Avoka Transact

Transact Maestro - v18.05 Release Documentation

Unknown macro: 'redirect'

Transact Maestro v18.05

This documentation is related to the latest Transact Maestro release.

Looking for an earlier Maestro release? Please see Maestro - Previous Release Documentation for more information.

Search Maestro Documentation

What's New

View the Maestro Release Notes for information on new features and changes to Maestro 18.05.

Changes to documentation include:

- Semantic Versioning of Libraries
- Translation and CSV Files
- Shared Domain Models
- Card Content Template Component

Documentation Content

- Introduction to Maestro
- Maestro Forms

Recently Updated

Transact Maestro - v18.05 Release Documentation
May 28, 2019 • updated by Anonymous • view change

Maestro Forms
May 28, 2019 • updated by Anonymous • view change

Introduction to Maestro
May 28, 2019 • updated by Anonymous • view change

Card Content Template
May 28, 2019 • updated by Anonymous • view change

Repeating Block Template
May 28, 2019 • updated by Anonymous • view change

Introduction to Maestro

Unknown macro: 'redirect'

This section will give you some general information about Maestro and how it fits within the Avoka Transact Platform.

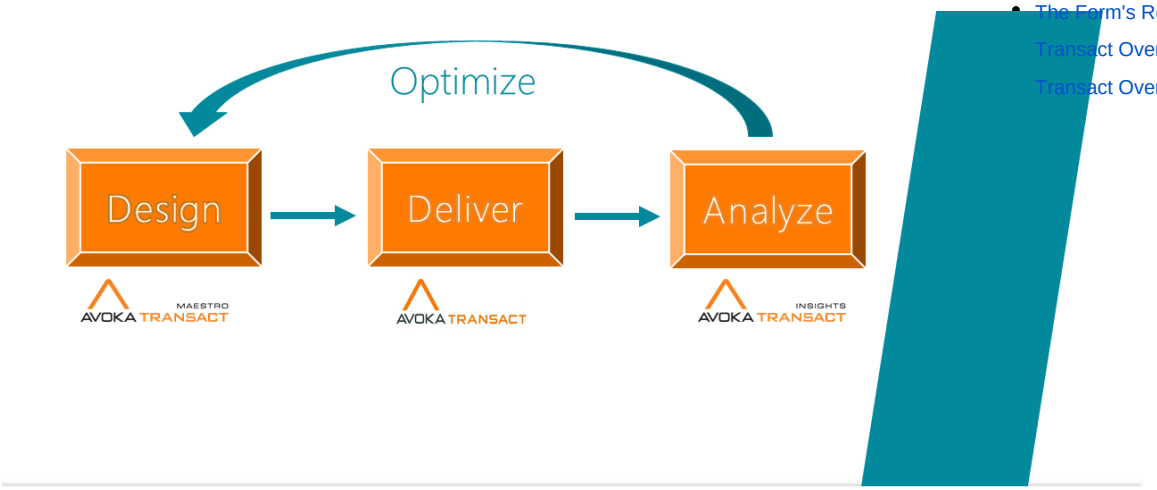
- Transact Overview
- Development Technologies and Programming Languages used in Transact
- Accessibility for Maestro Form Developers
- Maestro for Composer Developers
- The Form's Role in a Transaction
- Maestro Roles
- Terminology
- Release Details

Transact Overview

Unknown macro: 'redirect'

The Avoka Transact design tools allow creation of highly customized forms that incorporate all of the features and intelligence of the platform. Everything created is mobile responsive and can be previewed in formats for phone, tablet, and desktop.

A single design runs on all devices, takes the screen size into account, and gives an optimal experience without creating versions for each device.



There are 3 main components in the Transact Platform:

- Maestro is a cloud based, HTML5, design environment that allows your business teams to build sophisticated and elegant forms quickly and easily.
- Transact Manager provides an enterprise forms management system enabling sophisticated customer-interaction and system integration between form transactions and back-end business processes.
- Insights captures end-customer behavior to drive detailed transaction analytics on how to reduce friction and abandonment within a form.

Related Pages:

- Maestro Roles
- Maestro Roles
- Terminology
- Terminology
- The Form's Role in a Transaction
- The Form's Role in a Transaction
- Transact Overview
- Transact Overview

Development Technologies and Programming Languages used in Transact

Unknown macro: 'redirect'

Introduction

A Maestro form is a device used to collect information from a user. The purpose of a Maestro form will vary depending on the organization that is building and implementing the form. The development technologies and programming languages that are available to Maestro Form Builders vary depending on these purposes. However, there are several development technologies and programming languages that are used in every Maestro form. These development technologies and programming languages are identified in this page as well as their role in the development of a Maestro form and/or their place in the Transact Platform.

HTML

Maestro forms are fundamentally built using HTML 5. This means the structure of the form and the text displayed in the form are all configured and established using HTML5. HTML is the standard markup language for creating Web pages and stands for **H**yper**T**ext **M**arkup **L**anguage. HTML 5 describes the structure of a Maestro form.

JavaScript

JavaScript is a scripting (programming) language that allows Maestro Form Builders to implement complex rules that add additional functionalities to a form.

JSON

JSON is a syntax for storing and exchanging data and is used in Maestro to store forms. If a Form Builder wants to transfer their form to another system, they can do so by using the Download JSON file button. JSON stands for JavaScript Object Notation.

CSS

CSS stands for Cascading Styles Sheets and is used to define styles for a Maestro Form, including the design, layout, and variations in the form's display on different devices and screen sizes. The CSS on a Maestro form is primarily taken care of by the Template Designer. The Template Designer will implement the branding and/or styling of the form's assets.CSS can be applied to the template of the form, individual components, and any other asset associated with a Maestro form.

CSV

CSV stands for Comma Separated values and is commonly associated with Microsoft Excel and spreadsheets. Maestro stores all translation files as CSV files. These files can be edited by language translators when they want to add new languages to a Maestro form.

XML

XML stands for eXtensible Markup Language and was primarily used in Maestro to store and transfer data. However, the move has recently been made to use JSON to store and transfer Maestro form data.

Groovy

Groovy is a powerful scripting language which runs on the Java Virtual Machine. Across the Transact platform, Groovy is used to create services in Transact Manager. These services are primarily used to create "plug-in" style systems that interact with Transact Manager.

Fluent

Fluent is a programming style that can be applied to a wide range of programming languages. Across the Transact Platform, fluent is commonly associated with the Fluent SDK. The Fluent SDK is intended for use in developer integrated developer environment's (IDE) and contains libraries and project files for all the different services it supports.

Accessibility for Maestro Form Developers

Unknown macro: 'redirect'

A Quick Reference Guide

Accessibility can be considered the degree to which electronic services and products can be accessed by the largest number of users possible. Although, often focused on disabled people using assistive technologies including screen-readers, this is relevant to all people, including those that are on slow broadband, using non-mainstream operating systems, using a mobile device or speaking non-native languages.

Since 1999 the primary international standard for website accessibility has been the Web Content Accessibility Guidelines (WCAG) developed by the W3C. This has formed the basis for accessibility legislation around the world, including:

- Section 508 in the US
- The 2000 Government Online Strategy in Australia
- The UNCRPD signed by all EU member states

Many Avoka clients care about accessibility because it is good for society and, in some sectors, compliance has been legislated.

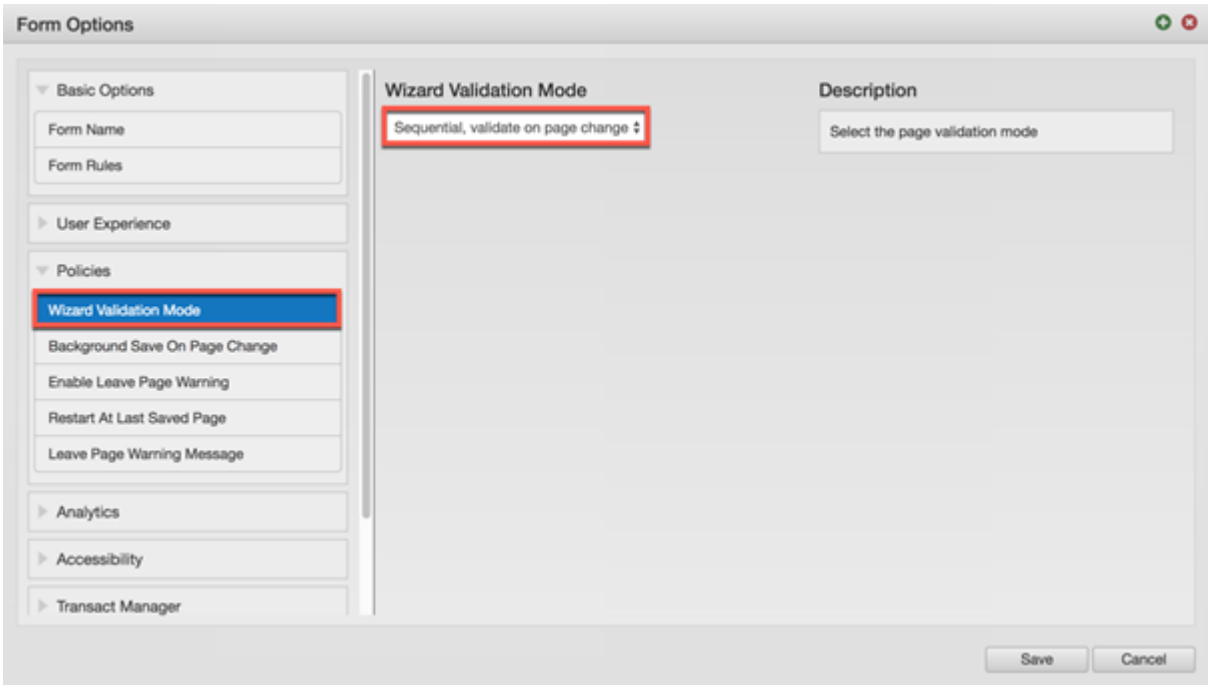
Avoka Transact Maestro provides as much of the basic, component-level accessibility as possible automatically. For instance, linking a component's label text with its input field. This document provides a guide of the additional tasks that a Maestro form developer must perform to achieve maximum accessibility.

Checklist:

1. Use sequential page navigation
2. Use page and section headers
3. Mark-up lists
4. Add alternate text to images
5. Do not use background images
6. Use fieldsets
7. Do not use label placement left
8. Check color contrast
9. Do not use placeholder text
10. Do not replace component labels with text
11. Do not disable anything

1. Use sequential page navigation

Form Options > Policies > Wizard Validation Mode

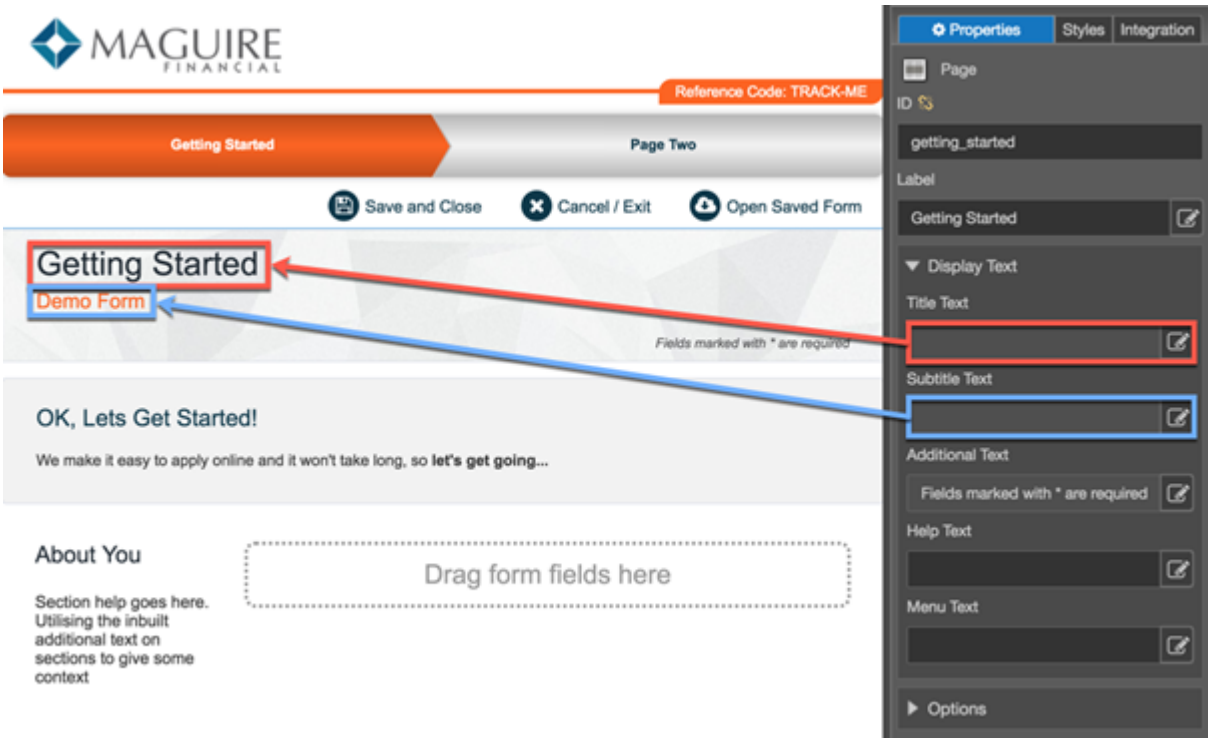


This policy property defines how validation and navigation operates for the form as a whole. Sequential page navigation is better for accessibility as it provides a simpler experience where users navigate through the form sequentially, one page at a time, and does not allow them to progress until the current page passes validation.

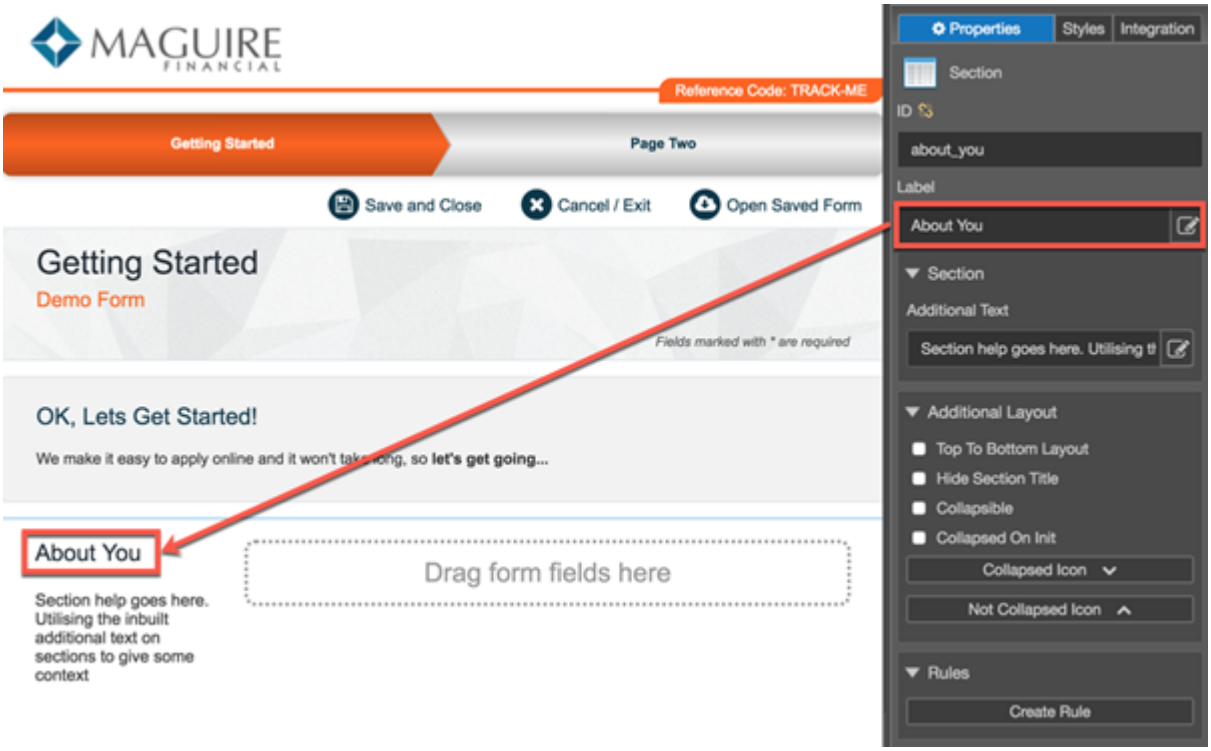
Unconstrained page navigation allows the user to navigate to any page, in any order which can be confusing (particularly on mobile devices) and does not perform full form validation until they attempt to submit from the final page. Having to then potentially correct issues across multiple pages can be a challenge.

2. Use page and section headers

Page > Properties > Display Text



Section > Properties > Label



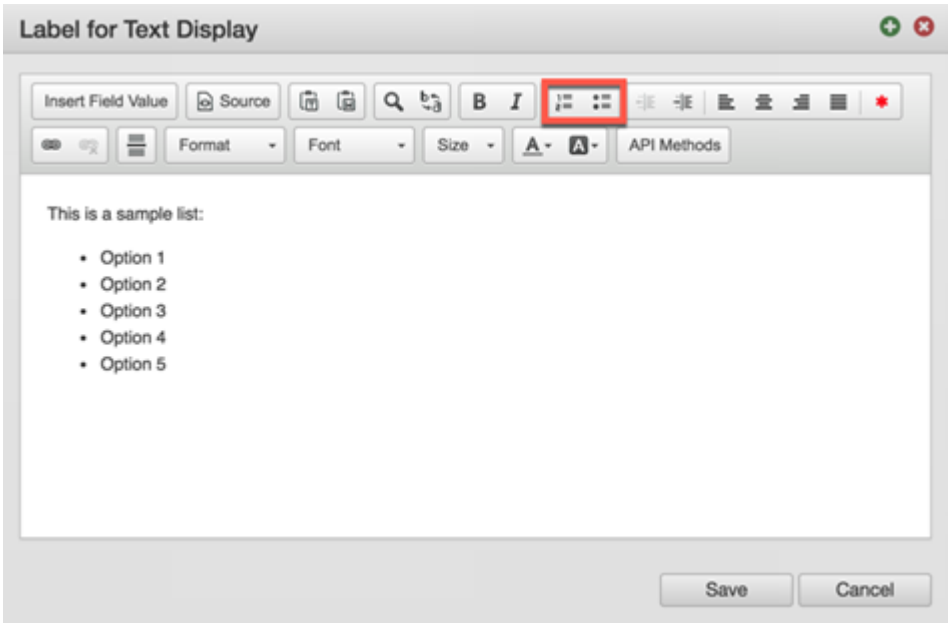
Correctly marked-up headings are provided automatically by the page and section headers at levels 1, 2 and 3. Use page and section headers and do not manually create them with text display fields.

3. Mark-up lists

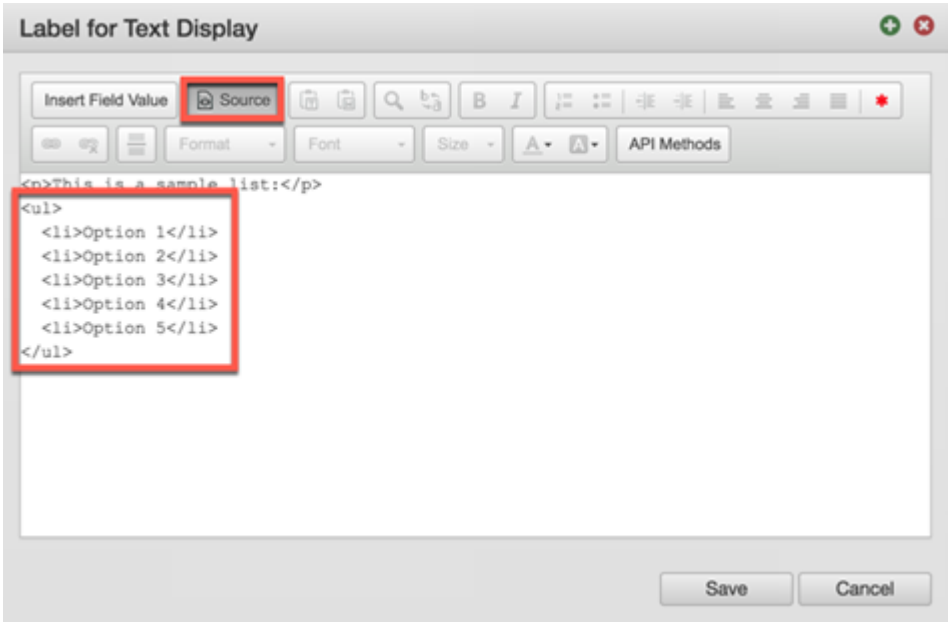
When creating lists, it is important to ensure that the correct HTML mark-up is used. When copying and pasting content from external sources (e.g. Word Documents) the mark-up is often lost and replaced by characters representing the numbers or bullets in the list. Whilst visually this may look OK, the lack of correct HTML mark-up will prevent screen readers from interpreting the list as a list.

Using the built in rich text editor you can add lists using the ordered or un-ordered list buttons shown below:

Text Display > Label

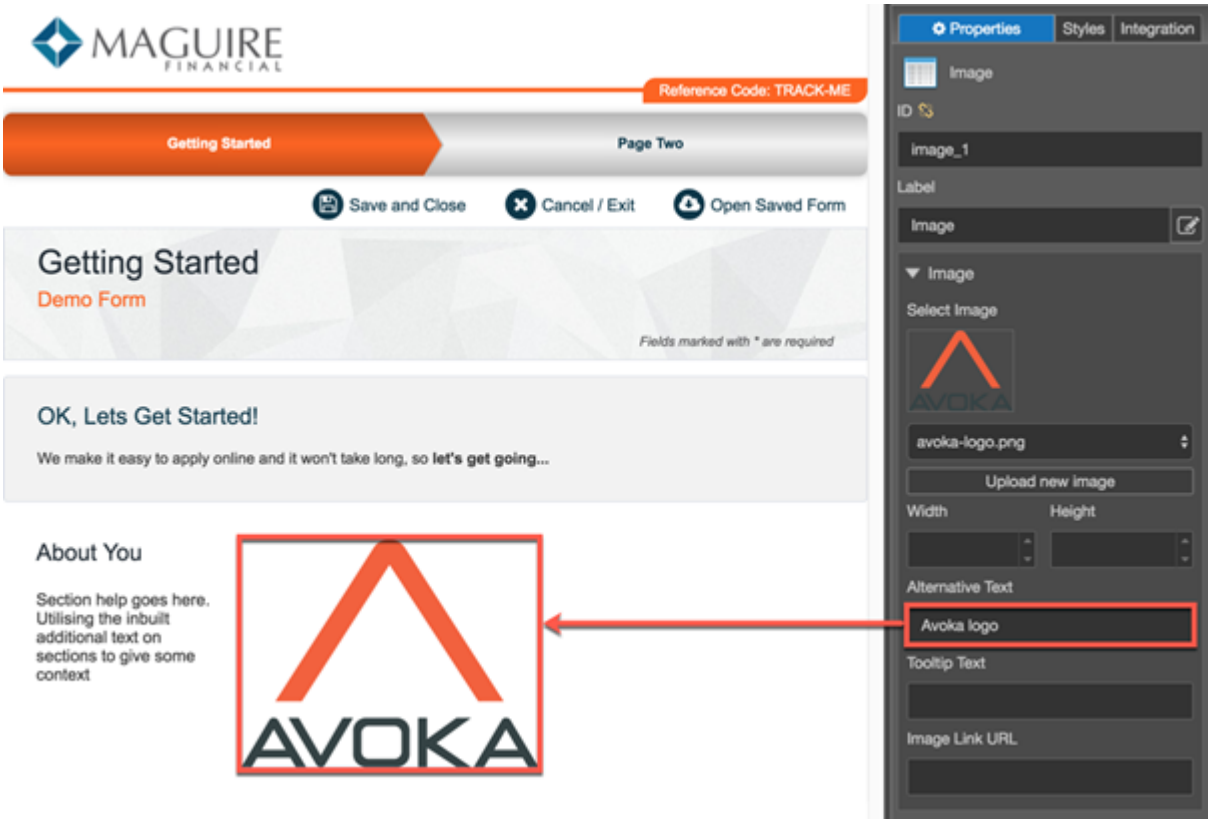


If copying and pasting content containing lists, please review the source and verify that correct HTML mark-up has been used:



4. Add alternate text to images

Image > Properties > Image > Alternative Text



Images can be informative or decorative. The form developer must use their judgement on whether an image should be treated as decorative or informative:

Decorative

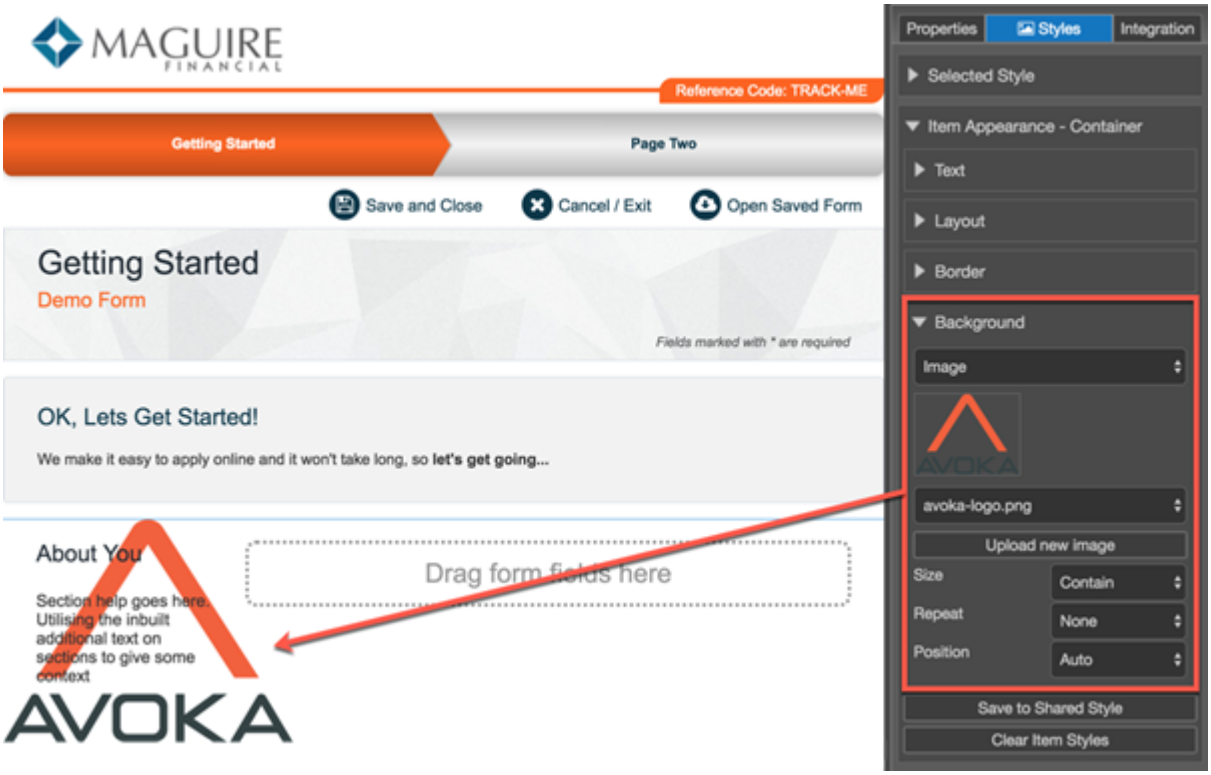
Decorative images do not convey information to the user and are typically used to make the form more visually attractive by adding visual components such as separators and textured backgrounds. An image is also considered decorative if it conveys information that is also fully provided by surrounding text. Decorative images should not have alternate text.

Informative

Informative images do convey information to the user that is not provided by surrounding text and should have alternate text that conveys the meaning or content of the image. The alternate text need not be a literal description of the image and should be as succinct as possible.

5. Do not use background images

Component > Styles > Background



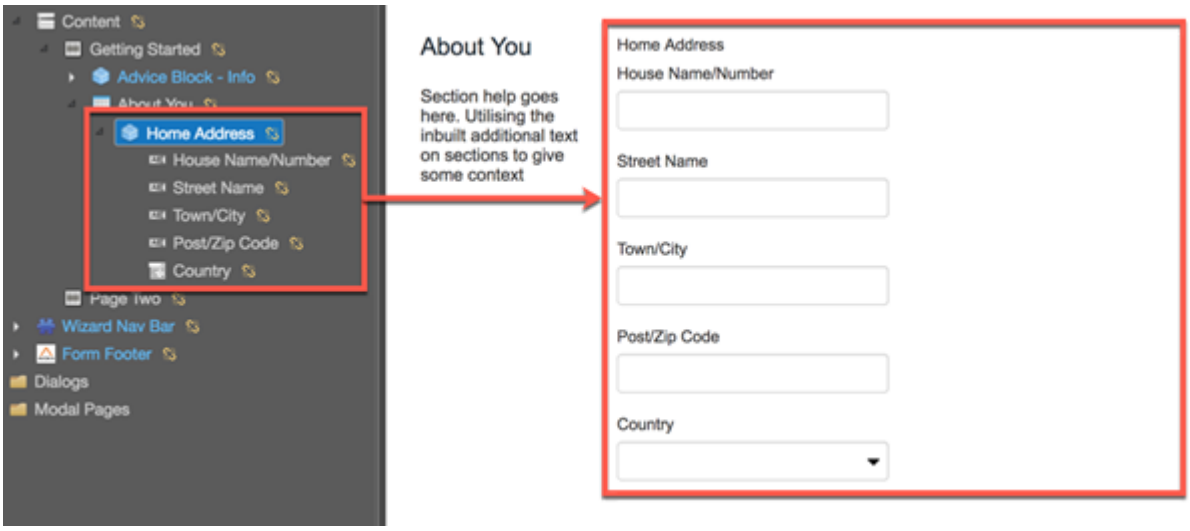
Informative images should not be used as a background image as it is not possible to provide them with alternate text and some browsers remove background images when put in high contrast mode. It is fine to use decorative images as a background image.

See item 4. Add alternate text to images, for a definition of informative and decorative images.

6. Use fieldsets

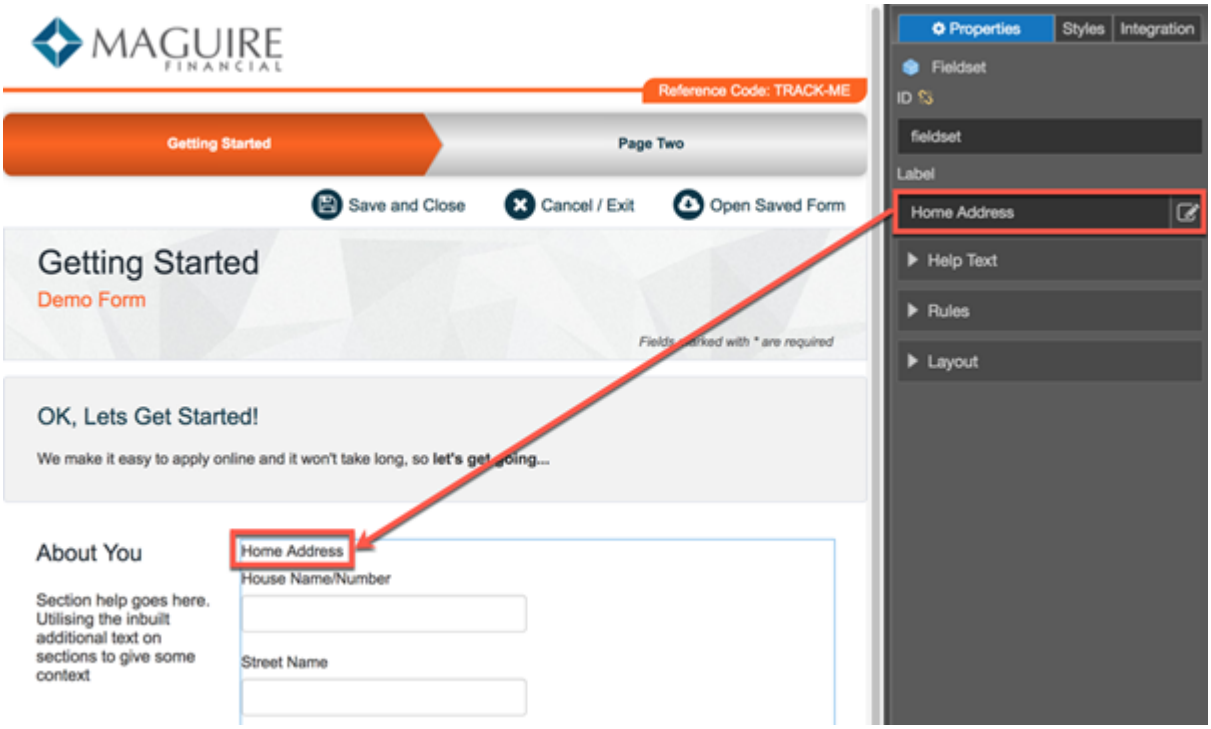
Fields that are not stand-alone and only make sense in the context of a group of fields should be contained within a fieldset. For instance, when using:

- Multiple Checkbox components (e.g contact preferences)
- Multiple components (e.g. telephone region code and number, home address etc.)



A meaningful description of the group should be provided in the fieldset's label (legend text) which is displayed on-screen as a heading:

Fieldset > Properties > Label



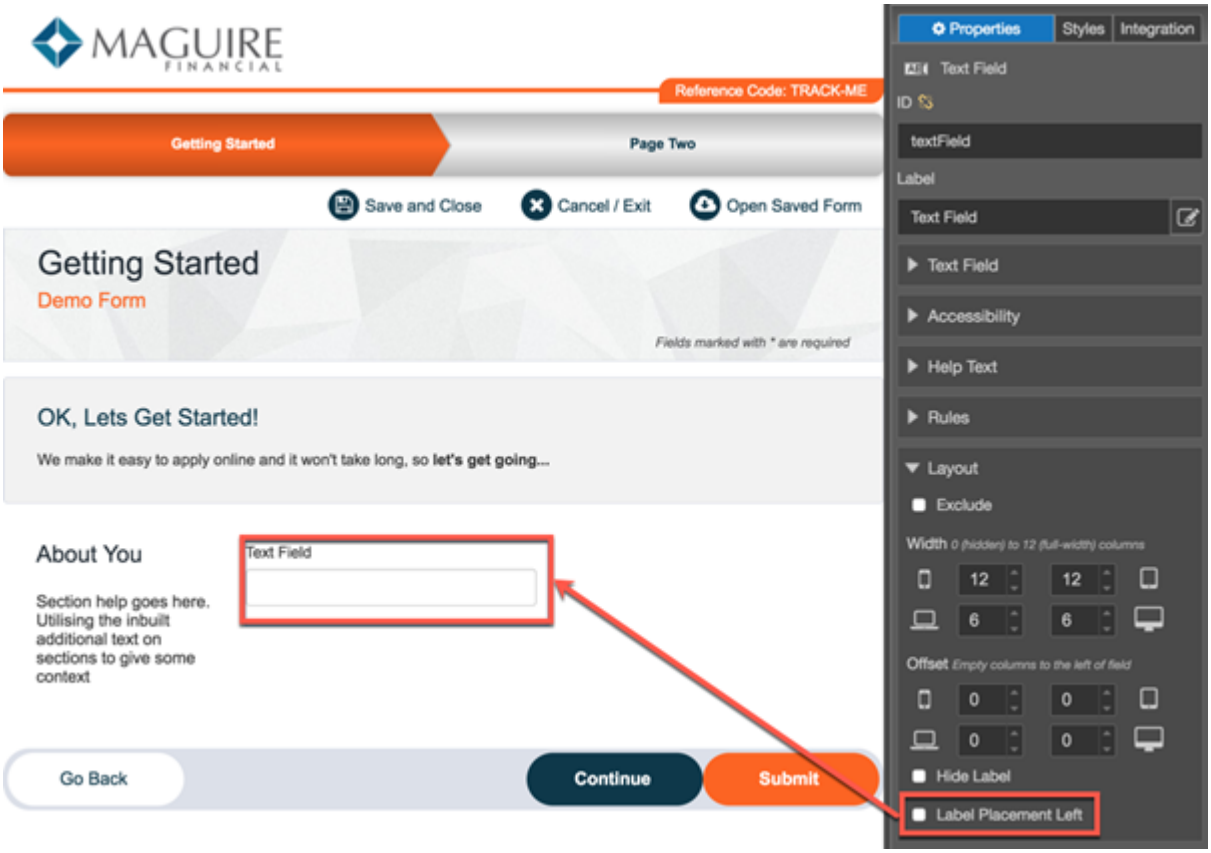
The fieldset's legend text will be used by assistive technologies to inform the user of the context for fields within the fieldset. For instance, tabbing to a checkbox button whilst a screen-reader is running might inform the user that they are currently focused on a checkbox button with caption "Email" for the fieldset "How would you like us to contact you?":

Without the fieldset, the user would only be informed that they are focused on a checkbox button with caption "Email" which is clearly insufficient. To avoid repetition, the majority of assistive technology will only inform the user of the fieldset legend for the first field focused within the group, regardless of which field that is i.e. not necessarily the first field in the group by position.

It should be noted that when using the Radio Button Group component that there is no need to use a fieldset as this component includes one already.

7. Do not use label placement left

Field > Properties > Layout > Label Placement Left



Label placement above the field (Maestro default) is best for accessibility as it keeps the caption closest to its field. In particular, this makes it easier for users of magnifier software to locate the field associated with a label without having to move the cursor too far and reduces the chances of finding the wrong field by mistake.

Top placement also provides the best scan line between captions and inputs for fields. A scan line refers to eye movement along a straight line as a sighted user browses a form or searches for a specific field. Other label placement results in additional, unnecessary movement as the sighted user's eye jumps around between labels and inputs that are less aligned.

8. Check color contrast

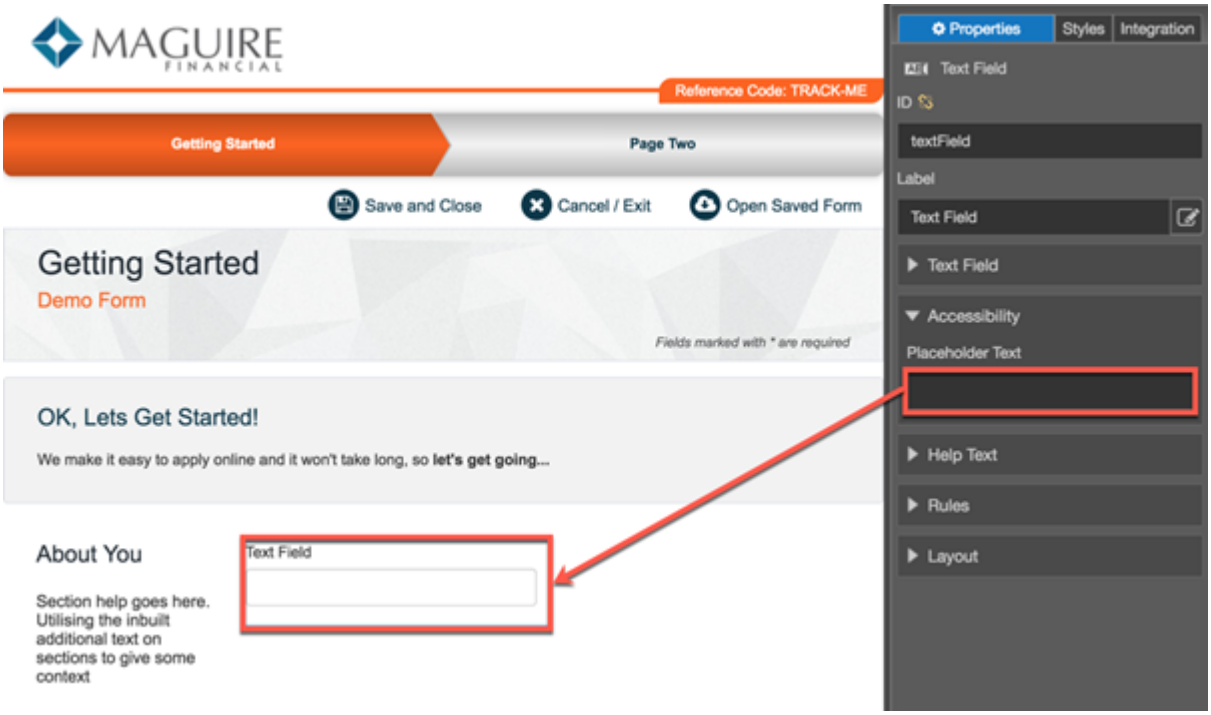
Colours used in a form must meet accessibility requirements for contrast of foreground and background objects for partially sighted and colour-blind users.

You can check your form using a contrast tool such as the Colour Contrast Analyser from The Paciello Group:

https://developer.paciellogroup.com/resources/contrastanalyser/

9. Do not use placeholder text

Field > Properties > Accessibility > Placeholder Text



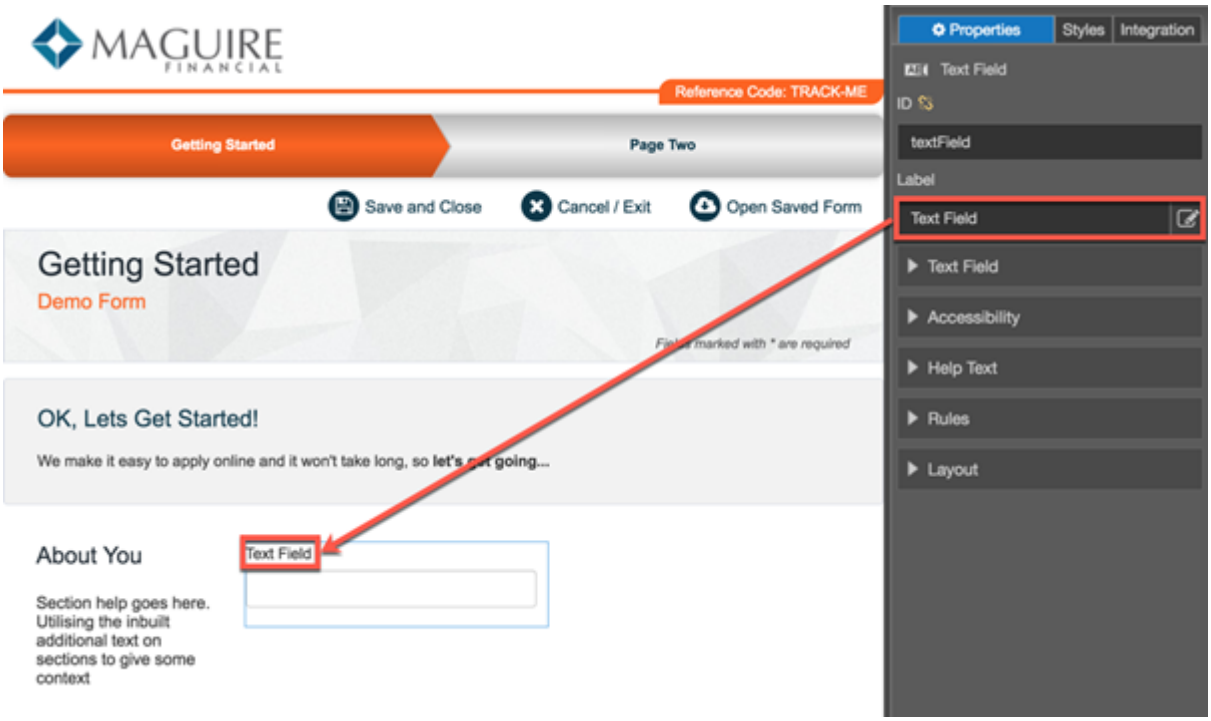
Placeholder text is the light grey text that is displayed in a field until data is entered at which point it is removed.

Placeholder text is not accessibility compliant due to the lack of consistent support in assistive technologies. Some assistive technologies ignore placeholder text entirely. Some always notify the user of the placeholder text regardless of whether there is data in the field or not and makes no differentiation between the placeholder text versus the entered data.

Only a small proportion of assistive technologies do the correct thing in only notifying the user of the placeholder text when it is visible on-screen.

10. Do not replace component labels with text

Component > Properties > Label



It can be tempting to hide the built-in field label and create a replacement with a separate text display component in order to achieve layouts such as a label that is wider than its field.

Although, this approach may look identical on-screen to sighted users, this does not provide the browser functionality where selecting a label sets focus to its field.

To resolve the specific case of long labels, it is better for accessibility to use a combination of shortening the label, widening the component and moving non-critical information from the label to a popover tooltip or a message box located close to the component.

11. Do not disable anything

Disabled form content is problematic for accessibility and should be avoided.

Typically, disabled content is visible on-screen for sighted users but not active and visually indicated as disabled with grey-out styling. Disabled content is not discoverable for users of assistive technology which results in a mismatch of information. For instance, in the case of disabling the continue button for a page until certain actions are performed a sighted user can see that the button is disabled and may be able to deduce that they need to perform an action before it will be enabled.

However, a user of assistive technologies will have no knowledge of the existence of the continue button at all which is likely to lead to confusion of what action is expected of them in order to progress.

In this case, it is better for all users to keep the continue button enabled and inform the user what is expected of them with error messaging if they attempt to progress without completing necessary actions.

The other common alternative approach to disabling components is to hide them.

Maestro for Composer Developers

Unknown macro: 'redirect'

For Maestro users that have experience with Composer, the following items will give you a brief overview of the changes (and similarites) you may notice as you begin to work with Maestro.

Some things you will probably like:

- Most developers report that the development process is faster and simpler.
- Maestro uses native CSS as its primary mechanism for styling. (Composer uses its own mechanism because it needs to work with both PDF and HTML.) There are a lot of online resources on CSS, and so it's easier to find out how to do more sophisticated styling in Maestro.
- With component labels, you can now "embed" the value of any component into your label. This is particularly useful for section headings, etc. (In Composer, you had to set up fairly complex business rules to do this, or use a floating field which has limitations.)
- Drag and drop is improved, particularly a better indication of the insertion point. Both structure pane and editor allow you to drag and drop above, below, or within a container.
- You can preview easily in different layout sizes.
- Scripts are validated for syntax errors, and prevent you from saving an invalid script. (This doesn't work for all types of errors, just syntax errors.)
- There is a comprehensive undo capability, as well as a history of edits. This makes it easier and safer to change things, because you know you can undo if necessary.
- You can easily define shared and re-usable styles.
- The currency field looks more modern.
- Attachment component is improved. It supports thumbnails, drag and drop, and multiple attachments.
- You can animate hiding and showing components, which gives good feedback to the end-user.
- Radio buttons are much simpler now - just a single component with a set of option values (very similar to a drop-down list). In a small number of cases, you may miss some of the functionality you could achieve with separate radio buttons, but generally radio buttons are much easier to use.

Some things that are different, but which you will probably learn to like:

- When you start Maestro, you will not see the Component palette plus the form hierarchy view - they are tabs that you have to switch between. As a Composer developer, you will probably want to immediately click the small icon at the top of the palette - this will show these two views side by side. Unless you have a very small screen, this is probably the mode you want to be in.
- The component palette is on the left instead of the right, alongside the hierarchy. One of the advantages of this is that it is a shorter distance to drag a component from the palette to the hierarchy.
- There are two nice "tricks" for adding components. You can:
    - Double-click on a component (as you could in Composer)
    - Right-click on a component.
- Developers have reported that business rules are a little simpler.
- Layout is completely different. Maestro uses a simple 12 column grid for layout, with different column spans for each screen size. This is generally much simpler than Composer, but you may sometimes need to solve complex layout issues in a different way than you are used to.
- Repeats are handled differently in Maestro. In many cases, you will find this simpler once you understand it. For example, you may want to create a copy of a repeatable block on a summary page (or for some other reason). Instead of performing a relatively complicated calculation on each of the fields in the repeat, and trying to make sure that the number of repeats match in both places, you can simple point the second repeat at the data for the first one. The Maestro framework will keep everything in synch.
- There are no separate integer and decimal fields in Maestro unlike Composer. Maestro just has decimal field which behaves like an integer field if the decimal places are set to 0

Some things that might take a while to get used to:

- Previewing is a bit different - it's a tab in the same window rather than a pop-up window. If you're familiar with Composer's pop-up window, you might accidentally close it, instead of changing tabs. At the time of writing, the window closes without a warning - so be careful.
- When creating a business rule, you choose a field, but what ends up in your script is a reference to the underlying data element - not the field itself. This is an important difference that you should be aware of.
- Like Composer, internal names need to be unique. Composer handles this by referring to each component by a pathname, which identifies its location in the structure. Maestro, on the other hand, just add numeric suffixes to make fields unique. For example, "FirstName", "FirstName_1", "FirstName_2". When you are developing scripts, this is not a problem, as you will be navigating the field hierarchy to select the field. However, when you look at a script that someone else has written, it's harder to understand because you don't know whether "FirstName_1" refers to the name of the first or second applicant, or the guarantor, or someone else. It is possible to override these default generated names with more meaningful ones.
- It is possible to create native components (or composite widgets) that have elements (or sub-fields), and these elements are completely self-contained within the component. In other words, these elements do not appear in the hierarchy view, and you cannot access them directly. This means that you cannot, for example, make one of these elements mandatory or hidden. If you need fine-grained control over these elements, you will need to communicate with the component developer, and request that the component expose these capabilities at the top level.
- Note that in one of these self-contained native components, you can "reach down" into the sub-elements and specify custom styling. Refer to the styling documentation for details.
- The 12-column layout concept is generally a good thing, but can occasionally be confusing. For example, if you try to add a component that by default uses 6 columns, to a line that already has components using up 8 columns, the new component won't "stay" where you're tyring to place it, and will want to move to the line below.
- There is no way to specify that a field should stay on the same line as the previous one using properties. (In Composer you could set the "stay on same line" property.) The only way to do this is to drag the field to the previous line.

New items that are important to understand:

Note that many of these items will be improved over time, and therefore some of these issues may go away.

- When you create your first form, make sure you select a custom template or a placeholder template, not a Maguire sample template.
- Understanding the publish concept for shared components. You cannot use shared components immediately - you must first publish them to the library before you can use them in your forms. It's easy to do this, but confusing if you're not aware of this requirement.

- In JavaScript rules, Composer will "cast" the data to the data type you choose. In Maestro, all references are by default strings. You need to do all the casting yourself, in your Javascript. To cast to a number, prefix the reference with a '+'. To cast to a boolean, prefix with '!!'. (This is a boolean "not-not" - which casts to boolean without changing the value.)
- When a custom block is changed - the generated IDs may change, and your scripts may now refer to data elements which no longer exist. Your script will fail silently.
- Copy and pasting HTML text into a text property is easy, but sometimes it introduces unexpected HTML markup. Always go to the source tab to make sure you haven't accidentally introduced some unwanted HTML.
- Maestro is more aggressive than Composer in logging you out, if your session has expired it will open a new tab to log back in and take you to the dashboard and you can then continue editing your form in the original tab. It is still a good idea to save before going for coffee.

Some things you may miss:

Note that many of these items will be included into Maestro over time.

- There is no way of looking up dependencies, and in particular, broken dependencies.
- There are no assistants.
- No bulk editor.
- Publishing to the current Transact Manager server is much easier, because Maestro actually runs on the same server as TM. Publishing to another TM instance is more difficult. You have to publish to your current TM instance, and then go into the TM AdminUI, export from there to a file, and then import to the new TM environment.
- Save as is missing.
- SetInstance count is not available. There are different ways to handle repeats.
- All deletes are hard deletes so to get back deleted items requires going to backups.
- There is no user event logging, i.e. who edited the form/shared component/template/etc

The Form's Role in a Transaction

Unknown macro: 'redirect'

A Form is the device used to collect data from people. A Form moves through various events of the Transaction Experience, usually configured by a Business Analyst (BA) or Subject Matter Expert (SME) based on requirements gathered. Avoka refers to the BA or SME as the Form Designer.

A Transaction Experience is a sequence of events that includes Prefill, Data Capture, Save and Resume, Attachments, Payments, Assignments, Notifications, and Collaboration between parties.

Some of these events have an impact on the Form design, while others are configured in Transact Manager.

Organizations frequently have high-level concerns before beginning form development, including:

- Branding
- Integration
- Accessibility
- Security

These concerns are translated into action during a Form's development that can include:

- Styling
- Validation
- Data Structure
- Usability

Maestro provides the tools to address these concerns using the UX principles embodied in the software.

Related Pages:

- Maestro Roles
- Maestro Roles
- Terminology
- Terminology
- The Form's Role in a Transaction
- The Form's Role in a Transaction
- Transact Overview
- Transact Overview

Maestro Roles

Unknown macro: 'redirect'

Maestro breaks down the tasks for users based on their roles. There are two main roles within Maestro. These two roles will focus on different aspects of the form creation process.

1. Form Builder
2. Template Designer

The Form Builder may be further broken down into two roles, Form Builder and Form Developer. The Form Developer role is an advanced user that works to extend the form using JavaScript and dynamic data.

All Maestro forms are based on a template. Templates are the starting point for all forms.

Forms and Templates will contain all of the same panels – Palette, Structure, History, Properties, Styles and Data. The layout of both the form and template are similar, however certain aspects of each are tailored to the relevant role.

Form Builders are responsible for:

- Content specific components
- User's journey through the form

The form builder focuses on the specific form content as well as the journey of the form user. The form builder should not change any styling without first talking to the template designer. This will ensure that the corporate style guide is properly used and consistent across all forms.

Template Designers are responsible for:

- (almost) All styling
- Modal page and dialog content that is shared across forms
- Implementing brands
- Creating shared form options to make it easier for form builders to configure their form.

The responsibility of the template designer includes the styling of the form's chrome. This would include areas such as the navigator, footers, and styling of specific components. The template designer works with the corporate style guide to create a template.

The template designer does not focus on the actual content of the form nor do they look at the user's journey.

Please be aware that there is sometimes content added to the template. This content is considered template content and is used across most, if not all forms created. Template content may include the content within modal pages or dialogs, such as the submission confirmation modal page.

The template designer is also responsible for the management of published templates and styles so that they are made available to form builders.

Customization

You may find that your experience of Maestro doesn't quite match the screenshots and other details shown in this documentation.

This is because Maestro is highly customizable, and your organization may have customized Maestro in ways that make it look unfamiliar. In particular, you may be using a different template or brand, and you may see a different set of components

However, the concepts of Maestro are generally the same regardless of the customizations.

Related Pages:

- Maestro Roles
- Maestro Roles
- Terminology
- Terminology
- The Form's Role in a Transaction
- The Form's Role in a Transaction
- Transact Overview
- Transact Overview

Terminology

Unknown macro: 'redirect'

The table below provides common terminology used within Maestro.

Term	Explanation
Component	A component is a general name for anything that can be dragged from the Palette into a form. Components can be anything from an image to a single data entry field, to a large complex re-usable block.
Field	A field is a specific type of component used for displaying and entering data. Examples include the Text Field or Decimal Field.
Native Component	A native component is a component built out of HTML, JavaScript, and CSS. Under the covers, a native component consists of an HTML definition, a descriptor that tells Maestro what properties to display in the Properties tab, and optional JavaScript and CSS. A native component can only be built by someone who understands these web technologies, as well as some understanding of how Maestro components work and interact with other fields and the validation framework. Ho wever, as a Maestro user, you are shielded from this underlying complexity, and simply drag and drop the component onto your form. You can make a very simple native component or you can create arbitrarily complex components that include multiple input fields and visual elements.
Multi-field component, or complex component	A multi-field component is made up of more sub-parts than a regular component. For example, a Section consists of the Heading text, sub-text, help text, a collapse/expand button and a significant amount of JavaScript and CSS. An address block contains several different simple data entry fields. A component such as a Driver's License scanning component may have multiple data entry fields, a camera button, descriptive images, and more. Multi-field components can be built as either a native component or a re-usable block - see below.
Native multi-field component	A native multi-field (or complex) component is still a completely self-contained component and occupies only a single line in the structure pane.This component has exactly the same internal structure as a simple component, just there is more "stuff". It is still made up of HTML, JavaScript, and CSS. There is no fixed boundary between a simple and a complex native component, although any component with more than one visible data entry area within it would usually be considered to be complex.
Re-usable Block	A custom block is a collection of simple components, that can be re-used. An example of a re-usable block is an Address Block, which consists of two lines of text plus City, State and ZIP Code. Unlike a native multi-field component, each of the sub-elements of the custom block will be visible in the Structure tab, and can be modified individually by the person using your block.
Custom component	Sometimes it is useful to create a re-usable variant of either a simple or complex component. The main variations are: • A component with a re-usable business rule, particularly a validation rule. An email field is an example of this. • A component (radio button or drop down list) with a specific set of values. An example of this is a drop down list of States. • A component with a particular XML binding. See Create New Shared Components for more information.
Form User	The form user is the person completing the form. They may also be known as "user".
Form Builder	A form builder is the role name of the person that creates the form. A form builder focuses on the specific form content as well as the journey of the form user. The form builder should not change any styling without first talking to the template designer. This will ensure that the corporate style guide is properly used and consistent across all forms. See Maestro Roles for more information.
Template Designer	The template designer is responsible for implementing and maintaining templates which are the starting point for all forms created in Maestro. The key activity of the template designer is to implement the template content such as common page components that typically surround specific form content, including header, navigator, and footer. The template designer will also create styles that the form builder will use in creation of form content including text field styles, buttons, and section headers. See Maestro Roles for more information.
Editor Window	The editor window is the window that is displayed while working on Maestro assets, including forms, templates, and components. The Maestro editor is a window that consists of multiple panes. Some of the panes contain tabs that allow several different functions to be condensed into a particular area of the screen. This window may be known as the Form Editor (when working on a form), Template Editor (when working on a template) and Component Editor (when working on a component). See Working with the Maestro Editor for more information.
Design Mode	Design mode displays the editor window so that Maestro assets can be created and updated.
Preview Mode	Preview mode displays the form so that you can interact with the components on the form as if it were live. See Form Preview for more information.

Release Details

Unknown macro: 'redirect'

The content within this section covers information relating to Maestro releases.

The list below identifies the topics covered within this section.

- Maestro 18.05
- Clear Browser Cache

Maestro 18.05

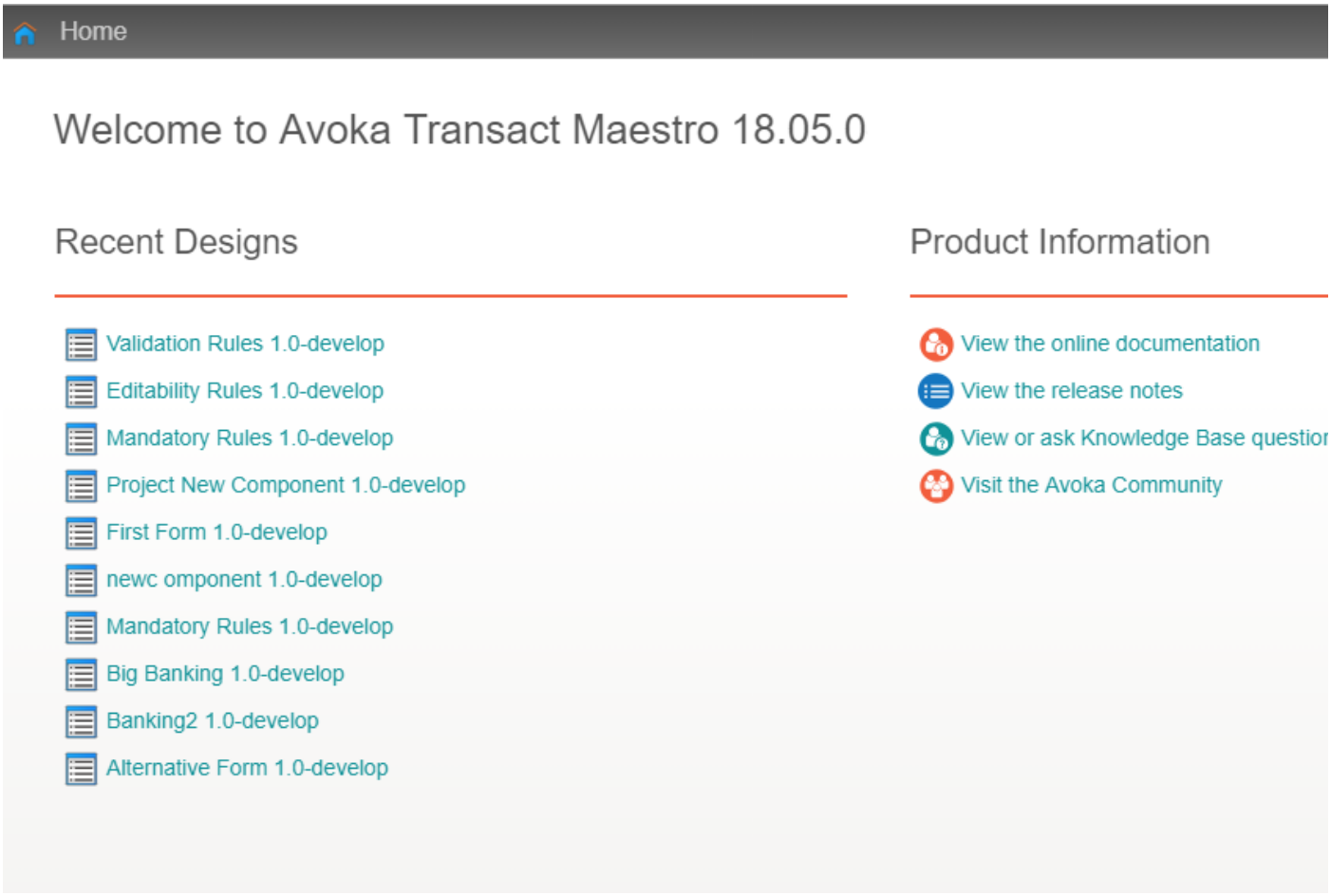
Unknown macro: 'redirect'

Important notice to all users of Transact Maestro

We strongly recommend that all users of Maestro clear their browser cache after the 18.05.0 upgrade of Maestro.

There have been instances where the browser has cached a previous version of Maestro and this can lead to unpredictable behavior in the tool. Flushing the cache ensures you get the latest version. We are working on a better solution to this issue in a future release.

For information on how to clear your browser's cache, please see Clear Browser Cache.



Maestro 18.05.0 Features

Below are the feature changes for the Maestro 18.05.0 release. Details about these changes can be found on the following pages:

- Semantic Versioning of Libraries
- Translation and CSV Files
- Shared Domain Models
- Card Content Template Component

Clear Browser Cache

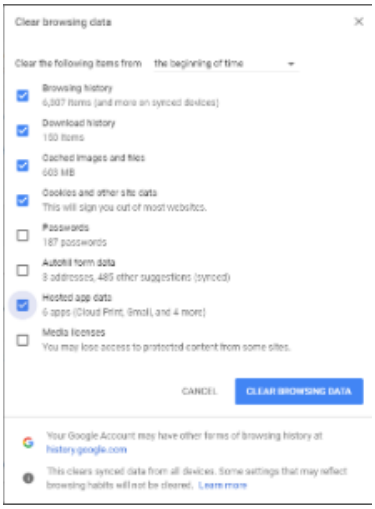
Unknown macro: 'redirect'

Clear Browser Cache

Use the following instructions to clear your browser's cache.

In Chrome

chrome://settings/clearBrowserData



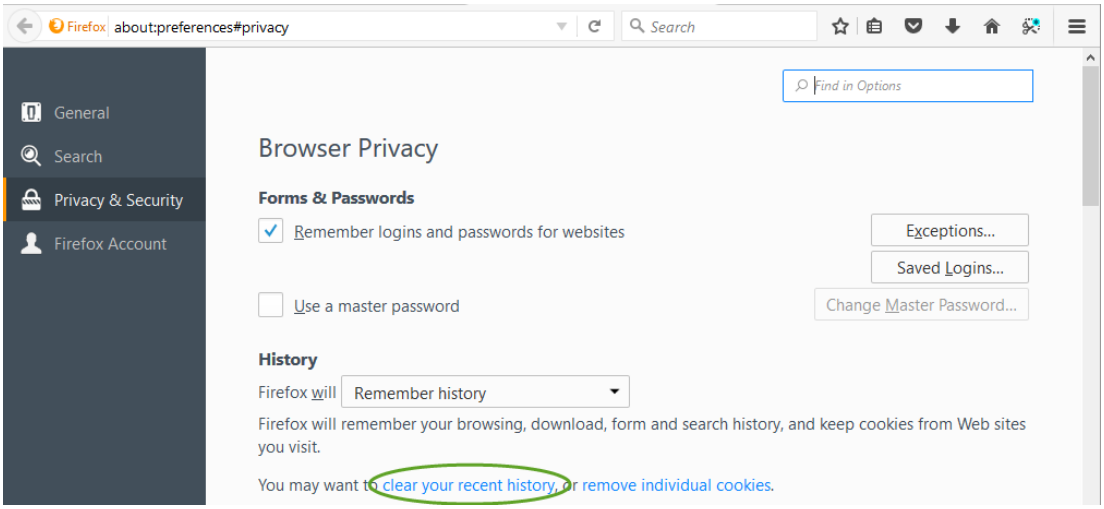
If this fails in Chrome then a deep clear of the cache will be required. Please refer to Deep Clear of Chrome Browser Cache article for more information.

In Firefox

about:preferences#privacy

In the Browser Privacy screen click *Clear your recent history.*

On the next screen, ensure cache is selected.



Maestro Forms

Unknown macro: 'redirect'

The content within this section covers information relating to Maestro forms.

The list below identifies the topics covered within this section.

- Management Dashboard
- Creating a New Maestro Form
- Managing Maestro Forms
- Configuring a Maestro Form
- Data Configuration
- Reference

Management Dashboard

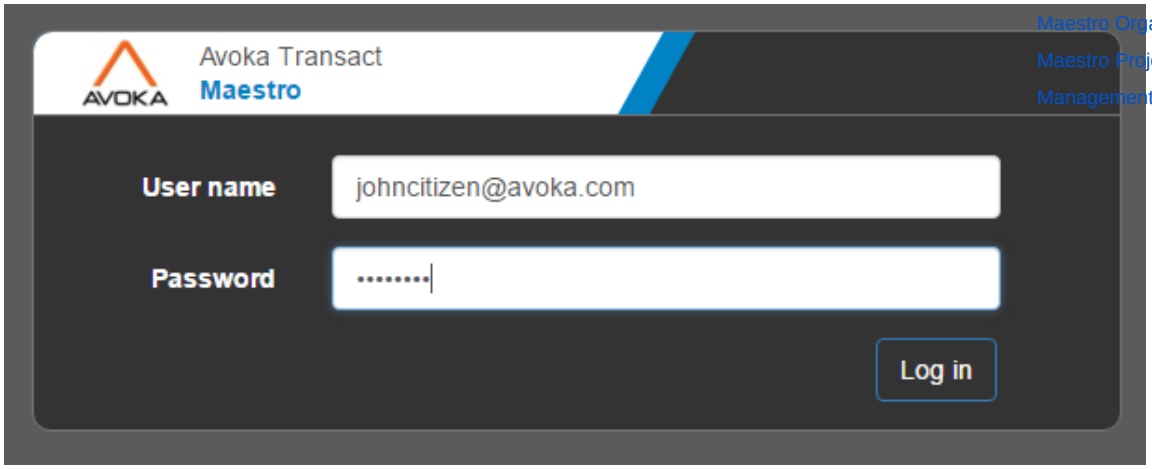
The content within this section covers information relating to the Maestro Management Dashboard. This includes logging in to Maestro and understanding the structure of Organizations and Projects.

The list below identifies the topics covered within this section.

Logging In

Unknown macro: 'redirect'

The login screen requires the user to enter their user name and password and select the Log in button. User access will be set up by the Transact Manager security administrator.



Maestro Sessions

Once logged in, your Maestro session is valid for a limited time. This is a security measure used to prevent unauthorized access to the Maestro development environment. There are some important things to remember about Maestro sessions.

Maestro sessions are valid in the browser of first log in

Maestro stores its session information in a cookie in the browser. This cookie is shared among all the windows of the browser you first logged in with, but not with other types of browsers. For example, if you initially logged in to Maestro using Safari, you can open a new Safari tab or window and continue to use Maestro without logging in again. However, if you try to access Maestro using Chrome, you'll be asked to re-authenticate with your username and password.

Maestro sessions expire

A configuration setting in Transact Manager determines the session validity duration. If your Maestro session expires, Maestro will pop up a new window and ask you to re-authenticate. Once you have re-authenticated, you can go back to the original window and continue working, without losing any unsaved changes.

Related Pages:

- Create a New Project
- Libraries
- Logging In
- Maestro Organizational Structure
- Maestro Projects
- Management Dashboard Overview

Management Dashboard Overview

Unknown macro: 'redirect'

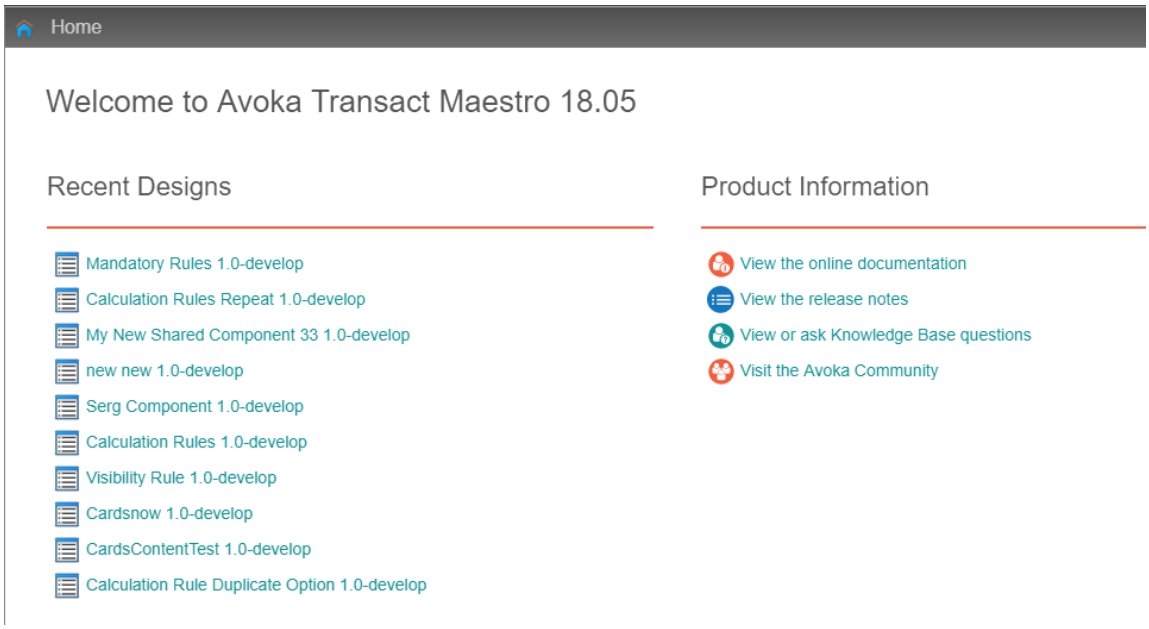
Page Contents:

iRelated Pages:

- Create a New Project
- Libraries
- Logging In
- Maestro Organizational Structure
- Maestro Projects
- Management Dashboard Overview

Overview

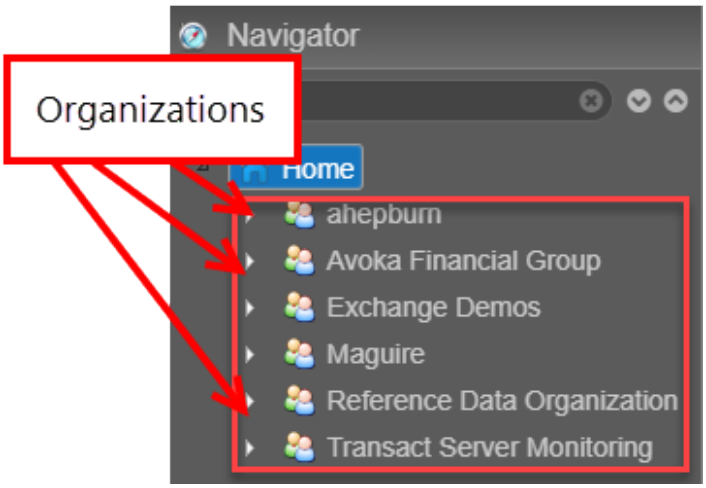
The Management Dashboard is the landing screen that is displayed after a successful log in and allows navigation to Organizations, Projects, Assets and Resources. The Dashboard also allows for easy navigation to recently designed forms and online product resources. The Dashboard is made up of the Navigator Panel and the Detail Panel. The screenshot below displays the Detail Panel showing the Maestro Welcome screen.



The management dashboard reflects the Maestro Organizational Structure.

Navigator Panel

The Navigator panel displays a tree of organizations, projects and the project's assets such as forms, components, and templates.. The content of the Navigator Panel is determined by the user's access rights. The detail panel displays all the information and actions available to the user at the level selected. The screenshot below displays the Navigator Panel with several Organizations. Selecting an organization will display the projects within the organization.

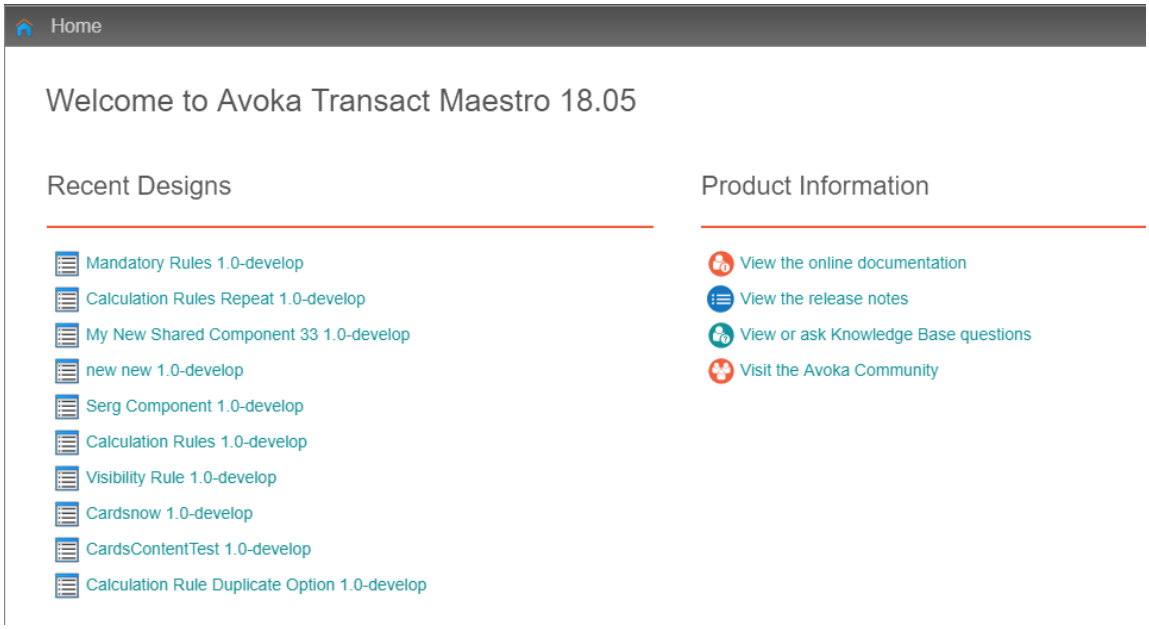


Details Panel

The Details Panel will change depending on the selection made in the Navigator.

Home

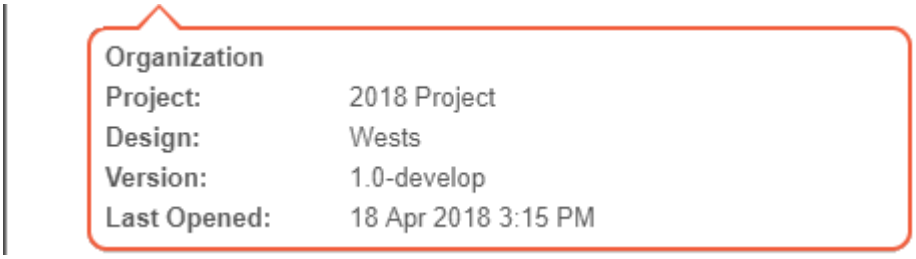
Selecting *Home* from the Navigator Panel will display the Maestro Welcome screen. The Welcome screen contains two sections - Recent Designs and Product Information. The screenshot below displays the Maestro Welcome screen.



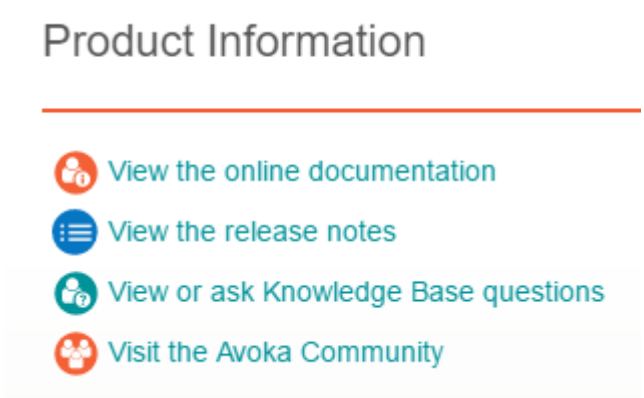
Recent Designs: This list displays up to 10 of the most recently viewed designs (a design can be a form, component or library). Selecting a design from the list will display the latest version details screen. From the Version Details screen, you can edit the design.

You can also view details about the design by hovering your mouse over the design name.

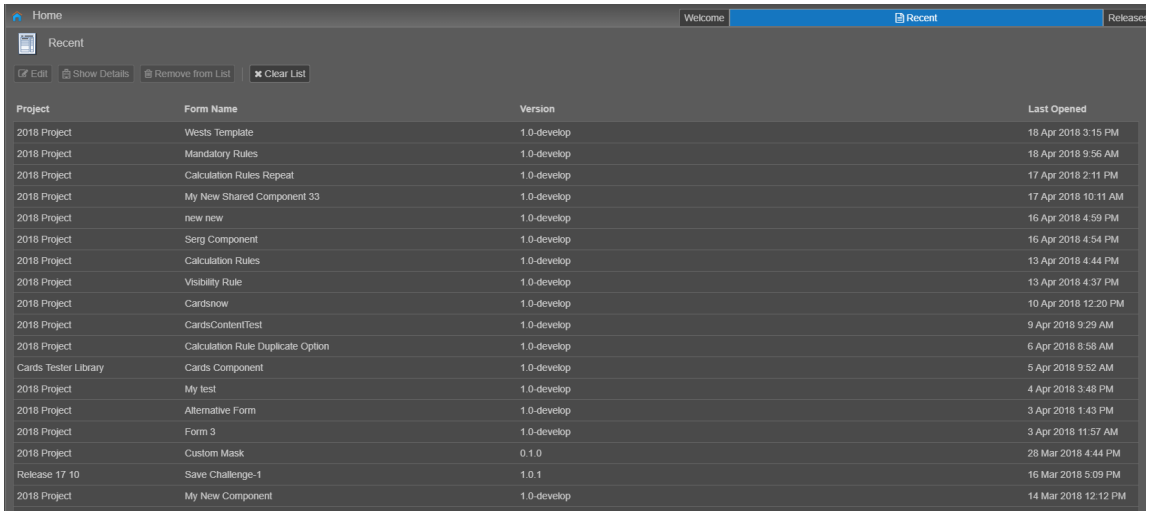
When you hover your mouse over the design name, you will see the Organization and Project where the designtn is stored, along with the design name, the version number, and the date and time the design was last opened.



Product Information: This list will display links to available online resources where you can get information about Avoka products.



The Welcome screen contains two additional tabs - The Recent tab and the Releases tab. The Recent tab allows you to open a selected design in the Editor window, as well as show the design's details. You can also remove a recent design from the list or completely clear the list if desired.



Project	Form Name	Version	Last Opened
2018 Project	Wests Template	1.0-develop	18 Apr 2018 3:15 PM
2018 Project	Mandatory Rules	1.0-develop	18 Apr 2018 9:56 AM
2018 Project	Calculation Rules Repeat	1.0-develop	17 Apr 2018 2:11 PM
2018 Project	My New Shared Component 33	1.0-develop	17 Apr 2018 10:11 AM
2018 Project	new new	1.0-develop	16 Apr 2018 4:59 PM
2018 Project	Serg Component	1.0-develop	16 Apr 2018 4:54 PM
2018 Project	Calculation Rules	1.0-develop	13 Apr 2018 4:44 PM
2018 Project	Visibility Rule	1.0-develop	13 Apr 2018 4:37 PM
2018 Project	Cardsnow	1.0-develop	10 Apr 2018 12:20 PM
2018 Project	CardsContentTest	1.0-develop	9 Apr 2018 9:29 AM
2018 Project	Calculation Rule Duplicate Option	1.0-develop	6 Apr 2018 8:58 AM
Cards Tester Library	Cards Component	1.0-develop	5 Apr 2018 9:52 AM
2018 Project	My test	1.0-develop	4 Apr 2018 3:48 PM
2018 Project	Alternative Form	1.0-develop	3 Apr 2018 1:43 PM
2018 Project	Form 3	1.0-develop	3 Apr 2018 11:57 AM
2018 Project	Custom Mask	0.1.0	28 Mar 2018 4:44 PM
Release 17 10	Save Challenge-1	1.0.1	16 Mar 2018 5:09 PM
2018 Project	My New Component	1.0-develop	14 Mar 2018 12:12 PM

The Releases tab displays updates to the libraries within Maestro. This includes libraries such as Cards, Core, etc.

Release: 18.04.1

Name	Description	Imported At
cards	Card Widgets	8 Mar 2018
core	Core Maestro Widgets	8 Mar 2018
identity	Identity Widgets	8 Mar 2018
maguire	Maguire Template Assets	8 Mar 2018
navigation	Navigation Widgets	8 Mar 2018

Release: 18.04.0

Name	Description	Imported At
cards	Card Widgets	27 Feb 2018
core	Core Maestro Widgets	27 Feb 2018
identity	Identity Widgets	27 Feb 2018
maguire	Maguire Template Assets	27 Feb 2018
navigation	Navigation Widgets	27 Feb 2018

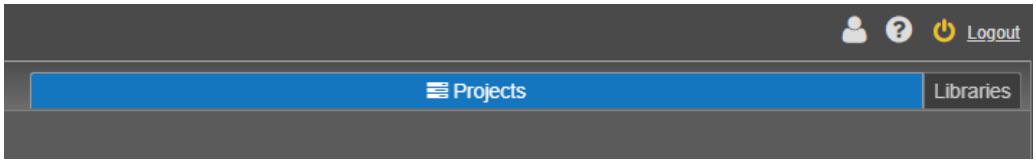
Levels within Maestro

There are three levels in the Maestro Organization Structure.

- Organization
- Project
- Form

Organization Level

The Organization Level describes assets and resources available to the Organization. There are two tabs available at the Organization level - Projects and Libraries.



1) **Projects**: List all the projects available to the selected Organization.

Actions:

- Create Project: This allows you to create a new project within the selected organization.
- Delete: This will delete the selected project. When you delete a project you delete all of the assets and resources within the project.
- Import Project: You can import a project that has been exported. This is useful when you want to move projects between organizations.
- Export Project: You can export a project so that you can import it to a different project. You can also export a project to save it to your computer as a backup.
- Import Legacy Project: You can import a legacy (prior to the current verison of Maestro) project into Maestro.

Selecting a Project will direct you to the Project Level. The project level is where you can control assets and resources that are associated with a specific project.

2) **Libraries**: A list of the libraries available to the organization and their origin. Libraries stored at the organization level will be available to all projects within the organization.

Actions:

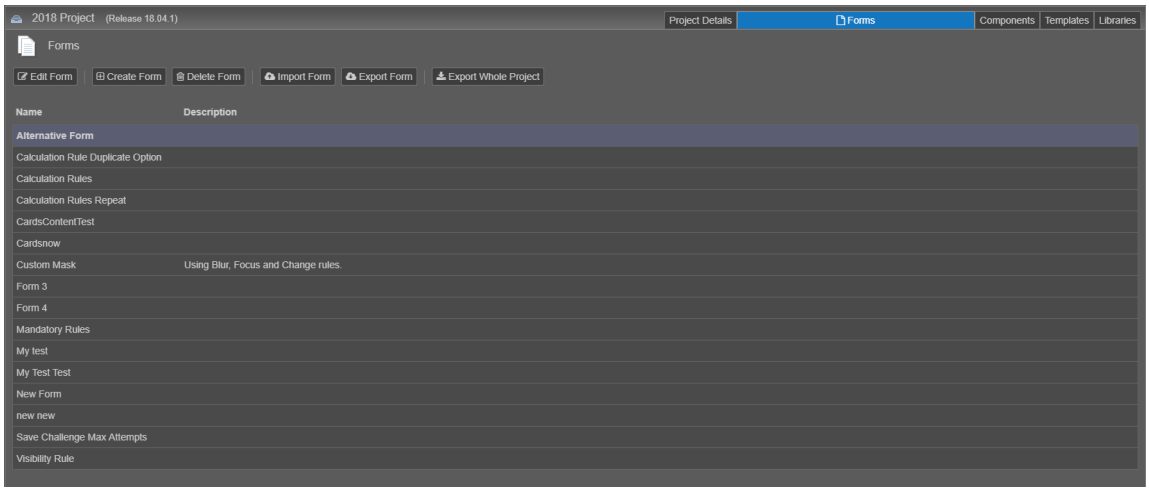
- Create Library: This allows you to create a new organization library. Organizations can have as many libraries as needed.
- Import: This allows you to import libraries from other projects to the organization level.

Project Level

The project level is accessed by selecting a project from the Maestro Navigator Panel. The project level consists of five tabs (as shown in the screenshot below). The default view at open is the Forms tab.

The available tabs are:

- Project Details
- Forms
- Components
- Templates
- Libraries



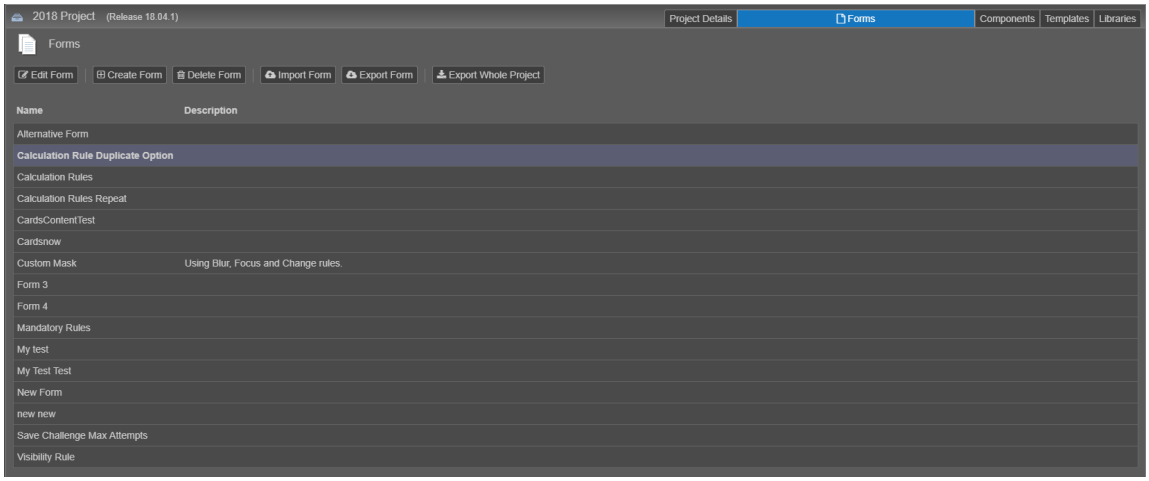
The Forms, Components and Templates folders (or tabs) are the development areas of the corresponding asset (e.g, The Forms tab is where you develop and edit forms). The Libraries tab is where the published versions of the assets and resources are stored. See Maestro Projects for more information.

1) **Project Details tab**: Displays and allows updates to the project name and description. You can also set the default template for the project. The default template will be selected when a new form is created in the selected project. However, a different template can be selected if needed. It is from this tab that you can create a clone of the form version to use in Source Code Management tools such as GIT to manage Maestro assets. To clone a form version, copy the SCM clone value and use it as a command in a Source Code Management tool such as GIT.

2) **Forms tab**: Displays a list of all available forms in the project.

Actions:

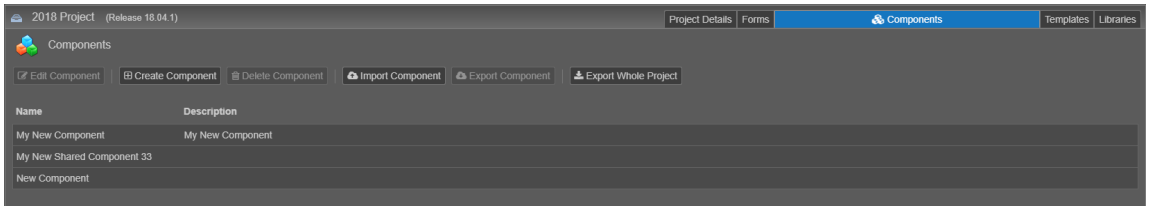
- Edit: This opens the most recent version of the form.
- Create Form: This allows you to create a new form in the project.
- Delete Form: This deletes the selected form and all of it's versions.
- Import Form: This allows you to import a form from a different template. You must first export a form.
- Export Form: This allows you to export a form. You can then import into a different project, import into the same project to create a copy, or save to your computer as a backup.
- Export Whole Project: This allows you to export the whole project (this includes all forms in the project and all assets and resources in the project)



3) **Components tab**: Displays a list of all editable components in this project. Note that these are not available to forms until they are published to a library.

Actions:

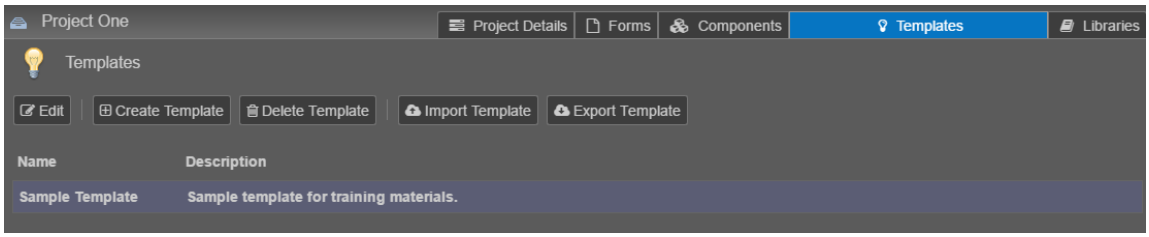
- Edit: This opens the most recent version of the component in the editor window.
- Create Component: This allows you to create a new component within the project.
- Delete Component: This allows you to delete the selected component and all of it's versions.
- Import Component: You can import a component that has been exported from a different project or from the same project.
- Export Component: You can export a component that can then be imported into a different project, the same project to create a copy or saved to your computer as a backup.
- Export Whole Project: This allows you to export the whole project (this includes all forms in the project and all assets and resources in the project)



4) **Templates tab**: Displays a list of all editable templates in this project. Note that these are not available to forms until they are published to a library.

Actions:

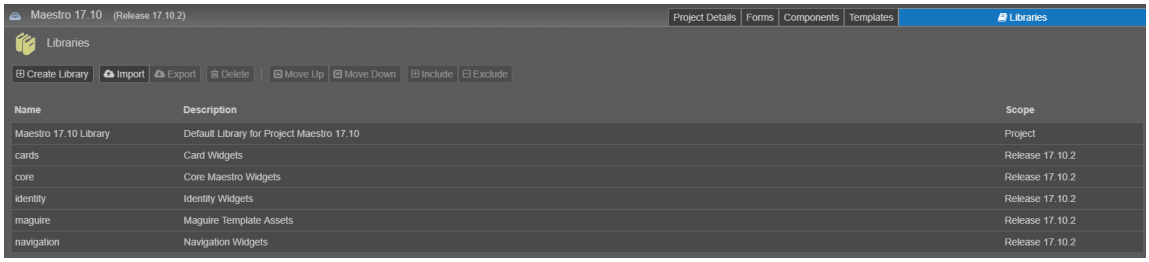
- Edit: This opens the most recent version of the template in the editor window.
- Create Template: This will create a new template within the project.
- Delete Template: This will delete the selected template and all of it's versions.
- Import Template: You can import a template that has been exported from a different project or from the same project which will create a copy.
- Export Template: You can export a template that can then be imported into a different project, the same project to create a copy or saved to your computer as a backup.
- Export Whole Project: This allows you to export the whole project (this includes all forms in the project and all assets and resources in the project).



5) **Libraries tab:** Displays all the libraries that the project can access. These can be local to the project or can be inherited from a higher level in the structure.

Actions:

- Create Library: This will allow you to create a new library within your project. Projects can contain as many libraries as needed.
- Import: This will allow you to import a library that has been exported from a different project.
- Export: This will allow you to export the selected library.You can then import the library to a different project or save to your computer as a backup.
- Delete: This will delete the selected library. You should be very careful when deleting libraries and this may effect forms that are using the assets and resources from this library.
- Move Up/Down: This will change the order of precedence. The library at the top of the list will be used first.
- Include/Exclude: It is possible to deactivate the library in the project and reactivate it again. The excluded library assets and resources will not be available for the project.



Maestro Organizational Structure

Unknown macro: 'redirect'

Page Contents:

Related Pages:

- Create a New Project
- Libraries
- Logging In
- Maestro Organizational Structure
- Maestro Projects
- Management Dashboard Overview

Overview

Maestro organizations are defined in Transact Manager and appear in the user's Management Dashboard at login. A organization in Maestro is a folder that holds projects and organization specific assets such as libraries (though these can also be held at the project level and be project specific).

The high level structure of Maestro can be broken down into two core concepts.

- **Organizations**: Organizations contain projects and Organization level libraries (organization level).
- **Projects**: Projects contain forms, components, templates and libraries (project level).

It is important to note that templates and components are not available to be used in forms until they are published to a library. Libraries are displayed in the dashboard tree (even if they have not been published) so that they can be easily created and edited.

Organization Level

Organization Level

The Organization level is the level designation for assets and resources that are available to the entire Organization. This means that All projects in the organization have access to the assets and resources identified at the Organization level.

Project Level

Project Level

The project level is the level designated for assets are resources that are applied to individual projects. This differs from the organization level as organization level assets and resources are available to ALL projects in the organization.

Forms, Components and Templates

All Project level Maestro resources and assets are stored in one of the three folders within the project – Forms, Components or Templates.

- Resources include styles, images, fonts and scripts.
- Assets include forms, components and templates.

The Forms, Components and Templates folders are the development areas of the corresponding asset. These assets can be edited and updated without impacting published versions. You can push the changes made to the published versions by re-publishing the asset or resource.

Libraries

Published assets and resources are placed into libraries within the libraries folder. Libraries are found at the organization and the project level.

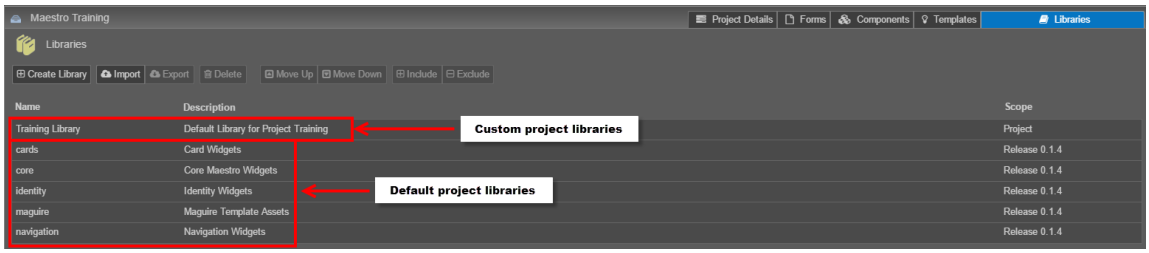
Organization level libraries will be available to all projects within the organization whereas project level libraries are only available within the project.

The image below shows the project libraries.

Every new project created in Maestro contains five default libraries – Cards, Core, Identity, Maguire and Navigation. These libraries contain all of the standard Maestro resources and assets.You cannot edit these libraries, they are only available to provide assets and resources to your project.

Any custom project libraries created will also be listed here. You can create new libraries as needed to store your published assets and resources. Organizations and projects can have as many libraries as needed.

See Libraries for more information.



