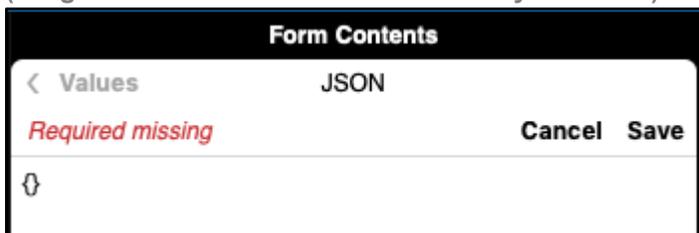
Click the JSON link in the toolbar to see the JSON data currently filled in to the form.

(Image 6.32 - Form Contents - JSON View - Form is Blank)



Click the Edit link to enable the text box. The JSON text (which consists right now of just { }) switches from light gray to black to indicate you can edit it.

(Image 6.33 - Form Contents JSON Ready For Edits)



Paste in the JSON text you want to use, then click the Save link.

Chapter 7: TransForm Programming Language

Alpha TransForm includes a powerful, client-side programming language called the TransForm Programming Language (TPL). With TPL, you can add programming code to your form types to create extra functionality in your forms.

For example, TPL can be used to look up values from an onboard database when the user performs a barcode scan. TPL can also be used to perform calculations and to embed business logic.

TPL shares some similarities with JavaScript, Visual Basic, and other languages, and developers who already know another programming language find TPL easy to master.

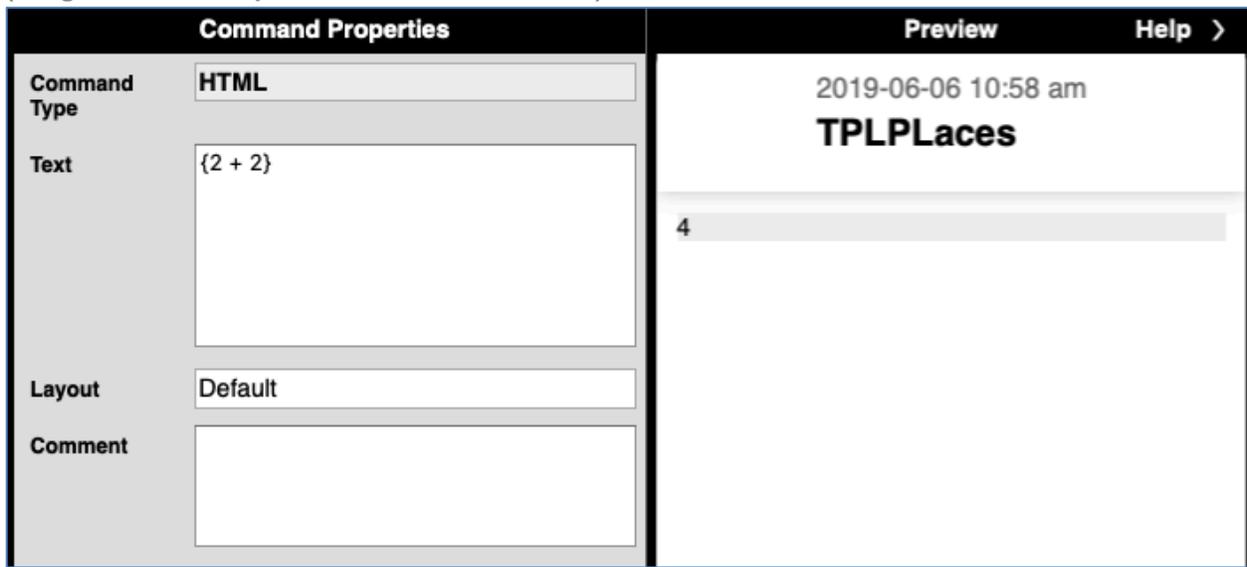
If you have limited programming experience, you can use TPL by copying the examples we provide here and modifying them for your own use.

TPL Expressions in IF Commands, HTML Blocks, Headers, and Templates

Many of the examples in this chapter demonstrate how to use TPL to run “event code,” that is code that runs when something happens – like when a field value is entered, or a button is clicked. However, you can also use one-line snippets of TPL (called “expressions”) in IF commands, HTML blocks, Heading Sections, and in Header and List Templates.

To use a TPL expression, enclose it in curly braces `{}`. The value that is returned by the expression is then inserted into the HTML, header, IF command, or template. In the following example, we have inserted a TPL expression to perform a simple calculation.

(Image 7.1 – TPL Expression in an HTML Block)

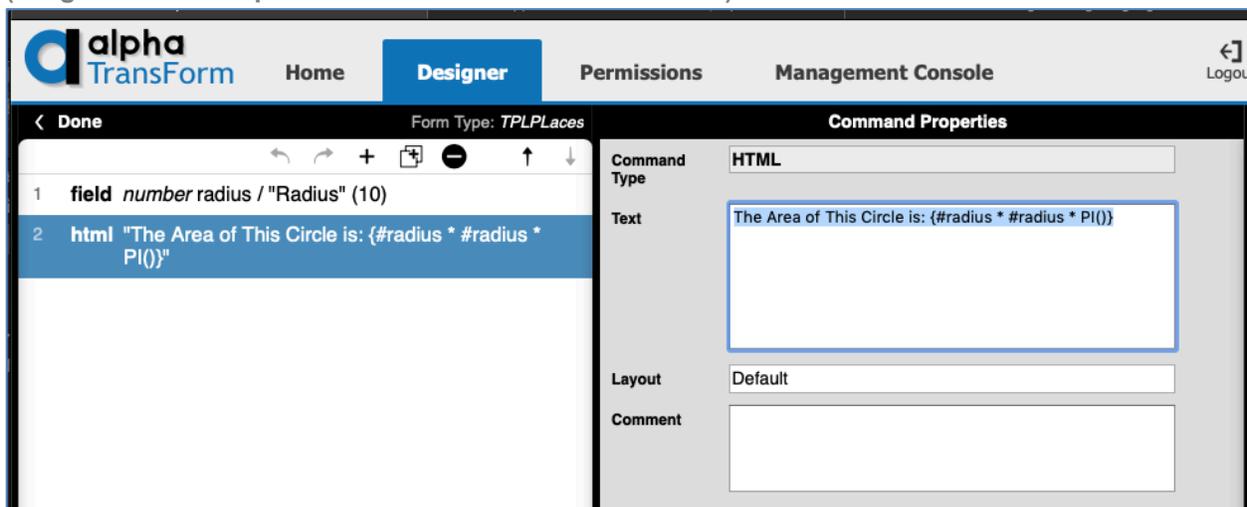


The preview window displays the result of 4.

In this next example, the value of the field “radius” is squared and multiplied by PI to determine the area of a circle. In this case the TPL expression is:

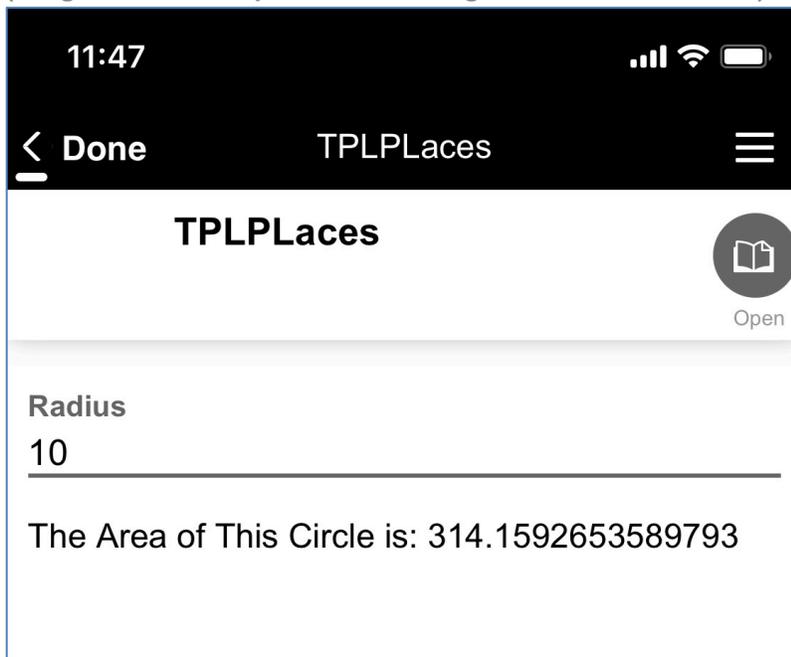
```
{#radius * #radius * PI()}.}
```

(Image 7.2 – TPL Expression to Calculate Area of a Circle)



In the example below, the form is run on a mobile phone:

(Image 7.3 – TPL Expression Running in a Form on a Phone)



When the user enters a value into the Radius field, the area of the circle is calculated and inserted into an HTML block.

NOTE: TPL expressions that refer to fields, user defined functions, and even some of the built-in functions cannot be run in the Preview window. To see their results, you need to open the form on a mobile device, as we've done in this example.

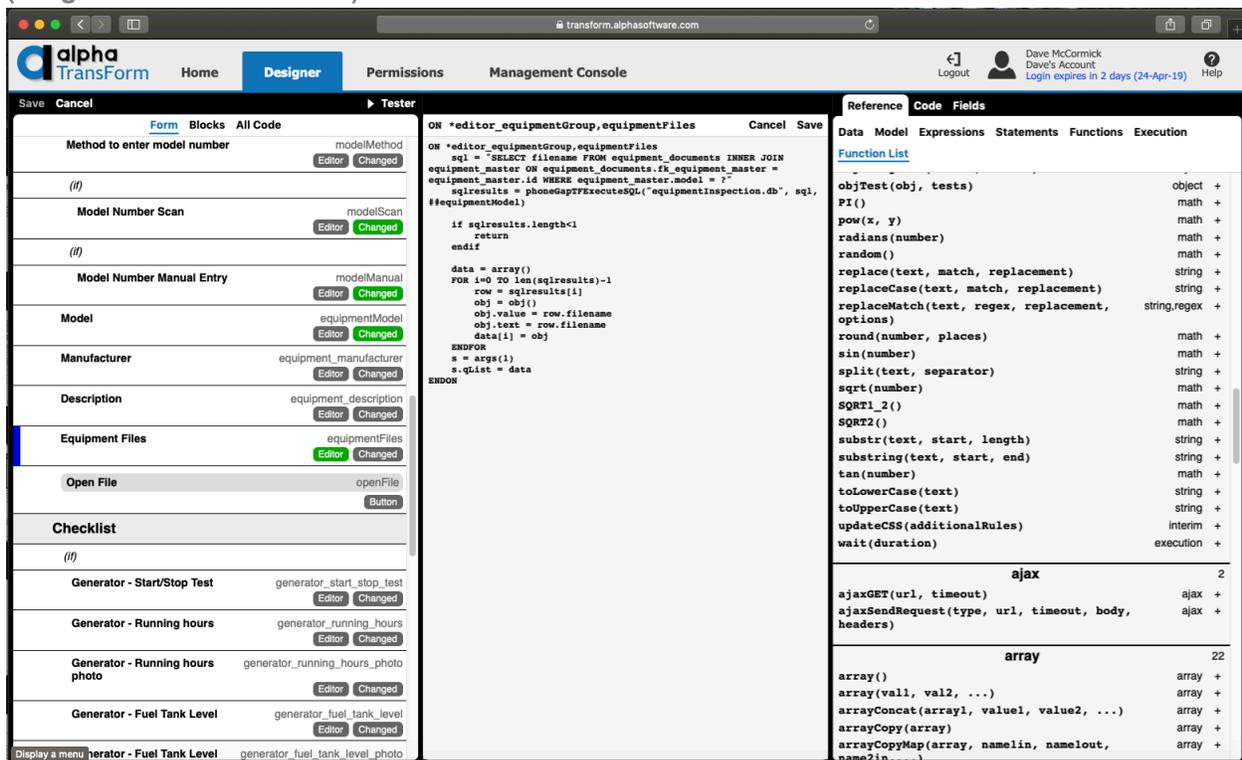
The TPL Code Editor

TransForm Central has a built-in TPL code editor that is opened in the Designer tab.

To open the TPL editor:

1. Log in to TransForm Central and click the Designer tab.
2. From the list of Form Types, select the form type you want to use.
3. Click the “Advanced Features section” to expand it, if necessary.
4. Set the “Enable Custom Code For This Form Type” to Yes.
5. Click the Edit Custom Code link. The TPL Code Editor appears.

(Image 7.4 TPL Code Editor)



The code editor is divided into three panels. The panel on the left shows all of the events to which you can attach code. When an event has code attached to it, the name of the event appears in green on the list. Events without code appear in gray. When an event occurs, the code associated with that event is run. See the “Events” section (below) for an explanation.

The center panel is for entering TPL code to associate with the selected event. And the right panel contains tools and online documentation to help you create your code.

Along with the three panels, there is also a link to the Tester, which you can use to help debug your code. For more information, refer to “Debugging Your Applications,” later in this chapter.

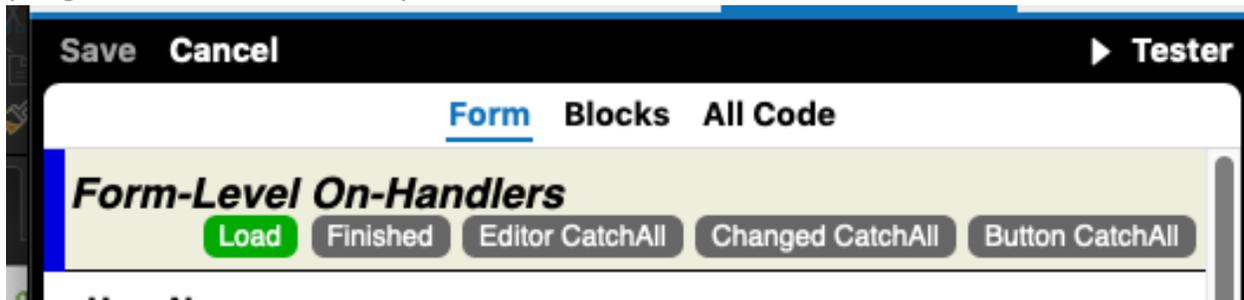
Events

Each field in your form has two events: the Editor event and the Changed event. When a user taps on a field in the TransForm app, TransForm runs any TPL code you have added to the Editor event for that field. When the user enters information into a field, TransForm runs any TPL code you have added to the Changed event.

Buttons have one event, simply called “Button”. The Button event runs when the user taps on the button.

In addition to the events specific to fields and buttons, there are five other events that apply to the form as a whole.

(image 7.5 – Form Level Events)



- Load – Code attached to the Load event runs when the form is first opened.
- Finished – Code attached to the Finished event runs when the user closes the form, either by tapping a submit button or by tapping the Done link in the app.
- Editor CatchAll – Code attached to the Editor CatchAll event runs whenever a user taps on or swipes to a field to edit it. If the field already has an Editor event defined for it, TransForm runs the code in the field’s Editor event and ignores the code in the Editor Catchall event.
- Changed CatchAll – Code attached to the Changed Catchall event runs whenever a user enters or changes a value in any field. If the field already has a Changed event defined for it, TransForm runs the code in the field’s Changed event and ignores the code in the Changed Catchall event.

- Button CatchAll – Code attached to the Button CatchAll event runs whenever the user taps on any button. If the button already has a Button event defined for it, TransForm runs the code in the button’s Button event and ignores the code in the Button CatchAll event.

Event Behavior You Might Not Expect

In general, the list above is accurate as to when TPL code will run. However, there are two cases where events may not behave as you might expect. Here are those cases:

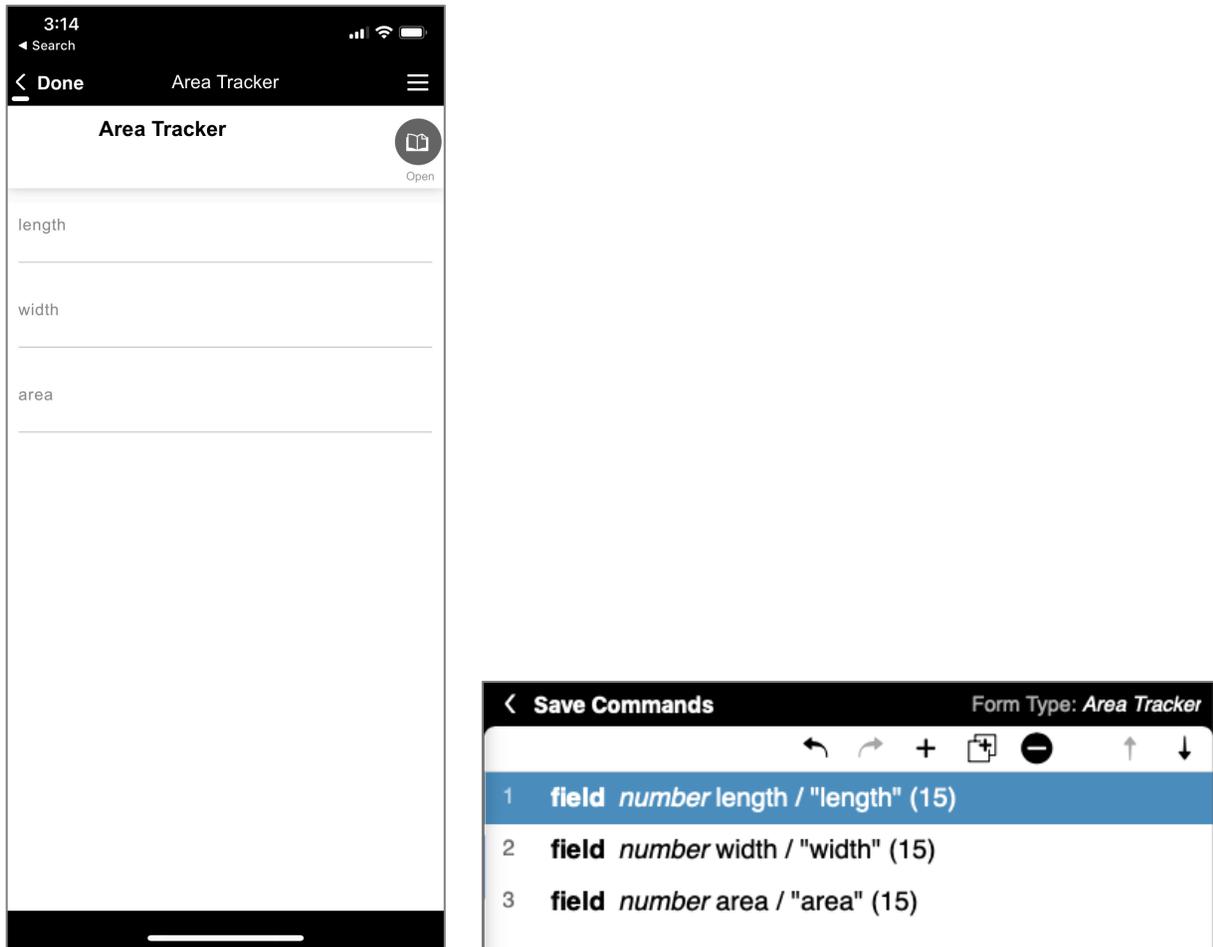
- When a list field is blank, opening its editor and closing it without making a selection will cause the Changed (or Changed CatchAll) event script to run. You might expect that leaving a field empty that was empty to begin with does not count as a change, but it does. Therefore, do not depend on the user having chosen a value from a list simply because the Changed (or Changed CatchAll) script ran.
- When the user taps on or swipes to a field, the Editor (or Editor CatchAll) event script runs as expected. However, unlike the Changed event script, it does not refresh the display when you leave the editor. This means, for example, if your Editor script sets a field to a new value, you won’t see that value until the display is refreshed. So, if you want to set a field value in an Editor event, you may want to consider adding a dummy script (like $A=1$) to the Changed event to force it to refresh when the user exits.