Libraries

Unknown macro: 'redirect'

Related Pages:

- Libraries

Page Contents:

Overview

A Transact Maestro library is a collection of non-volatile resources (styles, images, fonts and scripts) and assets (components and templates) packaged in a ZIP file, which can be made available to Maestro forms, components and templates. There are two types of libraries in Maestro:

- Default Project Libraries
- Customer Project or Organization Libraries

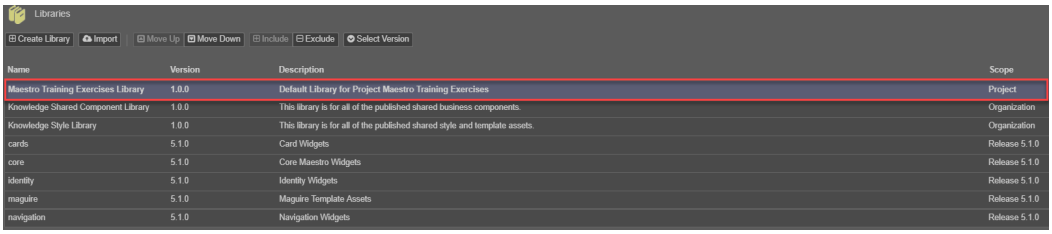
A library must be published in a correct library structure (see Maestro Organizational Structure) first, so a Maestro developer can use the library's shared resources and assets while working on a form, component or template.

If a resource or asset is not in a library, it will not be found and, therefore, cannot be used across one or more different forms in Transact Maestro.

Library Content

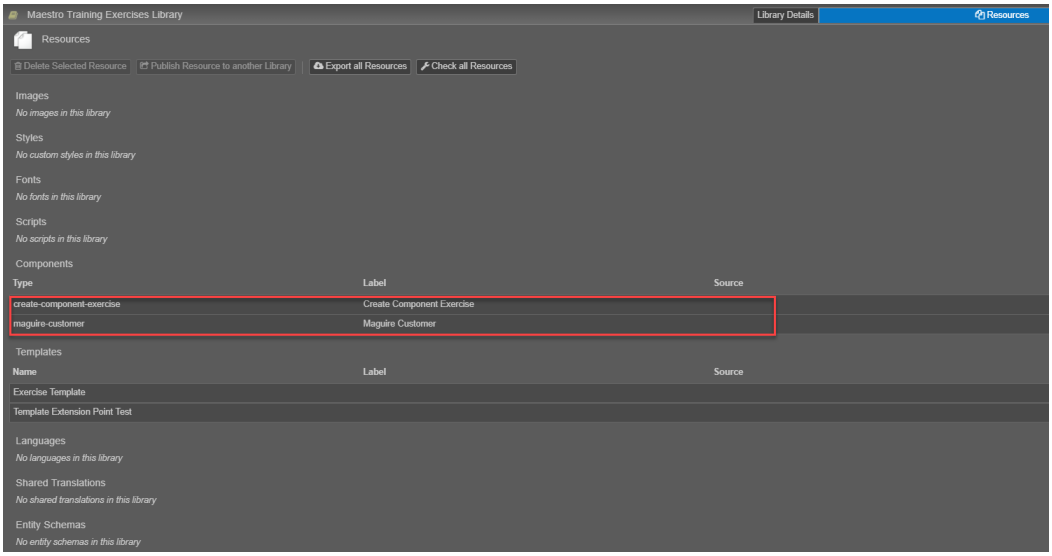
Library content is made up of resources and assets that are stored in the library ZIP file. You can see the list of customer library resources if you follow the steps below:

- Select the library

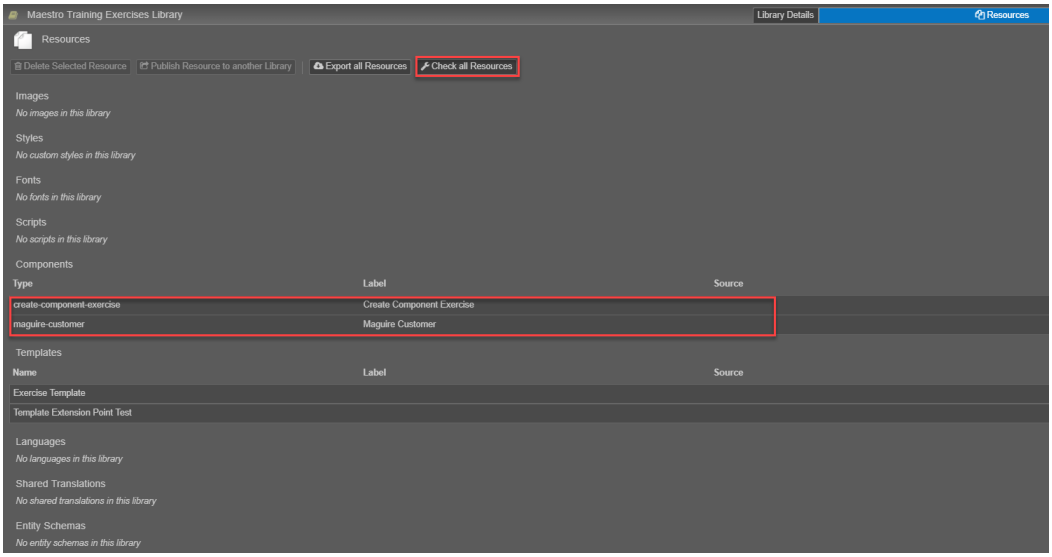


- Double-click on the selected library. Click on the Resources tab in the new window.

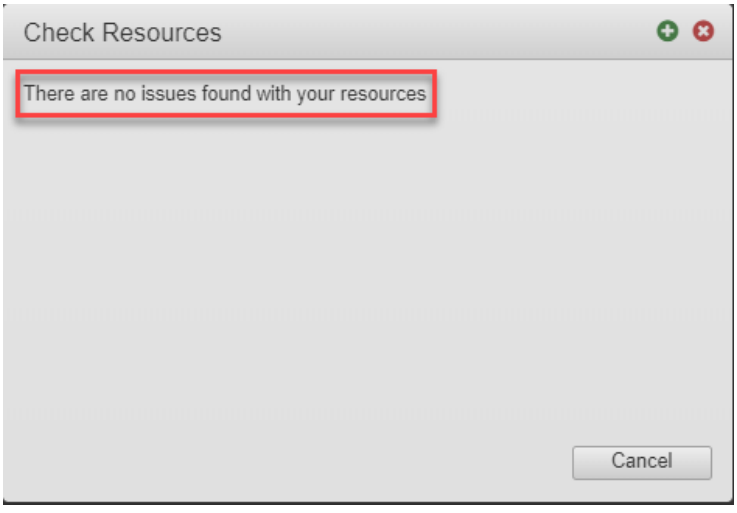
- Review the published resources in this library



- Click the Check all Resources button to see if there are any invalid resources in the library. Maestro will run a set of validation scripts to ensure that styles and scripts have valid structure and syntax. You may want to do this if some resources are not being included properly and to ensure that there are no issues with the resources in the library.



- The success or warning message will be displayed.



You cannot see the list of Default Library resources. Read more on Default Libraries below.

Library content is available as a flat structure to any Form, Component or Template that uses this library. This means:

- There is no association between content and a library file it has come from. In other words, you cannot say in which library a particular resource, for example style, is defined.
- Content can be easily overwritten by another component or template during publishing.

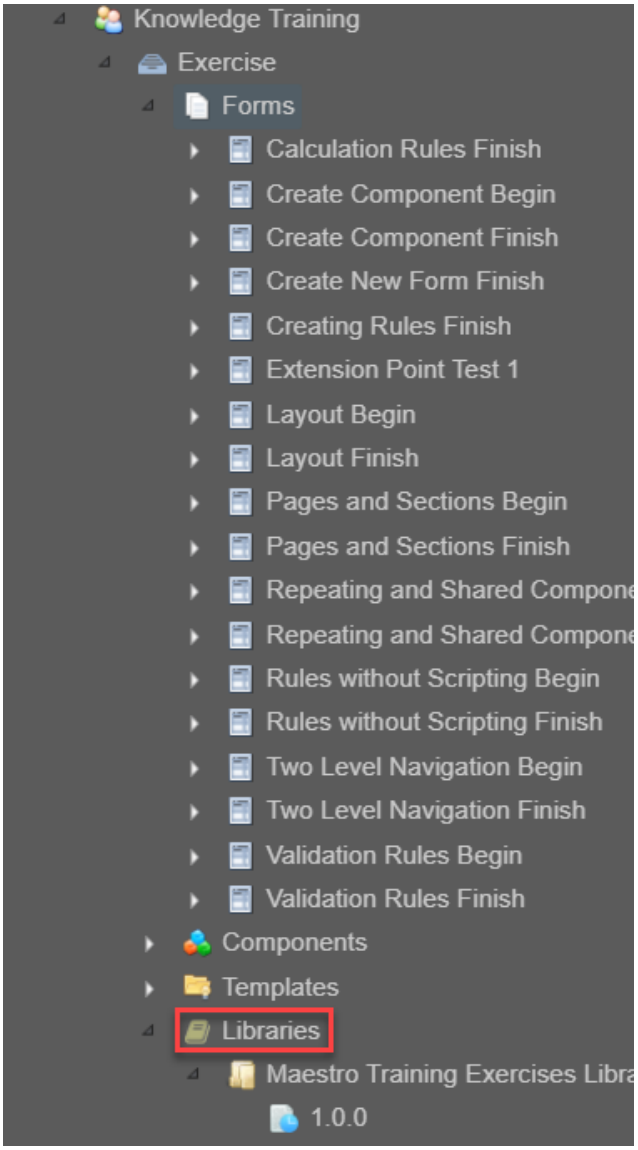
To illustrate this, imagine there are two styles using the same name created in a template and a component. They have different background color values, for example red and green. However, a Form will have no mechanism to distinguish between the two of them, so the first style, loaded by this Form, will be used. Let's say it is the style with the green background. Further, assume that the Component's content is changed and thus published to the existing library. This will overwrite the Template style previously published in the same library. As a result, the Form will have its background color as red after refreshing its content from the newly published library. To avoid this situation, please read Library Structure and Libraries Best Practices.

> Do not update styles on your Form unless it is really necessary and you keep track of the changes, because styles must be updated in a Template.

Library Structure

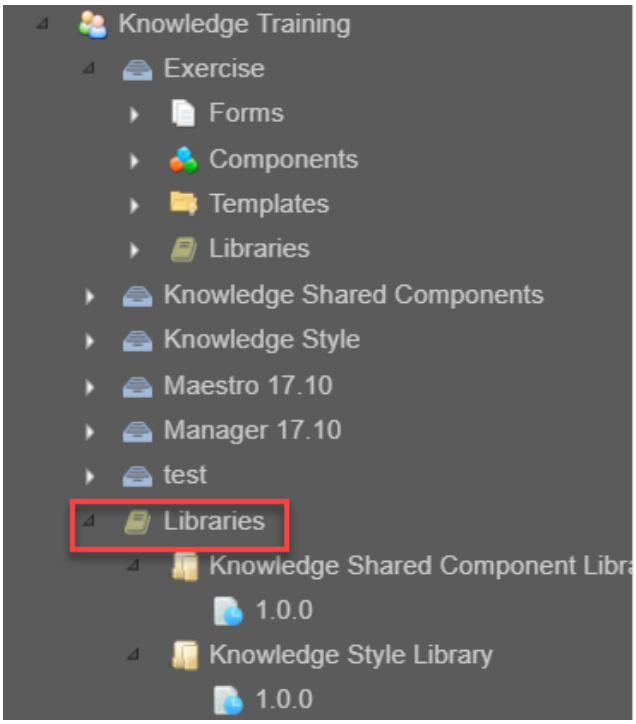
You can publish a Template and Component into a library under:

- A Project's library directory, which makes the library content available to all Forms, Components and Templates under this Project.
  In this example, the Maestro Training Exercises Library is available to all Forms.



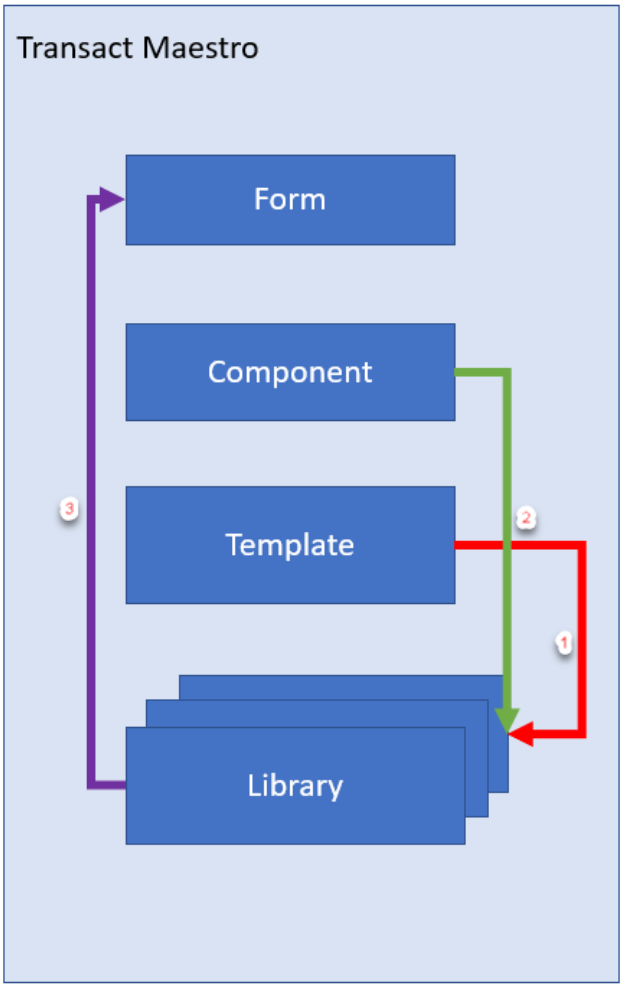
- An Organization's library directory, which makes the library content available to all Project Forms, Components and Templates.

  In this example, the Knowledge Shared Component Library is available to all Projects under Knowledge Training.



It is advised to publish a Template and a Component to a Library first, so a Form can make use of this Library, as all shared resources and assets will be available to this Form. This is illustrated in the diagram below where:

1. Template's content is published in the library first.
2. Component's content (e.g. address, contact details, etc.) is published in the library.
3. Form uses template's and component's content via the published library.



You cannot publish a Form into a library. Forms are not shareable resources in Maestro.

Default Libraries

Every new project created in Maestro will have five default libraries, which contain all of the standard Maestro resources and assets:

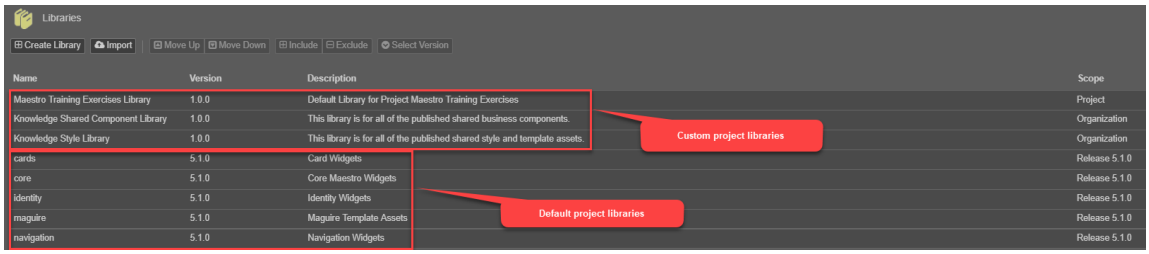
- Cards
- Core
- Identity
- Maguire
- Navigation

cards	5.1.0	Card Widgets	Release 5.1.0
core	5.1.0	Core Maestro Widgets	Release 5.1.0
identity	5.1.0	Identity Widgets	Release 5.1.0
maguire	5.1.0	Maguire Template Assets	Release 5.1.0
navigation	5.1.0	Navigation Widgets	Release 5.1.0

The order of the default libraries is important.

You cannot edit these libraries.

Any custom project libraries will also be listed under the library directory.



You can create many libraries to publish and re-use your assets and resources. There is no limit for the number of libraries an Organization and a project can have.

It is recommended to store style-related assets and resources and business components in different libraries. This allows you to keep elements organized and makes it easier to manage.

When publishing assets, not only should you think about which library to use, but you should also think about where your library should be stored, see Library Structure. Corporate styling and business components should be published to an organization library. This will give all projects within the organization access to the library, which will allow assets and resources to be used without being recreated for each project.

Project specific resources and assets should be stored in project level libraries. This may include resources related to a specific region or department which are not relevant to other projects.

The management of the libraries is important in order to enable sharing of assets and resources across the Maestro organization and avoiding shared resources been overwritten.

Library Precedence

The order that the libraries are listed is critical when it comes to assets and which ones get applied to your forms. The precedence order is from top to bottom. Let's say that you have created a style and published to one of your custom libraries. There is a default style with the same name in the Core library. **The library that is listed first in the library list will be used in your forms.**

Name	Description
Training Library	Default Library for Project Training
cards	Card Widgets
core	Core Maestro Widgets
identity	Identity Widgets
maguire	Maguire Template Assets
navigation	Navigation Widgets

You can change the order of the libraries by using the Move Up and Move Down buttons.

Most of the time your custom libraries will be listed first as they contain the assets and resources that you have specifically created.

If there is an organization library that you don't want to have available within a project, you can use the Exclude button. The Include button is available to reinstate any libraries that have been excluded.

The excluded library displayed in the list will change to red. This indicates that the resources and assets within that library are not going to be available for this project.

Name	Description
cards	Card Widgets
core	Core Maestro Widgets
identity	Identity Widgets
maguire	Maguire Template Assets
navigation	Navigation Widgets
Training Library	Default Library for Project Training

> Using the Exclude and Include buttons will change the library order, so you may need to re-order the libraries manually afterwards.

Import and Export Libraries

You can import and export libraries to other projects and/or organizations. You may also want to export the library to keep a backup copy of the published assets and resources. See Import and Export for more information.

> Import and Export operations may change the library order, so you may need to re-order the libraries manually afterwards.

Library Versions

Introduction

Maestro libraries support semantic versioning of three levels. This means that there can be multiple versions of the same library. Adding multiple versions of the same library can have the following benefits:

- Standardizing versioning of all Maestro design artifacts and making management of different versions easier for the end user.
- Aligning Maestro artifacts closer to the standard Transact versioning scheme.

- Assisting form builders who are developing forms concurrently.
- Locking down versions of the design artifact.

Why Use Library Versions?

- Allows you to return to a previous version of the library. This means that you can make incremental changes to a library version without affecting organizations and/or projects.
- Allows you to control when an organization/project library is upgraded.

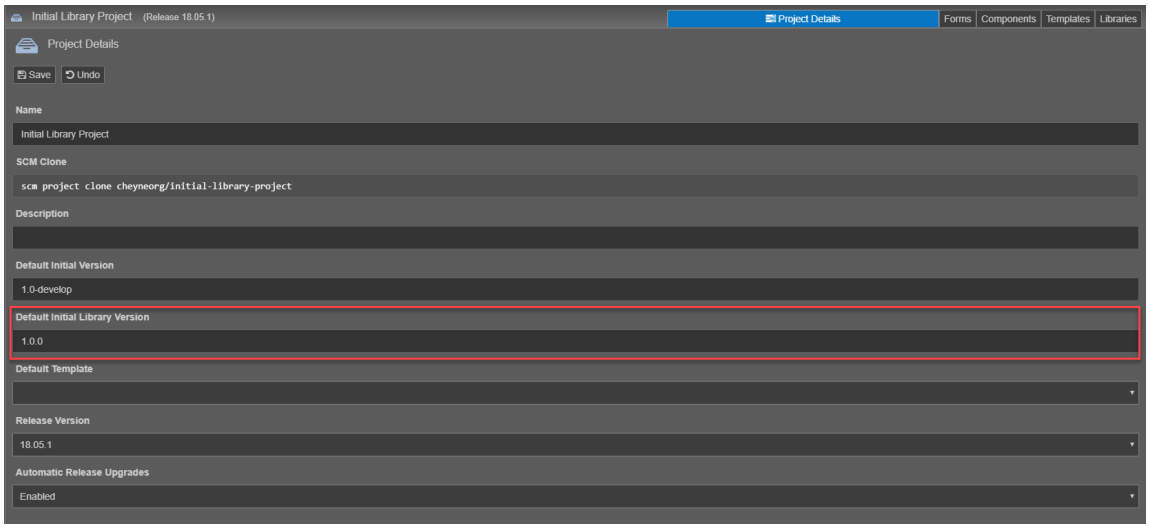
Set Default Initial Library Version

The default initial library version is the version that is the base for all other versions that are created for the project.The default initial library version is set at the project level.

Why set the Default initial Library Version

Setting the default initial library version is useful as it gives you a standard to base all newly created library versions.

To set the Default Initial Library Version, select a project and switch to the Project Details tab. The screenshot below displays this tab and highlights the Default initial Library Version tab. You can enter any version number into this field but it is recommended that you stick to a 1.0.0 library version until you need to create a new library version.



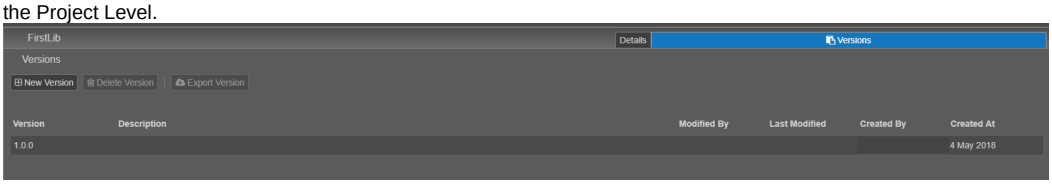
Once the default initial value has been set, every time you create a new library, the first version will be based on the value identified as the Default Initial Library Version.

Create a new library version

When a new library is created, a default version of the library is created. This default version is normally 1.0.0. As this library grows and expands, it may be necessary to create a new version of the library.

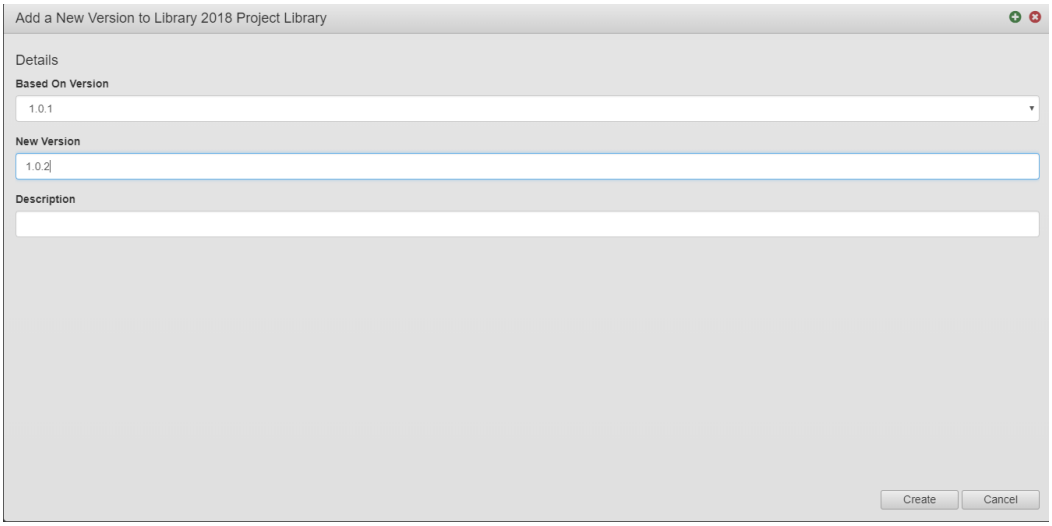
Follow the steps below to create a new version of a Maestro library (library versions can be created at both the organization and project levels).

1. Navigate to the Libraries folder in the Maestro Navigator tree of a project
   Navigating to the libraries folder will display the currently configured libraries for the selected project and any versions associated with each library. By default, when a new library is created, a default version of that library is created. The screenshot below displays a library named FirstLib with version number 1.0.0. This new version is based on the Default Initial Value that is defined at the Project Level.



2. Select the library that you want to create a new version for

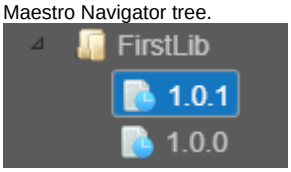
3. Click *New Version*
   Clicking *New Version* will display the Add a New Library Version dialog. The screenshot below displays this dialog.



Based on Version - This dropdown selection will identify which currently configured version of the selected library will be the base for the new version that is being created. This means that everything associated with the version that is selected as the *Based On Version* will be copied over to the new library version.
New Version - This is the name of the new library version. By default, this name will increment based on the previous version of the library.
Description - This is a non mandatory field that can be used to provide extra information that can be used to identify the new library version.

4. Complete the Add a New Library Version dialog by selecting a version to base the new library version on, and providing a name for the new library version.

5. Click *Create*
Clicking Create will create the new library and display it in descending order in the Maestro navigator structure. The screenshot below highlights the placement of the new version in the Maestro Navigator tree.



Delete a library version

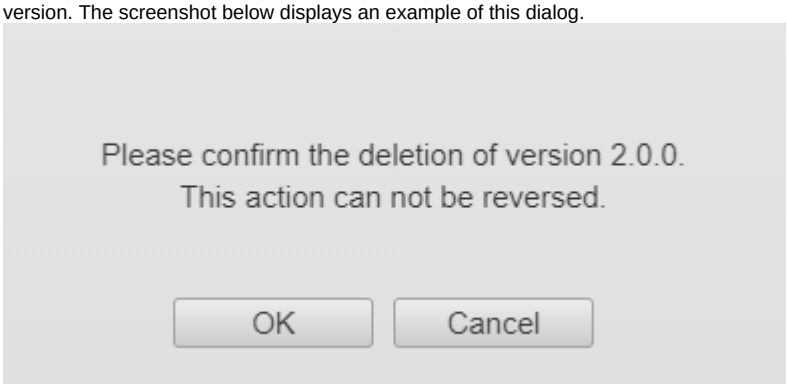
Just as Library versions can be created, they can also be deleted. However, only one library version can be deleted at a time (library versions can be deleted at both the organization and project levels).

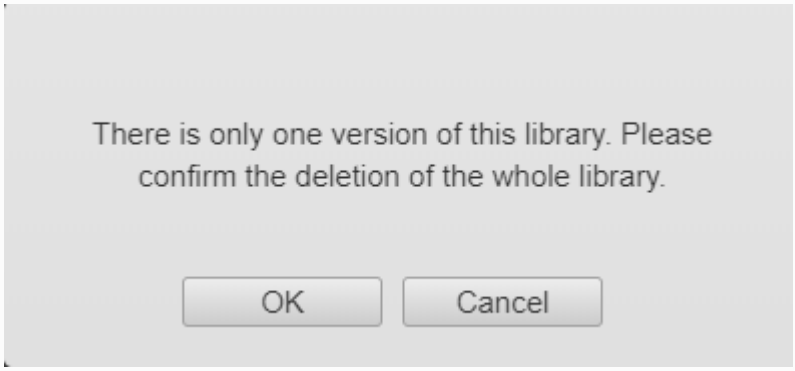
Follow the steps below to delete an individual version of a Maestro library.

1. Navigate to the Libraries folder in the Maestro Navigator tree

2. Select the library that contains the library version that you want to delete
Selecting the library will display the list of library versions that are associated with the selected library. The screenshot below displays the different library versions for the selected library.

3. Select the version you want to delete and click *Delete Version*
   Clicking *Delete Version* will open a dialog asking you to confirm the deletion of the selected library version. The screenshot below displays an example of this dialog.



> If there is only one version of a library, when you attempt to delete that one version, the following dialog will display asking you to confirm the deletion of the whole library (by deleting the only version of the library, you are deleting the whole library).
>
> 
>

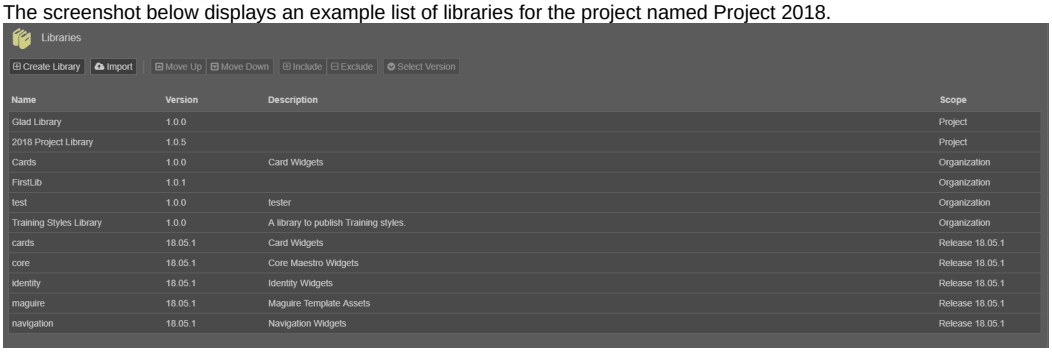
Select/Map A Library Version To A Project

> **Why Should You Map A Library Version To A Project?**
>
> Mapping a library version to a project will make the mapped library version the **ACTIVE** version of the selected project. The active library version is the version that the selected project will use.

When multiple versions of the same library have been created, it is important to select and map the correct library version to the current/correct project. Mapping a library version to a project means that the forms, components and templates associated with that project will have access to the mapped library version.

Follow the steps below to map a library version to a project.

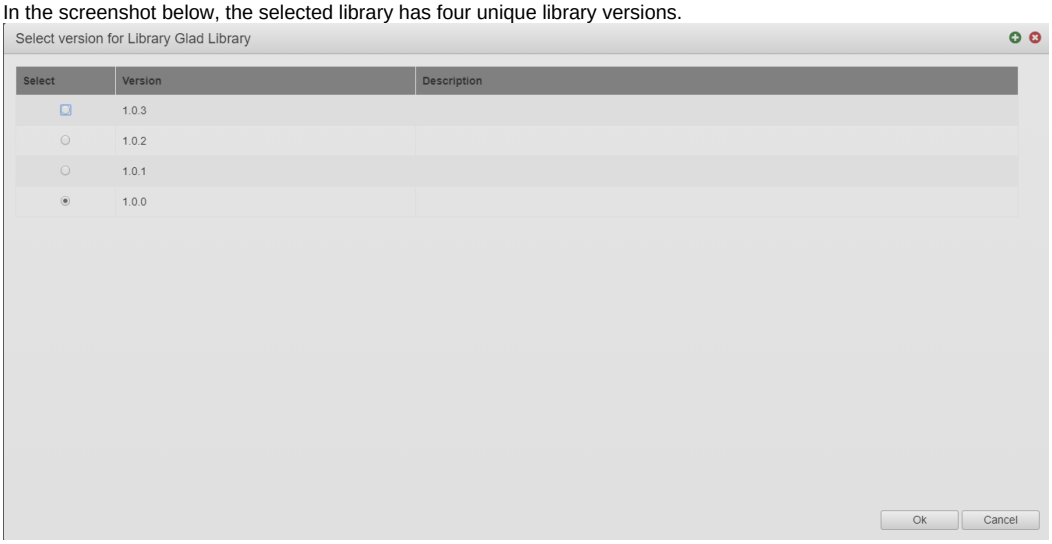
1. Navigate to a project and switch to the libraries tab
   The libraries tab displays all of the libraries that have been created for the selected project.
   The screenshot below displays an example list of libraries for the project named Project 2018.



> **Include and Exclude**
>
> At this screen you can exclude and if needed include (after excluding) a library version. A project library version that is excluded will not be available for selection and will display in red.

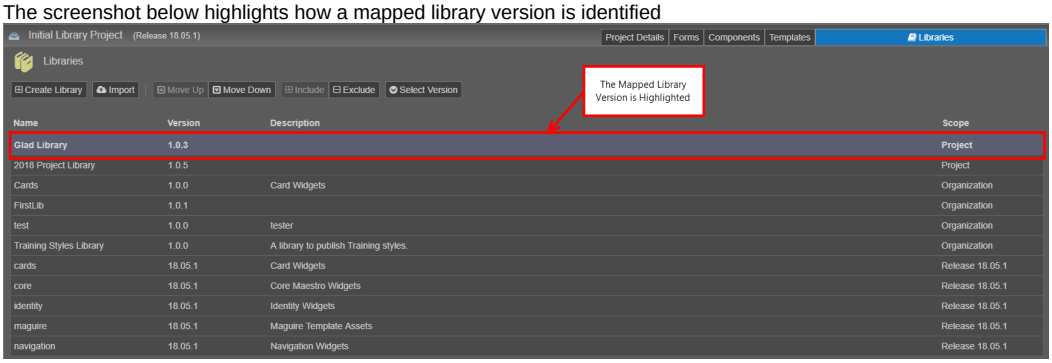
2. Select the library that contains the library version that you want to map to the selected project.

3. Click *Select Version*
   Clicking *Select Version* will display the list of library versions for the selected library.
   In the screenshot below, the selected library has four unique library versions.



4. Select the library version that will be mapped to the selected project.

5. Click *OK*
   Clicking *OK* will map the selected library to the selected project. The mapped library version for a project is identified by being highlighted in the list.
   The screenshot below highlights how a mapped library version is identified



Import and Export a Library Version

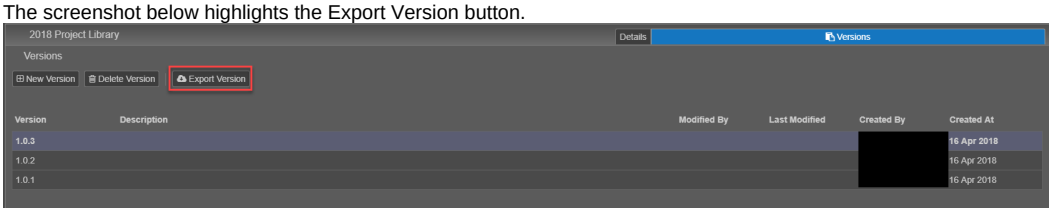
In order for library versions to be functional across multiple environments and systems, Maestro allows you to export library versions and import them into a different project or organization.

> **Why Should You Import and Export A Library Version?**
>
> The Import and Export library versions functionality should be used to move library versions from one project/system to a different project or organization.

> You cannot export and import a library version back into its original library. If you try to import a library verison back into its original library, an error message will display.

To export a library version (library versions can be exported at both the organization and project levels):

1. Select a library version at the project level
2. Click *Export Version*
   The screenshot below highlights the Export Version button.

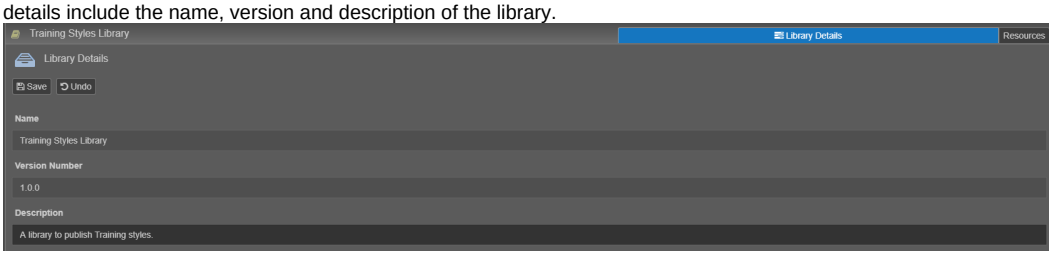


   Clicking Export Version will export the selected library version (in a zip file) and download it to your computer.

To import a library version (library versions can be imported at both the organization and project levels):

1. Open the Libraries folder in the selected project or organization
2. Click *Import*
   The screenshot below displays the libraries screen and highlights Import.

3. Select the exported library version zip file from your computer
   Successful import of the library version will display the details of the library versions. These details include the name, version and description of the library.



Publishing Shared Components to a Library Version

When publishing shared components, you will be given an option to publish the shared component to a specific library version. This means that the shared component will be made available and associated to the selected mapped library version. The screenshot below displays the publishing screen. When this screen displays, you can select the library that the component will be associated with. When you select the library, the mapped library version will also be selected.

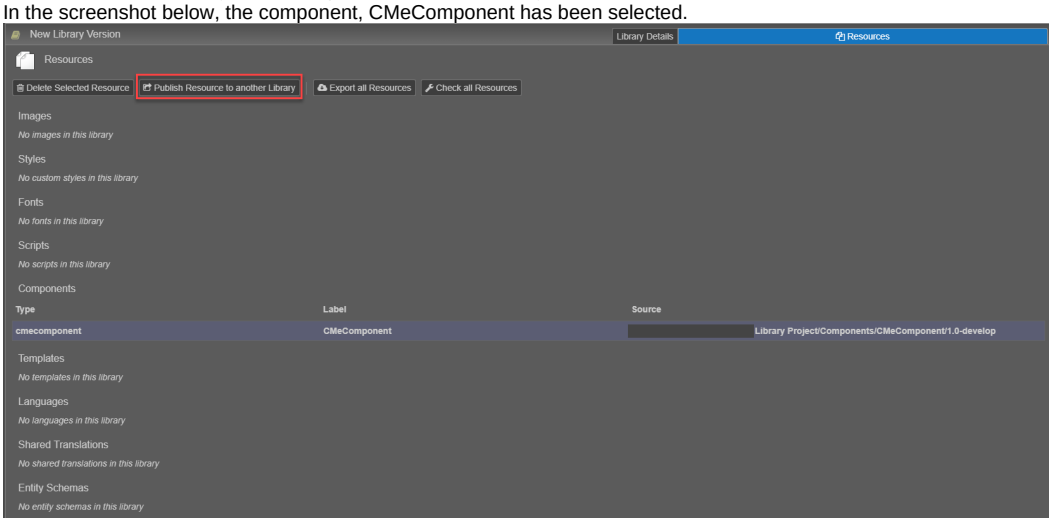
Publish Resources From Another Library

Once a resource such as a Shared Component has been published to a library version, you can publish the resource to a different library version.

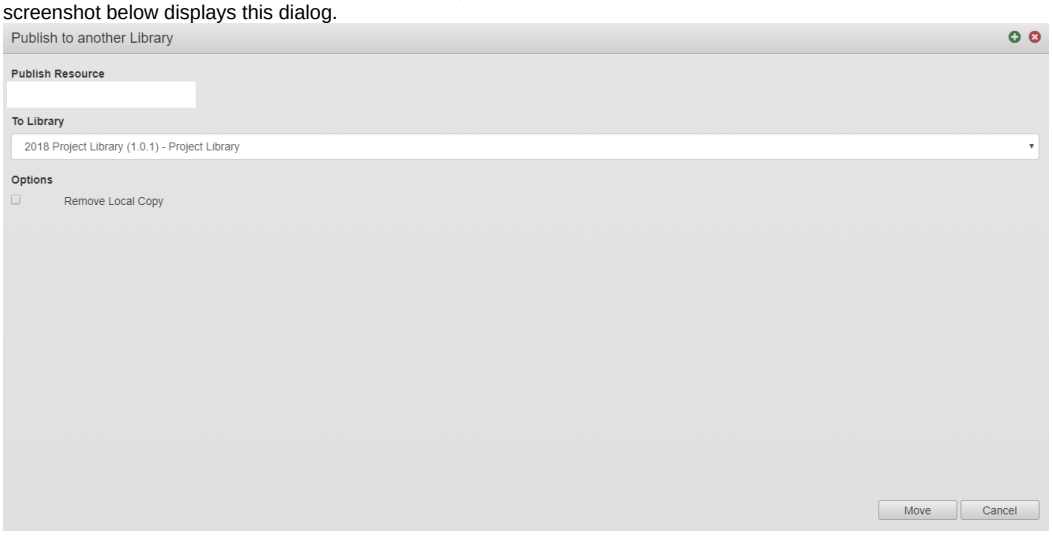
To publish a resource to another library:

1. Select a library version from the Maestro Navigator
2. Switch to the *Resources* tab

3. Select the resource that you want to publish to another library.
   In the screenshot below, the component, CMeComponent has been selected.



4. Click *Publish Resource to another Library*
   Clicking *Publsh Resource to another Library* will display the Publish Library dialog. The screenshot below displays this dialog.



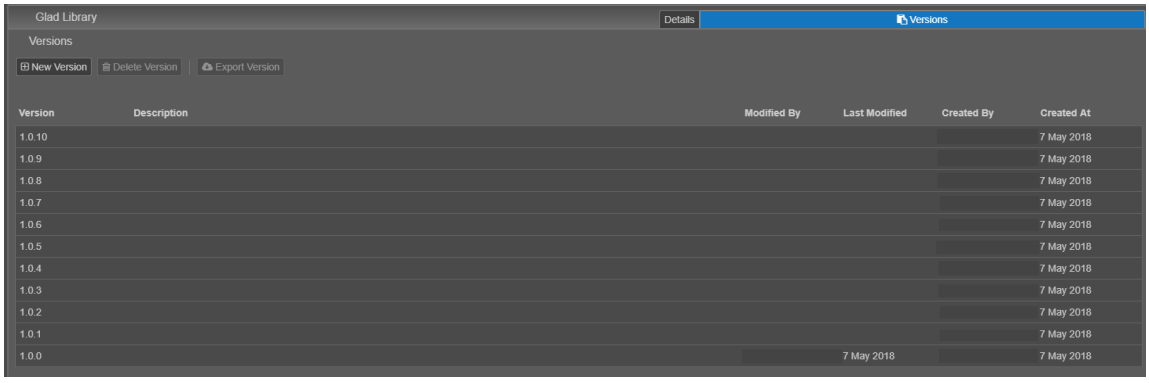
5. Select the library that the resource will be published to.

> If you want to move the resource completely away from the library where it is currently published, select the *Remove Local Copy* checkbox. Selecting this checkbox means that the selected resource will only be available to the library version that you are publishing to now.

6. Click *Move*

Sorting

Library versions are sorted based on major, minor and patch releases (e.g. 1(major).1(minor).3(patch)) and are displayed in the Maestro interface in descending order.The screenshot below displays a list of library versions being displayed in descending order.



Maestro Projects

Unknown macro: 'redirect'

Projects

- Create a New Project
- Upgrade Project
- Automatic Release Upgrades

Page Contents:

Related Pages:

- Create a New Project
- Libraries
- Logging In
- Maestro Organizational Structure
- Maestro Projects
- Management Dashboard Overview

Overview

Maestro is structured as organizations that contain projects. Within projects, you will find related assets and resources, such as forms, templates, components, styles, etc.

Maestro organizations can contain as many projects as necessary. Within your company, you can decide how to use organizations and projects. You may have one Organization that contains Projects for each region or department. Or you may decide to have an organization for each region and a project for each department. There are multiple options available for you to structure Maestro within your company.

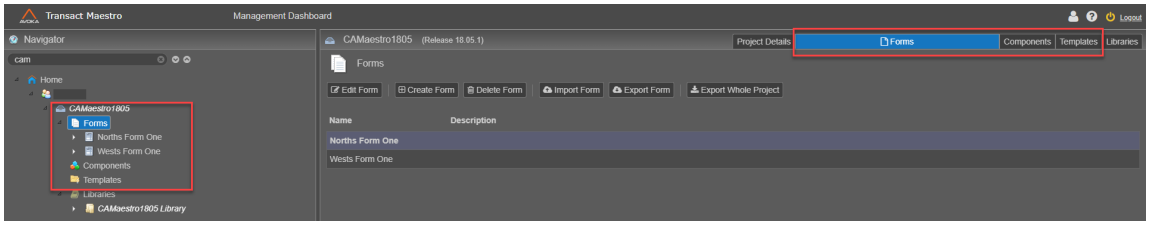
Projects contain the assets and resources needed to create Maestro forms. The forms, components and templates folders (or tabs) are the development areas of the corresponding asset. The library is where the published versions of the assets and the resources are stored.

The assets within the development areas can be edited and updated without impacting published versions. You can push the changes made to the published versions by re-publishing the asset or resource.

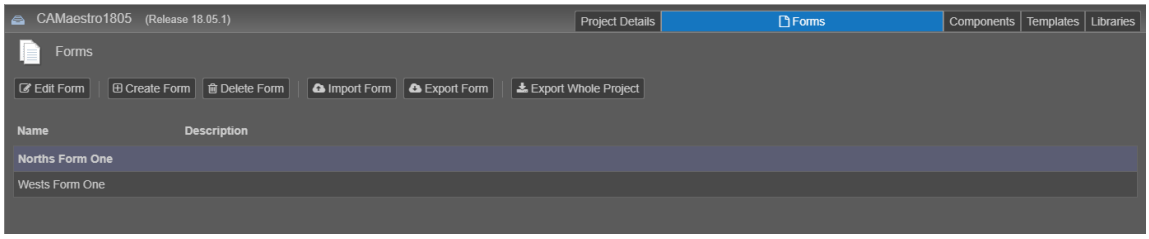
Project Details tab: This tab displays information about the selected project. You can update the project name and description, as well as set the default template to be selected when new forms within the project are created. When creating a new form, the form builder will be able to select a different template if needed. This tab also displays the release version of the project and identifies whether Automatic Release Upgrades have been enabled. For more information on Automatic Release Upgrades, please see the Automatic Release Upgrades section within the documentation.

Resources and Assets

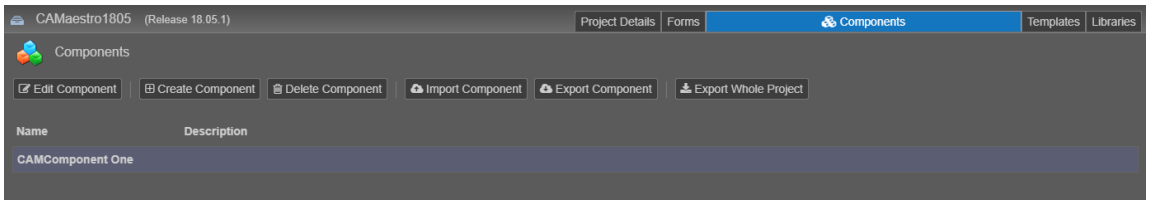
You can access the areas within the project by selecting the folders from the Navigator panel or by using the tabs at the top of the panel window.



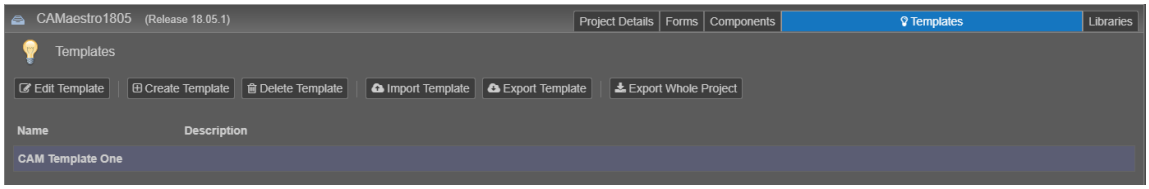
Forms: This tab displays a list of all available forms in the project. Remember, this folder is the development area of forms that are being created. The form builder can access the forms from here to create new forms, or edit existing forms. Once forms are ready to be deployed, the TM form version needs to be built and imported into Transact Manager.



Components: This tab displays a list of all editable components in the selected project. Note that these are not available to forms until they are published to a library.

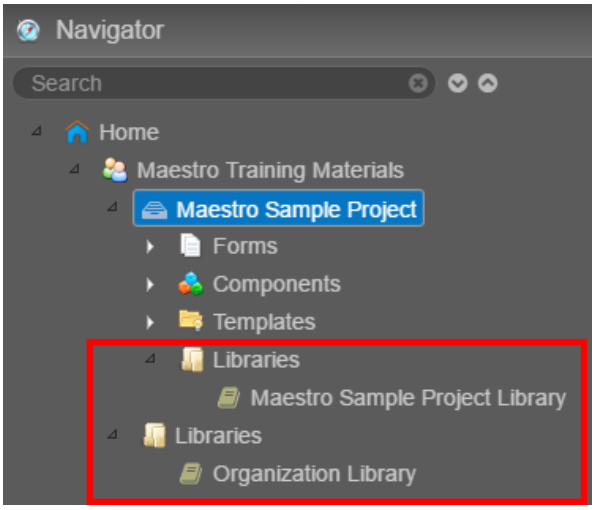


Templates: This tab displays a list of all editable templates in this project. Note that these are not available to forms until they are published to a library.

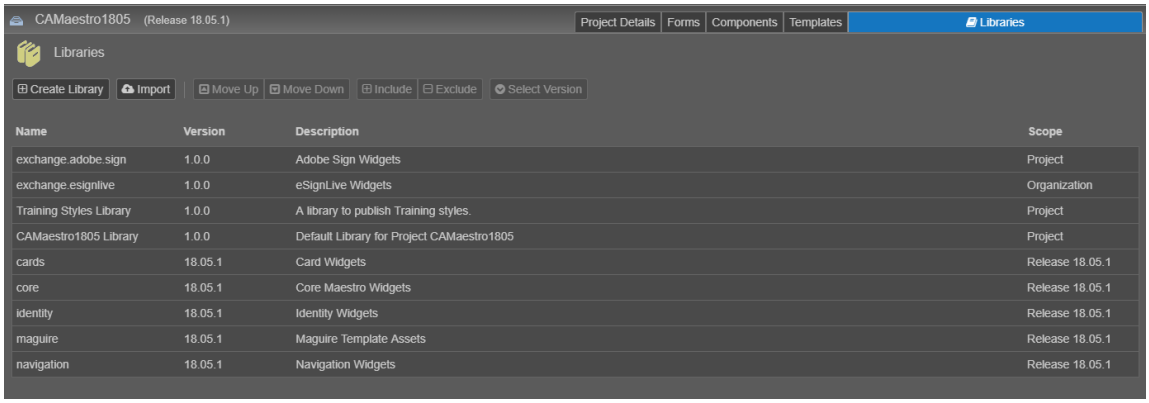


Published Assets and Resources (Libraries)

Published versions of the assets and resources are stored in a library. You can publish assets and resources to a project library or an organization library. Project libraries are only accessed within the project. Organization libraries can be accessed by all projects within the organization. For more information on setting up projects and libraries, please see the Best practice for resource management documentation.



Libraries: Displays all the libraries that the project can access. These can be at the organization level in the structure or local to the project.


CAMaestro1805 (Release 18.05.1)
Project Details
Forms
Components
Templates
Libraries
Libraries
Create Library
Import
Move Up
Move Down
Include
Exclude
Select Version
Name
Version
Description
Scope
exchange.adobe.sign
1.0.0
Adobe Sign Widgets
Project
exchange.esignlive
1.0.0
eSignLive Widgets
Organization
Training Styles Library
1.0.0
A library to publish Training styles.
Project
CAMaestro1805 Library
1.0.0
Default Library for Project CAMaestro1805
Project
cards
18.05.1
Card Widgets
Release 18.05.1
core
18.05.1
Core Maestro Widgets
Release 18.05.1
identity
18.05.1
Identity Widgets
Release 18.05.1
maguire
18.05.1
Maguire Template Assets
Release 18.05.1
navigation
18.05.1
Navigation Widgets
Release 18.05.1

Create a New Project

Unknown macro: 'redirect'

Create a New Project

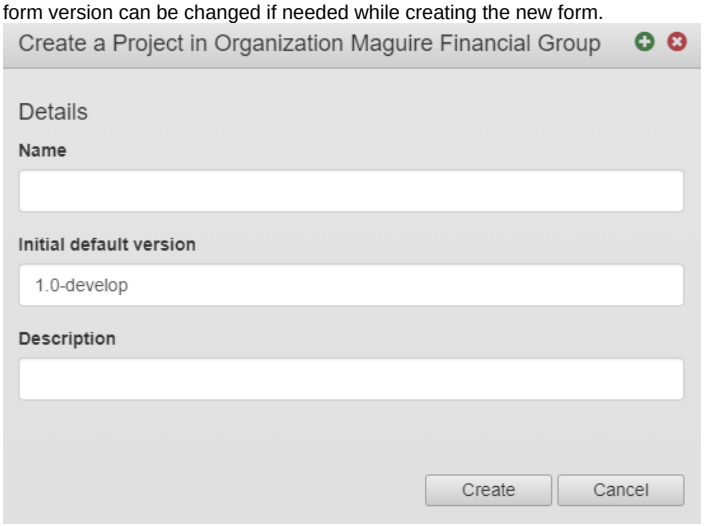
Projects contain the assets and resources needed to create Maestro forms. The Forms, Components and Templates folders (or tabs) are the development areas of the corresponding asset. The Library is where the published versions of the assets and the resources are stored. Each Organization can have as many Projects as needed. Projects allow you to keep assets and resources organized.

> It is recommended that you view Best practice for resource management for more information before creating a new project.

Most form related projects will have projects specifically available for certain resources. For example, you will likely see a project for shared business components, and another project specifically for templates and styles. This structure allows form builders and template designers to keep resources separate and organized.

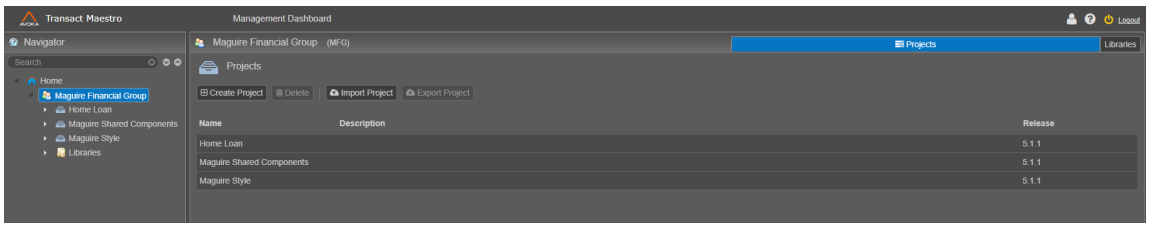
To create a project:

1. Select the Organizationthat you want to create a project in
2. Click *Create Project*
3. Enter a Name, Initial default version, and Description for the new project
   Initial default version - This is the starting version for all new forms created within the project. The form version can be changed if needed while creating the new form.



4. Click *Create*

The new project will display in the Navigator pane as well as the project list. You can then navigate within the project to create assets and resources.



Related Pages:

- Create a New Project
- Libraries
- Logging In
- Maestro Organizational Structure
- Maestro Projects
- Management Dashboard Overview

Upgrade Project

Unknown macro: 'redirect'

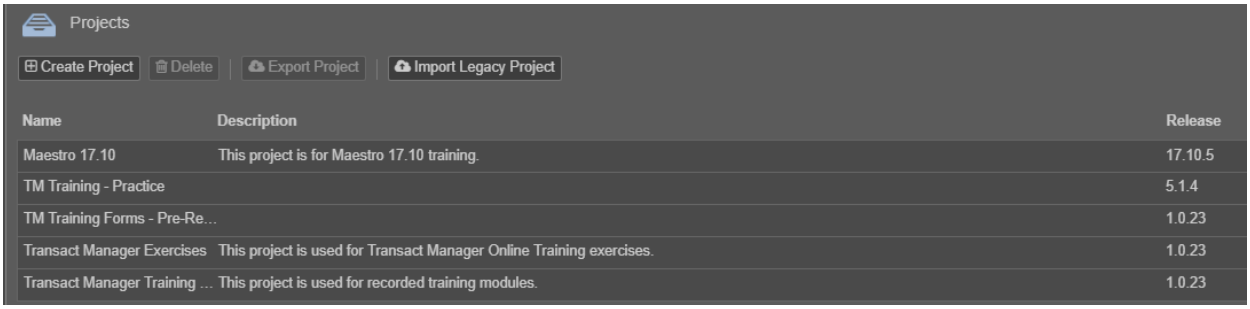
When a Maestro environment is upgraded, organizations can choose whether projects remain in the original release or are upgraded to use the new release features. By default, all projects remain with their original version until the project's release version is manually changed from the Maestro User Interface (UI). When a form, component or template is opened, the system will check the release version that the project is using. The project version determines the version of Maestro and the available libraries that are used when opening and editing the form.

Maestro supports a feature called compatibility mode.The compatibility mode feature allows organizations to continue working on the version they were using prior to the upgrade, ensuring there is no change in behavior to their forms.

It is important to note that once a project is upgraded, it cannot be returned to a prior version. Our recommendation for upgrading projects is to copy the project via an export process before upgrading the project. This allows you to restore from a previous version if required (via the zip file that is created during the export/import process).

Compatibility mode is only applied to the design view (editing forms, components, templates). The dashboard view (organizations, etc.) always uses the latest release version.

The screenshot below displays multiple projects with different release versions.



In release versions prior to version 5.1, Maestro releases were named using an 1.x.x decimal naming convention. For Maestro 5.1 projects, the release versions were displayed and named as 5.x.x. Starting with the 17.10 release, all releases will be named based on a year and month naming convention (e.g. 17.10 = 2017 October). Though the naming convention is based on the year and month of release, minor releases are still identified by decimal values (e.g. 17.10.1).

Upgrade Existing Projects

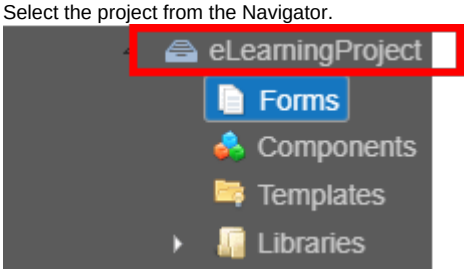
Once you update a project, you cannot rollback or downgrade the project to a previous version.

If you want to upgrade an existing project to the latest Maestro release, you can do so from the Management Dashboard by following the steps below. It is recommended that you create a copy of your project before upgrading the project. Creating a copy will ensure that you have the original project files to revert back to if there are issues when upgrading the project.

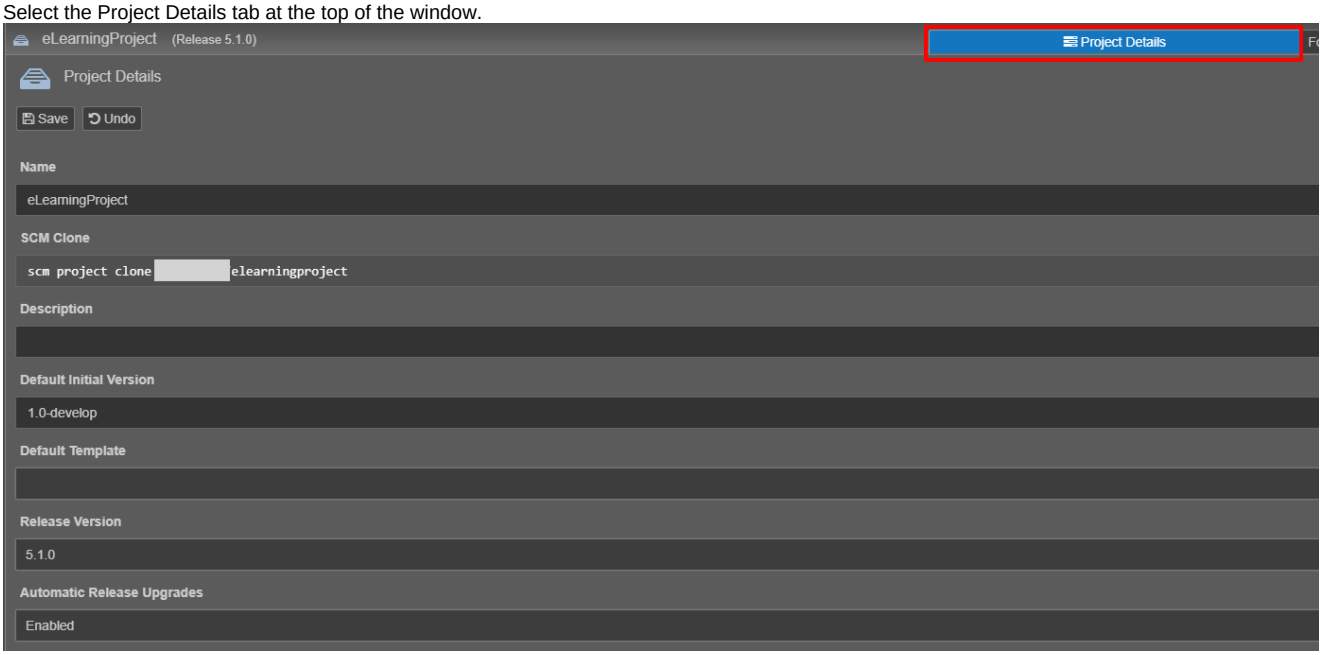
1. Navigate to the project that you want to upgrade and click the *Export Whole Project* button
   Clicking the *Export Whole Project* button will create a zip file of the project that will download to your computer (if you have the correct permission, the *Export Project* button may also be available for selection). This zip file is the copy of the project that you can keep and import back into Maestro if there are issues when upgrading the project.

   For ease of use, we recommend that you upgrade the existing project (the project that is **ALREADY** available in Maestro) and keep the exported (copied) version of the project as a backup.

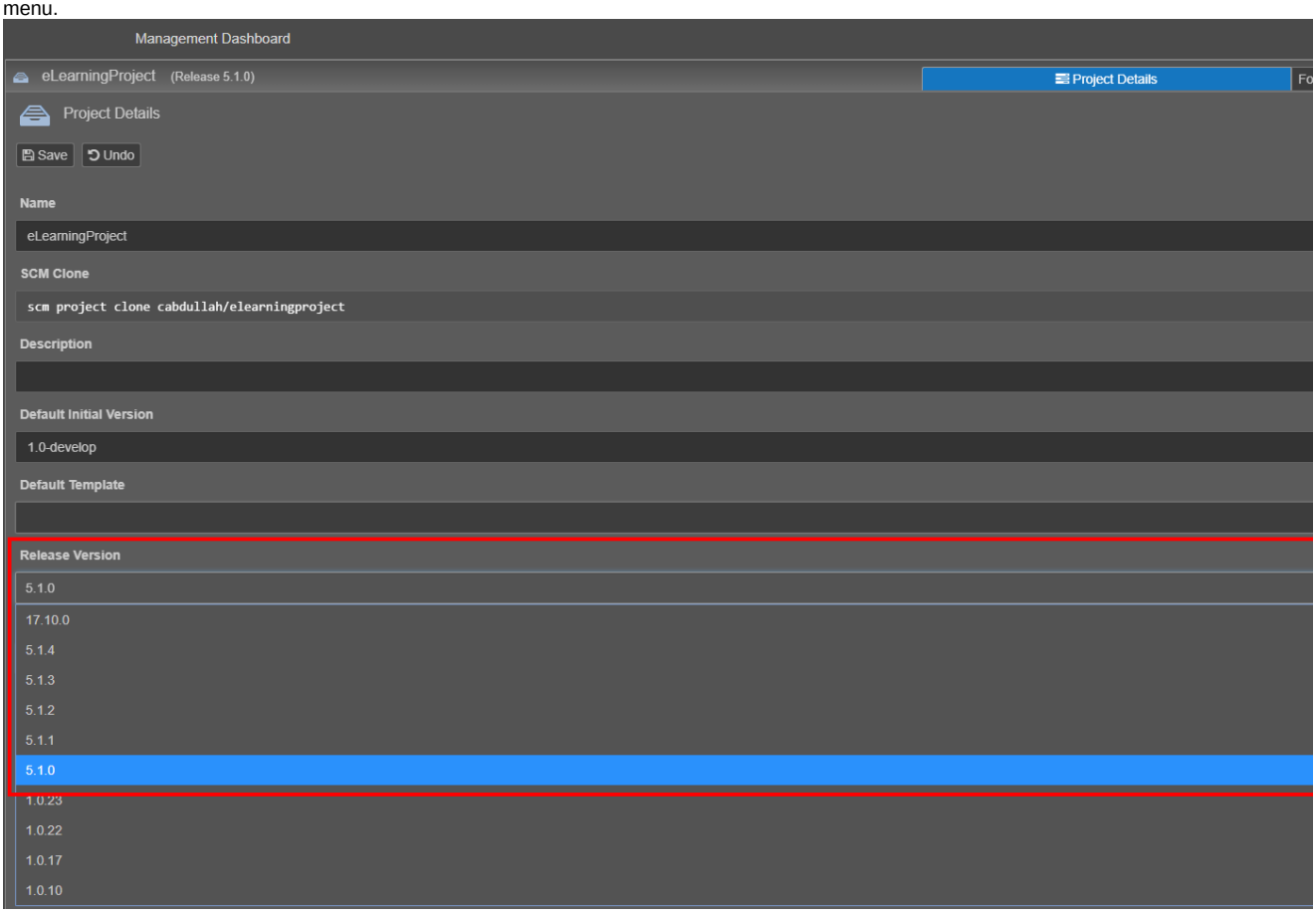
2. Select the project from the Navigator.



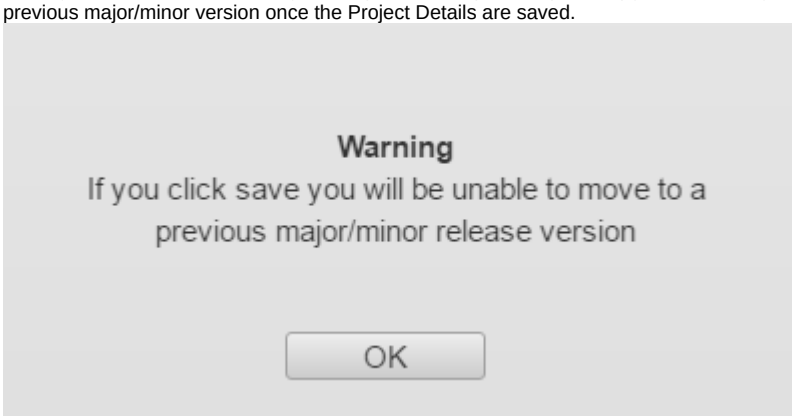
3. Select the Project Details tab at the top of the window.



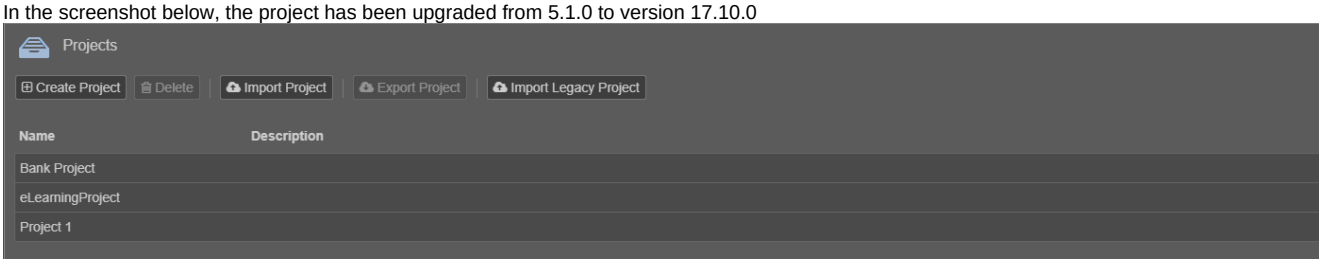
4. Use the Release Version dropdown to change the version.
   You can select any available version from the dropdown menu. The screenshot below display version 17.10 being selected from the dropdown menu.



5. Click OK when the warning message appears.
   When you select a new version, you will get a warning message letting you know that you will not be able to change the release version back to a previous major/minor version once the Project Details are saved.



6. Click Save to keep the updated release version.
   In the screenshot below, the project has been upgraded from 5.1.0 to version 17.10.0



Rebuilding and Testing

Once you have upgraded the project, we recommend that you rebuild and test each of the forms/artifacts in the project.

Automatic Release Upgrades

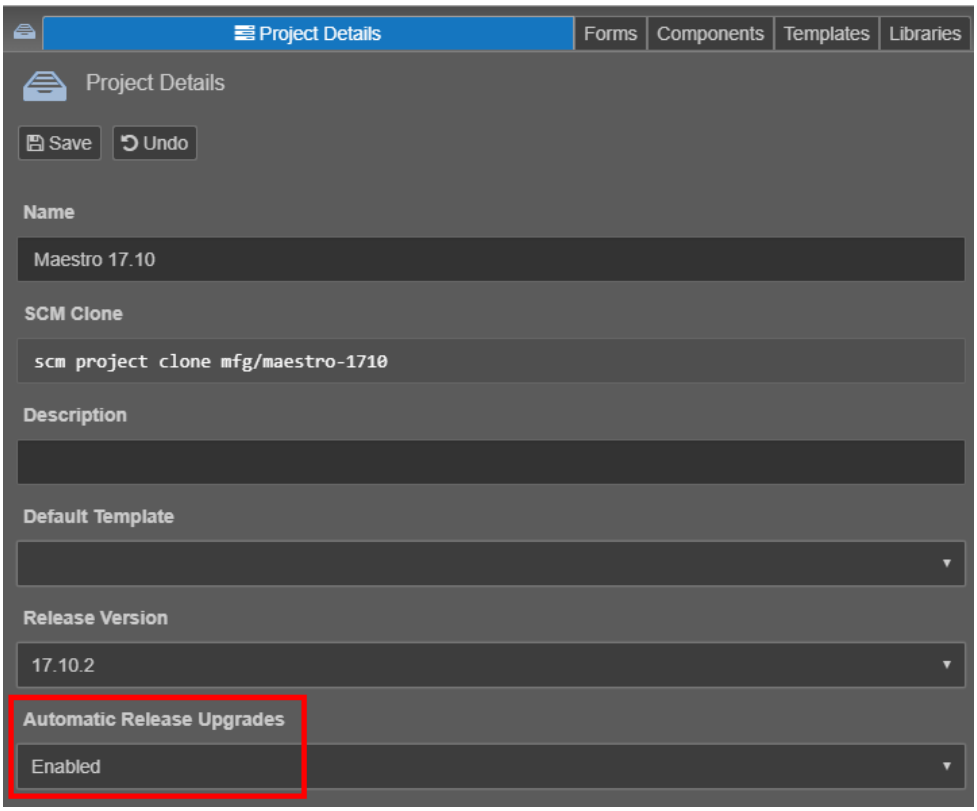
Unknown macro: 'redirect'

All Maestro projects are enabled, by default, with a feature called *Automatic Release Upgrades*. This feature enables projects to be automatically upgraded with minor releases of Maestro. For example, when this feature is enabled, a project will automatically be upgraded each time a minor upgrade of Maestro (eg 17.10.3) is released.

The purpose of this feature is to ensure that all Maestro projects are continuously up to date with the latest minor release of Maestro. The *Automatic Release Upgrades* feature also means that you do not have to plan to upgrade to minor releases, as when enabled, projects will automatically be upgraded to the latest minor release of Maestro. This means that you only have to decide on the timing of an upgrade to a major release of Maestro.

Please note that enabling *Automatic Release Upgrades* will only upgrade the project with minor release updates, it WILL NOT upgrade to MAJOR releases of Maestro. This means that when a major release of Maestro is released, the project will stay at its current release until it has been manually upgraded to the latest major release.

The screenshot below highlights the *Automatic Release Upgrades* feature option.



Creating a New Maestro Form

Unknown macro: 'redirect'

The content within this section covers information relating to creating new forms in Maestro.

The list below identifies the topics covered within this section.

- Form Template Overview
- Create a New Form
- Maestro Editor
- Add Components to a Form
- Form Preview
- Closing the Form

Form Template Overview

Unknown macro: 'redirect'

Related Pages:

- Overview of the Form Template
- Details for a Standard Template

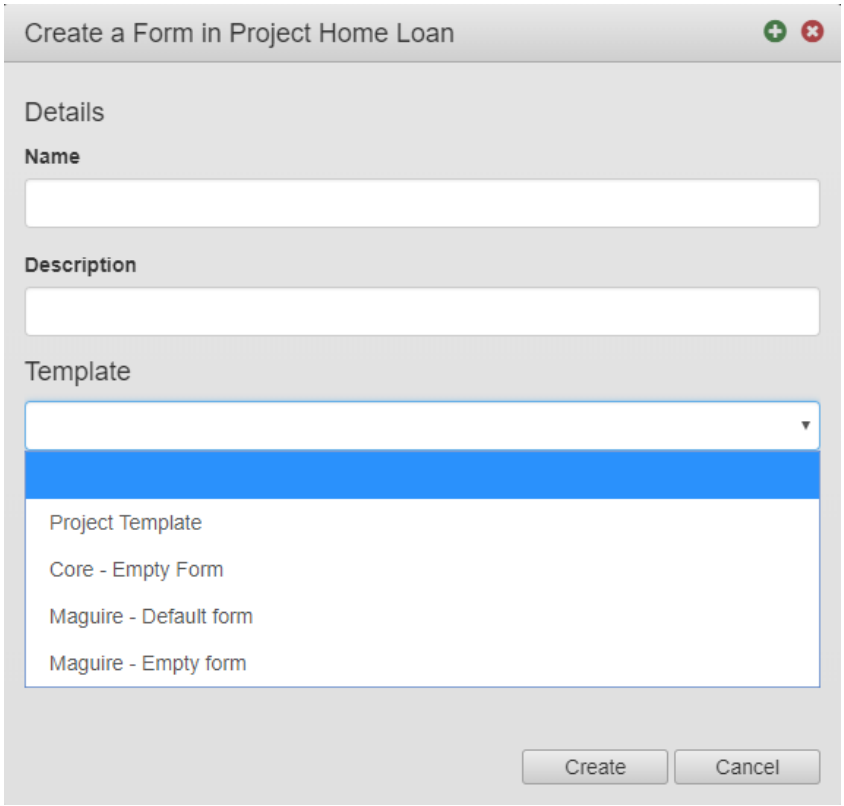
Related Pages:

- Create a New Form
- Customizing the Maestro Editor
- Form Template Overview

Overview of the Form Template

Throughout Maguire you will find default assets and resources named Maguire. Maguire assets and resources are starting points, or examples of certain items.These assets and resources are created and managed by Avoka. You should only use these resources as starting points when developing new resources.

When you create a new form you will see a few default templates available for selection. These include Core - Empty Form, Maguire - Default form, Maguire - Empty form. You will also see custom built templates for your business. You should not select any of the Maguire templates when creating new forms as these templates cannot be edited. This means that you would not be able to change the logo displayed on the form. The purpose of these default templates are for the creation of *new* templates. New project templates will likely be based on the Maguire - Default form as it is a good starting point for new templates. You can learn more about templates by viewing the Maestro Templates section.



Details for a Standard Template

The Maguire Template (Maguire - Default form) contains various elements that are common to most forms, which makes it a good starting point for new templates. The Maguire template should only be used to create new templates, not new forms. This template cannot be edited therefore you would not be able to make necessary adjustments to follow your own corporate branding, such as changing the logo.

Common Template Elements

Most areas of the template will be managed (created, edited, updated) in the template by the template designer. The form will usually have limited access to various areas, which allow the form to retain the corporate branding and consistency across multiple forms. In the template, the template designer will decide which areas the form has access to and which areas can be edited via the Form Options window.

Forms will always have access to the Content area, as this is the main area of the form and is where all components are added. The form may also have access to specific Dialogs and/or Modal Pages. In rare cases, the form may also have access to other elements of the form, but this is very unlikely. Each template is customized, so the options displayed and available may differ between templates. The information below is common across most templates and forms.

Form Header: The form header is at the top of the form. This header displays the logo and the reference code which is used when the user returns at a later time to complete a form. The Form Header is not usually edited in the form. The form may have access to multiple Brands which may change the logo displayed in the form.

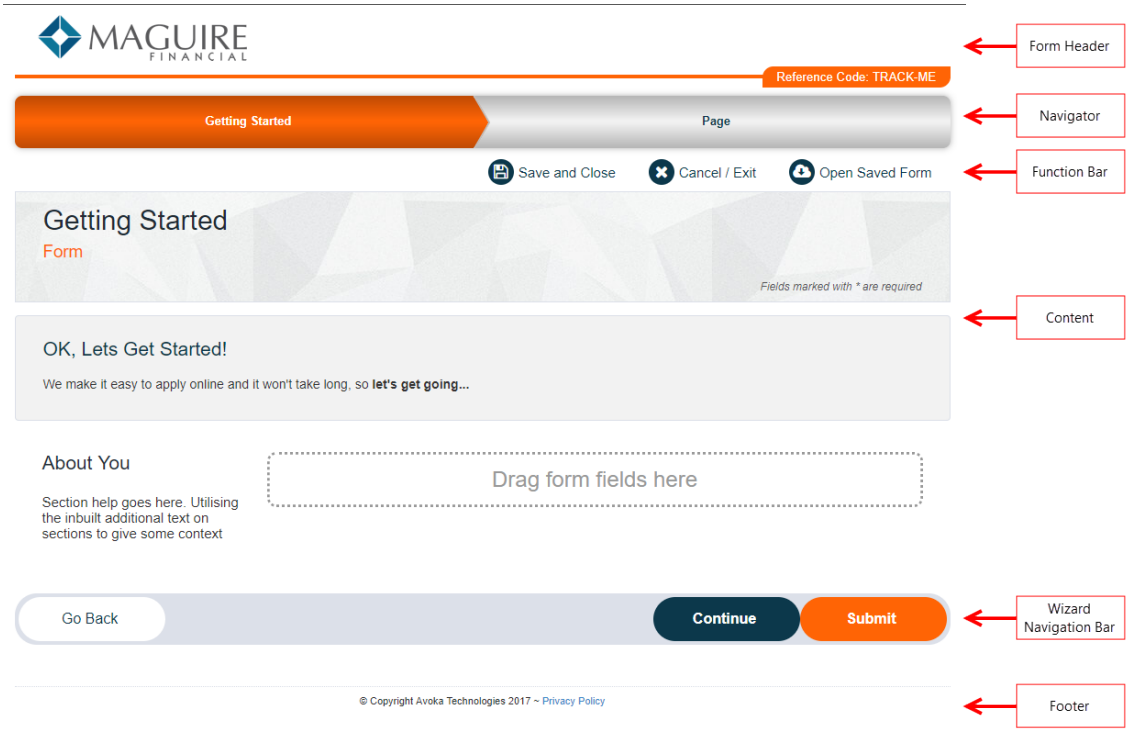
Navigator: The template also contains a navigator. The navigator displays all of the pages in the form and also allows for easy navigation.The form will builder will usually have the ability to edit page names and create page groups, which will impact the Navigator. The form builder may also have access to options such as adjusting the Navigator width and display of the Second Level Navigator.

Function Bar: The function bar contains buttons to save and close, cancel/exit and open a saved form. The Function Bar is managed in the template, however the form builder usually has the ability to enable /disable the buttons available in the Function Bar.

Content: The content area is where the Form Builder adds the necessary components to build the form. This will always be available to the form as this is the main development area.

Wizard Navigation Bar: This is at the bottom of the form and contains the navigation buttons to move from page to page, as well as submit the form when finished. The form builder will not usually have access to this area.

Form Footer: This is at the very bottom of the form and contains details such as copyright. The form builder will not usually have access to this area.



Create a New Form

Unknown macro: 'redirect'

Create a New Form

A form is a document designed to capture data and is fundamental to an end user experience called a transaction.

Form Builders create forms using Maestro, which are later imported into Transact Manager. End users request forms from, and submit them to, Transact Manager.

To create a new form:

1. Navigate using the Maestro Management Dashboard to the desired Project.
2. Click on the **Forms** tab, or Click on the **Forms** folder in the Navigator pane.
3. Click the **Create Form** button, which will display the Maestro Create Form dialog.
4. Give the form a Name, a Description and select the desired Form Template.

> Templates are a means of providing standardized appearance and behavior of a form. The template you select will make a big difference to what you see when the form opens. The template governs the "boilerplate" or "chrome" of your form, including the header, footer, and navigation bar.
>
> When selecting the template to use for the new form, you will notice the Maguire template options. These templates are managed by Avoka and should only be used as the starting point to new, custom templates. These should ***never*** be used when creating a form.

5. Click **Create.**

Maestro will open a new tab in your browser with the form opened in the Editor window. You can now begin to add content to your form.

If the form does not open after clicking "Create" you should check your pop-up blocker settings. When you first open a form (or other asset) from Maestro you will need to allow the pop-up.

It is a good idea to select the option to always allow pop-ups.

Related Pages:

- Create a New Form
- Customizing the Maestro Editor
- Error Blocks in Composer
- Form Template Overview

Maestro Editor

Unknown macro: 'redirect'

The content within this section covers information relating to working with forms in the Maestro editor.

Below you will find a list of topics within this section.

- Working with the Maestro Editor

Working with the Maestro Editor

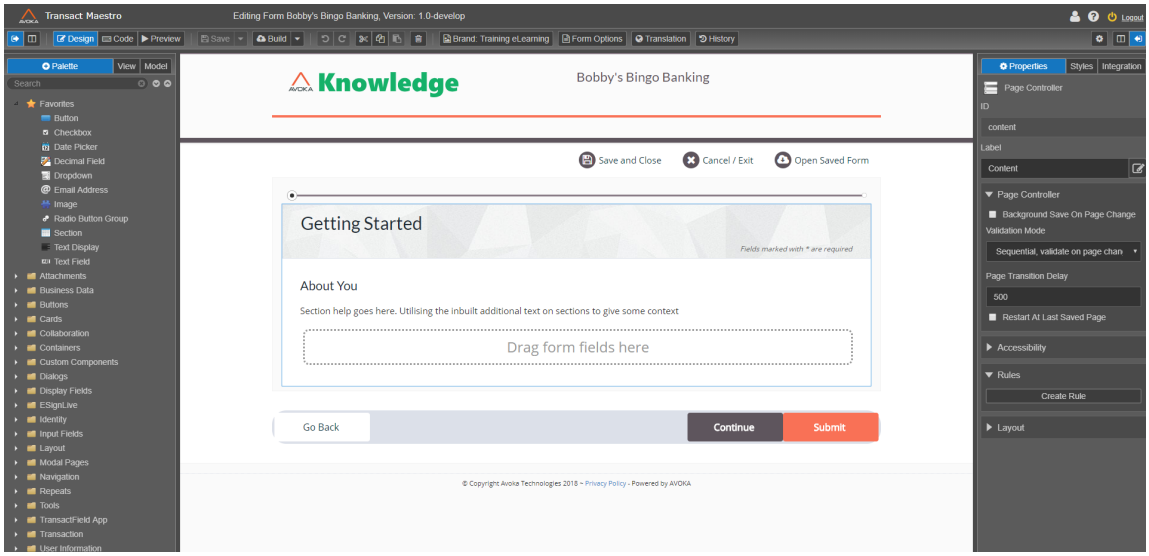
Unknown macro: 'redirect'

Related Pages:

- Create a New Form
- Customizing the Maestro Editor
- Form Template Overview

Overview of the Maestro Editor

Maestro assets are opened in the Design mode of the Editor window. The Maestro Editor is a window that consists of multiple panes.



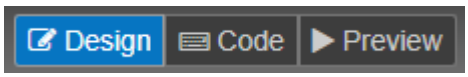
The structure of the Design view is as follows. Please see Customizing the Maestro Editor for information on how to customize the Maestro Editor layout.

- Toolbar. This provides functions that are related to the form as a whole.

- Left panel.
  - Palette Panel - This panel lists all available components (also known as fields or widgets) that you may add into your form.
  - View Panel - This panel provides a hierarchical structure of the form. The View panel is particularly useful with larger forms, where finding components can be difficult.
  - Model Panel. This panel displays data models and is used to manage and share Entity Models.
- Editor panel. This panel shows an editable preview of the form. If you click on a component, it's properties will be displayed in the right pane. You can drag and drop components within this panel.
- Right panel. All the panels on the right-side of the window display information about the currently selected component.
  - Properties panel. This panel displays the properties of the currently selected component. The properties will vary depending on what type of component is selected. This panel is also where business rules are defined.
  - Styles panel. This panel allows you to customize the look and feel of the currently selected component. Most, if not all, styling is done by the Template Designer in the Template, not the form.
  - Integration panel. This panel allows you to control the underlying data representation of the currently selected component.

If you have a wider screen (highly recommended) you can modify the layout of the editor by Customizing the Maestro Editor to display some of the tabs side-by-side. This can help to make you more productive and help you navigate the window.

The Maestro Editor has three different modes (or views), which are shown as icons.



- Design mode. This is the default view, and displays the design window shown above. This is where you work with the open form, template or component.
- Code mode. Code View allows a developer to view, create and modify all business rules in one place. See Code View for more information.
- Preview mode. Click the Preview button to show the Preview Mode. This is a working preview of your form so you can check the components and some functionality without having to build the TM form version. See Form Preview for more information.

Customizing the Maestro Editor

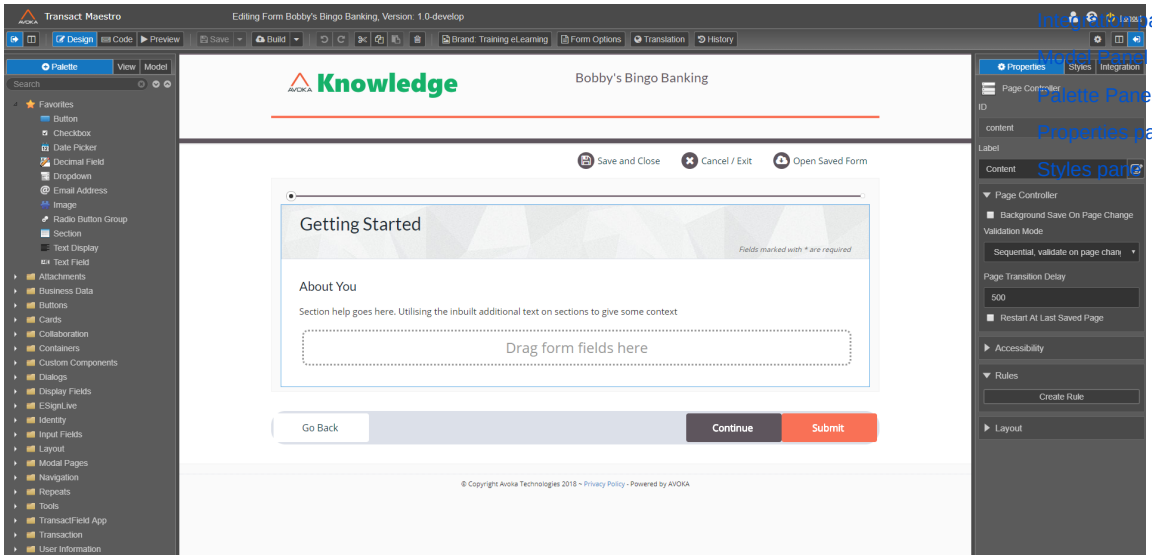
Unknown macro: 'redirect'

Related Pages:

- Closing the Form
- Create a New Form
- Customizing the Maestro Editor
- Form Preview
- Form Template Overview
- Integrated Panel
- Model Panel
- Palette Panel
- Properties pane
- Styles panel

Customizing the Editor Layout

By default, Maestro is configured so you can only see one of the tabs on each side of the form, as shown below. In this case, the Palette and the Properties tabs are shown.

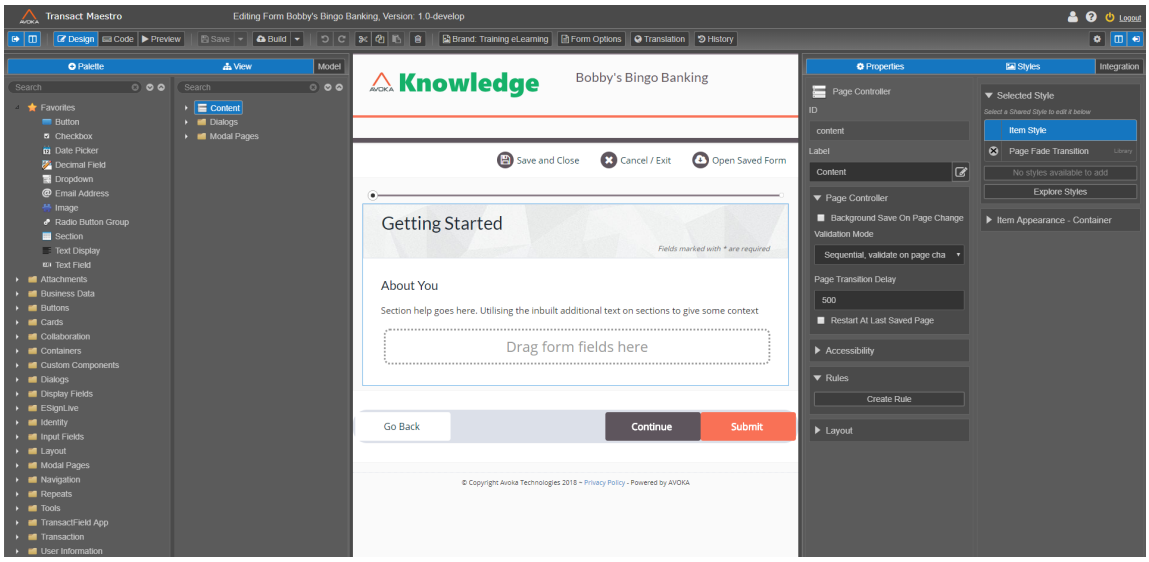


The two small toggle icons at the top of each panel controls which panels are displayed:

 These icons, when selected (as shown here) will display the left and right-hand panels respectively. If you deselect these icons, the entire panel will be hidden. This is useful on smaller screens.

 This icon, when de-selected (as shown here) will show only one tab in the respective panel. If you click this icon, the panel will display two tabs side by side. This can be really useful on larger screens. You can adjust the size of the tabs within the panels depending on the width of your screen.

The screenshot below shows four tabs displayed simultaneously.



Tip

If you have a larger screen, the four-tab view can be very powerful.

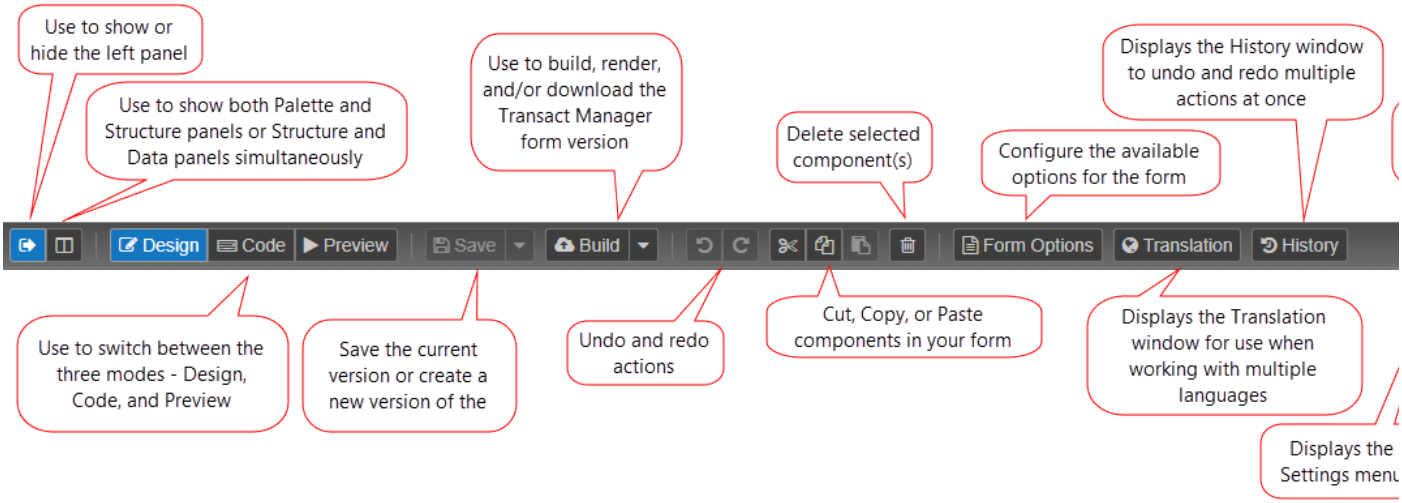
On the left side, this allows you drag and drop components from the Palette panel directly into the View panel. Many seasoned Maestro developers find this a quicker and easier way to add components.

On the right side, this allows you to view the Properties and Styles panels together, which can also make editing a little faster.

Toolbar

Unknown macro: 'redirect'

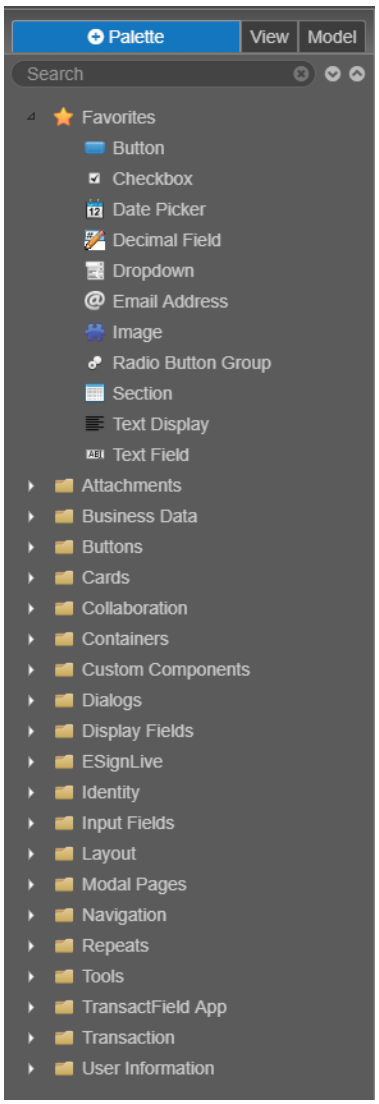
The toolbar displays at the top of your form in the editor window. The following images show the buttons available, along with a description.



Palette Panel

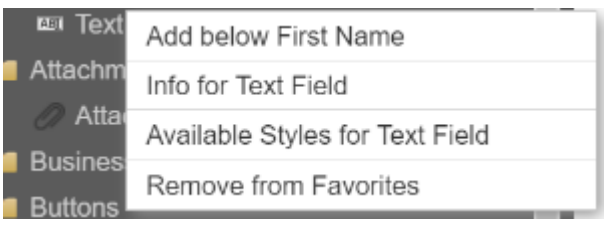
Unknown macro: 'redirect'

The Palette panel contains the available components used to build form and transaction experiences. The available components may be standard Avoka components or shared components.



The Palette Panel derives its name from its similarity to an artist's palette which contains all the paint in different colors that an artist will apply to their painting.

- Components are grouped in folders according to functional use.
- You can browse for a component by expanding the category folders, or you can search for a component whose name you know by typing part of its name in the search box.
- The list of components displayed may be different for different customers, or even different projects. This is because Maestro is highly configurable and extensible.
- Right-click on a component to display a menu of actions relating to that component. In the screenshot below, First Name has been right-clicked in the form.



Add or Remove Components from Favorites List

The Favorites folder contains a list of the most commonly used components. This listing may be customized by individual form builders.

To add a favorite component:

Related Pages:

- Closing the Form
- Customizing the Maestro Editor
- Form Preview
- History Button
- Integration panel
- Model Panel
- Palette Panel
- Properties pane
- Styles pane
- View Panel

1. Right-click the component you want to add
2. Select Add to Favorites

To remove a component from the Favorites list:

1. Right-click the component you want to remove
2. Select Remove from Favorites

Power User Tip

Most new users will feel comfortable dragging fields into the Editor pane. However, as you gain experience, you may find you can be much more productive by using one of these tips:

- Drag component from the Palette to the View panel. This is quicker and can be more precise. (You must enable side-by-side tabs as shown here: Customizing the Maestro Editor)
- Double click on a component to add it immediately below the currently selected item.

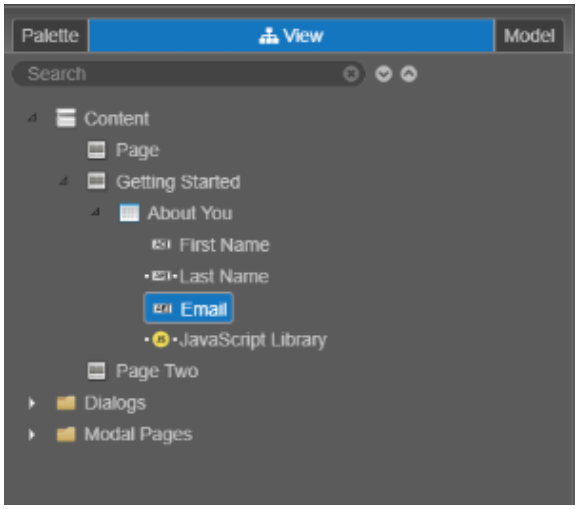
.

View Panel

Unknown macro: 'redirect'

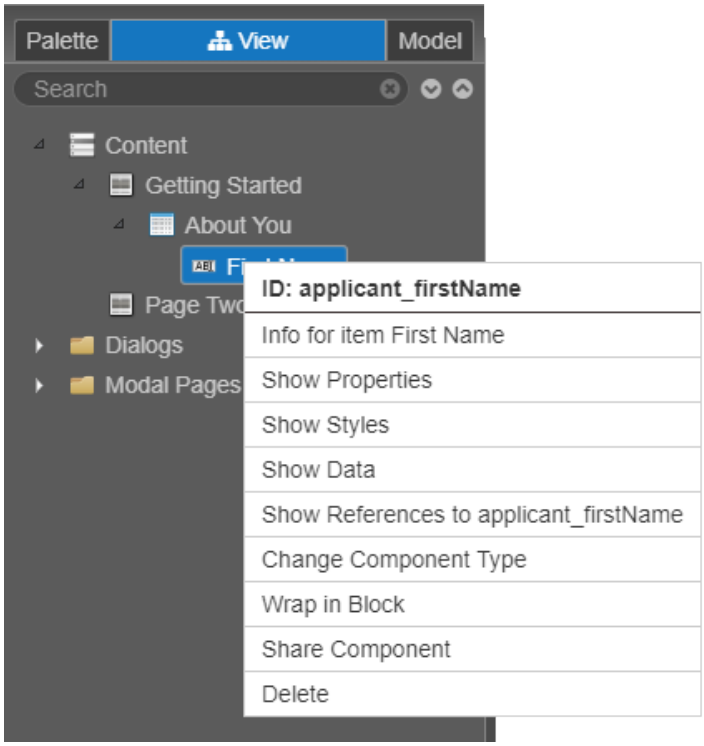
The View Panel provides an outline view of the form presented as a hierarchy. This helps to conceptualize the form's structure of containers and components. The hierarchy you see on opening a new form is determined by the template the form is based on. Dialogs and Modal Pages will always be displayed, even if the form builder does not have permission to make any changes to existing dialogs and modal pages.

The View Panel provides a quick way to navigate through a longer form. This panel also has functional meaning, such as the division of the form into pages and sections (the views the applicant will see). You can also groups components together for layout requirements, or add business rules at the group level.



Using the View Panel

- You can move components from one location to another by dragging and dropping them within the hierarchy.
- You can select multiple components by using keyboard keys SHIFT or CTRL.
- Right-clicking on a component, gives you a list of actions specific to the component.



Some component types must be contained within other component types. For example, a Page cannot exist underneath a Dropdown. The Editor will prevent you from moving or creating components under the incorrect parent.

Related Pages:

- Closing the Form
- Customizing the Maestro Editor
- Form Preview
- History Button
- Integration panel
- Model Panel
- Palette Panel
- Properties pane
- Styles pane
- View Panel

Model Panel

Unknown macro: 'redirect'

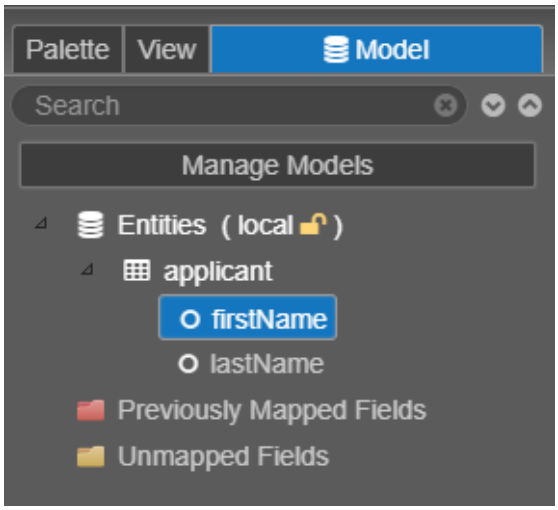
Model Panel

The Model Panel shows the hierarchical data structure of the form and offers a view from a data perspective. The component IDs, along with their related entities are displayed within this panel. The panel shows the data elements with bound components and allows for an easy way to organize the IDs and show a representation of the XML output.

The Entities item displays the entities that have been created in the form.

The Not Assigned folder displays components that are not part of an entity.

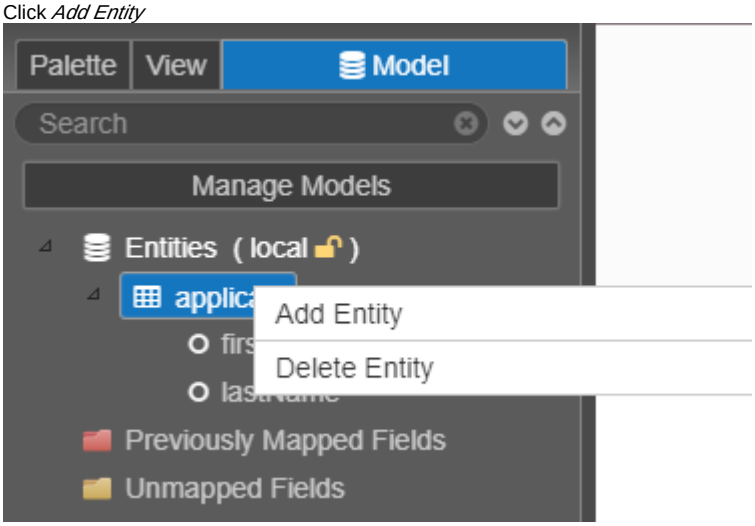
For more information on component IDs and Entities see Understanding the Component ID.



For more information on entities and entity models, please refer to the Shared Domain Model section of the documentation.

Create a new entity within a domain model

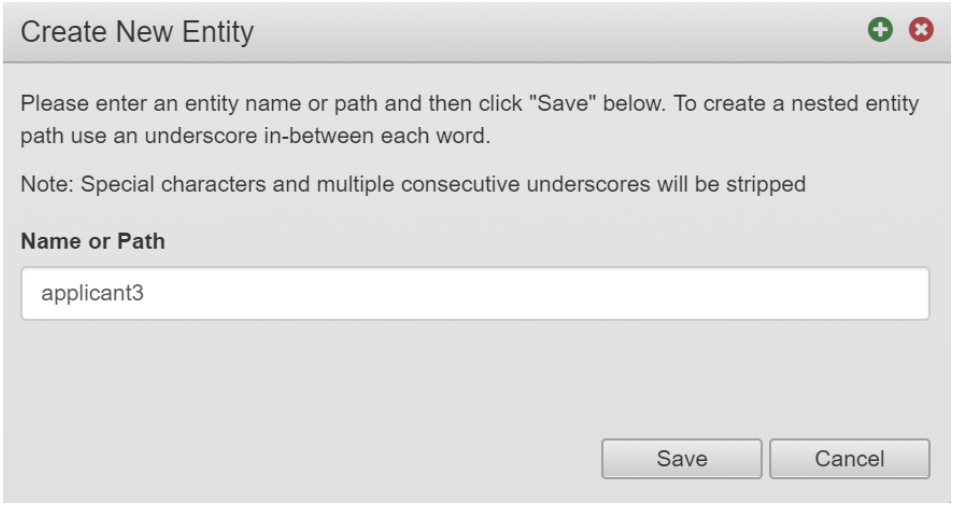
1. *Right-click* where you want the new entity to be placed.
   To create a nested entity right-click on an existing entity and it will place the new entity at that level.
2. Click *Add Entity*



3. Enter a name for the new entity
   You can also create a nested entity by using underscores between each entity name.

Related pages:

- Closing the Form
- Customizing the Maestro Editor
- Form Preview
- History Button
- Integration panel
- Model Panel
- Palette Panel
- Properties pane
- Styles pane
- View Panel



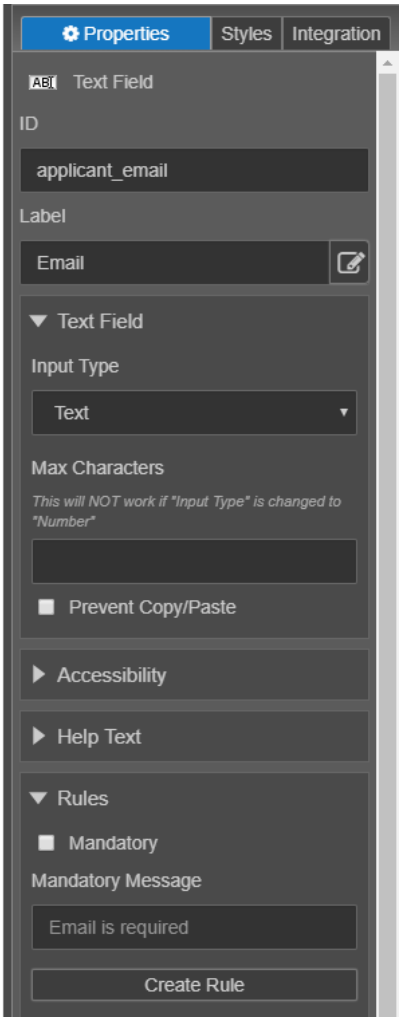
4. Click *Save*
   The new entity will be added to the list on the Entities panel.

Properties pane

Unknown macro: 'redirect'

Properties Panel

The Properties panel includes information about the currently selected component. It is used to change the default properties of a component thereby affecting either the way it functions or its layout in the form.



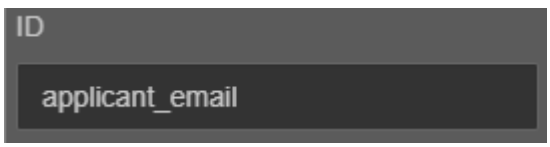
Once you add components to the form you are able to use the Properties panel to adjust the component as needed. Many of the property options will be similar across the components, but some components will have their own custom settings.

The properties are grouped by related settings. You can use the arrow to expand the group and view the properties available.

Common Properties

The properties below cover some of the more common properties available across different components.

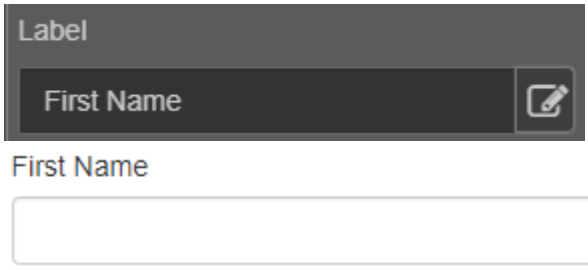
ID: The ID is the underlying Maestro name of the component currently selected. For more information see Understanding the Component ID.



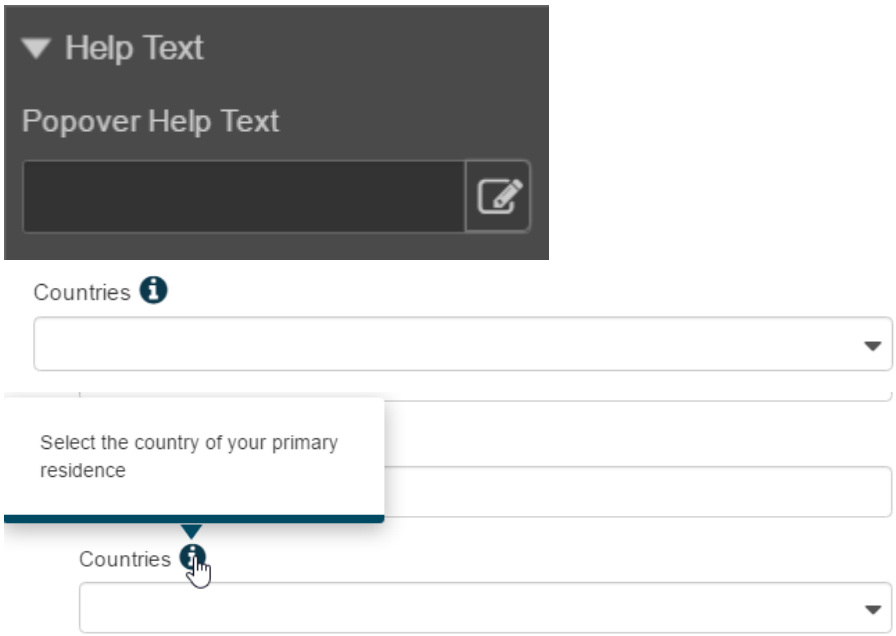
Label: The first property of every component is the label of the component. This will be displayed to the user on the form (however the form builder may choose not to give the component a label). The subsequent properties will vary depending on what type of component is selected. For more information see Change Component Label.

Related Pages:

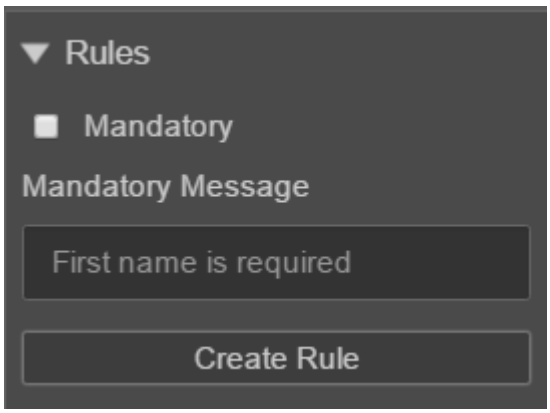
- Closing the Form
- Customizing the Maestro Editor
- Form Preview
- History Button
- Integration panel
- Model Panel
- Palette Panel
- Properties pane
- Styles pane
- View Panel



Help Text > Popover Help Text: When you enter text in the Popover Help Text an information icon will display next to the component. The form user will be able to select the icon or roll their mouse over the icon in order to display the help text entered.



Rules: The Rules section is where you can mark components as mandatory, as well as create additional rules such as visibility, validation, calculation, or editability.



Layout: The Layout section is where you can change the position and width of the selected component.

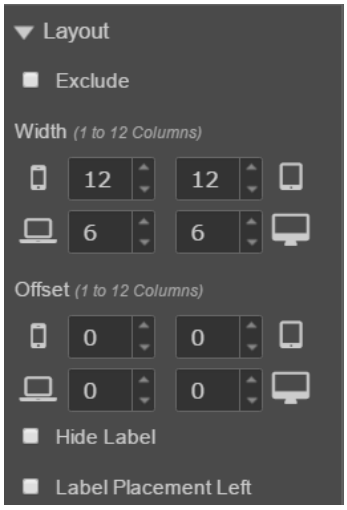
Exclude - This allows you to remove the component from the form, without deleting it.

Width - The four width options allow you to set the width of the component based on the breakpoints, or screen size, of the device viewing the form.

Offset - This allows you to set the horizontal position of the component based on the breakpoints of the device viewing the form.

Hide Label - If this option is selected the label of the component will not be displayed on the form.

Label Placement Left - The default position of label components is above, selecting this option will move the label to the left of the selected component.

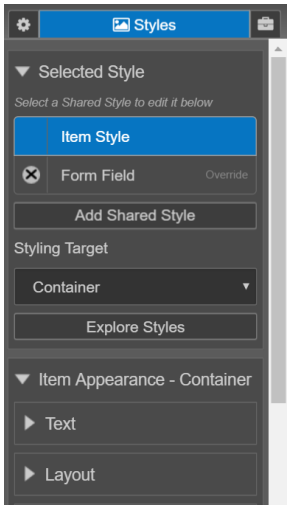

Layout
Exclude
Width (1 to 12 Columns)
12
12
6
6
Offset (1 to 12 Columns)
0
0
0
0
Hide Label
Label Placement Left

Styles pane

Unknown macro: 'redirect'

The Styles panel allows you to modify the visual look of a component.

> **Tip**
>
> Form builders should not be using the Styles panel very often, if ever. In most cases the template designer will manage all styling in the template, which will be based off of the corporate style guide. You should always speak to the template designer before making any styling changes.



Related Pages:

- Closing the Form
- Customizing the Maestro Editor
- Form Preview
- History Button
- Integration panel
- Model Panel
- Palette Panel
- Properties pane
- Styles pane
- View Panel

Integration panel

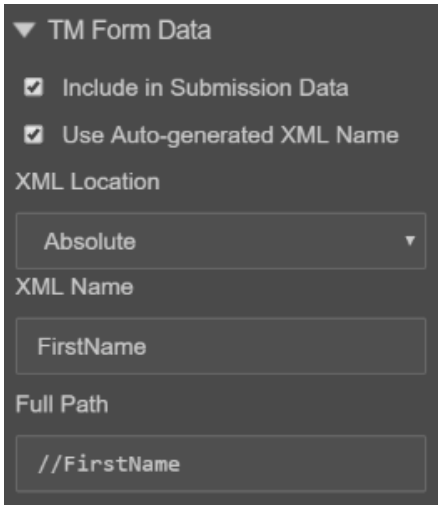
Unknown macro: 'redirect'

Integration Panel

TM Form Data

The TM Form Data section will be different depending on the component data structure.

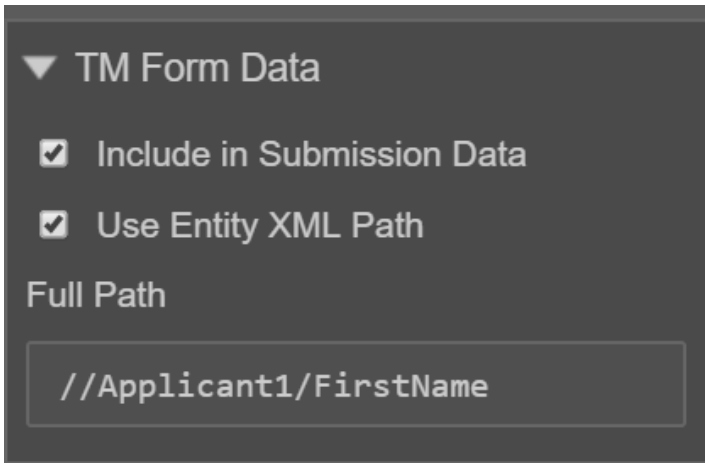
Component without an Entity



Include in Submission Data - Determines if the component is included in the submission data. If *Include in Submission Data* is deselected, you will not see any other options in this section.

Use Auto-generated XML Name - Selecting this option populates default values for the *XML Location, XML Name,* and *Full Path* properties. These details are based on the Component ID property. For more information on these properties, see the Binding section of the documentation.

Component with an Entity



Include in Submission Data - Determines if the component is included in the submission data. If *Include in Submission Data* is deselected you will not see any other options in this section.

Use Entity XML Path - Selecting this option populates default values for the *Full Path* properties. These details are based on the Component ID property, including the data structure based on the entity structure. For more information on these properties see the Binding section of the documentation.

Transact Integration

This section may be used by the form builder to specify how default values are defined for a component and how to set up submission data extract mappings to be used in Transact Manager.

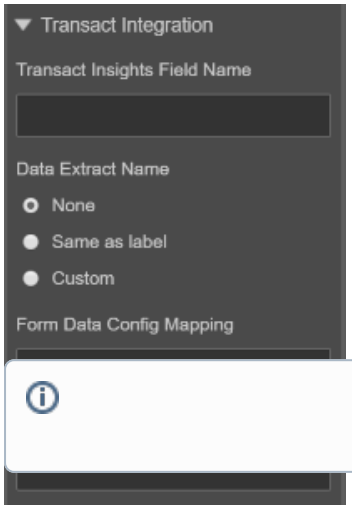
Transact Insights Field Name - This is used for integration with Insights. For more information, see the Insights documentation.

Data Extract Name - The name of the submission data extract field in Transact Manager. See Submission Data Extract for more information.

Form Data Config Mapping - These are specialized system values used in Transact Manager processes. Options include *Contact Email, Contact Phone,* and *Save Challenge*

Related Pages:

- Closing the Form
- Customizing the Maestro Editor
- Form Preview
- History Button
- Integration panel
- Model Panel
- Palette Panel
- Properties pane
- Styles pane
- View Panel



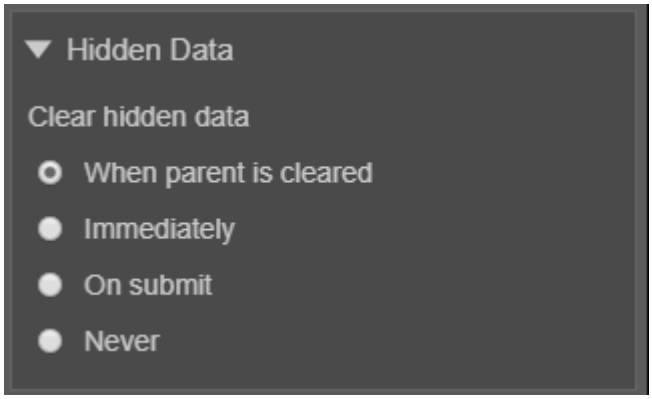
Initial Data - The default value displayed when the form is loaded.

Hidden Data

The Hidden Data property section may be used by a form builder to determine what happens to form data in an item when it is hidden by a visibility rule.

The clear hidden data property only checks for visibility on the parent level of the component. It does not check for visibility on the children level of the component if this level is made visible.

In order for the hidden data property to function correctly, the current block which contains the component in the form must be hidden.



Clear hidden data - These options determine when (if ever) the component data is cleared.

When parent is cleared - use the parent's clearing rule to clear this item.

Immediately - clear the item as soon as it's hidden.

On submit - preserve the data in the item until the form is submitted.

Never - preserve the data regardless of the item's visibility.

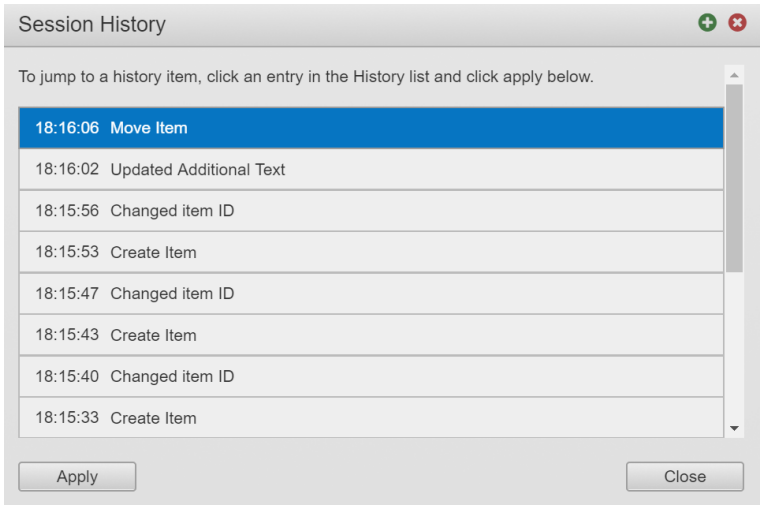
History Button

Unknown macro: 'redirect'

History Button

The History button will show all of the actions completed since the asset was opened. Assets include forms, templates, and components.

You can use the History button to undo the most recent action, or group of actions. To revert or undo an action, or multiple actions, you can select the item from the list, then click the Apply button. This will undo all actions back to the selected point.



The History button will show all actions completed since the asset was opened. If you want to view a history of each time the asset was saved see Save History for more information.

Related Pages:

- Closing the Form
- Customizing the Maestro Editor
- Form Preview
- History Button
- Integration panel
- Palette Panel
- Properties pane
- Styles pane
- View Panel

Add Components to a Form

Unknown macro: 'redirect'

Page Contents:

Related Pages:

- Closing the Form
- Customizing the Maestro Editor
- Form Preview
- Integration panel
- Model Panel
- Palette Panel
- Properties pane
- Styles pane
- View Panel

Add Components to a Form

Fundamental to form design is the process of adding components to capture the data required to satisfy a Transaction Experience. The term component is the general name for anything that can be dragged from the Maestro Palette into a form. Components can be anything from an image, to a single data entry field, to a large complex re-usable block. Components come in a variety of control types with different properties and behaviors. The available components are displayed in the Maestro Palette Panel. For more information on this panel, refer to the Palette Panel section of the documentation.

Choosing a component is done in the Palette, and placing it onto the form is done by dragging and dropping the component onto the form or dragging and dropping the component into the hierarchy of the View Panel.

Components are organized within category folders. You can browse each category to find the component you want to use on your form, or you can use the search box at the top of the panel.

At the top of the component list, you will find the favorites category. This category displays the most commonly used components. You can add or remove components from the favorite list based on your own personal preferences. The changes you make to the Favorites list will be stored in your Maestro account, not within the form. The components you add or remove to favorites will be displayed each time you open a form in Maestro.

There are two options available to add a component to a form.

Option One

1. Choose a component from the Palette and drag it onto the form
2. Drag the component onto or near where you want it to appear on the form, observing the insertion bar shape and orientation
3. Drop the component when the insertion bar indicates the desired location

Option Two

1. Select the component immediately above where you want the component to be inserted. You can do this either in the Editor, or in the View Panel.
2. Double click on a component in the Palette. This will insert the component directly below the existing component. You can also right-click the component from the Palette and select Add Below.

This option can be a more effective way of adding components, as it is less effort than dragging and dropping.

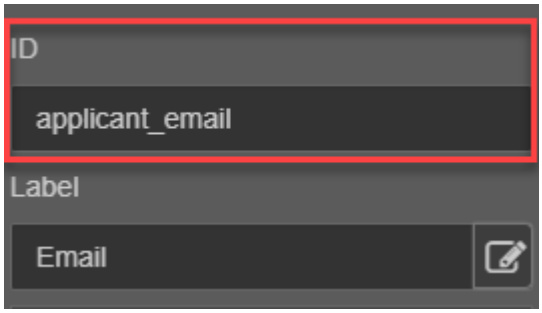
If you have multiple components to add, simply keep double clicking - the component will be added sequentially.

Once you have added a component, you will generally want to update the *ID Name* and confirm the *Label* from the Properties panel.

The ID property of the component will always be highlighted automatically immediately after adding a component, so you can just start typing. The Label is based on the component ID, but can be changed as needed.

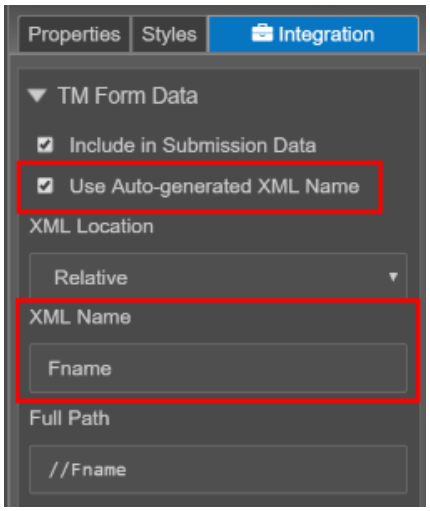
Component ID

When you add a new component to a form, the ID field of the Properties panel will be highlighted automatically. A component ID is a unique identifier for a component. More information on component IDs is available from the Understanding the Component ID section of the documentation.



The XML Name and Label will be automatically generated from the component ID.

Once the initial ID is provided, the Label will not be updated should the ID change. The XML Name and Path will be updated based on the ID, unless the *Use Auto-generated XML Name* checkbox is deselected. If the checkbox is deselected, the XML Name will not be updated should the ID change.



For more information on the component ID, please refer to the Understanding the Component ID section of the documentation.

Determining where the component will be inserted

Insertion using the Editor

The insertion bar indicates a wide variety of possible insert locations by using different positions, sizes and shapes to represent the insertion point. The insertion point is represented by a selection area, which indicates the parent layout of the component, and an insertion bar which indicates its position to its siblings.

The insertion bar will appear either as a vertical or horizontal bar. A horizontal insertion bar will insert the component vertically between two other components, and a vertical bar will insert a component horizontally between two components.

If the insertion point is at the beginning or end of the list of siblings, then the component will be inserted regardless of the insertion points horizontal or vertical orientation. If the the insertion point is vertical, and there is space for the new component to occupy within the layout, the new component will appear on the same line as an previously placed components. If there isn't sufficient space to accommodate the component, it will be placed on the following line.

Insertion using the hierarchy

Similar to the Editor, an insertion point is displayed when you drag a component from the Palette panel to the View Panel.

In the View Panel, the insertion point appears as a triangle *above, below* or *on* the desired component if it's a container. You can change the insertion point while dragging by moving up and down the hierarchy. If the drag target is a component, the insertion point will appear *above* indicating the dragged component will be inserted *before* the drag target, and the same is true for *below* and *after*.

If the drag target is a container of other components, you can insert the component into the container by placing the insertion point *on* the drag target. Releasing the dragged component will add the component to end of the children. If you drag a component onto a container that is collapsed, the container will expand, displaying any children it contains, allowing you to navigate deeper into the structure without releasing the dragged component.

> You can change any mistakes or misplaced components by simply dragging the component to the correct location. You can also use the Undo button, moving back the History or simply deleting the component and trying again.

Shared Components

Another option for components is to use a shared component. A shared component is a custom built component within the business. A shared component can be added to your form just like any other component. The difference with a shared component is that it can be updated outside of the form. When it is updated all forms using that shared component will be updated. This is a great way to re-use common components, but also keep them consistent across multiple forms.

Shared components may be developed within the Components folder in the Maestro Dashboard or within a form.

Shared components can be published to any category within the Palette panel. The default location, which can be changed, is the Custom Components folder.

For more information on shared components view Create New Shared Components.

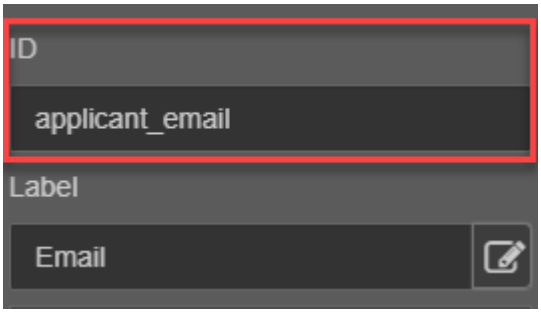
Understanding the Component ID

Unknown macro: 'redirect'

Page Contents:

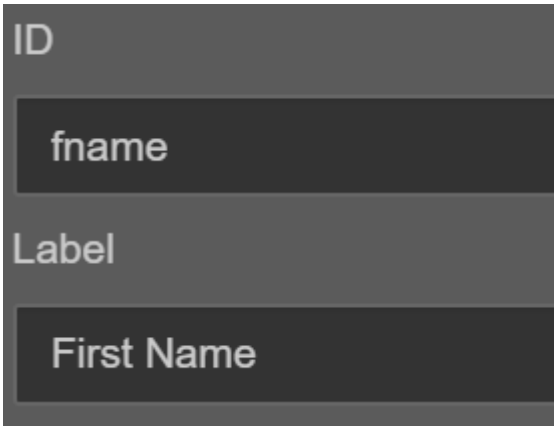
Understanding the Component ID

When you add a new component to a form, the ID field of the Properties panel will be automatically highlighted. You can begin typing the ID into the highlighted field. IDs can include an entity or prefix. When you start typing the ID, if there is an existing entity, it will display as an autocomplete option. Please see the Entity section for more information.



Changing the ID will update the Label and XML Name. Once the ID has been entered, the Label will not be updated should any changes be made to the ID.

For example, if you change the ID of a new Text Field to applicant_firstName, the Label will be updated to display the First Name. You can change the label directly in the Label field if needed.

The screenshot below shows a new text field added to the form. The ID was changed to *firstName* and the Label was automatically updated to *First Name.*	The screenshot below shows the ID being updated to *fname*, h as *First Name*.
	

The ID property is also linked to the XML Name and/or Full Path. If the ID changes, the XML Name and/or Full Path will also be updated, unless the *Use Auto-generated XML Name* or *Use Entity XML Path* on the Integration tab is deselected. If the option is deselected, the XML Name and/or Full Path will not be updated when the ID changes and will need to manually be changed as required.

Duplicate Component ID

If you enter an ID that has already been used, you will receive a duplicate ID error message.

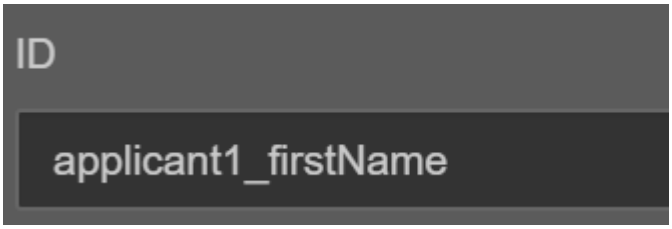
Selecting **Review** will let you re-enter an ID property with something that doesn't cause an error.

Selecting **Accept** will add a number to the end of the ID (e.g. firstName_1). When selecting Accept, the XML Name will display as the default name for the selected component type. You will need to manually update the XML Name as required by the Integration panel.

Entity

An entity is a prefix to a component ID denoted by an underscore. This prefix is used to include the data item within a data defined model.

For example, the entity "applicant1" prefixed to firstName would mean that the firstName component belongs to "applicant1"



The range of entities will be stored in the form JSON manifest as an object with keys for each entity. The value will be another object which is either empty (default) or has its own keys for child entities.

Example

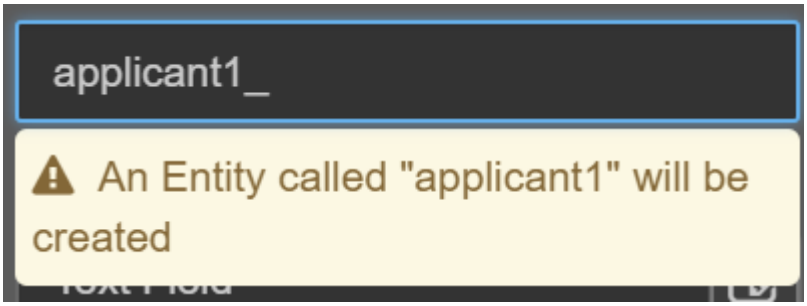
```
{
  applicant: {
    homeAddress: {}
    deliveryAddress: {}
  },
  asset: {}
}
```

Please note any existing IDs, including those with an underscore, will not cause entities to be created. Entities are only created when a user modifies an existing ID.

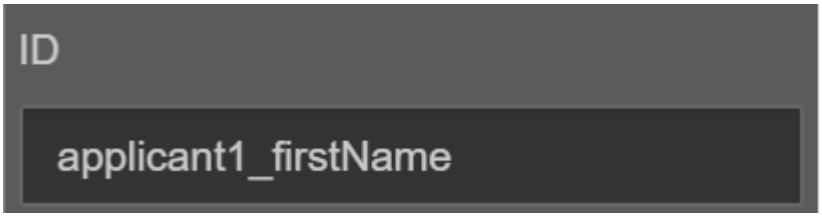
Create Entity

An entity is created directly from the ID field. When you want to create a new entity, enter the entity name followed by an underscore (i.e. applicant1_).

When you enter the underscore (_), a message will display letting you know that a new entity will be created.



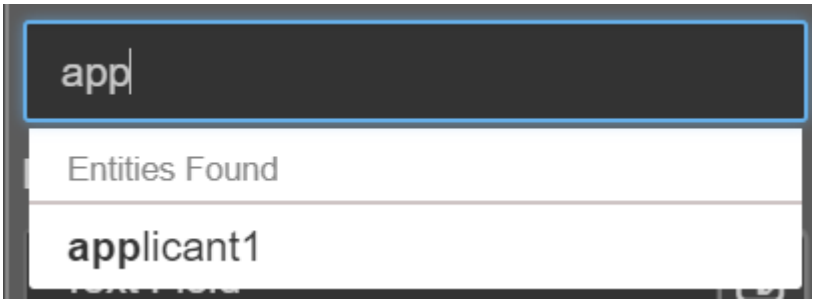
After the underscore, you can enter the ID name.

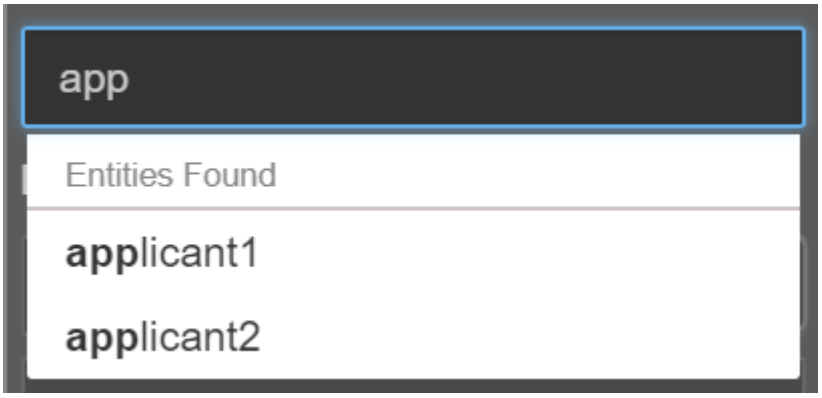


Using Existing Entities

An existing entity can be used in the ID field. When you begin typing in the ID field, if an existing entity matches the entered characters it will display in a dropdown menu.

If more than one existing entity matches the entered characters you will be able to select from the dropdown menu.





You can select the entity from the menu, which will populate the ID field. After the populated underscore, you can enter the ID name.

Create Label from ID

When the ID is initially entered, the entered data will be formatted correctly for the ID and will be copied to the Label field. It will only be copied to the Label field based on the initial entry. If you enter an ID and then change it later, the Label will not be updated.

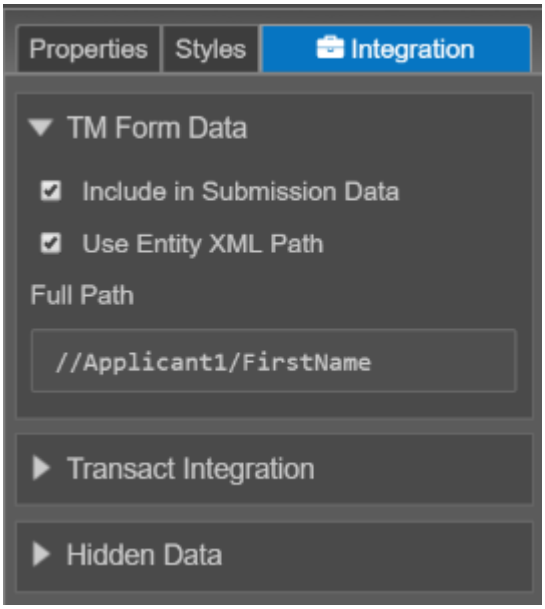
If a label value is detected in the ID field, it will be reformated and copied into the Label field. For example, if you type "First Name" into the ID field, it will convert the ID to "firstName" and copy the Label as "First Name".

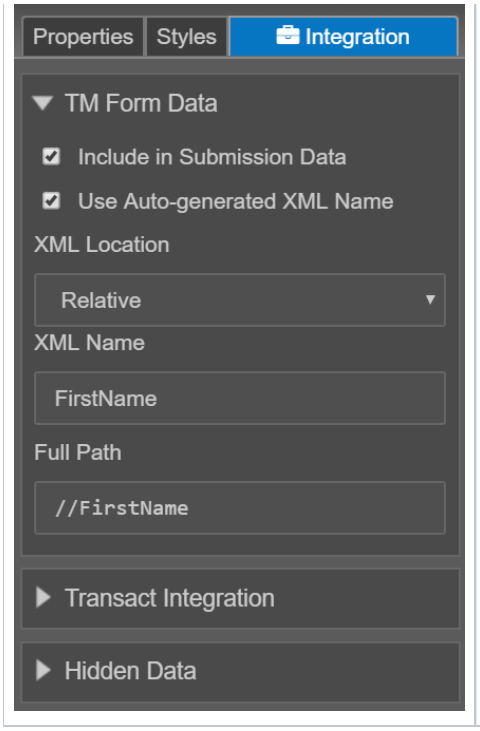
When a label is created based on the ID, the label will convert camel case characters after the last underscore to a readable label. For example, if you type "applicant1_First Name" in ID it will remove the space in the ID to "applicant1_FirstName" and copy the label as "First Name".

XML Name and Entity XML Path

On the Integration panel, you will find the XML Name and/or Full Path fields. By default, the XML Name and/or the Full Path will be populated based on the ID name.

In the screenshots below you can see the difference between an ID without an entity and an ID with an entity.

ID without Entity	ID with Entity
	



You can deselect *Use Auto-generated XML Name* to change the XML Location, XML Name, and Full Path. Deselecting the *Use Auto-generated XML Name* will stop the XML Name and Full Path from being automatically updated when the component ID changes.

You can deselect *Use Entity XML Path* to change the Full Path. Deselecting the *Use Entity XML Path* will stop the Full Path from being automatically updated based on the component ID changes.

Working with Component IDs

Unknown macro: 'redirect'

Page Contents:

Related Pages:

- Add Shared Component to Form
- Change Component ID
- Delete a Component

Duplicate ID

When entering the ID for a component, you may receive a warning if the ID already exists on another component.

You will have the option to *Review* the ID that you entered, or *Accept* an updated ID which will contain a suffix.

Accepting a suffix will add an underscore and a number to the end of the ID you entered. Please note that the XML Name will be the default name used for the component type selected. You will need to manually update the XML Name on the Integration panel.

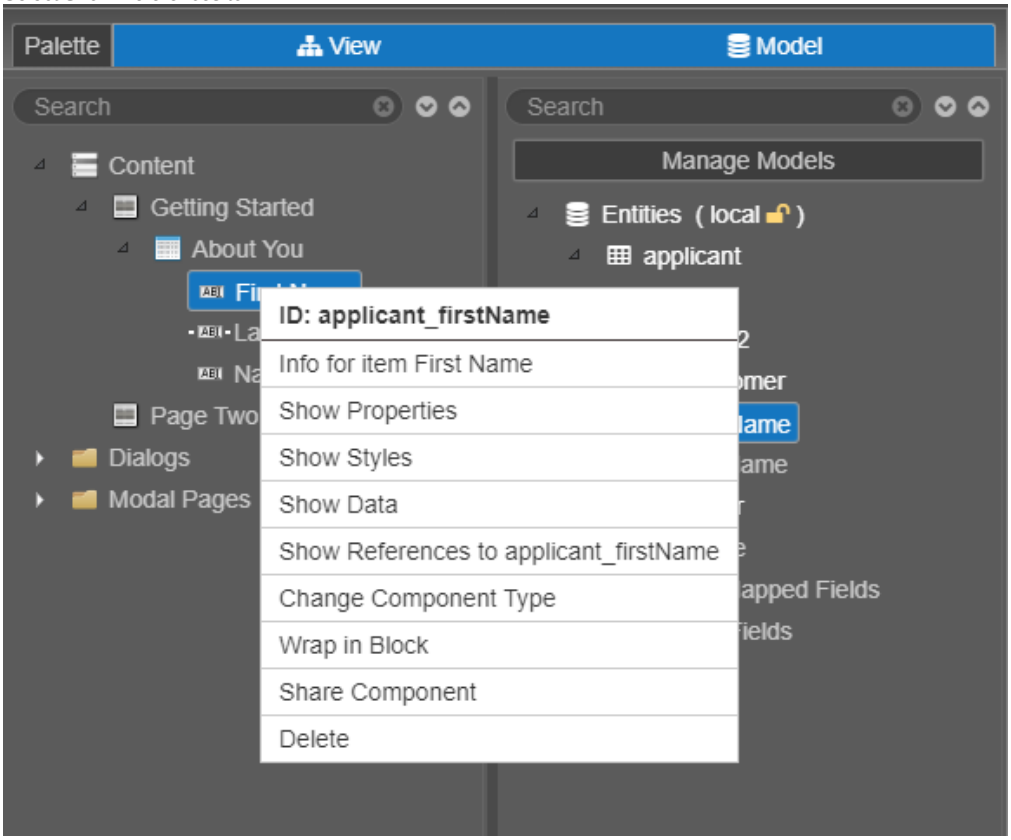
Show References

The *Show References* option will allow you to view all references of the selected ID. The *Show References* option scans the code and properties for field references.

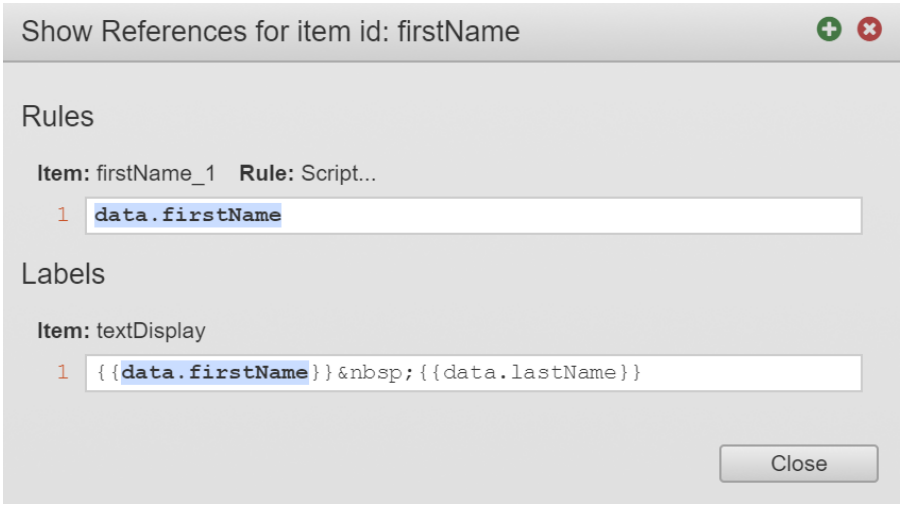
To view the references for a component:

1. Right-click the component in the View Panel

2. Select *Show References to*



This will display a dialog box that shows all references to the selected component ID.



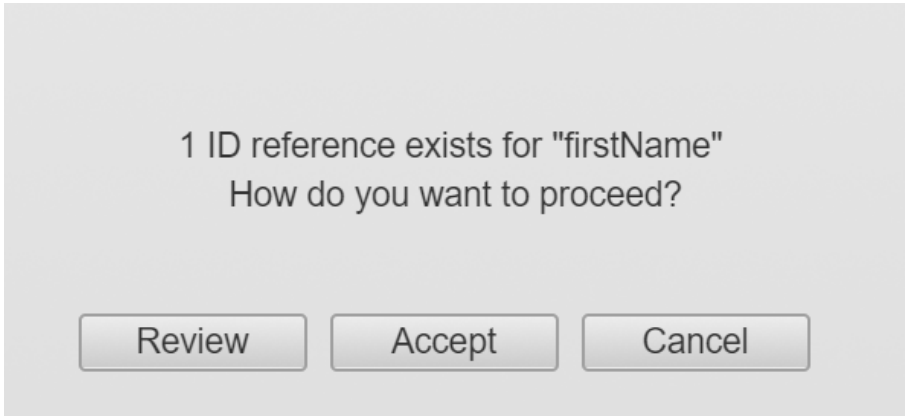
Change Component ID

Unknown macro: 'redirect'

Change Component ID

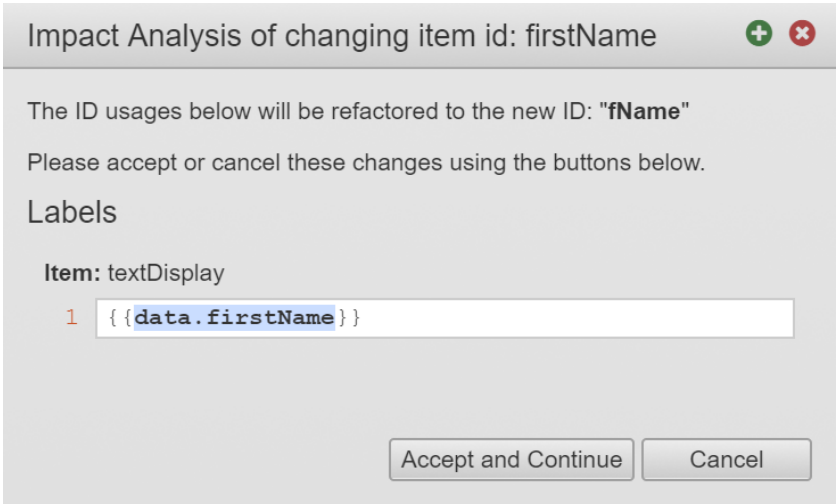
When you first add a new component to the form you will be prompted to change the ID property. Once you enter an initial ID you can change the ID later if needed. When you change the ID the XML Name and /or Full Path will be updated, but the Label will not. The Label is only updated the first time the ID is created.

When you change the ID, if there are any references to the existing ID a pop up window will display. You will have the options to Review, Accept, or Cancel.



Review: Clicking Review will open the impact analysis dialog.This dialog will display a list of references to the current ID (prior to changing).

From the impact analysis dialog you can select Accept and Continue or Cancel. Clicking Accept and Continue will change the ID and refactor all of the references listed. Clicking Cancel will close the dialog and the ID will not be changed.



Accept: Clicking Accept will change the ID and refactor all references.

Cancel: Clicking Cancel will close the dialog and not change the ID.

Related Pages:

- Add Shared Component to Form
- Change Component ID
- Delete a Component

Delete a Component

Unknown macro: 'redirect'

Page Contents:

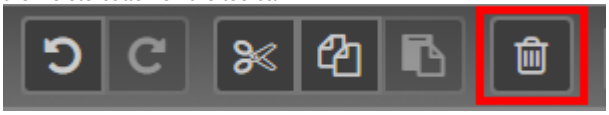
Related Pages:

- Add Shared Component to Form
- Change Component ID
- Delete a Component

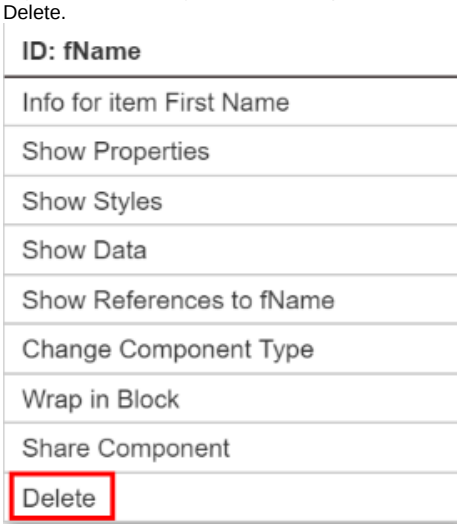
Delete a Component

You can delete a component from a form using one of the following methods:

1. Delete Button: Select the component from the Structure panel or directly on the form, then click the Delete button on the toolbar.



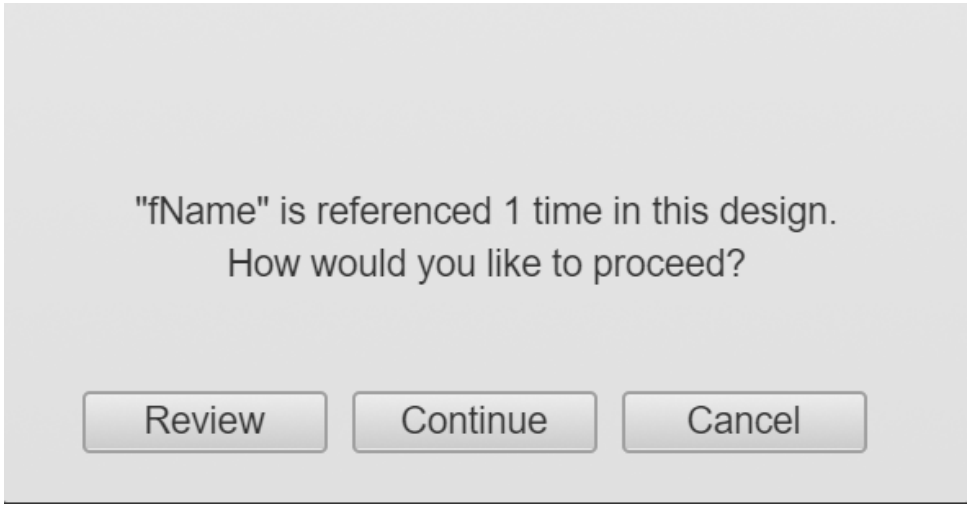
2. Right-click menu: Right-click the component in the View Panel or directly on the form, then click Delete.



3. Delete button (keyboard): Select the component from the Structure panel or directly on the form, then press Delete on your keyboard.

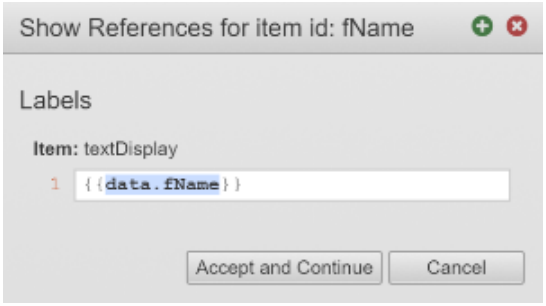
Impact Analysis of Deleting a Component

If the component you are deleting has any references to it, a dialog will display when you delete the component.



Review: This will display an impact analysis which will list any references to the component you want to delete.

In the impact analysis dialog you
can select Accept and Continue or Cancel. Accept and Continue will delete the selected



component. Cancel will close the dialog and not delete the component.

Selecting Accept and Continue will keep any references to the deleted component. You will need to make changes in the form as required.

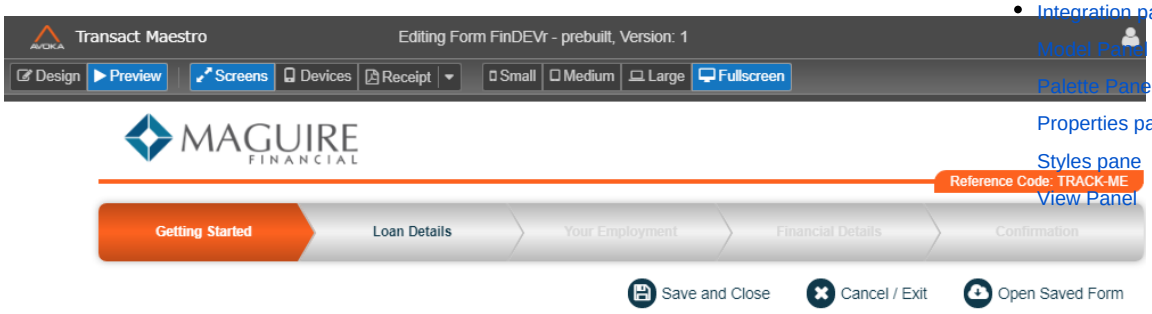
Continue: This will delete the component, without reviewing the current references. Any references to the deleted component will remain in the form. You will need to make changes as required.

Cancel: This will close the dialog and keep the component.

Form Preview

Unknown macro: 'redirect'

When you are in Design Mode, you see the form layout, and can click and drag components, but you can't actually interact with it as an end-user. In order to interact with the form, click on the Preview button in the toolbar.



In Preview mode, you can interact with the components on the form as if it were live. A number of additional options are available on the toolbar:

- **Screens** - This allows you to select from Small, Medium, Large and Fullscreen screens, to see how your form will display and behave at each of these screen sizes.
- **Devices** - This allows you to preview your form on a small number of devices. This mode does not actually run on the device, but simply shows what it would look like.
- **Receipt** - This shows how the receipt rendition of your form will look. This has all navigation and buttons removed and displays all pages inline. It also often reduces the color in the form to make contrast better and use less ink.

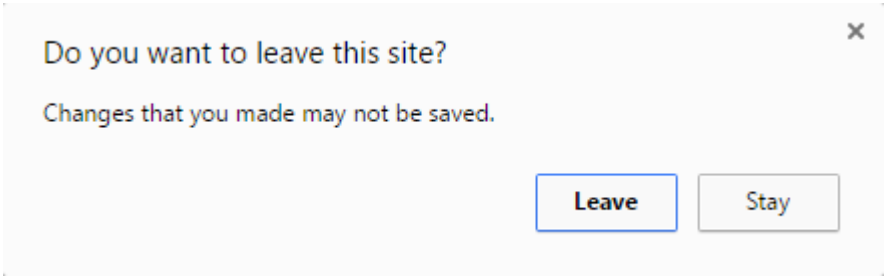
Related Pages:

- Closing the Form
- Customizing the Maestro Editor
- Form Preview
- Integration panel
- Model Panel
- Palette Panel
- Properties pane
- Styles pane
- View Panel

Closing the Form

Unknown macro: 'redirect'

Once you have finished working with your form you can close the form browser tab. If you have not saved any recent changes you will receive a warning message prior to closing the form.



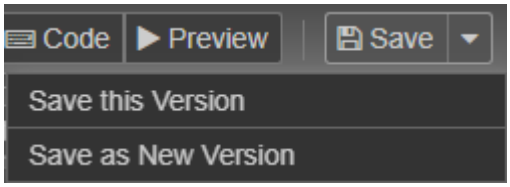
If you want to save your changes you will need to click *Stay*, then click the *Save* button on the toolbar at the top of the form window.

If you don't want to save the changes you can click *Leave*. This will close the form without saving.

When you edit or create a new form it is opened in a new browser window. When you close a form you will be able to navigate to the Dashboard window to work with other Maestro assets or resources. If you are done working with Maestro you can logout of Maestro and close the browser window.

Maestro will log you out automatically after a certain period of inactivity.

Save Dropdown Options



Clicking the *Save* button is the same as selecting *Save this Version.* This will save the open version of the form.

Save as New Version will create a new version of the open form. The changes made to the form will be saved in the new version and the previous version will revert back to the previous save state. See Form Versions for more information.

elated Pages:

- Closing the Form
- Customizing the Maestro Editor
- Form Preview
- Integration panel
- Model Panel
- Palette Panel
- Properties pane
- Styles pane
- View Panel

Managing Maestro Forms

Unknown macro: 'redirect'

The content within this section covers information relating to managing a form in Maestro.

Below you will find a list of topics within this section.

- Form Operations
- Form Versions
- Component Versions
- Change Template
- Save History
- Import and Export

Form Operations

Unknown macro: 'redirect'

Related Pages:

- Change Template
- Form Operations
- Import and Export
- Save History

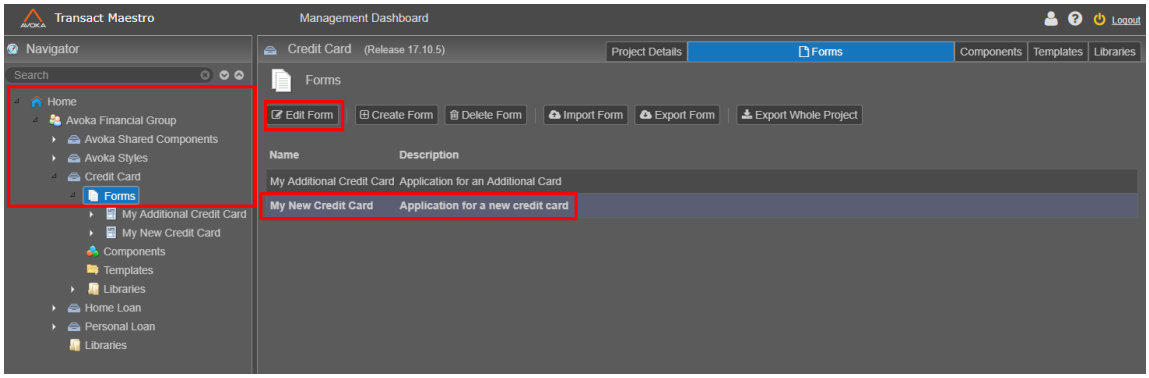
Page Contents:

Open an Existing Form

A form can be opened in the Editor window from the Management Dashboard. There are two methods to open the form.

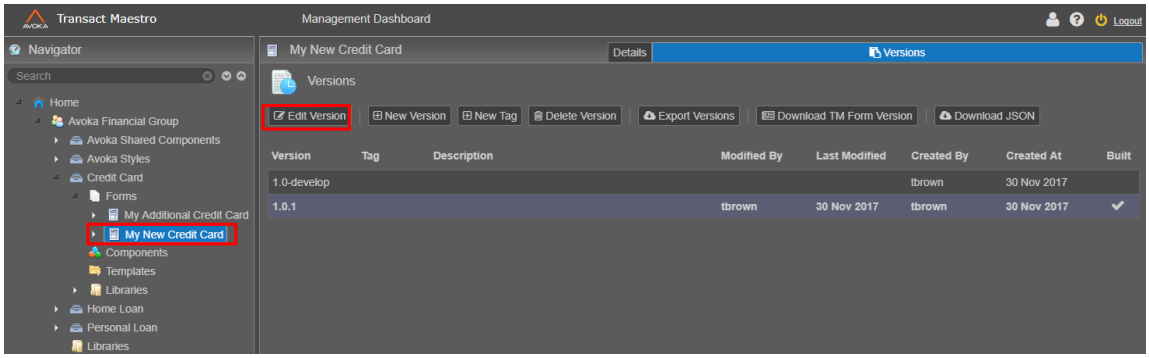
Method 1:

1. Expand *Home*
2. Expand the *Organization*
3. Expand the *Project*
4. Select *Forms*
5. Select the form you want to open
6. Click the *Edit Form* button.



Method 2:

1. Expand *Home*
2. Expand the *Organization*
3. Expand the *Project*
4. Expand *Forms*
5. *Double-click* the form from the Navigator pane or select the form then click *Edit Version*

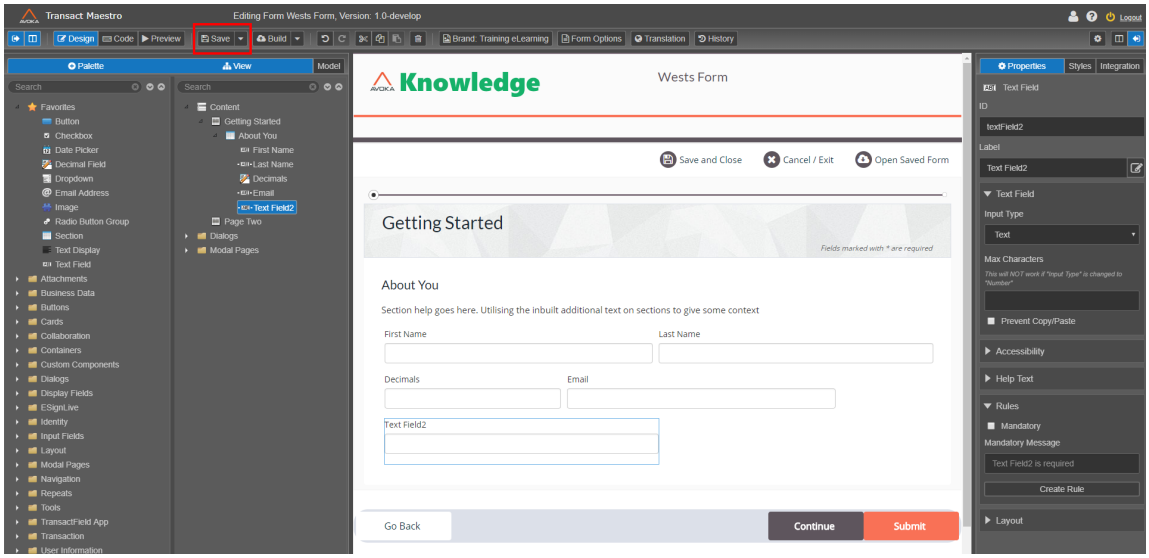


Using either method, the form will open in the Editor window where you can change or preview the form.

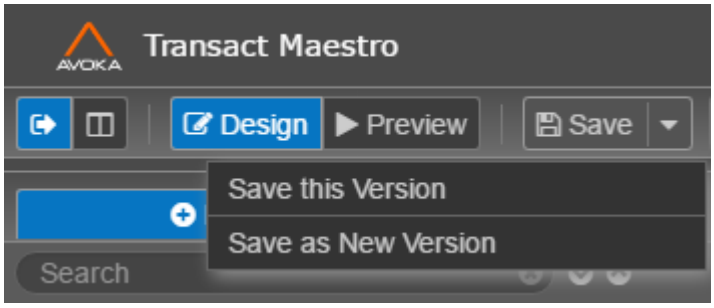
Save a Form

When you are working on form it is good practice to save regularly.

In the Editor window you will see the save button on the toolbar. The save button will be active only if there are changes to save.

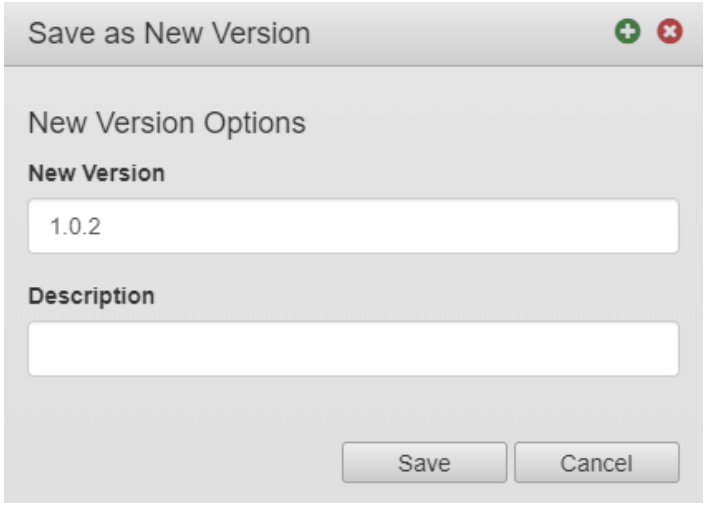


There are multiple save options available:



Save this Version: This option is the same as clicking the Save button. It will save the changes to the currently open form version.

Save as New Version: This option will allow you to save the changes to a new version of the form. When you select this option the Save as New Version window will display. You can enter the new version number and a description in the window. For more information on versions see Form Versions.



Close a Form

When you open a form for editing, the form will open in a new browser tab and the Management Dashboard will remain open in its original tab. When you are finished editing a form you can close the relevant tab and return to the Management Dashboard. You will be prompted to save your changes if you close without saving. You can open multiple forms and other assets as needed. Each will display in their own browser tab.

Copy a Form

In Maestro you can copy existing forms if needed. This will allow you to reuse existing forms that may have similar components to a new form. Reusing existing forms will save you time since you don't have to start the new form from scratch. See Import and Export for more information.

Form Versions

Unknown macro: 'redirect'

Page Contents:

If you are using Maestro 17.10.x with a TM 5.x environment, and your form version number is longer than 10 digits (e.g. "1.0-develop"), you will get an error when you attempt to import the 17.10.x form to the TM 5.x environment. In this case, it is recommended that you shorten the default suffix when creating new form versions in Maestro 17.10.x.

Overview

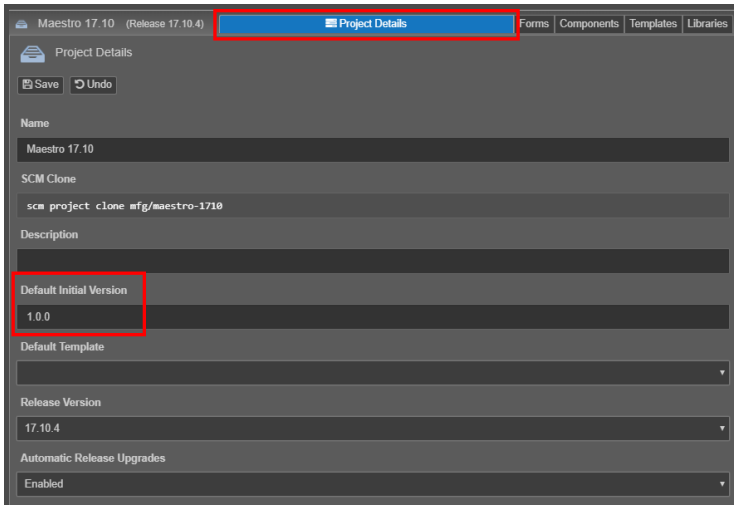
Form Versions are a great way to save a unique copy of a form. You may want to create a new form version each time the TM form version is built, or when you are making significant changes to the form.

Creating a new form version at these points may save you the hassle of fixing your form should something go wrong. For example, say you're adding a feature or changing the behavior of existing form items but while making these changes something goes wrong. Form versions allow you to easily return to a previous working version of the form.

The Transact platform follows open source semantic versioning conventions. By following this convention, all new forms are recommended to start at version 1.0-develop until they reach production.

When you create a new form, the initial form version will depend on the "Default Initial Version" applied from the *Project Details* page. The "Default Initial Version" will be set when a new project is created.

The screenshot below highlights the "Default Initial Version" field in the *Project* Details tab.



The "Default Initial Version" can be changed at any time by switching to the *Project Details* tab and changing its value. This will only impact new forms and assets; version numbers of existing forms are unaffected. Forms are displayed in the Maestro interface, sorted by form version in descending order.

Semantic versioning in Maestro provides the following benefits:

- Standardize versioning of Maestro design artifacts which in turn makes management of different versions easier.
- Align more closely with TM versioning.
- Assist Form Builders who are developing concurrently.
- Lock down a form version (using tags).

When working on any form, it it recommended that you do all your work on version 1.0-develop (the initial form version) until the form is ready for production. When a form is ready for production, it is recommended that you create a tagged version of the form.

Related Pages:

- Change Template
- Collaboration Job Information for a Form Version (Manager v5.1)
- Custom Services for a Form Version (Manager v5.1)
- Delete a Form Version (Manager v17.10)
- Delete a Form Version (Manager v5.1)
- Export a Form Version (Manager v17.10)
- Export a Form Version (Manager v5.1)
- Form Operations
- Form Version Categories (Manager v5.1)
- Import and Export

Form Version Numbers

A form version can contain numbers and periods (.), and will always be up to three levels. Form version numbers are based on major, minor and patch releases (e.g. 1(major).1(minor).3(patch)). If a version number includes less than three levels, Maestro will add the missing levels as "0". For example, if "1" is entered as the new form version, it will become "1.0.0" when you navigate away from the field.

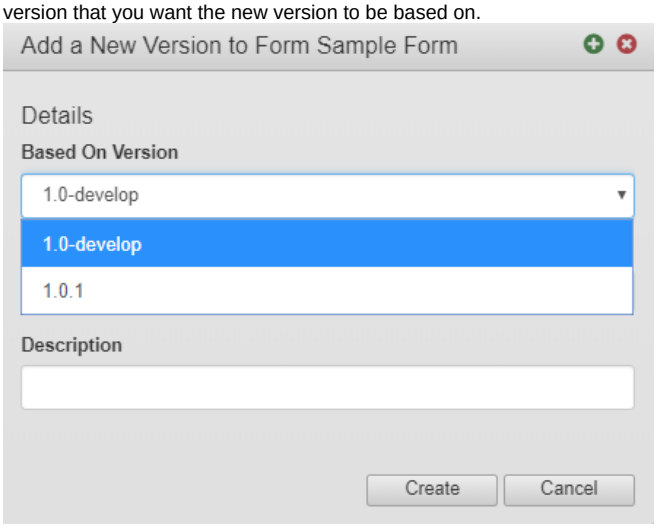
The default version number for a form is an incremented value based on the project's Default Initial Version property and stripping away any non-numeric suffix. For example, if the initial version is "1.0-develop", the next version will be "1.0.1".

Create a New Version

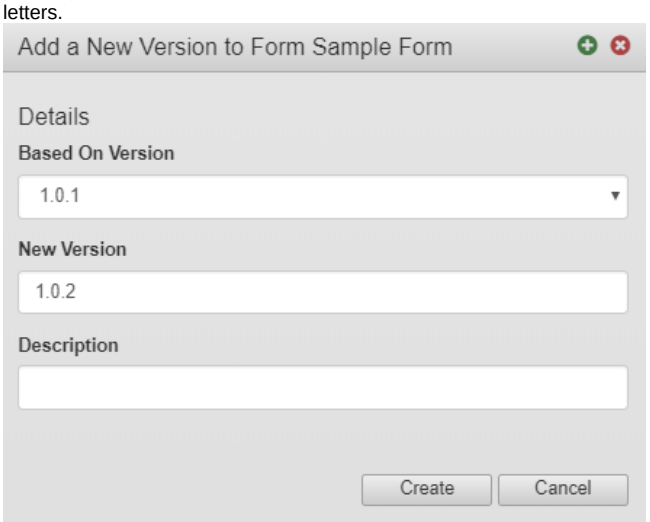
There are two ways to create a new form version for existing forms - from the Maestro Management Dashboard or from the Maestro Editor window.

From the Management Dashboard

1. Navigate to the Forms folder within the project.
2. Select a form to view the existing versions (you can also expand a form from the Navigator panel to see all form versions).
3. Click *New Version*.
4. Complete the Add a New Version dialog.
    a. *Based On Version* - The Based On Version field allows you to select a previous form version that you want the new version to be based on.

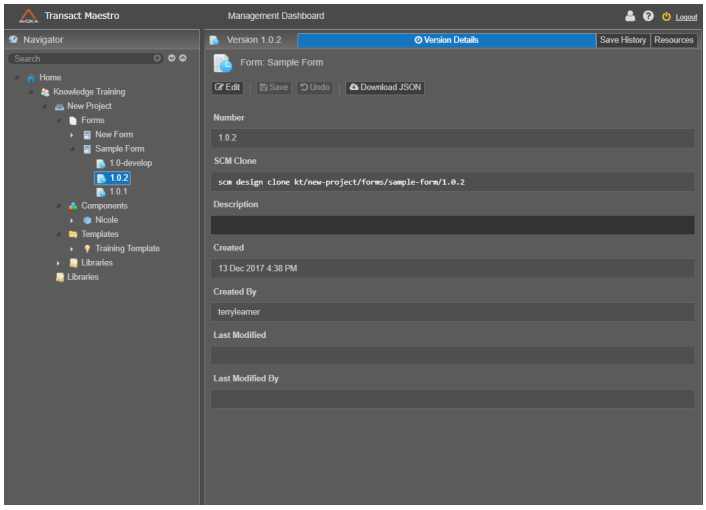


    b. *New Version* - The New Version field allows you to enter the version number you want to use for your new form version. You can also include a suffix that contains numbers and letters.



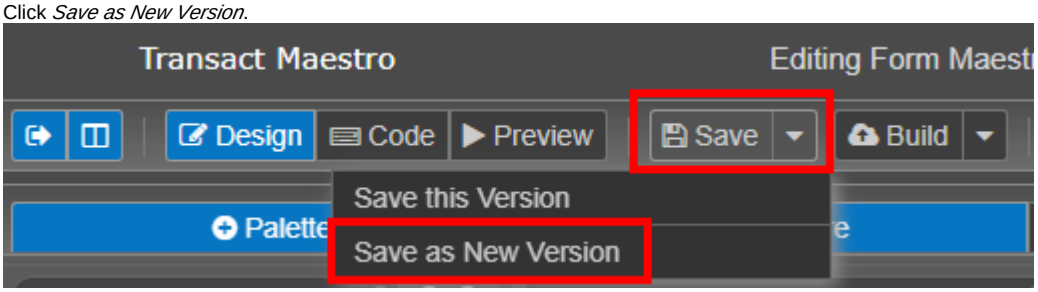
    c. *Description* - Enter a description to help organize the different versions of a form.
5. Click *Create*.

The new form version will be created and its details will be displayed in the Dashboard. You will also see the new form version listed under the form in the Navigator panel. With the version selected, you can click *Edit* to open the selected version in the Editor window.

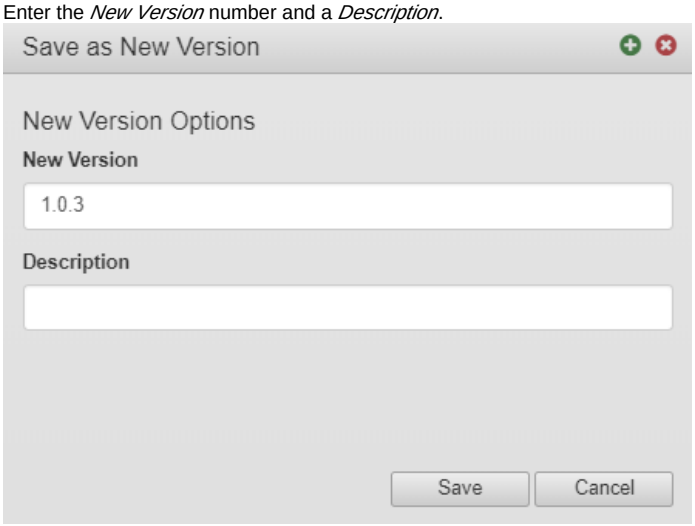


From the Editor Window

1. With the form open in the Editor Window, select the dropdown arrow on the *Save* button.
2. Click *Save as New Version*.



3. Enter the *New Version* number and a *Description*.



4. Click *Save*.

The new form version will be opened in the same browser tab.

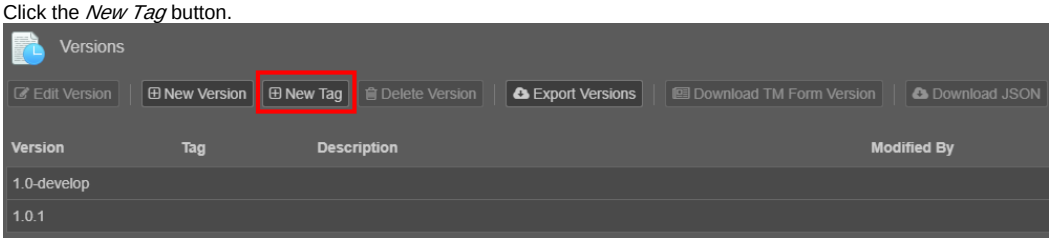
Form Tag Version (locked version)

A tagged version of a form is a read-only version that prevents any changes from being saved to the form. A tagged version is an effective way of keeping a clean, final record of a form. It is recommended that you create a tagged version each time a form is deployed to an environment. For instance, while you are developing a form, you should be using the "1.0-develop" form version; once this version of the form is ready for production and deployment to an environment, it is time to create a tagged version of the form.

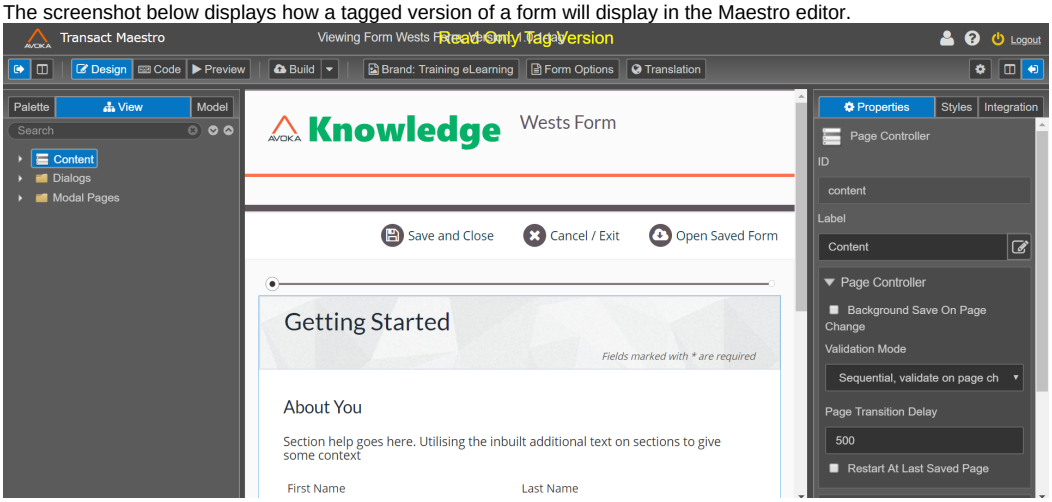
To create a tagged version of a form, follow the steps below:

1. Select the form from the Navigator on the Dashboard.

2. Click the *New Tag* button.



3. Complete the options in the New Tag window.
    a. Based On Version - this option allows you to select which version of the form you want the tagged version to be based on.
    b. New Version - this option allows you to enter a version number for the tagged version.
    c. Description - this option allows you to enter a description for the tagged version.
4. Click Create.
   A new tagged version of the form is created.
   The screenshot below displays how a tagged version of a form will display in the Maestro editor.



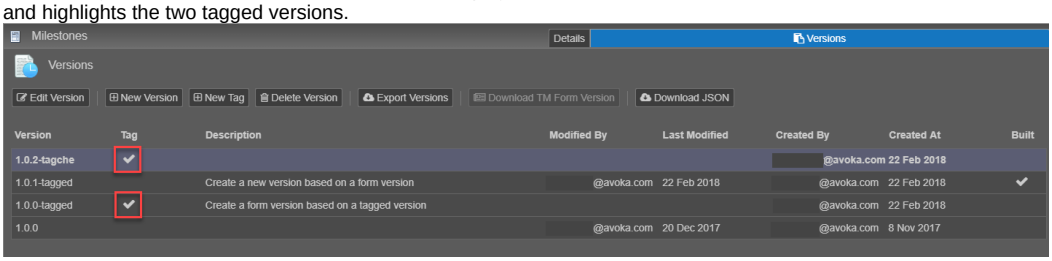
From the Versions list, you will see a check mark in the column next to the tagged version. If you open the tagged version in the editor window, you will see the read-only text at the top of the form. In the read-only version, you will notice that you are not able to add new components to the form, and there is no option to save any changes you make.

Create a New Version of a Form based on a Tagged Version

To create a new version of a form based on a tagged version, follow the steps below.

1. Navigate to the Forms folder within the project.

2. Select a form to view the existing versions (you can also expand a form from the Navigator panel to see all form versions). If the form contains any tagged versions, each tagged version will be identified by a check mark. The screenshot below displays a list of available versions for a form and highlights the two tagged versions.



3. Select a Tagged Version (in the above screenshot, the tagged version "1.0.2-tagche" has been selected).

4. Click *New Version.*

5. Complete the Add a New Version dialog.
    a. *Based On Version* - The Based On Version field allows you to select a previous form version that you want the new version based on. By selecting the tagged version "1.0.2-tagche" (in Step 3 above) before clicking *New Version* (in Step 4 above), this will be the form version that the new form version will be based on.

b. *New Version* - The New Version field allows you to enter the version number you want to use. You can also include a suffix that contains numbers and letters. When creating a new version, the third (patch) part of the version number will be incremented (by 1) to the previous version of the form. Since you are creating a new version of the form based on a tagged version, it may be appropriate to increment the first (major) part of the version number by 1 (for example, version 1.0.2-tagche will become 2.0.0- develop). The screenshot below displays a new version being named 2.0.0-develop.

c. *Description* - You can enter a description to help organize the different versions of a form.

6. Click *Create*
   The new form version will be created and its details displayed in the Maestro Dashboard. You will also see the new version listed under the form in the Navigator panel. Even though the new form version is based on a tagged version, you can edit it because it wasn't created as a tagged version. You can edit the new form version in the usual ways; for example, select the new form version in the Navigator and click Edit to open it in the Editor window.

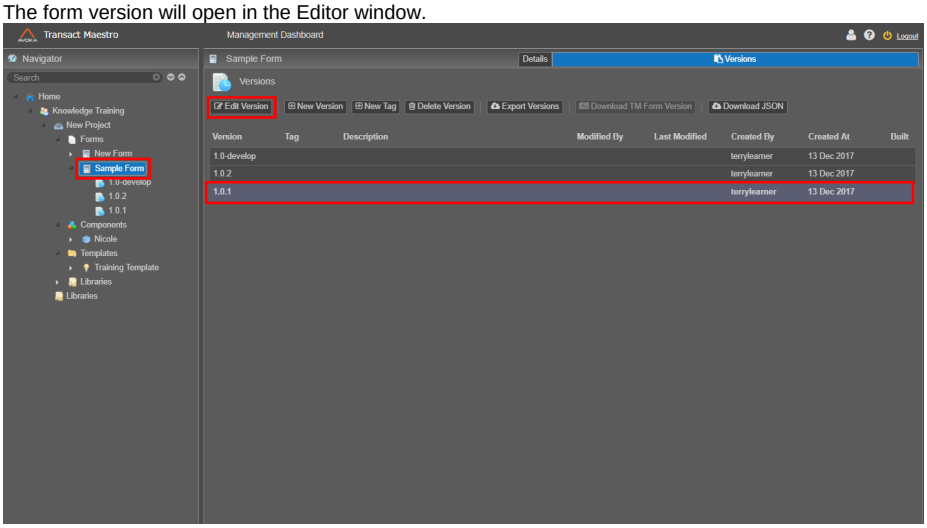
Open an Earlier Version of a Form

When you open an existing form by double-clicking the form in the Navigator, the latest version of the form will be opened. Although this is normally what you want to do, sometimes you may want to open a previous version of the form. There are two methods to open an earlier version of a form in Maestro.

Method 1:

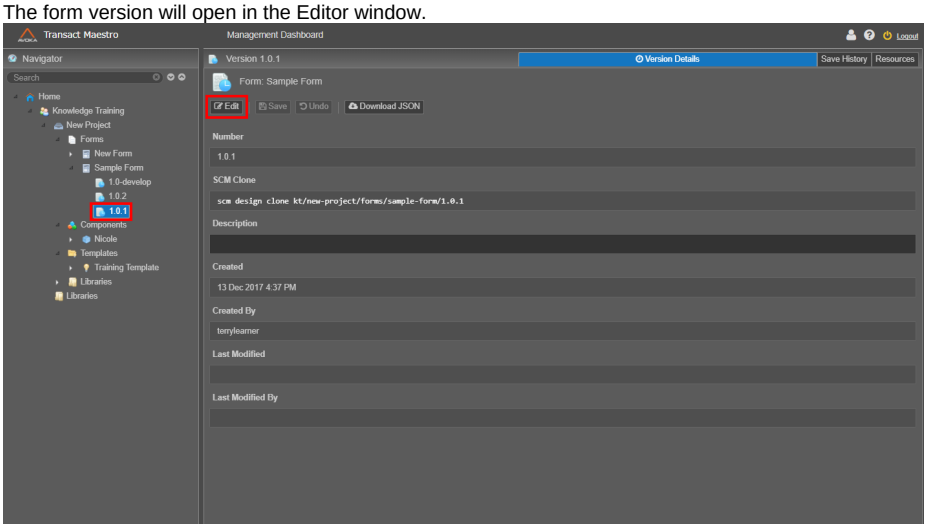
1. Select the form from the Navigator panel
2. Select the version you want to open

3. Click *Edit Version*
   The form version will open in the Editor window.



Method 2:

1. Expand the form in the Navigator panel
2. Select the version you want to open
3. Click Edit
   The form version will open in the Editor window.

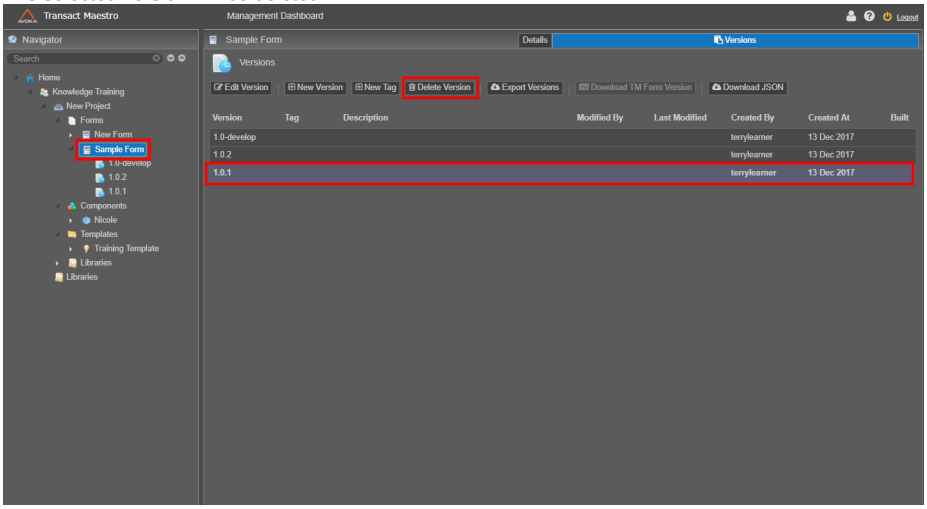


Delete a Form Version

To delete a form version:

1. Select the form from the Navigator panel
2. Select the version to delete
3. Click *Delete Version*

4. Click *OK* in the confirm deletion window.
   The selected version will be deleted.

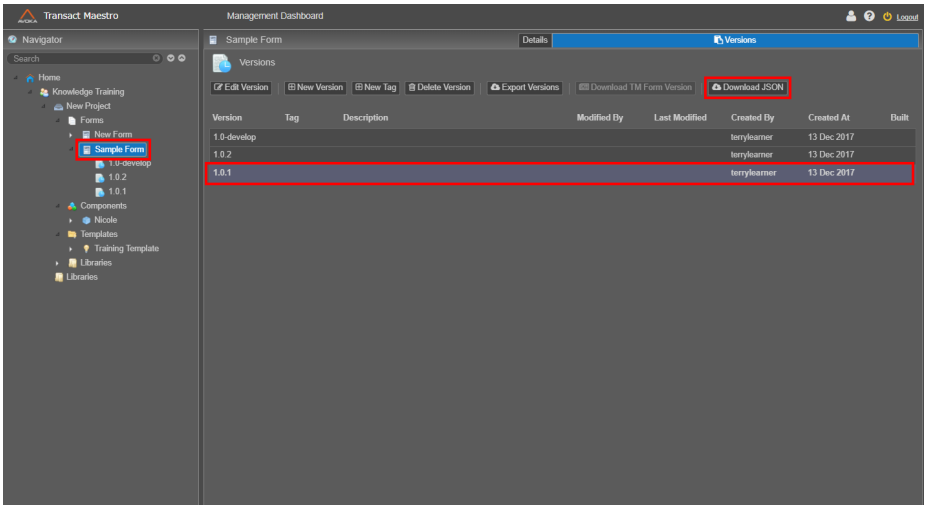


Compare Form Versions

If you have multiple versions of a form, you can download the JSON file of each version, and use an external tool to compare the differences between the form versions. There are two methods of downloading the JSON file of a form version.

Method 1:

1. Select the form from the Navigator panel
2. Select a version from the list
3. Click the *Download JSON* button

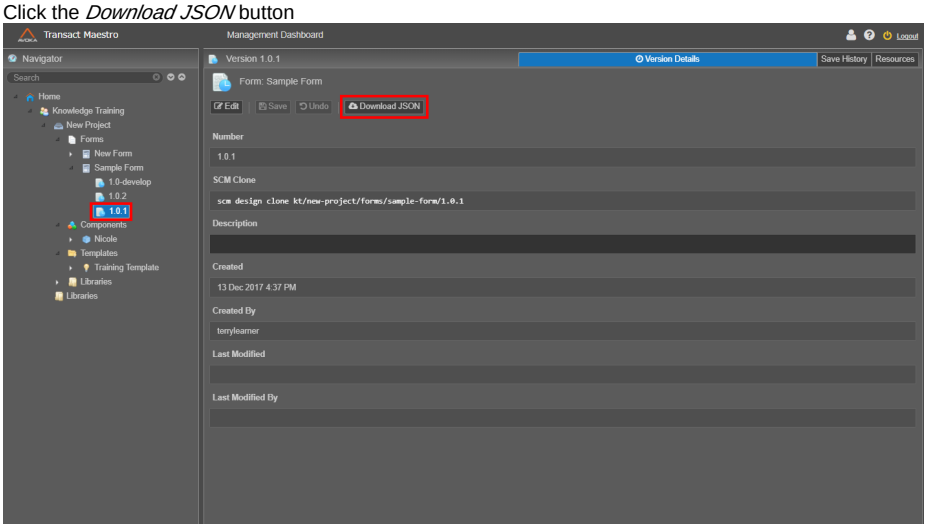


4. Select a different version from the list
5. Click the *Download JSON* button
   Using an external tool, you can compare the differences between the two downloaded JSON files.

Method 2:

1. Select the version from the Navigator panel

2. Click the *Download JSON* button



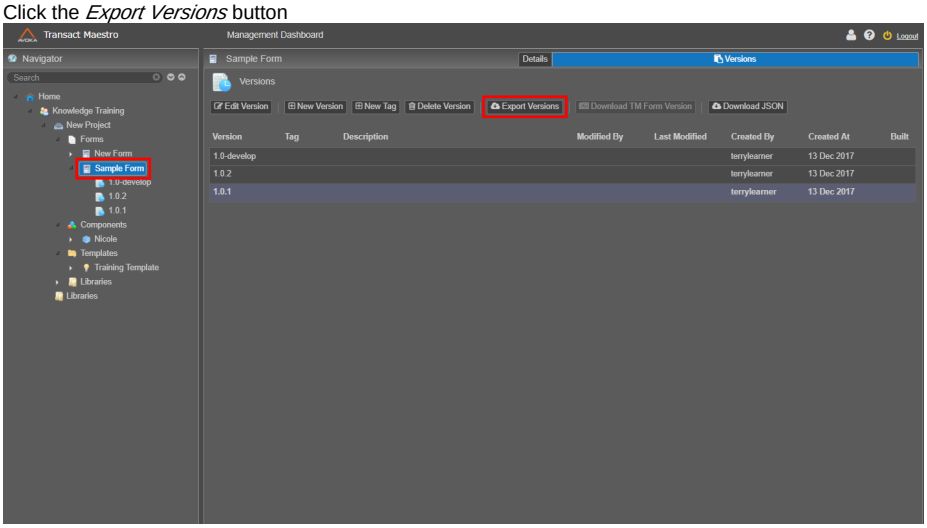
3. Select a different version from the Navigator panel
4. Click the *Download JSON* button

Export Form Versions

Maestro allows you to export all versions, or selected versions of a form.

To export a single form version, or multiple versions of the same form:

1. Select the form from the Navigator panel
2. Click the *Export Versions* button



3. Select the versions you want to export
   You can select one or more versions to include in the export. You can also use the *Select All* butto
   n to easily select all versions of the form.

4. Click *Export*
   Once the form versions are selected, click Export.
   The selected form versions will be packaged into a single zip file and the zip file will be downloaded to your computer.

Form Version Display Order

Form versions are sorted based on major, minor and patch release numbers and are displayed in Maestro in descending order. The screenshot below shows a list of form versions displayed in Maestro.

Component Versions

Unknown macro: 'redirect'

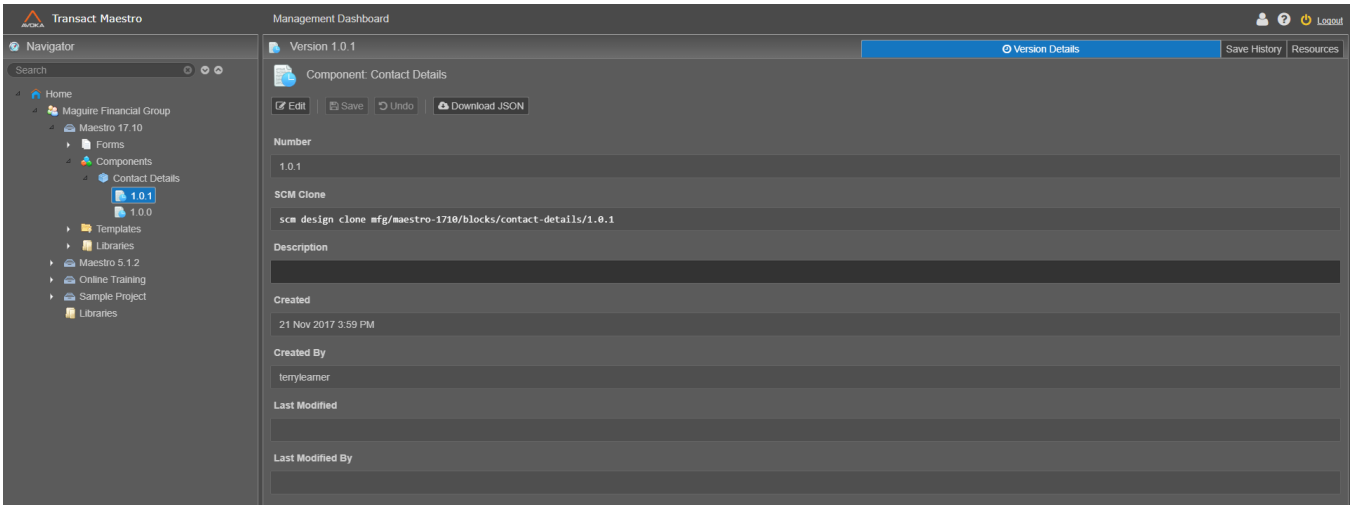
Create a New Component Version

Component versions work similarly to Form Versions. Once you have created a shared component you can create new versions to keep track of any changes.