Example: Calculation

A basic use for TPL is to perform calculations. For example, suppose you had a form that captured the length and width of a room, and you wanted to calculate the area. As soon as you fill in the length and width fields, TPL can multiply these two numbers and place them in a field called “area”.

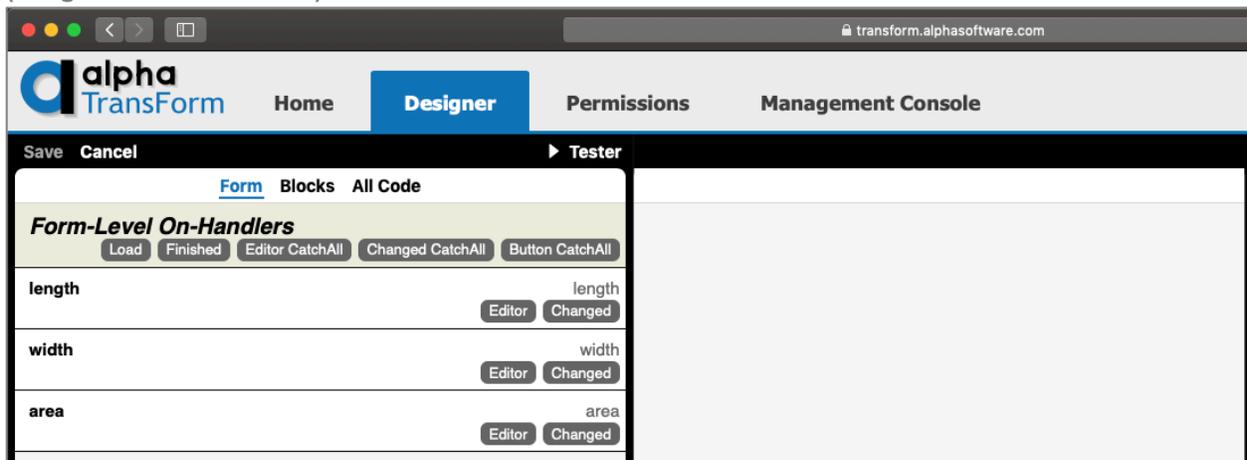
To do that, we’d first create a form type with the three numeric fields: length, width, and area.

(Image 7.6 – Simple Form with Fields for Length, Width, and Area)



With the form created, the next step is to open the Code Editor.

(Image 7.7 – Code Editor)



Because we want to calculate the area when we change either the length or the width, we need to add code to the Changed events for both the length and width fields.

The code to enter for both is:

```
#area = #length * #width
```

This line of TPL instructs TransForm to set the field called Area to the value of Length times the value of Width. (Note the syntax of “#” in front of the fieldname.)

After you have entered the events, the TransForm Central screen looks like this:

(Image 7.8 – Code Editor with Code for Two Events)



The Changed buttons are now green, indicating that there is code attached to them. Once the code has been saved to the form and the form type has been saved to TransForm Central, you can test it out.

(Image 7.9 – A Form That Performs a Calculation)

3:44

< Done Area Tracker

Area Tracker

Open

length
5

width
6

area
30

If you enter 5 for the length and 6 for the width, the area field should automatically display 30. Furthermore, changing either of the fields should cause the area to recalculate.

NOTE: In this example, the code ($\text{area} = \text{length} \times \text{width}$) was very short, so duplicating the code for each event made sense. There may be cases, however, in which you have more sophisticated code that spans many lines, and you want to use that code in multiple events. In those cases, it is better to create a function, and then have the event call the function rather than run the code directly. For more information, see the section “Creating Functions,” later in this chapter.

Example: Populating a List Field

In the previous example, we used TPL to set the value of a field. In some cases, you might instead want the user to set the value of a field by choosing from a list.

Furthermore, you might want the list to be smart and only include choices that make sense based on a previous answer.

For example, imagine a form for collecting data about animals, which contains two fields: Class and Type. The choices for class are Mammal and Reptile. If the user chooses Mammal, the Type list contains the choices: Dog, Cat, and Squirrel. If the user chooses Reptile, the choices should instead be Snake, Lizard, and Turtle.

To do this, you would add a TPL script to the Editor event for the Type field.

(Image 7.10 – Code to Populate a List Field Attached to Editor Event)



Here is the code:

```
ON *editor_type

s = args(1)
list = array()

IF #class == "Mammal"
list[0] = obj("value", "Dog")
list[1] = obj("value", "Cat")
list[2] = obj("value", "Squirrel")
ELSE
list[0] = obj("value", "Snake")
list[1] = obj("value", "Lizard")
list[2] = obj("value", "Turtle")
ENDIF

s.qList = list

ENDON
```

The code is run when the Type field is edited. When it runs, a variable called “s” is created and assigned a value using the args() function. This, in turn, creates an attribute in s called qList. At the end of the script, s.qList is set to the appropriate list of choices from which the user can choose.

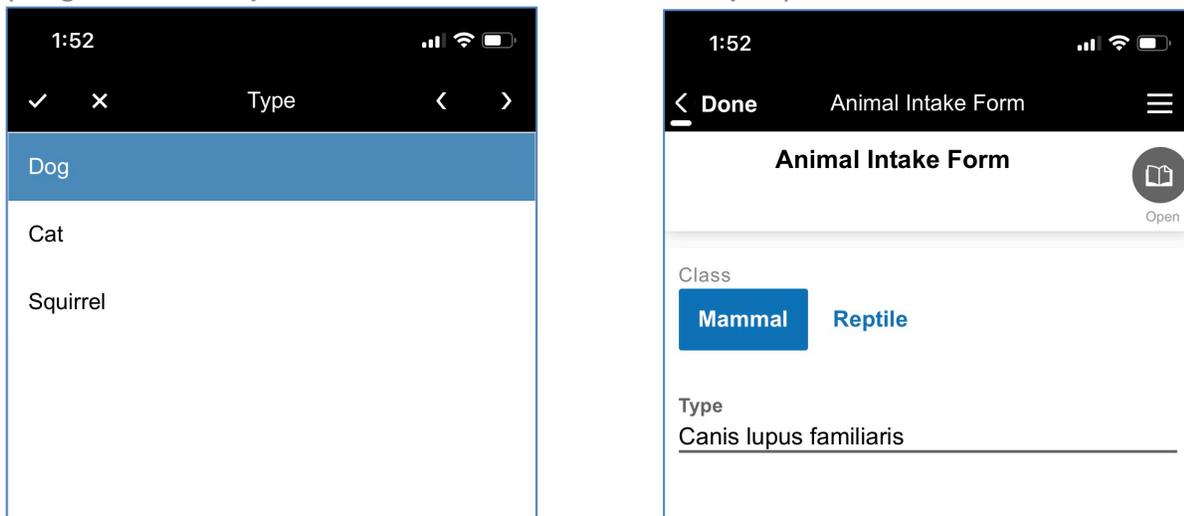
The code in the middle determines which animals should be in the Type list based on what they chose in the Class field.

Besides qList, fields can have other arguments as well that control their behavior. For details, refer to the online documentation for the args() function.

Displaying One Value and Saving Another

In our previous example, we populated a list with different animal names based on whether the user selected Mammal or Reptile. Now suppose that instead of the common names for each of the animals, we wanted instead to use the scientific names. So instead of “Dog,” the choice would be “Canis lupus familiaris.” Furthermore, suppose we don’t expect users to know the scientific names, so we want them to choose from a list of common names. When they select a common name, the scientific name is saved instead.

(Image 7.11 – Example of the text and value attributes of qList)



Here is the code to do that.

```

ON *editor_type

    s = args(1)
    list = array()

    IF #class == "Mammal"
        list[0] = obj("text","Dog","value","Canis lupus familiaris")
        list[1] = obj("text","Cat","value","Felis catus")
        list[2] = obj("text","Squirrel","value","Sciuridae")

    ELSE
        list[0] = obj("text","Snake","value","Serpentes")
        list[1] = obj("text","Lizard","value","Lacertilia")
        list[2] = obj("text","Turtle","value","Testudines")

    ENDIF

    s.qList = list

ENDON

```

You may have noticed that all of the coding examples that use qList also use the obj() function. That is because qList is an array of objects, not an array of strings. Each of these objects has an attribute called “value”, which is a string, and it is this attribute that is set to the name of the choice. qList objects also have another, optional, attribute called “text”. When the list is displayed, TransForm checks each list choice to see if there is a text attribute associated with that choice. If there is, its text attribute is displayed in the list instead of its value attribute.

In the example above, the text attribute for each choice is assigned a common name, and the value attribute is assigned a scientific name. The user sees the common names, but the system saves the scientific names.

Example: Populating a List Field From an API

In the previous example, TPL was used to populate a list field with predetermined (static) choices that were hardcoded into the script. Now let’s say instead we wanted those choices to be dynamically populated by referring to a web service, so the list is updated in real-time.

For example, maybe you have a list of available appointment times, or a list of products in inventory, or a list of employees on duty. In each case, having up-to-the-minute information could help the person in the field make a better decision.

You can make AJAX callbacks to services using one of two functions: `ajaxGet()` and `ajaxSendRequest()`. The `ajaxGet()` function is the simpler of the two functions, in that you only provide it with a URL and, optionally, a timeout value. While `ajaxSendRequest()` is more sophisticated and allows you to include a body, headers, and other parts to your AJAX request.

In this example, the `ajaxGet()` function is used to make an API call to a web service in order to retrieve a list of names and use them to populate a list field.

```
ON *editor_partners

  s = args(1)

url="https://server.alphasoftware.com/hubfs/bitbucket/sampleData/PartnerList.json"

  r = ajaxGET(url)

  if (r.error)
    msgShowWarning("Could not connect to service", r.error)
    return
  endif

  rdata = JSONparse(r.responseText)
  list = array()
  row = obj()
  for i=0 to len(rdata)-1
    row = rdata[i]
    item = obj()
    item.value = row.name
    list[len(list)] = item
  endfor

  s.qList = list
```

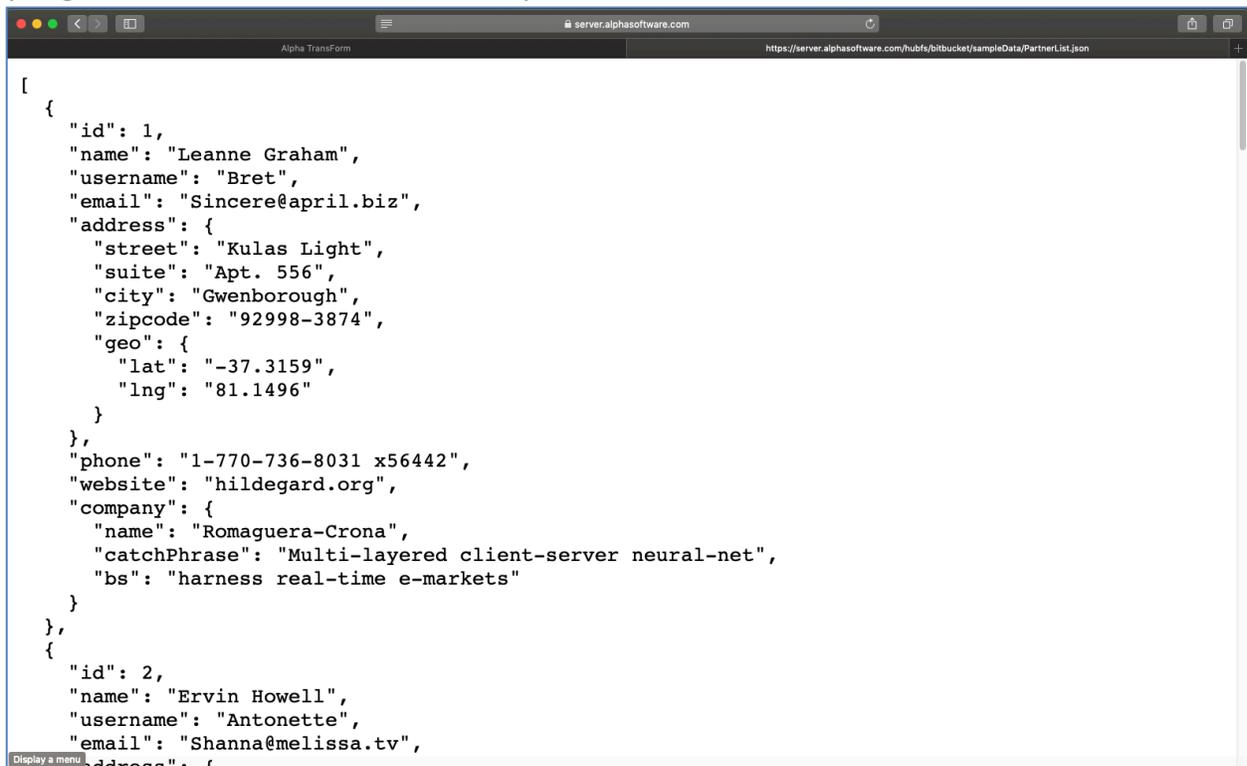
ENDON

This code is similar to the previous example in that once again a variable called “s” is created and set using the args() function to create an s.qList attribute. At the end of the script, s.qList is set to the list of choices.

In the middle of the script four things happen. First, the ajaxGet() command is used to make a call to a web service whose URL is defined in the url variable. Next, a check is performed to see if there is an error; if there is, an error message is displayed. After the error check, the JSON data that is returned from the server is parsed using the JSONparse() function. And finally, a FOR loop is used to iterate through the response data to extract the names and use them to populate an array called “list”, which in turn populates the list by setting the s.qList attribute.

By the way, feel free to use this web service. It was set up specifically to allow developers to test their TPL applications. You can see the format of the data that is returned by pasting the URL into a web browser.

(Image 7.12 – JSON Data from a Web Call)

A screenshot of a web browser window showing a JSON response. The browser's address bar contains the URL: https://server.alphasoftware.com/hubfs/bitbucket/sampleData/PartnerList.json. The main content area displays the following JSON data:

```
[
  {
    "id": 1,
    "name": "Leanne Graham",
    "username": "Bret",
    "email": "Sincere@april.biz",
    "address": {
      "street": "Kulas Light",
      "suite": "Apt. 556",
      "city": "Gwenborough",
      "zipcode": "92998-3874",
      "geo": {
        "lat": "-37.3159",
        "lng": "81.1496"
      }
    },
    "phone": "1-770-736-8031 x56442",
    "website": "hildegard.org",
    "company": {
      "name": "Romaguera-Crona",
      "catchPhrase": "Multi-layered client-server neural-net",
      "bs": "harness real-time e-markets"
    }
  },
  {
    "id": 2,
    "name": "Ervin Howell",
    "username": "Antonette",
    "email": "Shanna@melissa.tv",
    "address": {
```

<https://server.alphasoftware.com/hubfs/bitbucket/sampleData/PartnerList.json>

Example: Populating a List Field from an Onboard Database

In the previous example, TPL was used to populate a list using data from a web service. Obviously, this can only work if the device is connected to the internet. Unfortunately, there are many cases where a WIFI or cellular signal is not available, so an API lookup would not be a good solution. In these cases, you can use TPL to refer to a database that is stored right on the device, and it can be used regardless of whether or not the device is connected to the internet. The device only needs to be connected periodically (for example, at the beginning of the day) in order to refresh the database.

In this example, we have preloaded a database onto the mobile device called Customers (See “Creating a SQLite Database,” later in this Chapter). The Customers database contains a table that is also called Customers. When the user taps on the list field, the following TPL script is executed, which populates the list field with names from the customers table:

```
ON *editor_dbChoiceList
s = args(1)

'START SQLITE CODE
  sql = "SELECT CONTACTNAME FROM customers"
  sqlresults = phoneGapTFExecuteSQL("customers.db", sql)

  data = array()
  FOR i=0 TO len(sqlresults)-1
    row = sqlresults[i]
    obj = obj()
    obj.value = row.CONTACTNAME
    data[i] = obj
  ENDFOR
  s.qList = data

ENDON
```

As with the other example, the `args()` function is used to create an attribute in the variable “s” called “qList”.

The middle of the script executes a SQL statement using the `phoneGapExecuteSQL()` function. This returns a result set. A FOR loop then iterates through the result set and populates an array called “data”.