Create a New Version from the Management Dashboard

1. Navigate to the *Components* folder within the project
2. Select a *component* to view the existing versions (you can also expand a component from the Navigator panel to see the versions)
3. Click *New Version*
4. Enter a *Description*
5. Click *Create*

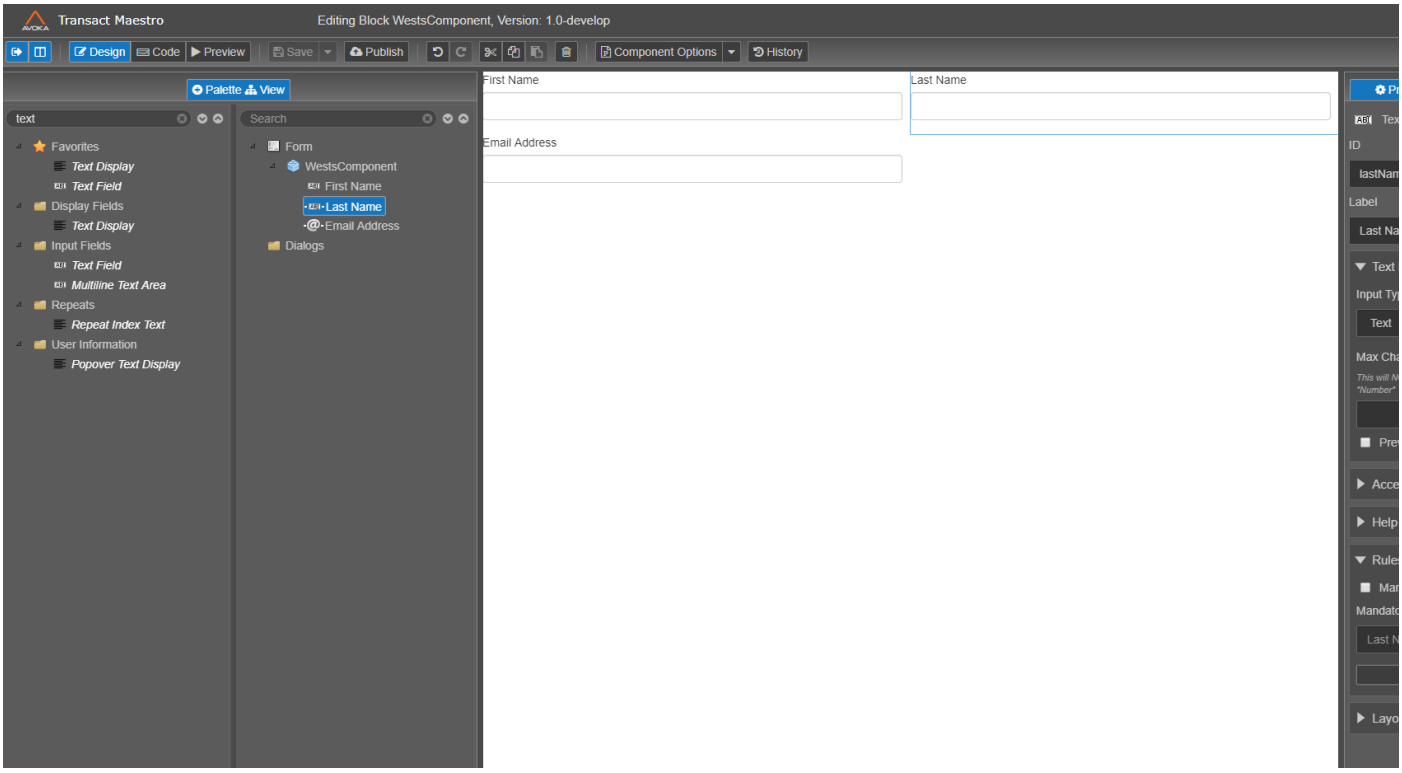
The new version will be created and the details for the version will be displayed in the Dashboard. You will also see the new version listed under the component in the Navigator panel. With the version selected you can click Edit to open the version in the Editor window.



Create a New Version from the Editor Window

1. With the component open in the Editor window select the dropdown arrow on the Save button
2. Click Save as New Version
3. Enter a Description
4. Click Save

The new component version will be saved. The new version will be displayed in the current browser window, unlike a new form version which opens in a new browser tab. At the top of the Editor window, you will see the component name along with the component version.



Please note that to see the new version displayed in the Dashboard you may need to refresh the browser window and unlike all other views, when editing/ creating a component, the Model Panel will not display.

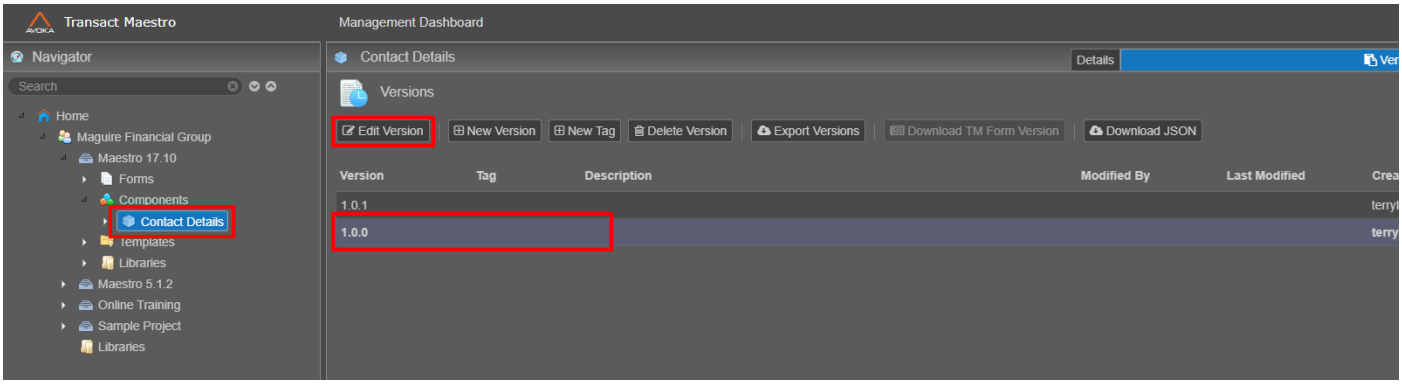
Open an Earlier Version of a Component

When you open an existing component by double-clicking the component in the Navigator, the latest version of the component will be opened.

Method 1:

1. Select the component from the Navigator panel
2. Select the version you want to open
3. Click Edit Version

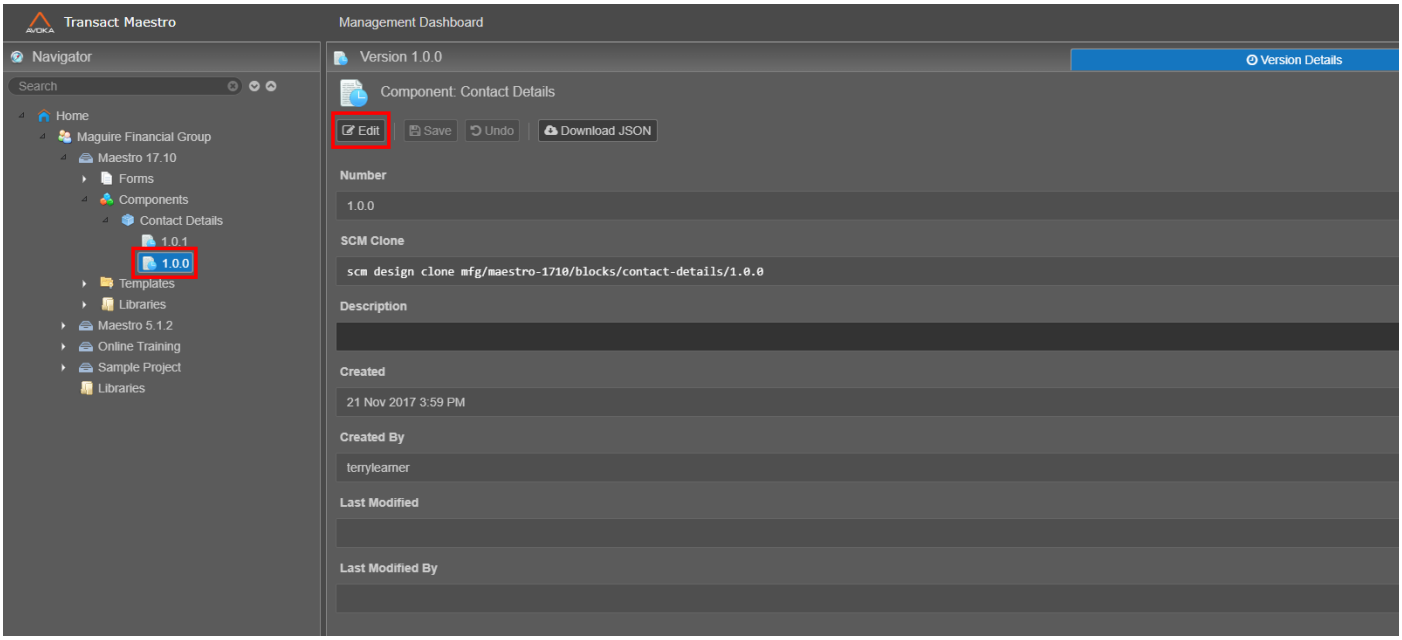
The component version will open in the Editor window.



Method 2:

1. Expand the component in the Navigator panel
2. Select the version you want to open
3. Click Edit

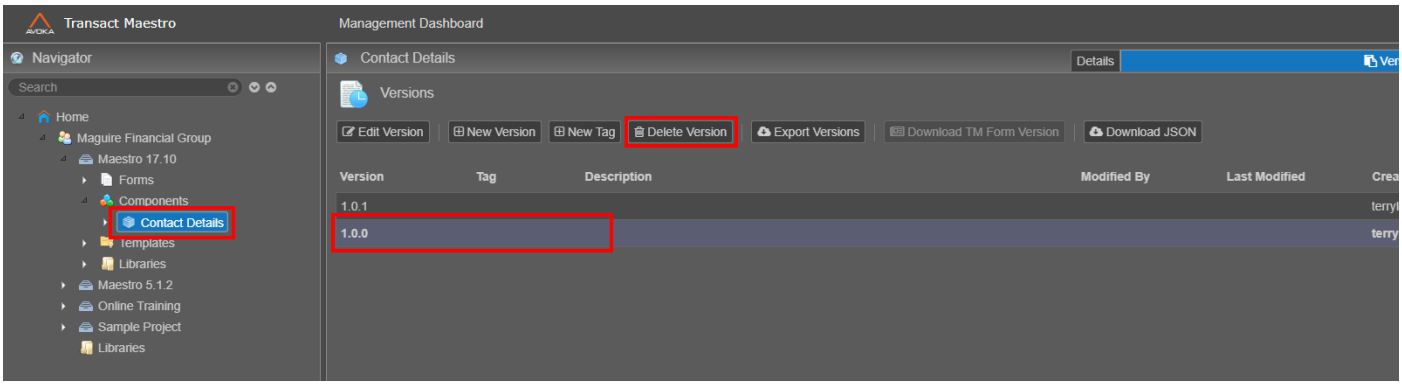
The component version will open in the Editor window.



Delete Version

1. Select the component from the Navigator panel
2. Select the version to delete
3. Click Delete Version
4. Click OK in the confirm deletion window

The selected version will be deleted.



Change Template

Unknown macro: 'redirect'

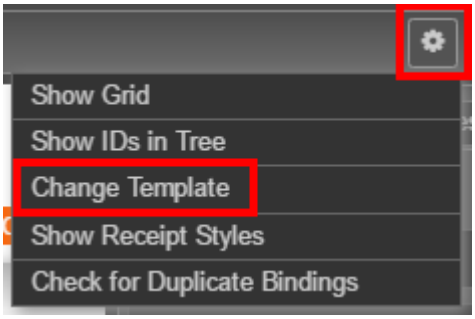
When you first create a form you need to select which template to use. It is important to select the correct template when the form is created, however if it is necessary to change the template of a form you can do so.

> It is recommended that you only change the template when absolutely necessary as changing the template may effect the form.

In a template the template designer can limit the form builder's access to certain areas within the form using extension points. When the template of the form is changed, the extension points in the new template may be different to the extension points in the original template used. If the extension points are different, the form builder may lose content within the form.

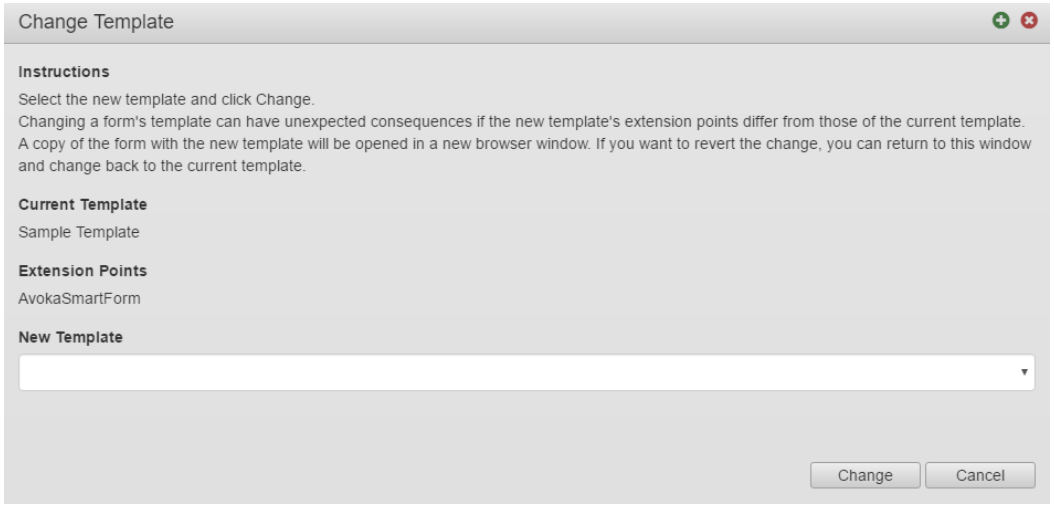
To change a template:

1. Open the form in the Editor window
2. Select the *Settings* icon (located in the top right corner of the window, above the Properties pane - see screenshot below)
3. Select *Change Template*
4. Use the dropdown to select the new template
5. Click *Change*



When you select Change Template the Change Template window will display.

Notice the Extension Points section. This displays the extension points in the current template. AvokaSmartForm means that no specific extension points have been set. When no extension points are set the form builder has access to all areas of the form. Most of the time, the template will have extension points set to limit the form builders access and prevent changes to company branding and/or styleguides.



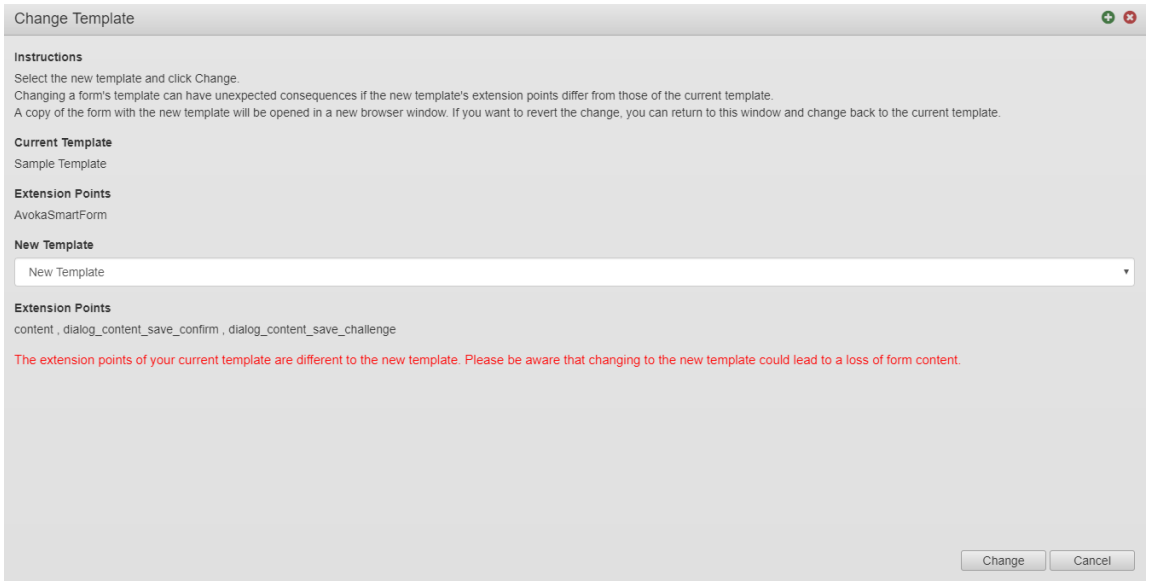
If you select a new template that has different extension points to the current template a warning message will display.

You will also see a list of the Extension Points in the new template. In the screenshot below the template has an extension point set on the content (this is the main area of the form) and two dialogs (save confirm and save challenge).

If you want to continue with the change you can click the Change button.

Related Pages:

- Change Template
- Create a New Template
- Create a New Template
- Creating custom Composer templates
- Customizing the Palette
- Email Templates (Manager v17.10)
- Form Operations
- Form Template Overview
- Import and Export
- Introduction to Templates

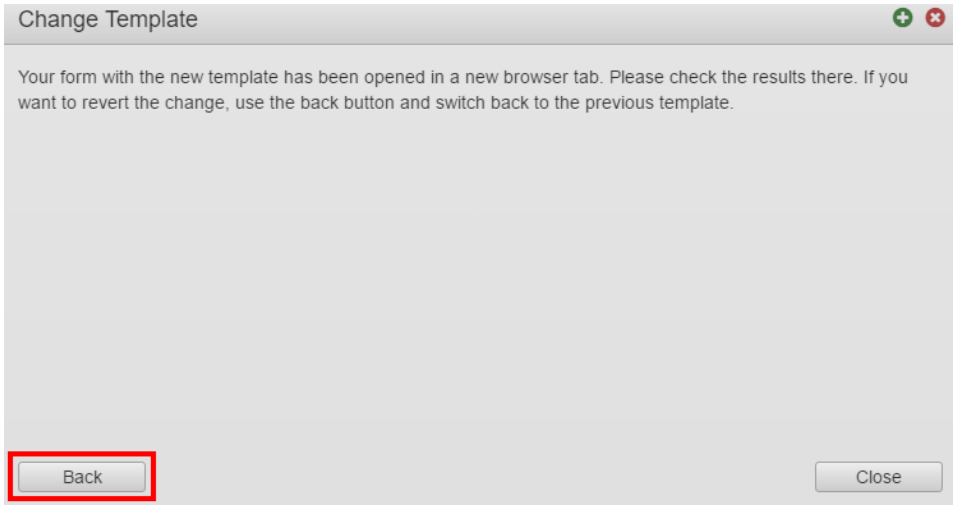


The form with the new template will open in a new browser tab. This will allow you to view the form with the changed template and check the results. If there are any major issues, you can close the new browser tab and return to the form in the original browser tab. If you are happy with the template change you can close the form in the original browser tab and continue working with the form in the new browser tab.

In the Change Template window you will see a message with an option to return to the original template.

1. Click the *Back* button
2. Select the original template from the dropdown
3. Click *Change*

The form will again open in a new window with the selected template applied. You can close the form in the original browser tab and continue working with the form in the new browser tab.



Save History

Unknown macro: 'redirect'

Page Contents:

Related Pages:

- Change Template
- Form Operations
- Import and Export
- Save History

Overview

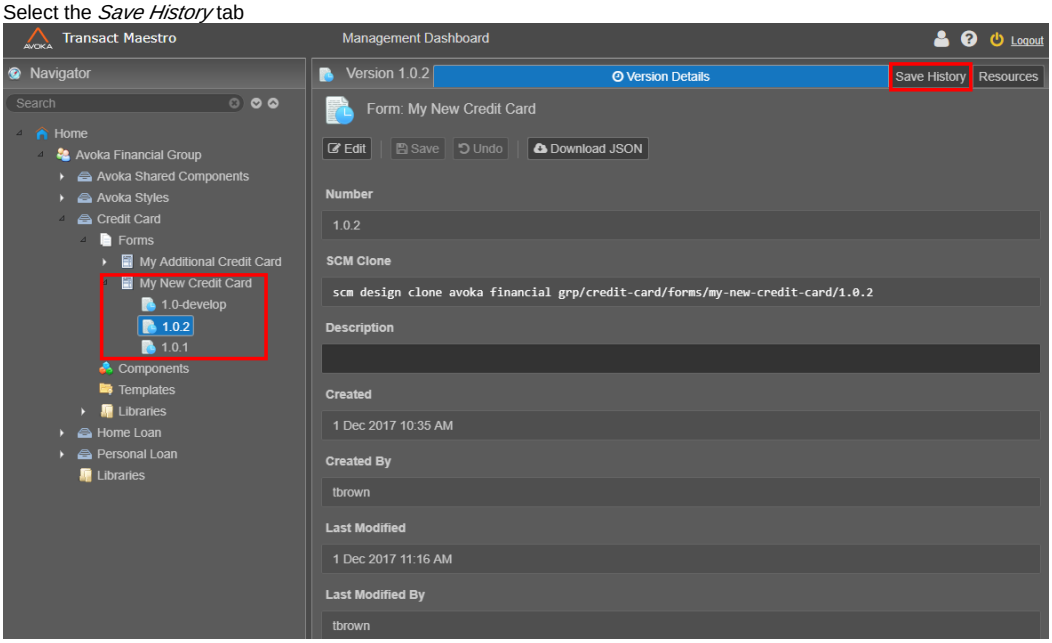
Every time a form, template or component is saved, a compressed copy of the asset is sent to the server. This allows users to view the save history of assets, and restore the asset to a previously saved version.

The saved assets can be viewed from the Form Dashboard. If needed, you can also revert an asset to a previous saved version.

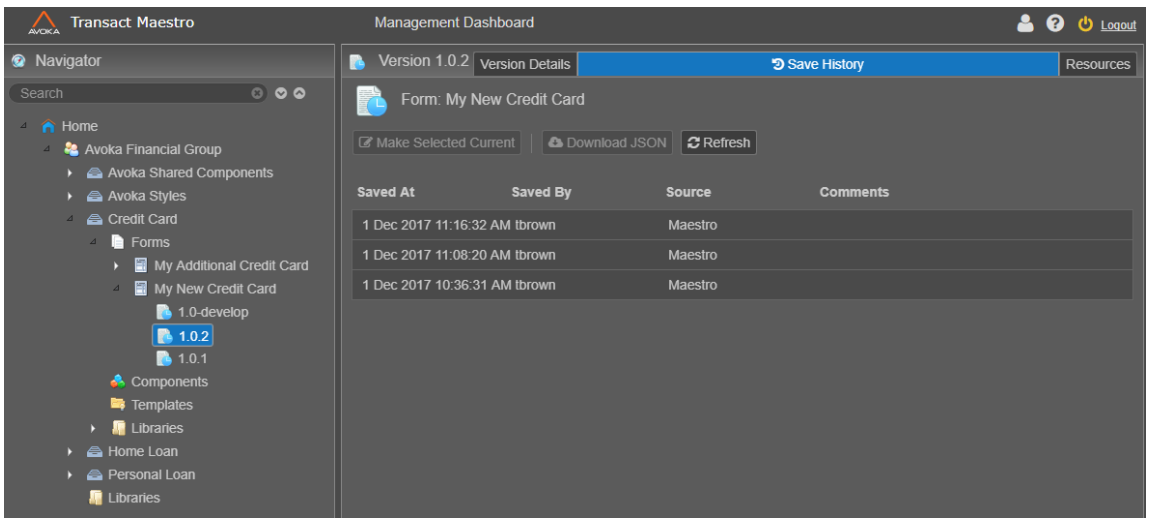
View Save History

To view the save history you will need to select the form version in the Management Dashboard. The same process can be used to view the save history of a template or component.

1. Select the *Form Version* you want to view
2. Select the *Save History* tab



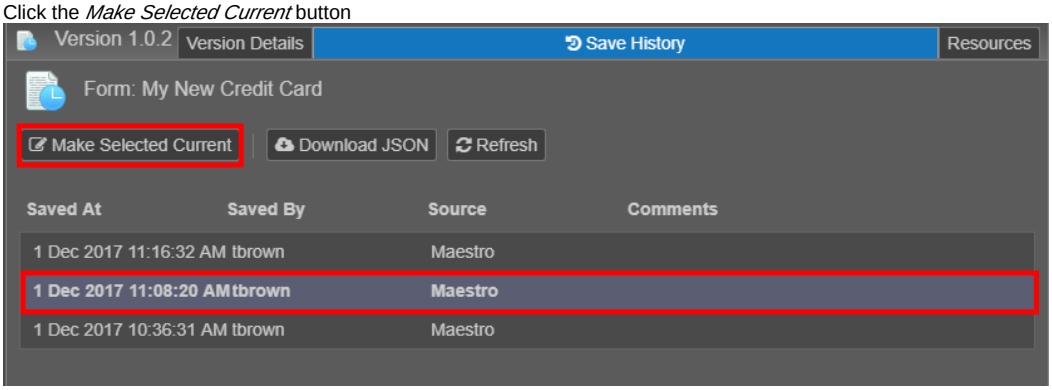
The Save History tab will display a list that shows each time the form was saved.



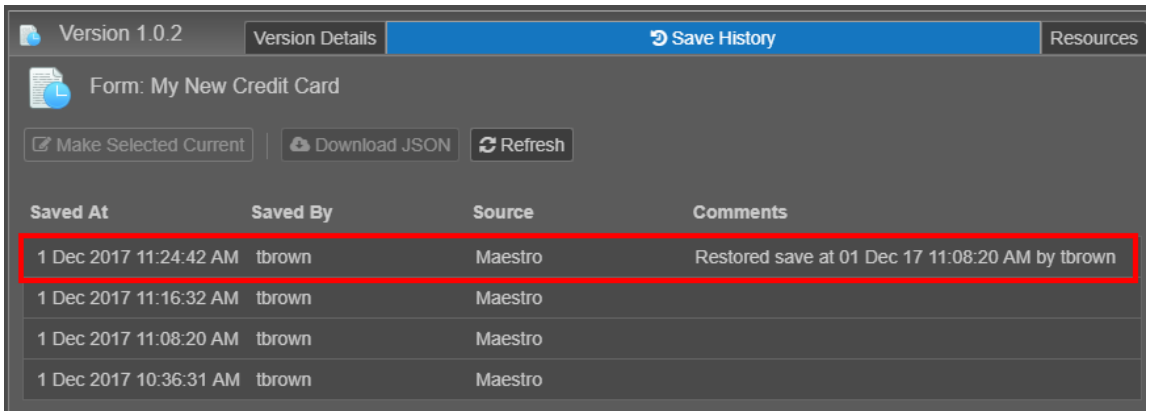
Revert to Previous Save

If needed, you can revert your form (or template or component) to a previous saved version.

1. From the *Save History* list select a previous saved instance
2. Click the *Make Selected Current* button



A copy of the previous saved version will be created and added to the top of the list. Notice that a note is added next to the saved version that indicates which previous saved version was restored.

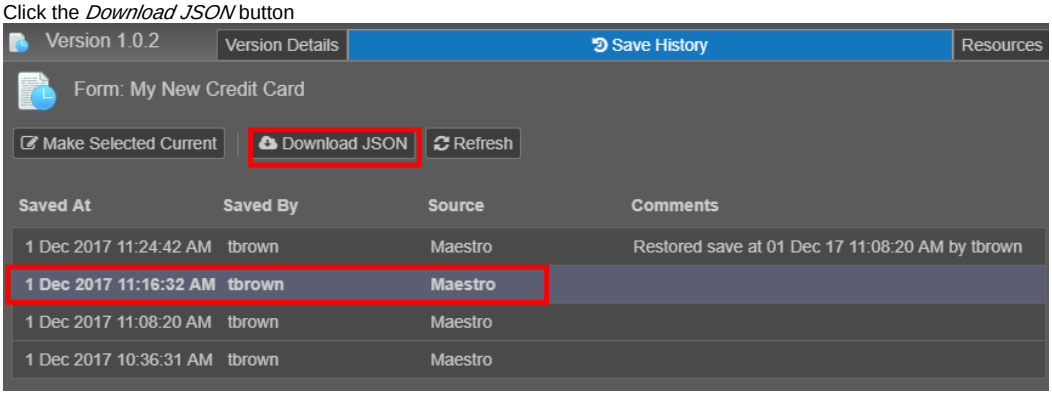


> Remember - Forms, Templates and Components all have the Save History available.

Compare Previous Save Instances

If you have multiple instances in the save history list you can download the JSON for each instance and, using external tools, compare the differences.

1. From the *Save History* list select a previous saved instance
2. Click the *Download JSON* button



3. Select another instance from the *Save History* list
4. Click the *Download JSON* button.
   Using an external tool, you can compare the differences between the two downloaded JSON files.

Import and Export

Unknown macro: 'redirect'

Page Contents:

Related Pages:

- Change Template
- Could not set property 'formDataEncryptFlag' on entity TemplateVersion
- Form Operations
- Import and Export
- Rendering a newly imported Form generates an IllegalArgumentException error
- Save History

Overview

In Maestro, you can import and export forms, components, templates, libraries and projects. When importing and exporting items, make sure that you are importing items into the correct location. For example, if you export a template, you must import the template into the template folder of the project (you cannot import a template into the forms folder).

Export an Asset (form, component, template, library, or project)

To export an asset:

1. Navigate to the project where the asset is stored
2. Select the asset folder within the project
3. Select the asset
4. Click the Export button

This will download the asset in a zipped folder to your computer. You will use the downloaded folder to import the asset into Maestro. You can import the asset into the original project which will allow you to create a copy of the asset. If you want to move the asset, you will import it into a different project.

> The process for downloading assets will depend on the browser you are using.

Import an Asset (form, component, template, library, or project)

Copy Asset

1. Select the asset folder within the project where the asset is currently stored
2. Click the Import button
3. Navigate to the downloaded asset zipped folder, select the folder, then click Open
   If an asset with the same name already exists you will see a number added to the end of the asset name. You can use the Details tab to edit the name.

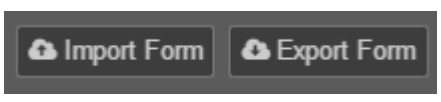
Move Asset

1. Select the asset folder within the project where the asset is currently stored
2. Click the Import button
3. Navigate to the downloaded asset zipped folder, select the folder, then click Open

Import and Export a Form

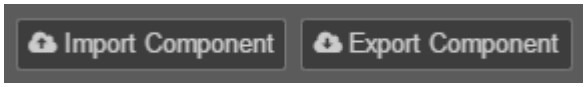
When a form is created it is stored in a specific project. There are times when you may need to move that form to a different project or copy the form to create a new similar form. To move or copy the form you will use the Import and Export buttons as described above. Be aware that when you move a form the resources used within that form will remain in their original project. If you move the form on its own you may notice the appearance does not match the original and that there are resources missing. This would include the template used to create the form, any shared components and styles. To ensure the form displays as originally designed you will need to move the assets used to the new project along with the form. This can be done by moving the individual assets to the relevant project. Another option is to move the library, which will contain the published assets and resources. You can move the library to the new project or to the organization library where all projects will have access to the published assets and resources.

It is recommended that you speak to the Template Designer before importing and exporting templates, styles and libraries.



Import and Export a Component

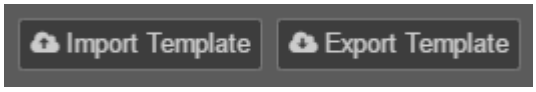
When a component is created it is stored in a specific project. You can move or copy the component to other projects if needed. When you move or copy a component please be aware that the component will not be available within the project until it is published to a library that the form as access to view. If you have imported the entire project or the library, the published version will already be available, however if you import the component on its own you will need to publish it before it is available to forms within the project. See Create New Shared Components for more information.



Import and Export a Template

When a template is created it is stored in a specific project. You can move or copy the template to other projects if needed. When you move or copy the template please be aware that it will not be available to forms within the project until it is published to a library that the form as access to view. If you have imported the entire project or the library, the published version will already be available, however if you import the template on its own you will need to publish it before it is available to forms within the project.

It is recommended that you speak to the Template Designer before importing and exporting templates as they are responsible for the management of templates.



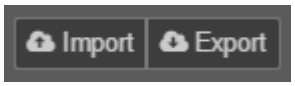
Import and Export a Library

Libraries contain published assets and resources such as templates, components, styles and images. If you export a library, you are exporting all of the published assets and resources within that library.

Please be aware that the resources within the library imported are the published versions. You are not able to edit published versions. In order to edit the resources you will need to have access to the original template, style or component, which are stored in the corresponding folders within a project.

If a library contains assets and resources that need be used across multiple projects you can move the library to the Organization library folder. Any assets and resources in the organization library will be available to all projects within the organization. You will need to follow the same export and import process, however you will need to return to the project library folder and delete the original library from the project. Since all projects have access to organization libraries there is no need to keep multiple versions of the same library.

It is recommended that you speak to the Template Designer before importing and exporting libraries as they are responsible for the management of the templates and styles within those libraries.

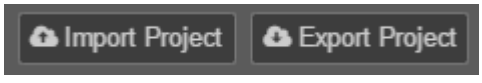


Import and Export a Project

When you export a project you export all of the assets and resources within that project. This includes the original templates, components, and styles along with the published versions.

It is unlikely that you would need to import a project into the same organization as the original, so this option is mostly used if you need to move or copy the project to a different organization. You follow the same process for exporting and importing projects as we have done with the other assets.

It is recommended that you speak to the Template Designer before importing and exporting projects as they are responsible for the management of the templates and styles within those projects.

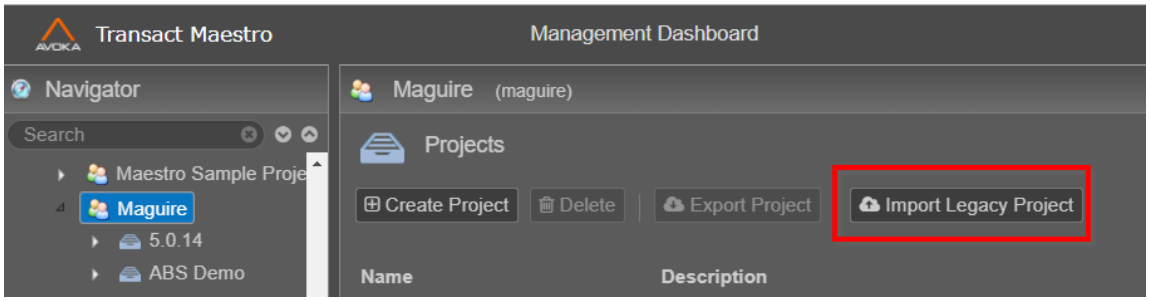


Import Legacy Project

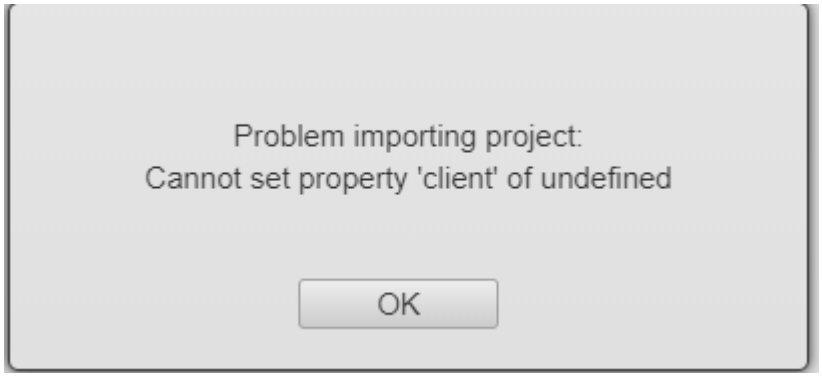
Maestro offers an additional import option for projects created and exported from legacy (previous) versions of Maestro (a project is considered Legacy if it was created and exported from any version of Maestro prior to v17.10). This option works just like the Import Project option, but is specifically for legacy versions of Maestro, and is only available to Maestro users assigned the Maestro Site Administrator Role. If you need to import a legacy project, speak to your Maestro account manager.

Any v5.x projects exported from a v17.10 environment will use the normal import button. The Import Legacy Project button is only for projects exported from a v5.x and earlier release.

The screenshot below highlights the *Import Legacy Project* button.



If you try to import a legacy project using the regular *Import Project* button, you will receive the following error.



Configuring a Maestro Form

Unknown macro: 'redirect'

The content within this section covers information relating to form configuration in Maestro.

Below you will find a list of topics within this section.

- Working with Components
- Form Behavior
- Styling
- Migrating to Transact Manager
- Working with Transact Insights
- Domain Models

Working with Components

Unknown macro: 'redirect'

The content within this section covers information relating to components in Maestro.

Below you will find a list of topics within this section.

- Add Components
- Change Component Label
- Repeating Components
- Creating Maestro Native Components
- Structure and Layout
- Shared Components Section

Add Components

Unknown macro: 'redirect'

Page Contents:

Related Pages:

- Closing the Form
- Customizing the Maestro Editor
- Form Preview
- Integration panel
- Model Panel
- Palette Panel
- Properties pane
- Styles pane
- View Panel

Add Components to a Form

Fundamental to form design is the process of adding components to capture the data required to satisfy a Transaction Experience. The term component is the general name for anything that can be dragged from the Maestro Palette into a form. Components can be anything from an image, to a single data entry field, to a large complex re-usable block. Components come in a variety of control types with different properties and behaviors. The available components are displayed in the Maestro Palette Panel. For more information on this panel, refer to the Palette Panel section of the documentation.

Choosing a component is done in the Palette, and placing it onto the form is done by dragging and dropping the component onto the form or dragging and dropping the component into the hierarchy of the View Panel.

Components are organized within category folders. You can browse each category to find the component you want to use on your form, or you can use the search box at the top of the panel.

At the top of the component list, you will find the favorites category. This category displays the most commonly used components. You can add or remove components from the favorite list based on your own personal preferences. The changes you make to the Favorites list will be stored in your Maestro account, not within the form. The components you add or remove to favorites will be displayed each time you open a form in Maestro.

There are two options available to add a component to a form.

Option One

1. Choose a component from the Palette and drag it onto the form
2. Drag the component onto or near where you want it to appear on the form, observing the insertion bar shape and orientation
3. Drop the component when the insertion bar indicates the desired location

Option Two

1. Select the component immediately above where you want the component to be inserted. You can do this either in the Editor, or in the View Panel.
2. Double click on a component in the Palette. This will insert the component directly below the existing component. You can also right-click the component from the Palette and select Add Below.

This option can be a more effective way of adding components, as it is less effort than dragging and dropping.

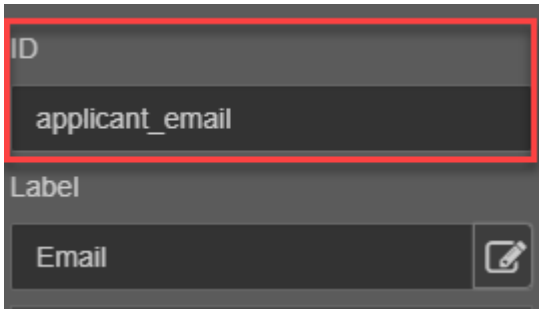
If you have multiple components to add, simply keep double clicking - the component will be added sequentially.

Once you have added a component, you will generally want to update the *ID Name* and confirm the *Label* from the Properties panel.

The ID property of the component will always be highlighted automatically immediately after adding a component, so you can just start typing. The Label is based on the component ID, but can be changed as needed.

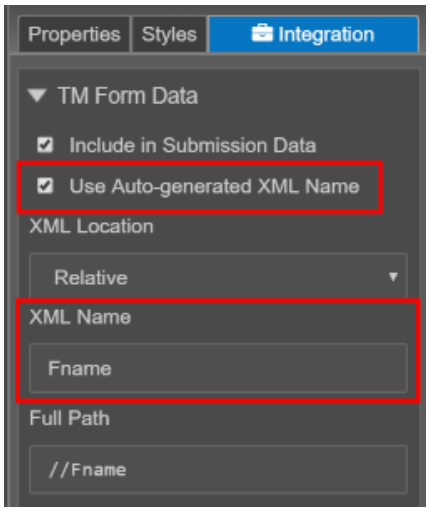
Component ID

When you add a new component to a form, the ID field of the Properties panel will be highlighted automatically. A component ID is a unique identifier for a component. More information on component IDs is available from the Understanding the Component ID section of the documentation.



The XML Name and Label will be automatically generated from the component ID.

Once the initial ID is provided, the Label will not be updated should the ID change. The XML Name and Path will be updated based on the ID, unless the *Use Auto-generated XML Name* checkbox is deselected. If the checkbox is deselected, the XML Name will not be updated should the ID change.



For more information on the component ID, please refer to the Understanding the Component ID section of the documentation.

Determining where the component will be inserted

Insertion using the Editor

The insertion bar indicates a wide variety of possible insert locations by using different positions, sizes and shapes to represent the insertion point. The insertion point is represented by a selection area, which indicates the parent layout of the component, and an insertion bar which indicates its position to its siblings.

The insertion bar will appear either as a vertical or horizontal bar. A horizontal insertion bar will insert the component vertically between two other components, and a vertical bar will insert a component horizontally between two components.

If the insertion point is at the beginning or end of the list of siblings, then the component will be inserted regardless of the insertion points horizontal or vertical orientation. If the the insertion point is vertical, and there is space for the new component to occupy within the layout, the new component will appear on the same line as an previously placed components. If there isn't sufficient space to accommodate the component, it will be placed on the following line.

Insertion using the hierarchy

Similar to the Editor, an insertion point is displayed when you drag a component from the Palette panel to the View Panel.

In the View Panel, the insertion point appears as a triangle *above, below* or *on* the desired component if it's a container. You can change the insertion point while dragging by moving up and down the hierarchy. If the drag target is a component, the insertion point will appear *above* indicating the dragged component will be inserted *before* the drag target, and the same is true for *below* and *after*.

If the drag target is a container of other components, you can insert the component into the container by placing the insertion point *on* the drag target. Releasing the dragged component will add the component to end of the children. If you drag a component onto a container that is collapsed, the container will expand, displaying any children it contains, allowing you to navigate deeper into the structure without releasing the dragged component.

> You can change any mistakes or misplaced components by simply dragging the component to the correct location. You can also use the Undo button, moving back the History or simply deleting the component and trying again.

Shared Components

Another option for components is to use a shared component. A shared component is a custom built component within the business. A shared component can be added to your form just like any other component. The difference with a shared component is that it can be updated outside of the form. When it is updated all forms using that shared component will be updated. This is a great way to re-use common components, but also keep them consistent across multiple forms.

Shared components may be developed within the Components folder in the Maestro Dashboard or within a form.

Shared components can be published to any category within the Palette panel. The default location, which can be changed, is the Custom Components folder.

For more information on shared components view Create New Shared Components.

Change Component Label

Unknown macro: 'redirect'

Labels, which are found on most form items, give the user an idea of what information is required in that component. They are typically located above the component in order to save screen real estate.

There are two types of labels - Static and Dynamic.

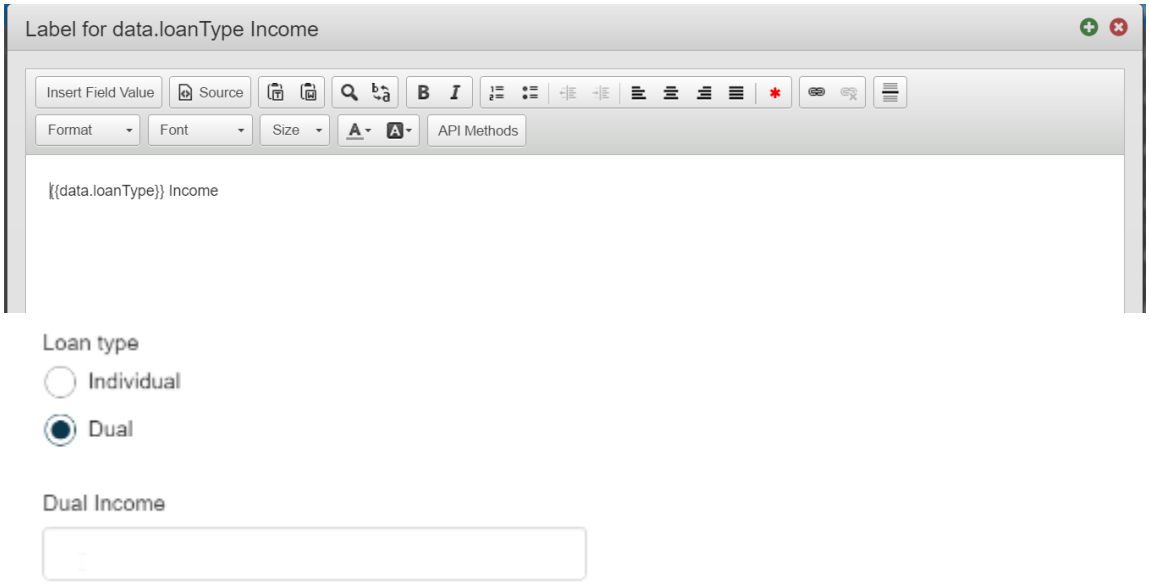
Static

To change the label, select the component on the form or in the structure pane and change the label in the Properties pane (first property in the pane). Use the edit button to include a longer label or to access other formatting options.

Dynamic

If you select the edit button, you have access to the text editor where you can reference other component values to make the label dynamic.

In the example below, the label of the text field is being determined by the selected option from the radio button group. If the user selects Individual as the Loan Type, the text field label will be Individual Income. If the user selects Dual, the label will display Dual Income.



Related Pages:

- Add Components to a Form
- Add Shared Component to Form
- Change Component ID
- Change Component Label
- Create New Shared Components
- Creating native components for Maestro (Eclipse/ IntelliJ Developers)
- Delete a Component
- Enforcing required configuration properties in custom native components
- How to use Data-Driven Components in Maestro
- Repeating Components

Repeating Components

Unknown macro: 'redirect'

Page Contents:

Related Pages:

- Add Components to a Form
- Add Shared Component to Form
- Change Component ID
- Change Component Label
- Create New Shared Components
- Creating native components for Maestro (Eclipse/ IntelliJ Developers)
- Delete a Component
- Enforcing required configuration properties in custom native components
- How to use Data-Driven Components in Maestro
- Repeating Components

Overview

Repeating components allow a variable number of instances of a component to be added to the form depending on the individual form user's requirements as opposed to having a fixed number of the same component which may not be sufficient. Repeating components are found in the Repeats folder in the Maestro Palette.

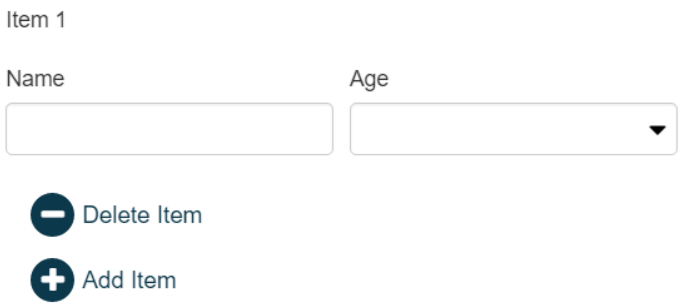
Repeating Block Template

The Repeating Block Template is the easiest way for the form builder to add repeating content to a form as it automatically includes all of the infrastructure required without the actual items (i.e. a label on each repeat and an add and delete button).

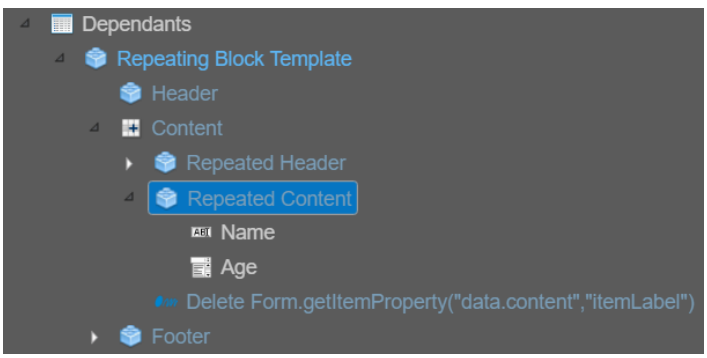
For example a form requires details of the form user's dependents.

1. Add the Repeating Block Template to your form

2. Add components to the Repeating Block Template



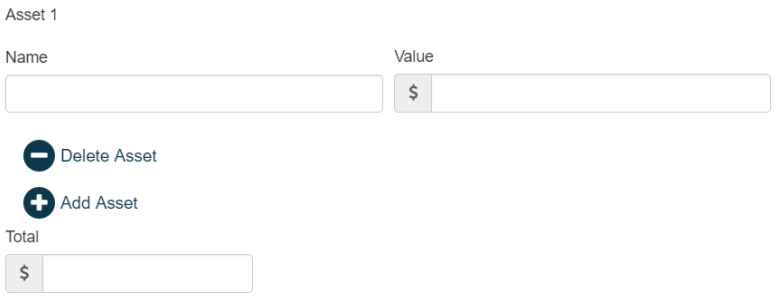
Note: The Repeating Block Template is made up of a number of different components. It is important to customize the correct part of the repeating block template depending on what you are trying to achieve. For example, when items are added to the Repeating Block Template, they are added to the Repeated Content block. However, if you would like to change the Repeat Item Label or the Min or Max items in the repeat, you will need to select the Content block.



Calculate a Repeating Component

You can create a calculation based on one or more specific components within a repeating block.

For example the form requires details on the user's assets. You want to total the value of all assets entered.



1. Add a component outside of the repeating block template.
2. Select the component and click the *Create Rule* button in the Properties pane.
3. Select *Script...* under Calculation.
4. Navigate to the component within the *Repeating Block Template* **>** Content **>** *Repeated Content* that contains the component you want a total of.
5. Right-click the component and select Get sum of repeating data
6. Click *Save*.

The calculation rule is added to the Properties panel. This rule will add the entered or calculated value for each repeating block.

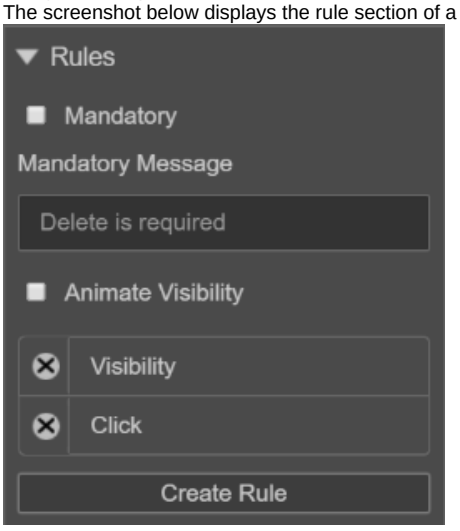
Hide Delete Button for a Single Item in the Repeating Block

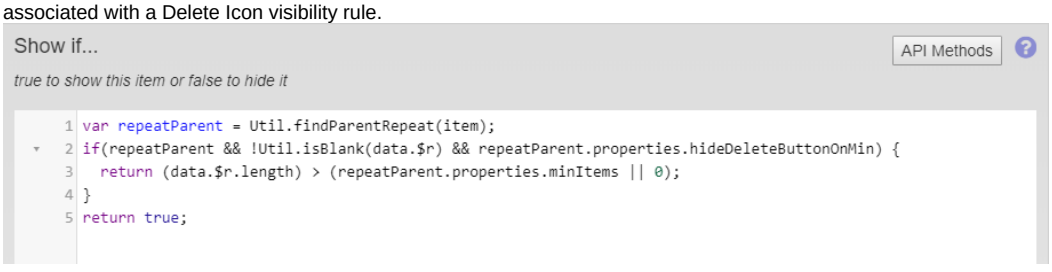
Each time a new item in the repeat is created, an *Add Item* button and a *Delete Item* button will be created for the item.These buttons are used to allow the form user to continue to add items until they have entered all items that are required of them. However, there may be times where a Form Builder will not want to give the form user the option to delete an item until a certain validation rule has passed. The Form Builder may also only want the delete button to appear on certain repeat items.

An example of how to hide a delete button for a single item in a Repeating Block is described below.

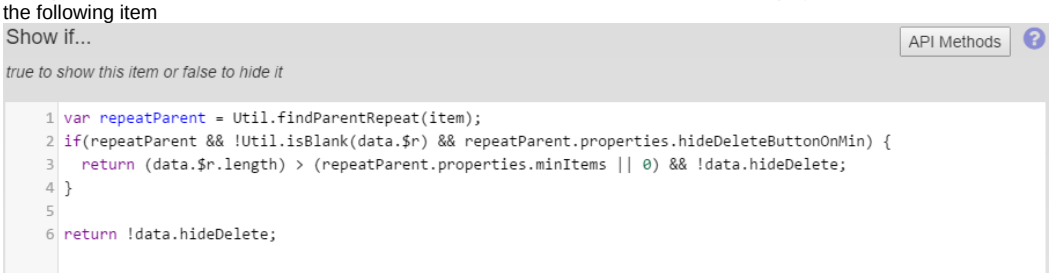
To hide a delete button for a single item in a Repeating Block:

1. Select the Delete Item button from the Repeating Block (Delete Item buttons in the Repeating Block Component have their own visibility rules that can be modified to create expanded validation and at times added functionality).

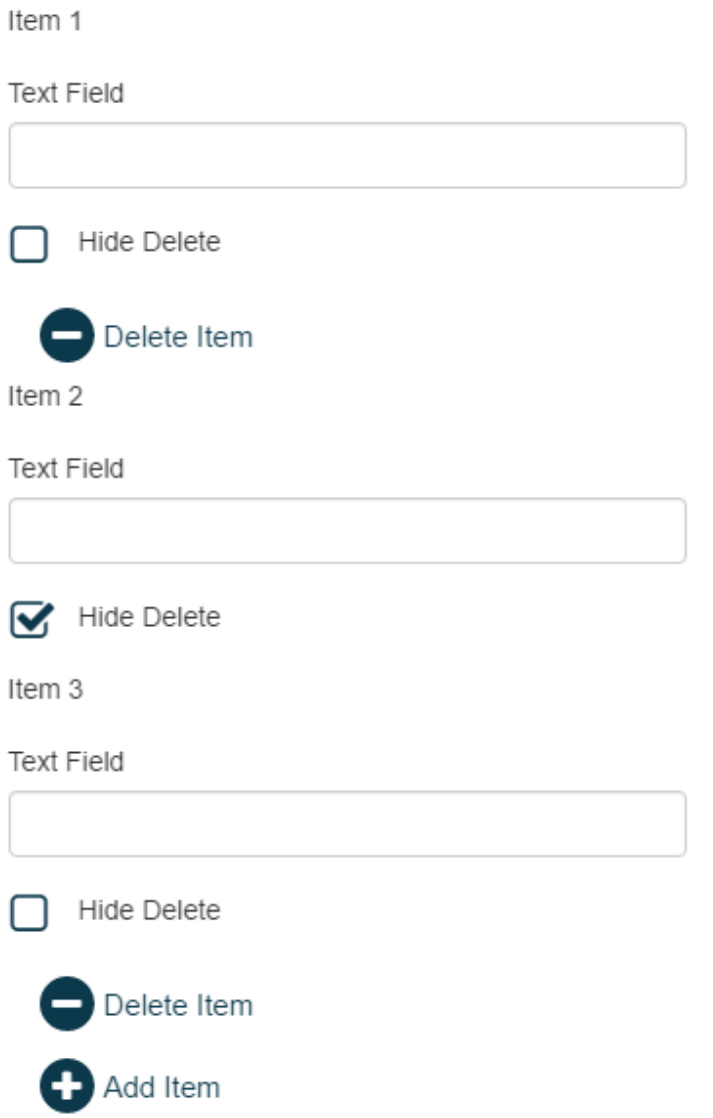
2. Navigate to *Properties* and scroll down to Rules
   The screenshot below displays the rule section of a default Delete item button.
   

3. Double click the Visibility Rule
   This rule is created by default for all Delete Item buttons in a Repeating Block.
   To hide the delete button, we need to add additional logic to this visibility rule. Double-clicking the visibility rule will open the rule editor dialog. The screenshot below displays the default logic associated with a Delete Icon visibility rule.
   

4. Edit the visibility rule.
   Depending on your needs, you can edit the visibility rule in many different ways. In this example, we will use a checkbox to decide which instances of the repeat block will display a delete icon button.
   The screenshot below displays an example of how you may wish to edit the visibility rule. In this example, when the Hide Delete checkbox is selected, the Delete Item button will not display for the following item



5. Click *Save*
   The screenshot below displays an example of when this visibility rule is applied to a Repeating Block. In this example, the Hide Delete checkbox has been selected after the first Text Field has displayed and the result of this selection is that the Delete Item button has been removed from the second item.



Creating Maestro Native Components

Unknown macro: 'redirect'

Welcome to the world of Maestro components. This article explains what Maestro components are, and shows how you can create new ones using the Maestro Native Component Development Toolkit.

At the bottom of this article, you will find detailed instructions to get you started with the Maestro Native Component Development Toolkit.

What is a Maestro Component?

Maestro components are the form building blocks you see in the Palette of the Maestro form design tool. Components are usually, but not always, visual, meaning you will see them on your published form. You drag and drop components onto your form as part of the form design process.

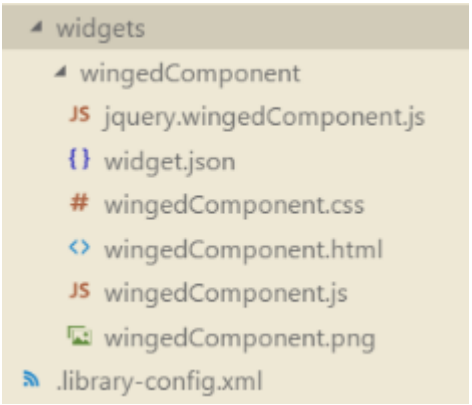
Maestro supports three types of components:

- Core components - these ship with Maestro out of the box - they are actually native components
- Custom components - you can create these on-the-fly within the Maestro form design tool by combining existing components, writing rules, bundling and sharing as a resource. They are not native components, so there are limitations on what you can achieve in a custom component
- Native components - you or somebody else can build and import these into your Maestro Libraries

What is a Native Component?

A native component is a collection of files that are packaged together into a specific folder structure in a Zip file. The Zip file can then be imported into a Project or Organization Library in Maestro. Once imported, you will see all the new components in the Maestro Palette and you can use them like any other type of component.

Specifically, the Zip file contains this folder structure:



- .library-config.xml - must be exactly this name, and be in the root of the Zip file - this identifies the Zip file as a Maestro Library and provides metadata such as dates and author
- widgets - must be exactly this name, and must be in the root of the Zip file
- <component sub-folder> - ('wingedComponent' in the screenshot) - each component must live in its own subfolder under the top-level 'widgets' folder
    - widget.json - must be exactly this name, and defines the component, including:
        - Naming and default sizing details
        - Editable properties that will appear in the Maestro Property tab
        - Names of related CSS, HTML, JavaScript and icon files
    - <filename>.png - a 24x24 pixel, 32-bit color image with a transparency** layer, used as the icon in the Maestro design tool
    - <filename>.html - the HTML for the component
    - <filename>.css - the CSS for the component
    - <filename>.js - the custom JavaScript for the component
    - <other files>.js - any other JavaScript files the component depends on, such as JQuery plugins

**NOTE: We recommend using Paint.Net from https://www.getpaint.net/index.html to create or edit your icons since it handles the transparency layer very well.

Native Component development toolkit

Avoka has produced the Maestro Native Component Development Toolkit to simplify the creation of custom native components through the use of templates, and automated tasks.

The development toolkit is based on the 'gulp' automation tool... http://gulpjs.com/. This tool executes custom tasks in a JavaScript file called 'gulpfile.js'

These tasks are defined in the toolkit's gulpfile.js:

1. 'new' - generates a new component in the 'src' folder, based on files in the template folders
2. 'zip' - creates a Maestro Library in the dist folder by bundling all files in the src folder into a Zip file
3. 'watch' - helps automate the code generation and packaging process
    a. continuously monitors all files in the src folder structure and runs the 'zip' task to generate the Maestro Library if it detects changes
    b. continuously monitors new-component.properties and runs the 'new' task to generate a new component if it detects changes
4. 'redist' - creates a new toolkit distribution file by packaging all of the templates into a new toolkit. This is useful if you customize your templates and want to share them with others

You can run any of these tasks in the toolkit installation folder by typing 'gulp <task>'. E.g. 'gulp new', 'gulp zip' or 'gulp watch'.

The 'new' task can optionally take an additional parameter to identify which template you want as the model for your new component. Usage is 'gulp new -templateName', where 'templateName' is the name of a sub-folder in the template folder. You must prefix the name with '-'. If you don't supply this optional parameter, '-default' is assumed, which will model your new component on the component template files in template/default.

A word of caution with the 'new' task... it will regenerate your component's source files from the files in the template folder, overwriting any changes you might have made to files in the 'src/<componentName>' folder. To avoid this happening by accident, change the value of the 'component_filename' property in the 'new-component.properties' file immediately after you generate a new component. That way, when you run 'gulp new', the new value in 'component_filename' will be used and won't overwrite your original code.

Installation Instructions

1. Install NodeJs from https://nodejs.org/en/download/
2. Create a new folder somewhere on your PC and unzip avoka-native-component-tools.zip into it
3. Open a NodeJs command prompt window and change directory to the folder you just created
4. Type 'npm install' and wait for the npm installer to finish - this will install the toolkit and might take a minute or so

Super-quick start

If you just want to see the toolkit in action, you can follow these two steps to generate a new Maestro Library with the default example component contained in the toolkit.

1. Type 'gulp new' - this will create a new component in a 'src' folder, based on the values in the two .properties files
2. Type 'gulp zip' - this will package your component into a Maestro Library - the output will be dist/maestro-lib-<filename>.zip

Quick start

To create a new component with customized filenames and other metadata, such as author:

1. Edit library.properties and new-component.properties to set your preferences
2. Type 'gulp new' - this will generate a new component in the 'src' folder
3. Edit your new component's files in the src/widgets/<filename> folder
4. Type 'gulp zip' to package all the components in the 'src' folder into a Maestro Library

Automating Library creation with Visual Studio Code

The gulp watch task is very useful to automate library creation while you are developing your components. It's extra useful if you use it in conjunction with an IDE.

Here are some simple instructions to get going with Visual Studio Code:

1. Install the IDE from https://code.visualstudio.com/download.
2. Start the IDE and select File/Open Folder - then open the folder where you installed the toolkit
3. Press Ctrl-` to open its Integrated Terminal - or select View/Integrated Terminal from the menu
4. Type 'gulp watch' into the Integrated Terminal window
5. Edit your component files in the src folder - when you save them with Ctrl-S or File/Save from the menu, the watch task will build your Maestro Library instantly
6. Import the generated library from the toolkit dist folder (in Maestro) whenever the library is ready to test your changes

If you don't want to continuously have the watch task running, just type 'gulp zip' into the Integrated Terminal window when you want to generate your Maestro Library.

Please see the Creating native components for Maestro (Eclipse/ IntelliJ Developers) article for additional information.

Structure and Layout

Unknown macro: 'redirect'

The content within this section covers information relating to structure and layout in Maestro.

Below you will find a list of topics within this section.

- Pages and Sections
- Component Layout

Pages and Sections

Unknown macro: 'redirect'

Related Pages:

- Component Layout
- Pages and Sections

Page Contents:

Overview

Many of the forms built in Maestro are long with many components. This can be daunting for end-users. Maestro forms use Pages to improve the end user experience by guiding form builders to group fields into logical areas. Adding a page to a form is repeated frequently as you add new areas to the form to provide users with concise data collection tasks.

If you have a navigator in your template (most templates do) then adding a page will automatically add a new item in the navigator menu.

If you have a larger form, it is strongly recommended that you create your pages first, and then start adding individual components. It is possible to move fields into pages later, but it is more productive to create pages first.

With a form in the Design mode, follow these steps to add a new wizard page to an existing form:

1. Ensure that you have the panels set up correctly so that you can see both the **Palette** and **View** panels simultaneously.
2. Locate the **Page** component in the Navigation group.
3. Drag the **Page** component onto the form hierarchy.
4. Drop the **Page** component at the desired location, between existing Pages or at the end.

Once the Page is in place, you can proceed with adding new components to the Page to complete your design.

Alternatively, you can:

1. Select the **View** panel.
2. Right click on the **Contents** item.
3. Select **Add Page** from the menu.
4. Drag and drop the newly created page to the correct location in the **View** panel.

Two Level Navigation

For very large forms, you can create two level navigation. This will add a sub-menu to the form.

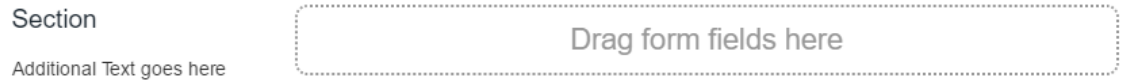
To do this, create a **Page Group**, and add **Pages** within them. If you are using Page Groups, you should generally ensure that all pages are located within a group, otherwise it may be confusing for end-users. Page Groups will appear on the first level navigator, and Pages within the Page Groups will appear on the second level navigator.

Follow the steps below to create two level navigation.

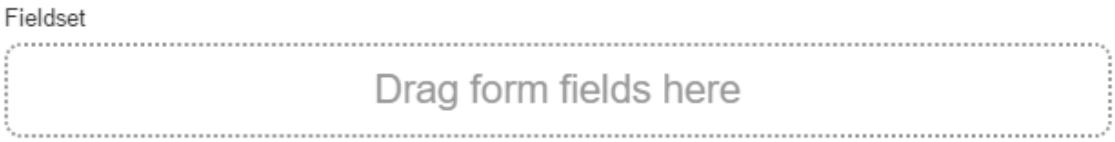
1. Find the Page Group component and add it to Content
2. Drag new or existing pages to the Page Group

Using Sections and FieldSets

If you want to further break up the content in a page, you can use Sections. A section includes a heading and optional explanatory text.



You can further break up your fields by grouping them into Fieldsets. A Fieldset is the recommended approach of grouping a number of related fields together.



For Pages to operate correctly, they must be inserted under the Content container in the hierarchy. Dropping the Page component onto the form design may result in incorrect placement.

Component Layout

Unknown macro: 'redirect'

Related Pages:

- Component Layout
- Pages and Sections

Page Contents:

Overview

A layout is a framework for the management of components that are displayed on a form. The layout is responsible for ensuring components appear according to the form design rules in the device screen size, or browser window.

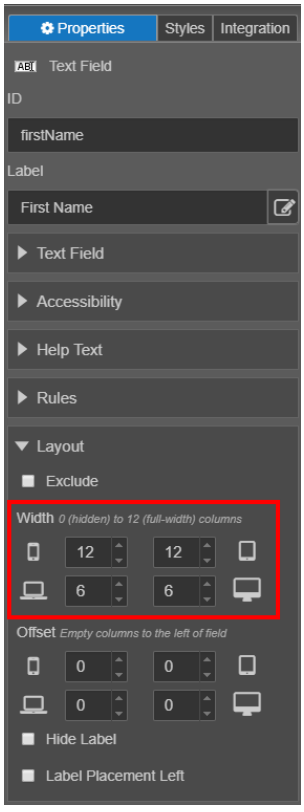
The layout determines a number of aspects of the fields on the form.

- The width of each field.
- The placement of fields relative to each other. In other words, does a field appear to the side or below of the field that precedes it.
- The adjustment of both the widget and placement of fields when the overall width of the screen changes.

Layout Concepts

Laying out fields in a form uses a very simple yet powerful technique.

1. The form itself defines three "breakpoints". This effectively creates 4 different standard screen sizes. Since modern devices come in so many sizes, and because devices can also be rotated into either portrait or landscape mode, these standard sizes are only roughly indicative of actual devices. However, the four sizes are roughly: SmartPhone, Tablet, Small Desktop Screen and Large Desktop Screen.
2. Each container in the form is broken down into a 12 column grid. Twelve is a useful number, because 12 can be divided in half, thirds, quarters, and in other ways, making it very flexible but simple. (It's no coincidence that many historical measurement systems use twelve as a base.)
3. Each field can be configured to use up any number of columns, from 1 to 12. This determines its width.
4. You can continue placing additional fields to the right of any other fields. However, as soon as the total column count of all the fields on a row exceeds 12, that field and any others will be moved to the next line.



In the displayed screenshot, a text field takes up half the available space on larger devices. However on smaller devices, the user is likely have issues reading the field labels or being able to enter data into the small field areas. Hence the default size on smaller devices is to take up all the space on a row. This is also why the default location of the field's label is always on the top of the field – this ensures that the label and field don't spread out too much on narrower devices.

The breakpoints for screen sizes are determined at the template level. These are set to appropriate defaults which usually cover the standard screen sizes that are available in the market today, but your template designer may change them if desired.

In the Maguire template, the breakpoint between a small screen and medium is 400 pixels, between medium and large is 560px and large to full screen is 780px. This means that anything screen size under 400 pixels is considered a small screen, screen sizes between 400px and 560 is medium, 560px-780px is large and anything over 780 is an extra large screen.

The screen size width will always equal 12 columns, no matter the actual pixel width. So 400px = 12 columns, 560px = 12 columns and so on. The only thing that changes is the width of the columns. Dividing 12 columns across 560 pixels would mean that the columns are wider than when you divide 12 columns across 400 pixels.

Basically each row in your form contains 12 columns. When designing your form you can decide how the 12 columns are divided. To get two equal text fields on one line you would need to change both text fields to 6 columns - 6 + 6 = 12.

If you didn't want them to be equal you could set one to 7 columns and the other to 5 columns – 7 + 5 = 12.

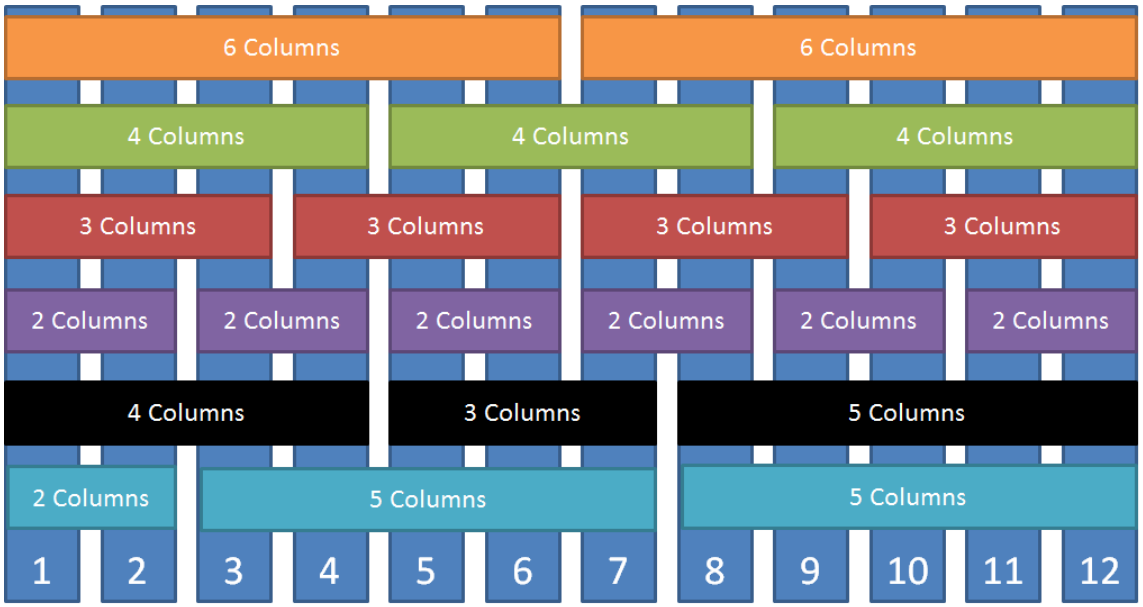
Another possibility with the 12 column grid is to use multiple blocks on a row to give more flexibility.

In the screenshot below, there are two blocks on a row, each block containing components. The blocks are both set to 6 columns wide. The components within the block have their own 12 column grid which allows them to be arranged within the available space of the block. In block one there are two text fields both set to 6 columns wide. In the second block there are three text fields, all three set to 4 columns wide.

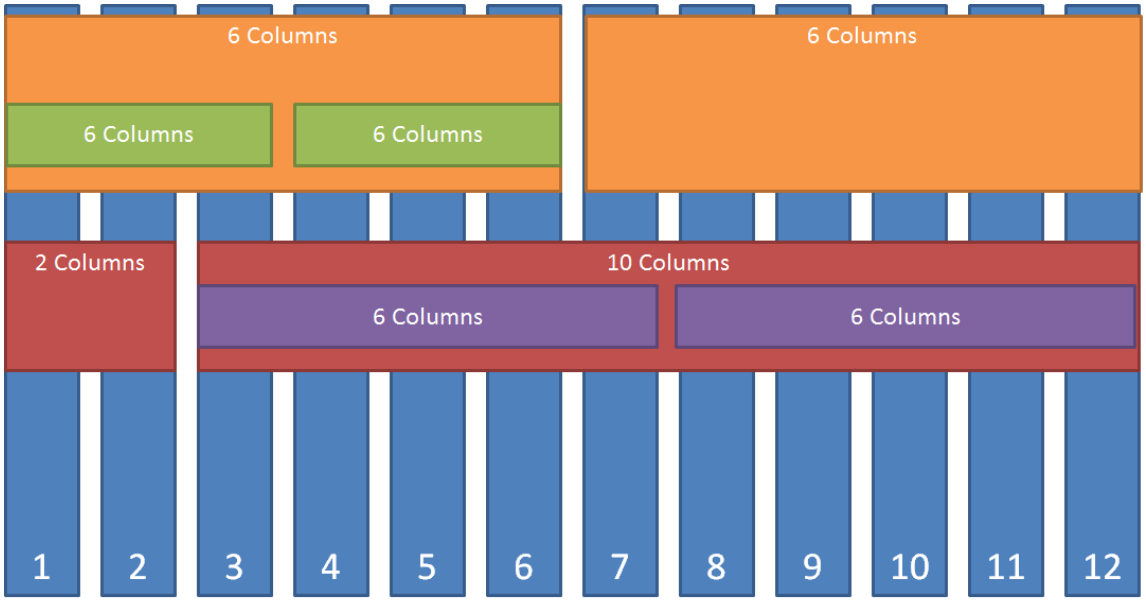
Layout Samples

The following images display various options and combinations available when laying out components on your forms.

`Image 1`



`Image 2`



Shared Components Section

Unknown macro: 'redirect'

The content within this section covers information relating to shared components in Maestro.

Below you will find a list of topics covered in this section of the documentation.

- Getting Started with Shared Components
- Create New Shared Components
- Add Shared Component to Form
- Edit Existing Shared Component
- Publish Shared Components
- Override Shared Components in a Form
- Component Options

Getting Started with Shared Components

Unknown macro: 'redirect'

Overview

Maestro encourages the re-use of form-building assets wherever possible.

Organizations usually build shared components for a number of reasons, including:

- Consistency - to ensure that a component looks and functions the same wherever it is used.
- Encapsulation of complexity - to allow complex components to just be dropped in and used, without having to understand the internal implementation. This is particularly useful when a component makes dynamic data calls to back-end services.
- Productivity - allow forms to be built faster.
- Enable change - if a component needs to be changed, all the forms that use it will automatically inherit the changes.

Shared components are custom built blocks within the business that can be re-used throughout the form and can be made available to other forms in the organization by publishing them to the organization libraries or publishing them to the project libraries.

Shared components can be created by form builders for re-use and consistency among forms, and can consist of a single item or a collection of items wrapped in a block. Shared components can be published to a library and made available to other form builders.

The creation of a shared component can be initiated from within a form using existing items or blocks, or by using the component editor from the Components folder in the Management Dashboard. See the Create New Shared Components section for more information.

Related Pages:

- Add Shared Component to Form
- Create New Shared Components
- Edit Existing Shared Component
- Getting Started with Shared Components
- Publish Shared Components

Create New Shared Components

Unknown macro: 'redirect'

Pages Contents:

Related Pages:

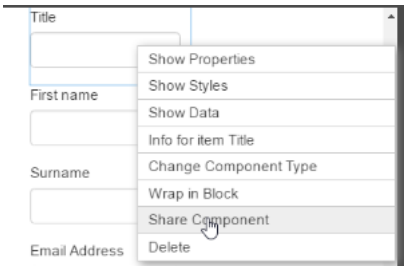
- Add Shared Component to Form
- Create New Shared Components
- Edit Existing Shared Component
- Getting Started with Shared Components
- Publish Shared Components

The creation of a shared component can be initiated within a form using existing items or blocks, or by using the component editor from the Components folder in the Maestro Management Dashboard.

Create From Existing Form Items

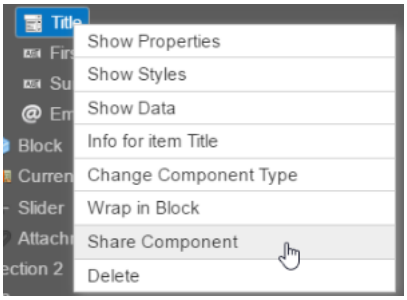
To create a shared component from existing form items, follow the steps below.

1. Build the component in the form.
2. Right click on the component in the form (as shown in the below screenshot)



OR

Access the View Panel and select the Share Component option (as shown in the below screenshot).



If you are creating a shared component that contains more than one component, you will first need to wrap all components in a block. To do this, select all of the components, and then right-click a selected component from the View Panel. From the menu that displays, select Wrap in Block.

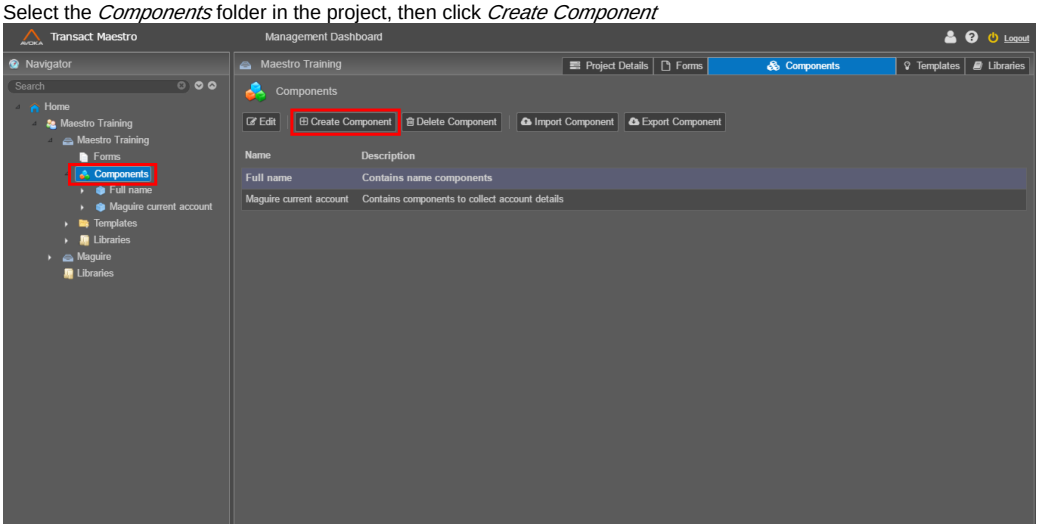
3. Complete the share component dialog:

   a. **Name**: The default name of the new component is the label of the item selected
   b. **Description**: A decription of the shared component (what is it?). Optional though recommended
   c. **Palette Folder**: default is Custom Components. This will be where the component is found in the Palette pane.
   d. **Instant Publish**:
      i. *Publish for immediate use* - the shared component will be immediately available to be used in other forms.
      ii. *Replace item with shared component* - the component used as the basis for the shared component will now be replaced by the shared component (i.e. it will now automatically reflect any changes made to the shared component).
   e. **Publish Library**: Publish to the project library if you want to make this shared component only available to forms in this project or to the Organization library if you want any form in the organization to access this shared component.
   f. **On Save**:
      i. You may edit the shared component in the Component editor *OR*
      ii. View the shared component in the Management Dashboard *OR*
      iii. Close the dialog and return to the form.

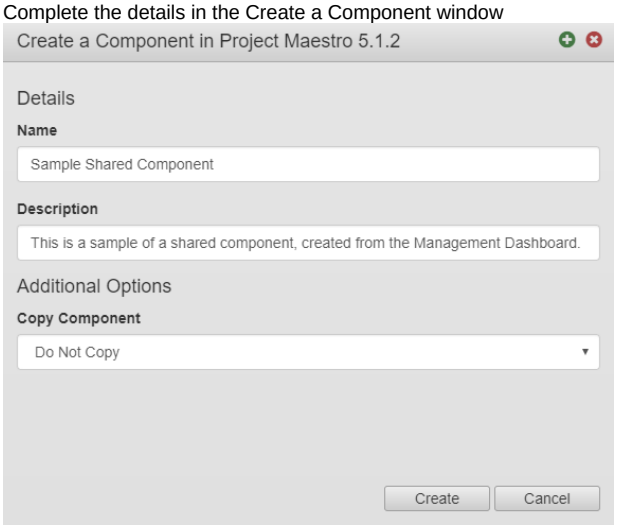
Create From the Management Dashboard

To create a shared component from the Management Dashboard, follow the steps below.

1. Select the *Components* folder in the project, then click *Create Component*



2. Complete the details in the Create a Component window



   a. **Name** - This is the name of the component, and will be used to identify the component in the Components folder.
   b. **Description** - This will display when the Components folder is selected.
   c. **Additional Options**: Copy Component
      i. Do Not Copy - This option allows you to create a brand new (blank) component. This option is recommended for shared components that you are building from scratch.
      ii. Copy and override an existing component - This option allows you to copy an existing component. The new component will replace the existing component in the library, which will update all forms using the existing component. The override will only replace the library resource, not the asset in the Components folder. Before selecting this option, it is important that you consider the usage of the component that you will be overriding.
      iii. Copy and create a new component - This option creates a new component based on an existing component. Selecting this option will create a copy without overriding the existing component.

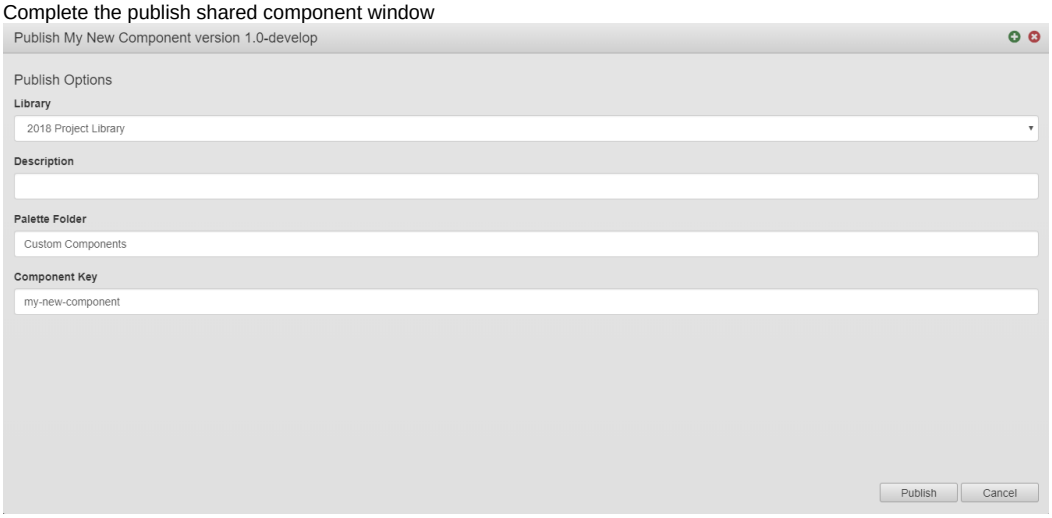
3. Click *Create*
   This will open the new component in the Editor window. Once finished, you will need to publish the component for use in forms. See below for more information.

Publish the Shared Component

To publish a shared component, follow the steps below.

1. Click the *Publish* button

2. Complete the publish shared component window



   a. **Library**: Publish to the project library if you want to make this shared component only available to the forms in this project or to the Organization library if you want any form in the organization to access this shared component.
   b. **Description**: optional though recommended. The description will display in the library.
   c. **Palette Folder**: default is Custom Components. This will be where the component will be found in the Palette pane.
   d. **Component Key:** The Component Key is automatically generated based on the name of the shared component. This key is used to uniquely identify the component. Though the Component Key can be changed, it is recommended that you do not change the key as changing it can lead to issues when the component is deployed to a Maestro form.
3. Click *Publish*

For more information on publishing shared components, please see the Publish Shared Components section of the documentation.

Add Shared Component to Form

Unknown macro: 'redirect'

Add Shared Component to Form

You can add existing shared components to a form as you would any other component. Once the shared component is added to the form, you can change the Component ID. If the ID includes an entity, you can apply the entity to the child components within the shared component.

To apply the entity to the child components, right-click the shared component block. Right-clicking the shared component will display the following menu:

From the menu, click *Apply Entity Path to Children*.

If the shared component is updated to include additional child components, you will need to reapply the entity path to children. To do this, follow the same process as outline above, however, when you receive the warning about child components already having an assigned entity (as shown in the screenshot below).

Selecting *Accept* will ignore the child components with the existing entity (as long as the entity has not changed), but it will apply the entity to the new child components within the shared component block.

Related Pages:

- Add Shared Component to Form
- Change Component ID
- Delete a Component

Edit Existing Shared Component

Unknown macro: 'redirect'

Related Pages:

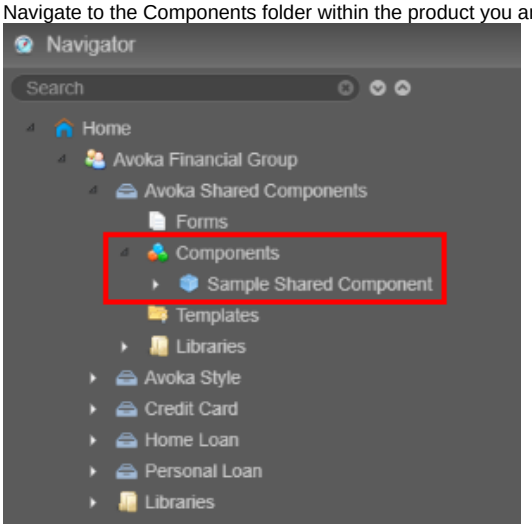
- Add Shared Component to Form
- Create New Shared Components
- Edit Existing Shared Component
- Getting Started with Shared Components
- Publish Shared Components

Edit Existing Shared Style

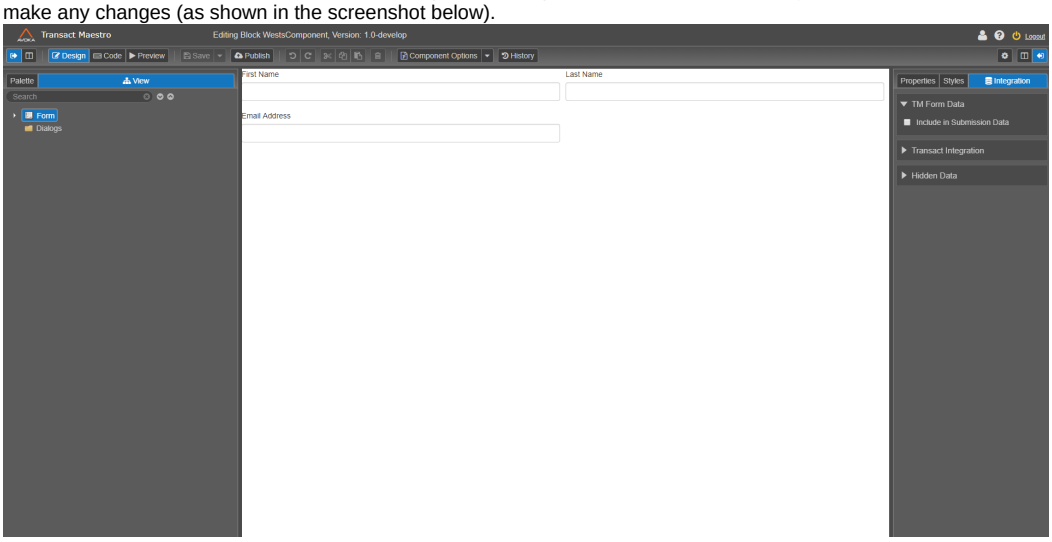
To update the shared style, you need to make changes in the Component editor window. Once the changes have been made, you will need to save and publish the updated shared component.

The steps below document the process of editing a shared style.

1. Navigate to the Components folder within the product you are working on



2. Open the shared component
   You can double-click the shared component from the Navigator or select the shared component and then click the Edit Version button. This will open the component editor window where you can make any changes (as shown in the screenshot below).



Once you have updated the shared component, you can save or publish it.

> Changes you make to the shared component will not be reflected in forms until you publish the shared component.

Publish Shared Components

Unknown macro: 'redirect'

Page Contents:

Related Pages:

- Add Shared Component to Form
- Create New Shared Components
- Edit Existing Shared Component
- Getting Started with Shared Components
- Publish Shared Components

Publish Shared Components from the Editor Window

When you create a shared component from the form editor window you will have the option to *Publish for immediate use*. See the Create From Existing Form Items section on the Create New Shared Components page.

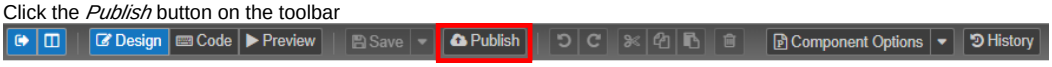
The Publish for immediate use option means that the shared component will be immediately available to be used in this form as well as other forms.

Publish Shared Components from the Component Editor window

When you create a shared component from the component editor window, you need to publish the shared component in order to make it available for use in forms. You will also need to publish the shared component when updates are completed and ready for use in forms.

To publish the shared component, follow the steps below:

1. Click the *Publish* button on the toolbar



2. Complete the publish options

Library: Select the library where the published shared component will be stored. The library used determines which forms have access to the shared component. It is recommended that you view the Best practice for resource management to understand the project and library structure, including where shared components should be published.

Library Version: The version of the library that has been mapped to the selected library.

Description: This is an optional field which allows you to provide a description to the shared component.

Palette Folder: This is the name of the folder where the shared component will be displayed in the Palette panel.

Component Key: The Component Key is automatically generated based on the name of the shared component. This key is used to uniquely identify the component. Though the Component Key can be changed, it is recommended that you do not change the key as changing it can lead to issues when the component is deployed to a Maestro form.

3. Once the publish options are completed, click *Publish*.

Override Shared Components in a Form

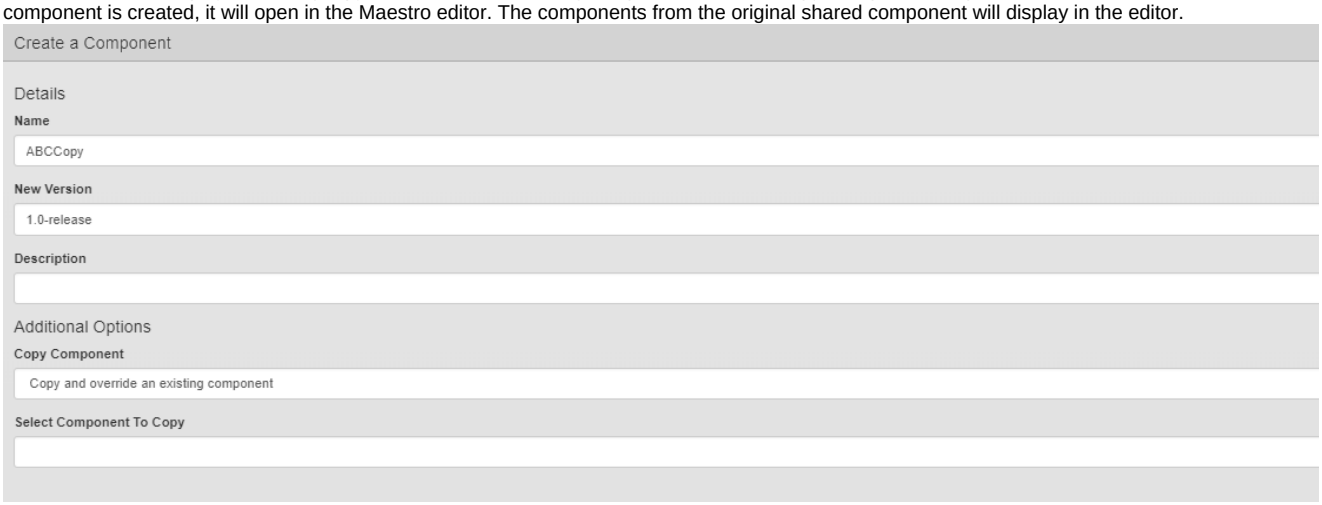
Unknown macro: 'redirect'

Though there are a variety of Maestro components available from the palette, there may be times where the available components do not quite fit the needs of your form and you may need to override a native component with a custom component that you can share across multiple locations in a form. Other reasons for overriding shared components include, wanting a component to fill a specific one-time only purpose, or needing a component that provides a specific functionality that is specific to your company.

To copy and override Maestro components:

1. Create and publish a shared component, for example, a component named ABC.
   You can create the new shared component from the Maestro Dashboard, or directly in a form. More information on both processes can be found in the Create a Shared Component section of the documentation.

2. If needed, add the published shared component (from step 1) to a form
3. From the Maestro Dashboard, create another new shared component.
   In order to use the copy and override option, you must create the shared component from the Maestro Dashboard. When you are creating this new shared component, it is important that you select c*opy and override an existing component* from the Copy Component dropdown list. The screenshot below displays selecting *copy and override an existing component* from the Copy Component dropdown menu. Once the shared component is created, it will open in the Maestro editor. The components from the original shared component will display in the editor.



4. From the *Select Component To Copy* dropdown, select the component that you want to copy and override (e.g. the first shared component, ABC)
   Selecting ABC means that all changes made in the ABCCopy shared component will replace the original ABC shared component.

5. Edit the shared component, then publish.
   In the editor window, make a change to the shared component, then click Publish. Information on publishing the shared component can be found in the Create a Shared Component section of the documentation.
   When you publish the shared component make sure you publish to the same library as the original. If you publish to a different library, it will not fully override the original and the shared component displayed in the form will be determined based on the precedence of the library used in the form.

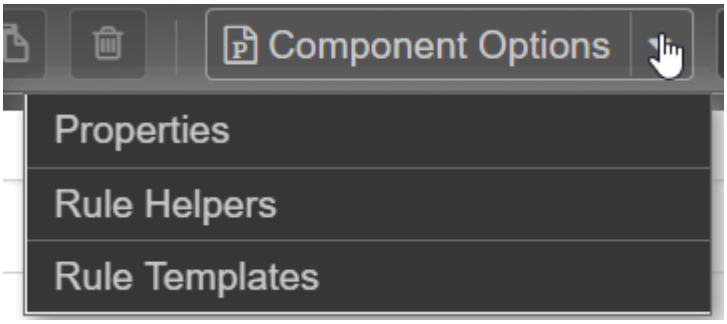
6. Once the new shared component is published, return to the form and refresh the window.
   Refreshing the form will update the content of the original shared component with the contents that you added to the ABCCopy shared component.

Component Options

Unknown macro: 'redirect'

Overview

When a new shared component is being created, there are several options available to identify and extend the functionality of the component. These options are only accessible through the component editor window. The image below displays the Component Options dropdown menu.



The available options are:

- Properties
- Rule Helpers
- Rule Templates

Each of these options are described below.

Properties

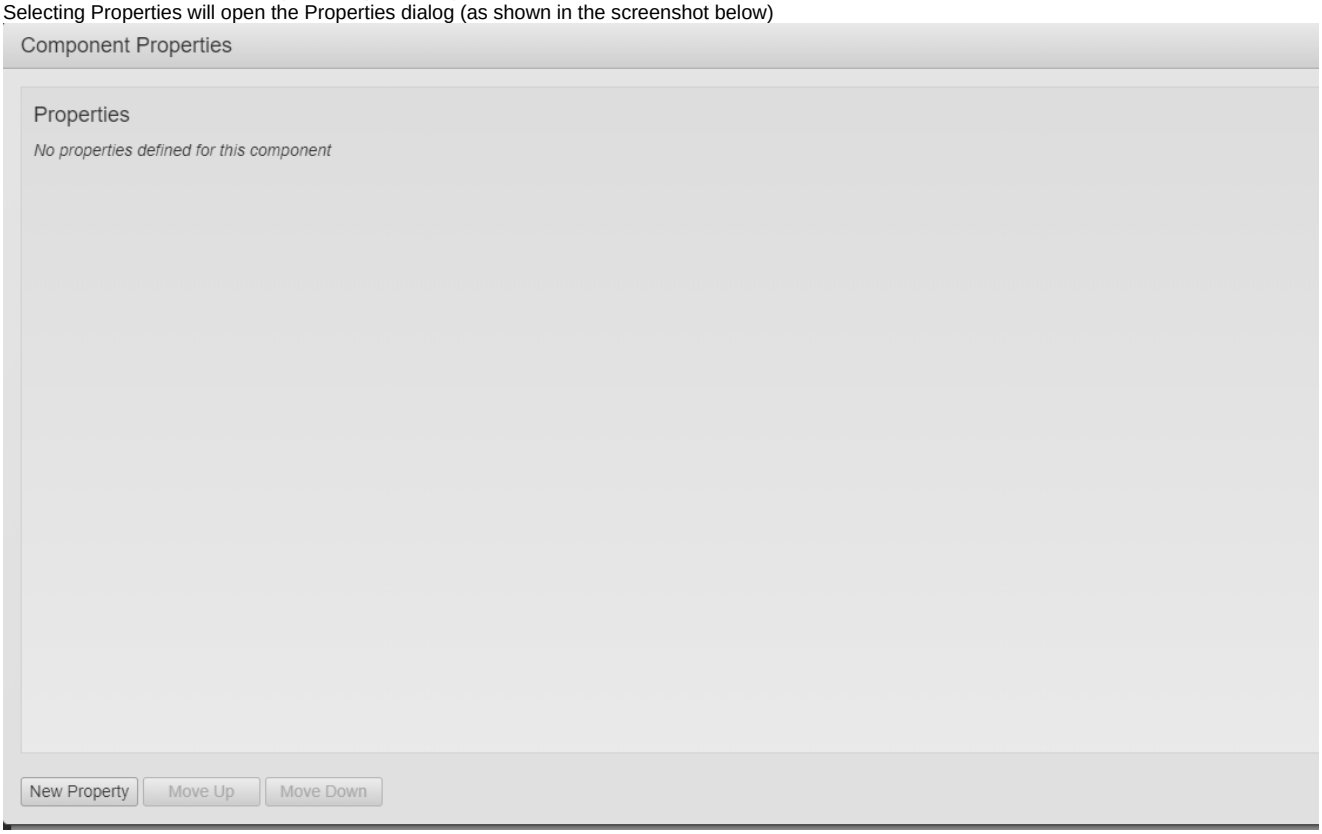
Component Properties are used to configure attributes of a component. These attributes may include branding, and pieces of help text. Individual component properties can be configured through the component editor window.

To configure a property through the component editor window:

1. Select the component and click *Edit Version* from the Component folder in the Dashboard
   Clicking *Edit Version* will direct you to the Component editor window.

2. Select Component Options
   Selecting Component Options will open the Component Options dropdown menu.

3. Select Properties from the Component Options dropdown menu.
   Selecting Properties will open the Properties dialog (as shown in the screenshot below)

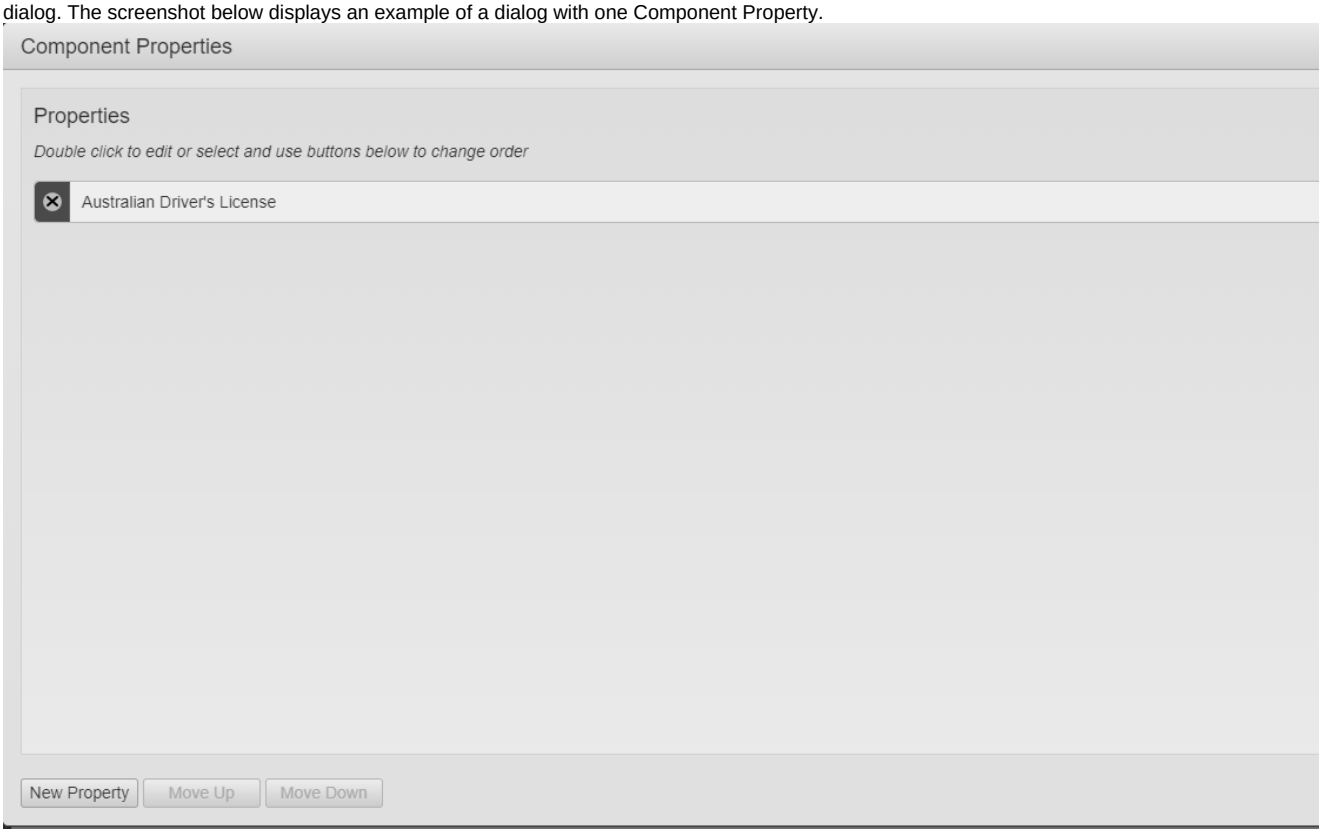


4. Click *New Property*
   Clicking *New Property* will direct you to the New Property configuration dialog screen (as shown in the screenshot below). This dialog is used to enter the specific details about the component.

5. Enter details to identify the Component Property
   Property Label - This is the label that will be visible on the component
   Property Name - The unique identifier of the property (automatically generated based on the label that is inputted)
   Category - The category of component i.e a Options, Input Fields, Background services, etc.
   Type - The type of property that you are creating i.e a Boolean, Options, Field Ref Map, Help Text, etc. The functionality of the property will differ depending on its type.
   Default Value - This field identifies whether the property should be configured with a default value i.e true.
   Help Text - The help text is the text that will be displayed when a cursor hovers over the component. An example help text could be, 'Controls whether to attempt a background check on a driver's license holder'.

   Note that the only mandatory field is the Property Label.
   The screenshot below displays an example of a component property dialog that has been completed with information.

6. Click *Done*
   Clicking Done will direct you back to the Component Properties dialog with the component property that you just created now displayed on the dialog. The screenshot below displays an example of a dialog with one Component Property.



7. Click Save

Rule Templates

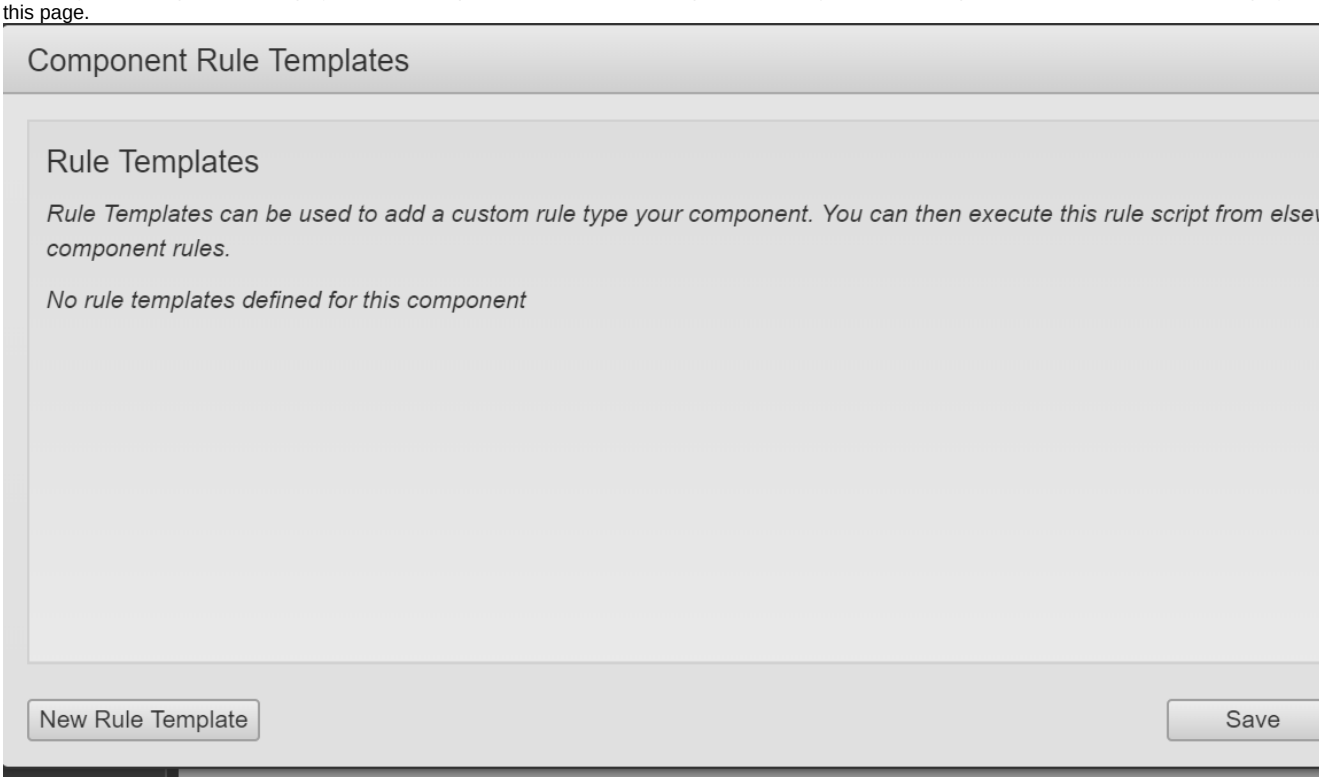
Rule Templates are used to add custom rule types to a component. Once the rule template has been configured, you can then execute this rule script from elsewhere (i.e. in your form)

To configure a Rule Template, follow the steps below.

1. Select the component and click *Edit Version* from the Component folder in the Dashboard
   Clicking *Edit Version* will direct you to the Component editor window.

2. Select Component Options
   Selecting Component Options will open the Component Options dropdown menu.

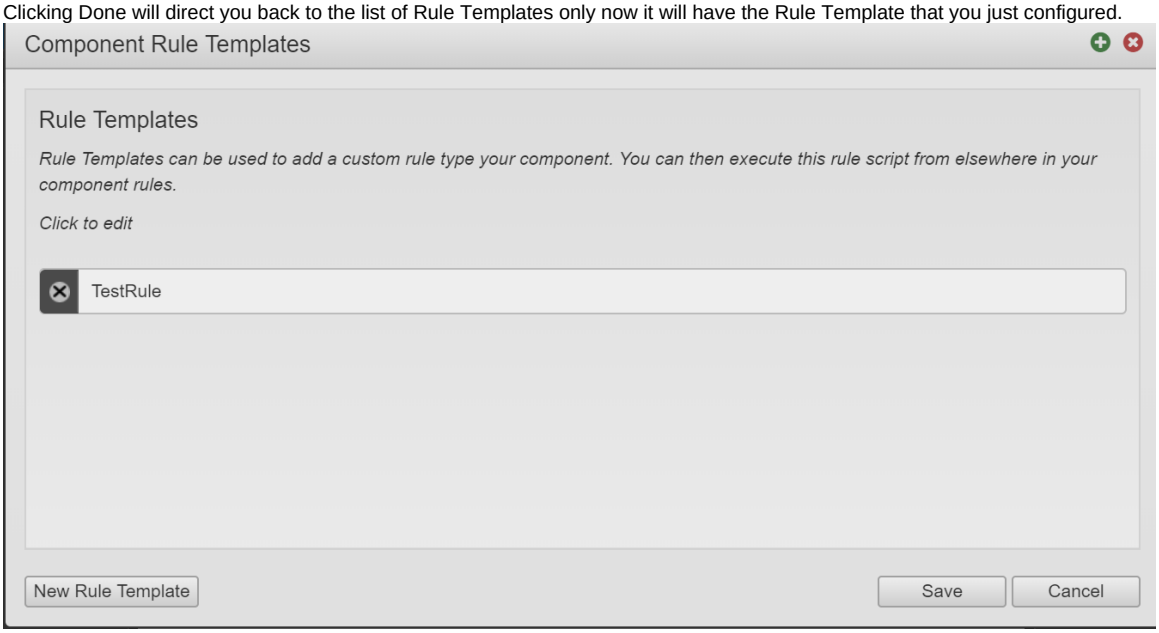
3. Select Rule Templates from the Component Options dropdown menu
   Selecting Rule Templates will display the Rule Templates screen with a list of previous configured Rule Templates. The screenshot below displays this page.



4. Click *New Rule Template*
   Clicking *New Rule Template* will open the New Rule Template dialog.

5. Complete the Dialog by providing a name for the Rule Template
   Providing a name for the Rule Template will automatically prefill the Type of Rule file and the code required to execute this code.
   It is recommended that you copy and paste this code somewhere where you won't forget as you will need this executable command later.

6. Click Done
   Clicking Done will direct you back to the list of Rule Templates only now it will have the Rule Template that you just configured.



Rule Helpers

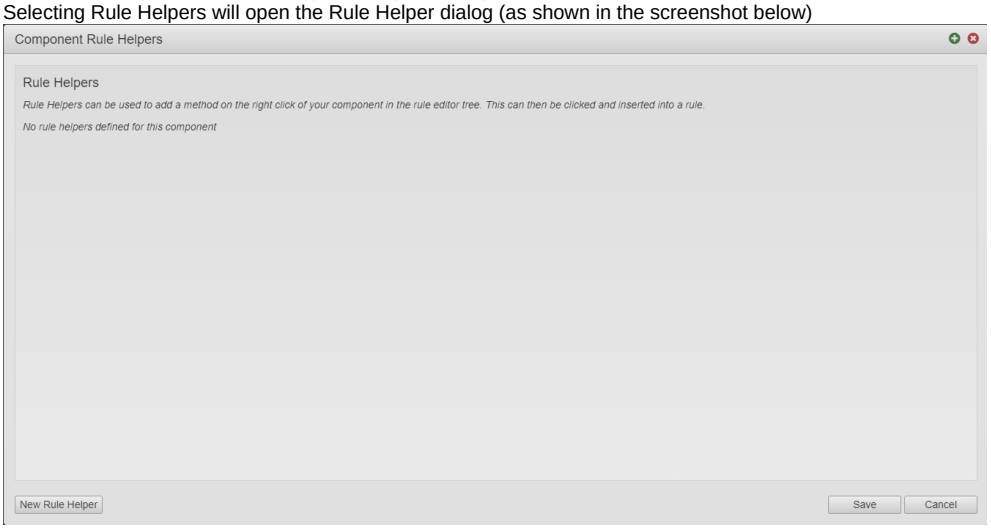
Rule helpers are used to add methods to a right click of a component in the rule editor window. When configured, a Rule Helper will appear as a selectable option when you right-click the component in the rule editor tree.

To configure a rule helper for a customized component, follow the steps below.

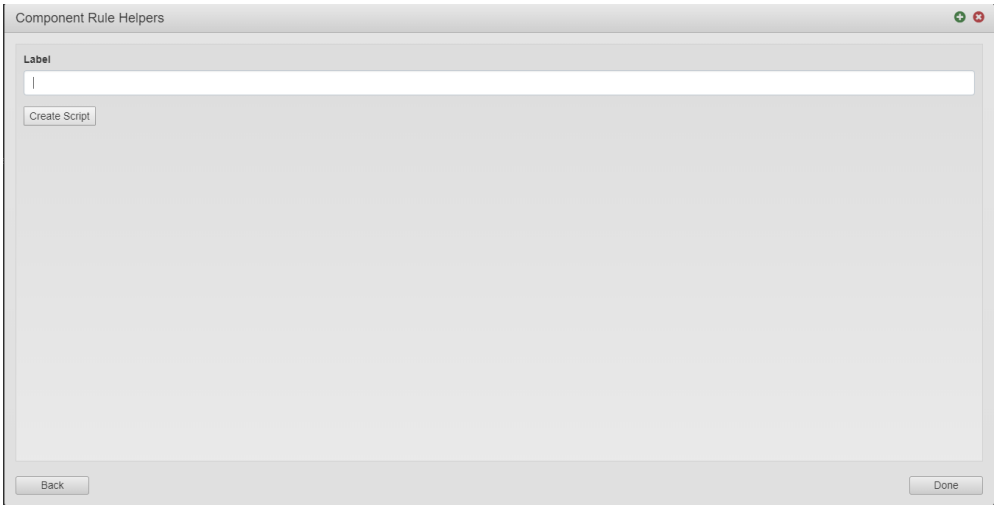
1. Select the component and click *Edit Version* from the Component folder in the Dashboard
   Clicking *Edit Version* will direct you to the Component editor window.

2. Select Component Options
   Selecting Component Options will open the Component Options dropdown menu.

3. Select Rule Helpers, from the Component Options dropdown menu.
   Selecting Rule Helpers will open the Rule Helper dialog (as shown in the screenshot below)

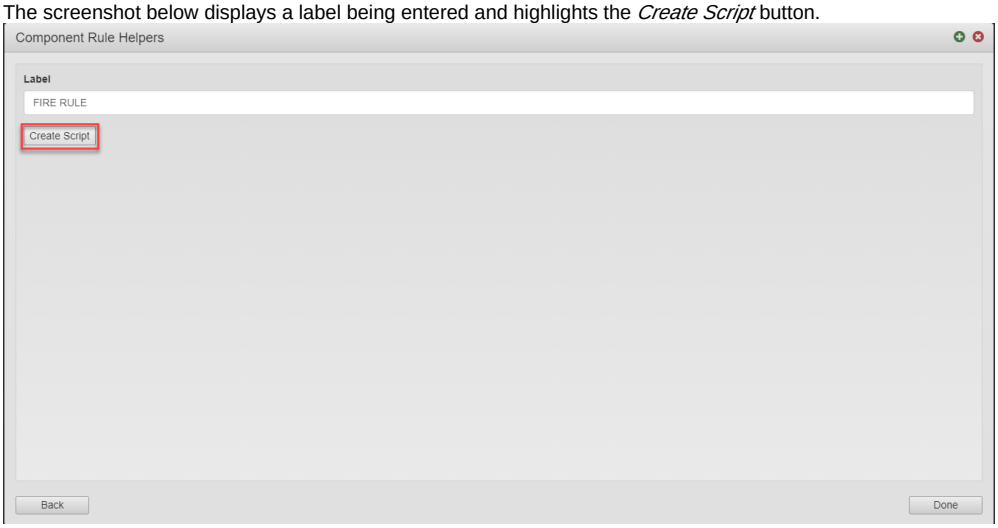


4. Click *New Rule Helper*
   Clicking *New Rule Helper* will direct you to the Create a Rule Helper dialog. The screenshot below displays this dialog.

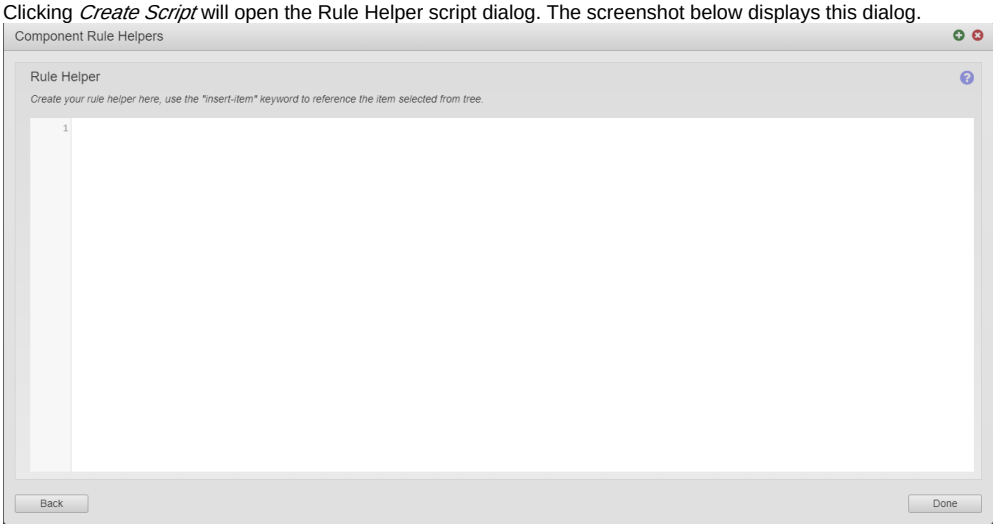


5. Provide a Label for the Rule Helper that you are creating
   This label will identify the rule helper when a user right-clicks the component in the editor tree.

6. Click Create Script
   The screenshot below displays a label being entered and highlights the *Create Script* button.



   Clicking *Create Script* will open the Rule Helper script dialog. The screenshot below displays this dialog.



7. Enter a script for the Rule Helper.
   This script should take the form of Form.fireRule ("name of the Component Template", Form.items.component_ID, data);

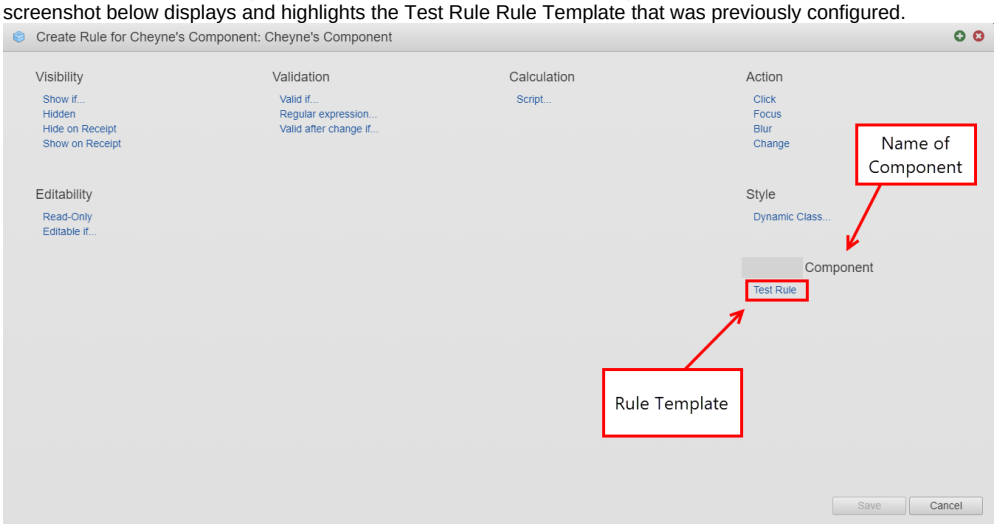
```
Form.fireRule("testRule", Form.items.componentRuleTestID, data);
```

   In the above example, TestRule is the name of the Component Template and componentRuleTest is the component ID of the component that the Rule helper will be associated with (i.e. when a user right-clicks the component with the component ID componentRuleTestID, the configured Rule Helper will be available as a selectable option)
8. Click Done

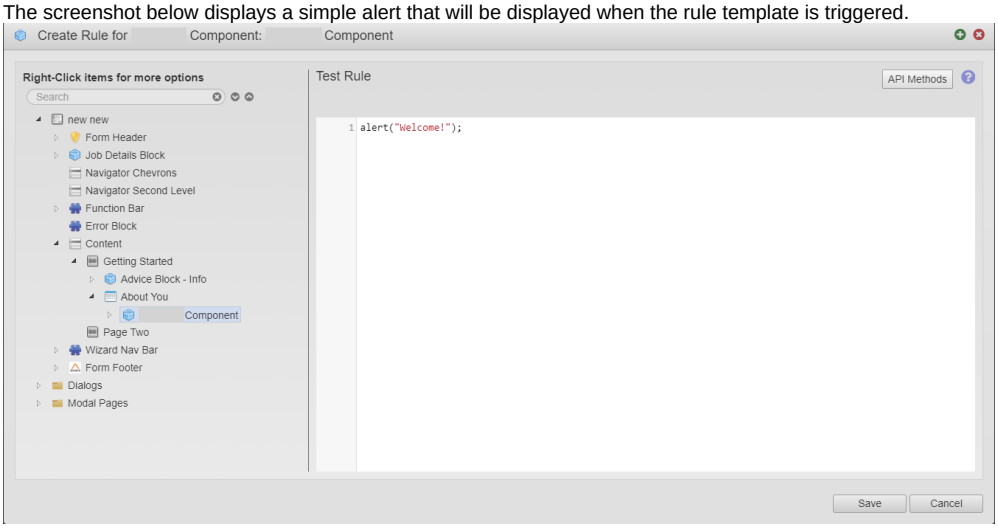
Working with Rule Templates in a Maestro Form

Once you have configured both the Rule Template and Rule Helper, you can put them to work in a Maestro form.

1. Open a form within the project that you just created a rule helper and rule template for
2. Select and edit a form within the project
3. Add the component (the one with the Rule Helper and Template, for this example, the component with the Test Rule Template that was created in the Rule Templates section of this page) to the form
4. Select the component from the Structure tree
5. Click *Create Rule* from the properties tab
   Clicking Create Rule will display the Create Rule Dialog. if the rule template has been configured correctly, it should be a selected rule option. The screenshot below displays and highlights the Test Rule Rule Template that was previously configured.



6. Select the Rule Template under the name of the customized component (in the above screenshot, this selection would be Test Rule)
7. Selecting the Rule Template will display the Create Rule dialog.
   This dialog will vary depending on the configurations of the Rule Template.
8. Add a rule
   The screenshot below displays a simple alert that will be displayed when the rule template is triggered.



9. Click Save
10. Add another component to the form that will execute the rule (for example, you may add a button with a click rule that will execute the rule when the button is clicked).

```
Form.fireRule("testRule", item, data)
```

Form Behavior

Unknown macro: 'redirect'

The content within this section covers information relating to form behavior in Maestro.

The list below identifies the topics covered within this section.

- Business Rules
- Form Options
- Background Save
- Save Challenge (Save and Resume)
- Dialogs and Modal Pages
- Translation

Business Rules

Unknown macro: 'redirect'

The content within this section covers information relating to creating and working with business rules in Maestro.

The list below identifies the topics covered within this section.

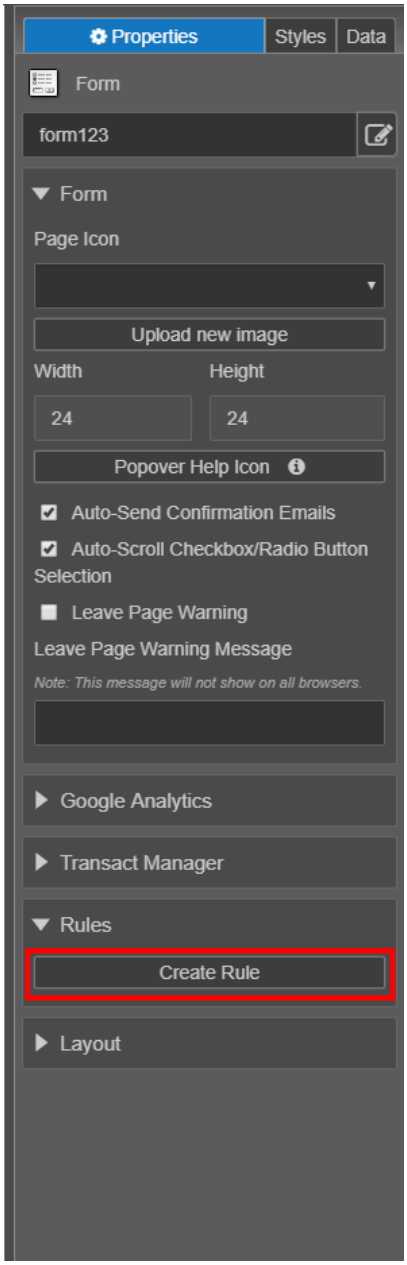
- Creating Rules
- Code View
- Debugging Rules

Creating Rules

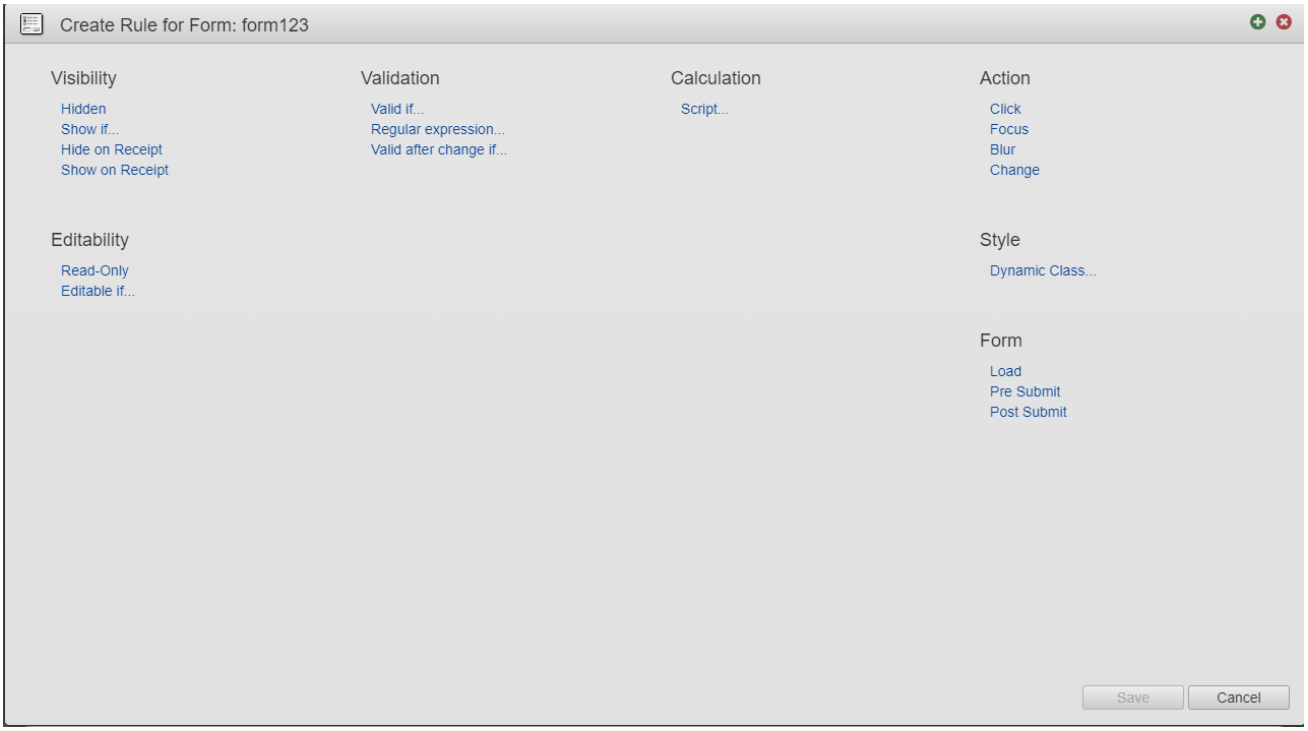
Unknown macro: 'redirect'

How are rules created?

Maestro business rules are created by clicking the **Create Rule** button in the Properties tab of the Maestro editor when a form field is selected. The screenshot below highlights the *Create Rule* button.



Clicking the *Create Rule* button opens the Create Rule dialog. It is through this dialog that you can select the type of rule you want to create. You can only create one rule of each type on each form field. The screenshot below displays the Create Rule dialog.

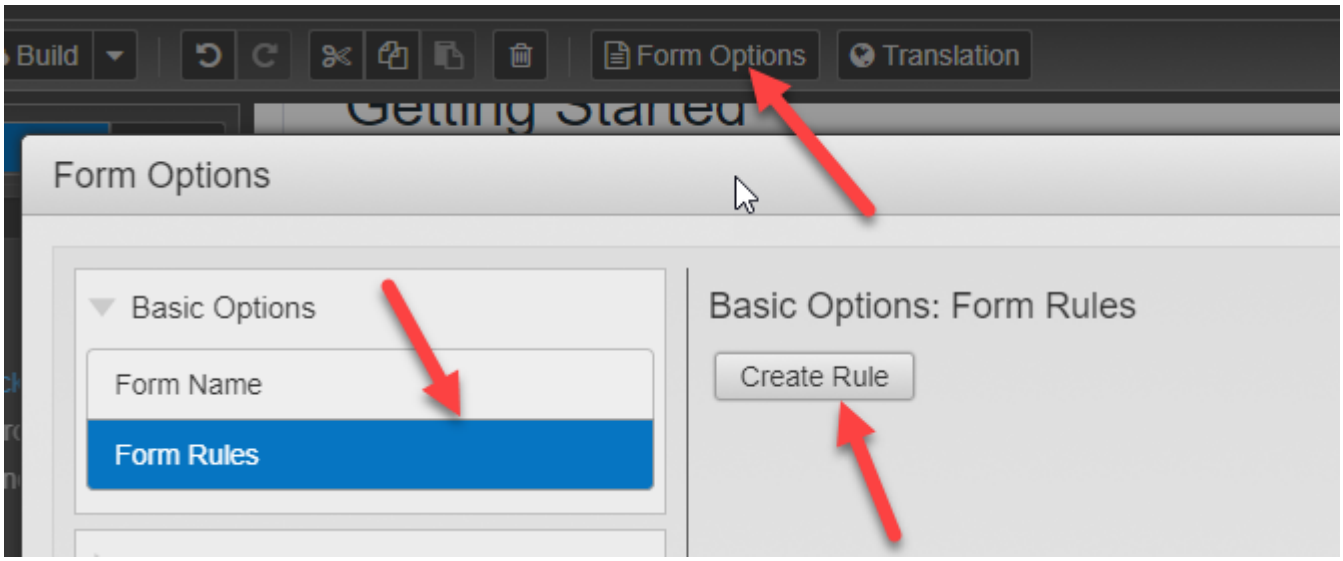


Load Rules

Form Load rules are rules that are applied to the whole form and not single fields. Form Load rules are created slightly differently to regular field rules as they apply to the whole form rather than a single component.

To create a Form Load rule:

1. Click Form Options from the Maestro toolbar
2. Click Form Rules
3. Select Create Rule
   Selecting Create Rule will direct you to the Create Rule dialog where you can create a Load Rule.
   The screenshot below highlights the Form Options, Form Rules and Create Rule selections.



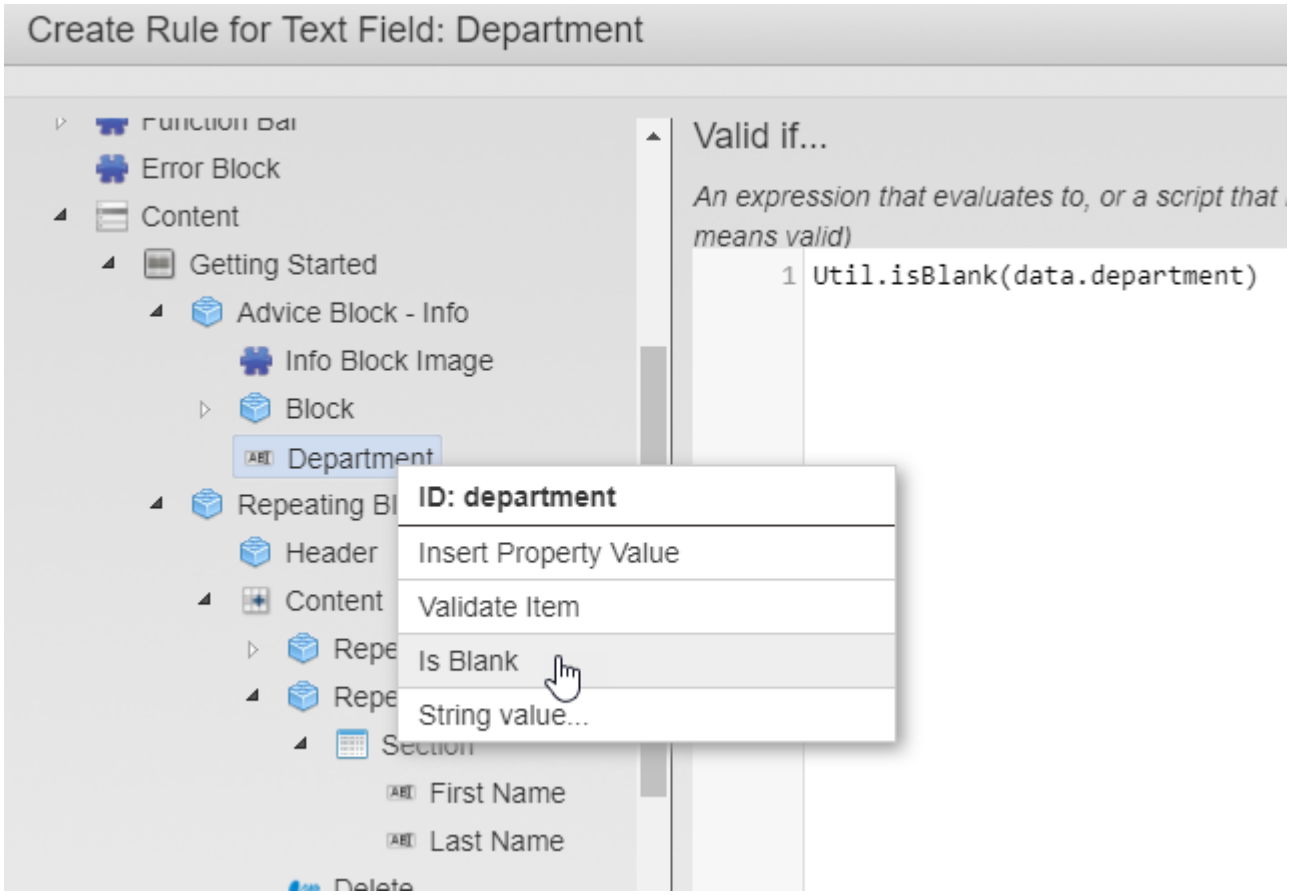
Authoring Rules

Now you have created a rule, you can start to write your rule code. You write the rule code yourself, but there are many Rule Helpers to assist you to write typo-free code. These Rule Helpers are available inside the rule editor, and allow you to pick form fields from a tree structure, then generate the chosen field name, or associated method calls into the editor source.

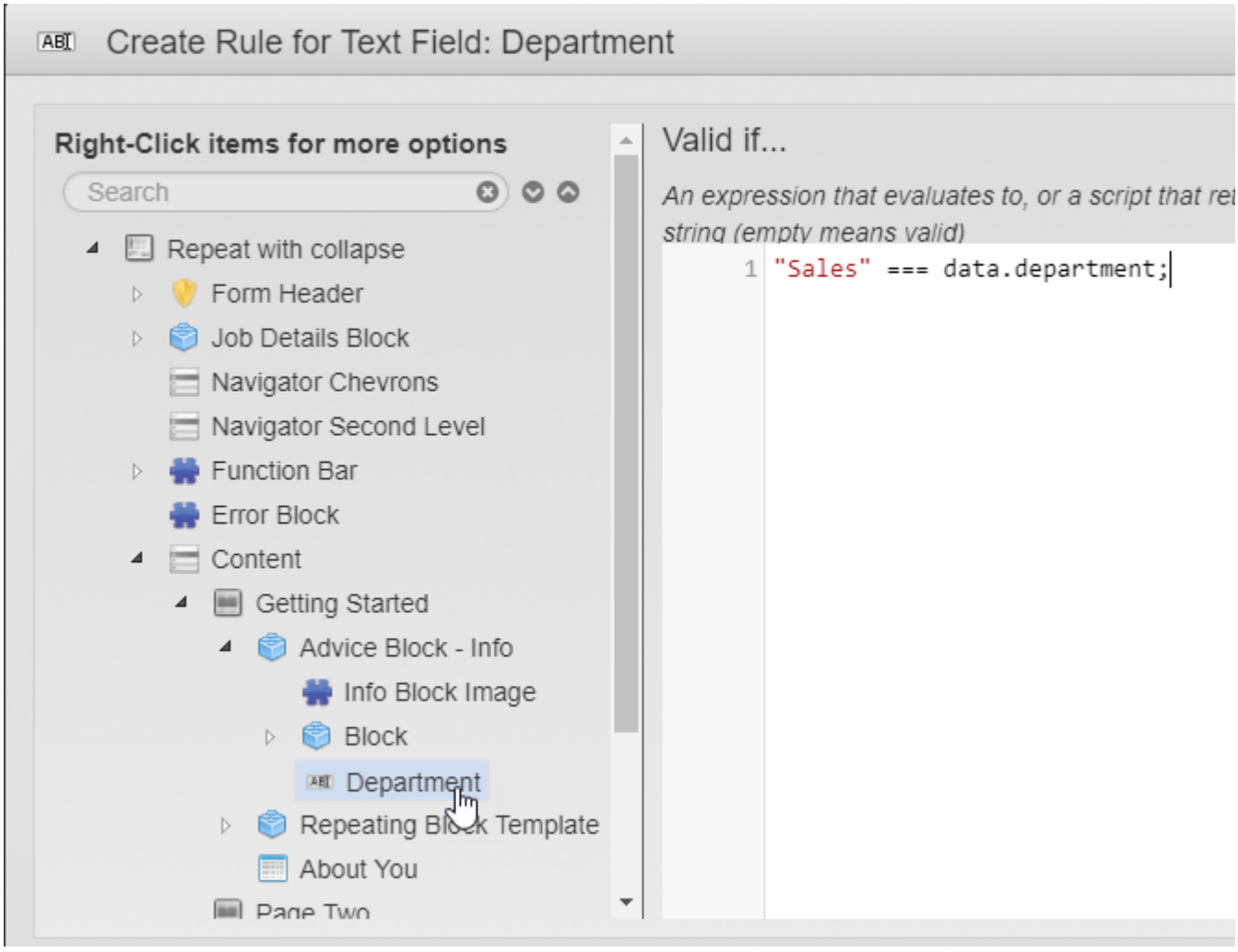
Double-clicking on an item in the tree will insert the field's data reference into the code at the cursor.

Right-clicking on an item in the tree will bring up the list of rule helpers applicable to that item type, and selecting from the list will insert that templated rule code into the source, with the item data reference contained within it.

The screenshot below shows a Rule Helper used to insert code in the rule source. Once generated into the rule source, it is only now necessary to place a '!' in front of the generated code and the non-empty check code is complete.



In the screenshot below, 'data.department' was inserted by double-clicking 'Department' in the tree.



Types of Rules

The list below identifies each of the rule types that are available in Maestro.

- Calculation Rules
- Visibility Rules
- Editability Rules

Calculation Rules

Unknown macro: 'redirect'

Page Contents:

Related Pages:

- Calculation Rules
- Code View
- Editability Rules
- Visibility Rules

Calculation Rule Overview

The Calculation Rule group consists of the Script... rule. The Script... calculation rule provides the freedom to create a variety of calculation rules that can be used for many different purposes.

Creating a Calculation Rule

There are three different options for using the calculation rule. Additional information on each option is detailed below.

1. Mathematical Calculation - Creating a mathematical calculation (+, -, *, etc.) based on numerical data entered into components, or based on defined number values.
2. Duplicate Data from Another Component - The data entered in a component will be used to populate other components in the form.
3. Concatenate Multiple Components - This option allows you to take data entered into multiple components and combined the entered data into one component. An example of this is asking for a First Name and Last Name and combining them into a single Full Name component.

Mathematical Calculation

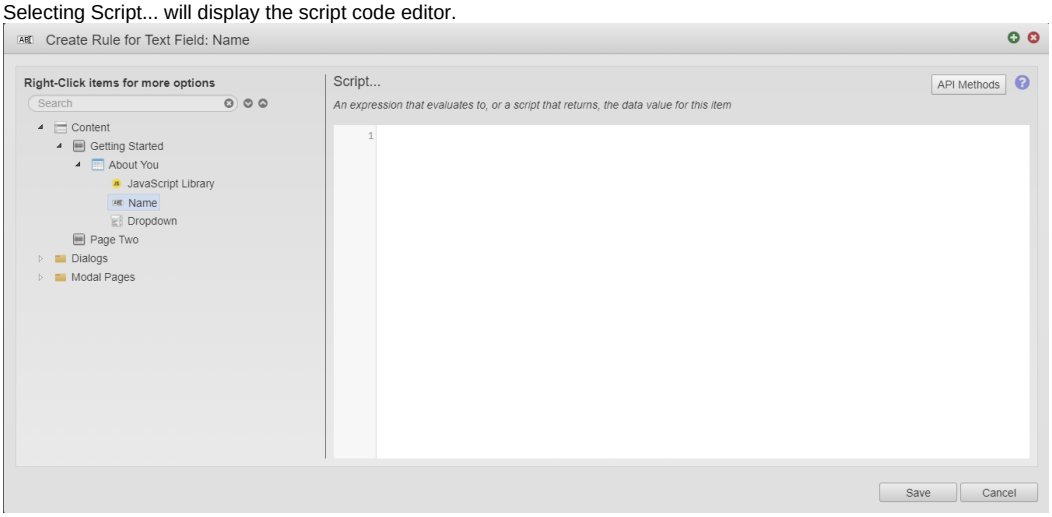
> **When to Use the Mathematical Calculation Option**
>
> Use the Mathematical Calculation option when you are working with values representing numerical figures.

The Mathematical Calculation option allows you to create simple and complex mathematical calculations (+, -, *, etc.). Calculations can be created using data entered in components on the form, or by using defined numerical values.

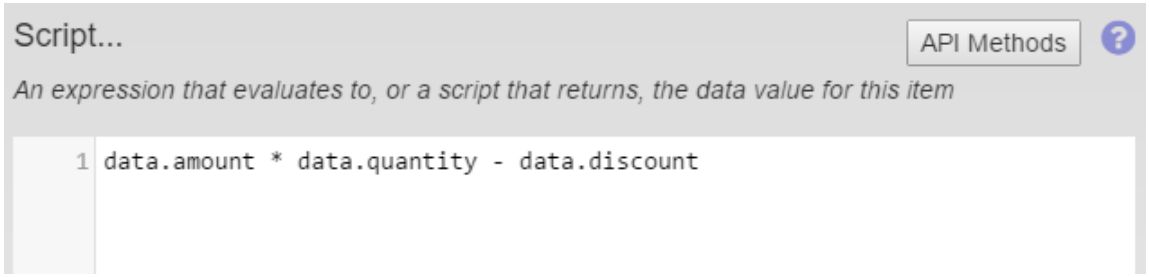
The steps below identify the process of creating a Mathematical Calculation.

1. Select the component that will display the result of the calculation.
2. Click the Create Rule button from the Rules section in the Properties pane of the Maestro editor.
3. Select Script... under Calculation.
   The screenshot below highlights the Script... option.

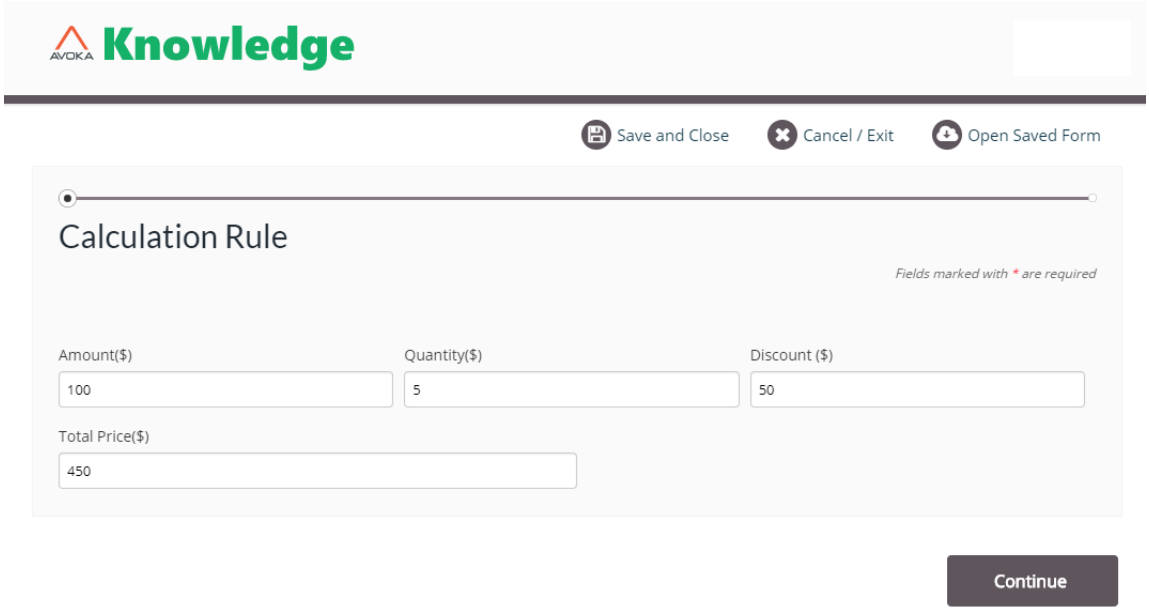
Selecting Script... will display the script code editor.



4. Expand the form structure as needed and double-click the component you want to include in the calculation
5. Insert a mathematical symbol (for example +,( ), - , *, /)
6. Double-click the next component you want to include in the calculation
7. Repeat steps 5 and 6 until you have completed the calculation
8. Click Save
   The screenshots below display an example of a mathematical calculation rule. In this example, the amount value is multiplied by the value entered into the quantity value and then the discount is minused from the value.



The screenshot below displays how this rule will display when the form is built and rendered.



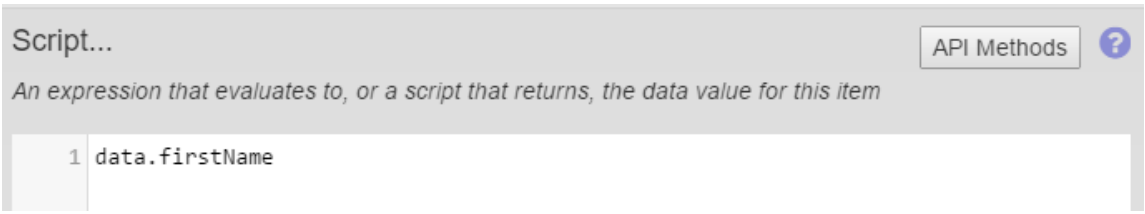
Duplicate Data from Another Component

When to Use the Duplicate Data from Another Component Calculation Option

Use the Duplicate Date option when you want to display a

The Duplicate Data option allows you to use data from different components in other parts of the form. For example, you could use an entered value such as first name in a text display. This will allow you to dynamically customize areas of the form. You could also re-use entered data so that the form user doesn't have to re-enter data multiple times.

1. Select the component where you want the entered value from another component to displayed.
2. Click the Create Rule button from the Rules section on the Properties pane.
3. Select Script... under Calculation.
4. Expand the form structure and double-click to add the component with the entered value you want to display.
5. Click Save



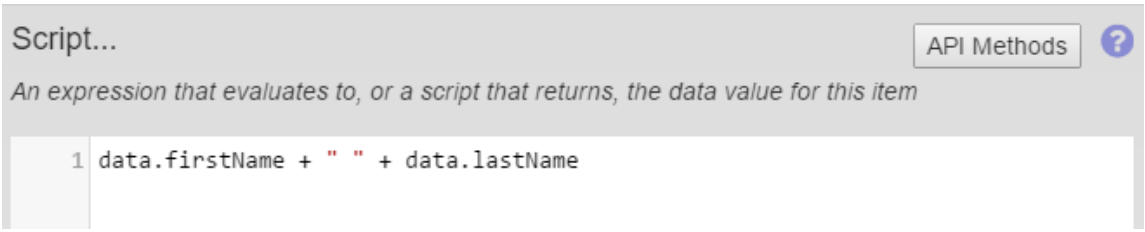
The script will be added to the window. This will basically copy the entered data from one component to another.

Concatenate Multiple Components

When to Use the Concatenate Multiple Components Option

The Concatenate Multiple Components option allows you to combine the value from more than one component into a different component. For example, you may have text fields for First Name and Last Name and you want the entered values from both to display in the Full Name field.

1. Select the component where you want the result to be displayed.
2. Click the Create Rule button from the Rules section on the Properties pane.
3. Select Script... under Calculation.
4. Expand the form structure as needed and double-click the component with the entered value you want to display in the select component.
5. After the entered script we need to add the following: + " " +
6. Double-click another component you want to add to the selected component.
7. Repeat steps 5 and 6 until all components necessary have been added.
8. Click Save

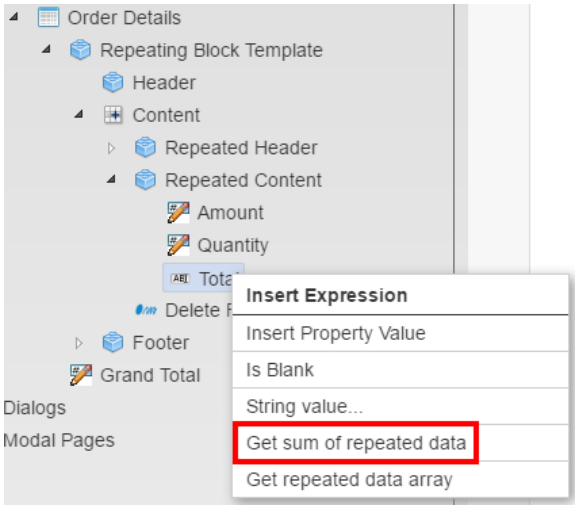


After the plus sign we will enter a space surrounded by quotation marks along with another plus sign. You may use either single or double quotes as long as you are consistent. The plus sign will concatenate when related to text input and add when relating to numbers. The " " adds a space between the combined components. The quotations need to be around any characters, including any spaces that you want to display in the resulting component.

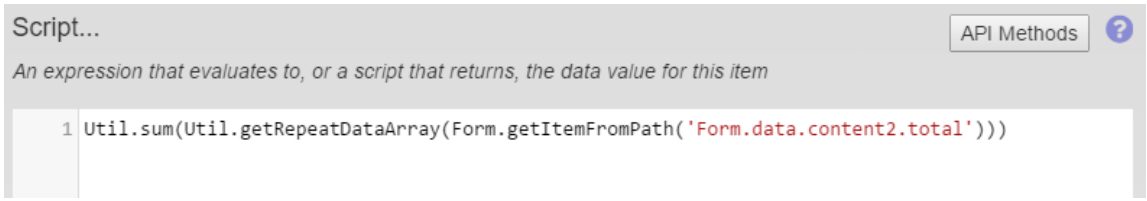
Script... for Repeating Blocks

You can create a rule on a component outside of a repeating block that will sum a selected component within each repeat.

1. Select the component outside of the repeating block
2. Click the Create Rule button from the Rules section of the Properties pane.
3. Select Script...
4. Expand the repeating block template (Repeating Block Template > Content > Repeated Content)
5. Right-click the component you want to sum
6. Select Get sum of repeated data



The following script is added to the window. This will sum the component within each repeating instance.



Visibility Rules

Unknown macro: 'redirect'

Related Pages:

- Calculation Rules
- Code View
- Editability Rules
- Visibility Rules

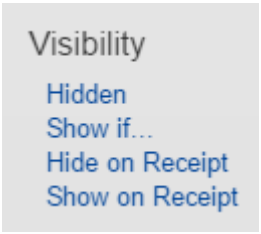
Page Contents:

Visibility Rule Overview

In Maestro, a visibility rule is a rule associated with the visibility of components or assets in a Maestro form.

There are four rules available under the Maestro Visibility Rule group.

- **Hidden** - A Hidden rule will hide a component on a Maestro form.
- **Show if...** - A Show if... rule will hide the component on a form and only show it if certain conditions are met.
- **Hide on Receipt** - A Hide on Recipt rule will hide the selected component on the generated receipt. The component will display on the form, but not in the receipt.
- **Show on Receipt** - A Show on Recipt rule will hide the component on the form and only show it on the generated receipt. A component with a Show on Recipt rule will ONLY be visible on generated receipts.



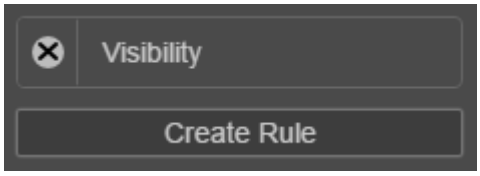
Hidden

The Hidden rule will hide the selected component on the form.

To create a hidden rule for a component:

1. Select the component you want to hide
2. Click the *Create Rule* button from the Properties pane
   Clicking *Create Rule* will display the Create Rule window
3. Select *Hidden*.
   Selecting *Hidden* will close the Create Rule window and when the form is built and rendered, the selected component will be hidden from the form user.

When a Hidden rule is configured, a Visibility rule item will be displayed in the Rules section of the Properties panel for the selected Maestro form. If you want to make any changes to the existing rule, you can do so by selecting this item from the panel. You can also click the X icon to remove the rule from the component.



Hidden rule Examples

Hide a component until a selection is made ... Example

Show If...

When to use the Show if... Rule

The Show if... rule should be used when you want to only show a component when certain conditions have been met. These conditions could be answers to questions or selections based on previous components.

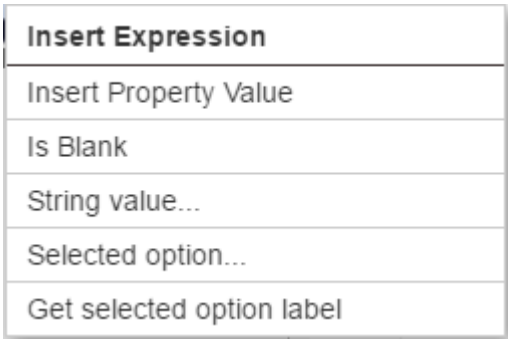
To create a show if rule:

1. Select the component you want to show or hide
2. Click the *Create Rule* button from the Properties panel

3. Select *Show if...*
4. Right-click the component that determines if the selected component is visible or hidden
5. Select the correct expression from the menu
6. Click *Save.*

Insert Expression

The options available from the right-click menu are based on the selected component type and may not be relevant for the specific rule you are creating, as all expression options are displayed.



Is Blank will create a show if rule that will display the hidden component if the selected component is blank.

Selected option... will create a show if rule based on one of the options created for the selected component. This would be relevant for radio button groups or dropdowns.

Insert Property Value and **Get selected option label** are not relevant to the show if rule. Insert Property Value will insert the value of the selected property. Properties include options, arrangement, popover help text. Get selected option label is not relevant for the show if rule. Get selected option label will pull the label from the selected component.

Show If... rule Examples

Hide a component until a selection is made

In this example, a show if rule has been used to only show a Mobile Number (AUS) component if the answer to the radio button question is, "Yes".

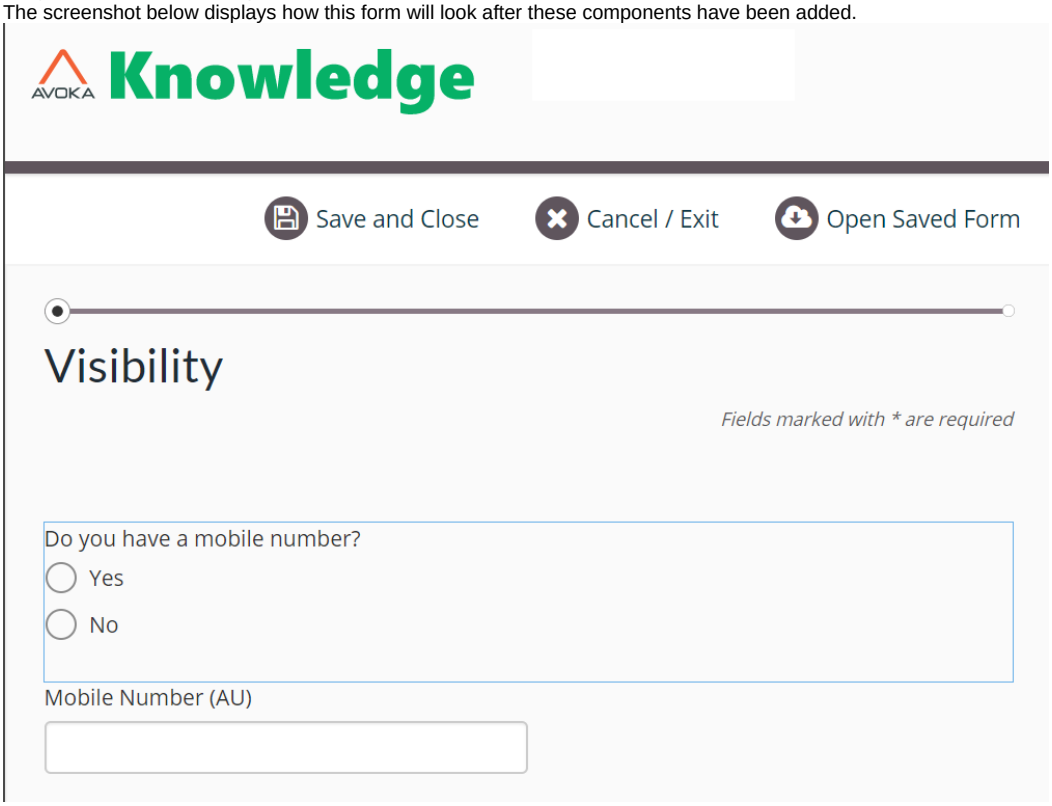
What does this example show?

If a form user answers, Yes, indicating that they do have a mobile number, then a mandatory text field is displayed for the form user to enter his/her mobile phone number.

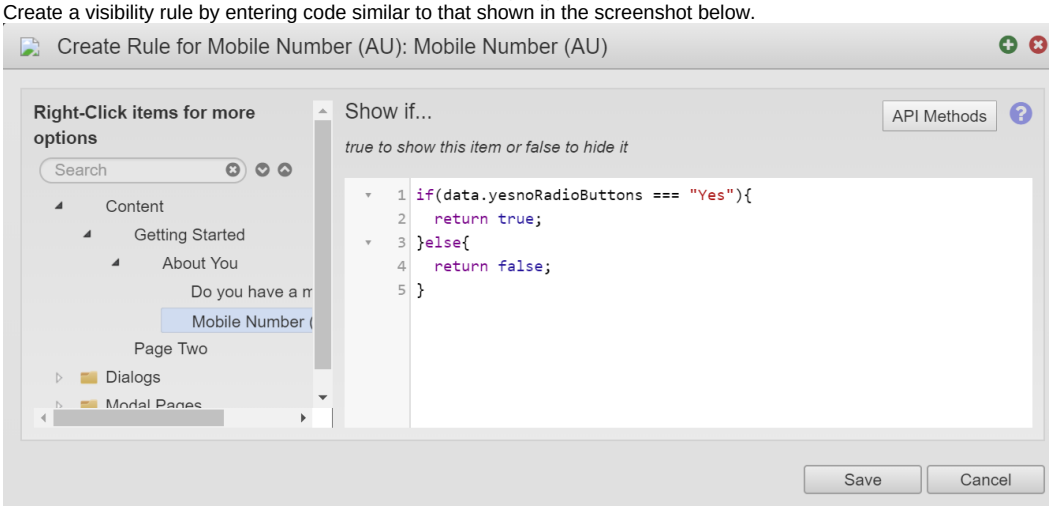
To build this example:

1. Drag a radio button group component to a form

2. Drag a Mobile Number component to a form
   The screenshot below displays how this form will look after these components have been added.

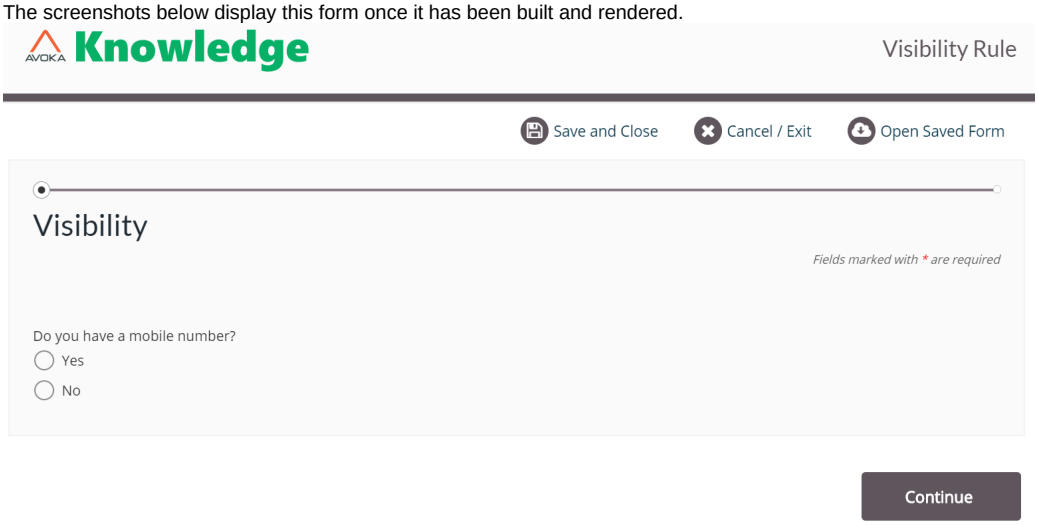


3. Select the component that you only want to show if YES is selected from the radio button group.
4. Click Create Rule and select Show if... from the Visibility group of rules
5. Create a visibility rule by entering code similar to that shown in the screenshot below.

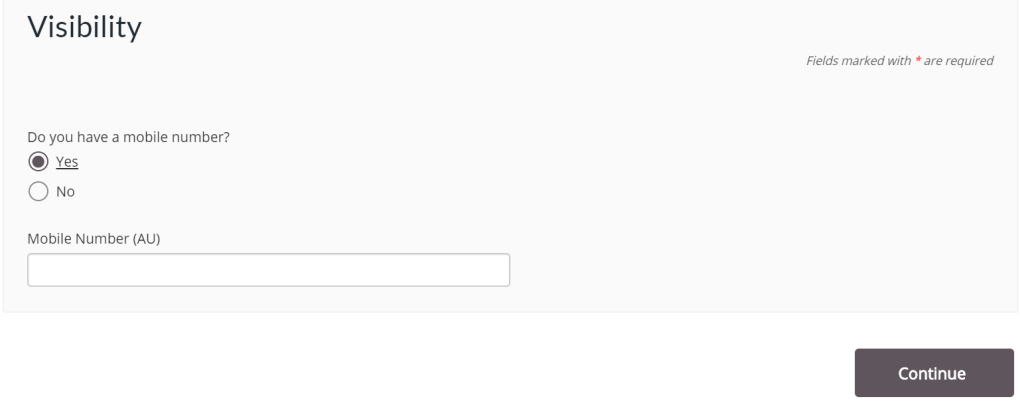


In the above screenshot, if Yes is selected in the radio button group, the script will return true and the mobile number component will show in the form, otherwise the script wil return false and the the mobile number component will not show in the form.

The screenshots below display this form once it has been built and rendered.



If Yes is selected, the Mobile Number component will show.



Hide or Show on Receipt

When to Use the Show or Hide on Receipt Rules

The Hide or Show on Receipt should be used when you want to control what form users will see on a receipt.

When a transaction is completed by a form user, a receipt is rendered and emailed to the user. This receipt is used as a record for form users. By default, most components will display on both the form, and on the generated receipt. You can use the Hide or Show on Receipt rules to hide components on the form and only show them on a receipt, or hide the components on the receipt and only show them on the form.

Hide on Receipt

When applied, the Hide on Receipt rule will hide the component from displaying on a generated receipt (though the component **WILL** display on the built and rendered form).

To configure a Hide on Receipt rule:

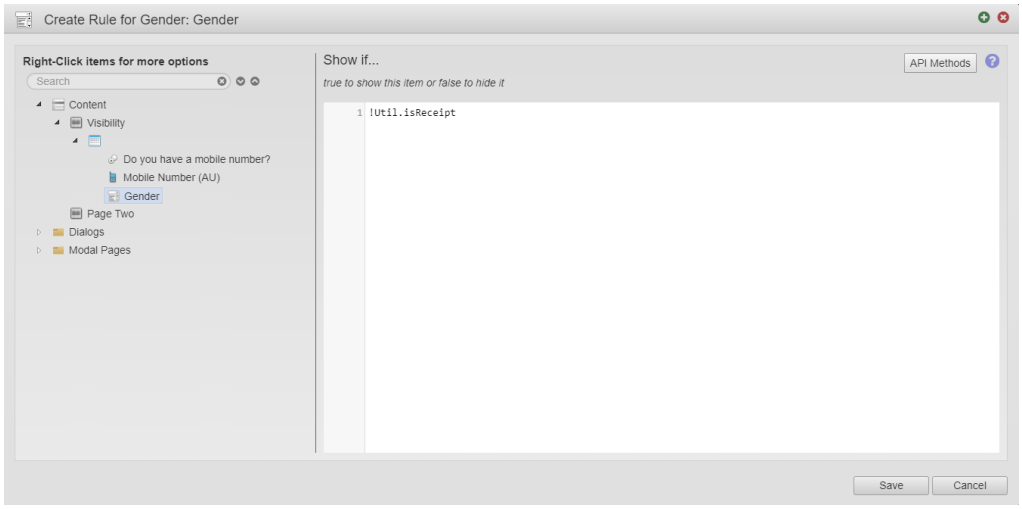
1. Select the component you want to hide on the receipt
2. Click the *Create Rule* button
   Clicking *Create Rule* will display the Create Rule window
3. Select *Hide on Receipt*
   Selecting *Hide on Receipt* will close the Create Rule window and when the form is built and rendered, the selected component will be hidden from the form user.

When a Hide on Receipt rule is configured, a Visibility rule item will be displayed in the Rules section of the Properties panel for the selected Maestro form. If you want to make any changes to the existing rule, you can do so by selecting this item from the panel. You can also click the X icon to remove the rule from the component.

Hide on Receipt Example

In this example, the Gender component has been configured to not show on a rendered receipt when this form is submitted by a user.

The screenshot below displays the script that is created when the Hide on Receipt rule is applied to a Gender component.



When this form is submitted, the generated receipt will NOT display the gender of the form user.

Show on Receipt

When applied to a component, the Show on Receipt rule will hide the component on a form and only show it on generated receipts. The component will still be displayed in the Editor window, however, when you preview or build the TM form version, the component will not be displayed.

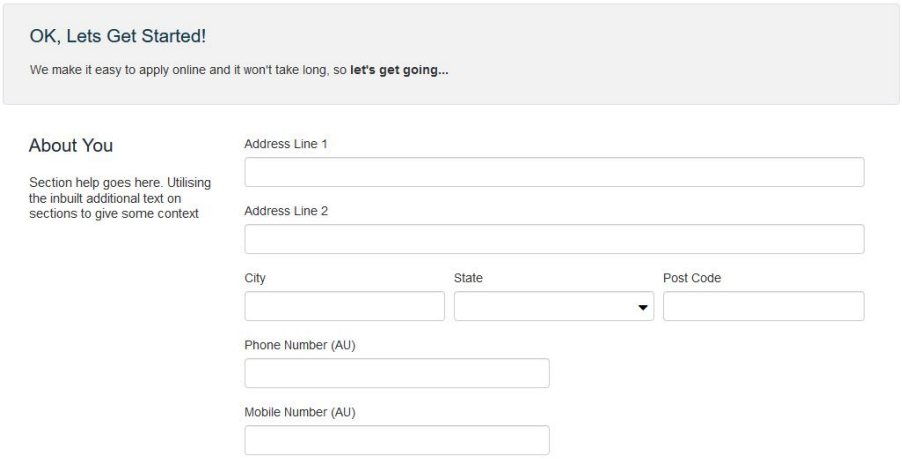
To configure a Show on Receipt rule:

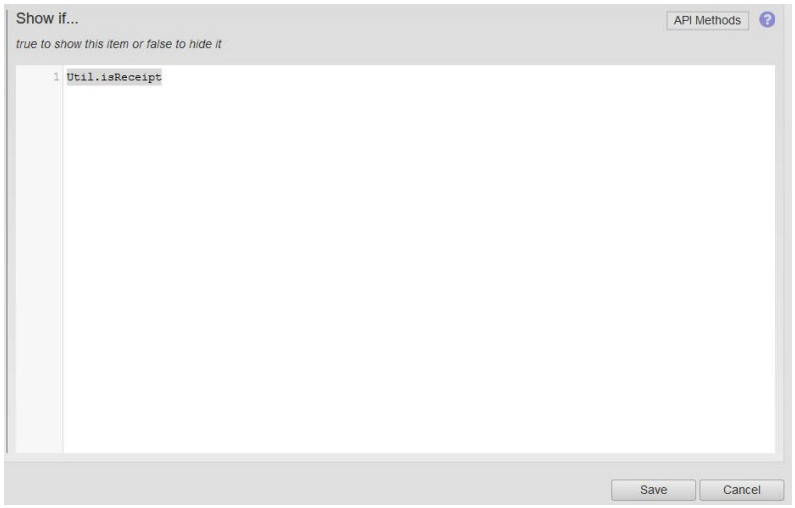
1. Select the component you want to hide on the receipt
2. Click the *Create Rule* button
   Clicking *Create Rule* will display the Create Rule window
3. Select *Show on Receipt*
   Selecting *Show on Receipt* will close the Create Rule window and when a receipt is generated, the selected component will be shown on a generated receipt.

When a Show on Receipt rule is configured, a Visibility rule item will be displayed in the Rules section of the Properties panel for the selected Maestro form. If you want to make any changes to the existing rule, you can do so by selecting this item from the panel. You can also click the X icon to remove the rule from the component.

Show a Component in a Recipt Example

In this example,




Show if...
API Methods
true to show this item or false to hide it
1 Util.isReceipt
Save
Cancel

Editability Rules

Unknown macro: 'redirect'

Page Contents:

Related Pages:

- Calculation Rules
- Code View
- Editability Rules
- Visibility Rules

Editability Rule Overview

Editability Rules are rules that limit the ability to edit the contents inside a component.

There are two editability rules available from the Maestro Create Rule window - Read-Only and Editable if...

Read-Only

The Read-Only rule can be applied to any component on a form. When this rule is applied, the user will not be able to select the component, therefore they will be unable to make any changes.

> **Why Use the Read-Only Rule**
>
> The Read-Only Rule is useful when you only want to display content in a field.

To create a Read-Only rule:

1. Select the component you want to make read-only
2. Click the *Create Rule* button
3. Select *Read-Only*

The read-only rule is applied to the selected component. You will see the editability rule listed in the Rules section of the Properties panel.

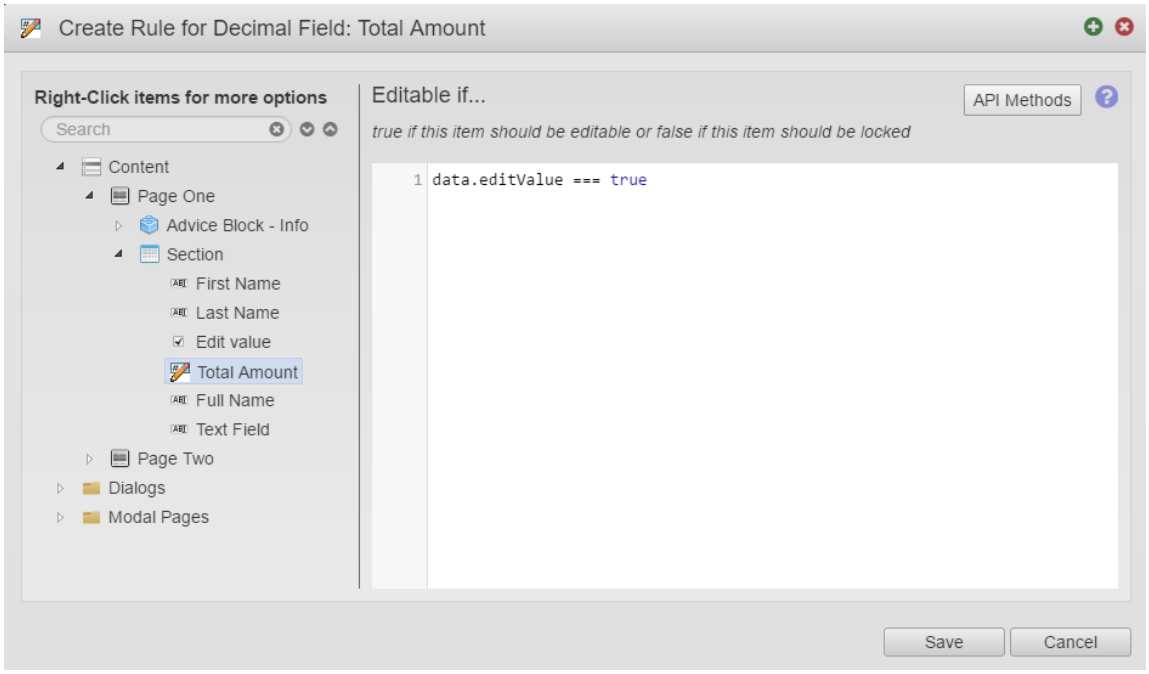
Editable if...

The Editable if... rule makes a field editable only if certain conditions are met. By default, a component with an Editable If... rule will not be editable until the configured condition is met.

To create an Editable If... rule:

1. Select the component you want to apply the rule to
2. Click the **Create Rule** button
3. Select Editable if...
4. You can use the right-click options for the existing components or you can enter your own script to create the editable if rule

The screenshot below shows an editable if rule stating that if the Edit value checkbox is selected then the Total Amount decimal field is editable.

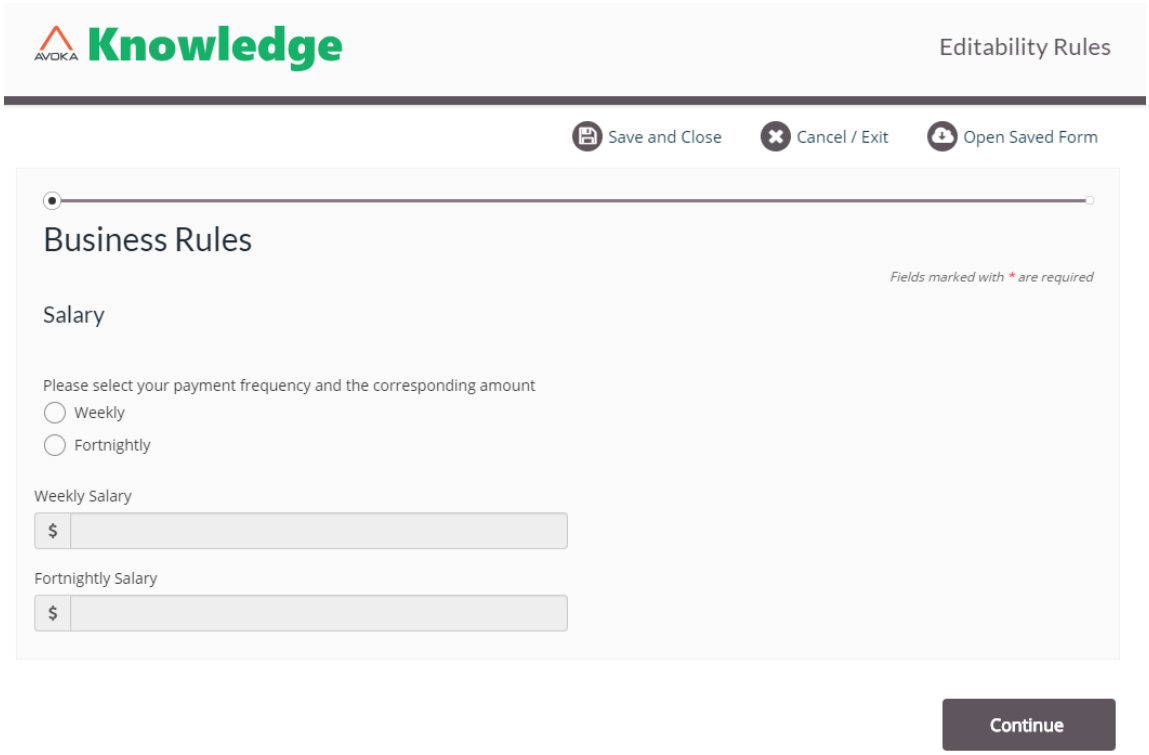


Editable if... Rule Examples

Salary

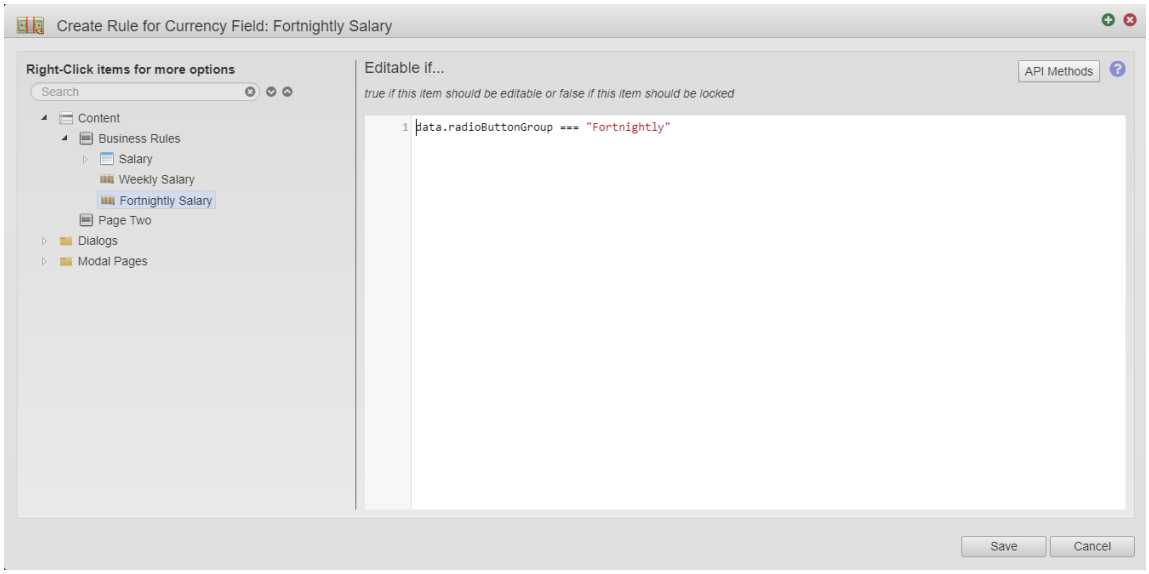
In this example, the Editable If... rule has been used to only allow a user to enter data into specific components. If the user selects the Weekly radio button option, then the Fortnightly Salary currency text field will **NOT** be editable. Alternatively, if the user selects the Fortnightly radio button option, then the Weekly Salary currency text field will **NOT** be editable.

The screenshots below describe this example.



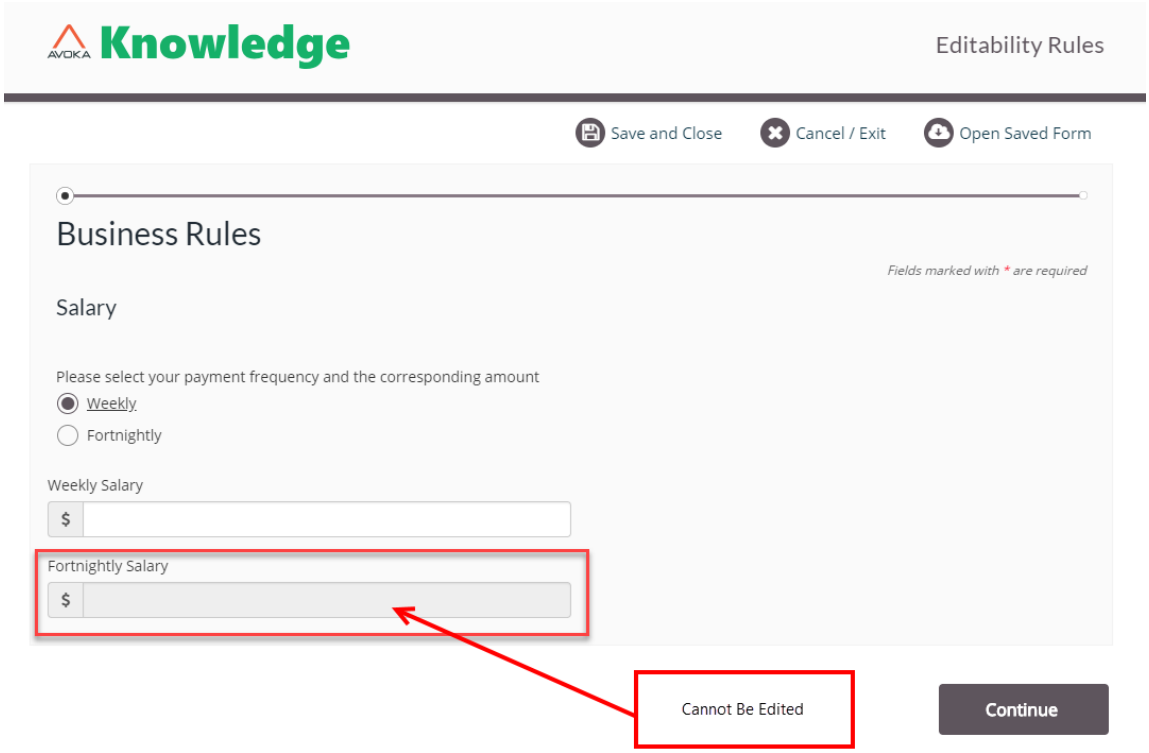
The above screenshot displays a form that has been configured with a radio button group and two currency fields. The form asks the user to select their payment frequency button and the amount that they will pay.

The above screenshot displays the editable if rule that has been applied to the currency field named Weekly. The script in this example says that if Weekly is selected from the radio button group, then the currency field named Weekly will be editable (i.e the user will be able to enter the amount that they will pay in weekly payments).



The above screenshot displays the editable if rule that has been applied to the Fortnightly currency field. The script in this example says that if Fortnightly is selected from the radio button group, then the Fortnightly currency field will be editable (i.e the user will be able to enter the amount they will pay in fortnightly payments).

The above screenshots display calculation rules that have added to extend the functionality of this example.



The above screenshot displays the form once the editability rules have been added to the form. When Weekly is selected from the radio button group, the user will not be able to enter input into the Fortnightly Salary currency field.

Code View

Unknown macro: 'redirect'

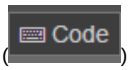
Page Contents:

Related Pages:

- Calculation Rules
- Code View
- Editability Rules
- Visibility Rules

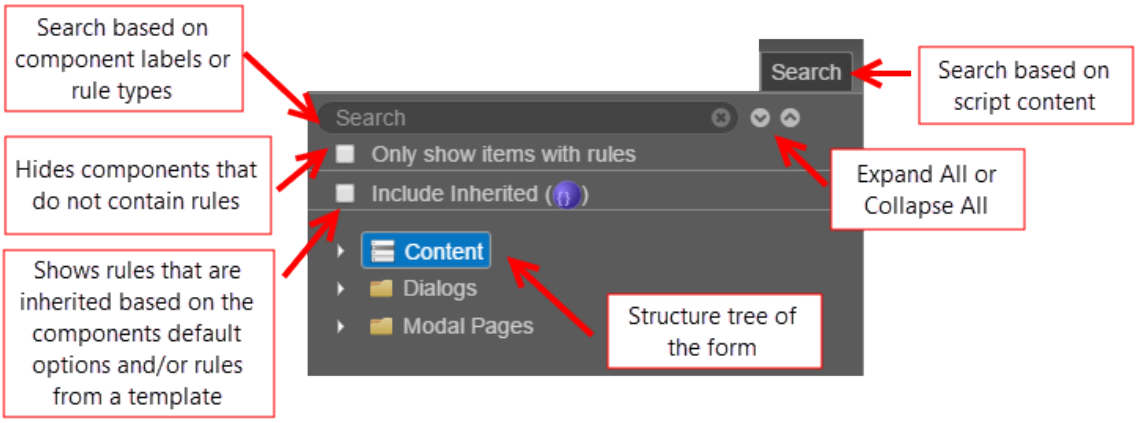
Code View Overview

Maestro forms can contain a large number of business rules which can make editing, managing, and adding new business rules quite complex. To address this complexity, Maestro offers the Code View. The Code View alows you to view, create and modify business rules all in one place.

To access the Maestro Code View, click the Code (

) button from the Maestro editor.

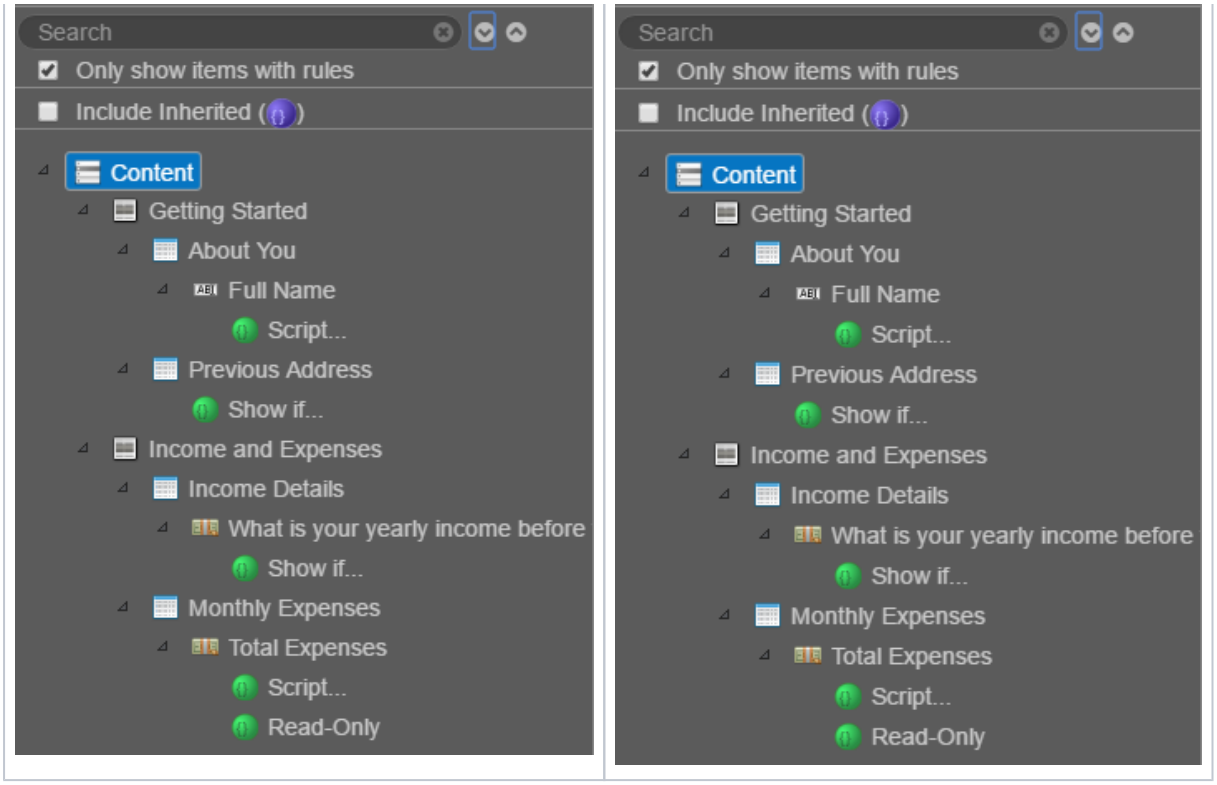
The Code View tree contain business rules associated to components. The screenshot below highlights the features of the Code View tree.



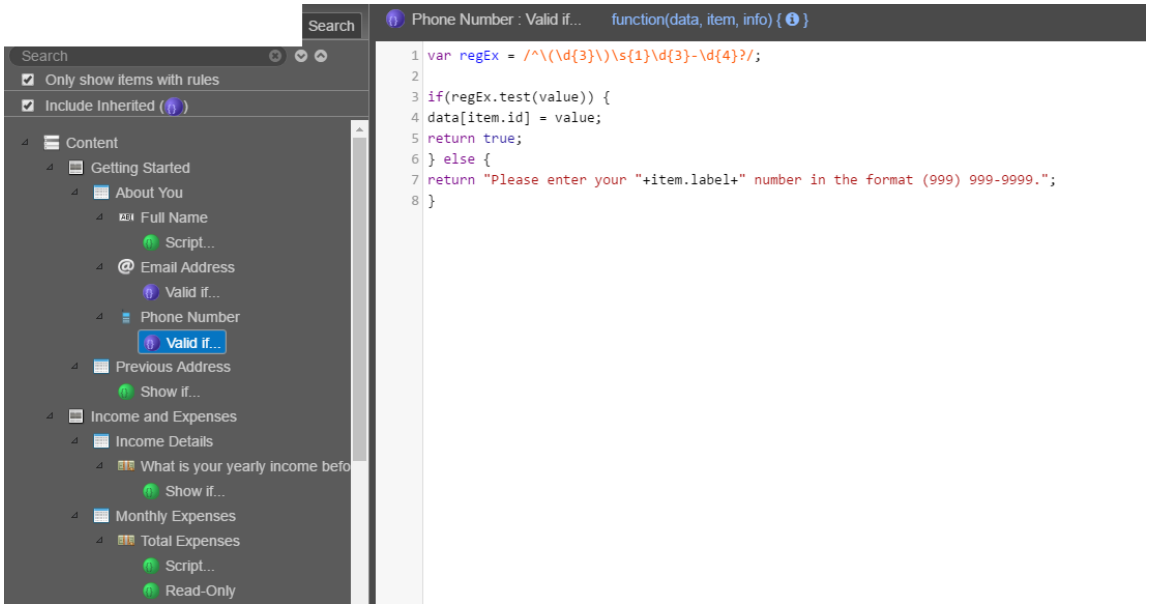
You can expand the tree to view the components and associated rules. The Maestro Code View allows you to search and filter rules that have already been configured for the selected form.

The following screenshots display the Code Vew with the business rules associated with the selected form shown under each of the elements contained in the form.

Without the inherited rules checkbox being selected	*With* the inherited rules checkbox being selected



When a rule is selected from the tree, the code of the rule will display in the editor area (right side of the window). This area and view is similar to the existing rule editor, where the user can edit the code directly or create new rules by right-clicking any component in the displaying panel. The screenshot below displays the editor area of the Code View.



Search in Code View

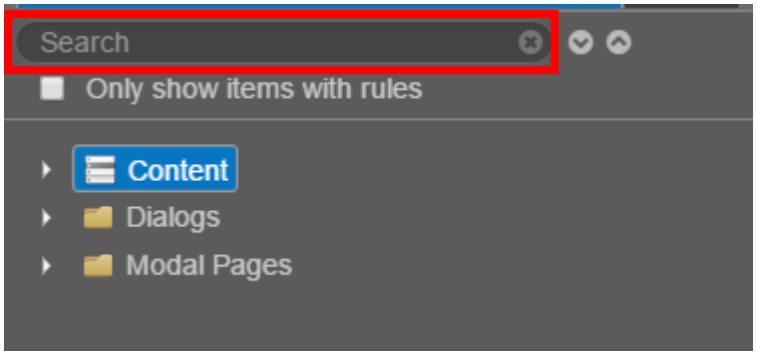
When working with large complex forms, it can sometimes be difficult to find specific code within the form. Maestro has two search methods available in Code View to help find the code that you are looking for.

The two methods available for searching are:

- Search box on displaying panel
- Search tab

Search Box on Displaying Panel: This search box allows you to filter criteria based on component labels and/or rule names. This search does not search through the actual code, it only looks at the content visible on the design tree.

The *Only show items with rules* checkbox will filter the displayed components to only show those that contain rules. This may reduce the amount of components displayed, making it easier to find the rules you are looking for.



Search tab: Switching to the *Search Tab* displays a search box and several filter checkboxes that can be used to refine the results of your search. This search box allows you to filter based on the content within the code. This search will not look at the component label or rule name, but rather at the code within the rule. The search results will show all methods that match the search criteria, including lines within the method that match the search criteria.

The list below identifies the available filter options.

Match Case - Select this option to filter the search results to only show those that match exactly according to the case stucture used in the original search.

Regular Expression - Select this option to filter the search results to only show regular expressions.

Include Inherited - Select this option to filter the search results to include any inheritance that exists in the rules.

Working in Code View

While in Code View you can view existing rules, edit existing rules and create new rules.

To find the code you want to view and/or edit, you can use the search methods discussed above, or browse the tree in the displaying panel. Once you find what you are looking for, you can then select the item from the panel.