At the end of the script, `qList` is set to an array called “data” which contains the choices that were pulled from the database.

For information on the syntax of SQL statements you can use, please refer to the SQLite documentation: <https://sqlite.org/lang.html>

For information on how to load a SQLite database onto a mobile device and how to refresh it, refer to the section “Creating a SQLite Database,” later in this chapter.

Example: Looking Up Data from a Barcode Scan or List Choice

Sometimes when you make a selection in a list, you would also like other fields to be filled in at the same time. For example, suppose you chose a customer from a list of names. It might be helpful to automatically look up and fill in that customer’s address and contact information automatically. Or similarly, you might want to scan a VIN number from a car and automatically fill in its make, model, and owner.

The example below shows how to look up information based on a selection made from a list field. In this case, the user chooses a name from a list, and TPL looks up the company, address, and phone number associated with that name and fills it into fields on the form.

```
ON *changed_dbChoiceList

sql = "SELECT COMPANYNAME, ADDRESS, CITY, POSTALCODE, COUNTRY, PHONE FROM customers
WHERE CONTACTNAME = ?"

sqlresults = phoneGapTFExecuteSQL("customers.db", sql, #dbChoiceList)

    if len(sqlresults)==0
        return
    endif

    row = sqlresults[0]
    #company = row.COMPANYNAME
    #address = row.ADDRESS
    #city = row.CITY
    #postcode = row.POSTALCODE
```

```
#country = row.COUNTRY  
#phone = row.PHONE
```

```
ENDON
```

Unlike previous examples, this code is attached to the Changed event instead of the Editor event. That's because we want this code to run after the user has made a selection.

The code starts by creating a variable called "sql," which is used to store the SQL statement that will be executed. Note that at the end of the SQL statement is a "?". This is an argument, and it will be replaced by a value when the command is executed.

The value that is inserted into the argument is the value in the dbChoiceList field. Although it is possible to build the SQL statement string without using an argument, arguments are recommended for security reasons and can make your code easier to read.

Once the statement is executed, a check is performed to see if any results were returned. If no results were found, the script ends. Otherwise, the first (and only) row of the result set is selected and used to fill in the fields on the form.

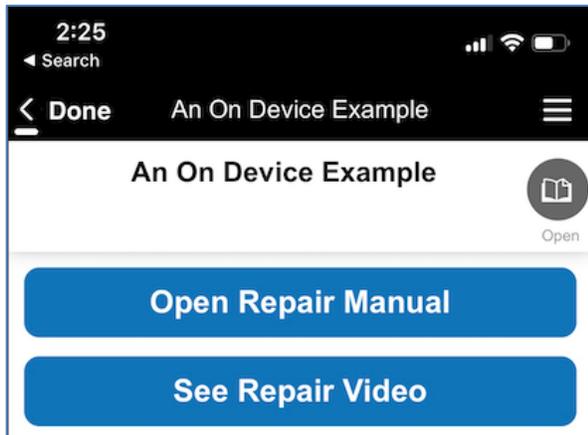
This example used a list field, but you can use this same code with (almost) any kind of field, including a Scan field. That way, you can use the mobile device to scan a code and then use that code to look up and fill in other fields on the form.

Launching Videos, PDFs, Spreadsheets and Other Documents

In previous examples, you saw how you can save time by performing calculations and lookups to make data entry faster and more accurate for workers in the field. TPL can also be helpful in other ways, such as providing access to repair manuals, blueprints, instructional videos and other assets. These are stored on the device, so they are available offline.

In this next example, a form has two buttons on it, one opens a PDF manual and the other plays an instructional video.

(Image 7.13 – Example Form with Buttons to Launch a PDF and Video)



The code to open the repair manual is:

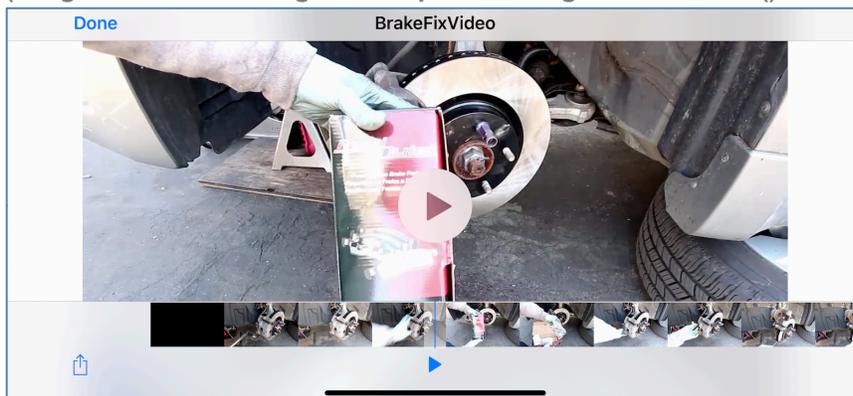
```
ON *button_openManual
  fileviewer("pdfDocs/ZamboniManual.pdf")
ENDON
```

And the code to open the instructional video is:

```
ON *button_repairVideo
  fileviewer("videos/BrakeFixVideo.mp4")
ENDON
```

Each of these scripts use the fileviewer() function. This function automatically opens a file using the correct application for the filetype. For example, PDF documents open in the default document reader for the mobile device (possibly Adobe Acrobat if that is installed). Video files open in the default video player.

(image 7.14 – A Training Video Opened Using the Fileviewer() Function)



When the user has finished with the file, they tap done at the top left corner of their screen to return to the TransForm application.

Getting Driving Directions to an Address

In previous examples, we used the `fileviewer()` function to open files stored on the device. TPL has a similar function called `inappbrowser()` that opens a web page. There are many reasons for doing this, but one particularly useful reason is that it lets you open and display driving directions to workers in the field.

In this example, when a worker opens a form on their mobile device, they see the address where the work should be performed along with a button called `Navigate to Site`. Clicking that button opens Google in a web browser with the coordinates preloaded.

In this case, the Google Maps app was also installed on the phone, so when the web page appeared, another button appeared to launch the Google Maps app.

(Image 7.15 – Form That Opens Google Maps to Show Driving Directions)

9:36

< Back Equipment Inspection *

Site/Location details

Site/Building name
Springfield Office

Address
510 Parler Street

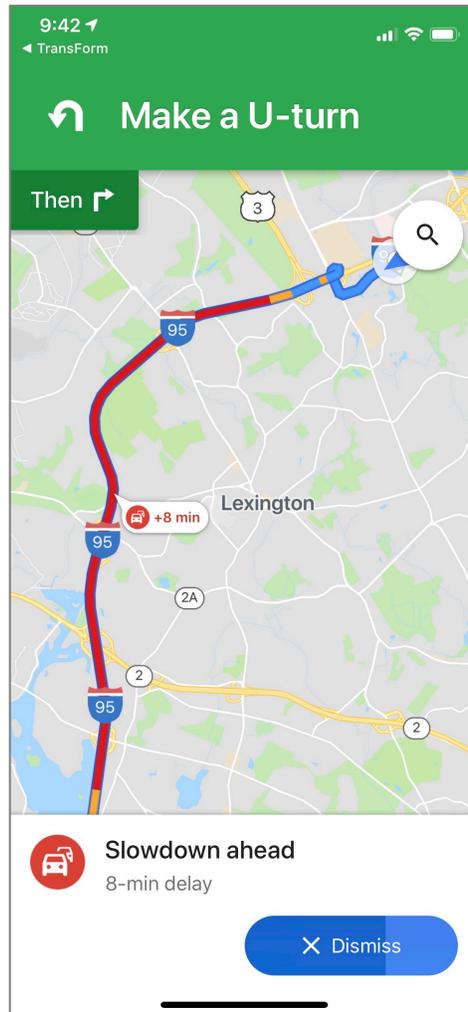
City	State/Region	Zip Code
Springfield	MA	00129

Country	Latitude	Longitude
USA	42.1015	75.5898

Site notes

Navigate to Site

Weather forecast



This is a live service, so it is not suitable for offline use. But because it is live, you can see traffic information and alternate routes in real time.

The code to do this is below, and it assumes that you have field called latitude and a field called longitude in your form, and that those fields contain the coordinates.

```
ON *button_btnNavigate
  lat_lon = #latitude & "," & #longitude
  inappbrowser("https://www.google.com/maps/search/?hl=en&api=1&query=" &
  lat_lon)
ENDON
```

Alternatively, if you do not have the latitude and longitude, you can look up a location using its street address using this code:

```
ON *button_navigateStreet
inappbrowser("https://www.google.com/maps/search/?hl=en&api=1&query=" & #address &
"%20" & #city & "%20" & #state_region & "%20" & #zip_code)
ENDON
```

Creating Functions

If you have programming experience in another language, you are probably already familiar with the concept of functions. They are reusable blocks of code that save you programming time and make your code easier to maintain.

Here is a simple example. Suppose you have a form that is used to measure temperatures of materials in a factory. You want to store all of the values in Celsius, but some of the equipment displays temperatures in Fahrenheit, so you'll need TPL to make the conversion.

The following code runs when the value of the Temp F field changes. When this occurs, the Converted Temp field is set the Celsius value using a formula.

```
ON *changed_tempF
#convertedTemp = (#tempF - 32) * (5/9)
ENDON
```

Now imagine that you needed to perform that exact same calculation in many different places in your application. You could simply copy and paste the code, which would be fine in this case because it is a very simple example; there's just one line of code. Suppose though, instead of one line of code, it were many lines of code. Perhaps the code makes an API callback or a database lookup. You could instead wrap that code up into a function, and then call the function, like in this example:

```
ON *changed_tempF
#convertedTemp = @convertToCelsius(#tempF)
ENDON
```

In above example, the temperature in Fahrenheit is passed into a function called `convertToCelsius`, and the function performs the calculation. The code for that function is as follows:

```
FUNCTION @convertToCelsius

    tempIn=args(1)
    tempC=(tempIn-32)*(5/9)
    return tempC

ENDFUNCTION
```

The first line of the function sets a variable called `tempIn` to the value of the Fahrenheit temperature that was passed into the function. This is done using the `args()` function. (You may recognize the `args()` function from earlier examples in which we populated list fields with choices.)

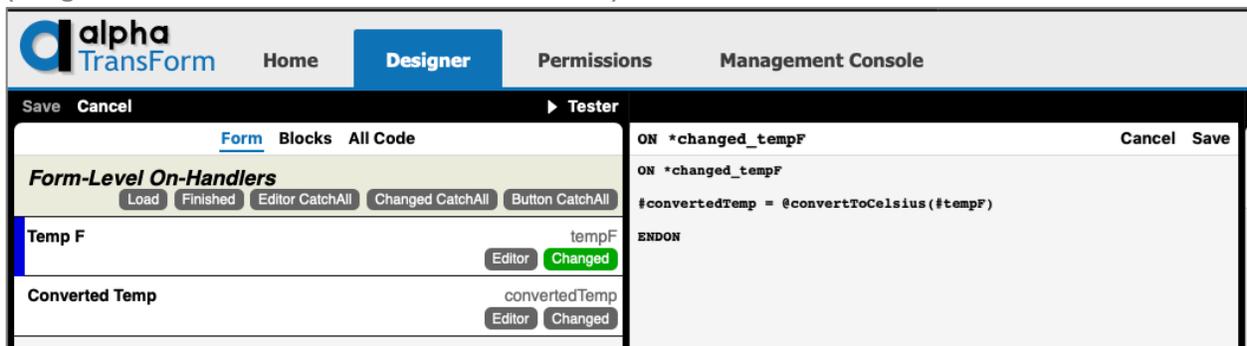
Next, a variable called `tempC` is set to a value computed using the `tempIn` variable.

Lastly, the value of `tempC` is returned, where it is used to set the `convertedTemp` field equal to the Celsius temperature.

Creating a New Function

Most of the TPL code so far was added by clicking on one of the Events for the form, as shown below where a short script was added to the `TempF` field.

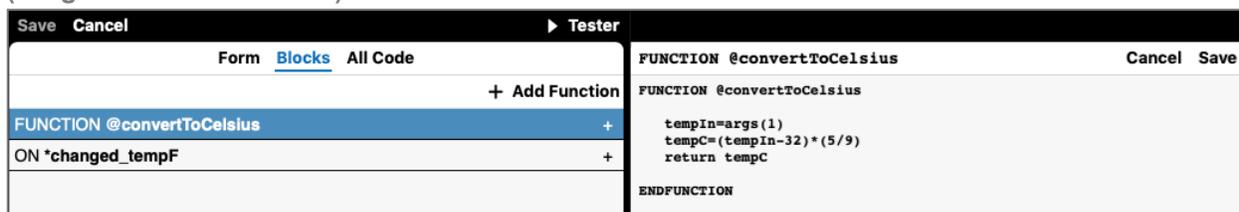
(Image 7.16 – TPL Code with a Custom Function)



The screenshot shows the 'alpha TransForm' Designer interface. The top navigation bar includes 'Home', 'Designer' (active), 'Permissions', and 'Management Console'. Below the navigation, there are buttons for 'Save', 'Cancel', and 'Tester'. The main workspace is divided into two panes. The left pane, titled 'Form-Level On-Handlers', shows a list of form fields: 'Temp F' and 'Converted Temp'. The 'Temp F' field is selected, and its handler is 'tempF'. The right pane shows the TPL code for the selected handler, which is a function call: `#convertedTemp = @convertToCelsius(#tempF)`. The code is enclosed in `ON *changed_tempF` and `ENDON` tags. The interface also shows 'Editor' and 'Changed' buttons for each field.

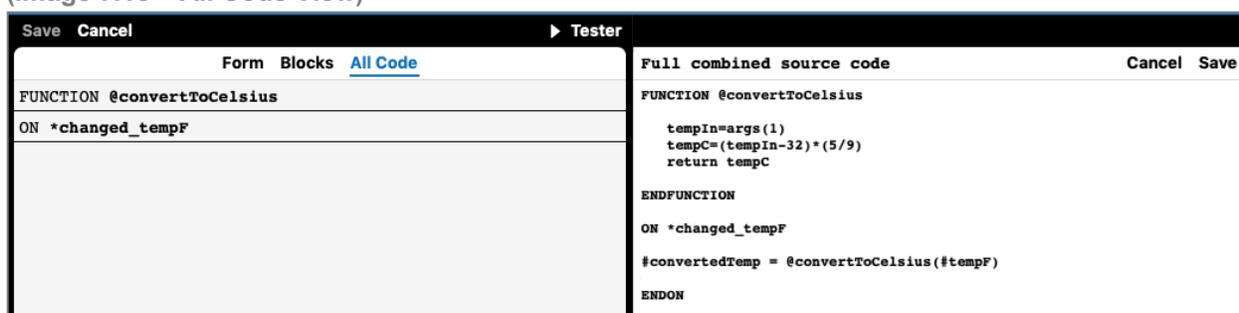
When you create functions, however, they are not associated with a particular field on the form. Instead, you click the `Blocks` link, as shown below:

(Image 7.17 – Blocks View)



In the Blocks view, your functions are listed in the left pane. To see or edit code for a particular function, click on it, and it appears in the center pane for editing. To create a new function, click the Add Function link. Alternatively, you can see all of your code at once in a single view by clicking the All Code link as shown in the following image.

(Image 7.18 – All Code View)



This view is handy if you prefer to work with your code in another editor, like VIM or ATOM, since you can copy and paste all of your code at once.

Debugging Your Applications

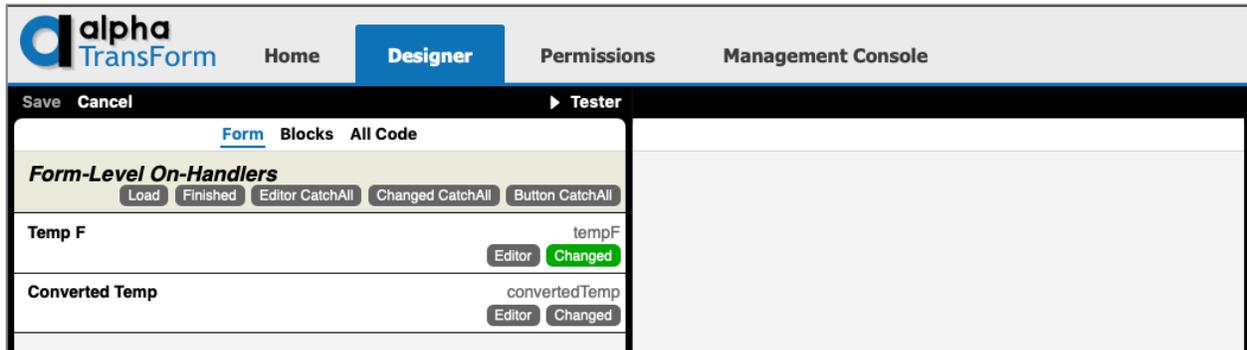
When your TPL code is causing errors to be displayed or is not operating the way you expect, the Tester and the Debugger can help you figure out why. The Tester is a tool built into TransForm Central that lets you experiment with your code before you load it onto a mobile device. The Debugger is a tool built into the mobile application that lets you experiment with your code while it is on a mobile device.

Both the Tester and the Debugger can be used to experiment with and fix your TPL code, and each have different benefits. The Tester is useful because you find errors and fix them all in one place. While the debugger shows how your code runs in a native environment, though you cannot make changes to the code.