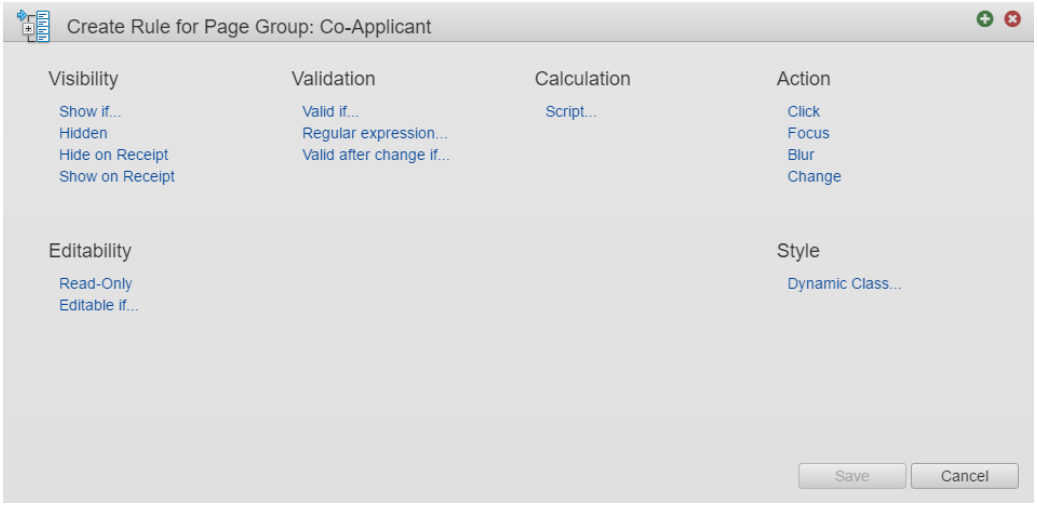
The code for the selected rule will display in the editor area of the window.

You can edit the rule directly in the code editor, or right-click the component to use any of the code assist options to edit the selected rule.

Add New Rule

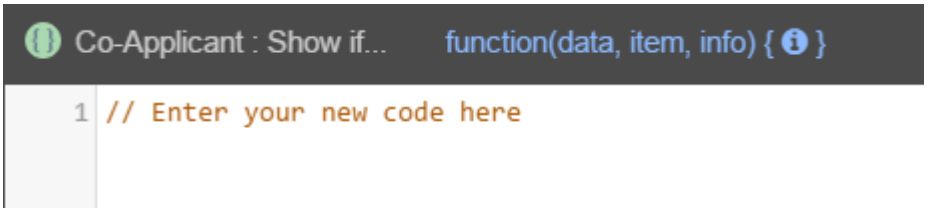
When you right-click a component while in Code View, you can select **Add New Rule.**

Selecting **Add New Rule** will display the Create Rule window. This is the same window displayed when you click the Create Rule button on the Properties panel in Design View.



Once you select the type of rule you want to create, the Create Rule window will close and you can enter your new code into the editor area.

You can enter the code directly (in the edito window), or you can right-click the component to use the code assist options.

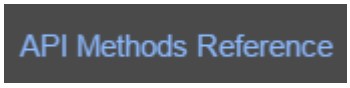


Additional Code View Resources

There are additional resources available in the Code View to assist with rule creation.

API Methods Reference

In the top right corner of the window you will find the API Methods Reference link. Selecting this link will open a webpage that contains information on API methods published for use in Maestro business rule functions. It is organized by *Namespaces*, which are the references injected into the content of rule functions.



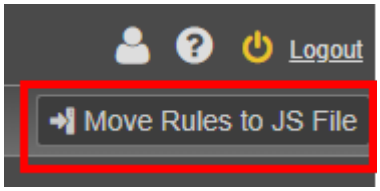
Maestro Rule Functions

Above the editor area you will find a function(data, item, info) link. This link will open a new browser tab that contains information on Maestro rule functions. These rule functions can be created in the Code View or in the rule editor window accessed from the Rules section of the Properties panel.

Move Rules to JS File

Traditionally, Maestro and other Transact tools relied on ìn-built coding editors to create, edit, and manage Maestro business rules, but with the move to integrating Source Code Management (SCM) tools such as Git into the Transact platform, Maestro has introduced the ***Move Rules to JS (JavaScript) File*** feature. This feature moves all selected business rules from a Maestro form to an external JavaScript file that can be referenced in any Maestro form. The Move Rules to JS File feature is used to create a one place JavaScript resource that holds all the JavaScript rules (or a selected few) for a Maestro form. This file can then be opened and edited in any SCM tool before being referenced into a Maestro form by using the JavaScript library component from the Maestro Palette.

The screenshot below highlights the **Move Rules to JS File** button.



Why Use the Move Rules to JS File Feature

The Move Rules to JS File feature should be used when you are looking to extend the functionality of your form by using Source Code Management (SCM) tools such as Git. Or if you want to consolidate all the rules for your form into one place to make editing them easier as your form grows and matures.

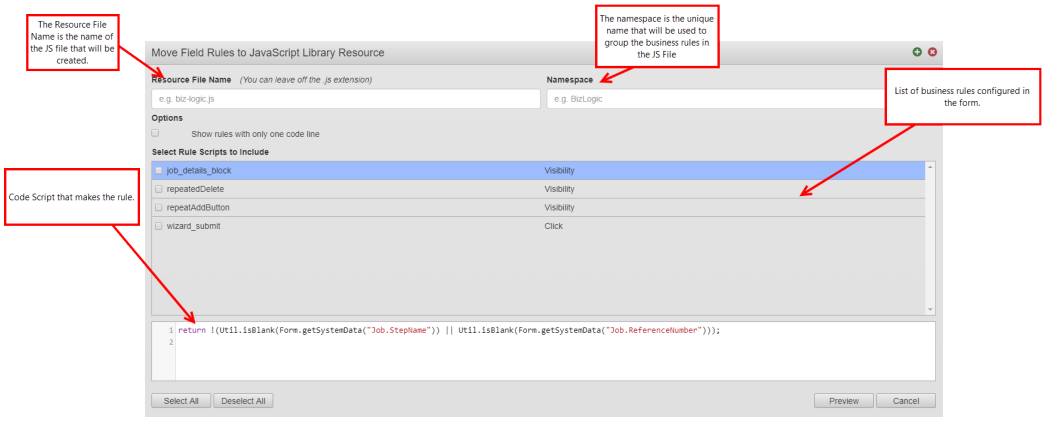
Using the Move Rules to JS File Feature

The Move Rules to JS File feature is used to create a JavaScript resource that is used to store Maestro rules in a JavaScript file.

1. Open a form that has multiple business rules configured
2. Click the Code button
3. Click *Move Rules to JS File*
   Clicking this button will open the Move Rules to JS File dialog. The screenshot below displays this dialog.

   This dialog displays all the business rules that have been configured for the selected form. Each rule is identified and if selected, the script that creates the rule will be displayed (in JavaScript) in the code box below the list of rules.

   If you have rules in your form that are only 1 line of code, a checkbox will appear that you can select to display these one line rules.



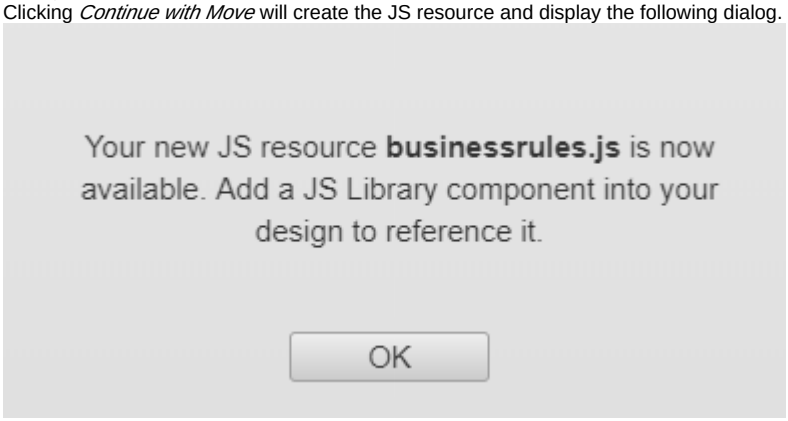
4. Provide a Resource File Name for the JavaScript file
   The Resource File Name is the name of the JavaScript (JS) file that will be created when you move the selected rules.

5. Provide a Namespace for the JS File
   The Namespace is the unique name that will be used to group the rules in the JS file.

6. Select the rules that you want to move to a JavaScript File
   You can click the *Select All* button to select all the available rules.

7. Click *Preview*
   Clicking Preview will display a preview of the JS file. The screenshot below displays an example of a preview for a JS file.

8. Click *Continue with Move*
   If you want to return to the previous dialog and select/deselect other rules, click the Back to Config button.
   Clicking *Continue with Move* will create the JS resource and display the following dialog.



9. Click *OK*

Once the Maestro rules have been moved to the JS file, you can then reference the created JS file directly in the form that you just used to create the JS file (using a JavaScript Library component), and then download the JS file and edit it using a SCM tool before uploading the JS file back into Maestro.

Referencing the JS File into a Form

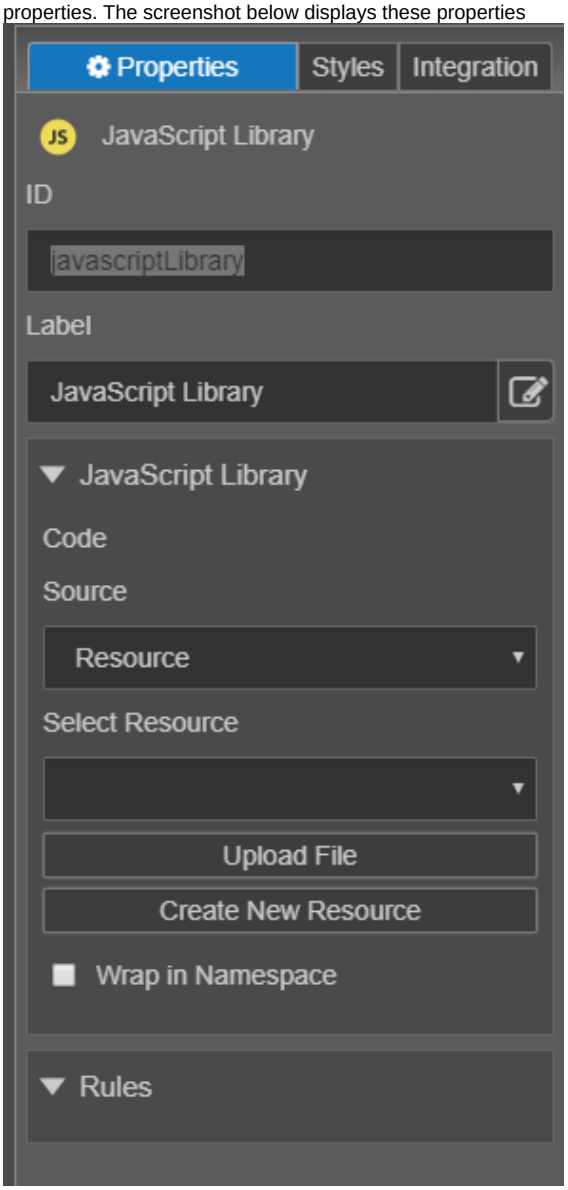
Once the rules of a Maestro form have been moved to a JS file, it is time to reference the JS file back into the form. It is important to understand that this step is essential, as when you *Move Rules to a JS File,* the business rules will no longer be a part of the form, but rather they are ONLY associated with the (now) external JavaScript file.

To reference the JS file into a form:

1. Open the form that you moved the JS rules from
2. Use the Maestro Palette and select the JavaScript Library component
   The screenshot below displays the JavaScript Library component in the Maestro Palette.

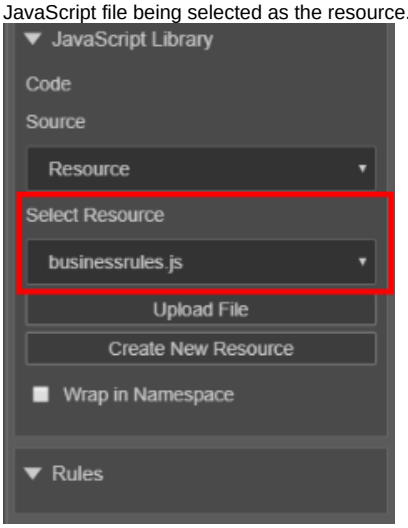
3. Drag and drop the JavaScript Library component onto the form
   Dropping the JavaScript Library component onto the form will display the JavaScript library

properties. The screenshot below displays these properties



4. Select the Source of the JavaScript file
   For a JS File that is created using the *Move Rules to JS File* feature, the source will always be, **Re source** (which is selected by default).

5. Select the Resource from the Resource dropdown menu
   This dropdown list will identify each of the JS files/resources that have been created for the form. The screenshot below highlights the Select Resource dropdown and displays the *businessrules.js* JavaScript file being selected as the resource.



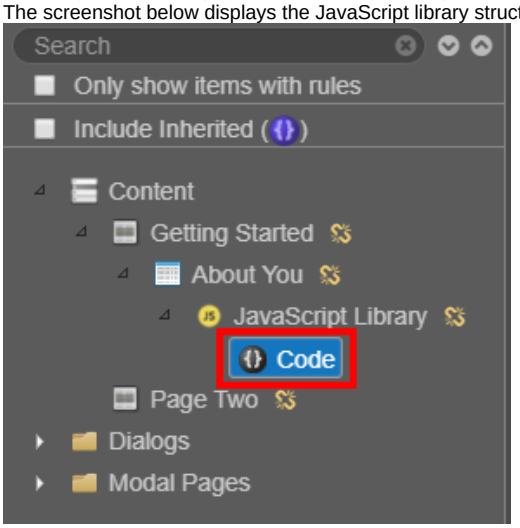
6. Save the Form
   Saving the form will apply the business rules from the JavaScript file to the form.

Downloading and Uploading the JS File

Once the rules of a Maestro form have been moved to a JavaScript file and referenced in a form, you have the option of downloading the JavaScript file and editing it in a tool outside of Maestro. After editing the form outside of Maestro (in an IDE), you can then upload the JavaScript file back into Maestro before referencing it again in the Maestro form.

To download a JS File:

1. Switch to Code View
2. Expand the tree to display the JavaScript library and select Code
   Selecting Code will display the code that makes up the JavaScript file.
   The screenshot below displays the JavaScript library structure.



3. Click Download
   Clicking Download will download the JavaScript file to your computer.
   The screenshot below highlights the Download link.

After downloading the JavaScript file, you can then open it in any IDE (integrated development environment) or text editor and edit the rules in any way that you like. Once you have made your edits, you can follow the above steps again but once you get to Step 3, click **Upload** instead of Download. The screenshot below highlights the *Upload* link.

Debugging Rules

Unknown macro: 'redirect'

Debugging Rules

Chrome developer tools include many useful extensions for debugging Maestro applications.

One of them is the debugger which allows you to set breakpoints on any JavaScript code in the app, including code you have written in rules.

A quick way to trigger the debugger from inside your own rule code is to include the 'debugger;' statement in your code as you are developing it. When the statement is executed in a rule, the debugger will be activated and you can then inspect the values of variables in the debugger, or in the Console tab. Be sure to remove any debugger statements when their job is done.

For more advanced debugging techniques, you can view the Advanced Debugging of Maestro Forms article.

Form Options

Unknown macro: 'redirect'

The content within this section covers information relating to form options in Maestro.

The list below identifies the topics covered within this section.

- Form Options Overview
- Form Options > Basic Options

Form Options Overview

Unknown macro: 'redirect'

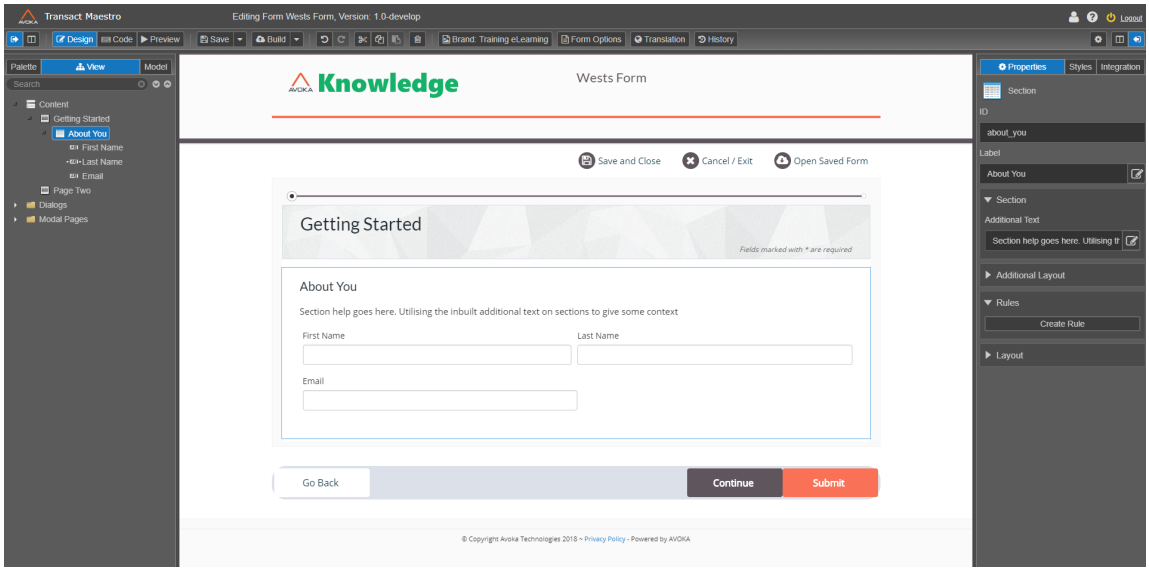
Related Pages:

- Form Options > Basic Options
- Form Options Overview

Form Options Overview

The areas that you can access or edit in your form will depend on the access granted by the template designer at the template level. Limiting the access in the form allows the form builder to focus on the areas that they need to manage. Template designers use extension points to control the structure of the form that the form builder has access to. For more information on extension points, see the Modifying Extension Points topic in the Maestro Templates section.

Looking at the View Panel of the form pictured below, the form builder only has access to the Content, Dialogs and Modal Pages. Dialogs and Modal Pages will always be displayed, even if the form builder does not have permission to make any changes to existing dialogs and modal pages.



There may be times when the form builder needs to remove the navigator from the form or disable a function bar option. For example, if it is a short one or two page form, the navigator may not be necessary. Or there may be forms where the form builder does not want the user to save the form and therefore they want to remove the Save and Close button from the function bar.

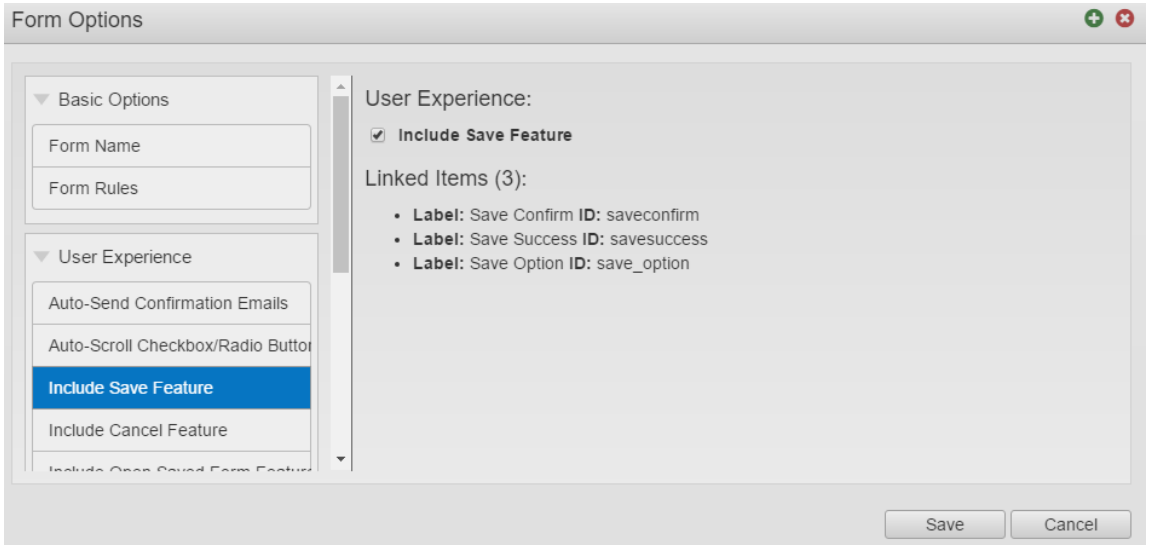
When the template designer locks down the form, it is likely that they will still give the form builder access to basic form functionality, such as the examples mentioned above. In this case, the form builder can use the Form Options button to make the aforementioned configuration changes.

Selecting the Form Options button will display the Form Options window.

In the window you will see various categories available. The categories displayed in the image above may be different to the options you have available in your forms. Maestro is highly customizable and therefore the options the template designer selects may be different depending on the template and the business preferences for the forms.

To make a change to a form option you can expand the relevant category, then select the option you want to view.

The image below shows the Include Save Feature. Continuing the example above, the form builder could deselect Include Save Feature. This would hide the Save and Close button, and disable all other save features for the form.



If an option that you require is not available in the window speak to the template designer about adding it to the options. For more information on form options the Maestro Templates section.

Form Options > Basic Options

Unknown macro: 'redirect'

Related Pages:

- Form Options > Basic Options
- Form Options Overview

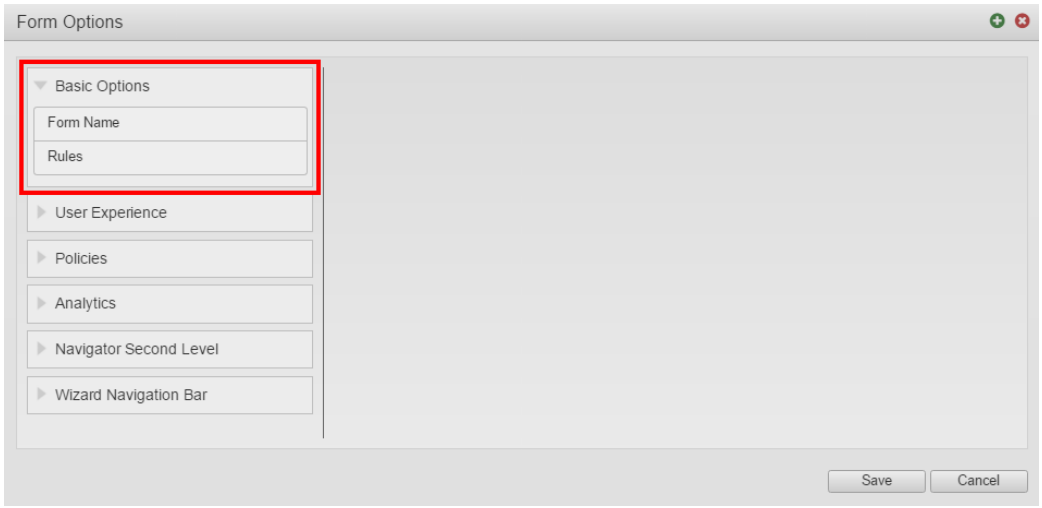
The options displayed in the Form Options window may be different to the options you have available in your form.

Page Contents:

Overview

To access the Form Options window click the Form Options button on the toolbar.

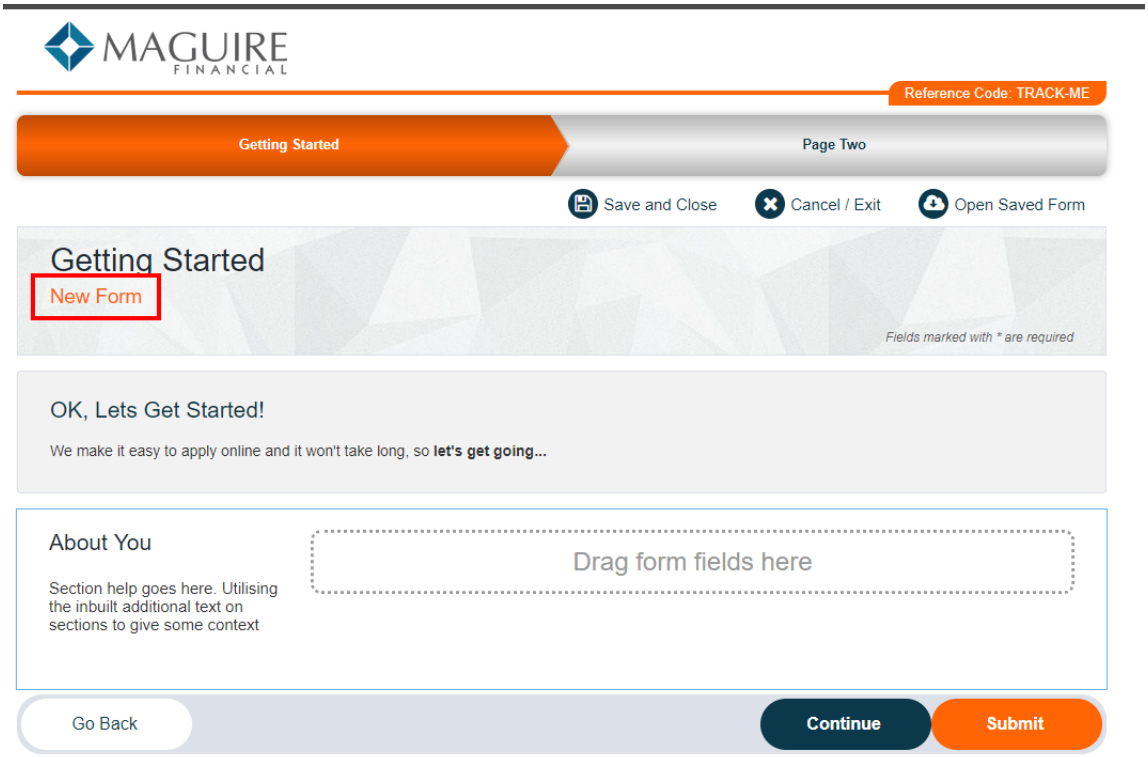
In the Form Options window you will find the Basic Options section. The items available in this section usually include Form Name and Rules.



Form Name

Form Name allows you to change the form name metadata. Changing the form name in the Form Options window will only impact areas on the form that reference the form name metadata.

In the default template the form name displays near the top of the form. This display may be different in your form depending on the template used.



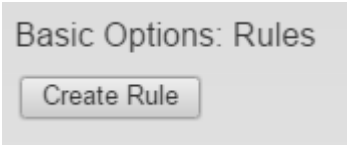
Changing the Form Name through the Form Options window will *not* change the form name displayed in the Management Dashboard. To change the form name on the dashboard you will need to edit the form details directly in the Management Dashboard.

Likewise editing the form name in the Management Dashboard will not change the form name displayed/referenced in the form. You will need to use the Form Options window (as noted above) for those changes.

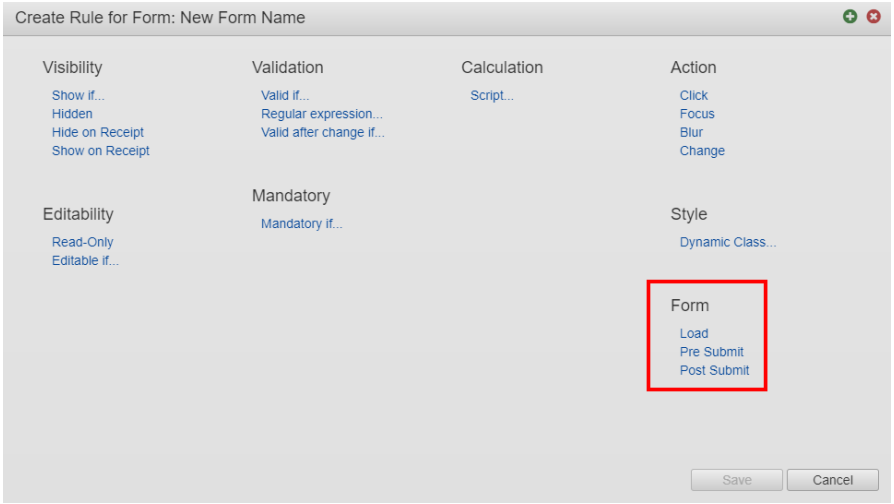
Rules

To create a form rule you will need to access the Create Rule window through the Form Options window.

Selecting Rules in the Form Options window will display the Create Rule button. Clicking this button will open the Rules window.



This will open the usual Create Rule window, however the Form rules will be displayed.



Background Save

Unknown macro: 'redirect'

Page Contents:

Related Pages:

- Background Save
- Language Translator in Maestro
- Language Translator in Transact Manager
- Save Challenge (Save and Resume)

Overview

Background save automatically saves the form to the Transact Manager server, without the user having to explicitly click the Save button. When background save is implemented, the form will be saved each time the user moves from one page to another page in the form.

Background save allows a user to return to a form even if something goes wrong while they are entering data. For example, the user is completing a 10-page form and they get to page 8 and all of a sudden their computer locks up and closes the browser window. With background save implemented, this data can be recovered using the form reference code, and the user can return to page 8 of the form.

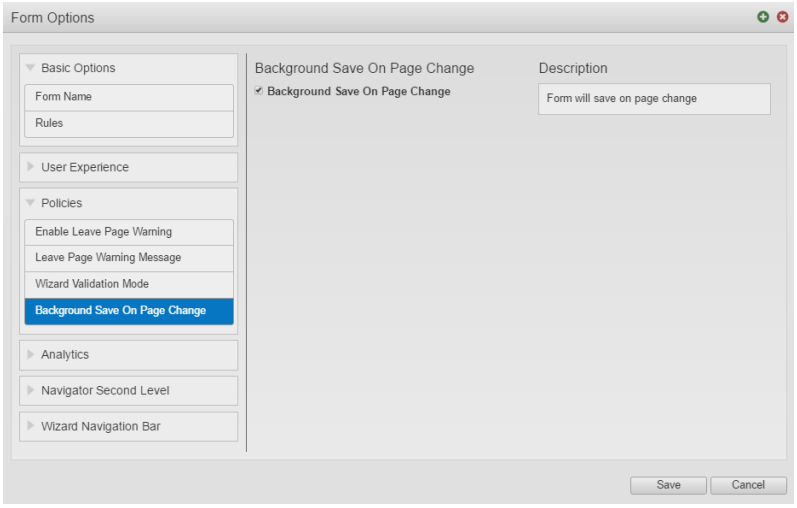
Though the user may lose the data of the page they were currently on when the computer locked up, redoing one page of data is much better than having to redo all 7 of the previous pages.

If the user did not record their reference code, operators of the call center portal can provide a form link to them.

To enable background save, use the Form Options window or select Content from the View Panel and use the Properties panel of the Maestro editor.

Enable Background Save using Form Options

1. Select the *Form Options* button
2. Expand *Policies*
3. Select *Background Save on Page Change*
4. Select the Background Save on Page Change checkbox
5. Click *Save*

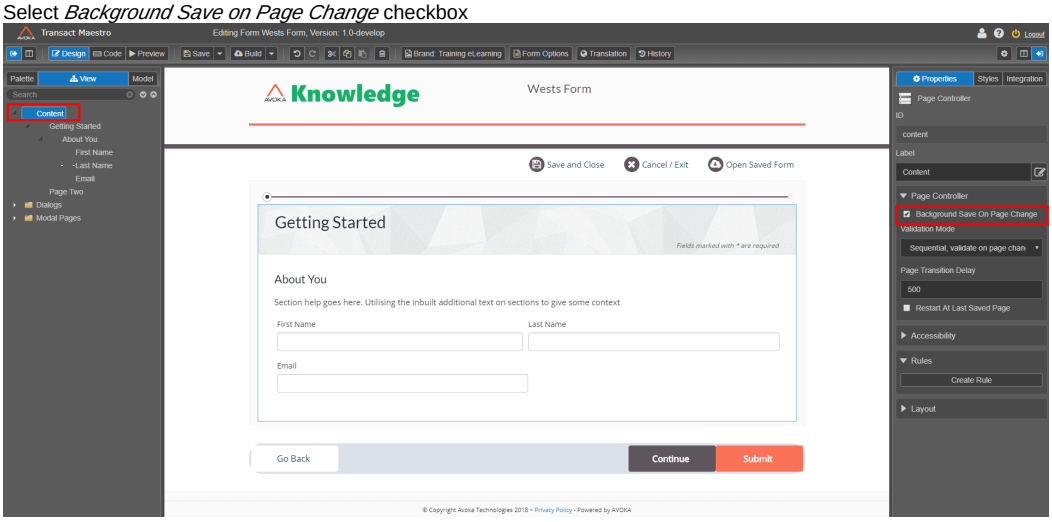


The form will now save each time the user changes to a different page. This feature is exposed through Form Options in the standard Maguire template, although your template designer may or may not have exposed this capability to your forms.

Enable Background Save using the Content Container

1. Select *Content* from the View Panel
2. On the Properties pane, expand *Page Controller*

3. Select *Background Save on Page Change* checkbox

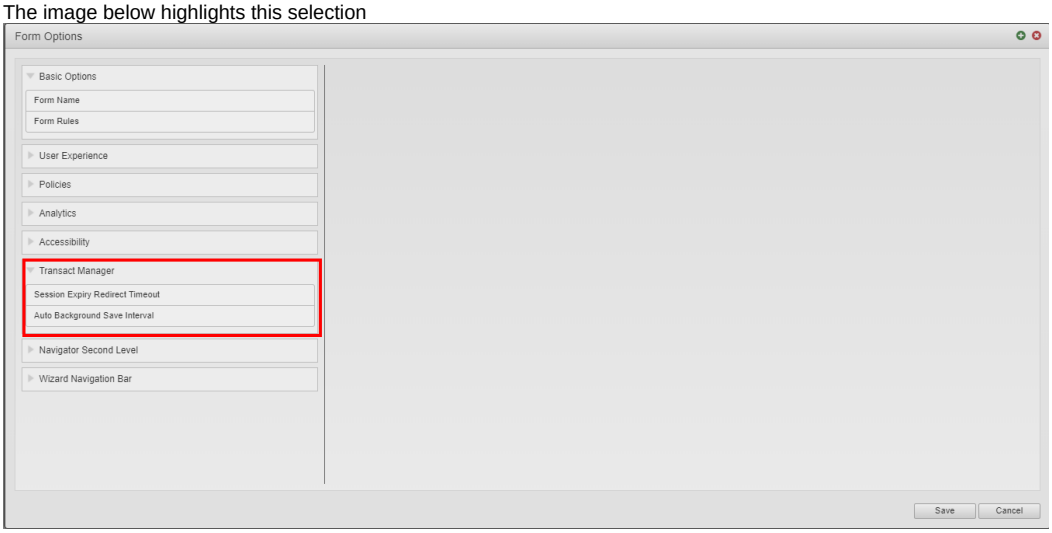


Background Save Interval

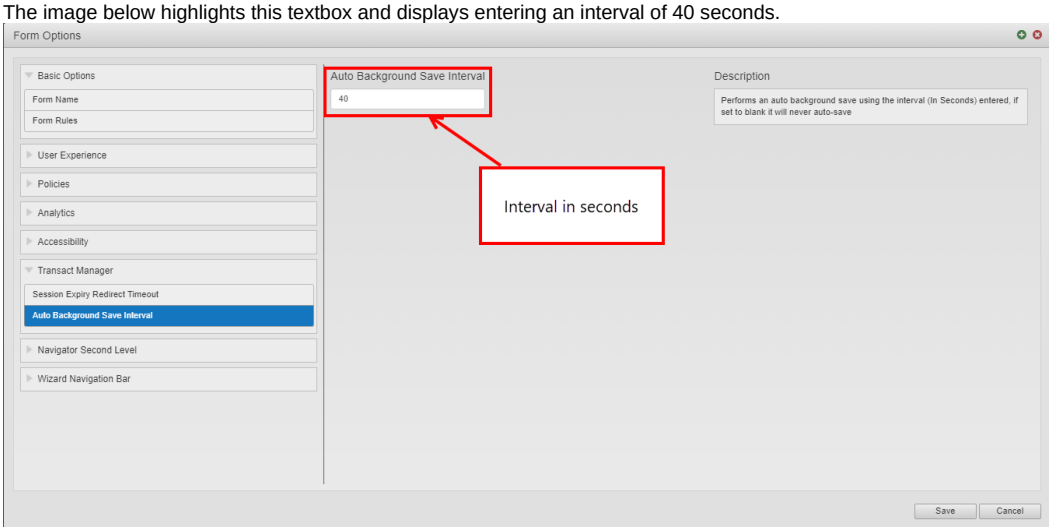
As well as configuring forms to be saved each time a user moves from one page to another, you can also build Maestro forms to be saved at defined intervals. Implementing background save intervals means that the form will be saved at defined intervals of seconds. For example, you may want a form to be saved every 40 seconds. Background save intervals can be extremely useful when you are building forms that are quite long.

To configure background save intervals:

1. Navigate to *Form Options* from the Maestro editor
2. Select and expand *Transact Manager*
   The image below highlights this selection



3. *Select Auto Background Save Interval*
4. *Enter* a numerical value into the *Auto Background Save Interval* textbox (eg 40) OR use the arrow buttons in the textbox to select the interval in seconds that the form will be saved.
   The image below highlights this textbox and displays entering an interval of 40 seconds.



5. Click *Save*

Save Challenge (Save and Resume)

Unknown macro: 'redirect'

Page Contents:

elated Pages:

- Background Save
- Language Translator in Maestro
- Language Translator in Transact Manager
- Save Challenge (Save and Resume)

Overview

The Save Challenge feature allows users to save a form they have been working on, and open it later to finish it from where they left off.

When the user saves a form, they are given an automatically generated reference code. This reference code is defined in Transact Manager and is needed in order to open the form. Along with the reference code, the form builder can also include a security question. Using the reference code, along with a security question, creates two factor authentication.

If implemented, when saving a form, a form user will be prompted to answer the security question. If answered correctly (ie the form user provides the correct details), the form user will be able to save the form with their data still inplace when they return to the form. When returning to the form, the form user will be prompted to answer the security question again, and provide the reference code that they were given when they initially saved the form.

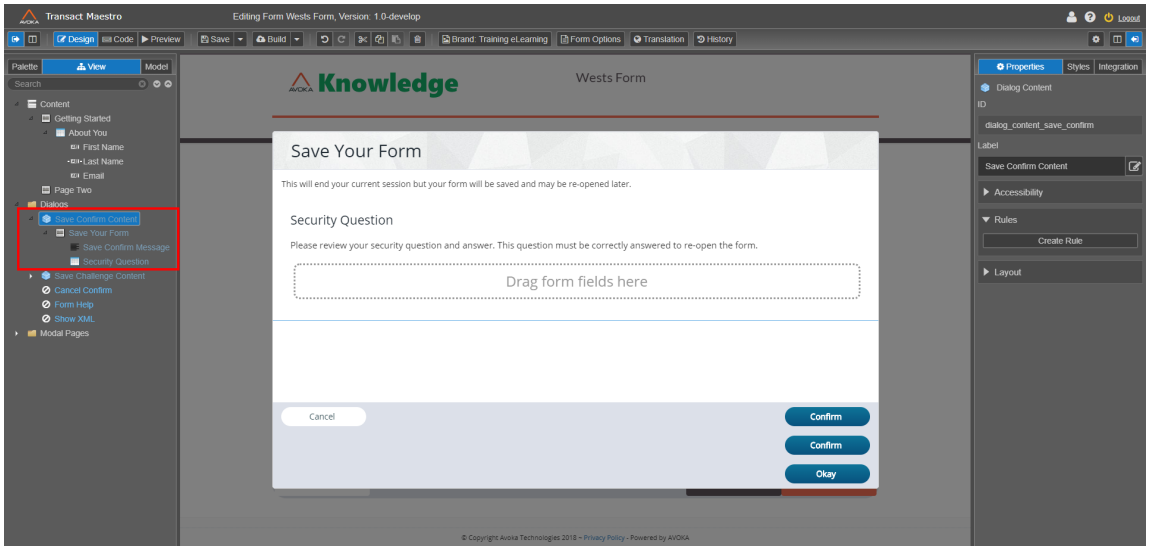
Any component type can be used for the security question (commonly a text field).

When the user saves their form, they will be presented with the Save Confirm dialog. When the user returns to resume their form, they will be presented with the Save Challenge dialog.

Configuring the Save Confirm Dialog

In order for the Save Confirm dialog to function correcty, extension points must be configured in the template of the form. More information on extension points can be found in the Modifying Extension Points section of the documentation.

It is from the Save confirm Dialog settings that you configure the security question.



Setting up the security question

Follow the steps below to setup a security question in a Maestro form.

1. Expand Dialogs
2. Select Save Confirm
3. Add a component to the security question section within the dialog
4. Map the component as the save challenge
    a. From the Data pane expand Transact Integration
    b. Select Save Challenge from the Form Data Config Mapping dropdown

You can only add one security question to the Save Confirm dialog. However, you can use a calculated data field to configure and use multiple components for the security question. Please note that when using multiple components/questions to create the security question/answer, ensure that the calculated field that combines the multiple questions, is the first field on the Challenge Dialog (alternatively you can update/override the script).

Save Challenge Dialog

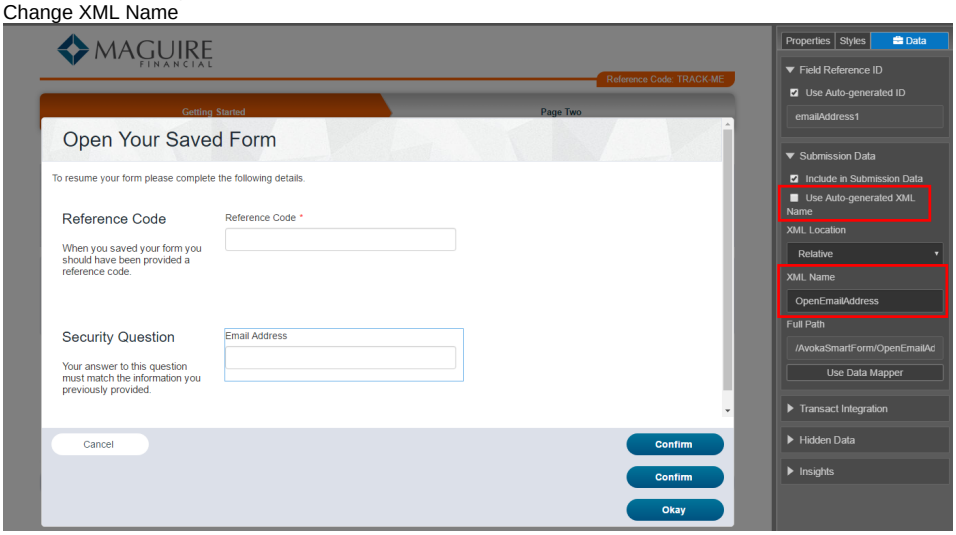
The Save Challenge Dialog needs to be set up with the same security question as the Save Confirm Dialog. The Save Challenge Dialog will be displayed when the user returns to complete a saved form.

The Save Challenge includes a section to enter the generated reference code that displays when the user saves their form. It also includes the security question section. This is where the user needs to enter the same answer entered when the form was saved.

When configuring the components within the two dialogs, you will need to ensure that the XML Name is different. By default, if the label is the same, the XML Name will also be the same. This will cause a duplicate binding error when you publish your form. You can decide to only change one component XML Name or change both component XML Names.

Setting up the security question:

1. Expand Dialogs
2. Select Save Challenge
3. Copy the component using the Copy button on the toolbar or CTRL + C
4. Paste the component within the Save Challenge security question section (the Save Challenge security question does not need to be mapped to the Save Challenge System Mapping. You only need to map the component displayed on the Save Confirm dialog).
    - Expand Save Challenge
    - Expand Save Challenge Content
    - Select Security Question
5. Change XML Name
    a. Select the Data tab
    b. Uncheck Use Auto-generated XML
    c. Change XML Name



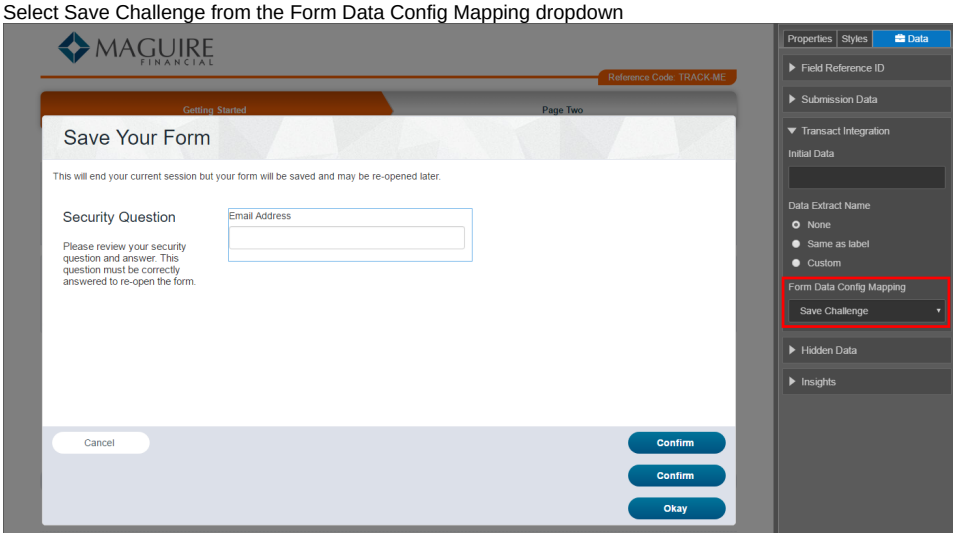
Populating Security Question from User Entered Data

Another option for the security question is to populate the answer from a component the user has already completed within the form.

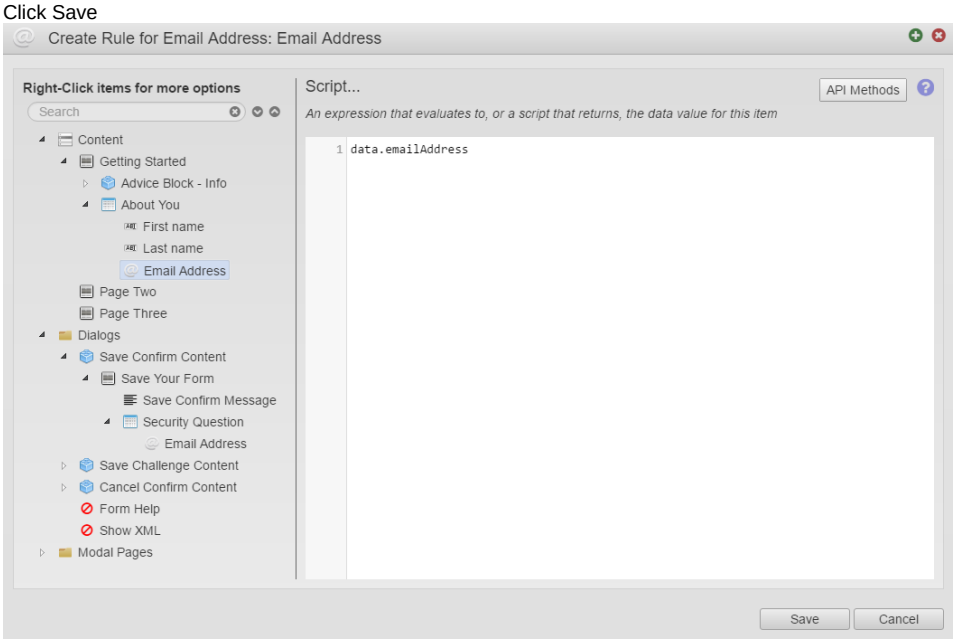
For example, if the answer to the security question is the user's last name, you have probably already asked the user for their last name within the form. You can calculate the field in this dialog to be equal to the same data they have already typed in the form. This makes the save process easier for the user.

1. Expand **Dialogs**
2. Select **Save Confirm**
3. **Add** a component to the security question section within the dialog
   You can copy the component on the form and paste it to the Save Confirm dialog if preferred or you can drag a component from the Palette pane to the Security Question section of the dialog.
4. **Map** the component as the save challenge
    a. From the Data pane expand Transact Integration

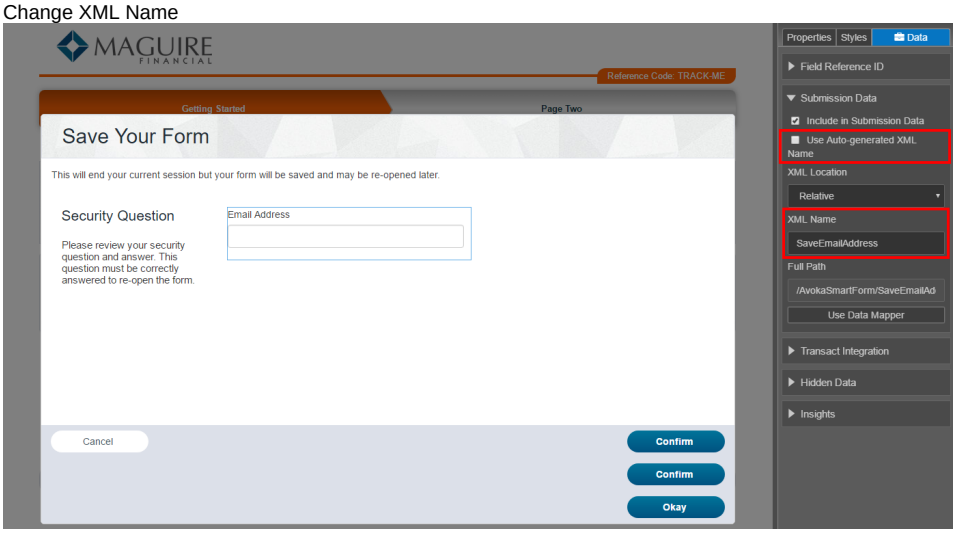
b. Select Save Challenge from the Form Data Config Mapping dropdown



5. Create **calculation**
   a. With the component selected click Create Rule on the Properties pane
   b. Select Script...
   c. Navigate to the existing component within the form and double-click to add it to the calculation
   d. Click Save



6. Change **XML Name**
   a. Select the Data tab
   b. Uncheck Use Auto-generated XML
   c. Change XML Name

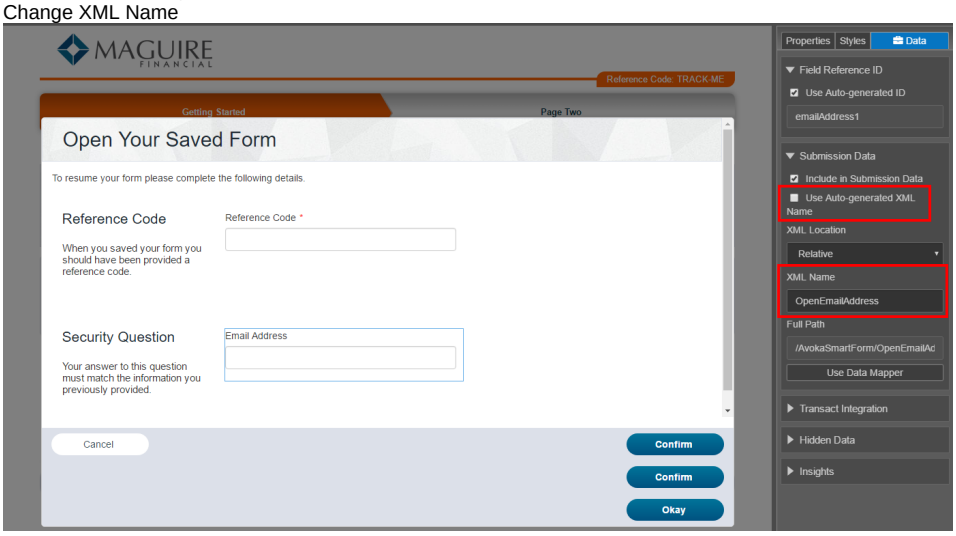


Remember that you cannot have multiple components with the same XML Name. In this scenario you will end up with three components that could have the same XML Name - the

component on the form, the component on the Save Confirm dialog and the component on the Save Challenge dialog. You need to ensure that all three components have different XML Names. In most cases you will likely change the components within the dialogs and leave the component on the form as an auto-generated XML name.

Remember you need to add the security question component to the Save Challenge dialog. Rather than recreating the component along with the calculation rule you may find it easier to copy and paste the component from the Save Confirm dialog to the Save Challenge dialog.

1. Select the **Save Confirm** security question
2. **Copy** the component using the Copy button on the toolbar or CTRL + C
3. **Paste** the component within the Save Challenge security question section
    a. Expand Save Challenge
    b. Expand Dialog Content
    c. Select Security Question
    d. Click the Paste button on the toolbar or CTRL + V
4. Change **XML Name**
    a. Select the Data tab
    b. Uncheck Use Auto-generated XML
    c. Change XML Name



Importing to Transact Manager

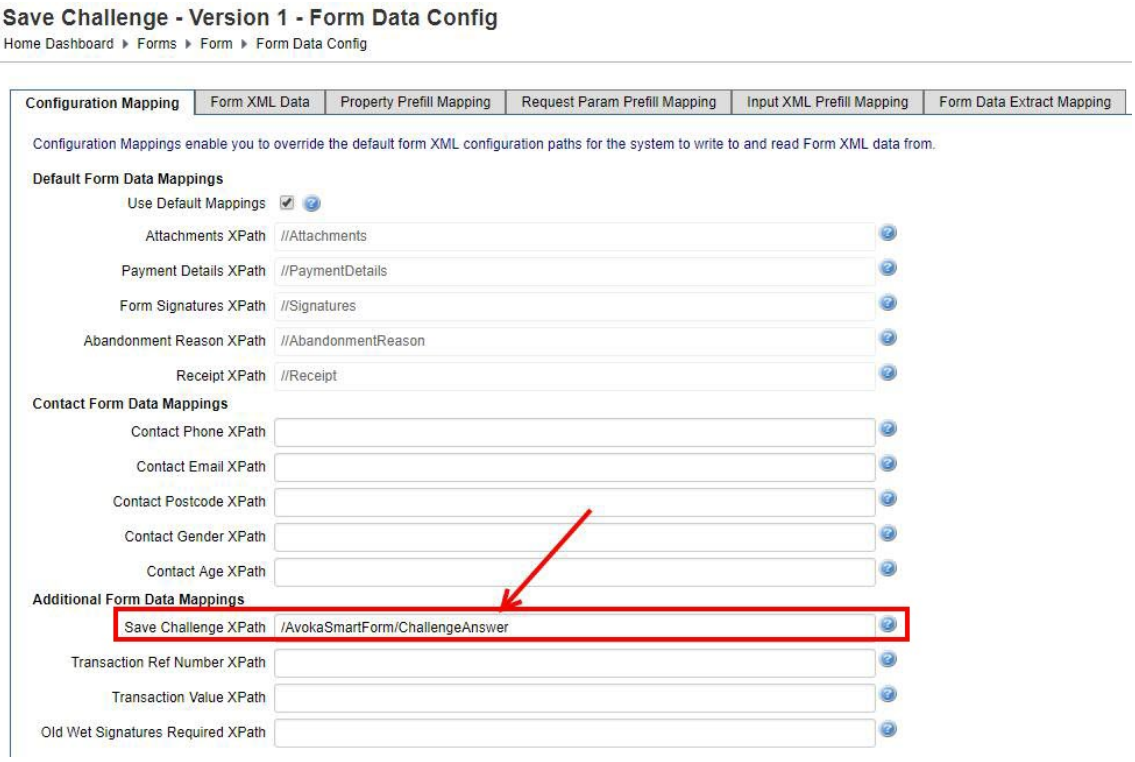
Once the Save confirm and Save Challenge dialogs have been configured, it is recommended that you export the form from Maestro and Import it into Transact Manager (there are a number of methods to do this and they are covered in the Transact Manager Documentation).

After you have imported the form into Transact Manager, you need to ensure that the Data Config properties are set correctly for the form in Transact Manager (Information on Data Configurations can be found in the Transact Manager Documentation).

The screenshot below displays and highlights a correctly configured Save Challenge XPath configured in the Data Config options in Transact Manager. To get to this screen, follow the steps below.

To set up your form version's configuration mappings:

1. Log in to Transact Manager
2. Navigate to *Forms* from the TM menu bar, and then click *Forms* from the dropdown menu.
3. Select the form that you want to configure Save challenge XPath and click the edit icon.
4. In the *Form Versions* section of the Form Dashboard, find the form version and click the *Data Config* link
5. The *Configuration Mapping* tab will be displayed (if not, switch to it).
6. Configure the options, as shown in the screenshot below
7. Click the *Save* button



- Create Save Challenge Using Multiple Components

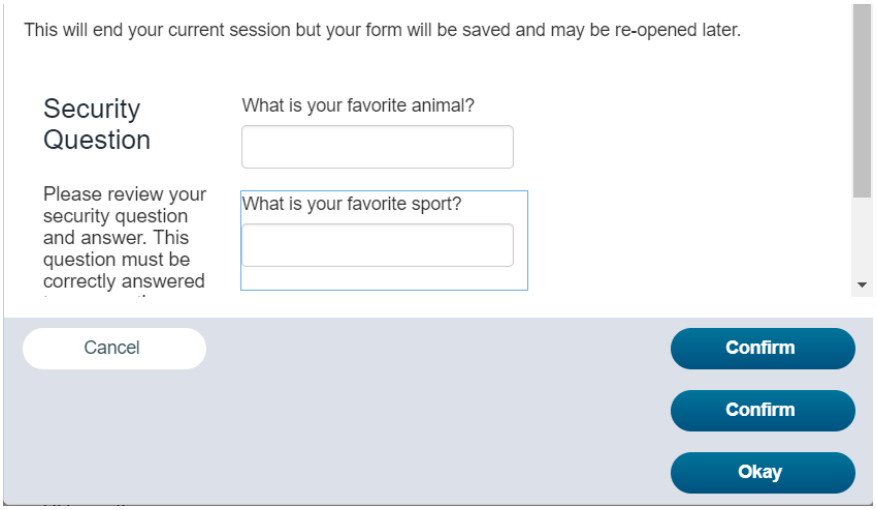
Create Save Challenge Using Multiple Components

Unknown macro: 'redirect'

Overview

The Save Challenge feature allows users to save a form they have been working on, and open it later to finish it from where they left off. Though the Save Challenge is commonly implemented as a one question and one answer method, it is also possible to configure multiple questions using multiple Maestro components.

The screenshot below displays an example of a form asking two security questions.

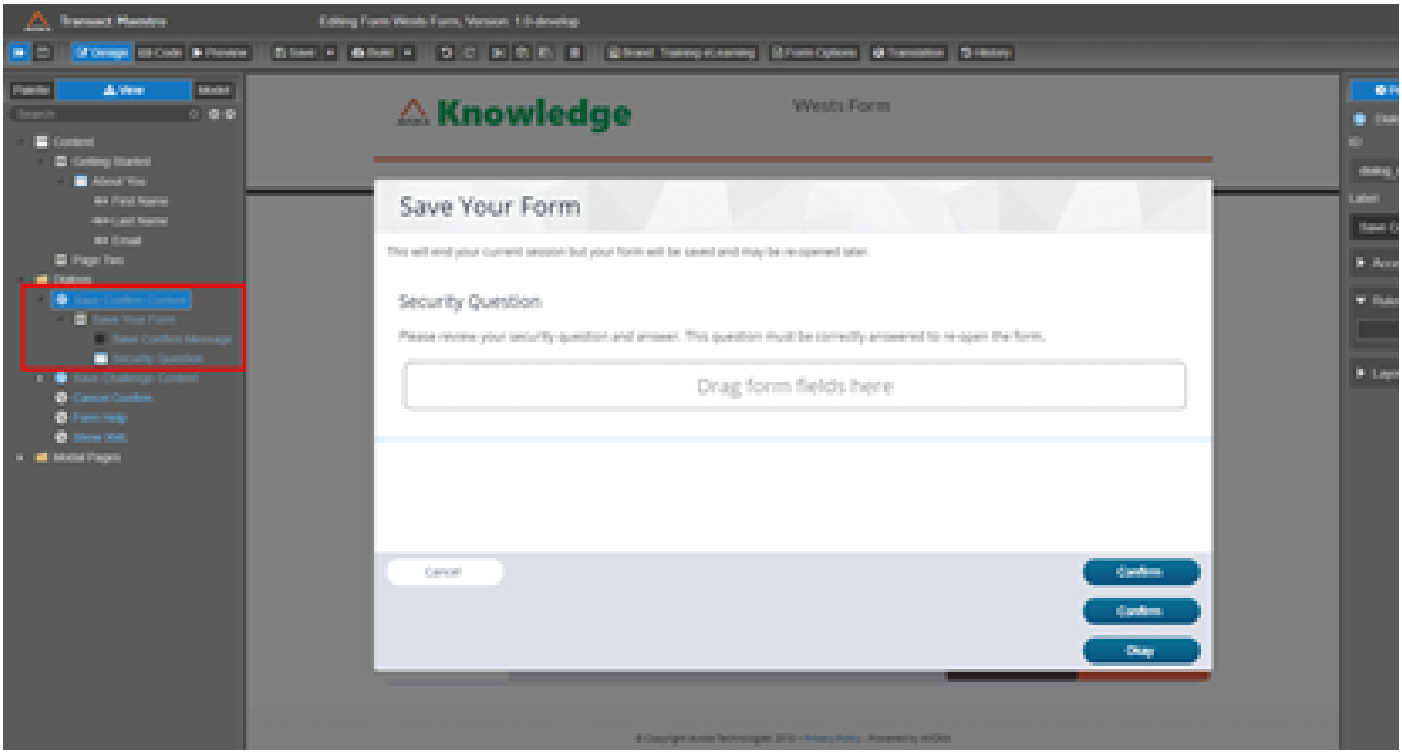


The first step in configuring the Save Challenge with multiple questions is configuring the Save Confirm Dialog.

Configuring the Save Confirm Dialog

The Save Confirm Dialog is the screen that the form user will see when they select, 'Save and Close' when completing a form. This screen is used to collect the answers to the configured security challenge questions. In order for the Save Confirm Dialog to function correctly, extension points must be configured in the template of the form. More information on extension points can be found in the Modifying Extension Points section of the documentation.

It is from the Save Confirm Dialog settings that you configure the security questions (i.e the questions that will be presented when the user saves the form).

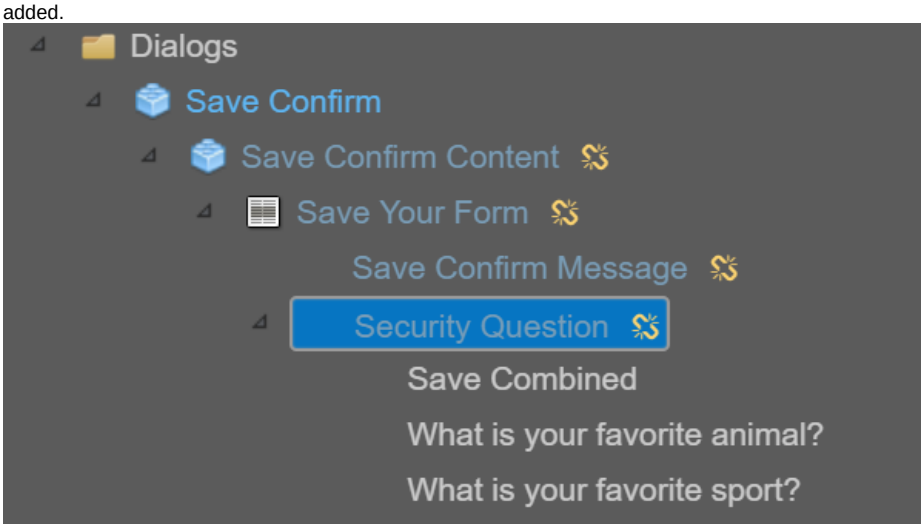


Setting up security questions in the Save Confirm Dialog

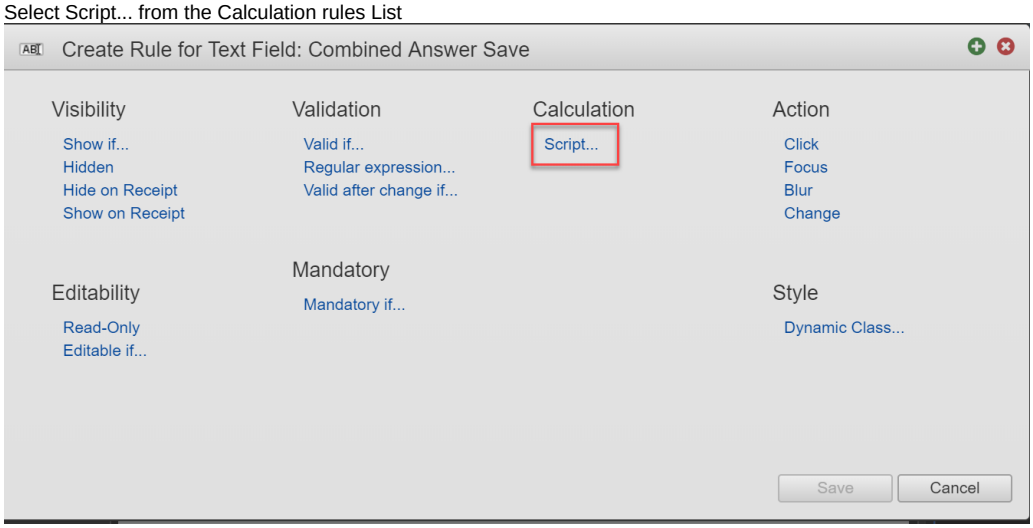
Setting up multiple security questions in the Save Confirm Dialog is almost identical to setting up a single security question Save Challenge.

Follow the steps below to set up multiple security questions in the Save Confirm Dialog.

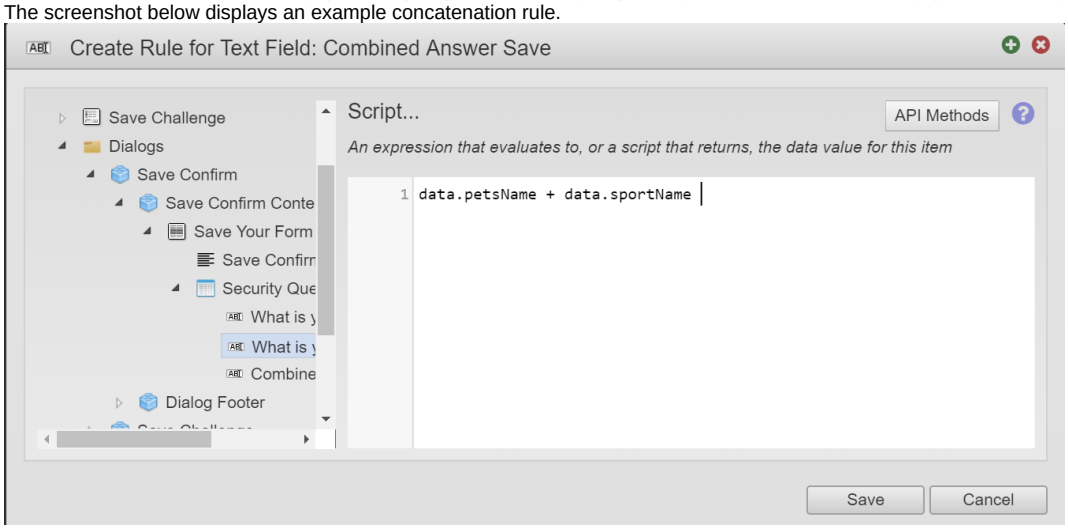
1. Expand Dialogs from the tree
2. Select Save Confirm > Save Confirm Content > Save Your Form > Security Question
3. Add multiple components to the security question section within the Dialogs section (for example you could add multiple text fields and use the labels as questions)
4. Add a separate data field to the security question section (name the text field something like *Save Combined).* This data field will be the hidden field that will collect the answers to all the security questions. The screenshot below displays the tree once the data field and text fields have been added.



5. Create a calculation rule on this data field to concatenate the answers to the questions
   To create this calculation rule:
   - Select the data field Component
   - Navigate to the Properties tab and scroll down to Rules
   - Select Rules
   - Click Create Rule
   - Select Script... from the Calculation rules List



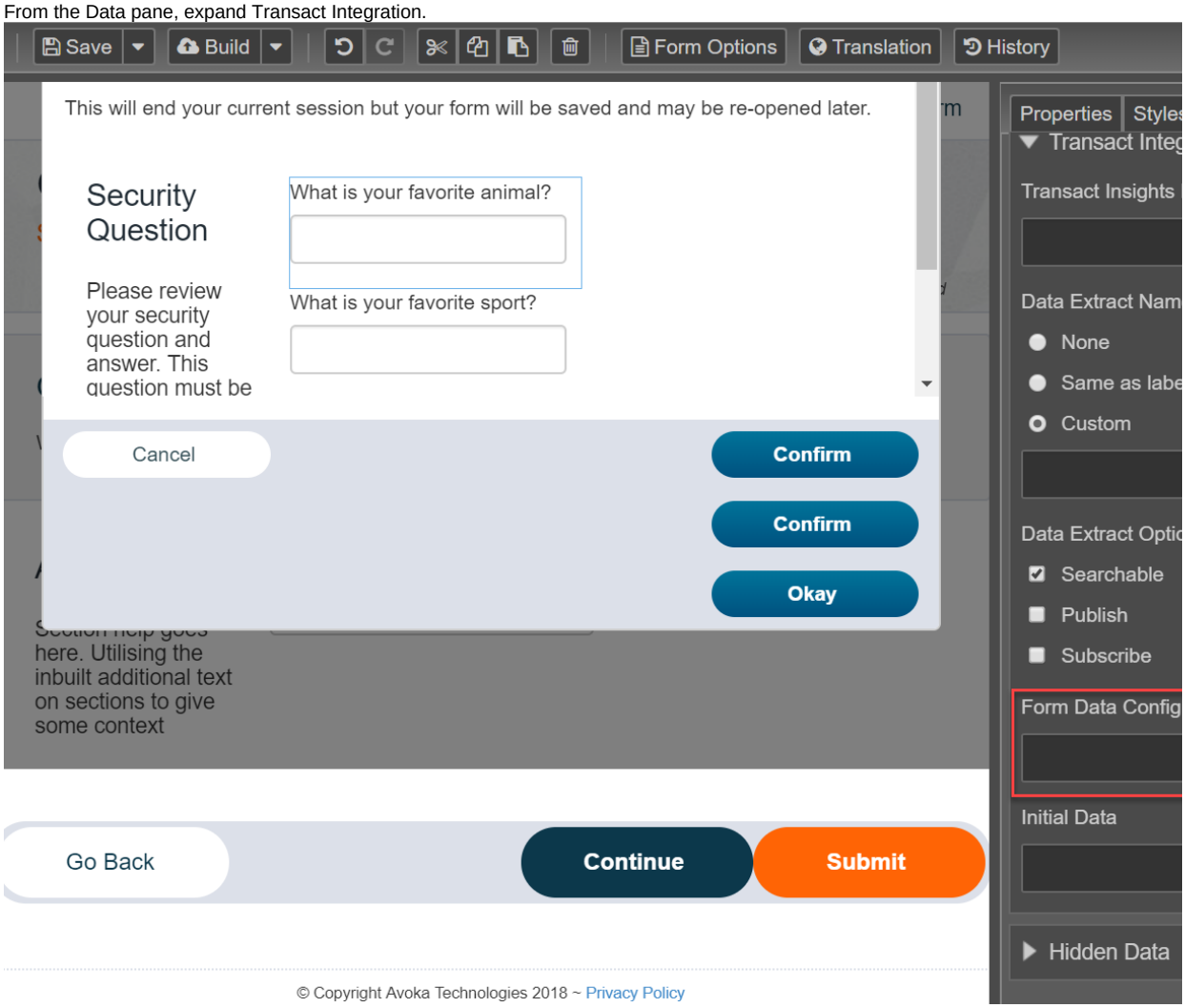
   - Create a concatenation calculation script by double-clicking the first security question and adding an addition (plus) sign and then double-clicking the second security question (this process will vary depending on the number of security questions that you implement). The screenshot below displays an example concatenation rule.



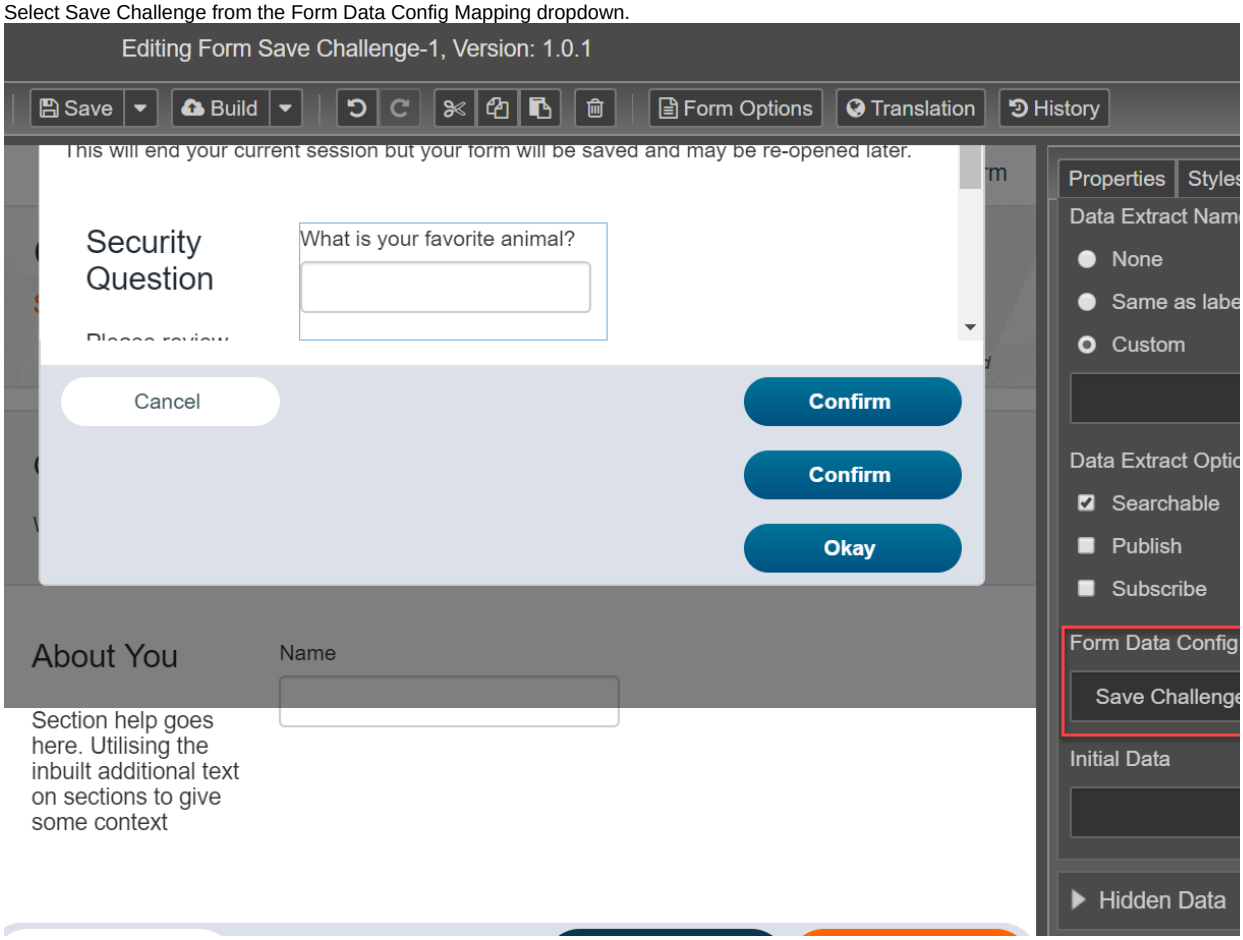
   - Click Save

7. Map the data field as the Save Challenge (by following the steps below). Mapping the data field ensures that TM knows what to search and compare when a form user answers each question.

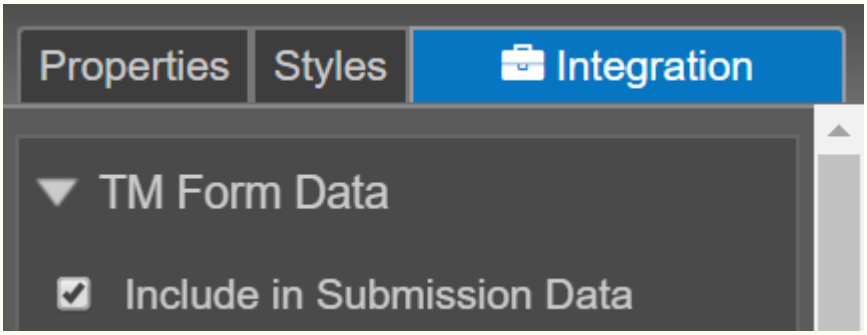
a. From the Data pane, expand Transact Integration.



b. Select Save Challenge from the Form Data Config Mapping dropdown.



Once you have mapped the data field to be the Save Challenge, ensure that the Include in Submission Data checkbox has been checked. This checkbox is found on the Integration Tab. The screenshot below displays the Include in Submission Data checkbox.



Once you have mapped the data field as the Save Challenge, you can move on to configuring the Save Challenge Dialog.

Save Challenge Dialog

Once the Save Confirm Dialog has been configured, the next step is setting up the Save Challenge Dialog.

The Save Challenge Dialog needs to be set up with the same security questions as the Save Confirm Dialog. The Save Challenge Dialog will be displayed when the user returns to complete a saved form. Technically, Maestro only allows one question and one answer to be configured for a Save Challenge but this can be overcome by using a calculation rule to combine the multiple answers (creating one concatenated answer).

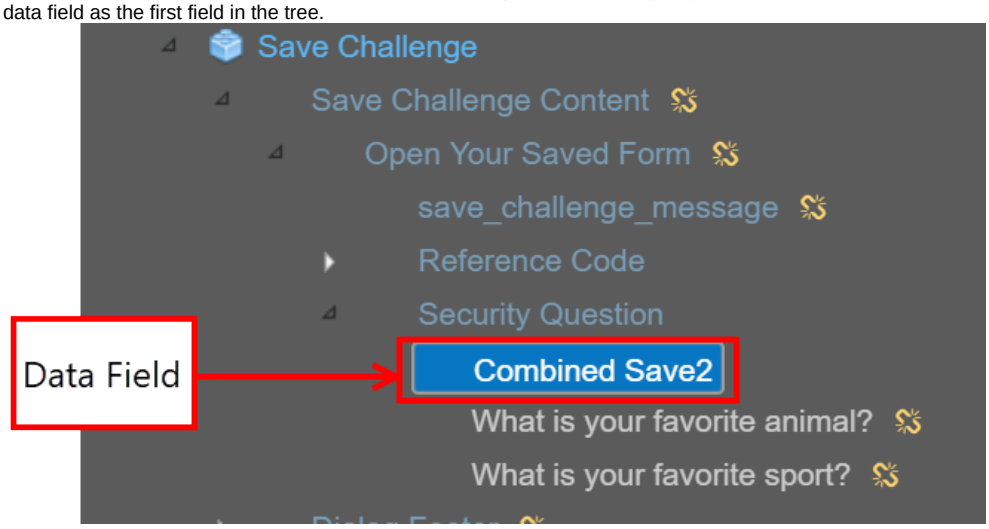
The Save Challenge Dialog usually includes a section to enter the generated reference code that displays when the user saves their form and a security question section displaying the configurated security questions. The Save Challenge Dialog is where the user needs to enter the same answers they entered when they originally saved the form.

When configuring the components within the two dialogs (Save Confirm and Save Challenge), you need to ensure that each XML Name for each component is unique (even though they are copies of the same questions). By default, if the label is the same, the XML Name will also be the same. Multiple components with the same XML name will cause a duplicate binding error when building the TM form version.

Setting up the security questions in the Save Challenge Dialog:

1. Expand Dialogs
2. Select Save Challenge > Save Challenge Content > Open Your Saved Form > Security Question

3. Copy the components you configured in the Save Confirm Dialog using the Copy button on the toolbar or CTRL + C (copy the question text fields, along with the hidden combined answers data field).
4. Paste the components into the Save Challenge security question section (the Save Challenge security question does not need to be mapped to the Save Challenge System Mapping).
   When copying these components, it is essential that the data field is the first field in the tree. The data field should be the first field as this will ensure that the data field is the field that TM will collect and compare with the originally entered answers. The screenshot below displays and highlights the data field as the first field in the tree.

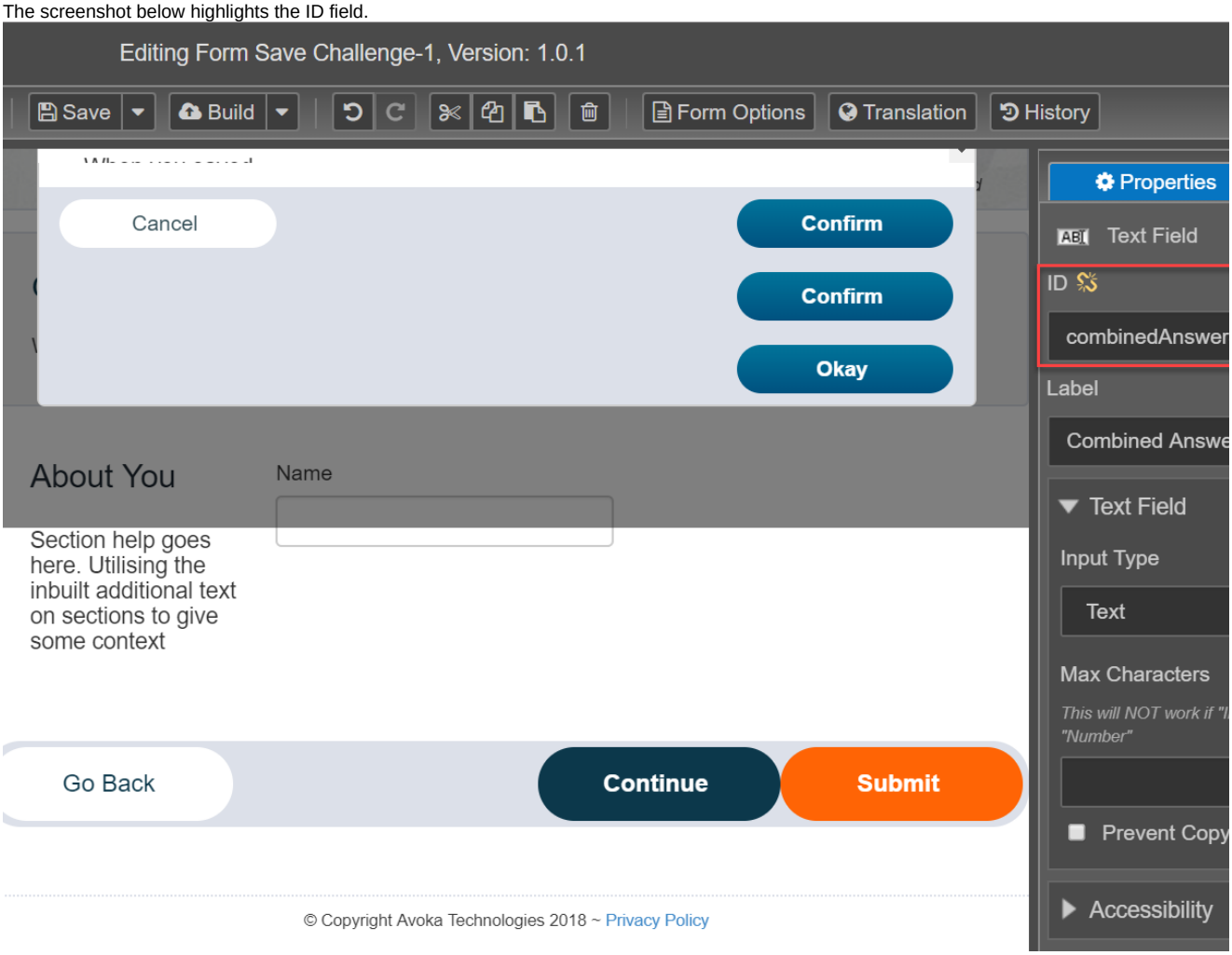


5. Change the IDs of the pasted text fields and data field

   To change the IDs of the components:

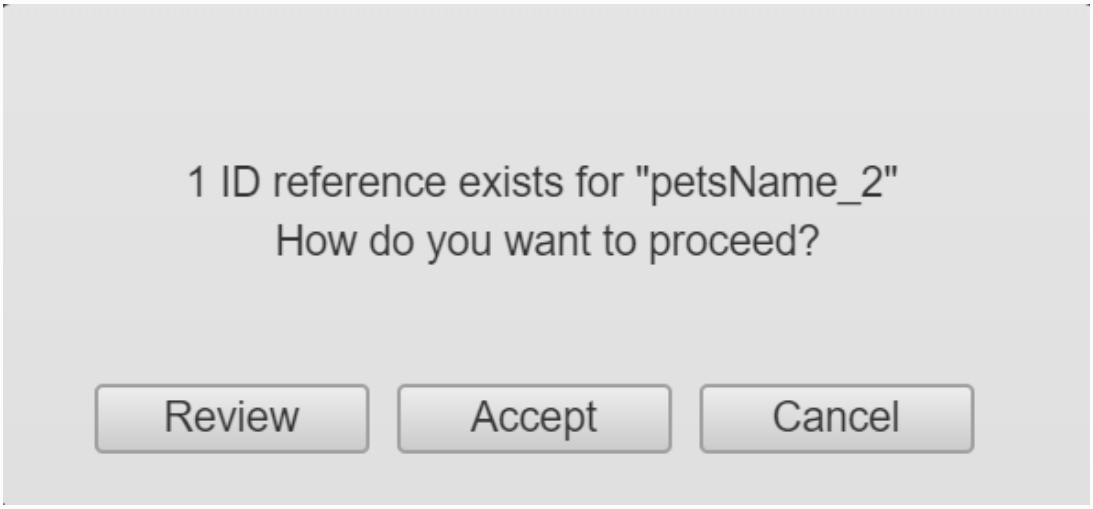
   - Select the component
   - Navigate to the Properties tab and change the ID in the ID field

   The screenshot below highlights the ID field.

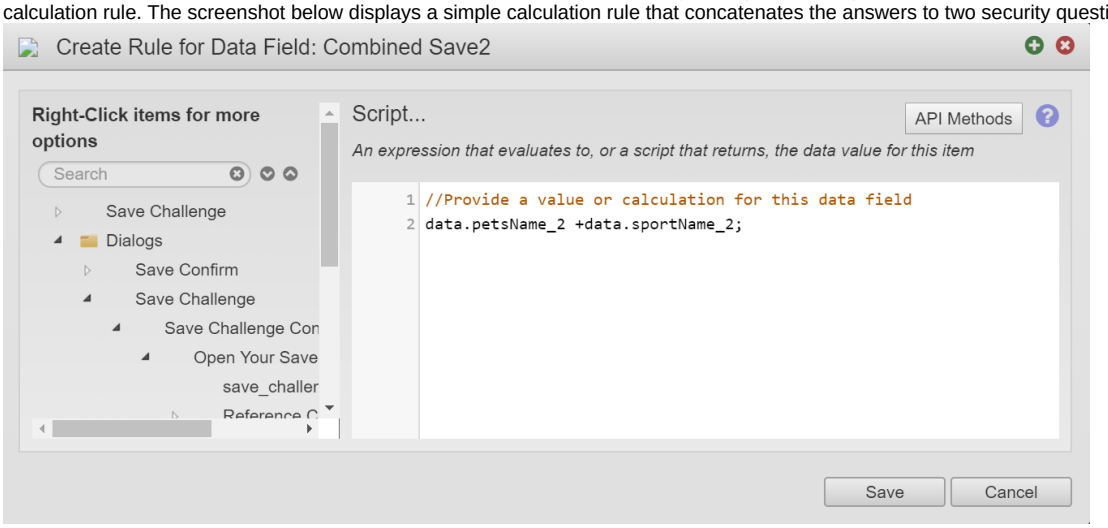


> It is recommended that you change the ID of all security question fields AND the ID of the data field.
> Changing the IDs will update the XML Name which will prevent any duplicate binding errors.

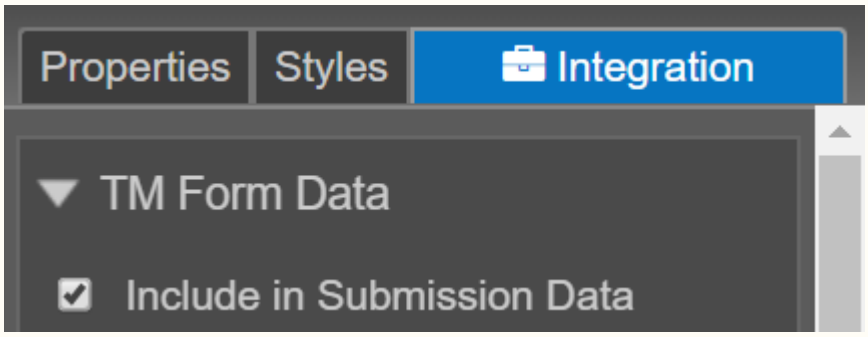
When you change an ID of a text field that is included in a calculation rule, Maestro will warn you that the ID you are attempting to change is currently referenced somewhere within the form. In this case, the one ID reference is coming from the calculation rule that was configured in the Save Confirm Dialog. The screenshot below displays this warning.



6. Ensure that the calculation rule that was copied from the Save Confirm dialog was updated to represent the new IDs. Do this by double-clicking the calculation rule. The screenshot below displays a simple calculation rule that concatenates the answers to two security questions.



Ensure that the Include in Submission Data has been checked. This checkbox is found on the Integration Tab. The screenshot below displays the Include in Submission Data checkbox.

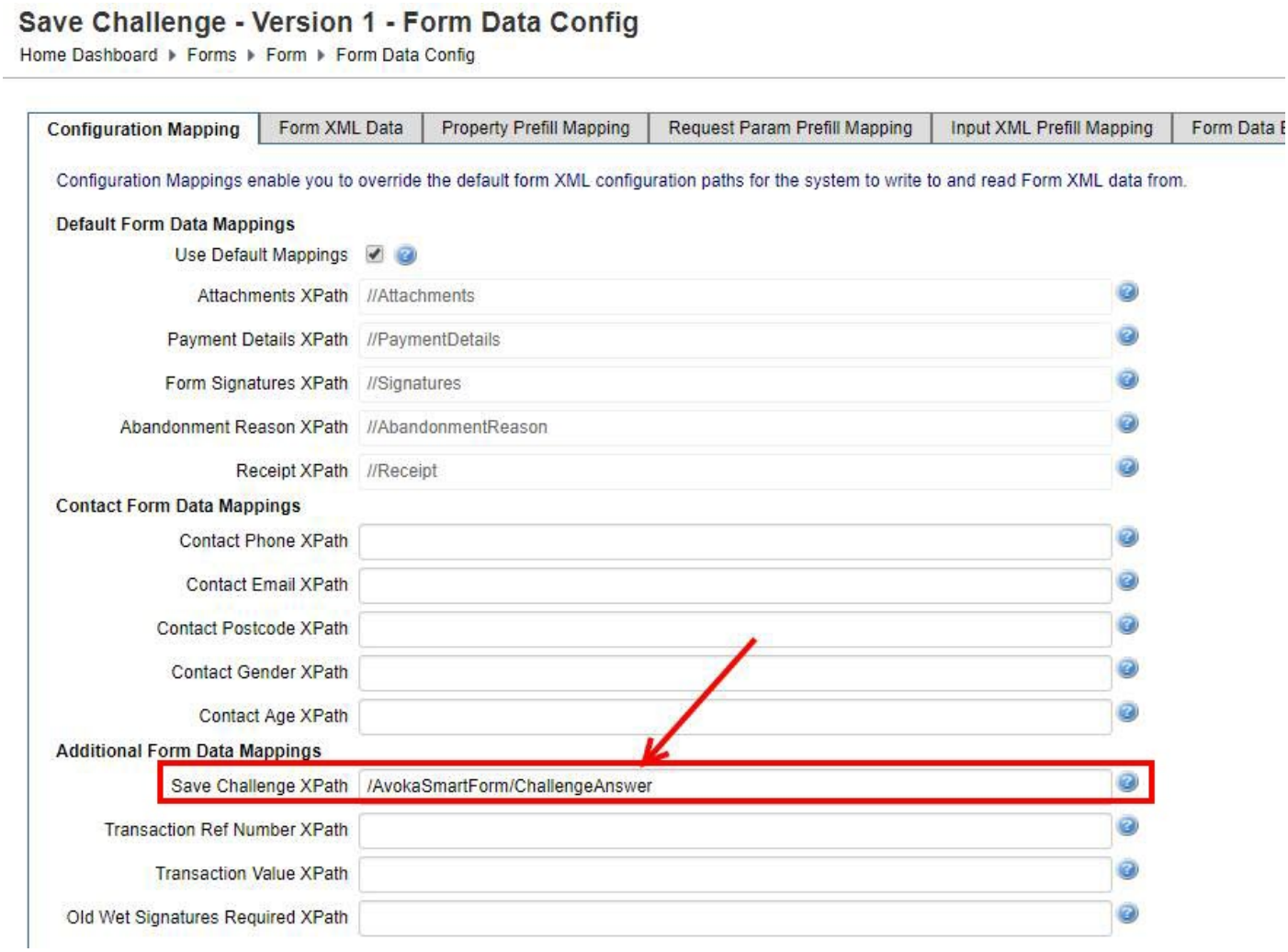


Transact Manager Configuration

Once the Save confirm and Save Challenge dialogs have been configured, it is recommended that you build the TM form version, and Import it into Transact Manager (there are a number of methods to do this and they are covered on the Build TM Form Version page).

After you have imported the form into Transact Manager, you need to ensure that the Data Config properties are set correctly for the form in Transact Manager (information on Data Configurations can be found in the Transact Manager Documentation).

The screenshot below displays and highlights a correctly configured Save Challenge XPath configured in the Data Config options in Transact Manager.



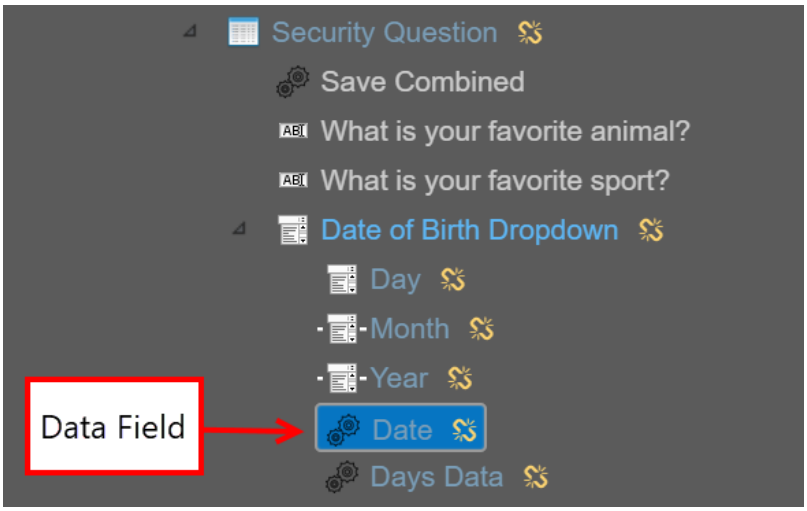
To get to this screen, follow the steps below.

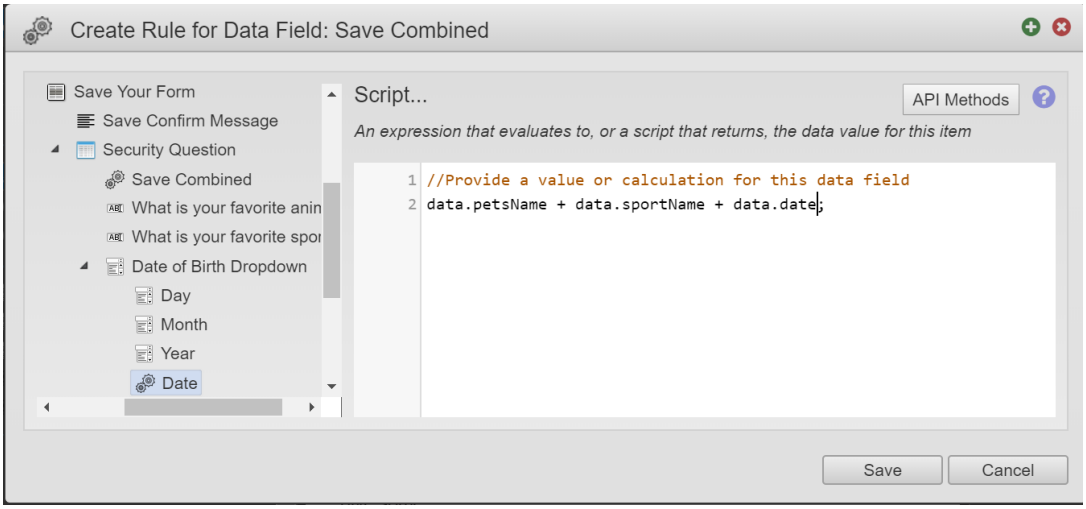
To set up your form version's configuration mappings (after it has been imported into TM):

1. Log in to Transact Manager
2. Navigate to *Forms* from the TM menu bar, and then click *Forms* from the dropdown menu.
3. Select the form that you want to configure Save challenge XPath and click the edit icon.
4. In the *Form Versions* section of the Form Dashboard, find the form version and click the *Data Config* link
5. The *Configuration Mapping* tab will be displayed (if not, switch to it).
6. Configure the options, as shown in the screenshot below
7. Click the *Save* button

Tips

When working with a Date of Birth component (or any compounded component), it is important that you use the pre-configured data field to create the concatenation calculation rule. The screenshots below highlight that date data field that comes pre-configured for all Date of Birth components and show how this data field can be referenced in a concatenation calculation rule.