Using the Tester: The Basics

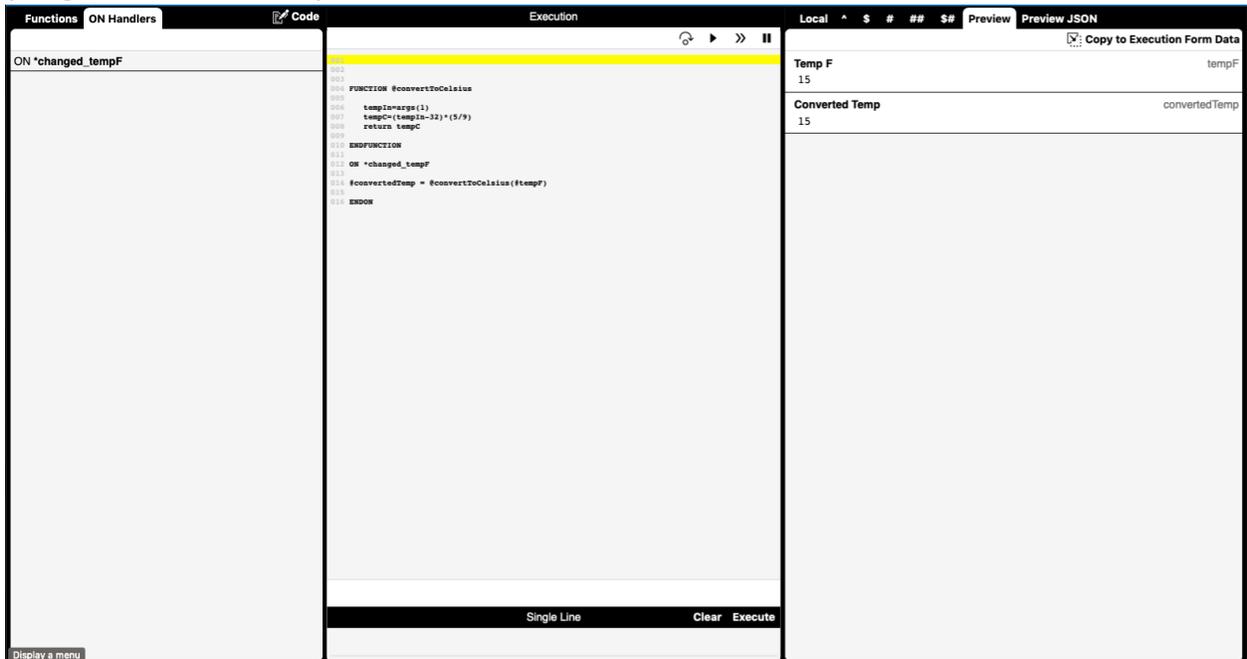
When you open the code editor, a link to the Tester appears in the top right corner of the left pane.

(Image 7.19 – Tester Link in Middle Towards the Top of the Screen)



Click the Tester link to switch from the Code Editor to the Tester. The Tester appears.

(Image 7.20 – TPL Tester)



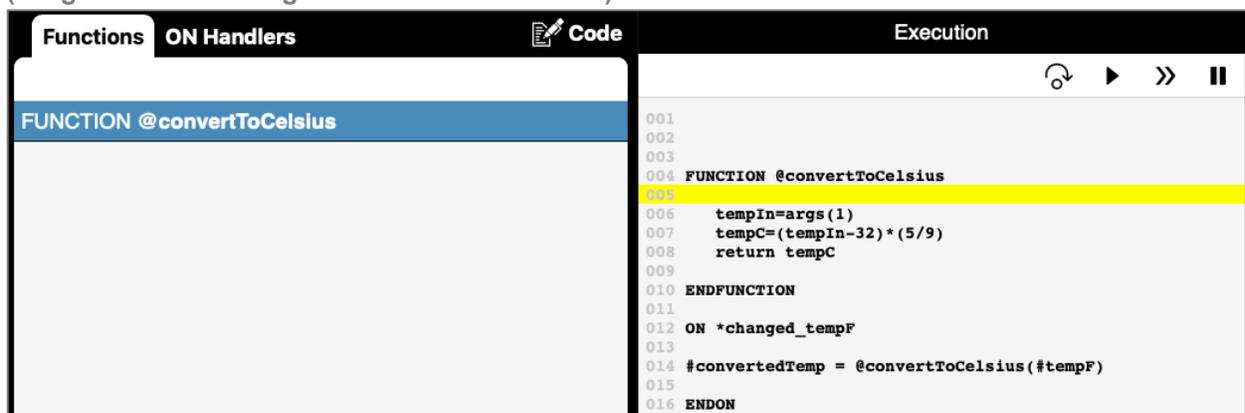
The Tester is divided into three panes. The pane on the left allows you to select a block of code to test. It is divided into two tabs, Functions and On Handlers. The Functions tab shows you a list of the custom functions you have created. The On Handlers tab

shows you a list of the events for which you have code defined. Clicking on a custom function or an event handler brings you to that section of code.

The middle pane displays all of the code. A yellow bar shows the next line to be executed. You can move the position of this bar by clicking on a function or event in the left pane.

In the following example, clicking on the function called `@convertToCelsius` moves the bar to the first line of the function.

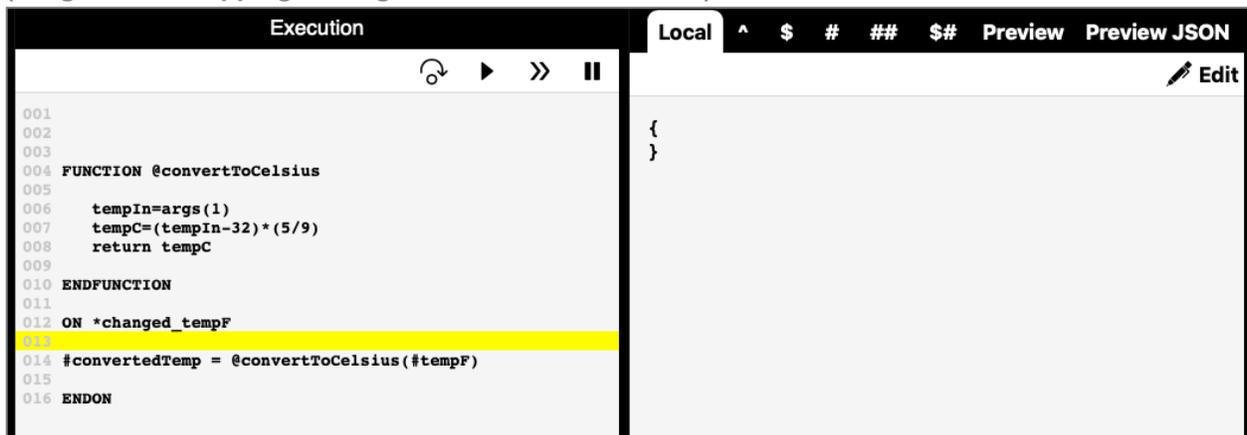
(Image 7.21 – Selecting a Function in the Tester)



Once the yellow bar is in position, you can run the code in one of three ways. Clicking the `>>` button runs the code at regular speed. Clicking the `>` button runs the code, but at a slower speed so you can watch as it executes (and click the pause button to stop it if necessary). And finally, clicking the `↺` button runs just one line of the script and then pauses until you click the button again. This allows you to step through the script.

While the code is running, you can see the values of the fields, variables, and system information in the pane on the right.

(Image 7.22 – Stepping Through TPL Code in the Tester)

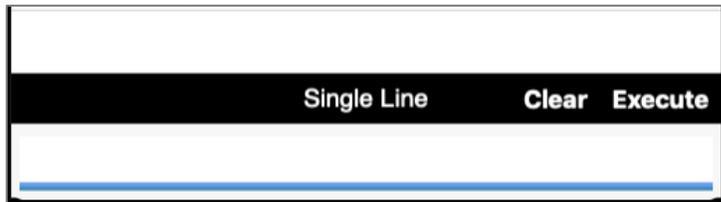


This information is divided into tabs as follows:

- Local – Shows you the values of variables you have set in TPL.
- ^ - Shows Global data. See the “Data Model section” in the TPL Reference Manual.
- \$ - Shows all of the form data and meta data
- # - Shows the field values
- ## - Also shows group data
- \$# - Shows metadata fields
- Preview – You can also use the Preview tab to copy test data into your script using the Preview values for each field.

Besides showing you code and data values, the Tester has one other feature that might be helpful: the ability to evaluate a TPL expression. This feature appears at the bottom of the middle pane and is especially helpful for testing out syntax before you use it in a script.

(7.23 – TPL Expression Tester)



The line you enter needs to evaluate to a value. So, for example, if you entered $2 + 2$ and clicked the Execute tab the result would be 4. If you entered `#tempF`, the result would be the value of the `tempF` field. In the example below, the `tempF` field is set to 212, and the expression below converts that value from Fahrenheit to Celsius.

(7.23 – TPL Expression Tester in Use)



Using the Tester: An Example

In this example, we will use the Tester to make sure that our code for converting Fahrenheit to Celsius is working properly. This is based on the example found in “Creating Functions,” earlier in this chapter. To check if the code is working, first we click the Tester button and select the `On changed_TempF` event to move the yellow bar to the beginning of the event code.

(Image 7.24 – Stepping Through a Function in the Tester)



The next line of code to be executed calls a function and passes in the value in the tempF field. But, because this code is running in the Tester (and not on a real device), we have not yet typed anything into that field. To make the test more realistic, we should populate that field with a value before we call the function. We can do that in the right pane.

We'll begin by clicking the # tab to see the field values for our form. Currently they are blank.

(Image 7.25 – Examining Field Values in the Tester)



To set the value of the field, we'll click the Edit button and enter a value using JSON notation, as shown in the next image. Note, you do not need a # in front of the fieldname when setting field values.

(Image 7.26 – Setting a Field Value in the Tester)



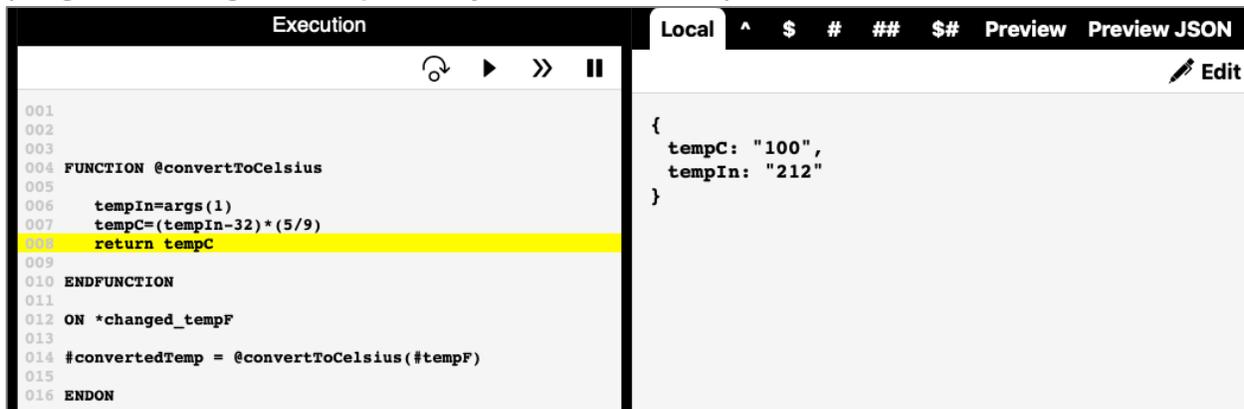
Now that the value of tempF is set, we can click the Step Button  twice. This calls the convertToCelsius() function and moves the yellow bar up into the convertToCelsius() function as shown below.

(image 7.27 – Watching Field Values While Stepping Through Code)



Next we click the step button 3 more times in the middle pane and click the Local tab in the right pane.

(Image 7.28 – Image Value Updated by Code in the Tester)



The local tab now shows the value of the two variables. The `tempIn` variable was created in the `convertToCelsius()` function and it stores the value that was passed into the function from the `#tempF` field. The `tempC` variable was created next, and it now contains the calculated Celsius value based on `tempIn` – thus confirming that the code is working correctly.

Using the Debugger

The debugger is an on-device tool that lets you monitor your TPL code to help you diagnose errors or unexpected behaviors. The benefit of the debugger over the tester is that you can see what is happening on the device itself as opposed to in a simulated

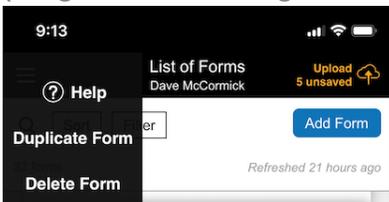
environment. It also allows you to test with data and functions that are not available in the Tester, such as live GPS data, the `inAppBrowser()`, `fileViewer()`, and AJAX functions.

In order to access the debugger, you need to have sufficient permissions. By default, the only roles allowed to access the debugger are Administrators and Developers (though this can be changed on the Permissions tab in TransForm Central to allow other roles).

To access the debugger:

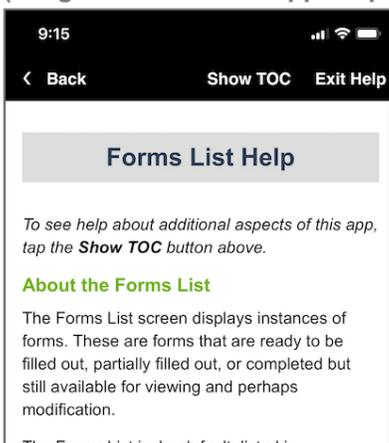
1. Open the TransForm Mobile App and log in, if necessary. Once you have logged in, tap the Hamburger icon to open the menu.

(Image 7.29 – Hamburger Menu in Mobile App)



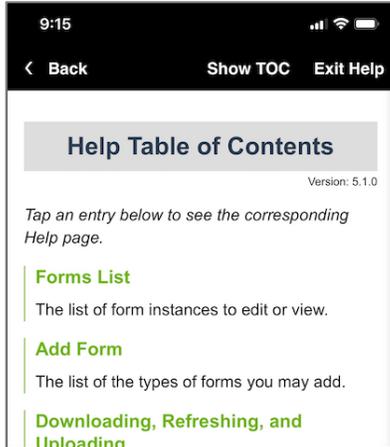
2. From the menu, choose Help to open the Help system.

(Image 7.30 – Mobile App Help System)



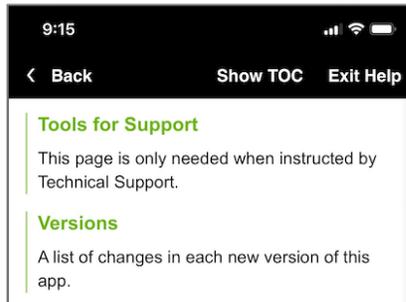
3. At the top of the help system, tap the Show TOC link to display the table of contents.

(Image 7.31 – Mobile App Help System Table of Contents)



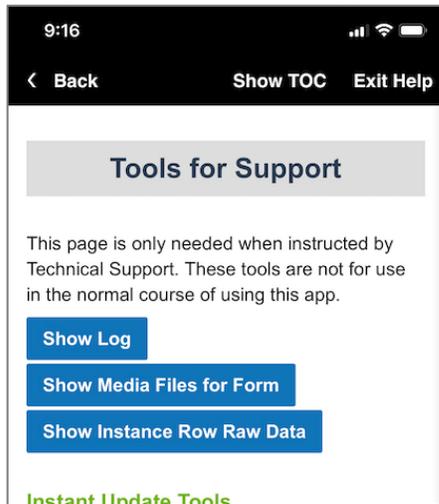
4. Scroll down through the table of contents until you find Tools for Support.

(Image 7.32 – Tools for Support Link)



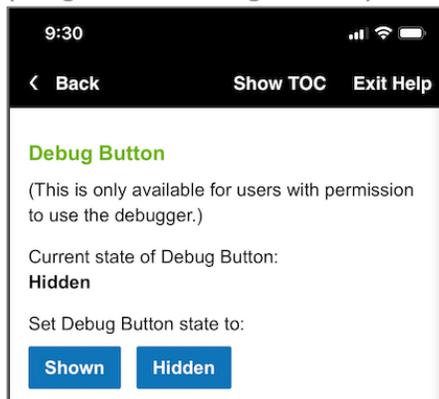
5. Tap the Tools for Support link to open the Tools for Support section.

(Image 7.33 – Tools for Support Section)



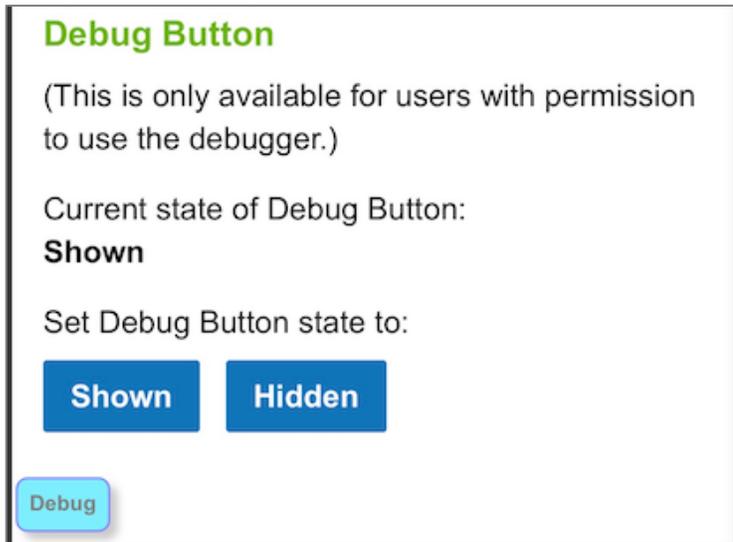
6. Scroll down the Tools for Support page until you come to the Debug Button section.

(Image 7.34 – Debug Section)



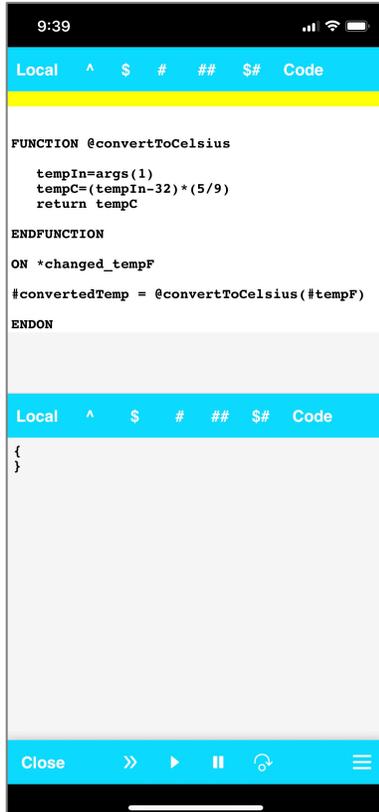
7. Tap the Shown button to show the Debug button on the screen. A small Debug button appears and remains visible throughout the application until you return here and click the Hidden button.

(Image 7.35 – Debug Button)



Now that the Debug button is active, you can activate the debugger whenever you need it by tapping the button. However, it really only makes sense to activate the debugger when you are viewing a form, as shown in the following image.

(Image 7.36 – Debugger)



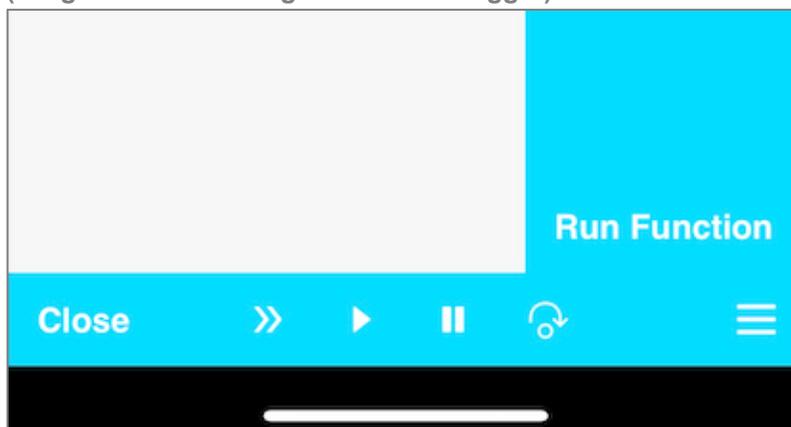
The debugger works very much like the Tester in Transform Central, described earlier in this chapter. You can select a function or event code to run. You can run code slowly and pause it. And, you can step through it line by line. You can also examine the value of user fields, systems fields, and variables to determine if your code is working as it should.

The debugger is divided into two panes, one above the other. That way you can watch as code executes in one pane and see the resulting changes to fields and variables in the other pane.

To Test Code in the Debugger:

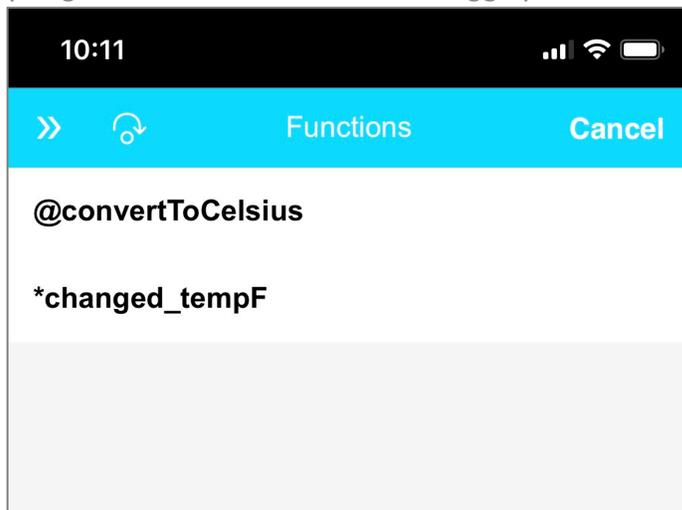
1. Tap the Hamburger icon . The Run Function menu option appears.

(Image 7.37 – Hamburger Menu in Debugger)



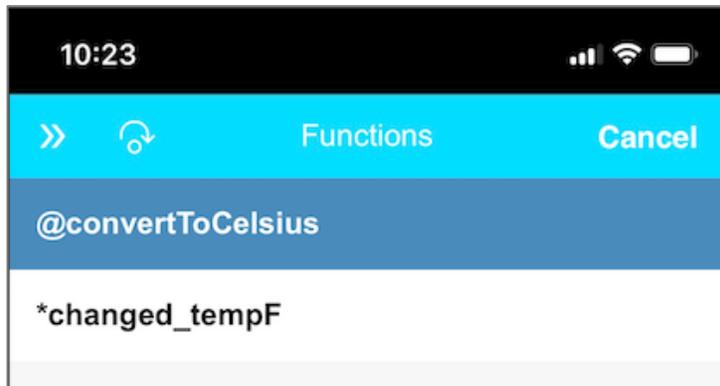
2. Tap Run Function to open the functions list.

(Image 7.38 – Functions List in Debugger)



3. Tap on the block of code you would like to examine to select it.
4. Tap either the >> link to execute the code and close the debugger or tap the Step button  to open the code in the debugger.

(Image 7.39 – Debugger with Function Selected)



Once in the debugger, the controls for running the code appear at the bottom. The controls for executing the code are the same as in the Tester. The >> button runs the code at regular speed, the > control runs the code slowly so that it can be paused with the || button, and the Step Control  steps you to the next line.

(Image 7.40 – Debugger Stepping Through Code)



Also, like the Tester, you can see the variables, field values, and other data in the form by tapping on one of the following options:

- Local – Shows you the values of variables you have set in TPL.
- ^ - Shows Global data. See the “Data Model” section in the online documentation.
- \$ - Shows all of the form data and meta data
- # - Shows the field values
- ## - Also shows group data
- \$# - Shows metadata fields
- Code – shows Code view instead of data

For more information, refer to the “Data Model” section in the online documentation.

How to Load, Refresh and Delete On-Device Assets

Earlier in this chapter, you were shown how to perform a lookup from a local SQL database, how to launch a video, and how to open a PDF manual – all while the device was offline. In order for these examples to work, the needed files must be preloaded onto the device. This is done in TransForm Central.

Here's how that works.

1. The required files are placed on a server, so they are accessible online. You can use Amazon S3, a web hosting account, a service like Dropbox, or any server that lets you access the files via URL.
2. A Manifest file is created (in JSON) format, which lists all of the files you want to load onto the device. The manifest file is then added to TransForm Central.

3. Another file, called the “Policy” file is also created. It defines under what circumstances the assets are loaded onto the device.
4. When the user logs in with the mobile application, the app refers to the manifest and policy files, and then downloads the required assets onto the device.

Creating the manifest file by hand can be tedious, but if you use Alpha Anywhere (the low code development platform from Alpha Software), you can use the built-in On-device Data Builder to do much of the work for you. See the next section.

On the other hand, if you would like to create manifest files and policy files yourself, details on how the file should be structured can be found in the online documentation at this link:

<https://documentation.alphasoftware.com/TransFormDocumentation/index?search=ondevice%20assets%20policy>

Using the Alpha Anywhere On-Device Data Builder

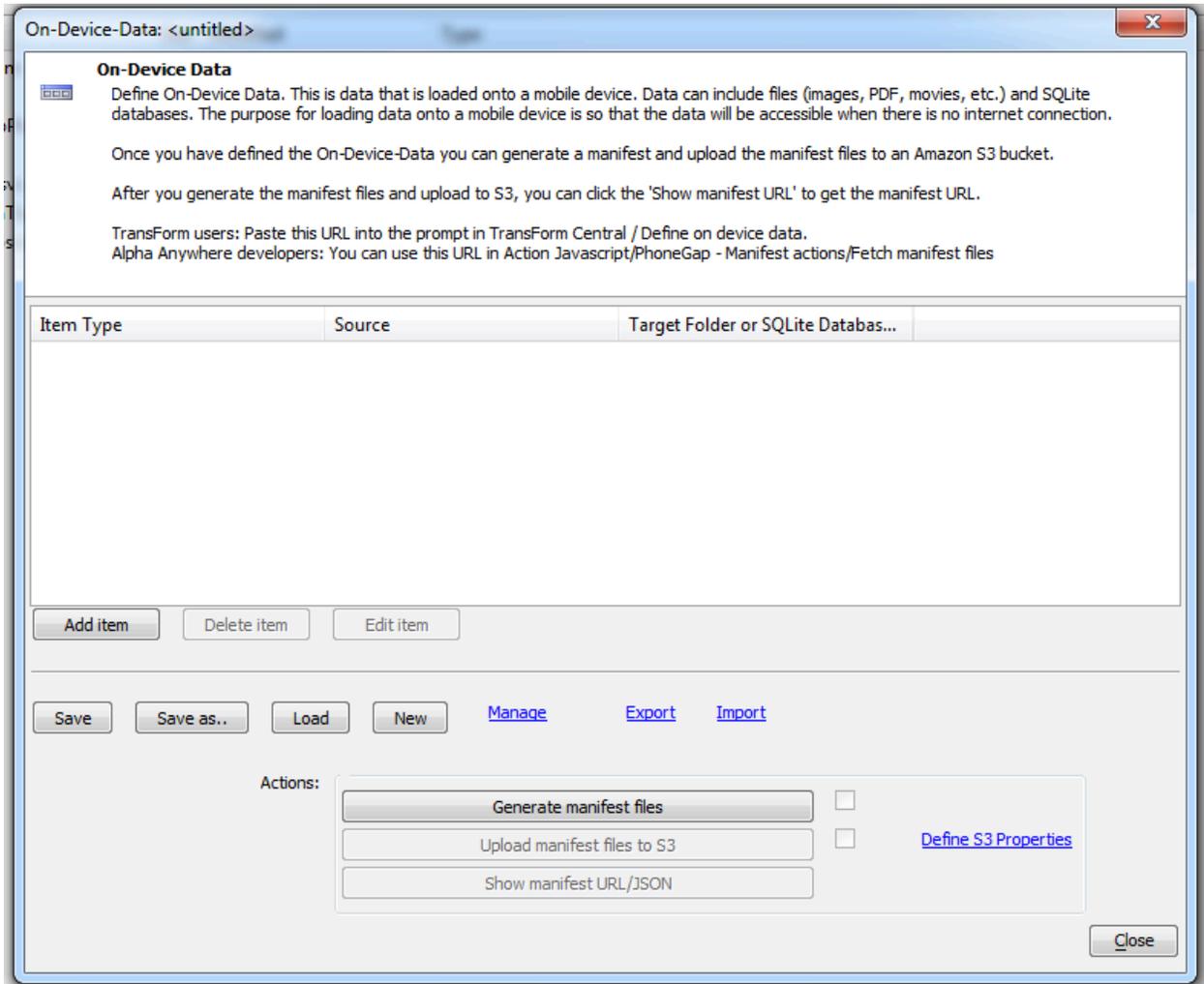
Alpha Anywhere is Alpha Software’s low-code development platform for building and deploying web and mobile applications. Part of the Alpha Anywhere system is a Windows-based development tool (IDE) called the Alpha Anywhere Developer, and you can use the Alpha Anywhere Developer to simplify loading on-device assets, as well as for performing other tasks in TransForm.

You can download Alpha Anywhere here: <https://www.alphasoftware.com/get-started-2018>

To Load On-Device Assets into TransForm using Alpha Anywhere:

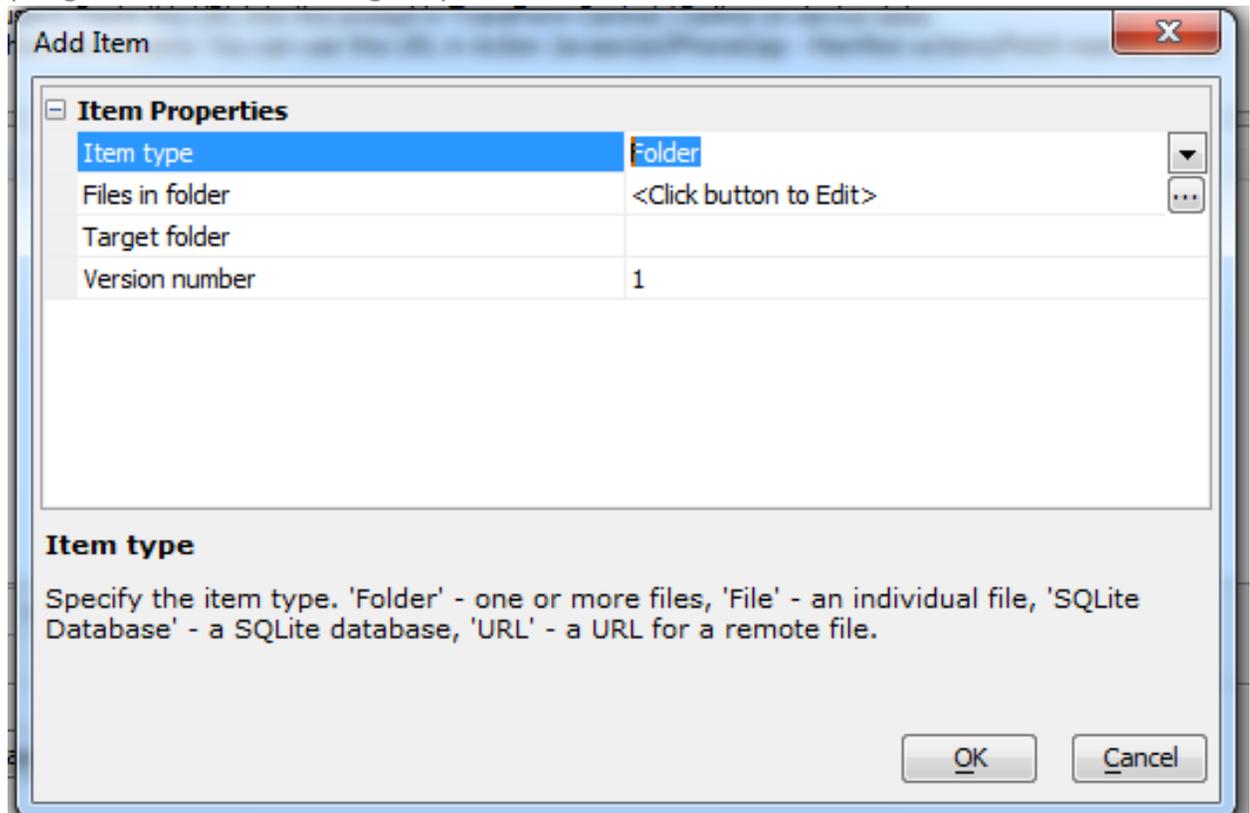
1. Open Alpha Anywhere and create a new workspace or open an existing workspace.
2. Click the Web Projects  button to open the Web Projects Control Panel (if it is not already open).
3. From the Tools menu, choose: More... > TransForm Utilities > On-device Data Builder for TransForm Applications. The On-Device Data dialog box opens.

(Image 7.41 – Alpha Anywhere On-Device Data Builder)



4. Click the Add Item button to add a new file, folder, or database. The Add Item dialog box appears.

(Image 7.42 – Add Item Dialog Box)



5. From the Item Type drop-down box, choose one of the following:

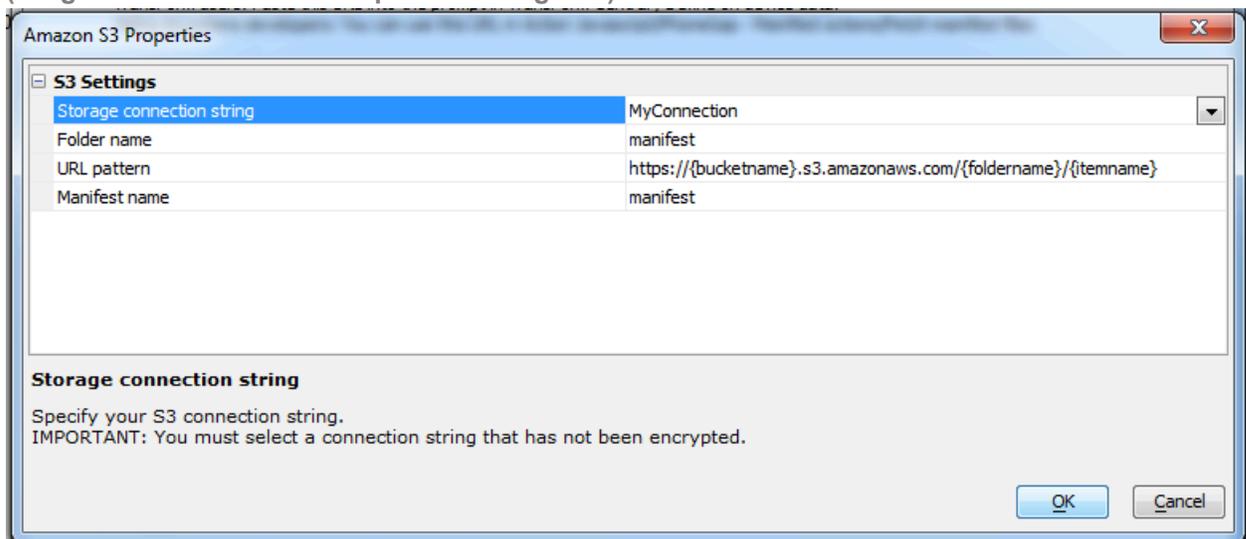
- Folder – Allows you to select multiple files at once from a local drive.
- File – Allows you to select an individual item from a local drive. If you choose this option, you can assign a version number to the file, which Transform uses to determine if the file on the device needs to be updated.
- SQLite Database – Selects an existing SQLite database or allows you to create a new one. For instructions on creating a new one, see the section “Creating a SQLite Database,” later in this chapter.
- URL – Allows you to specify the location of a file on the internet (rather than on your hard drive).

- Specify the Target Folder where you would like these assets to be saved on to the mobile device. (This does not apply if you selected a SQLite database, which only works when stored in the root – so there is no option for the Target folder.)
- Repeat these steps for all of the items that you would like to add.
- Click the Save button and give this list of assets a name. This allows you to return to this dialog box, without having to select all of your files again.
- Click the Define S3 Storage Properties and choose your S3 Connection string.

If you do not have a storage connection string defined yet, close the Amazon S3 Properties dialog box, then close the On-Device Data dialog box. Refer to the next section, “Setting up an S3 Storage Bucket.” When you are done, return to this step.

- In the Amazon S3 Properties dialog box, choose your Storage Connection String.
- Enter the name of the folder and file under which you want to save the manifest.