Populating Multiple Security Questions from User Entered Data

Though it is quite useful to ask users questions that are completely removed from the application experience, it can also be useful and often times more convenient to ask users security questions based on information that they previously entered in the form. For example, to return to a saved form, a user must provide the email address that they entered when they were originally completing the form.

To populate multiple security questions, follow the above steps except you will need to create change rules on each of the components that will work as the answers to the security questions (these change rules should be applied when configuring the Save Confirm Dialog).

Examples of change rules

```
if(!Util.isBlank(data.lastName
)) {
     data.lastName1 =data.lastName;
```

```
}

if(!Util.isBlank(data.day)&&!Util.isBlank(data.month)&&!Util.isBlank(data.year))
{
     data.day1 = data.day;

     data.month1 = data.month;

     data.year1 = data.year;

     data.date1 = data.date;

     data.daysData1 = data.daysData;

}
```

Dialogs and Modal Pages

Unknown macro: 'redirect'

Page Contents:

Related Pages:

- Background Save
- Language Translator in Maestro
- Language Translator in Transact Manager
- Save Challenge (Save and Resume)

Dialog

A dialog is a pop up window that displays in the form after a set action. A dialog may be used to gather information from the user that is not part of the normal data entry of the form, for example , when a user clicks the Save and Close button a dialog will display confirming the save. You can also use dialogs for a lookup or search action. This data may then be used in the form or used to display messages on a modal page.

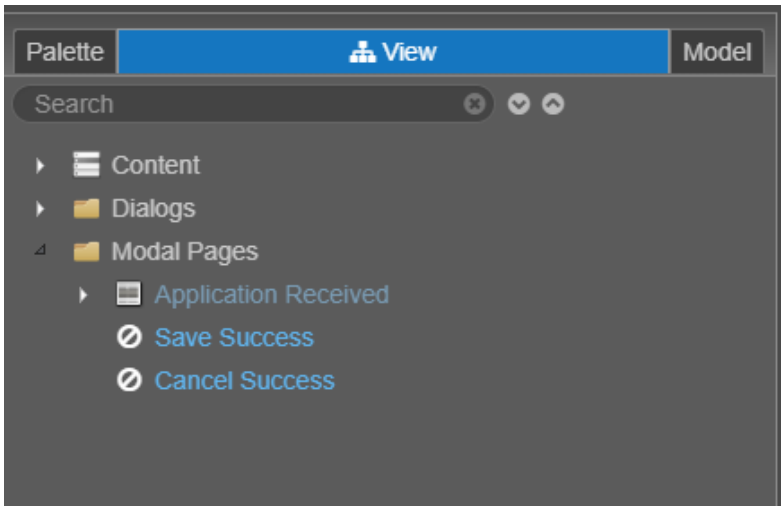
From the View panel we can see a folder for Dialogs. You will always see this folder, however the options within the existing dialogs may be restricted due to the access provided by the template designer.

If you don't have access to a dialog you will see the white cross-out ( ) icon. This indicates that the dialog can only be edited from the template.

Modal Page

A modal page is a special type of page within your form. It is usually used at stopping points in the form, such as on submit or confirmation of a form being saved or canceled. It is called a modal page because once that page has been displayed, your experience is "modal". This means that you cannot navigate to other pages, or do anything else with the form, until you have dismissed the modal page. A modal page is similar to a modal dialog, except it occupies the same area as the main pages of the form, rather than being a smaller pop-up page. You can only show one modal page at a time. You cannot have modal within a modal, a modal within a dialog or a dialog within dialog.

From the View Panel, we can see a folder for Modal Pages. You will always see this folder, however the options within the existing modal pages may be restricted due to the access provided by the template designer.



If you don't have access to a modal page you will see the white cross-out ( ) icon. This indicates that the modal page can only be edited from the template.

Translation

Unknown macro: 'redirect'

The translation feature allows multiple languages to be added to a single form. The user can easily switch languages in real time. Translation scripts can be set up in Maestro via the Translation UI or translation files can be created using CSV tools and uploaded to the form.

- Language Translator in Maestro
- Working with Translation CSV Files

Language Translator in Maestro

Unknown macro: 'redirect'

Page Contents:

Related Pages:

- Background Save
- Language Translator in Maestro
- Language Translator in Transact Manager
- Save Challenge (Save and Resume)

Overview

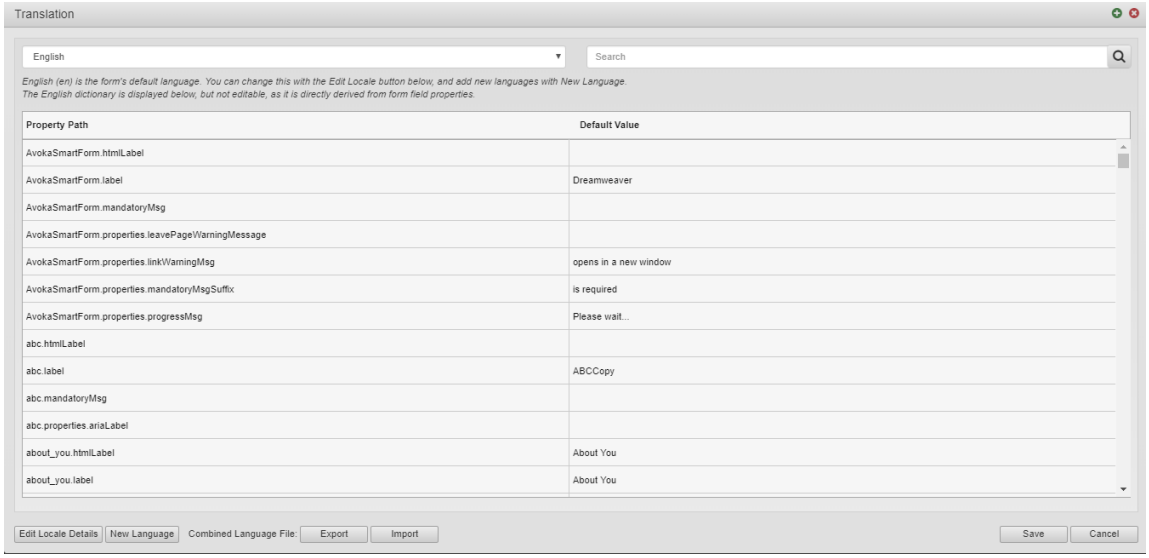
The translation feature allows multiple languages to be added to a single form. The user can easily switch languages in real time. Translation scripts can be setup in Maestro via the Translation UI or by using external CSV files.

Setup New Language in Maestro

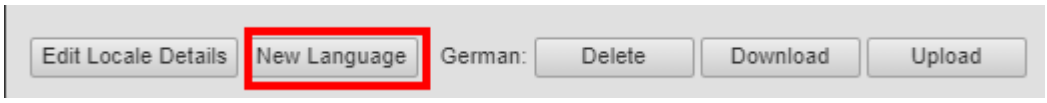
To setup translation in Maestro select the Translation button the toolbar.

This will display the Translation window. The window will display the default language for the form. If you have more than one language you can use the dropdown at the top of the window to switch between languages.

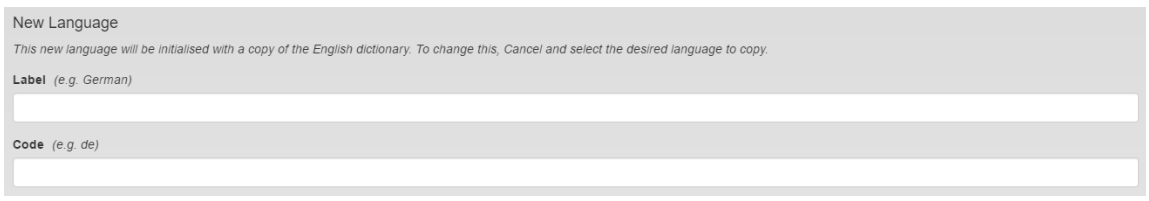
The first column in the window shows the property path (keys) for objects within the form and the second column shows the labels as displayed based on the selected language (default language). You are not able to edit the default language in the Translation window as it is directly derived from the form component properties.



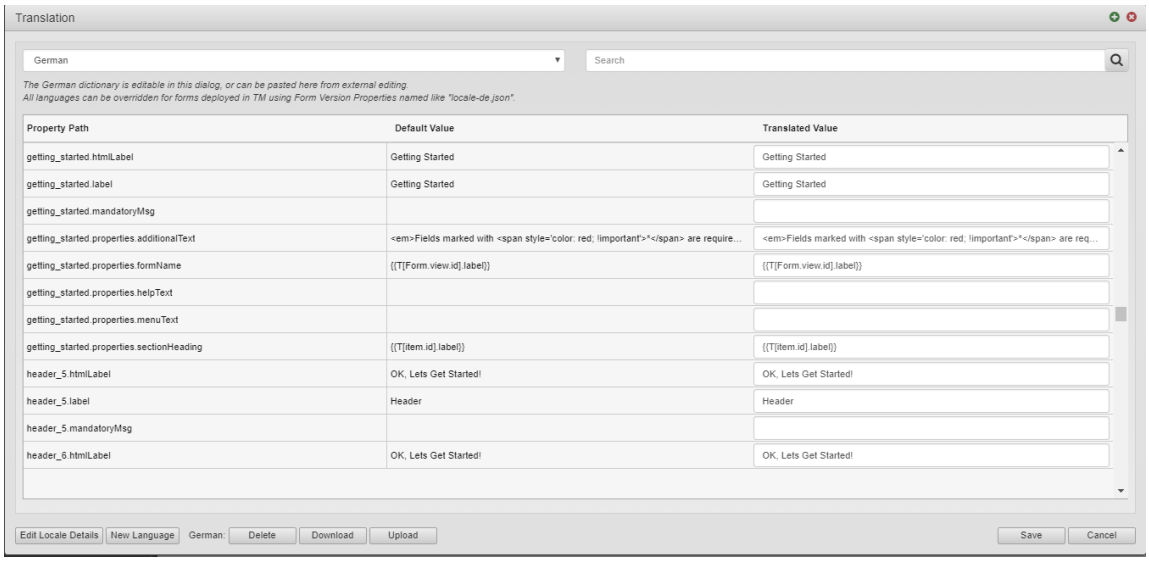
To create a new language click the New Language button at the bottom of the window.



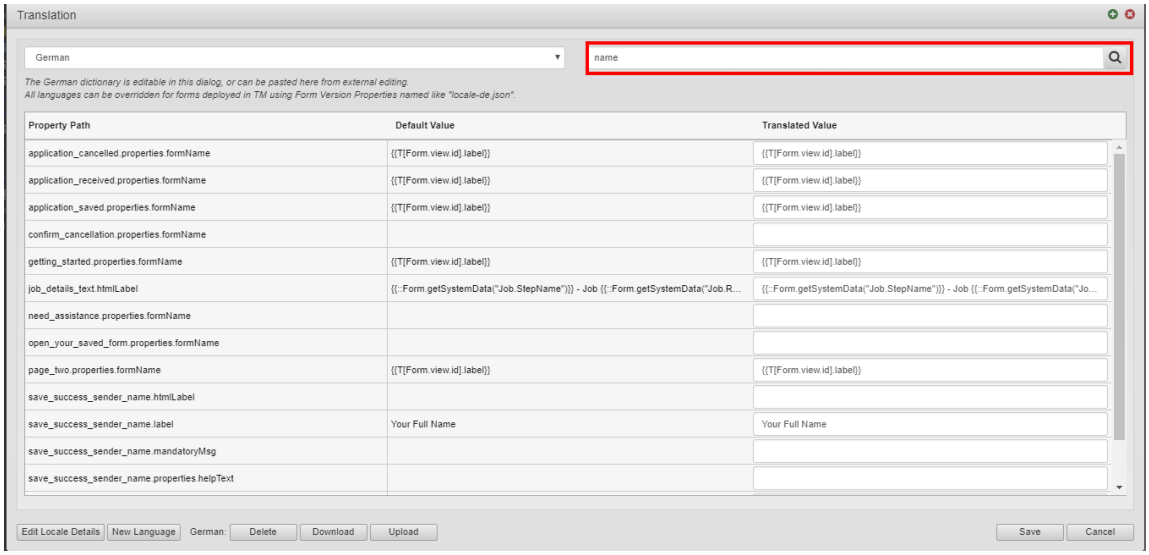
A new window will display where you can enter the language Label and Code.



Once the language has been created you will see the Translation window with three columns - property path (keys), default value and the translated values to use when the language is selected.



The search box at the top of the window will allow you to search based on the Property Path. The Property Path matching the entered criteria will display in the window.



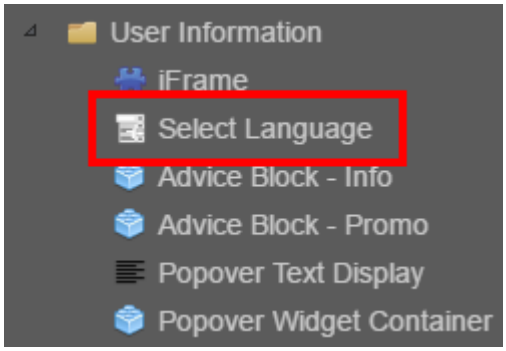
If you have a component selected when you click the Translation button, the list will scroll to the selected Property Path. The scroll position will remain as you switch between languages.

Once you have found the Property Path you want to translate you can enter the translated value in the third column.

Select Language Dropdown

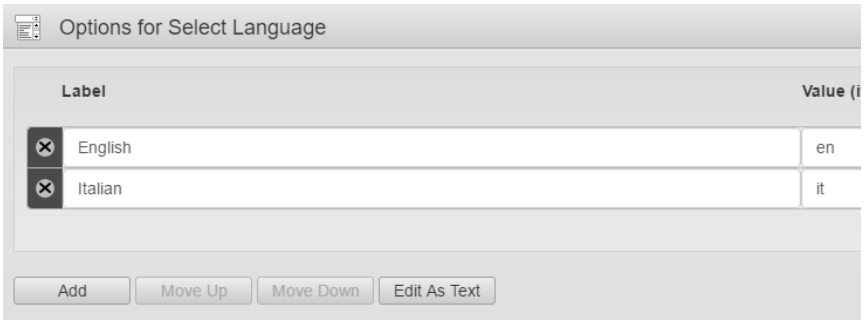
In order for the user to switch between languages you need to add the Select Language dropdown to the form.

In the Palette panel you can use the search box to find the dropdown or navigate to the User Information category.



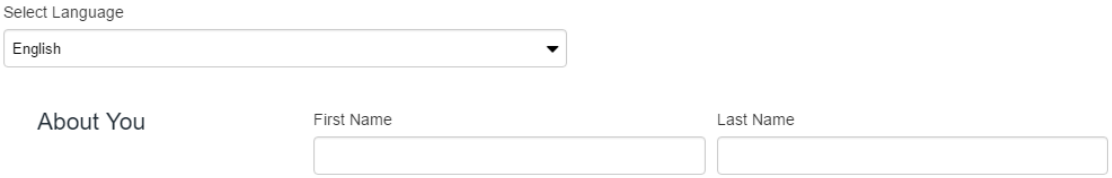
In the Properties pane select the *Options* button.

Configure the options to include the languages that have been created.The screenshot below shows English and Italian.

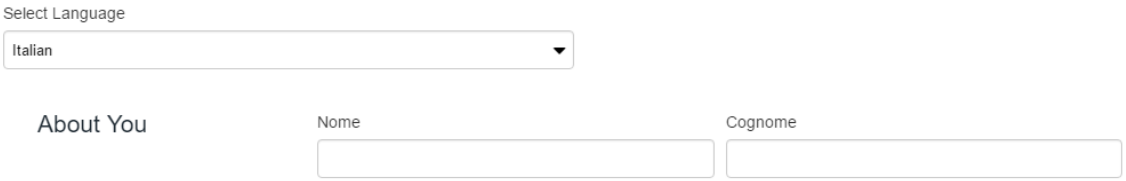


You can preview the form and, using the Select Language dropdown, switch between the languages.

The screenshot below shows English selected. Notice the First Name and Last Name labels.



The screenshot below shows Italian selected. Notice that the labels have changed to Italian, Nome and Cognome.

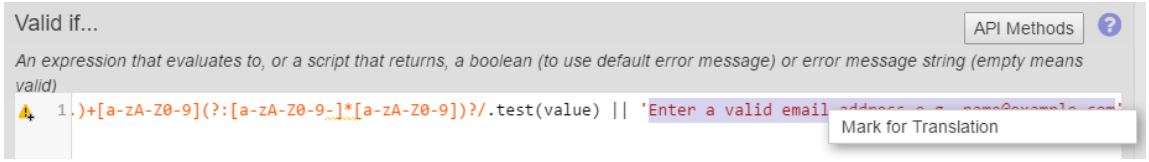


Mark for Translation

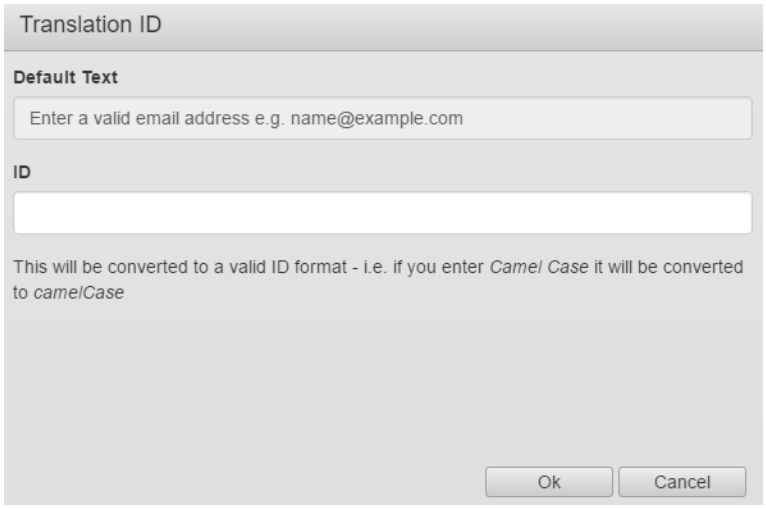
You can include any string, in any script, to be translated to the languages you have created.

In order to translate a string you will need to first mark it for translation, then enter the translated value. Marking a string for translation will create a translation ID for that string. The ID can then be found in the translation window, where you can enter a translation value.

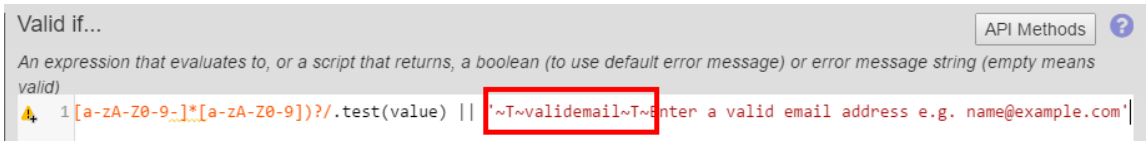
To mark a string for translation, right-click the string.This will display the *Mark for Translation* option.



Selecting Mark for Translation will display the Translation ID window. Default Text is the default string value. ID is the translation ID that you create.

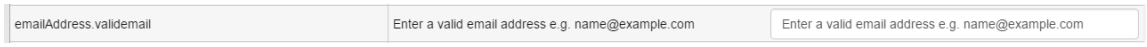


Once you have created the Translation ID you will see a token at the beginning of the string. In the code you can still see the default value for your string, and you can see that it has been marked with a token based on the Translation ID that you provided.



Once you have created the Translation ID you can return to the Translation window. You will need to use the dropdown at the top of the window to select the language you want to edit.

With the language selected you can find the new property key based on the Translation ID that was created. You will see the default value in the middle column and can enter the translation value in the field provided. Notice the translation value displays the default value, however you can change that value with the correct translation.



Language Translator in Transact Manager

Unknown macro: 'redirect'

Page Contents:

Related Pages:

- Background Save
- Language Translator in Maestro
- Language Translator in Transact Manager
- Save Challenge (Save and Resume)

Overview

To create a new language translation using Maestro see Language Translator in Maestro for more information.

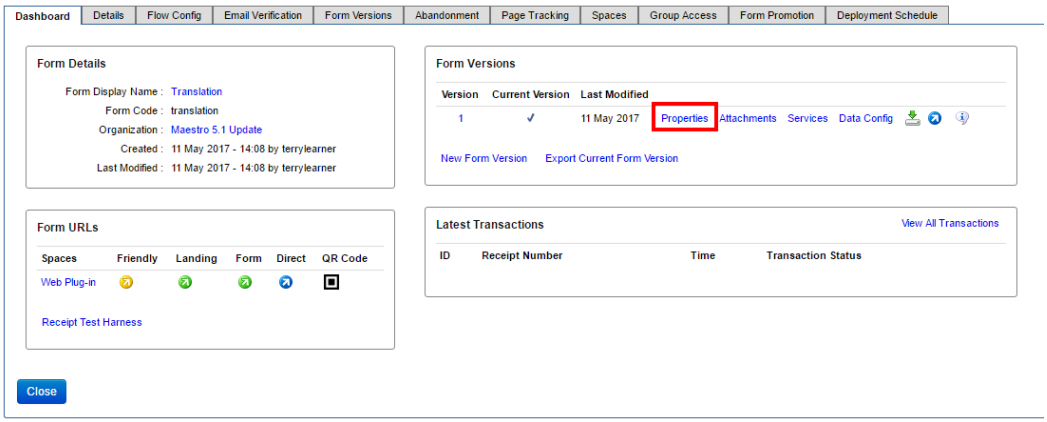
When the user switches between available languages the form will pull a language file from the server. This allows form builders the ability to edit the languages in either Maestro or Transact Manager. Editing the language in TM allows for different workflow options, for example a third party can create the translation file which can be loaded into TM. The JSON file created for each language is added to the form in TM via a Form Property. The translation can be created and edited outside of Maestro.

Even though the translation file will be completed outside of Maestro, the form will still need to have the Select Language dropdown (with the correct options configured) and the new languages will need to be created in the Translation window. In Maestro you will not enter any translation values as that will be done in the file outside of Maestro. For information on how to add the Select Language dropdown and how to create a new language see Language Translator in Maestro.

Once the translation file has been created you will need to create a form property for the JSON file.

Create New Form Property

1. Navigate to the Form Dashboard
2. Select *Properties* next to the version you want to update



3. Click *New*
4. Enter the property details
   Name needs to be entered in a specific format in order for the Select Language dropdown to correctly match the JSON file with the selected option. The Name should be */locale-en.json*, where "en" is the language ID used with the language was created in Maestro.
   The Data Type is JSON and the Value is the JSON file.

5. Click *Save*

The new property is created. You can render the form, use the Select Language dropdown and the form content should translate based on the details in the JSON file.

Working with Translation CSV Files

Unknown macro: 'redirect'

Translation has undergone several changes to the way that translation files are stored, edited, and configured by Maestro users and translators. The key concept of these changes is that translation files are now stored as CSV files. This concept provides Maestro users and translators more flexibility when working with and implementing multiple languages in applications.

- Working with Translation CSV files for Individual Languages
- Working with Translation CSV files for Multiple Languages

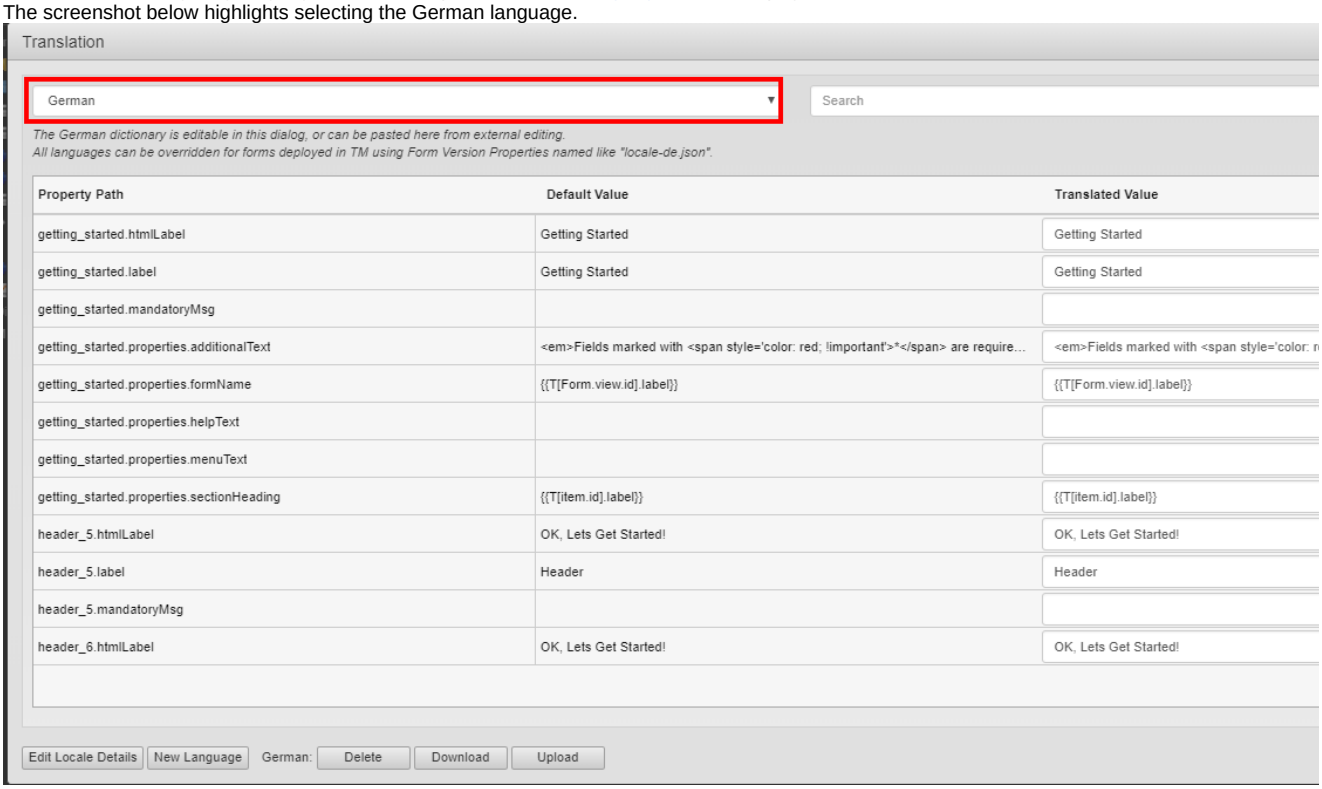
Working with Translation CSV files for Individual Languages

Unknown macro: 'redirect'

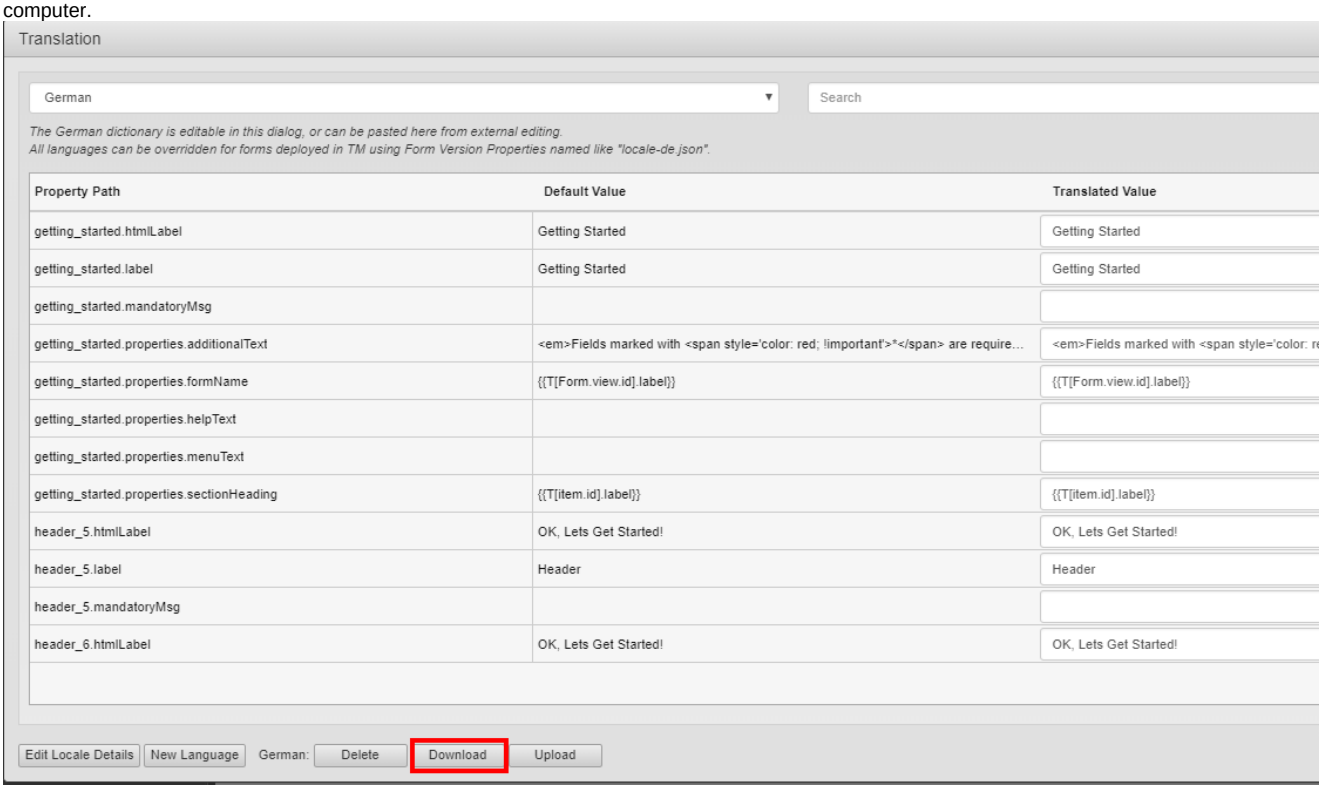
Maestro allows you to edit single translation files by downloading the CSV translation file, editing it, and uploading it back into the Maestro form.

To download, edit and upload a single translation CSV file, follow the steps below.

1. Open a form with multiple languages configured
2. Select *Translation* from the toolbar
   Selecting *Translation* displays the Translation window.
3. Select a language using the language dropdown menu. The languages in this list are established from the languages configured in Maestro, for information on configuring languages in Maestro, please see the Language Translator page of this documentation.
   The screenshot below highlights selecting the German language.



4. Click Download
   Clicking Download will download the German CSV translation file (normally named locale-xx.csv though the name of the file is arbitrary) to your computer.



5. Open the downloaded CSV file in any CSV editing tool (e.g. MS Excel).
   The screenshot below displays an example of a downloaded Translation CSV file. The first column represents the labels of the form, the second

column contains the base language and the third column is where the translated language will display. If the translated language has not been entered yet, then the base language will appear as a placeholder in the third column and should be updated with the new language.

	A	B	C
1	AvokaSmartForm.label	TransTest2	TransTest2
2	AvokaSmartForm.properties.mandatoryMsgSuffix	is required	is required
3	AvokaSmartForm.properties.progressMsg	Please wait...	Please wait...
4	AvokaSmartForm.properties.linkWarningMsg	opens in a new window	opens in a new window
5	form_header.label	Form Header	Form Header
6	form_header.properties.trackingCodePrefix	Reference Code:	Reference Code:
7	logo.label	Logo	Logo
8	job_details_block.label	Job Details Block	Job Details Block
9	image.label	Image	Image
10	job_details_text.label	Job Details Text	Job Details Text
11	job_details_text.htmlLabel	{{::Form.getSystemData("Job.StepName")}} - Job {{::Form.g	{{::Form.getSystemData("Job.StepName")}} - Job {{::Form.getSystemData("Job.Refer
12	navigator_chevrons.label	Navigator Chevrons	Navigator Chevrons
13	navigator_chevrons.htmlLabel	Navigator Chevrons	Navigator Chevrons
14	navigator_chevrons.properties.navAriaLabel	Secondary wizard status	Secondary wizard status
15	navigator_second_level.label	Navigator Second Level	Navigator Second Level
16	navigator_second_level.htmlLabel	Navigator Second Level	Navigator Second Level
17	navigator_second_level.properties.navAriaLabel	Secondary second level wizard status	Secondary second level wizard status
18	function_bar.label	Function Bar	Function Bar
19	save_and_close.label	Save and Close	Save and Close
20	save_and_close.properties.ariaLabel	Save and Close, opens a new in-page window	Save and Close, opens a new in-page window
21	cancel__exit.label	Cancel / Exit	Cancel / Exit
22	cancel__exit.properties.ariaLabel	Cancel / Exit, opens a new in-page window	Cancel / Exit, opens a new in-page window
23	open_saved_form.label	Open Saved Form	Open Saved Form
24	open_saved_form.properties.ariaLabel	Open Saved Form, opens a new in-page window	Open Saved Form, opens a new in-page window
25	error_block.label	Error Block	Error Block
26	error_block.properties.header	Please resolve the following issues before proceeding	Please resolve the following issues before proceeding
27	error_block.properties.subHeader	Click on an issue to go directly to the related section of the f	Click on an issue to go directly to the related section of the form.
28	error_block.properties.mobileHeader	{{Form.validation.errors.length}} error(s) found	{{Form.validation.errors.length}} error(s) found
29	error_block.properties.mobileSubHeader	Please touch here to resolve the first error.	Please touch here to resolve the first error.
30	error_block.properties.mobileA11yMessageBefore	The following {{Form.validation.errors.length + (Form.valid	The following {{Form.validation.errors.length + (Form.validation.errors.length == 1 &
31	error_block.properties.mobileA11yMessageAfter	Please select the screen to go to the first error.	Please select the screen to go to the first error.
32	content.label	Content	Content
33	content.htmlLabel	Content	Content
34	getting_started.label	Getting Started	Getting Started
35	getting_started.htmlLabel	Getting Started	Getting Started
36	getting_started.properties.sectionHeading	{{T[item.id].label}}	{{T[item.id].label}}
37	getting_started.properties.formName	{{T[Form.view.id].label}}	{{T[Form.view.id].label}}
38	getting_started.properties.additionalText	Fields marked with <span style='color: red; !importan	Fields marked with * are required
39	advice_block__info.label	Advice Block - Info	Advice Block - Info
40	info_block_image.label	Info Block Image	Info Block Image

6. Edit the CSV file by entering translated values into the third column. For example Hello translated to German is **Hallo.** Please note that Maestro does not support multiple sheets in a single workbook, so when you edit the CSV file, it is recommended that you **DO NOT** create/use multiple sheets.
7. Save the CSV file. The name of the file **DOES NOT MATTER** but please see the information below for the actual file type that should be used to save your file.

> If your translation (language) uses non-ASCII characters, you need to "Save as" a UTF-8 CSV File.
>
> ASCII characters include the digits 0 to 9, lowercase letter a to z, uppercase letters A to Z, and punctuation symbols (e.g %,$,*,&). More information on ASCII can be found at https://www.computerhope.com/jargon/a/ascii.htm

8. Navigate back to the Translation Window (in Maestro) and click the *Upload* button
9. Navigate to the location of the saved CSV file on your computer and select it as an upload option.
   Selecting the CSV file will upload it to the opened form.
10. Click the *Save* button
    This will apply the edited CSV file and translated values to the form.
11. Save the form
    It is recommended that you click Save to ensure your changes are saved in the form.

Working with Translation CSV files for Multiple Languages

Unknown macro: 'redirect'

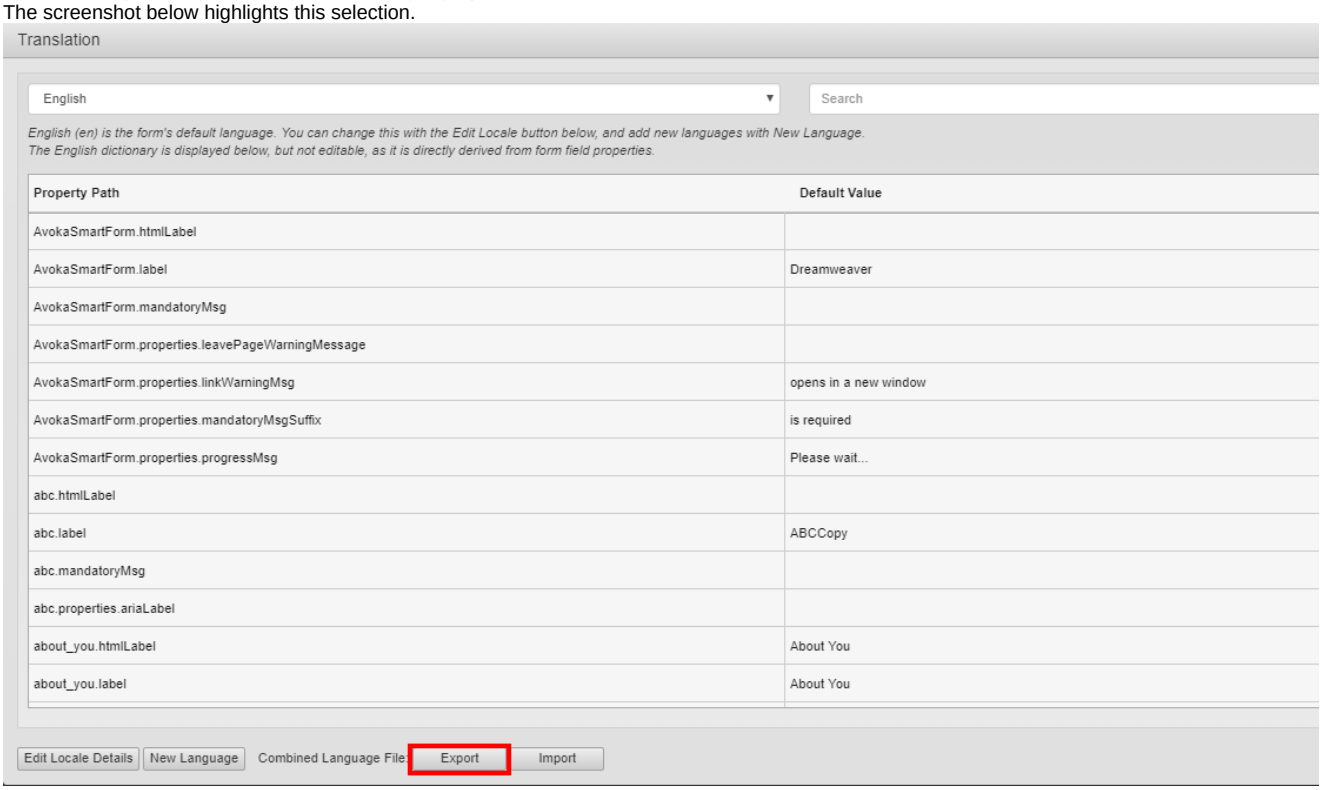
Maestro offers users the option to export the CSV file that contains all languages that are currently configured for a form. Once exported, you can edit the CSV file in any CSV editing tool and add columns with new languages for the form.

To export, edit and upload the CSV file with all configured languages, follow the steps below.

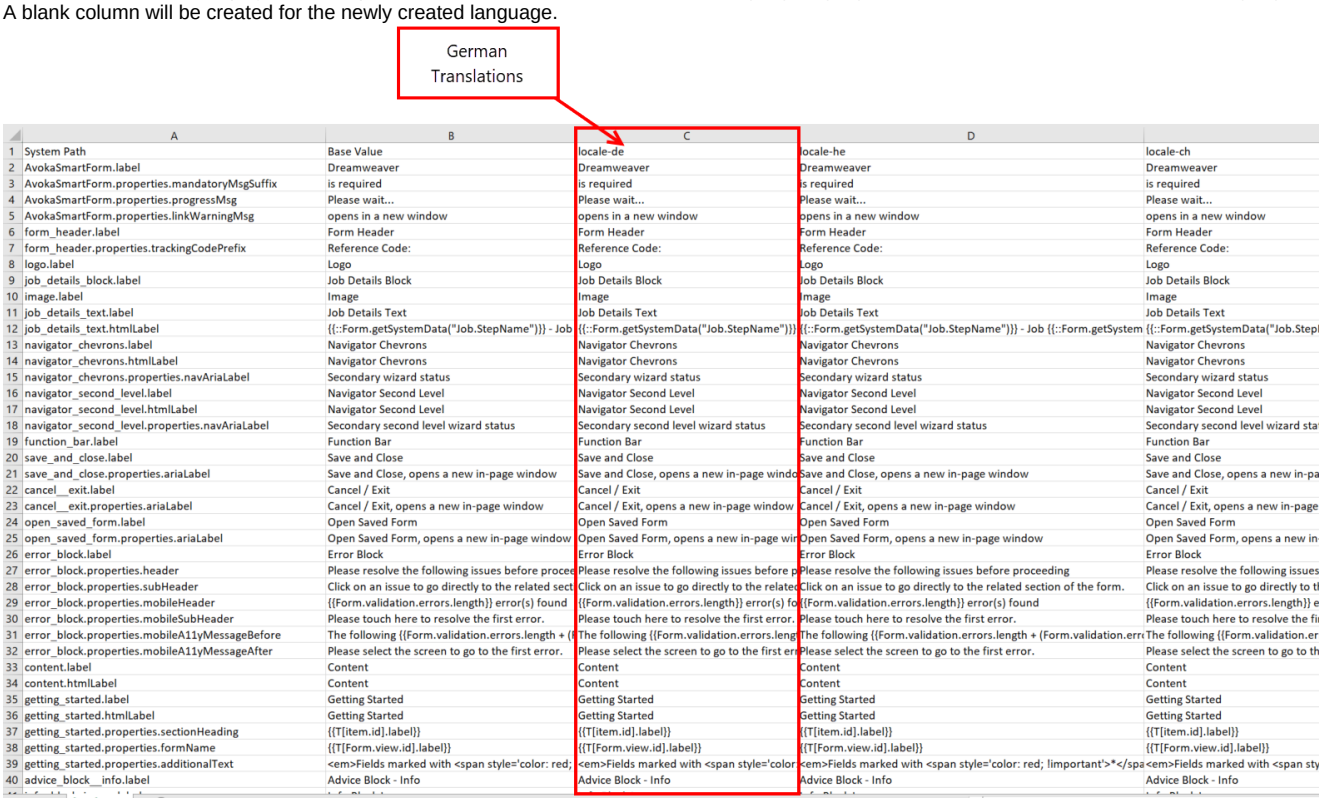
1. Open a form with multiple languages configured
2. Select *Translation* from the Dashboard
   Selecting *Translation* will open the Translation window with a list of translation values for each configured language.
3. Click *New Language* (if you want to add new languages to the CSV file. If you do not want to add any new languages, skip to Step 7)
   Clicking New Language will open the New Language screen. The screenshot below displays this screen.

4. Enter a label for the new language (ie the name of the language). For example, Hindi.
5. Enter a code for the new language. For example, hi.
6. Click OK. Clicking OK will save the new language

7. Click *Export* from the Combined Language Section of the window. Clicking *Export* will download the CSV file with all the configured languages of the opened form (including any newly created languages).
The screenshot below highlights this selection.



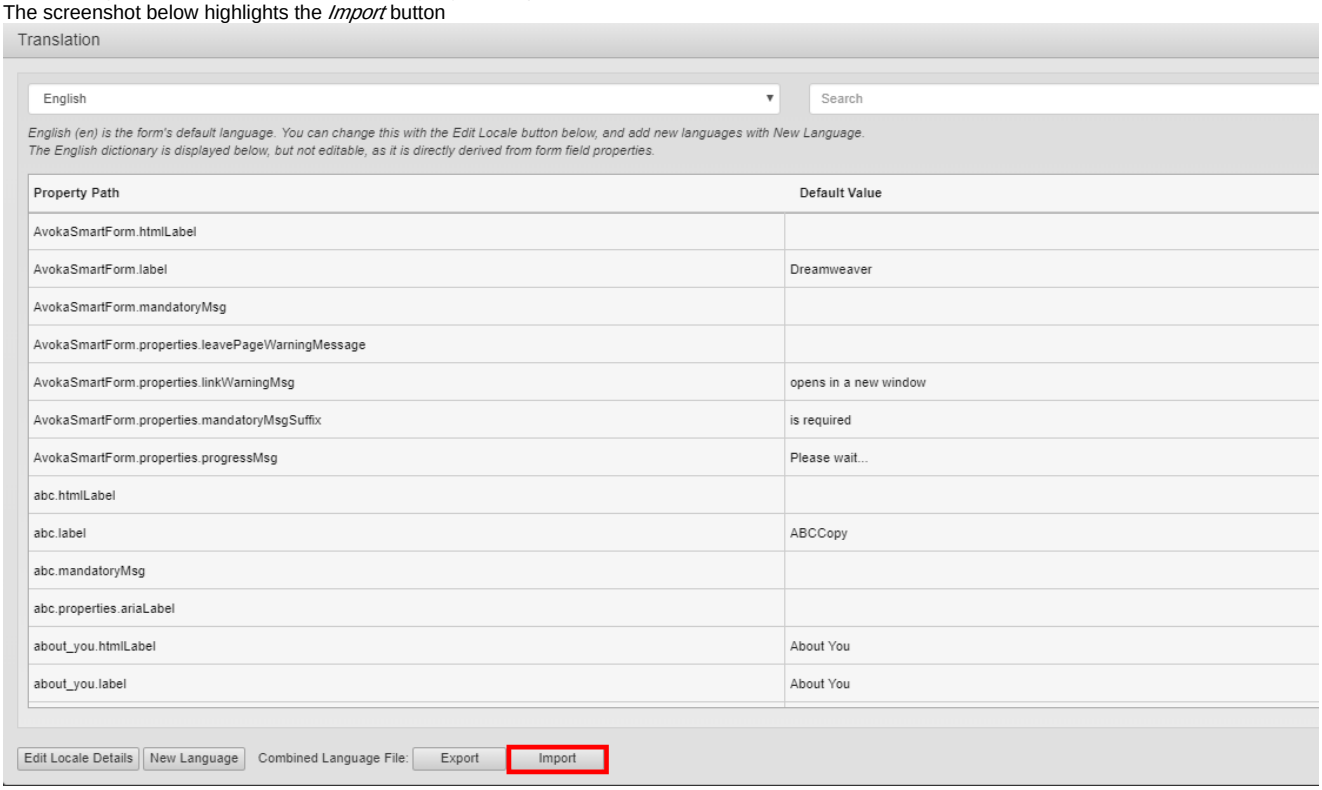
8. Open the CSV file in any CSV file editor tool (e.g MS Excel)
The screenshot below displays an example translation CSV file with the German language highlighted. Each column contains a different language. A blank column will be created for the newly created language.



9. Edit the CSV file to contain the translations for the newly created language or edit any of the languages that have already been configured. Please note that Maestro currently does not support multiple sheets on a single spreadsheet so when you edit the CSV file, it is recommended that you **DO NOT** create/use multiple sheets in a single CSV file.
10. Save the CSV file. The name of the file **DOES NOT MATTER** but please see the information below for the actual file type you should save your file as.

> If your translation uses non-ASCII characters, you need to "Save as" a UTF-8 CSV File.
>
> ASCII characters include the digits 0 to 9, lowercase letter a to z, uppercase letters A to Z, and punctuation symbols (e.g %,$,*,&). More information on ASCII can be found at https://www.computerhope.com/jargon/a/ascii.htm

11. Click the import button from the *Combined Language File* option.
    The screenshot below highlights the *Import* button



12. Select the edited CSV file
13. Click the save button and save the form
    Once you Save and Build the form, the changes you made to the translation CSV file will be reflected in the form.

Styling

Unknown macro: 'redirect'

The content within this section covers information relating to styling a form in Maestro.

The list below identifies the topics covered within this section.

- Selecting a Brand
- Fundamentals of Styling

Selecting a Brand

Unknown macro: 'redirect'

Related Pages:

- Selecting a Brand

A brand defines the look and feel of a form. It controls colors, fonts, and other aspects of the form.

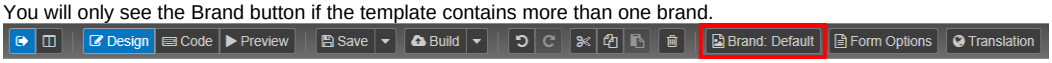
Brands are created in the template. Most organizations create at least one brand thus ensuring that all the forms created look the same without any effort by the form builder. This also means that if the organization changes their branding, all you need to do is modify the brand accordingly, and re-build the TM form version. You don't need to modify each form individually.

Some organizations may have several different brands, due to acquisitions or marketing requirements. You can add as many brands as necessary to a template.

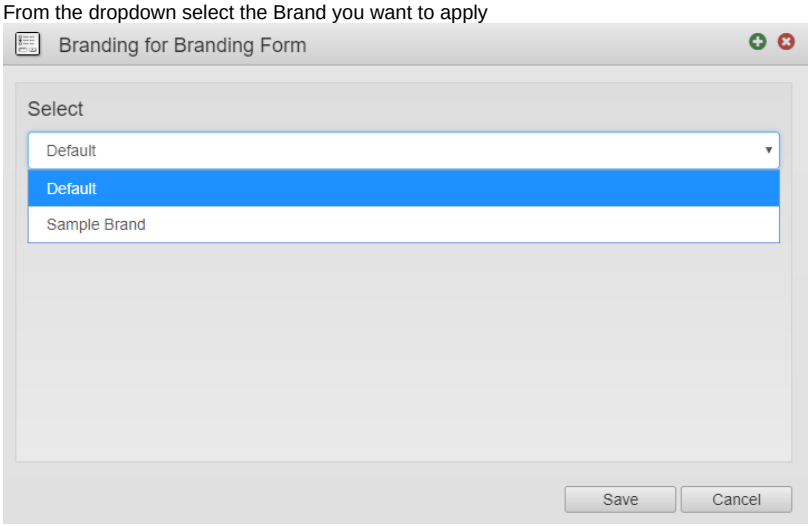
If a template has more than one brand, the Brand button will be available to the form builder. The form builder can use the Brand button to switch between the available brands. The Brand button also displays the brand that is currently being used by the form.

Switch between available brands in a form:

1. Click the *Brand* button on the toolbar
   You will only see the Brand button if the template contains more than one brand.



2. From the dropdown select the Brand you want to apply



3. Click *Save*
   The selected brand will be applied to the form.

If you want to use different brands on the *same* form you will need to apply the brand and then publish/build the Transact Manager version for each brand.

Fundamentals of Styling

Unknown macro: 'redirect'

Related Pages:

- Fundamentals of Styling
- Selecting a Brand

Before you do any styling within your form, you should carefully consider whether or not you should be making any styling changes. Most, if not all, styling should be done by the template designer within the template. This will ensure that all forms follow the corporate style-guide. As with most things, there are exceptions to the rule therefore styling can be done in the form if absolutely necessary, however it is always a good idea to speak with the template designer before doing so. The template designer can help to determine whether or not you should be making style changes in the form or if the template needs to be updated. If a change is needed and it applies to other forms, it is best that it is done in the template so that all forms can access the change. This type of management will allow all forms to continue to follow the corporate style-guide, which keeps all forms consistent.

Templates allow for corporate brand control as they will allow all forms to have a consistent look. Templates are created based on the corporate style-guide, by the template designer. The responsibility of the template designer includes the styling of the form's chrome. This would include areas such as the navigator, footers, and styling of specific components. The template designer works with the corporate style-guide to create a template.

Individual forms should never (or rarely) deviate from this corporate style. This means that a form builder focuses on the specific form content as well as the journey of the form user. The form builder should not change any styling without first talking to the template designer. This will ensure that the corporate style-guide is properly used and consistent across all forms.

Maestro attempts to provide a number of styling capabilities that are sometimes in tension with each other.

- Maestro gives you the full power of HTML5 and CSS when designing the style of your organization's forms.
- Maestro encourages re-use, so that you don't have to continuously re-create styles that you have already created.
- Maestro encourages consistency, so that all fields in a form, and all forms for an organization, look and behave the same.

In order to accomplish these goals, Maestro provides a number of features:

- A graphical style editor
- Access to CSS if required
- The ability to create and modify re-usable styles

Remember, most styling should be done in the template by the Template Designer. There are exceptions to this, however you should always speak with the Template Designer before making any changes.

Migrating to Transact Manager

Unknown macro: 'redirect'

The content within this section covers information relating to building a TM form version in Maestro for use in Transact Manager.

The list below identifies the topics covered within this section.

- Build TM Form Version
- Update TM Form Version
- Update TM Form Version (for TM versions 5.0.6 and higher)

Maestro was first released in August 2016, with Transact Manager 5.0. This is the first fully supported version of Transact Manager (TM). Publishing Maestro forms to any version prior to TM 5.0 will not be fully supported. Many features may work, but troubleshooting and support will not be provided for anything prior to 5.0. We are not able to maintain a matrix of features and releases which are unsupported in those earlier releases. For this reason, publishing Maestro forms to a TM server earlier than 5.0 is at your own risk.

Build TM Form Version

Unknown macro: 'redirect'

Page Contents:

Related Pages:

- Build TM Form Version
- Update TM Form Version
- Update TM Form Version (for TM versions 5.0.6 and higher)

Overview

There are two parts that make up the process of migrating (publishing) a form from Maestro to Transact Manager.

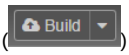
1. Build the TM form version in Maestro (creating a zip file)
2. Import the zip file (the built TM form version) to Transact Manager

The build button will create a TM form version. The TM form version is a FAR file that is included in a zip file, which then needs to be imported into TM. Once imported to TM the user will be able to configure the form as needed. You must complete the import process in order to work with the form in TM.

Build TM Form Version

The Build button will allow the user to:

1. Build a TM form version that can be downloaded for import into TM or
2. Build a TM form version which can be quickly viewed and downloaded later

There are four options available from the Build (

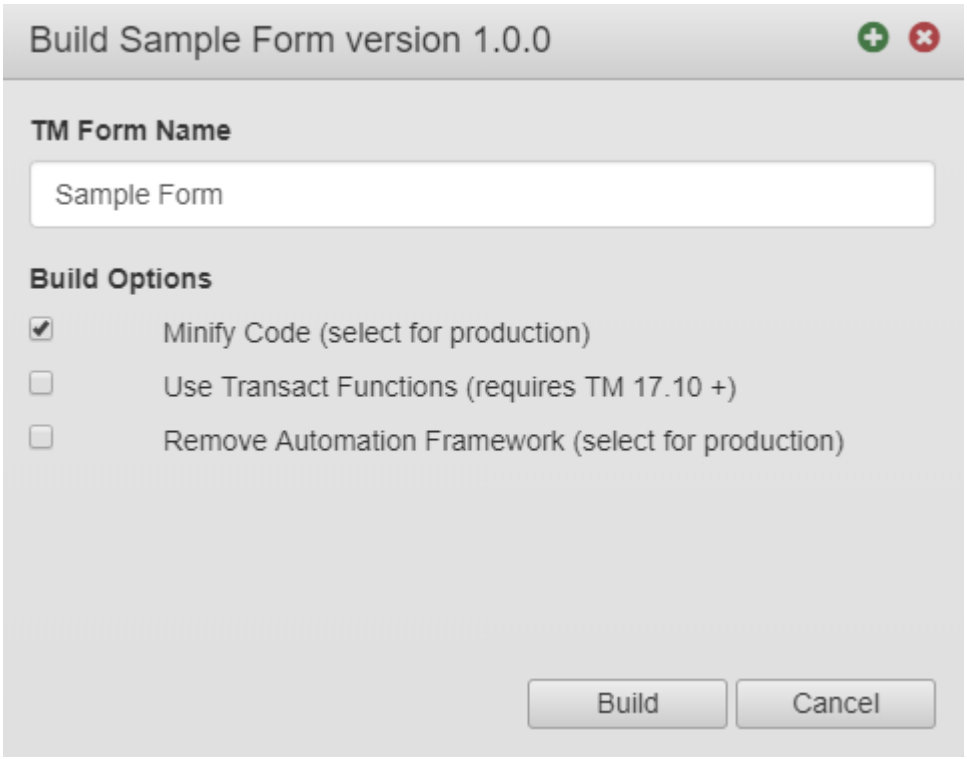
) button.

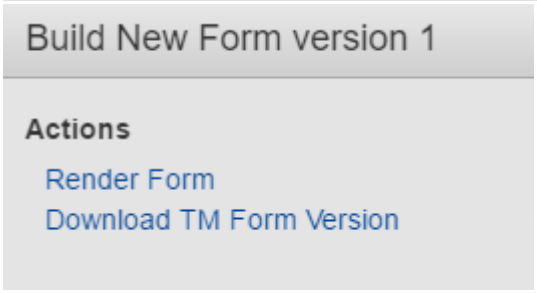
Build - Clicking the Build button (not the dropdown arrow) will build the TM form version. This will not download the TM form version or display the rendered form. This option is useful when you are re-building a TM form version that is already imported into Transact Manager. See Update TM Form Version (for TM versions 5.0.6 and higher) for more information.

Build and Download TM Form Version - This will build a TM form version then immediately download the zip file needed for import into TM.

Build and Render Form - This will build a TM form version (which can be downloaded from the Management Dashboard) and render the form in a new tab. This option will most likely be used when you need to view or test a rendered version of the form. In order to configure the form in TM the user will need to download the TM form version and import it into TM.

Build with Options - This will display the Build window with options available for preparing the TM form version. Once the TM form version is built you will have the option to download or view rendered form.





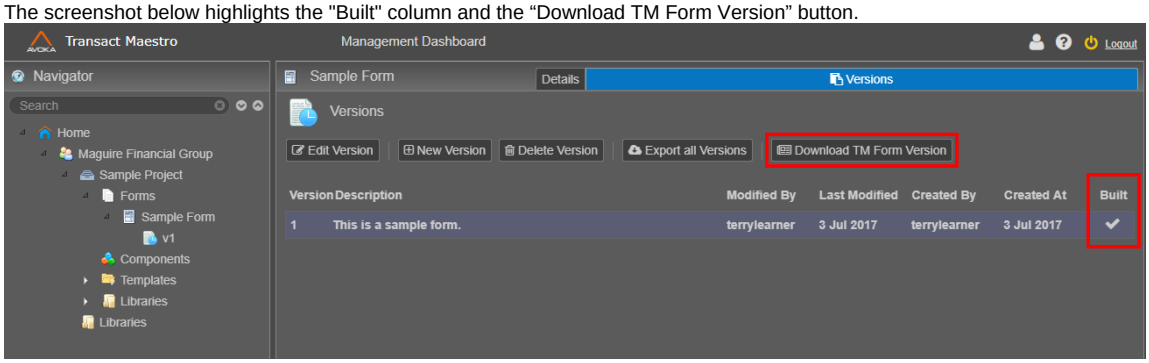
TM Form Name – This is the name of the form as it will appear in Transact Manager. By default it will pre-populate with the existing name of the form, however you can change it as needed.

Minify Code (Select for production) - Minify Code is the process that reduces the size of the form without changing its functionality. Selecting this option will make the form more efficient to transmit. However, reducing the code makes it difficult to read, so consider deselecting this option when debugging.

Remove Automation Framework (Select for production) - The Maestro Automation Framework (MAF) provides an API which can be used by other tools to automate and investigate the state of a Maestro form. Once a form is ready for production, it should be built with the Remove Automation Framework option selected to remove the MAF code. Selecting this option will reduce the download size of the form and prevent misuse of the API in production forms.

Please note that once a TM form version is built the user can download the form at any time from the Management Dashboard. The "Download TM Form Version" button is available on the Versions screen of the Management Dashboard. This button will only be available for selection once the TM form version has been built (in Maestro) at least once. Notice the "Built" column. This will indicate if the form version has already been built. If there is a checkmark the "Download TM Form Version" button will be available. If there is not a checkmark in the column you will need to open the form version in Maestro and build the TM form version from the editor window.

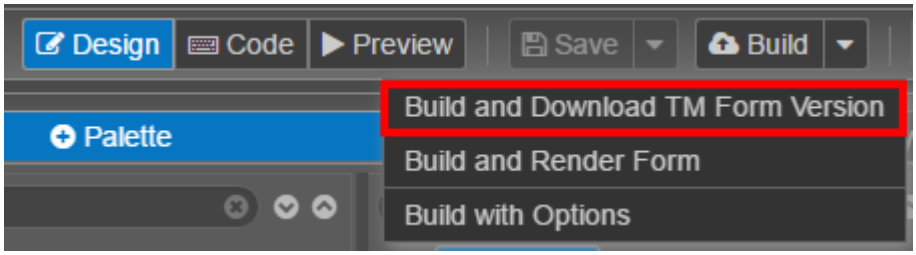
The screenshot below highlights the "Built" column and the “Download TM Form Version” button.



Import the zip file to Transact Manager

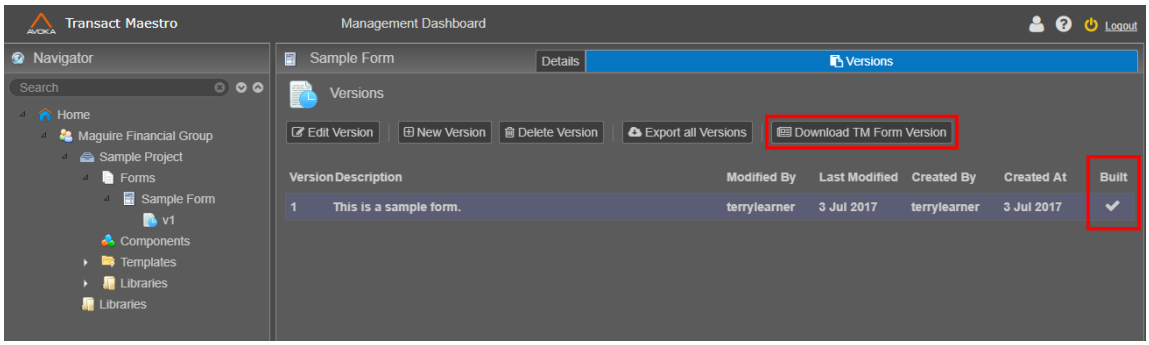
> The TM form version zip file from Maestro is usually only used for import into lower environments (such as Test or UAT). When promoting a form to a higher environment this is done using an exported file from TM.

Once the TM form version is built and downloaded you will need to import the zip file into TM. The easiest way to build the form and download the zip is to select Build and Download TM Form Version from the Build button in Maestro.



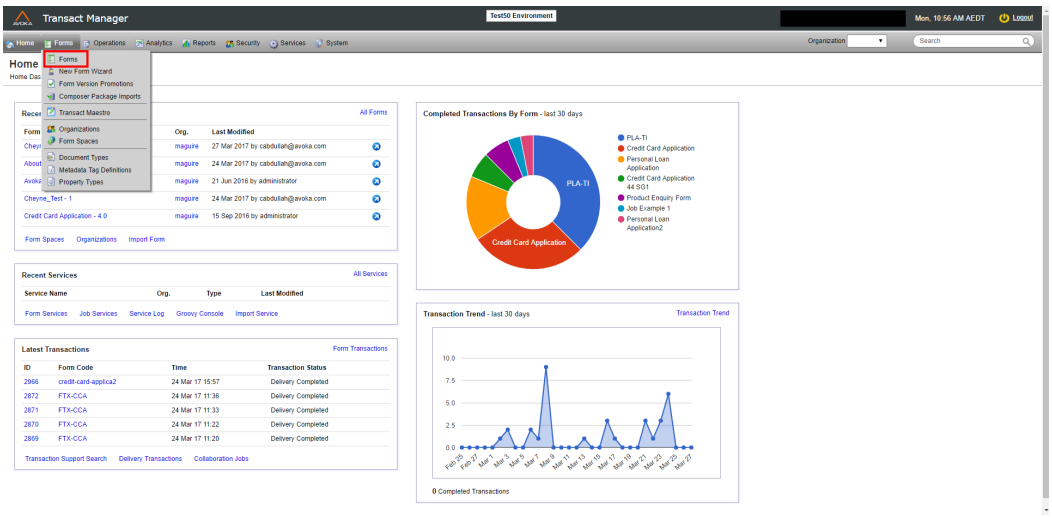
If you use one of the other Build options, or need to re-download the build TM Form Version you can do so from the Management Dashboard.

The *Download TM Form Version* button is available on the Versions screen of the Management Dashboard. This button will only be available for selection once the form has been built (in Maestro) at least once. The *Built* column indicates if the form version has been built in Maestro. The screenshot below highlights the "Built" column and the “Download TM Form Version” button.

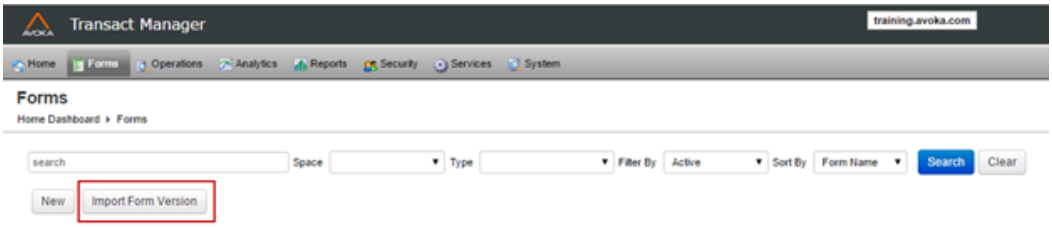


Now that you have built the TM form version and downloaded a zip file of the form, it is time to import the zip file into TM.

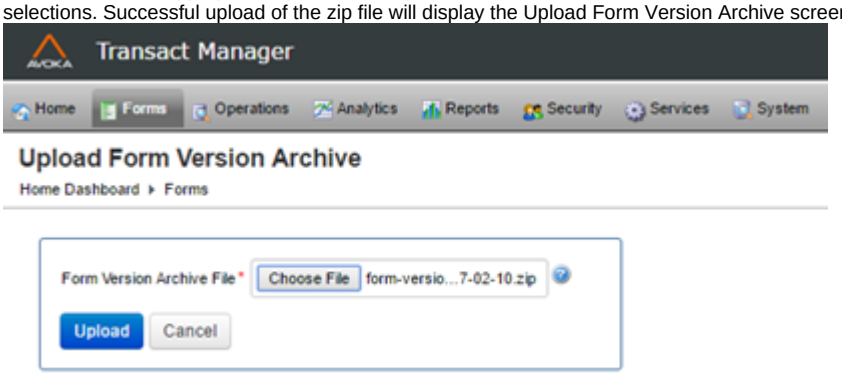
1. Navigate to the Transact Manager server that you wish to import the form into and log in.
   Logging in will redirect you to the Home Dashboard.
2. Select the Forms menu then click Forms.
   This selection will display the forms page. The forms page displays every form that has been uploaded to the TM server that you have chosen. The screenshot below illustrates this selection.



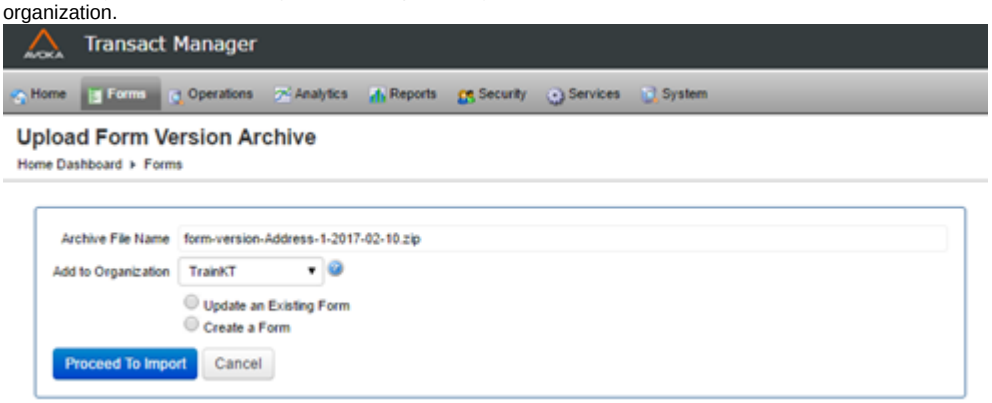
3. Select the *Import Form Version* button.
   The screenshot below highlights this button.



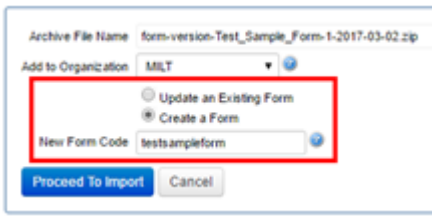
4. Select the *Choose File* button.
   Selecting this button will allow you to browse for and select the downloaded zip file. Once you have selected the appropriate zip file, click “Upload”. The screenshot below illustrates these selections. Successful upload of the zip file will display the Upload Form Version Archive screen.



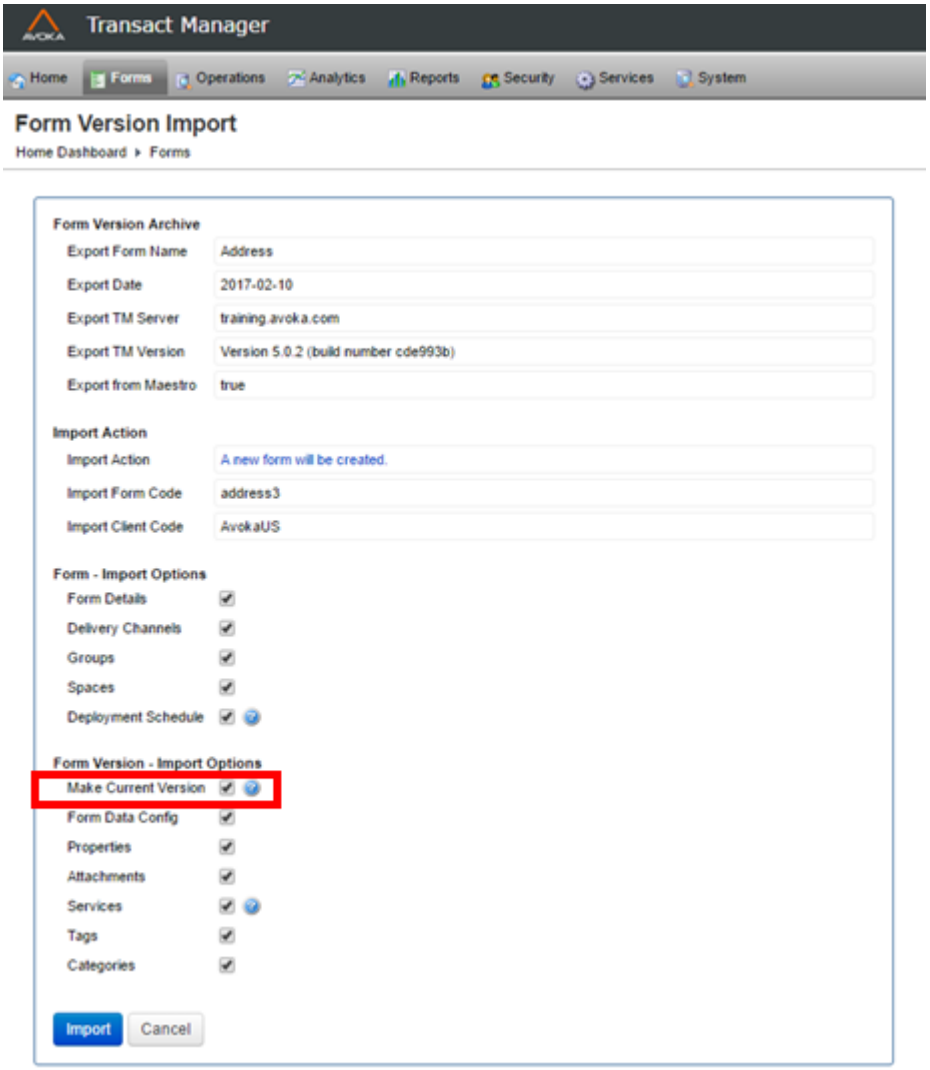
5. Select the appropriate organization (if relevant) for the form from the dropdown menu.
   You will only see the Add to Organization dropdown if you have access to more than one organization.



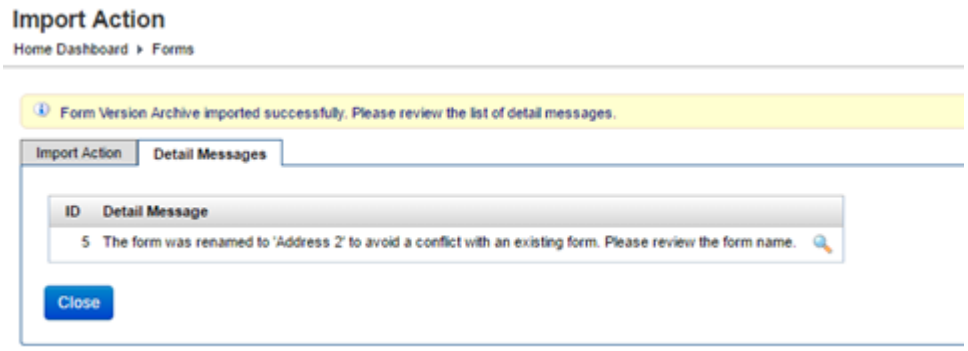
6. Select *Create a Form* from the available options.
   - **Create a Form** - Selecting this option will display the New Form Code textbox. Transact Manager will automatically fill this textbox with a suggested name for the form. If you are unhappy with the suggested name, you can change it by selecting the 'New Form Code' textbox. The 'Form Code name' uniquely identifies the form across all the organizations in the current TM server environment. The 'Form Code Name' is an alphanumerical string that cannot include spaces. The screenshot below illustrates this option.



7. Click *Proceed to Import* - Clicking this button will display the options screen for the form. It is on this page that you can select or unselect any options that you may want to override. Ensure that you select the option, 'Make Current Version', if you want to make the form that you are uploading the "current version" of the form (ie the form that your form users will be completing). The screenshot below illustrates the options screen and highlights the 'Make Current Version' option.



8. Select *Import* - Selecting import will officially import the form into the TM server environment and display the summary screen. Note that If the *Create a New Form* option was selected during the import process, and there was a form naming conflict (ie a form with the same name already existing on the server), the form will be renamed, and a message will be displayed in the Detail Message tab. The screenshot below illustrates a Detail Messages screen displaying a form name error.

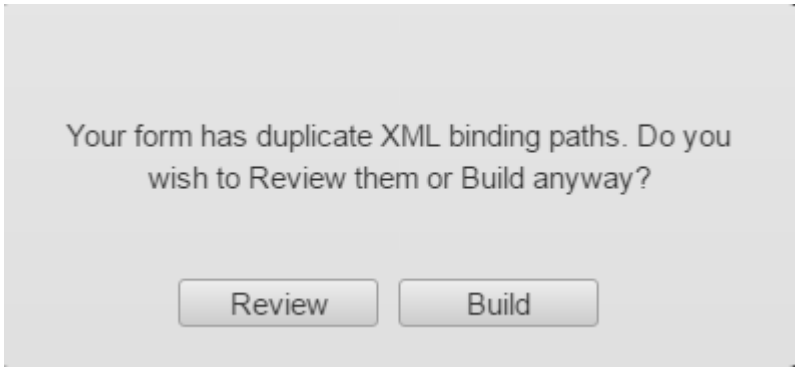


9. Select Close - Selecting close will redirect you back to the forms page.

Common Errors when Building a TM Form Version

Duplicate Binding

One common error that occurs when building a TM form version is a Duplicate Binding. This occurs when a form builder tries to build a TM form version with duplicate submission data bindings on the form. When this error is encountered by Maestro, the following message is displayed.



In building the TM form version, Maestro has detected that two or more components share the same data binding location; this message is a warning that users may experience unintended behavior during data entry. Clicking “Build” confirms that you are aware of the duplicate binding and wish to proceed; clicking “Review” stops Maestro from building the form and displays a list of the duplicate bound components.

For Composer Users

There are significant differences in the publishing/building process between Maestro and Composer.

As a standalone tool, Composer requires the form designer to choose the Transaction Manager server to host the form. When published the designer, or an administrator, is then required to choose the Organization and form update strategy (New Version or Update Existing).

By design, Maestro is a Space associated with a Transact Manager server, so it can only build to the same server. Also by design, Maestro requires Forms to be arranged in organizations that align with those configured on the server. Finally, the form builder controls the versioning of forms in the Maestro Management Dashboard.

Update TM Form Version

Unknown macro: 'redirect'

Related Pages:

- Build TM Form Version
- Update TM Form Version
- Update TM Form Version (for TM versions 5.0.6 and higher)

Page Contents:

Overview

Updating a form that has already been uploaded to a Transact Manager environment is a fairly common requirement. This requirement may be asked after customer feedback, or it may be due to continuous improvement processes within your company. Each time a form is updated in Maestro you will need to build the TM form version and upload the zip file to Transact Manager.

If you are using Transact Manager 5.0.6 or higher you can use the 'Update from Maestro' button in Transact Manager. See Update TM Form Version (for TM versions 5.0.6 and higher) for more information.

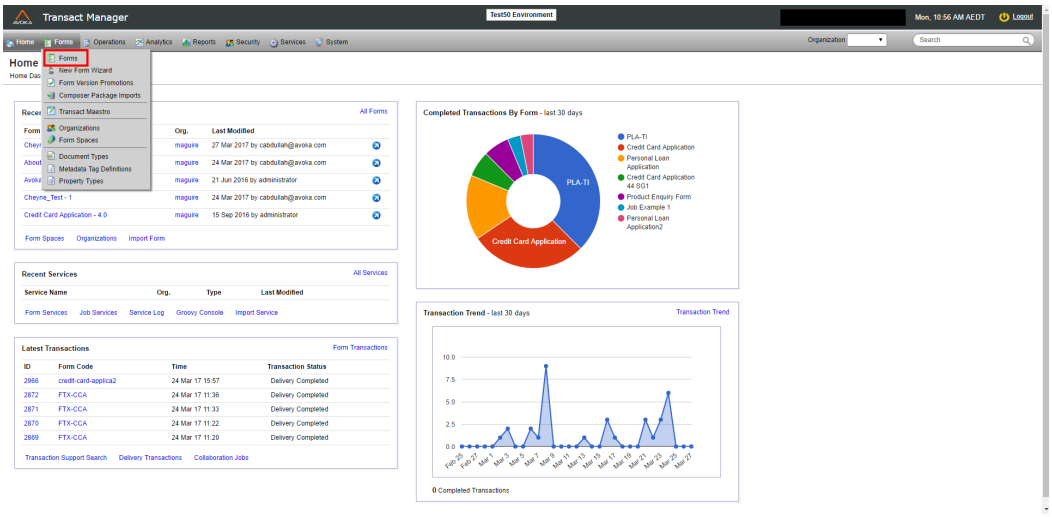
Create and upload the zip file to Transact Manager

Once you have updated the from in Maestro, you need to build and download the TM form version. The process of building and downloading the TM form version is exactly the same as the process outlined in the Build TM Form Version page of this documentation. View Build TM Form Version for detailed steps on building the TM form version in Maestro.

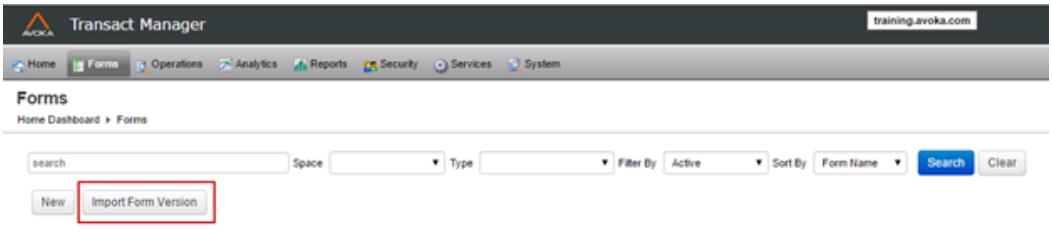
1. Select the Build button then click Build and Download TM Form Version in Maestro
   The Build and Download TM Form Version will build the TM form version and download the zip file.

Import the zip file to Transact Manager

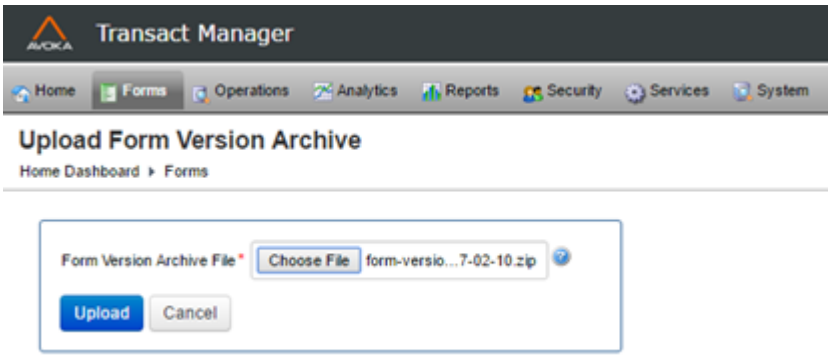
1. Navigate to the Transact Manager server that you wish to import the form into and log in.
   Logging in will redirect you to the Home Dashboard.
2. Select the *Forms* menu then click *Forms*.
   This selection will display the forms page. The forms page displays every form has been uploaded to the TM server that you have chosen. The screenshot below illustrates this selection.



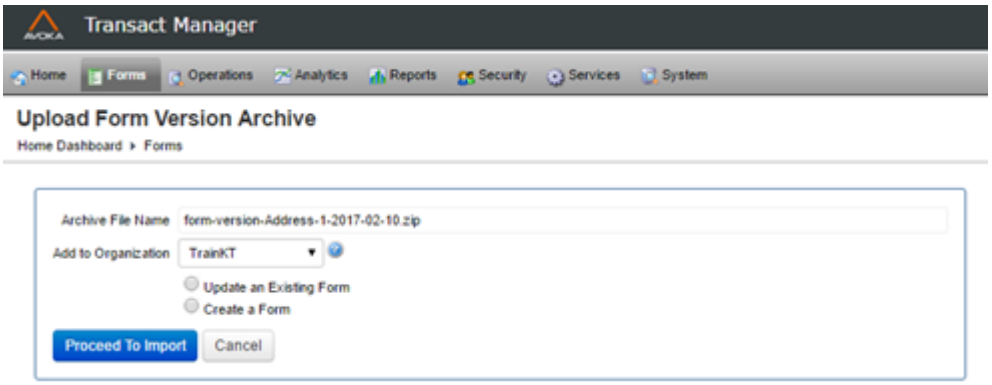
3. Select the *Import Form Version* button.
   The screenshot below highlights this selection.



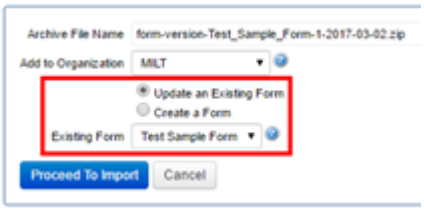
4. Select the *Choose File* button.
   Selecting this button will allow you to browse for and select the downloaded zip file. Once you have selected the appropriate zip file, click *Upload*. The screenshot below illustrates these selections. Successful upload of the zip file will display the Upload Form Version Archive screen.



5. Select the appropriate organization (if relevant) for the form from the dropdown menu. Ensure that you select the SAME organization where you originally uploaded the form that you are updating.

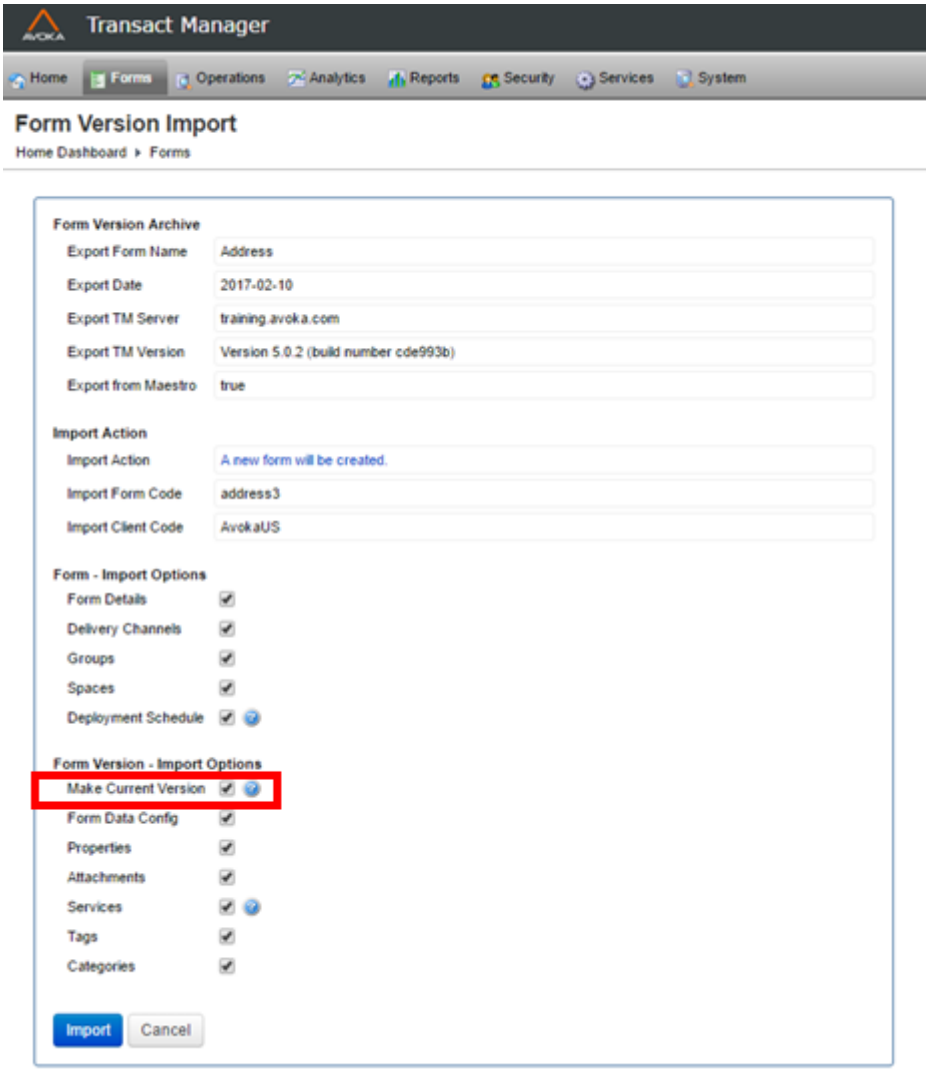


6. Select *Update an Existing Form* from the available options.
   - **Update an Existing Form** – Selecting this option will display the existing form dropdown menu. From the dropdown menu, select the name of the form that you are updating and uploading to the TM server. As you are updating a form that has previously been uploaded to the Transact manager server, the name of the form will appear in the list of choices (as long as you have selected the correct organization). The screenshot below illustrates this option.

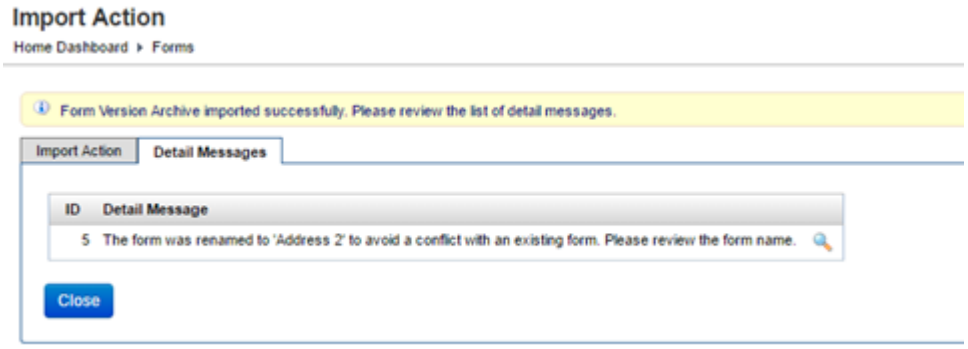


7. Click *Proceed to Import*
   Clicking this button will display the options screen for the form. It is on this page that you can select or unselect any options that you may want to override. Ensure that you select the option, 'Make Current Version', if you want to make the form that you are uploading the "current version" of the form (i.e. the form that your form users will be completing). The screenshot below illustrates the options screen and highlights the 'Make Current Version' option.

> When exporting a form directly from Maestro, it is important that you unselect the following options to ensure that you do not override any configurations in your local TM:
>
> - Delivery Channels
> - Groups
> - Spaces



8. Select *Import*
   Selecting import will officially import the form into the TM server environment and display the summary screen. Note that If the *Create a New Form* option was selected during the import process, and there was a form naming conflict (i.e. a form with the same name already existing on the server), the form will be renamed, and a message will be displayed in the Detail Message tab. The screenshot below illustrates a Detail Messages screen displaying a form name error.



9. Select *Close*
   Selecting close will redirect you back to the forms page.

Update TM Form Version (for TM versions 5.0.6 and higher)

Unknown macro: 'redirect'

Page Contents:

Related Pages:

- Build TM Form Version
- Update TM Form Version
- Update TM Form Version (for TM versions 5.0.6 and higher)

This option in only available to environments using Transact Manager 5.0.6 or higher.

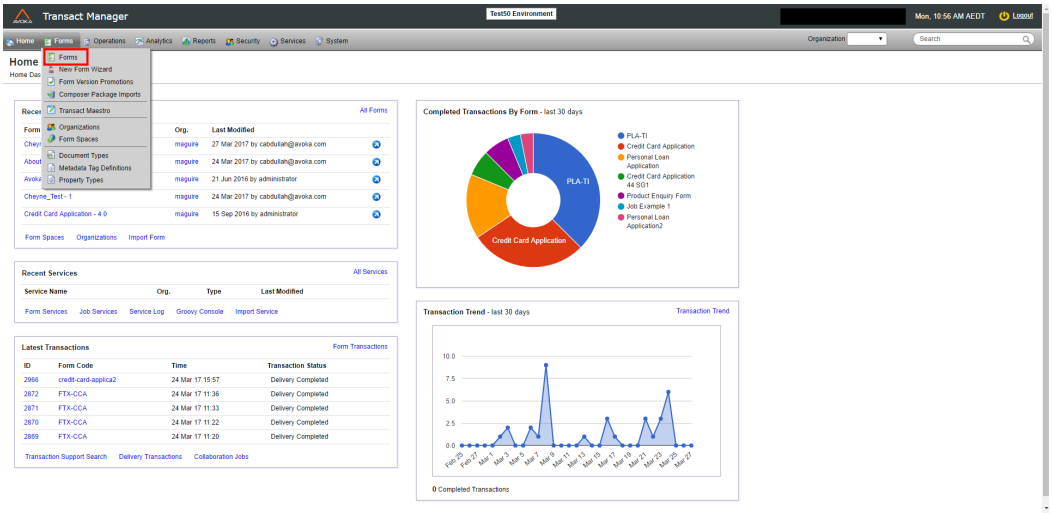
For on-premise and Amazon Virtual Private Cloud (VPC) [if the VPC is not managed by Avoka] TM installations, in order for the updating process to function correctly, the Transact Manager environment needs to send an outbound request over HTTPS to a Maestro host (e.g. maestro.avoka.com.au). For this reason, it is recommended that before you attempt any updating of a TM form version using Maestro, you first check your firewall rules and ensure that they allow this outbound request to be made.

Overview

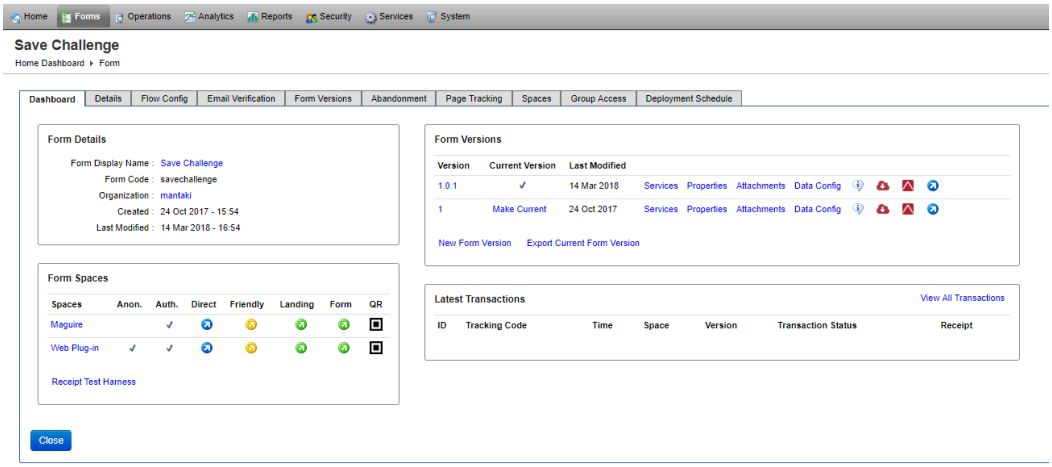
Once a form has been uploaded and imported into a TM server, it is possible that the Maestro form will need to be updated. In previous versions of Transact Manager, publishing an updated form to a TM server required form builders and TM administrators to repeat the entire process of building the TM form version and uploading it to the TM server (through the zip file). As of version 5.0.6 of Transact Manager, this is no longer the case, and a more streamlined approach to updating Maestro forms to TM has been implemented.

The steps below identify and describe the process of updating a Maestro form that has already been imported to a TM server (that is version 5.0.6 or higher).

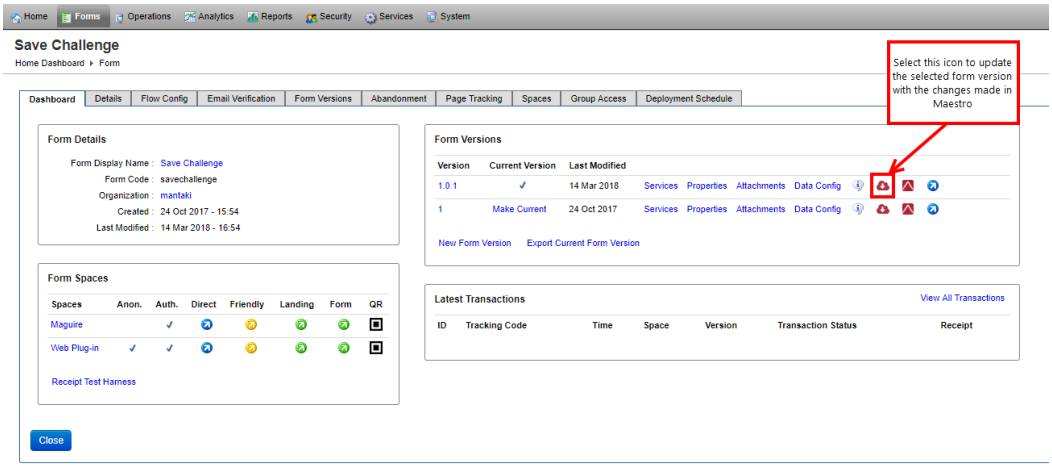
1. Update the form in Maestro
2. Build the TM form version in Maestro
   You can use any of the build options for this process, however clicking the Build button (not the dropdown) will build the TM form version without downloading or rendering the form in a new tab.
3. Navigate to the TM server where you originally uploaded the zip file and log in.
4. Select the *Forms* menu and then click *Forms*.
   Selecting *Forms* will direct you to the forms page and display a list of all the forms that have been uploaded to the TM server.



5. Select the form that you want to update (i.e. the form that you just updated in Maestro and had previously uploaded to the TM server).
   Selecting the form will display an individual dashboard displaying information about the form. The individual dashboard for a form is displayed in the screenshot below.



6. Select the *Update from Maestro* icon next to the form version you want to update.
   The screenshot below illustrates the icon and highlights this selection. Selecting this icon automatically updates the form from the Maestro server.