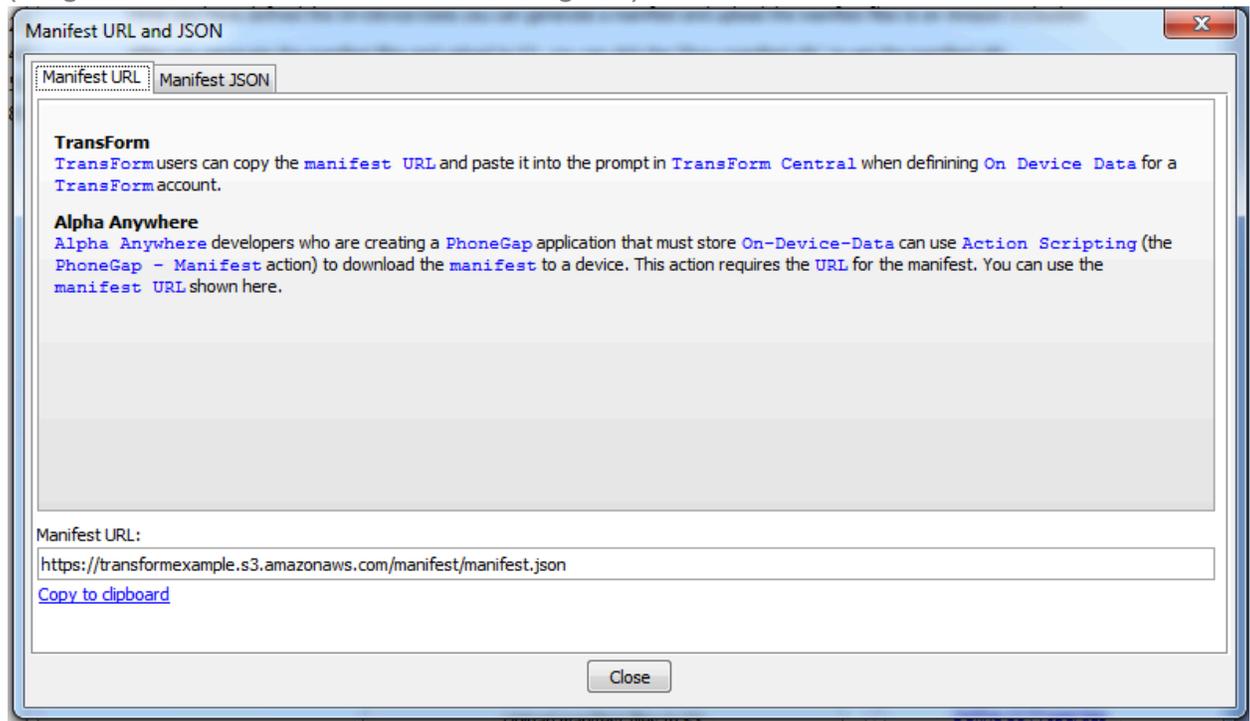
(Image 7.43 – Amazon S3 Properties Dialog Box)



- Click OK to close the Amazon S3 Properties dialog box.

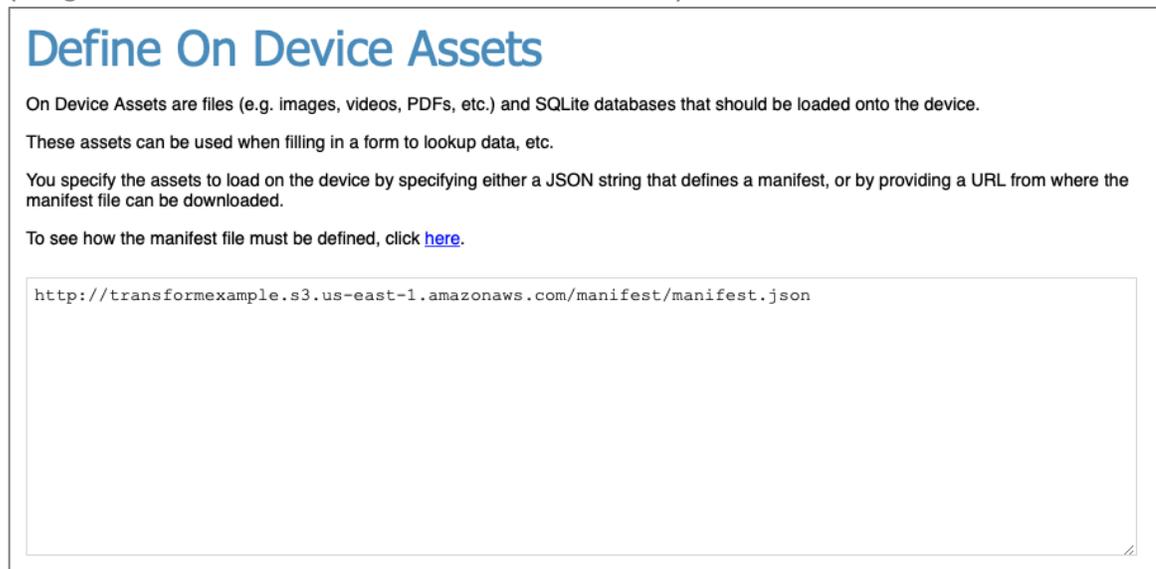
13. Click the Generate Manifest Files button to create the manifest file. A checkmark will appear next to the button when this is complete.
14. Click the Upload Manifest Files to S3 button. This will upload the manifest files and any locally stored assets to your S3 bucket.
15. Click the Show Manifest URL/JSON button. The Manifest URL and JSON dialog box appears.

(Image 7.44 – Manifest URL and JSON Dialog Box)



16. Copy the Manifest URL into the clipboard.
17. Paste the Manifest URL into the Manifest box in TransForm Central and click Save.

(Image 7.45 – Manifest File Box in TransForm Central)



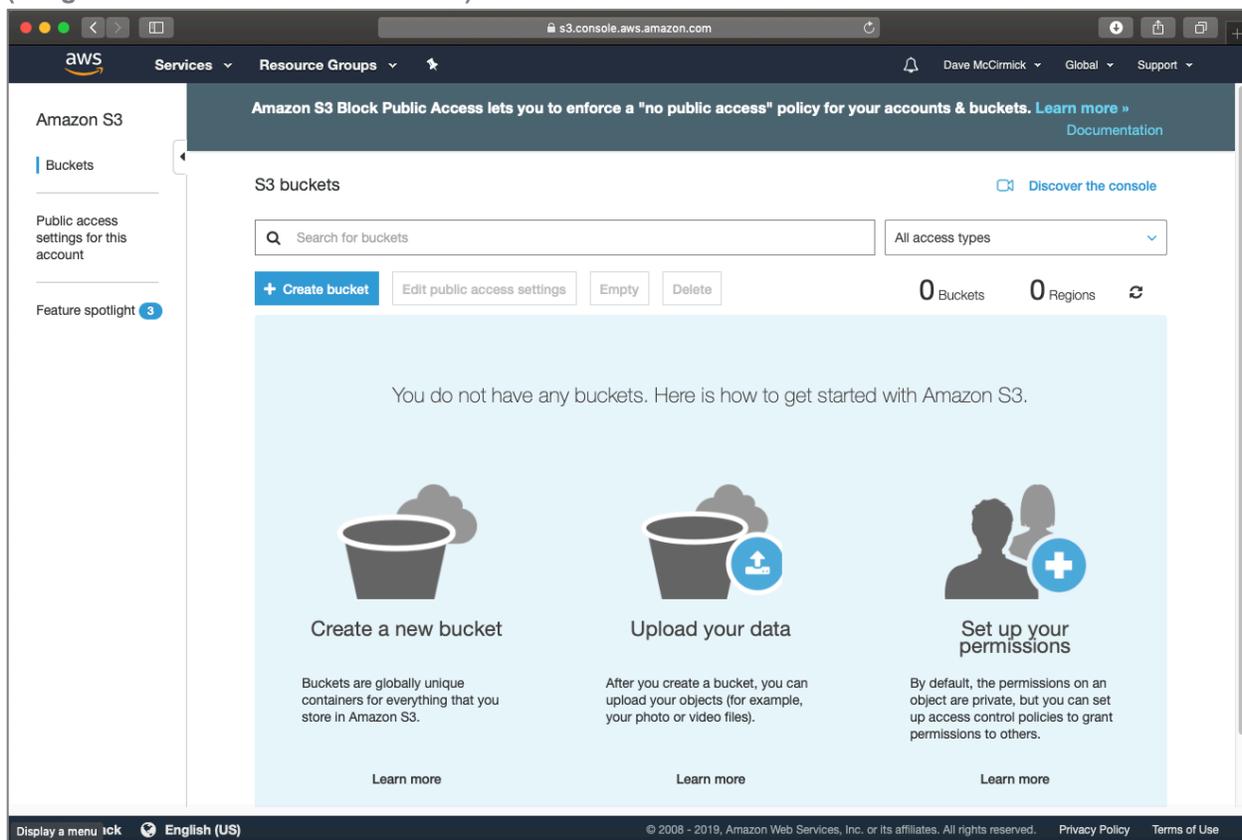
There are two ways of entering a manifest file. You can either paste in the JSON data directly. Or you can paste in a URL that points to a file containing the JSON data. In this example, we have done the latter.

NOTE: While the manifest box indicates which files to download, it does not indicate when they should be downloaded. That is controlled in the Policy box (which is below the Manifest box in TransForm Central). **By default, no files are downloaded unless you set a policy.** For information on how to set policies so that assets are downloaded, see “Setting the Download Policy for Assets,” later in this chapter.

Setting up an S3 Storage Bucket

Amazon S3 is a web service that provides file hosting. When defining your On-Device assets, S3 is used to store the assets themselves and the manifest file that TransForm needs to load the assets. In order to use S3, you first need to set up an AWS account and create an S3 bucket. You do this at: <http://aws.amazon.com/>.

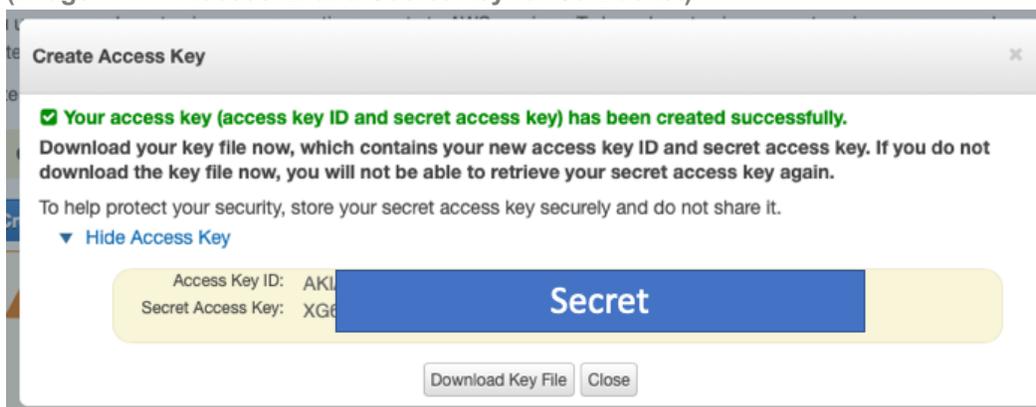
(Image 7.46 – Amazon AWS Console)



Once you have set up an AWS account, you then need to create an S3 bucket. While doing this, the default settings will work for your bucket, however, you should make sure that Public Blocking for your bucket is turned off. (By default, it is turned on)

Once the bucket is created, AWS will provide you with an Access Key ID and a Secret Access Key.

(Image 7.47 – Access ID and Secret Key for S3 Bucket)

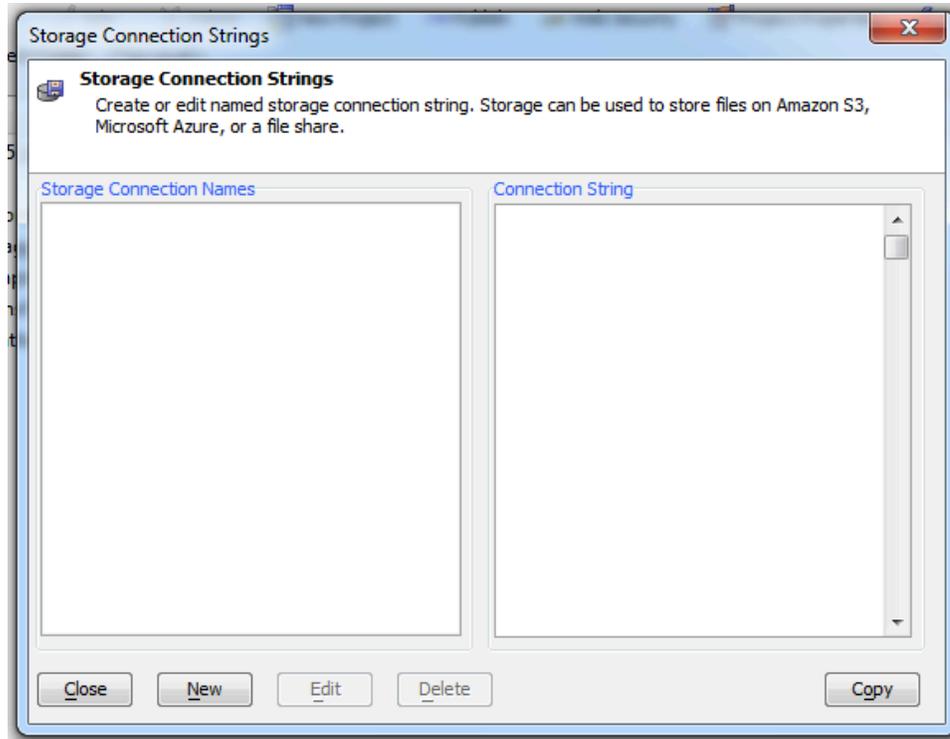


Copy these down and keep them in a safe place. You will need these to create the storage string in Alpha Anywhere.

To create a storage string in Alpha Anywhere:

1. In Alpha Anywhere, open the Web Projects Control Panel.
2. From the Tools menu, choose Storage Connection Strings. The Storage Connection Strings dialog box appears.

(Image 7.48 – Alpha Anywhere Storage Connection Strings Dialog Box)



3. Click New. The Add New Storage Connection String dialog box appears.

(Image 7.49 – Add New Storage Connection String Dialog Box)



4. Type a name into the Connection Name box. Then click the Build button. The Storage Connection String Builder appears.

(Image 7.50 – Storage Connection String Builder)

Storage Connection String Builder

Storage Provider
Azure

Use Test Storage

Region
US East (Virginia)

Account

Access Key

Show Access Key

Container Name

Triple-DES Object Encryption Key (optional):

Encrypt Connection String
Encryption Passphrase

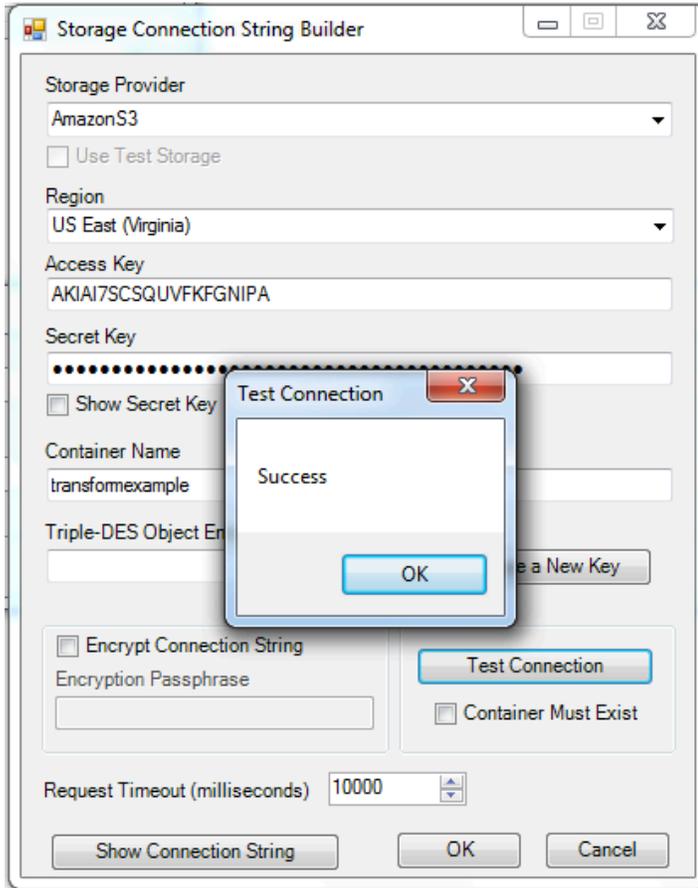
Container Must Exist

Request Timeout (milliseconds) 10000

5. From the Storage Provider drop-down, choose Amazon S3.
6. Change the Region if necessary. (It needs to match the region you selected when you set up the bucket.)
7. Enter in the Access Key and the Secret Key.

8. Type the name of the bucket into the Container Name box.
9. Click Test Connection to verify that you have entered the settings correctly. A dialog box should appear and indicate that the connection was successful.

(Image 7.51 – Storage Connection String Test Dialog Box)



Click OK to close the Test Connection Box, then click OK again to close the builder, and finally click OK once more followed by clicking the Close button to save the Storage Connection String.

You can now use this Storage Connection String in the On-Device Data Builder, as detailed above in the section “Using the Alpha Anywhere On-Device Data Builder.”

Setting the Download Policy for Assets

TransForm uses a Policy file to determine when the on-device files should be downloaded. This is done by entering JSON into the Policy box in TransForm Central, which is the bottom box on the Define On-Device Assets section.

(Image 7.52 – Policy File Box in TransForm Central)

Define On Device Assets

On Device Assets are files (e.g. images, videos, PDFs, etc.) and SQLite databases that should be loaded onto the device.

These assets can be used when filling in a form to lookup data, etc.

You specify the assets to load on the device by specifying either a JSON string that defines a manifest, or by providing a URL from where the manifest file can be downloaded.

To see how the manifest file must be defined, click [here](#).

```
http://transformexample.s3.us-east-1.amazonaws.com/manifest/manifest.json
```

You can also define (optional) options for the On Device Assets that control how often the On Device Assets are refreshed.

```
{
  "policy" : {
    "download" : {
      "onDemand" : {"checkprompt":false,"confirmprompt":false,"successprompt":true},
      "onLogin" : {"checkprompt":false,"confirmprompt":false,"successprompt":true}
    }
  }
}
```

Save

NOTE: Without a Policy file, assets will not be downloaded.

The Policy file controls when files are downloaded automatically. There are two options: at login and/or when the user chooses Update On-Device Data from the hamburger menu in the mobile app. You can also control when (or if) message boxes appear during the process.

Below is an example policy that you can use in your own account:

```
{
  "policy" : {
    "download" : {
      "onDemand": {"checkprompt":false,"confirmprompt":false,"successprompt":true},
      "onLogin" : {"checkprompt":false,"confirmprompt":false,"successprompt":true}
    }
  }
}
```

This policy tells Alpha Anywhere to download files to the device both when the user logs in (onLogin) and also when the user requests the data be updated from the menu (onDemand). In either case, a message box is displayed after the files have been retrieved.

The Policy file has other options as well, including some which are not related to downloaded assets. For example, the Policy file can control how forms are searched, as well as other behaviors.

In many instances, the above JSON is all you need. Just copy and paste it into your account. But for an in-depth discussion about the options, refer to the online documentation:

<https://documentation.alphasoftware.com/TransFormDocumentation/index?search=ondevice%20assets%20policy>

Whitelisting Files and URLs

Along with password protection, encryption, and other methods, one of the ways Alpha TransForm helps keep your data safe is by whitelisting folders, URLs, and data prefixes you use in your forms. A whitelist provides a master list of which resources the mobile application is allowed to use.

TransForm has two kinds of whitelists:

- Pathprefixes - used to whitelist images
- Ajaxprefixes – used to whitelist URLs in AJAX callbacks.

Pathprefixes for images are mandatory, while ajaxprefixes for AJAX callbacks are optional, unless you specify otherwise.

Whitelisting Image URLs and Folders

When you reference images using the tag, TransForm requires that you first specify the path of the image (whether it's in a local folder on the device or a remote file on a server) in the Policy file.

You can specify these paths/URLs using “pathprefixes”. This is shown in the following example:

```
"pathprefixes" : {  
  "mysite" : "https://www.mysite.com/images/",  
  "images" : "#products/"  
}
```

There are a few things to notice. First, this group of data starts with “pathprefixes”. Next, you’ll notice how each line consists of a pair of values. The first value is the alias. This is what you will use in the tag. The second is the location of the resource itself. In this case there is a URL and a local directory. Note that the directory name is prefixed with a “#” to indicate it is a local folder and not a URL.

```
Image from a website: <br/>  
Image from On-device Assets: 
```

In the top line, the “mysite.” alias is automatically swapped out with “https://www.mysite.com/images/”, since that was how it was defined in the pathprefixes section of the Policy box. Likewise, the “images.” prefix is automatically swapped out with “products/”. (The “#” is not needed in the image tag, just in the pathprefix section.)

Whitelisting URLs Used in AJAX (API) Calls

In the previous example, we created an alias to a URL by including the alias name and the URL in the pathprefixes section. We then used that alias in an tag to display an image. The concept is the same for URLs that you want to call in an AJAX callback.

Instead of the pathprefixes section, however, these URLs are placed in the ajaxprefixes section, as shown in the example code below:

```
"ajaxprefixes" : {
  "mySite" : "https://www.mysite.com/",
  "staticMap" :
  "https://maps.googleapis.com/maps/api/staticmap?key=YOUR_API_KEY",
  "currentWeather" :
  "https://api.openweathermap.org/data/2.5/weather?appid=YOUR_API_KEY"
}
```

In this example, three aliases were created: mySite, staticMap, and currentWeather. Each of these aliases point to different URLs. When the TPL performs an AJAX callback, using the ajaxSendRequest() or the ajaxGet() functions, the alias is used instead of the URL, like in the TPL code snippet below:

```
*ON button_getWeather
  lat_lon = "lat=" & #latitude & "&lon=" & #longitude
  weather = ajaxGET("currentWeather." & lat_lon)
ENDON
```

When this code runs, the “currentWeather.” alias is swapped out with the URL defined in the ajaxprefixes portion of the Policy file.

NOTE: If you do not include an ajaxprefixes section in your Policy file, TransForm assumes all URLs are allowed.

For more details on whitelisting, refer to the online documentation:

<https://documentation.alphasoftware.com/TransFormDocumentation/index?search=ondevice%20assets%20policy>

Creating a SQLite Database

In previous sections you have seen how to use a SQLite database with TPL code. You have also seen how to load on-device assets. In this section, you’ll see how to create a SQLite database that you can load and use.

In this example we are going to connect to an existing MySQL database and use that database to create a duplicate version as a SQLite database. Even though this example

uses MySQL, the instructions are the same for other data sources as well, including: SQL Server, Oracle, Excel, and many more.

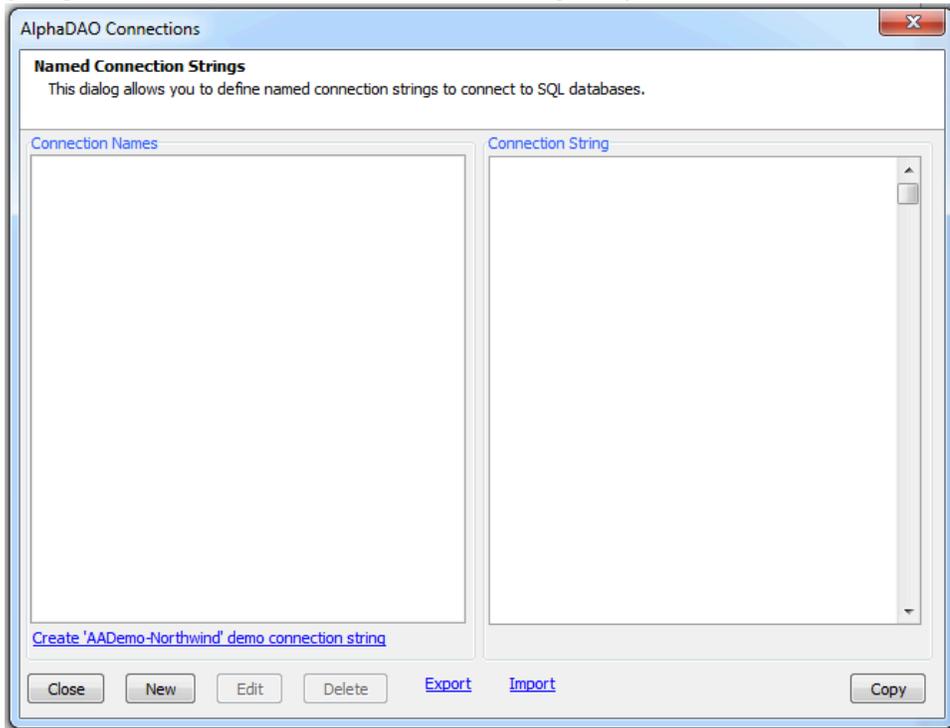
Once the SQLite database is created, we'll post it onto an S3 Bucket and update the manifest file to include the new database. And, as in the previous example, we will use Alpha Anywhere since it has all of the tools we need built in.

The instructions for doing this are covered in the next two sections. The first section shows you how to build a connection string to a MySQL (or any other) data source. The second section shows you to use that connection string to create a duplicate, SQLite version of the database.

Building a Database Connection String

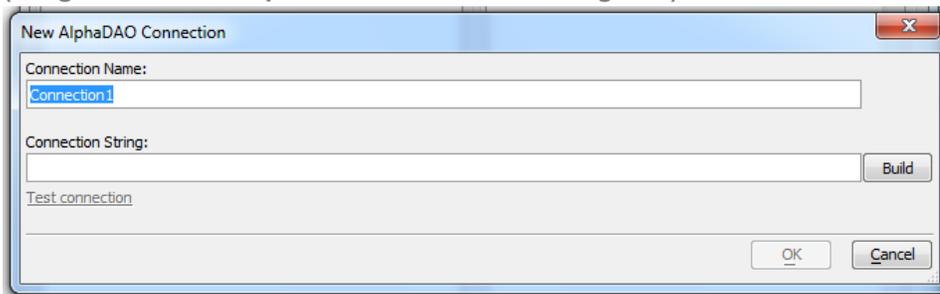
1. Open a workspace in Alpha Anywhere and go to the Web Projects Control Panel.
2. From the Tools menu, choose AlphaDAO Connection Strings. The AlphaDAO Connections box appears.

(Image 7.53 – AlphaDAO Connections Dialog Box)



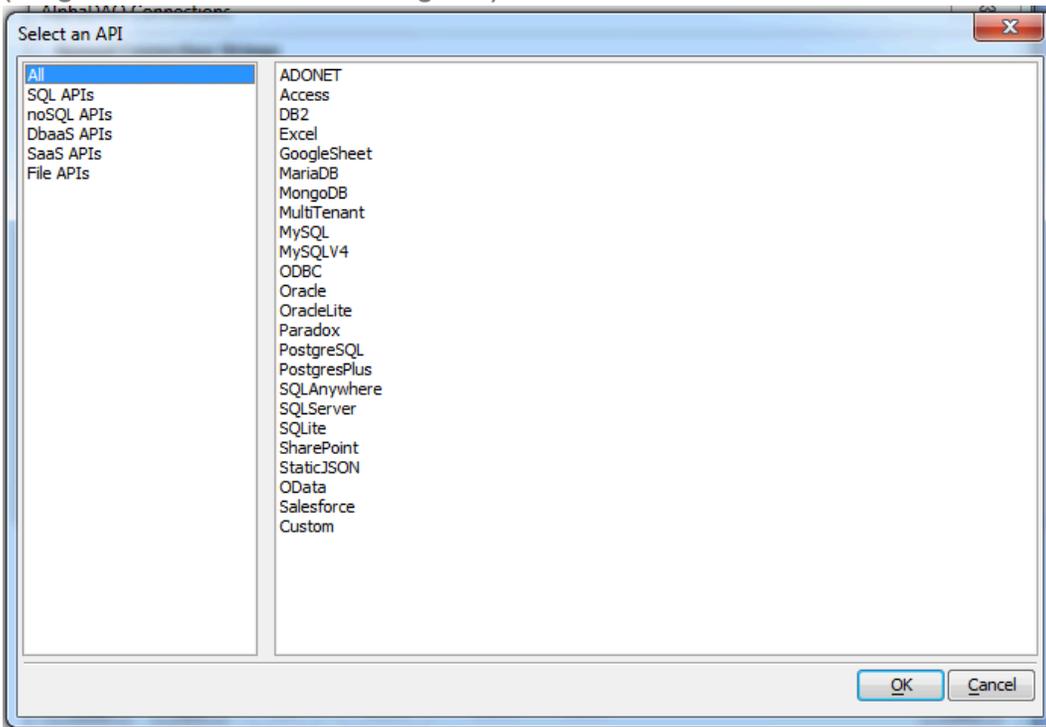
3. Click the New button. The New AlphaDAO Connection dialog box appears.

(Image 7.54 – New AlphaDAO Connection Dialog Box)



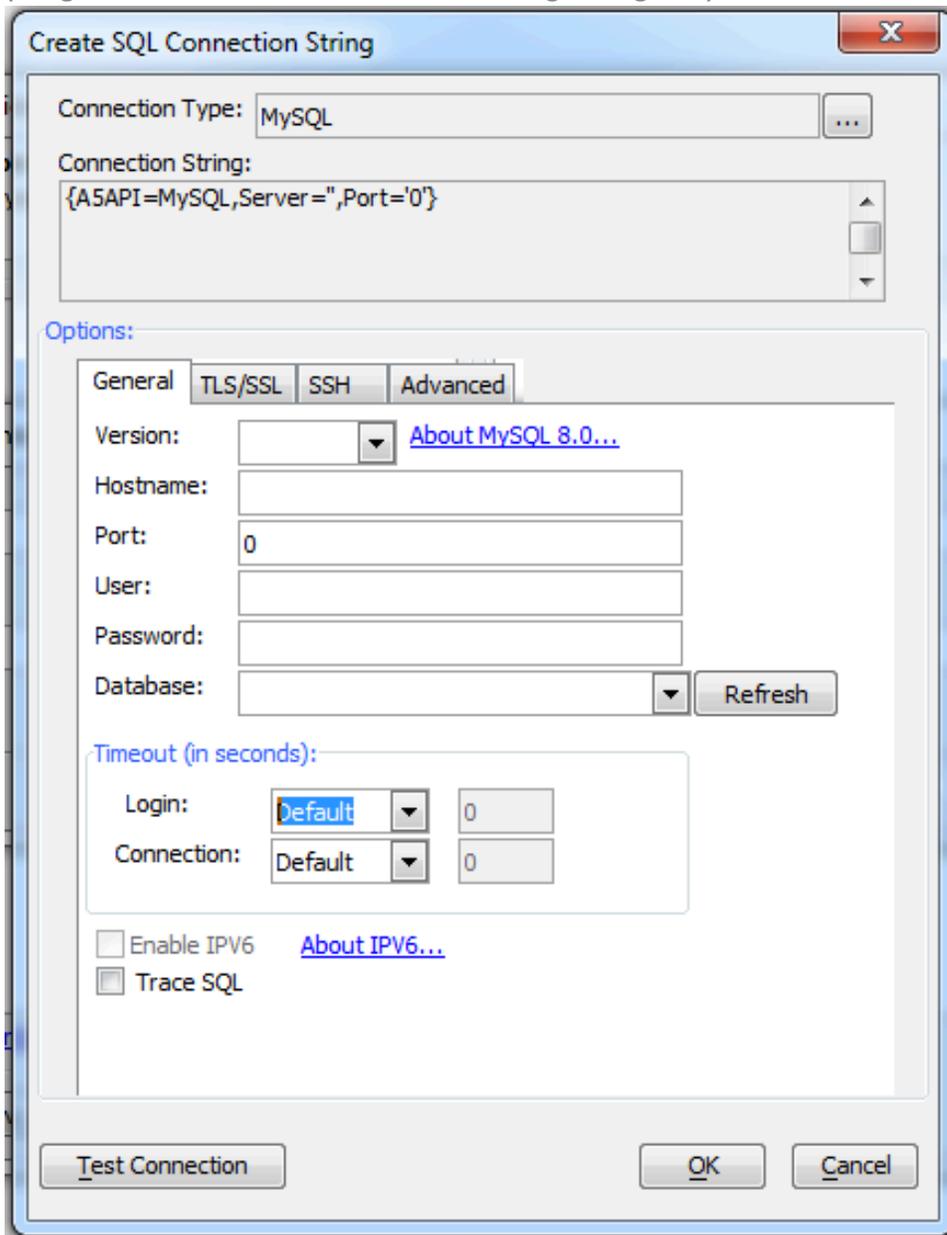
4. Click the Build button. The Select an API dialog box appears.

(Image 7.55 – Select an API Dialog Box)



5. Choose the type of data source you would like to use and click OK. (In this example, we will choose MySQL). The Create Connection String dialog box appears.

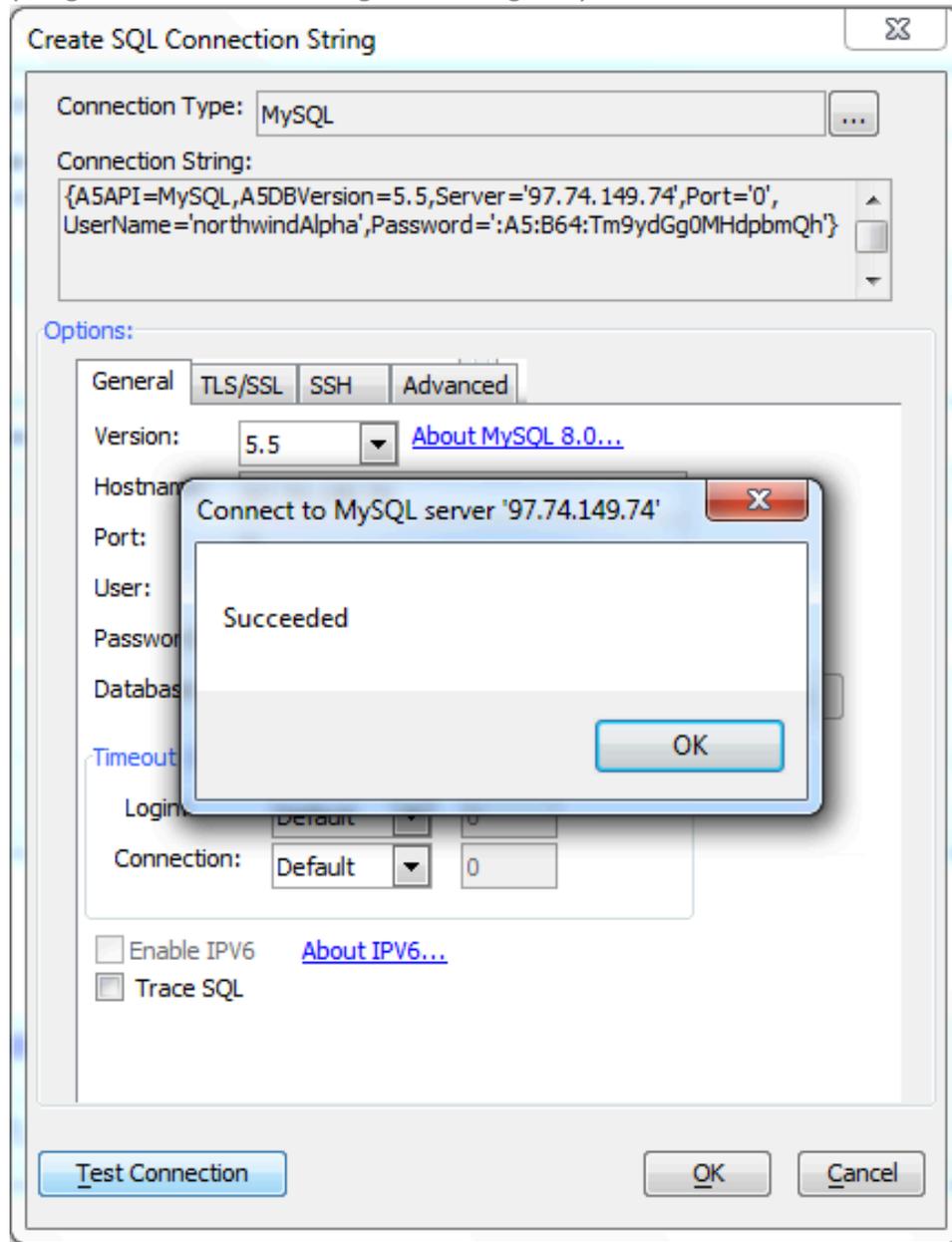
(Image 7.56 – Create SQL Connection String Dialog Box)



Because we selected MySQL, we are prompted for information needed to connect to a MySQL database. These options would be different if we selected a different data source. For example, if we chose Excel, we would be prompted for the location of the Excel file.

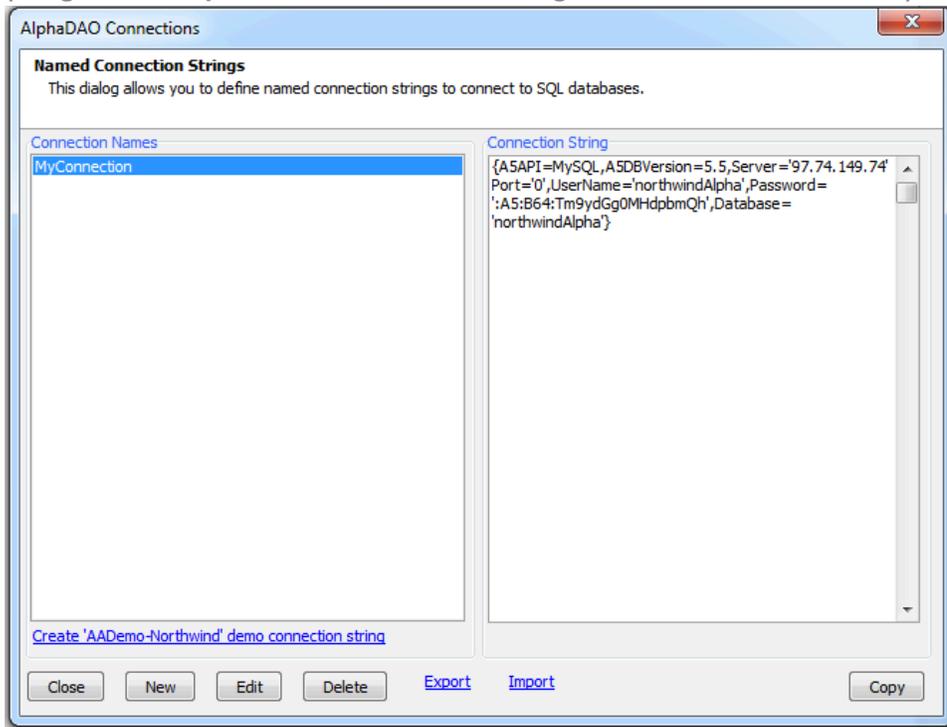
6. Fill in the connection information as needed. Then click the Test Connection button. A dialog box should appear indicating success. If not, check your settings and try again.

(Image 7.57 – Connect String Test Dialog Box)



7. Click OK to close the Success box. Then Click OK again to return to the New AlphaDAO connection dialog box. Then click OK again to return to the AlphaDAO connections box.

(Image 7.58 – AlphaDAO Connections Dialog Box With New Connection)



8. Make a note of your connection name, then click Close to close the dialog box.

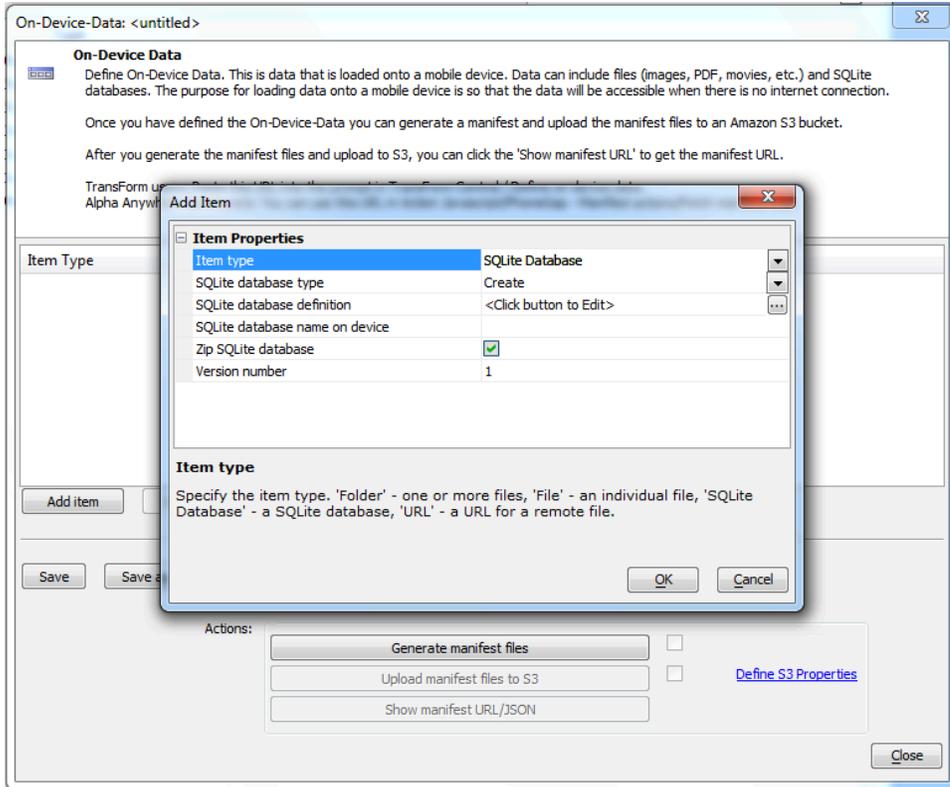
To Create a New SQLite Database in Alpha Anywhere

Once you have a connection string, you can use the On-Device Data Builder to duplicate the database, so that you have a SQLite version that can be stored on a mobile device.

1. Open a workspace in Alpha Anywhere and go to the Web Projects Control Panel.
2. From the Tools menu, choose More... > Transform Utilities > On-Device Data Builder for Transform Applications. The On-Device Data Builder appears.
3. If you already have On-Device Data settings that you would like to load, do that now by clicking the Load Button.
4. Click the Add Item button to add a new item.
5. Select SQLite Database as the Item type. New options appear in the dialog box relating the SQLite database you would like to add. (By default, the SQLite

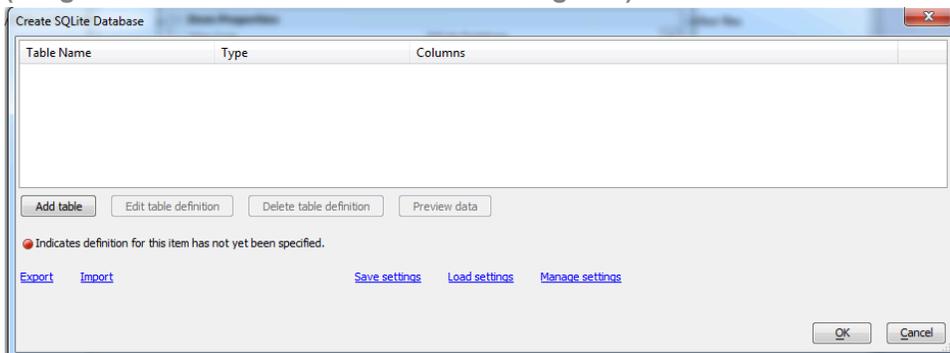
Database Type is set to Create, indicating that you want to create a new database.)

(Image 7.59 – Add Item Dialog Box, SQLite Database Selected)



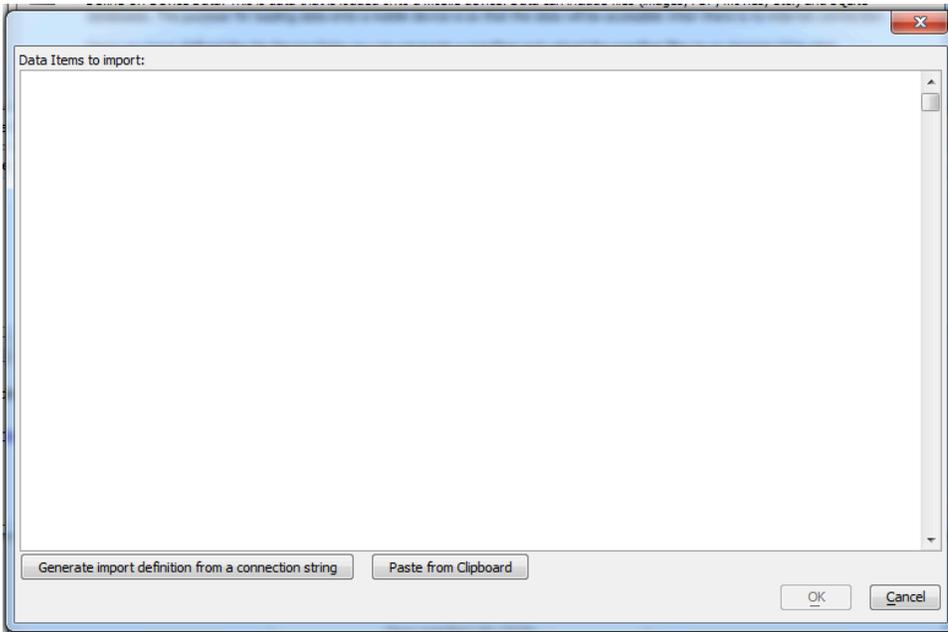
6. In the SQLite Database Definition settings box, Click the Ellipsis (...) button. The Create SQLite Database dialog box appears.

(Image 7.59 – Create SQLite Database Dialog Box)



7. Click the Import link to open the Import dialog box.

(Image 7.60 – Data Items to Import Dialog Box, Blabk)



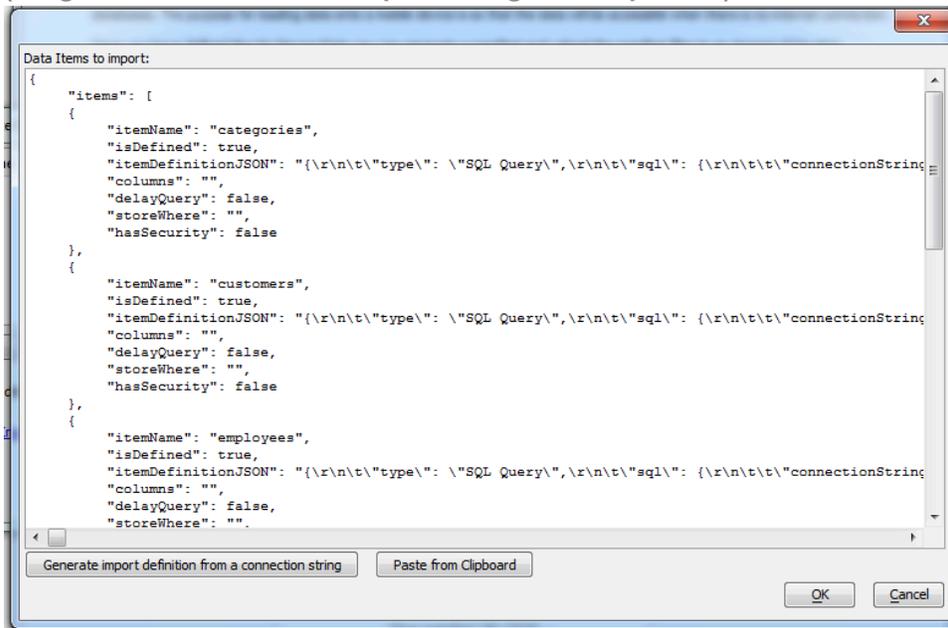
8. Click the Generate Import Definition From a Connection String button. The Connection String dialog box appears.

(Image 7.61 – Connection String Dialog Box)



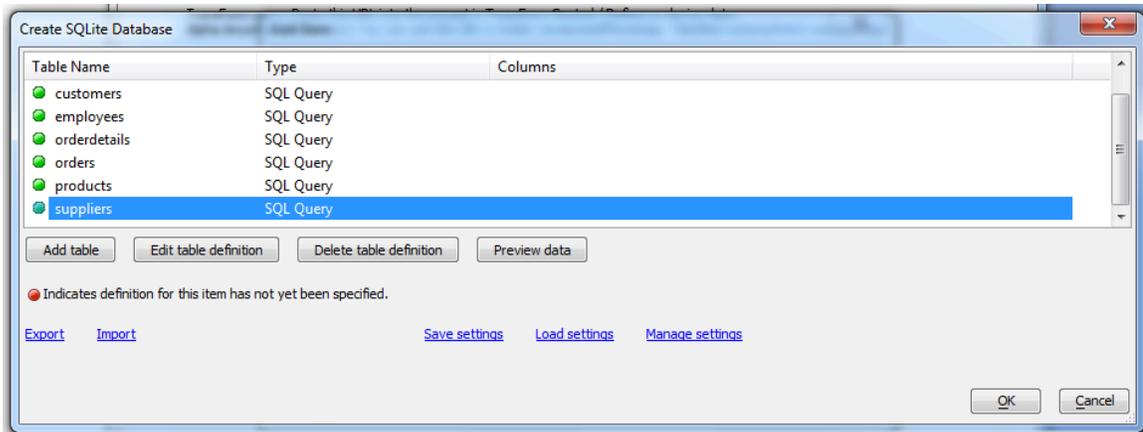
9. Type in the name of your connection string and click OK. TransForm creates a JSON list of the items in the database to import.

(Image 7.62 – Data Items to Import Dialog Box, Populated)



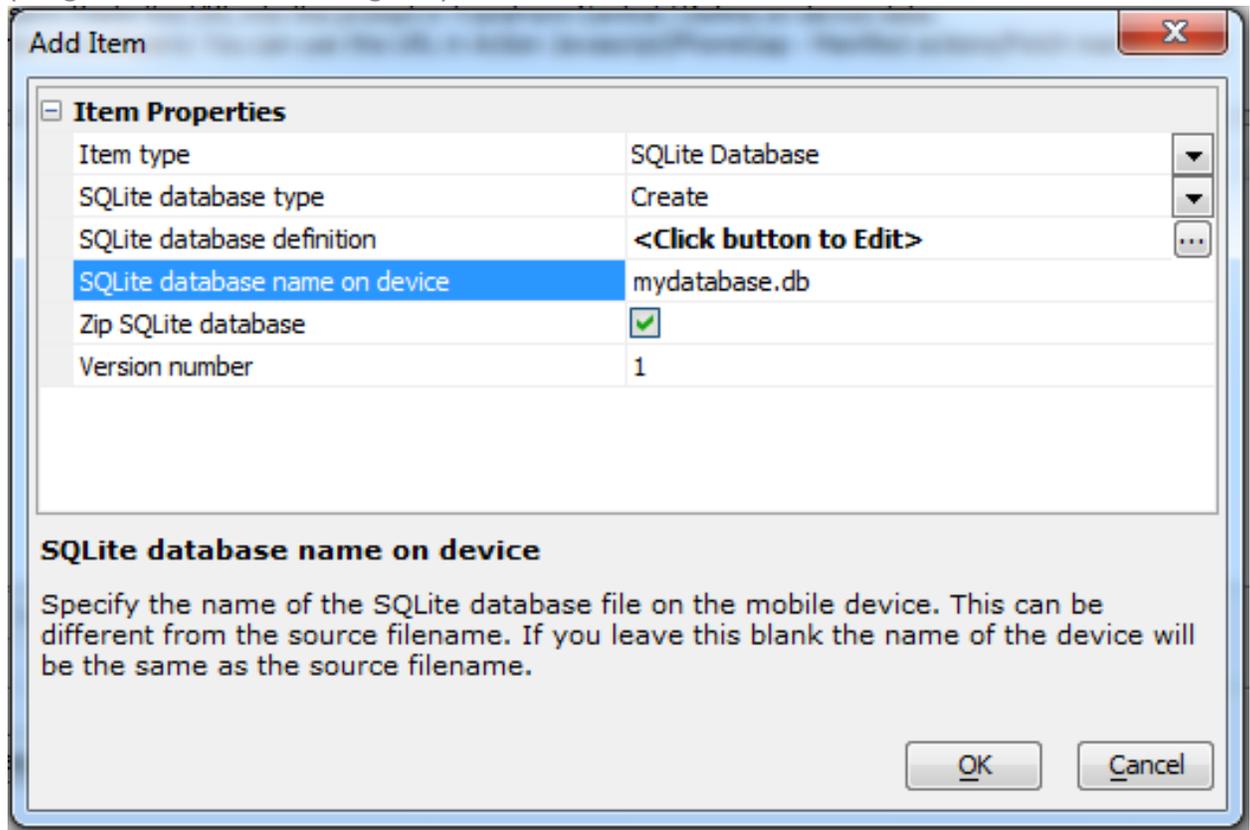
10. Click OK to perform the import. You are returned to the Create SQLite Database dialog box. A list of the imported tables now appears.

(Image 7.63 – Create SQLite Database, Now Populated with Tables)



11. Click OK to return to the Add Item dialog box.

(Image 7.64 – Add Item Dialog Box)



12. Give the database a name by filling in the SQLite Database Name on Device box. The name you assign must end in the extension “.db”.
13. Click OK to return to the On-Device Data Builder. You can now generate and upload the files to S3 so that the database is available to TransForm.

For more information on using the On-Device Data Builder, refer to the section, “How to Load, Refresh and Delete On-Device Assets,” earlier in this chapter.

Where to Get More Information

The information in this chapter was meant to give you a practical overview of how to use TPL in your forms. For more details refer to the online documentation at:

<https://documentation.alphasoftware.com/TransFormDocumentation/index?search=tpl%20overview>