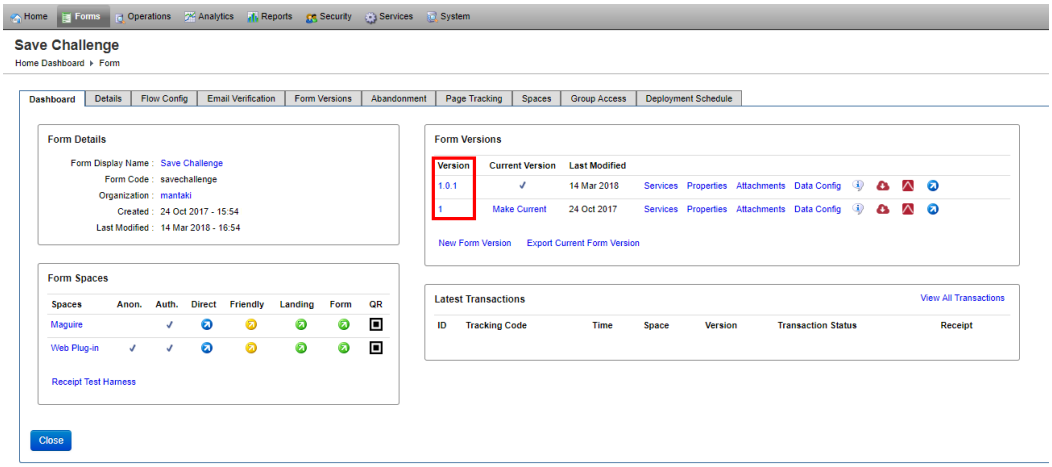
> If you receive any errors when completing this step or the update process does not function correctly, please double check that your firewall settings are correct and that they are allowing the outbound request over HTTPS to be sent.

Alternative Method

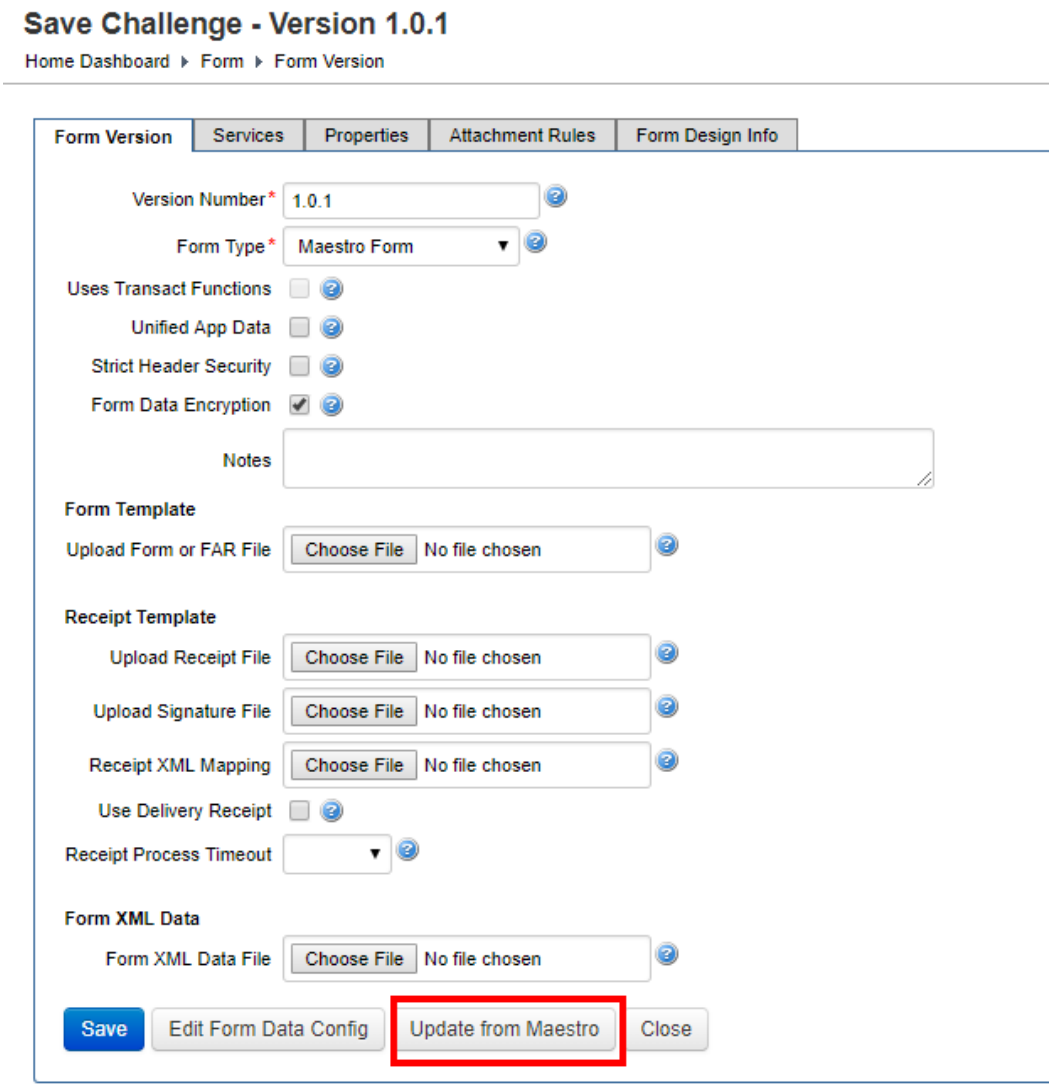
The alternative method of updating a form from Maestro occurs at step 6 (above). Instead of selecting the 'Update from Maestro' icon, select the version number.

Follow steps 1-5 above, then continue with the steps below.

1. Select the most current version of the form (i.e. the form version that you just updated in Maestro) from the 'Form Versions' section of the form's individual dashboard.
   Selecting the version will display information about that version of the form. The screenshot below illustrates this page and highlights this selection (in the screenshot below the current version is version 1.0.1).



2. Scroll down the page that is displayed and select the *Update from Maestro* button. Selecting this button will update the form version in the TM server. The screenshot below illustrates this page and highlights the 'Update from Maestro' button.



3. Select *Save*

Working with Transact Insights

Unknown macro: 'redirect'

Configuring Insights Milestones for Maestro Forms

Overview

A Milestone in the Transact Platform is a specified event that occurs in a transaction. For a milestone to occur, it must be identified during the creation of a Maestro form. Once a milestone has been identified and configured in Maestro, the Transact analytics tool Insights can collect, analyze and display data based on the milestones achieved across a selection of transactions.

A common example of where a milestone can be used in a Maestro form is when an applicant prefills data using the LinkedIn prefill exchange component. In this example, the moment the applicant clicks the, “Use This Profile” button, a click rule will initiate, and the milestone will be recorded by Transact Insights. This milestone documents that the applicant has used the LinkedIn prefill component to prefill data in the form.

It is important to note that once a milestone is sent, it will forever be recorded against this transaction in Insights. So, it is recommended that such milestones be immutable within the context of a transaction. Milestones should NOT be sent based on user input data if it is possible for that data to be changed in the course of that transaction.

Configure Milestones

Milestones should be configured using programmatic logic. For instance, if x is clicked, send milestone y. The screenshot below displays an example of the logic required to identify and configure a milestone in a Maestro form.

A script to run when the user clicks on this form item

```
1 Form.items.linkedinSignIn.$populateFields();
2 if(item.$$component.properties.insightsLinkedinPrefill === true) {
3   Insights.milestone("LinkedIn Prefill");
4 }
```

In the above example, when the LinkedIn prefill component is clicked, the “LinkedIn Prefill” milestone is initiated, recorded and sent to Insights. This type of milestone logic can be applied to any component in the Avoka Transact catalog. The logic to send milestones may also be included in any script in the form.

Applicant vs Review

Insights’ milestones are a great way to distinguish between transactions made by applicants and transactions made by reviewers. Across the Transact Platform, it is common practice that once an applicant has completed a form, the created transaction will experience some kind of review conducted by someone other than the applicant. In Transact Manager, the initial form completion (by the applicant) and the review are considered two separate transactions. In most cases, you will only want to consider and analyze transactions made by applicants. To address this issue, you can configure Transact milestones to identify when a transaction is an, “Applicant Transaction” (transactions initiated by applicants) and when a transaction is a,“Review Transaction” (transactions initiated by reviewers/someone other than the applicant).

The screenshot below displays the logic required to identify and configure applicant and reviewer milestones.

a script to run when the form loads

```
1  if (data.prefillApplicationType != "") {
2    data.applicationType = data.prefillApplicationType;
3  }
4  var dobItem = Form.items.dateOfBirth ;
5  dobItem.$hasValue = function(data){
6     return Util.children(dobItem).some(function(child) {
7        return !child.exLayout && !Util.isBlank(data[child.id]);
8     });
9  };
10 if (Form.isStep("Application Review")) {
11     Insights.milestone("Review Transaction")
12 } else {
13     Insights.milestone("Applicant Transaction")
14 }
15
```

In the above example, when the form is loaded, a milestone is sent to Transact Insights. This milestone will vary depending on the step that the form enters on loading. If the form is at the “Application Review” step, the,“Review Transaction” milestone is sent to Insights. If the form loads on any other step other than, “Application Review”, the “Applicant Transaction” milestone is sent to Insights.

Milestone compatibility

Product	Version
Transact Manager	No restriction
Maestro	From 5.1.x

Standard milestones

The following is a list of standard milestones used by Insights.

Milestone	Description
Opened	The transaction is created by the user but no other activity or interaction has been recorded as yet.
Resumed	The transaction is resumed after the user previously saved it.
Started	The transaction is started by some activity or interaction with the page, such as entering data in a field or clicking in a checkbox.
Saved	The transaction is saved by the user.
Submitted	The completed transaction is submitted by the user to Transact Manager.
AbandonedBounced	The transaction is abandoned in Transact Manager by the user leaving the application, without interacting with it.
AbandonedStarted	The transaction is abandoned in Transact Manager by the user leaving the application, after starting it.
AbandonedCancelled	The transaction is abandoned in Transact Manager by the user leaving the application, by canceling it.
AbandonedIneligible	The transaction is abandoned in Transact Manager by the user leaving the application, as it is deemed ineligible by, for example, a groovy script.
AbandonedSaved	The transaction is abandoned in Transact Manager by the user leaving the application, after saving, but never returning to it.
AbandonedSubmitted	The transaction is abandoned in Transact Manager by the user submitting the application, but it was not completed due to a post submission delivery service not completing.

Domain Models

Unknown macro: 'redirect'

This section of the Maestro documentation introduces and describes the concept of Entities and Domain Models in Maestro. Domain Models in Maestro are made up of entities.

- Shared Domain Models

More information on Entities can be found in the Understanding the Component ID section of the documentation.

Shared Domain Models

Unknown macro: 'redirect'

Why Use Domain Models

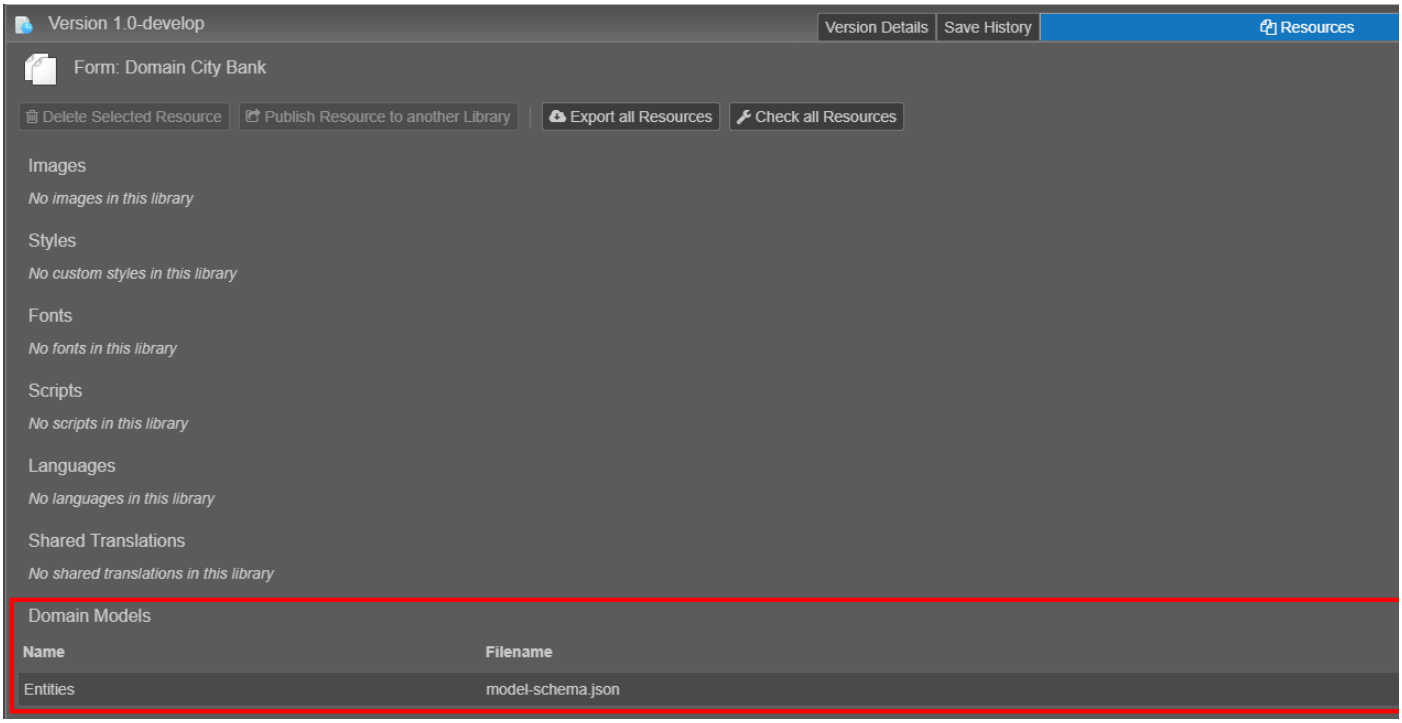
- Helps to focus on what the form does and how it does it.
- Provides Maestro with the ability to be more Object Oriented Programming, rather than Form Oriented Programming.
- Provides an alternative view on form design and development when data structure is used rather than form field elements.
- Provides Groovy developers with the ability to use domain models for data mapping.
- Helps to build two forms that consume the same data (e.g. collaboration jobs).
- Allows binding of fields to data while importing JSON.
- Allows retrieval of application data as an Object using an API.
- Makes it easier to look at the model structure to understand the logic behind the form.
- Allows the creation of rather complex rules that use relationships between several fields on a form, by using entity mapping and relationships as objects.
- Helps to use long field names, which are difficult to read, by using structured entities. This structure is created by using an underscore "_".

When a new form is created in Maestro, a domain model is created. This domain model is based on a JSON Schema (http://json-schema.org/). JSON Schema is a commonly used standard format for describing data structures. Previously, models were part of a Maestro form, but with the introduction of Maestro 18.05, domain models are now held in a separate JSON file.

Maestro uses limited denormalized JSON schema subset, so you will not be able to use all the features available from JSON schema, namely, the ability to create a type $ref (e.g. an applicant address type).

If a 17.10 Maestro form is imported into an 18.05 environment (which has entities configured), the entities configured in the form will be extracted and automatically moved to a domain model file.

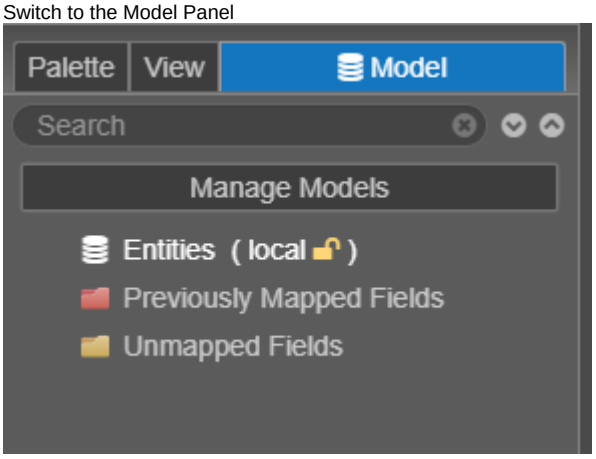
When a new form is created, a domain model file is added as a resource of the form. This model is stored in the resources folder of the form version and can be found by switching to the Resources tab of the form version. The screenshot below highlights the Domain Model JSON file in the Resources tab.



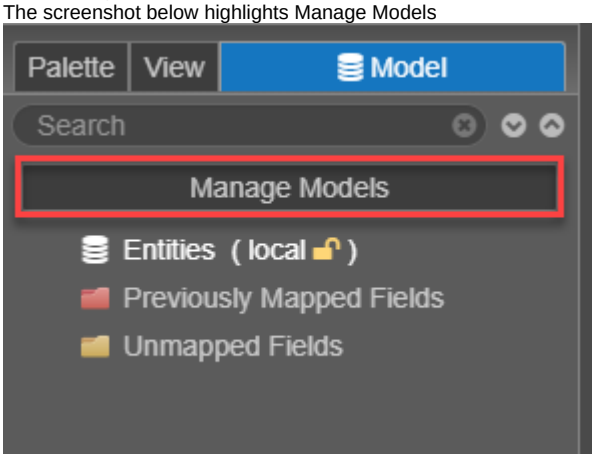
Create A Domain Model (based on a JSON or XML Seed File)

The steps below describe the process of creating a Domain Model from a JSON or XML Seed file.

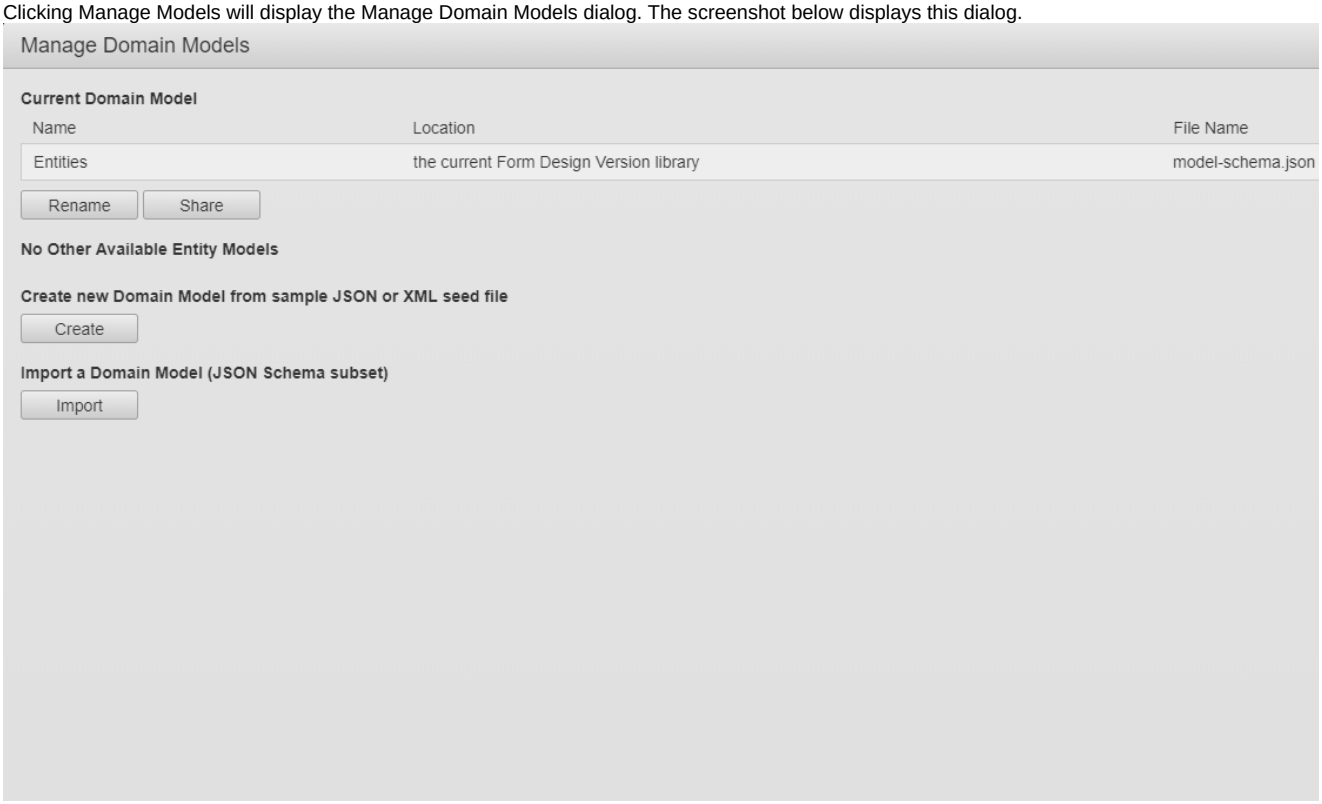
1. Switch to the Model Panel



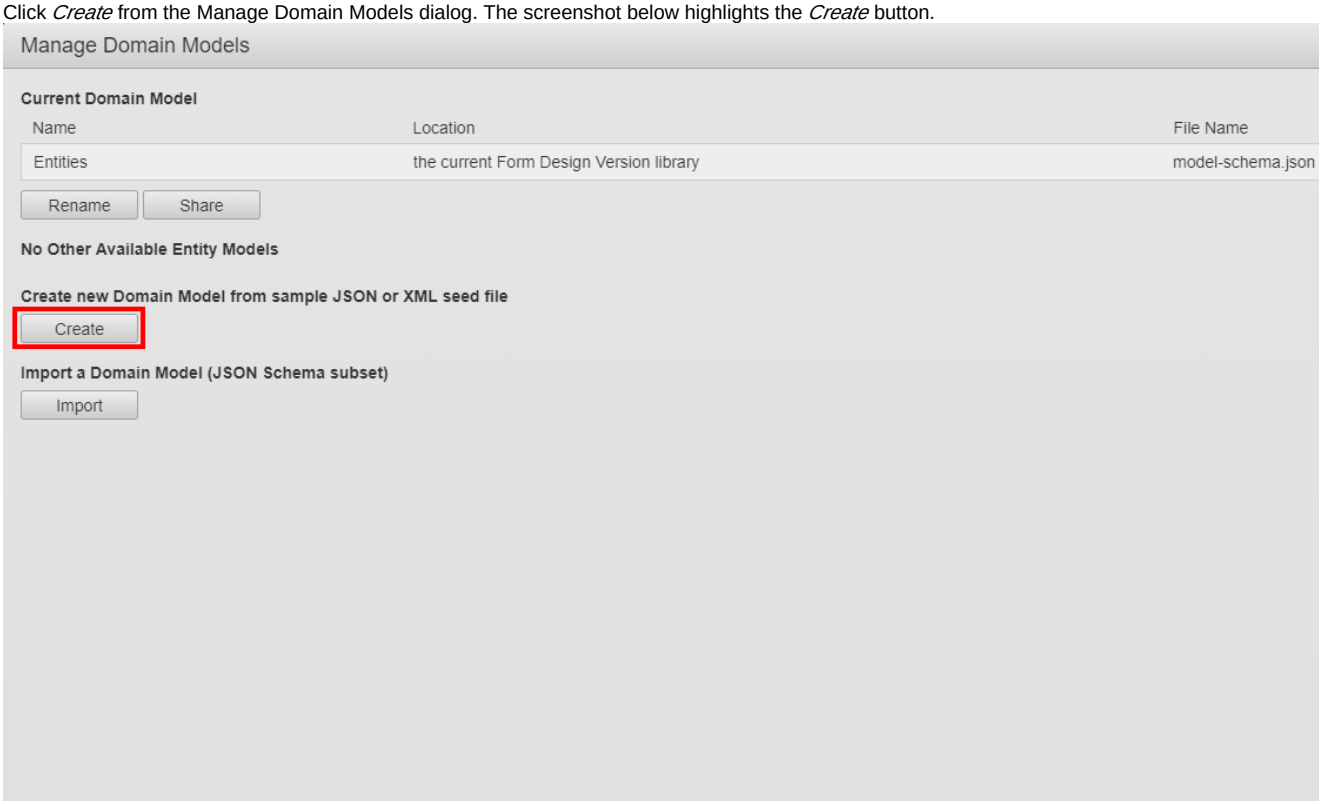
2. Click Manage Models
   The screenshot below highlights Manage Models



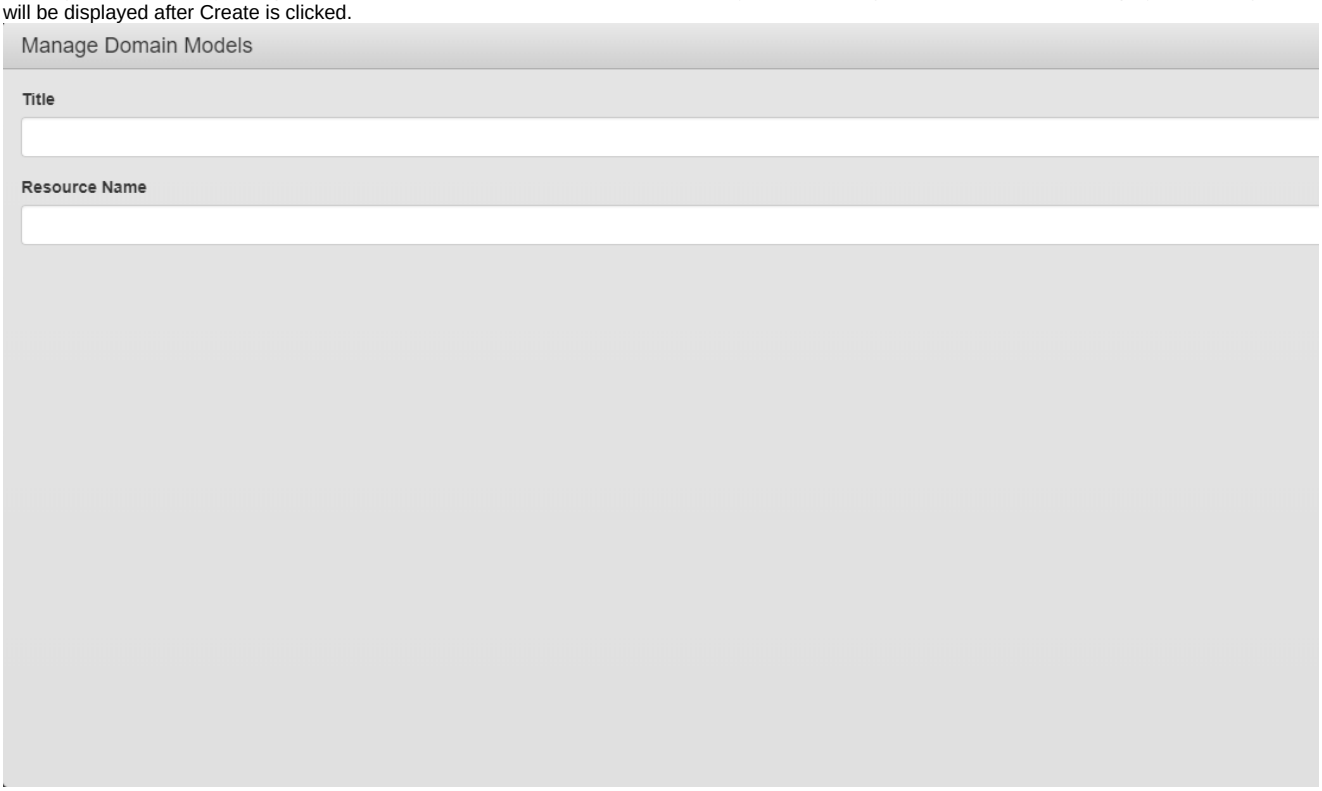
Clicking Manage Models will display the Manage Domain Models dialog. The screenshot below displays this dialog.



3. Click *Create* from the Manage Domain Models dialog. The screenshot below highlights the *Create* button.

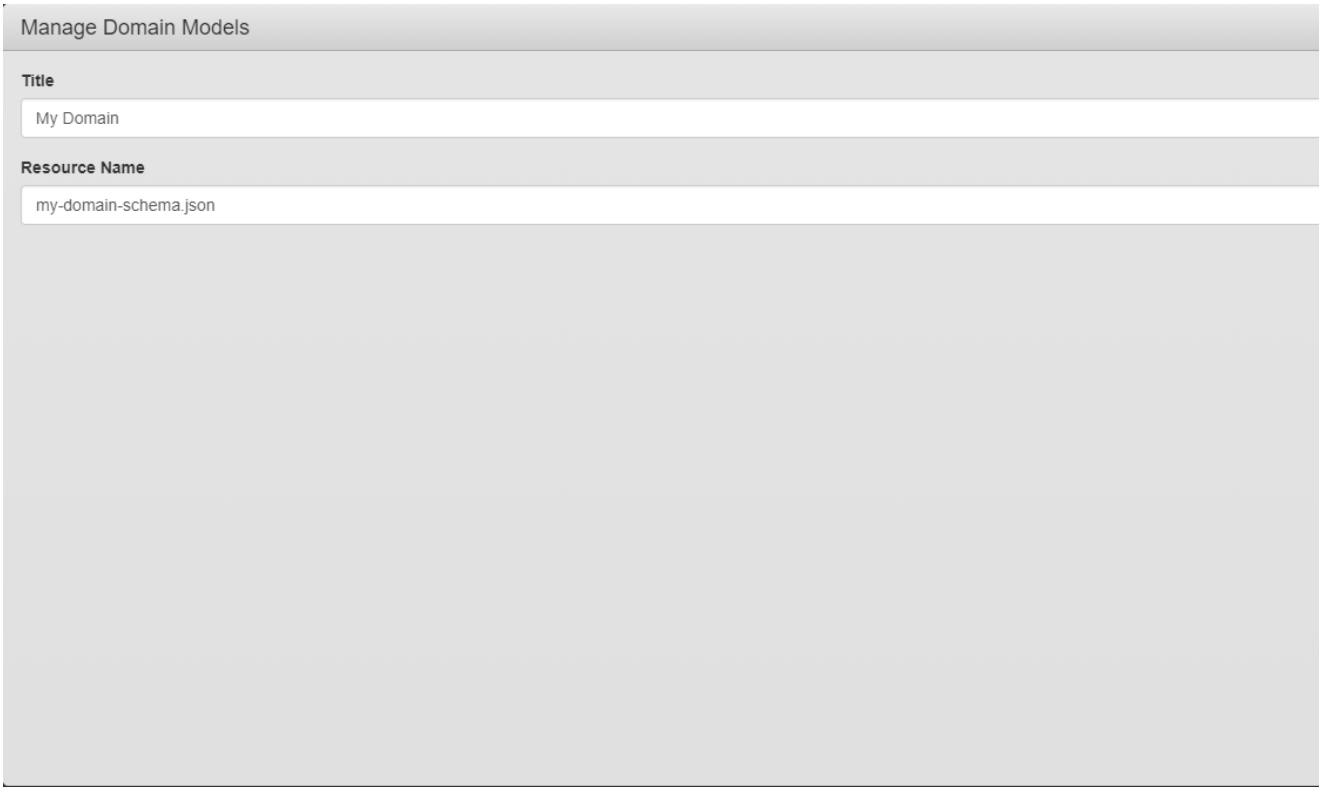


Clicking *Create* will ask you to provide a title and resource name for the model that you are creating. The screenshot below displays the dialog that will be displayed after Create is clicked.



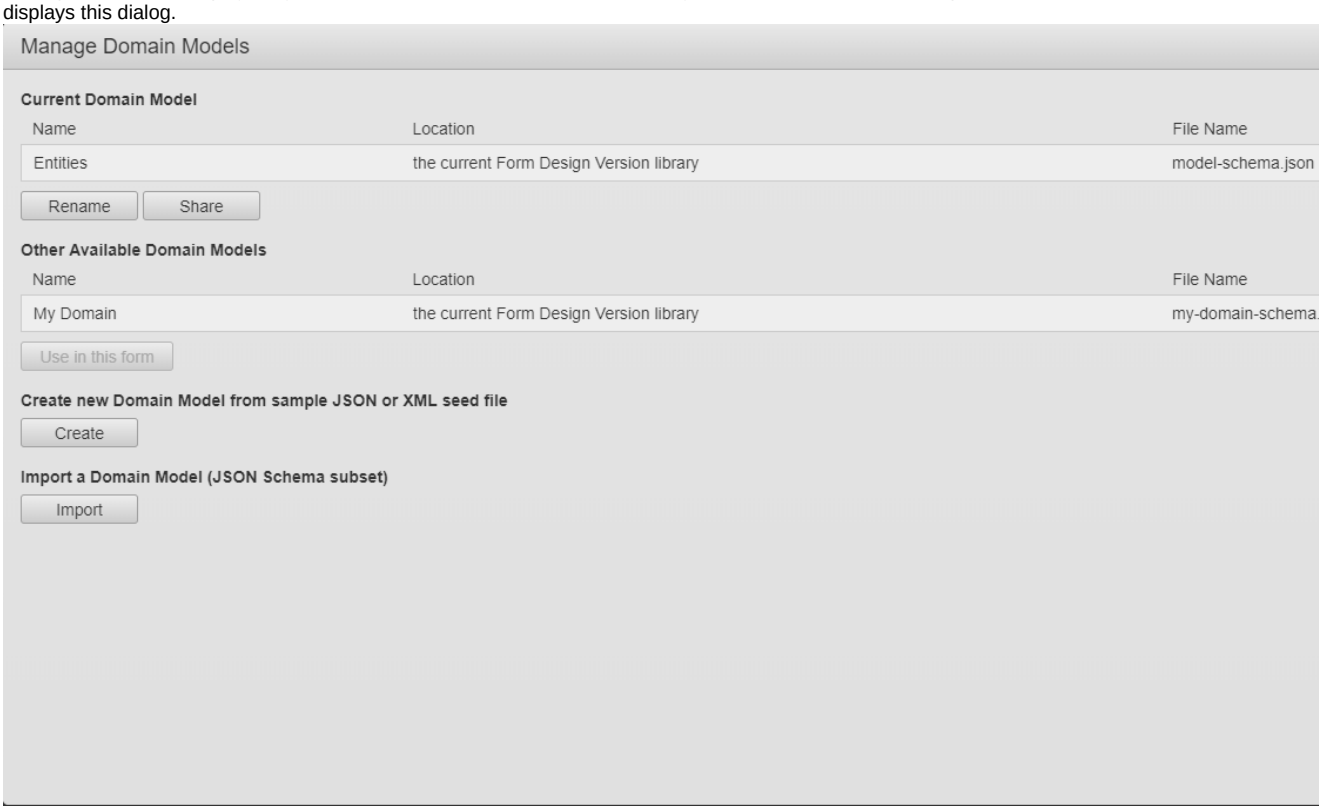
The title is the unique identifier of the model, and the resource name is the actual filename that will represent the imported Domain Model.

4. Enter a title for the Domain Model. Once a title has been entered into the Title field, the Resource Name field will populate automatically with the provided title and a -schema.json file extension added. The screenshot below displays an example of a Title (My Domain) and Resource Name (my-domain-schema.json).

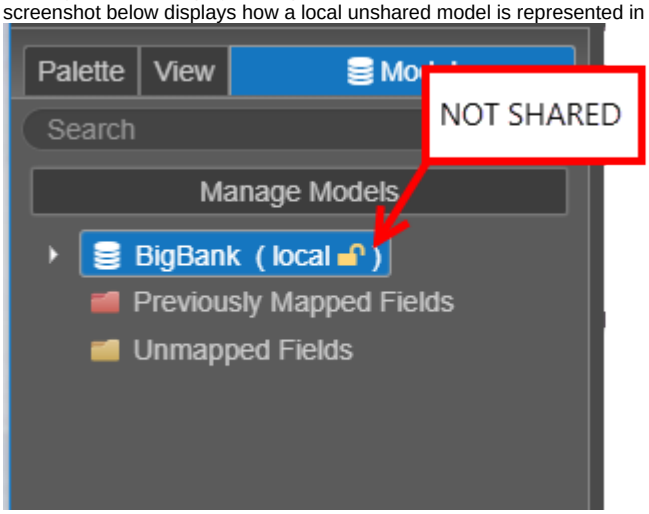


5. Click *Import*
   Clicking *Import* will direct you to select the model from your device.

6. Select the model file from your device. Maestro currently supports JSON and XML file formats for creating/importing (all imported models will be converted to the JSON file format).
   Selecting this file from your computer will direct you to the Manage Domain Models Sharing Overview dialog. This dialog provides an overview of the imported model, displays any other available domain models, and allows you to rename or share the imported model. The screenshot below displays this dialog.



7. Click *Close*
   Clicking *Close* will display the imported Domain Model in the Model Panel. By default, the Domain Model will remain local to the form. The screenshot below displays how a local unshared model is represented in the Maestro Model Panel.



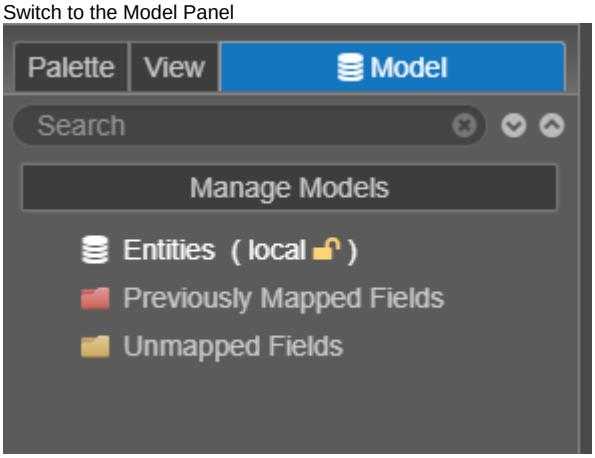
Import a Domain Model (based on a JSON Schema subset)

> A Domain Model can be exported from a form by using the "Export all Resources" functionality.

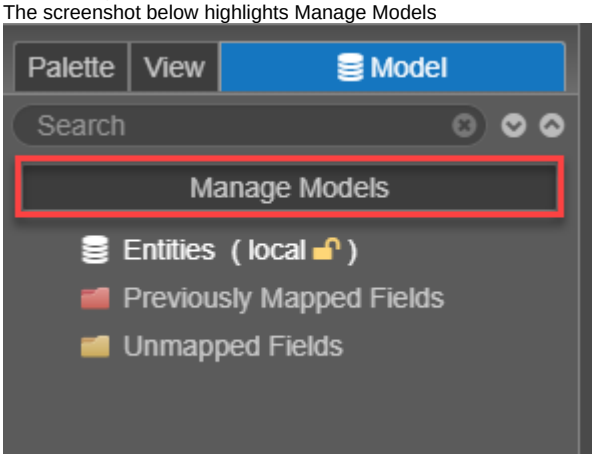
Maestro supports a limited subset of JSON schema.

To import a Domain Model based on a JSON Schema subset:

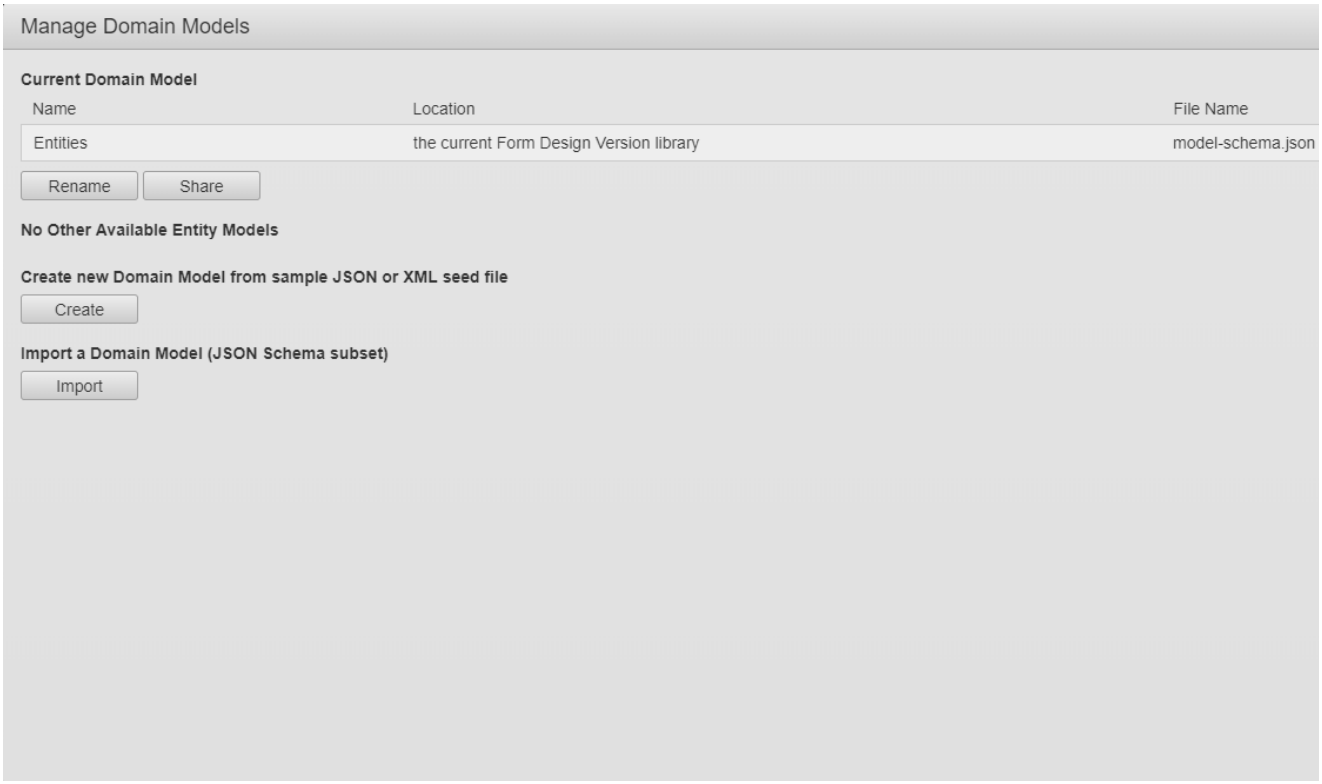
1. Switch to the Model Panel



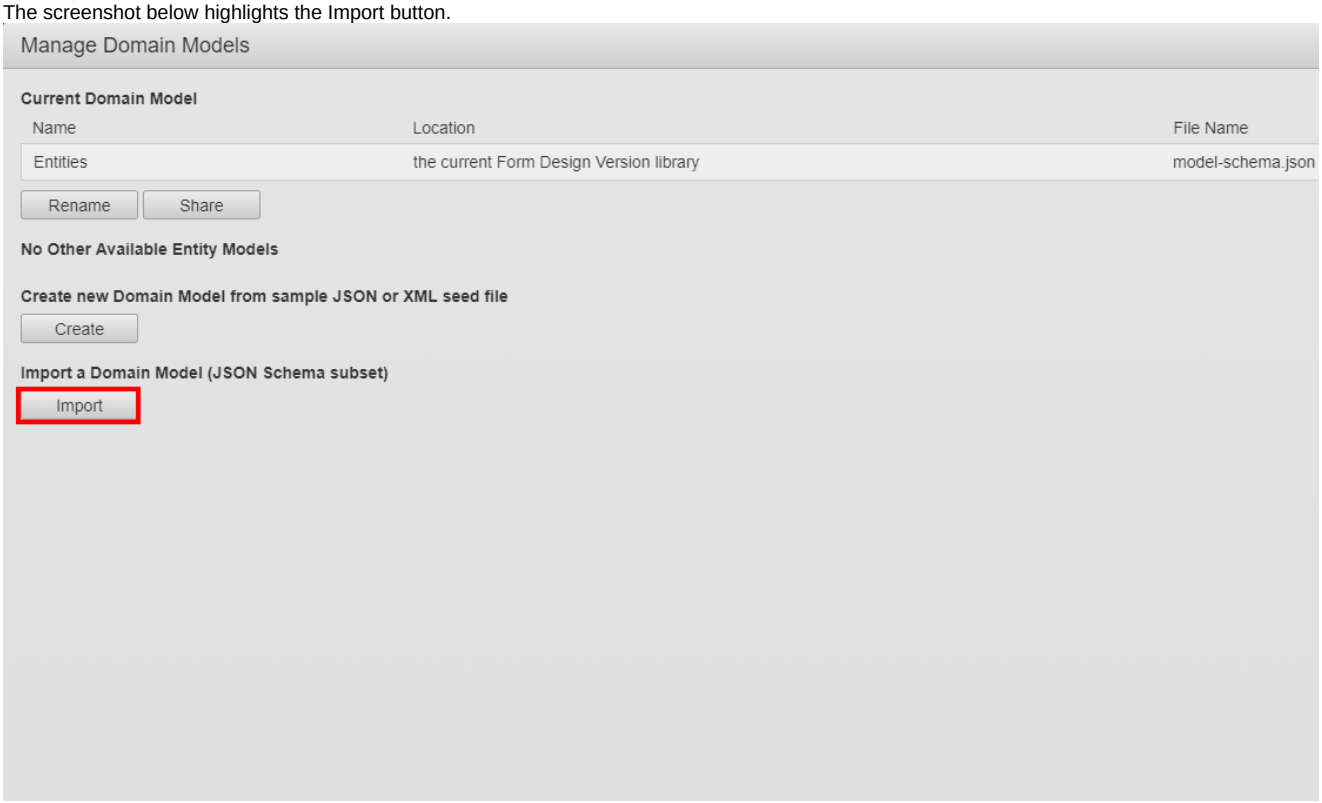
2. Click Manage Models
   The screenshot below highlights Manage Models



   Clicking Manage Models will display the Manage Domain Models dialog. The screenshot below displays this dialog.

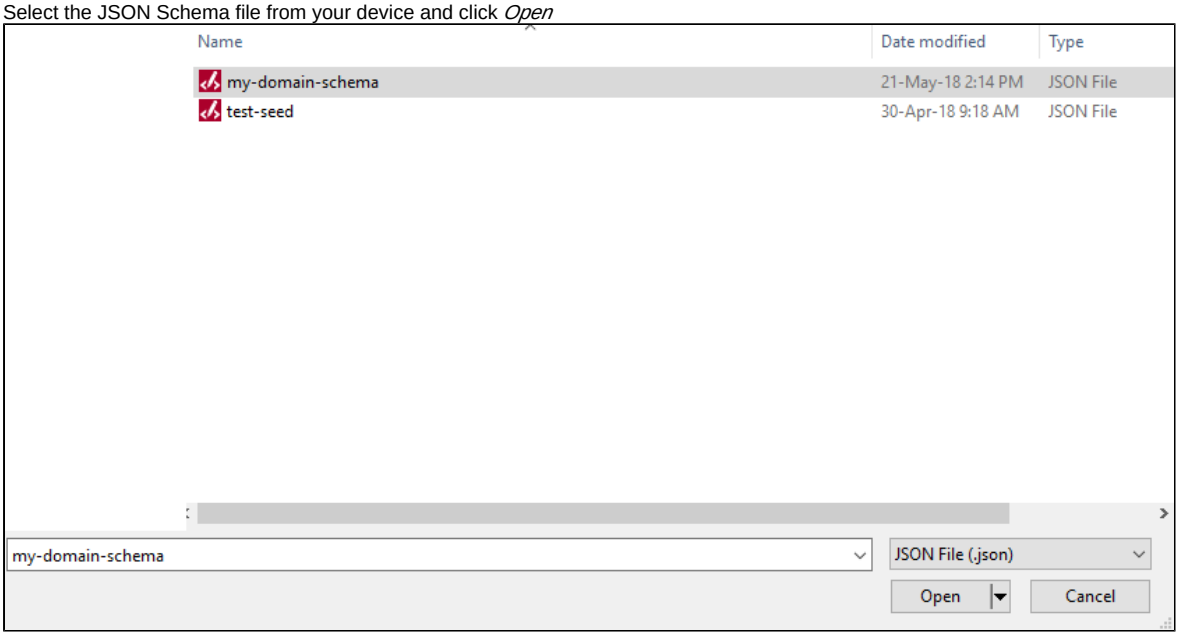


3. Click *Import*
The screenshot below highlights the Import button.



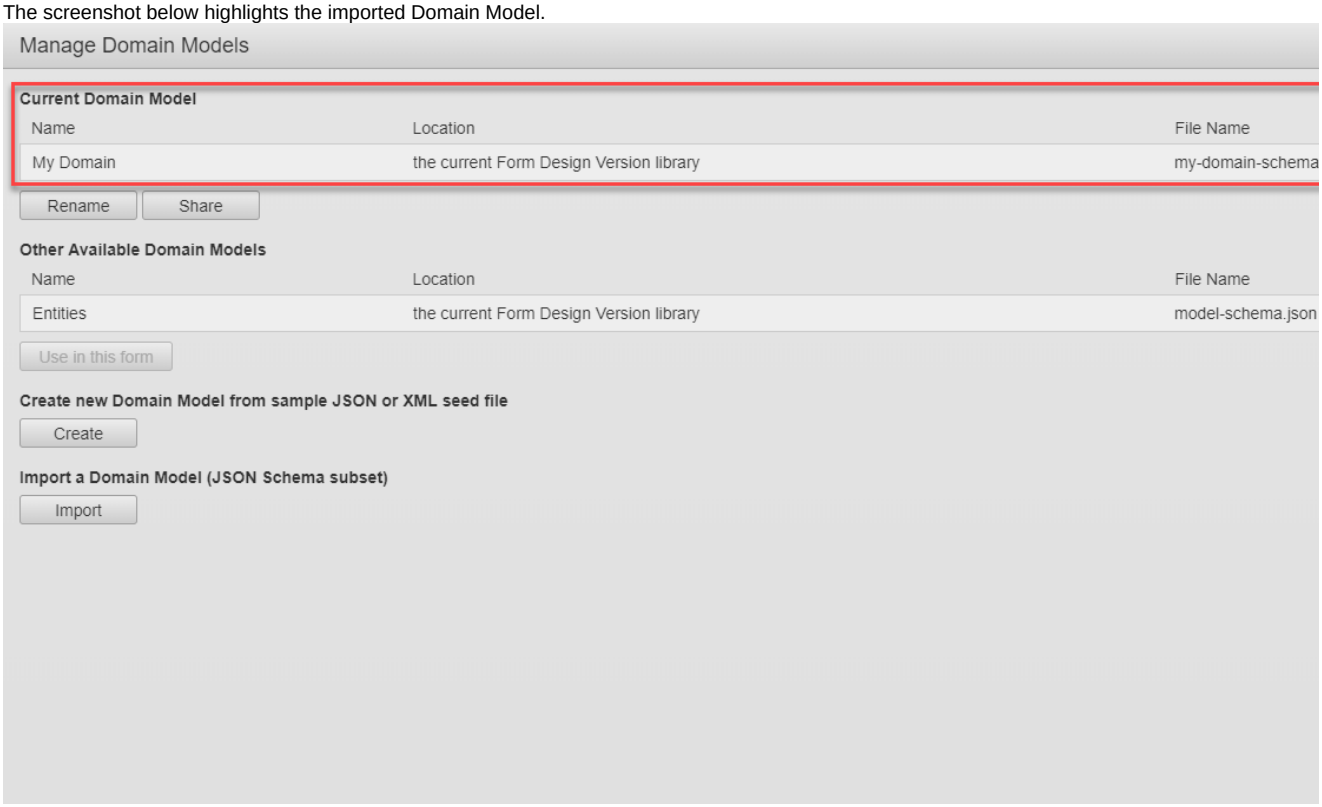
Clicking *Import* will ask you to select the JSON Schema file from your device.

4. Select the JSON Schema file from your device and click *Open*



Clicking Open will open the JSON schema file and select it as the **Current Domain Model** for the selected form.

The screenshot below highlights the imported Domain Model.



Share a Domain Model

The ability to share a Domain Model is only available to users with a Maestro Site Administration role. This authorization has been implemented to ensure that shared Domain Models are only overridden by users with the appropriate role and permissions.

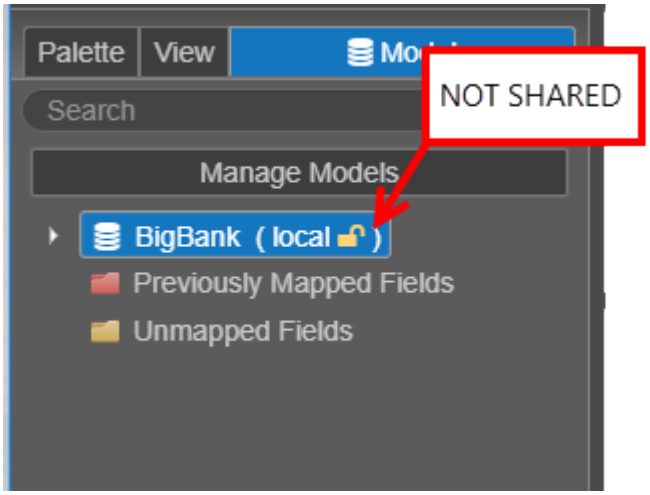
Why Share an Domain Model

A Shared Domain Model can:

- be used in global forms, e.g. collaboration jobs
- re-use existing elements, possibly created by other developers.

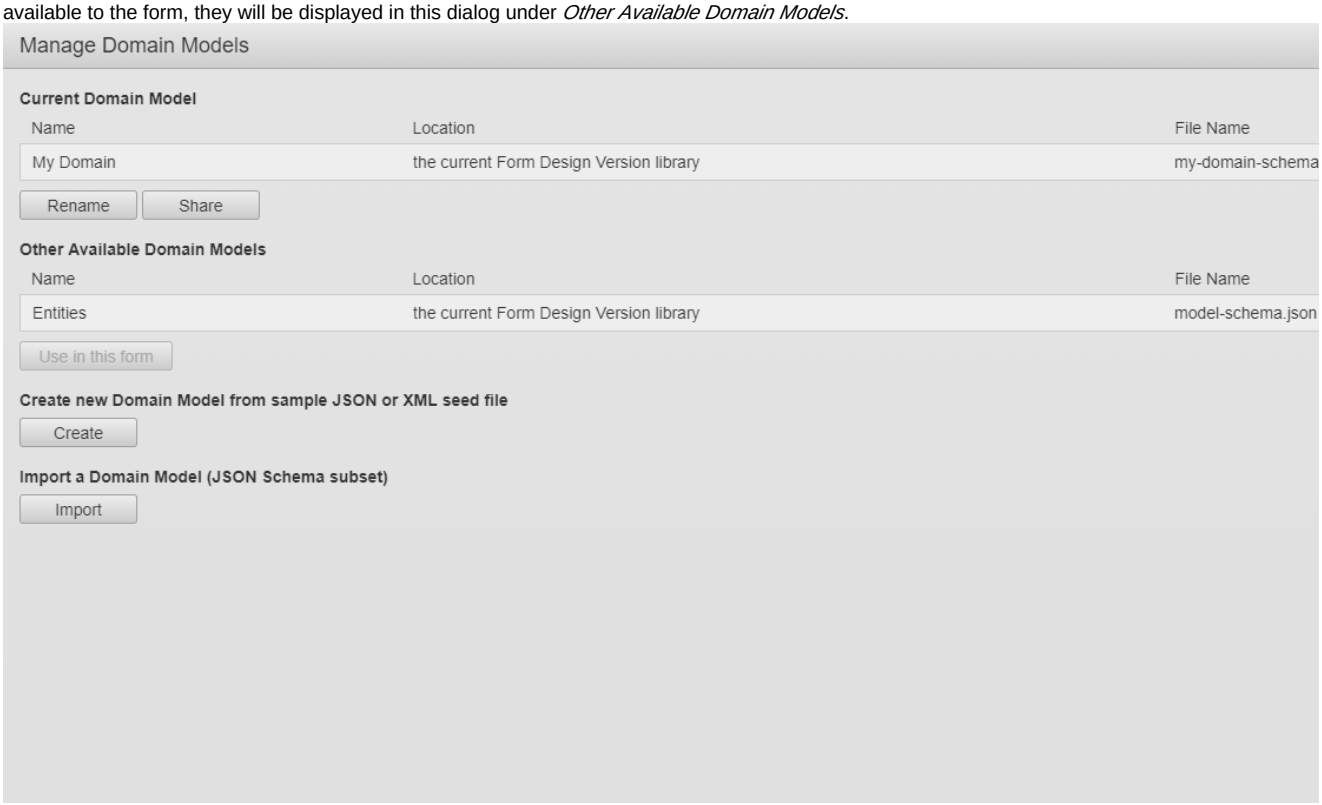
A single Domain Model is created automatically when a new form is created in Maestro. Domain models can also be imported from other sources and shared to a library.

The screenshot below displays how the Model Panel will display when a Domain Model HAS NOT been shared.

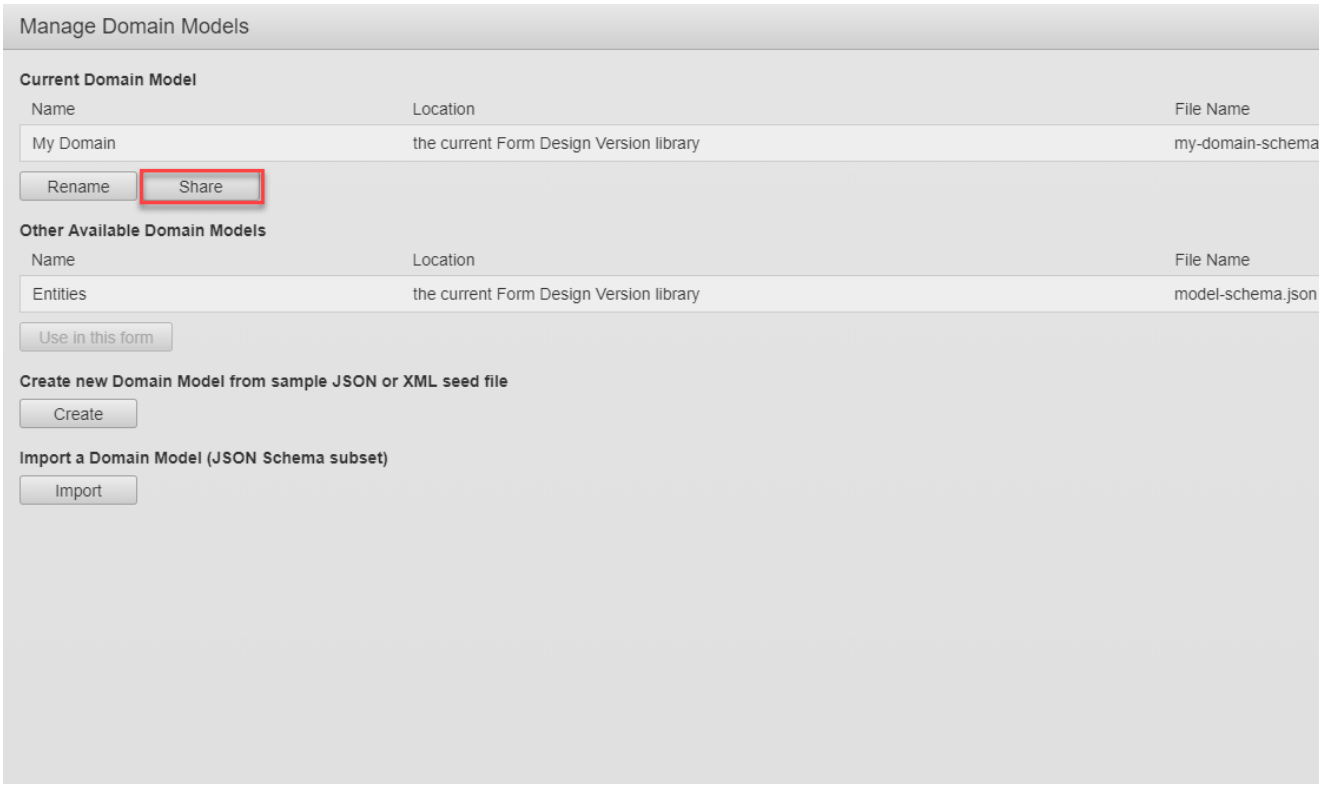


To share a Domain Model:

1. Click Manage Models
   Clicking Manage Models will display the Manage Domain Models dialog. The screenshot below displays this dialog. If any other models are available to the form, they will be displayed in this dialog under *Other Available Domain Models*.



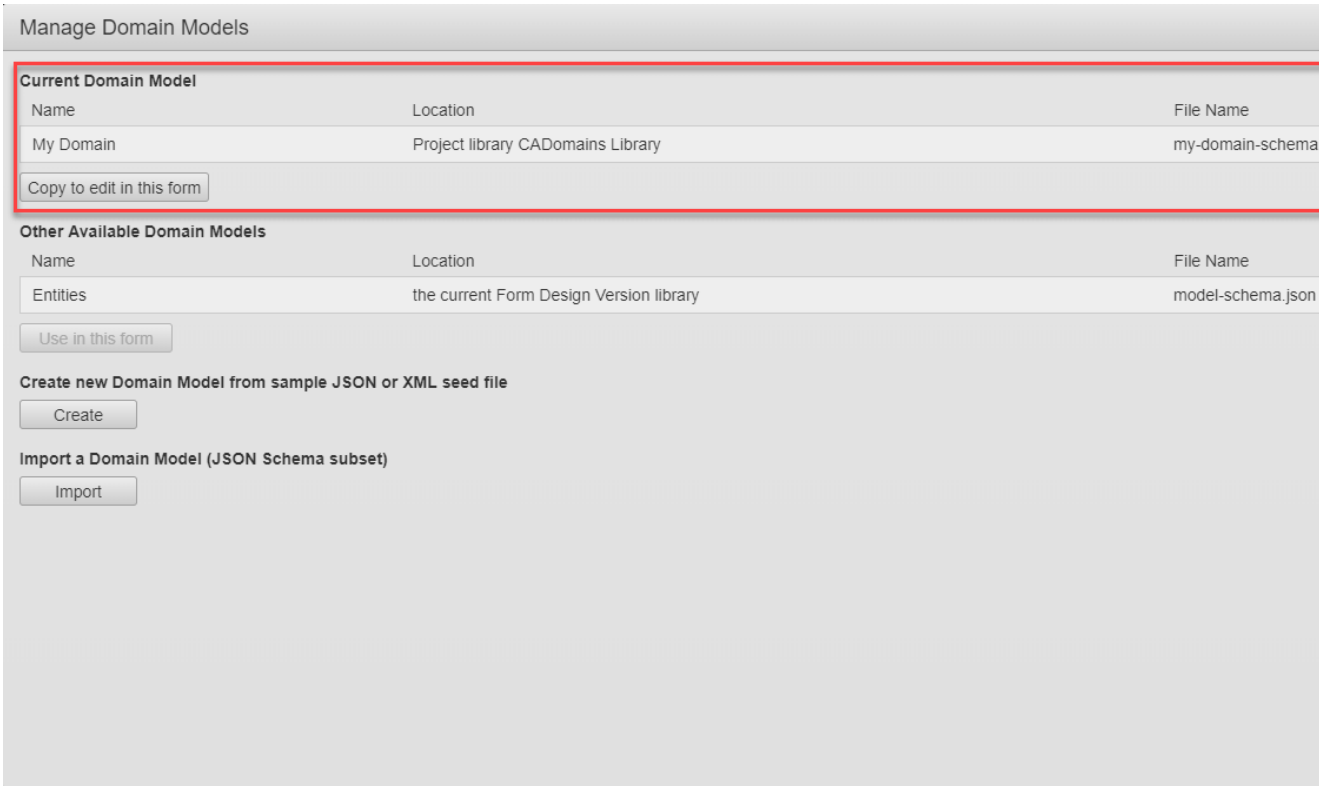
2. Click Share
   The screenshot below highlights the *Share* button.



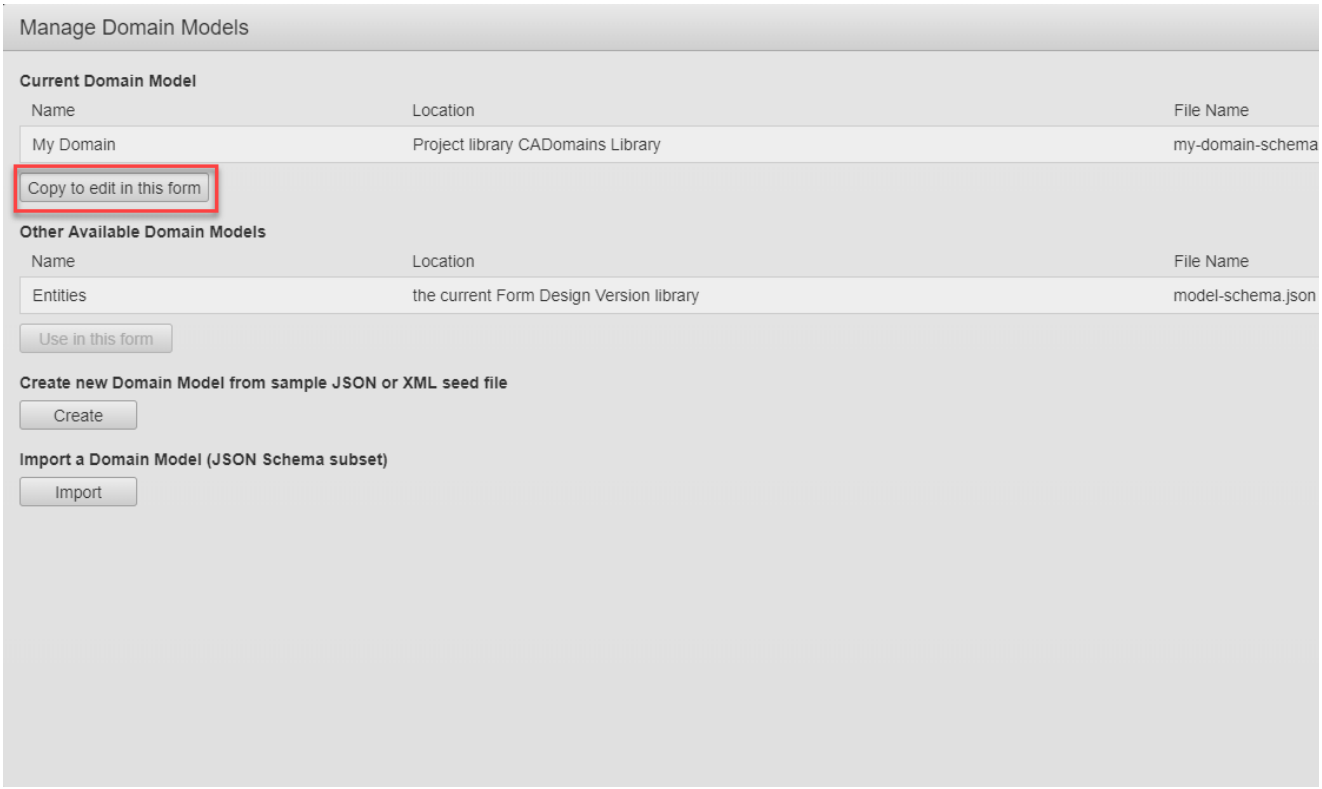
Clicking *Share* will update the display of the Manage Domain Models dialog. The screenshot below displays the updated Manage Domain Models dialog. This updated dialog allows you to provide a new Title and/or Resource Name for the model you are sharing.

By default, Maestro will keep a local copy of the domain model. If you do not want to keep a local copy of the Domain Model, deselect the *keep local copy for further editing* checkbox.

3. Select the library that you want to share the Domain Model to. When a library is selected, all forms with access to that library will have access to the shared model.
4. Click Share
   Clicking *Share* will update the Manage Domain Models dialog to display and identify the library that the Domain Model has been shared to. The screenshot below highlights the library that the imported Domain Model has been shared to.

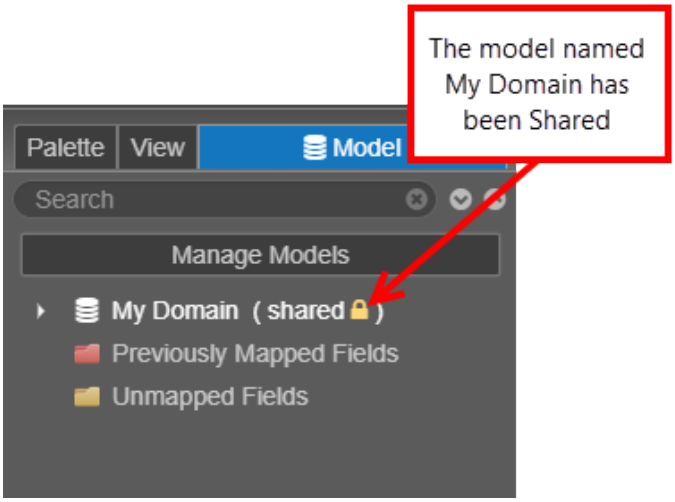


This dialog allows you to copy the model and edit it for use in the form. The screenshot below highlights *Copy to edit in this form.*



This dialog also displays any other Domain Models that are available to the selected form (you can select any of the available models and share them to a library).

5. Click Close
   Closing the dialog will refresh the Model Panel. The screenshot below shows how the Model Panel will display once the My Domain model has been shared.



If you share a model, the lock icon next to the Domain Model title will change from opened to closed (locked). When a Domain Model is shared and locked, all newly created entities within the Domain Model will not be mapped or shared with other forms and libraries.

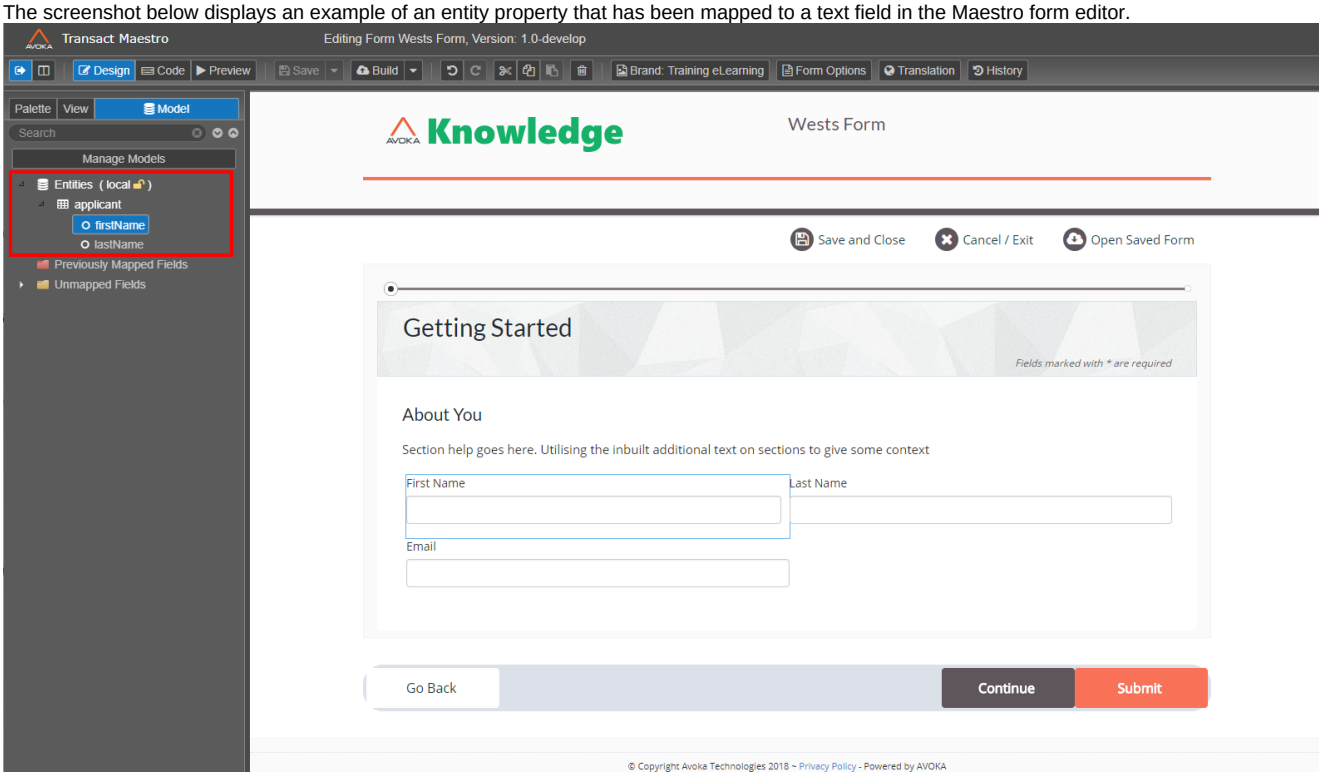
When Sharing a Domain Model, it is important to note that the local copy of the model takes precedence. This means that when you share a Domain Model and select the option to keep a local copy to edit, it is recommended that you rename the local copy of your Domain Model.

Map a Component to an Entity property

Components and Fields in a Maestro form can be "mapped" to an entity property. Mapping a component to an entity property can help easily provide identifiable labels to fields on the form and can help build additional data related tasks to the form.

To create and map an entity property:

1. Select a component (in this example, we will use a text field)
2. Navigate to the Properties Panel and create an entity (in this example, we will name the entity "applicant") and add an underscore with the name of the property that you want to create. In this example, we will create a property named firstName (e.g. applicant_firstName). Completing this step will map the property to the selected component.
   Creating an entity property will also populate the label for the selected component.
   The screenshot below displays an example of an entity property that has been mapped to a text field in the Maestro form editor.

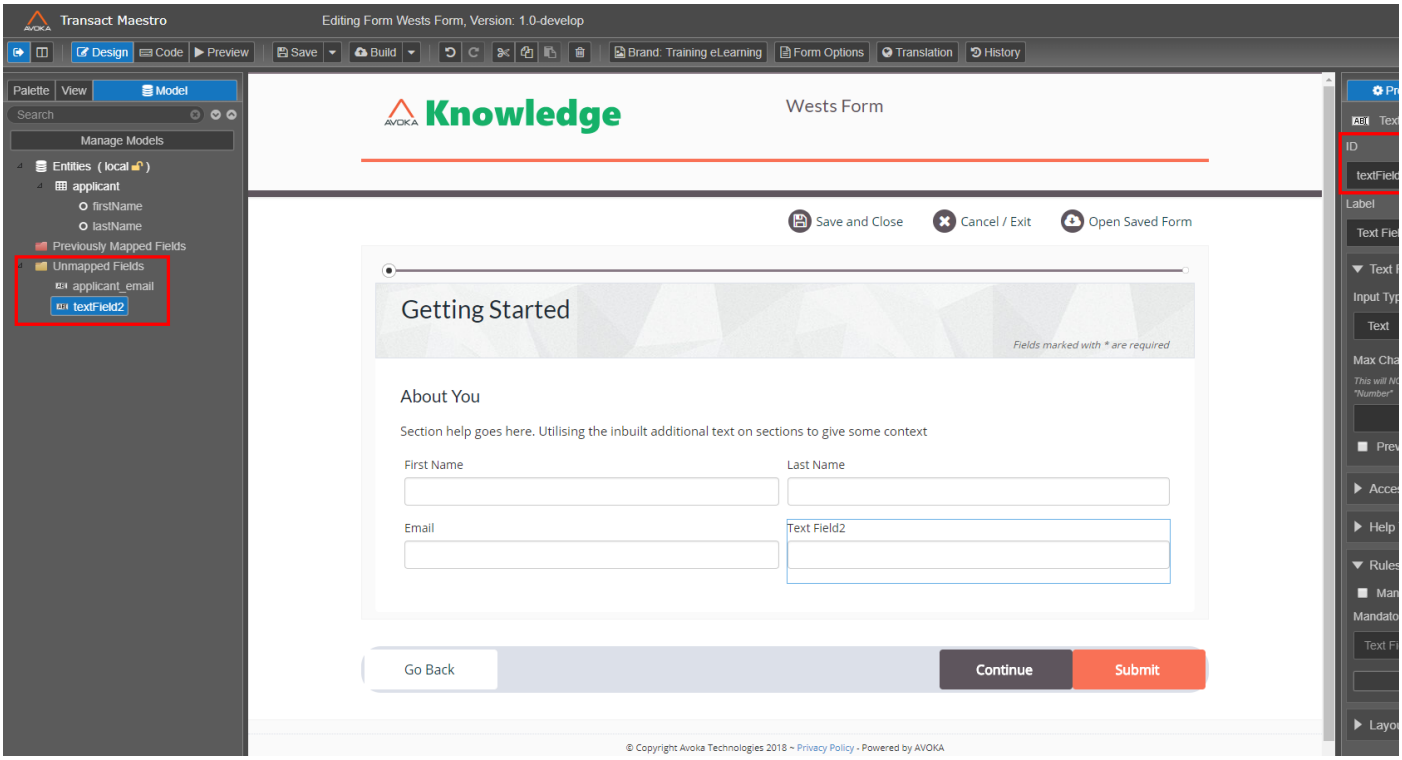


Even if the text field in the above image is deleted, the firstName entity property will still be available to be mapped to another component. This is because the Domain Model file holds all data (including entity properties) about all entities defined in the model.

Unmapped Fields

When new components are dragged onto the form without being mapped to an entity or entity property, they will appear in the **Unmapped Fields** folder of the Model panel. The component (identified by its ID) will stay in the unmapped fields folder until it has been mapped to an entity and/or entity property.

The screenshot below displays the Unmapped Fields folder with a text field that has not been mapped to an entity. Unmapped fields will be highlighted in the Model panel.

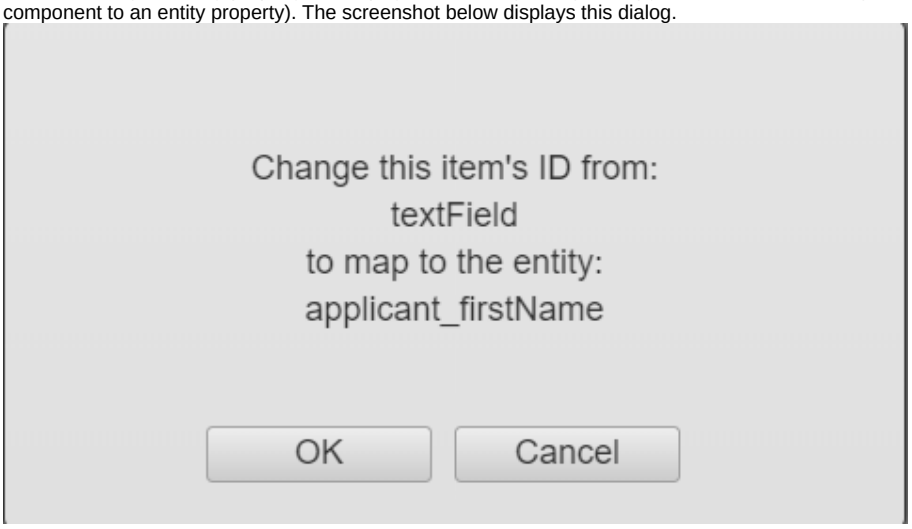


Map components to entity properties that have already been created

When entity properties are created, they are NOT saved within the data of the form, they are instead saved in the Domain Model. This means that even if you delete the component that was dragged onto the form when the entity property was originally created, the entity property will still be accessible to the form and you can map other components to the entity property.

To map a component to an entity property that has already been created in a model:

1. Select the unmapped component on the form
2. Double-click the entity property that you want to map to the selected component
   Double-clicking the entity property will display the following dialog asking you to confirm the change of ID (which is what happens when you map a component to an entity property). The screenshot below displays this dialog.



3. Click *Ok*
   Clicking OK will map the component to the selected entity/ entity property

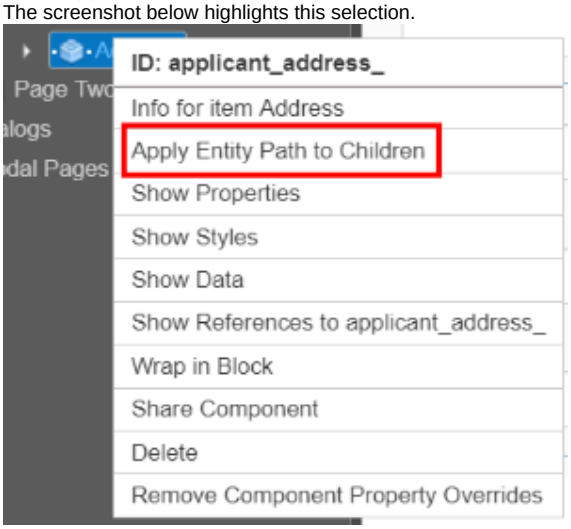
Applying an Entity Path to Children

For components that are made up of multiple components, for example, an address component, you can apply the same entity path to all components that make up the component. For example, if an address is dragged on to a form and an entity named address is created, the state, postcode, street, and child components that make up the parent component can be associated to the address entity.

> For an entity path to be passed off to the child of the component when the entity is created, you must end the entity name with a second underscore. For example, applicant_address_. This underscore at the end gives the component an idea of where it will fit in the broader Domain Model configured in the form.

To apply an entity path to the children of a component:

1. Drag/Select a component that is made up of multiple child components
2. Create an entity with a second underscore (e.g. applicant_address_)
3. Select the component from the View panel and right-click it
   Right-Clicking the component will display the Entity Menu
4. Select Apply Entity Path to Children
   The screenshot below highlights this selection.



Data Configuration

Unknown macro: 'redirect'

The content within this section covers information relating to data configuration in Maestro.

The list below identifies the topics covered within this section.

- The Data Model
- Names and IDs
- Binding
- Resolve Duplicate Bindings
- Submission Data Extract

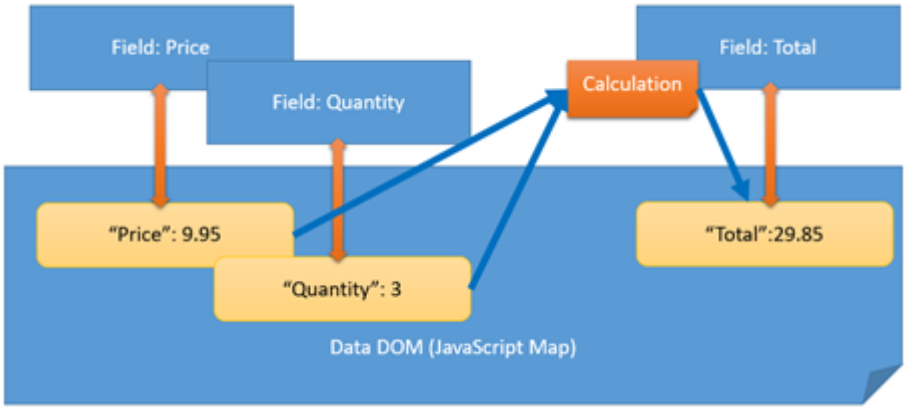
The Data Model

Unknown macro: 'redirect'

When you are developing in Maestro, it is useful (though not necessary) to understand the underlying data model. This data model is a powerful yet simple abstraction that allows much of the functionality within Maestro.

In Maestro, when you type a piece of data into a component, the data is not only displayed in the component itself, but is also copied to an internal data structure known as the data model, or sometimes the data DOM (Document Object Model). The component and its value are sometimes known as the view. These terms both derive from a well known programming pattern known as Model-View-Controller. In our case, the controller is any business rule that we have written.

The Data Model is illustrated in the diagram below:



The field ID (which is unique) is used to identify both the component and its value in the data model. Internally, the data model is implemented as a JavaScript map. You can think of the data model as being a little like a spreadsheet with two columns:

The first column contains the ID of the component, and the second column contains its value.

Field ID	Value
"Price"	9.95
"Quantity"	3
"Total"	29.85

The component's value and the value in the data model are automatically synchronized in both directions. Therefore, when you are programming in Maestro, it is usually much easier to modify the data in the data model, and let the automatic synchronization take care of updating the visible values in the components themselves. If you refer to the above diagram, you will see that the calculations refer to the values in the data model, rather than referring directly to the components themselves.

For example, when you are in the Script Editor, and double-click on the price component, what is actually inserted into your script is "data.price". This is not actually referring to the component itself, but rather referring directly to the component's value in the data model. The "data." is the clue that you are referring to the data model. The values in the data model can be both read and written, and the synchronization will take care of updating all the components in the view automatically.

Data Mapper

You can use the Data Mapper from the Settings menu to easily map an existing client schema (XML output, not full schema) into the form. This is a quick way to establish a form that is compliant to the desired output on submission.

For example, with a model represented in an XML file as so:

<root>

<firstName/>

<lastName/>

</root>

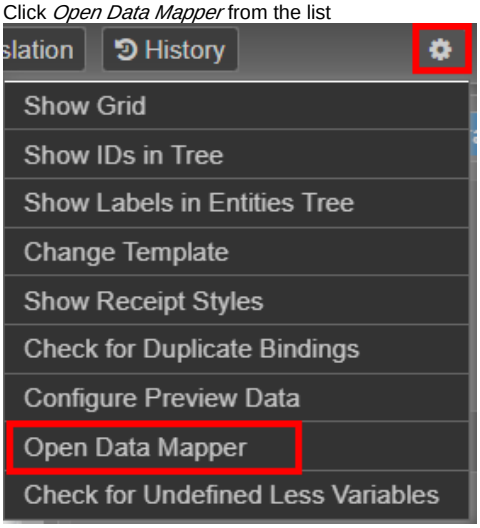
The steps below document the process of using the Data Mapper.

1. Select the *Settings* menu

Related Pages:

- Binding
- Submission Data Extract
- The Data Model

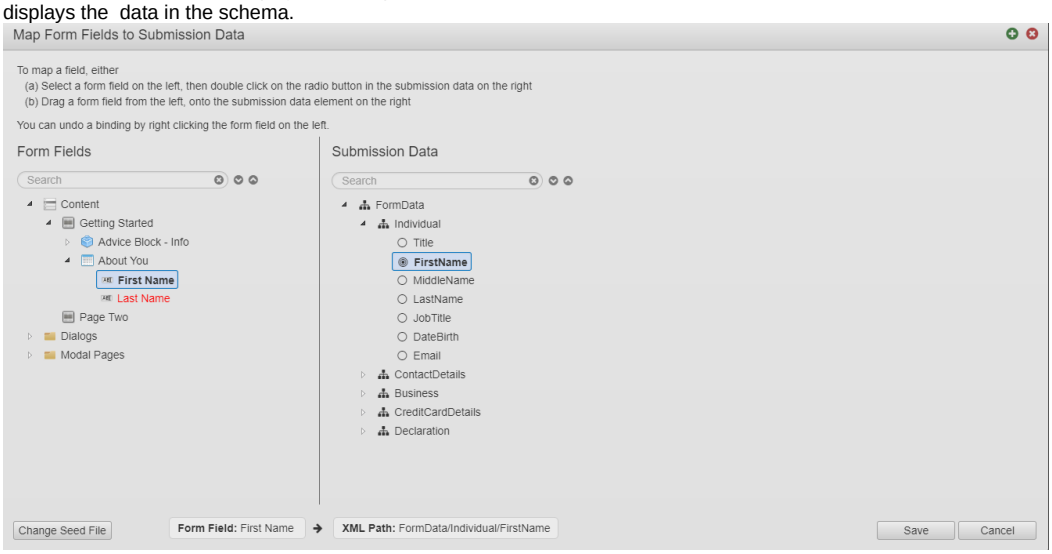
2. Click *Open Data Mapper* from the list



3. Click *Select Seed Document* button

4. Navigate to the XML file on your computer and click Load

5. Drag the components from the Form Fields column to where it maps to the Submission Data column.
   The Form Fields columns displays the components in the form. The Submission Data column displays the data in the schema.

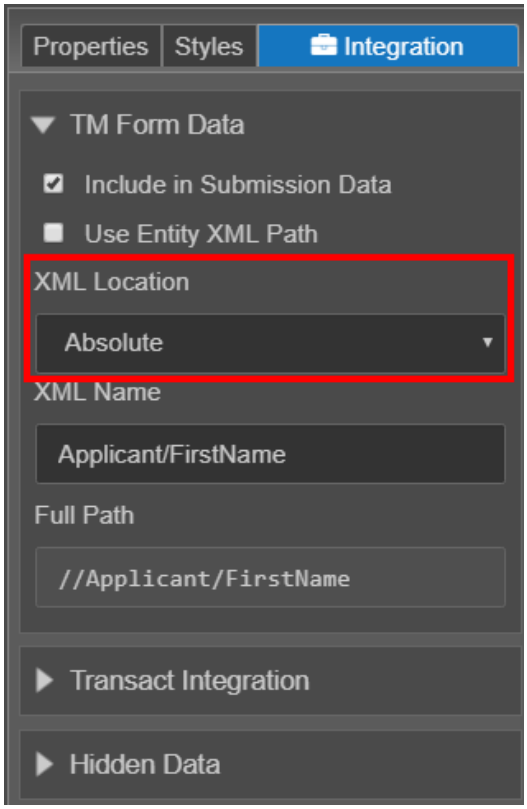


6. Click *Save* (when finished)

Relative vs Absolute

To guarantee the location of the XML upon submission of the form, you must ensure that in all cases, (other than inside repeating elements) where the component is not associated with an entity, that the XML location is set to "Absolute".

If it is set to "Relative", as the component is dragged between sections and blocks, it will assume a hierarchy relative to its location and thus break the data model enforcement.



In order to change the XML Location to "Absolute" you will need to deselect the, 'Use Auto-generated XML Name' checkbox.

Once the 'Use Auto-generated XML Name' checkbox is deselected, you will be able to select "Absolute" from the XML Location dropdown menu.

You will then need to update the XML Name.

Considerations

It is important to ratify your data model upfront as much as possible before setting out - and do not assume that a model which represents a downstream integration is a good way to operate.

For example, if you are integrating with *Vendor A*, and *Vendor A* requires the following format of output for "pets":

<Classification>

<PartA>

<Pets>

<Pet>

<Name/><Type/><Classification/>

</PartA></Pets>

It may be tempting to include this structure as part of your form. This has the benefit of making the payload meet the requirements for integration with little work. However, the forms will suffer as small tweaks to the schema for output require republishing every form.

Consider a client data dictionary, as it has several benefits, including:

1. Ownership – is in the clients estate, they have to agree data definitions up front and it removes uncertainty
2. Integration – switching vendors in and out becomes less problematic as you have one working model and various translation layers
3. Certainty – over time, model ownership synthesizes to good working practice, extensibility, and surety of operation

Names and IDs

Unknown macro: 'redirect'

Every component in Maestro has a number of different names that identify it. The table below provides details and examples.

Property	Description	Default Value	Simple Example	Long Example
ID	Every component has an ID. The ID is an internal name, which is used in calculations. The ID can also include an Entity which will allow you to group the structure of related component. The ID is used to create the Label and the XML Name.	The default value of the label is always the same as the name of the field type - but this is almost always immediately updated by the form designer. It is recommended that you always change the ID as soon as the new component is added to the form. The ID cannot include spaces and other prohibited JavaScript characters. If these characters are used, the ID will be truncated.	firstName	PrimaryApplicant_firstName
Label	The label is a human-readable name used to represent the component.	The default value of the label is based on the ID entered. You can change the Label so that it is not related to the ID. The label usually appears above the component when displayed in the form. For a more complex element such as a header, the label may be just one of several pieces of text that are contained in the component.	First name	My Postal Address is the same as my Residential Address
Name	Each component also has a name. The name is usually the same as the caption. The field's name is displayed in the Structure tab, as well as at the top of the Properties tab.	The name is usually the same as the label. The only time you might be aware of the name is when a components's label is very long. In this case, Maestro will give the compnent a truncated name.	First name	My Postal Address is the...
XML Name	The XML Name is used by Maestro to construct the XML payload for the form. The XML Name will be used as the element name for this component in the XML structure.	By default the XML Name is calculated from the ID. It has spaces removed, and will capitalize the individual words within the name. This naming convention more closely follows what many people use in XML Schemas. If you use the Data Mapper to bind fields to a seed file, then the XML Name will be updated accordingly.	FirstName	MyPostalAddressIsTheSameAs

You are strongly recommended to override the default values of both the ID and XML Name of any components which have a long or non-meaningful labels. This will assist greatly when you are writing calculations or working with the data.

Binding

Unknown macro: 'redirect'

Page Contents:

Related Pages:

- Binding
- Submission Data Extract
- The Data Model

Overview

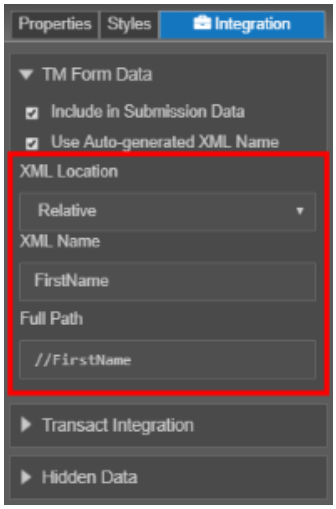
Binding is the process of "linking" a component in the form to the XML layer.

XML Location (relative vs absolute)

Relative Binding

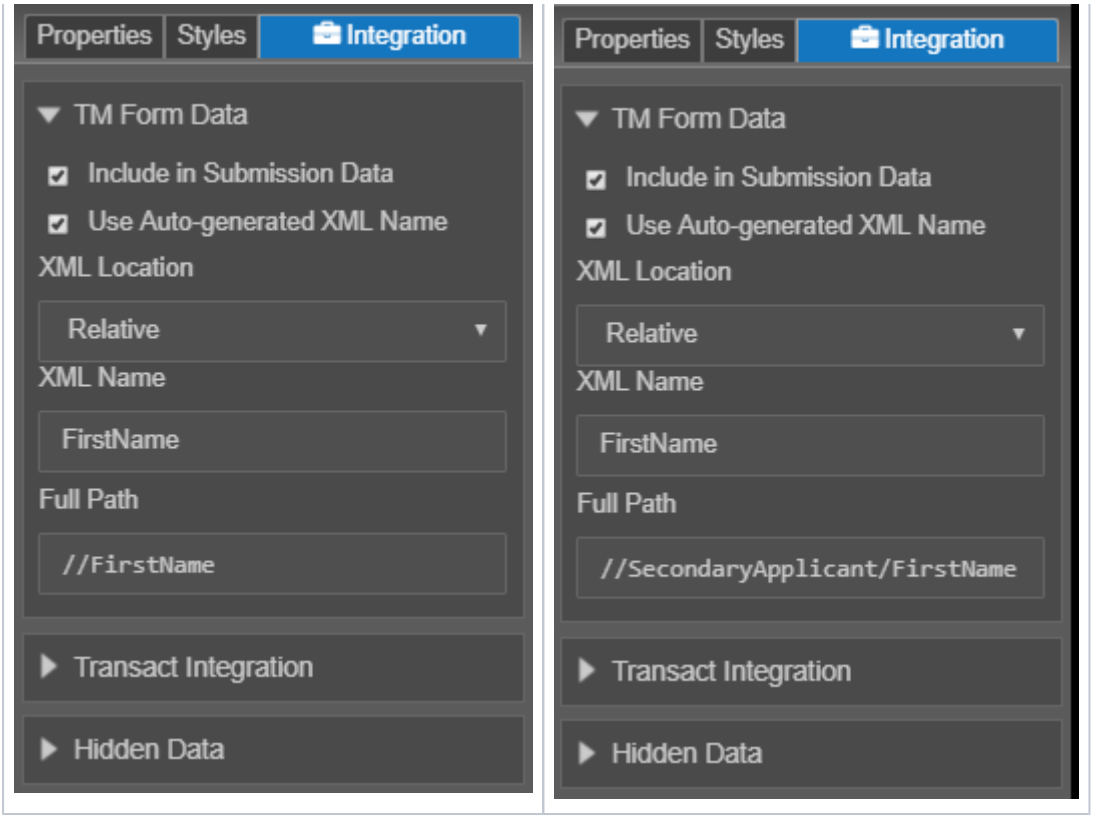
By default, most items (exceptions include pages, sections, blocks, and data fields) are bound under the root element in the XML layer, UNLESS their parent container has been included in the submission data. This is known as *Relative* binding. If an item's hierarchy changes, the XML Name will automatically be updated to reflect this.

On the Integration panel, you will find the *XML Location*, *XML Name*, and *Full Path*. These options will only display if the component is not part of an entity. Please see Understanding the Component ID for more information or view the Components Using Entity XML Path section on this page.



Notice, the *Full Path* in the screenshots below.

Parent not included in submission data	Parent is included in submission data



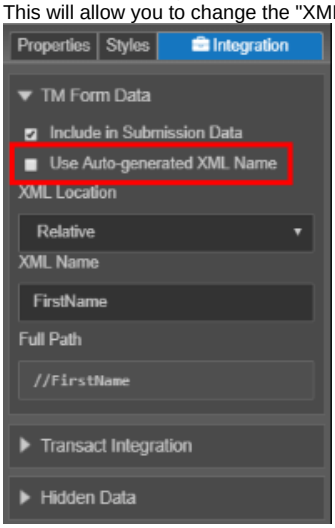
You can include components in the submission data, by selecting the component and then selecting "Include in Submission Data".

Absolute Binding

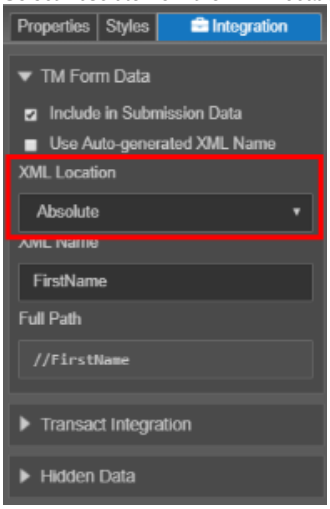
If you want to ensure that an item is *always* going to be bound to a particular XML tag, use "Absolute" binding.

To change to absolute binding:

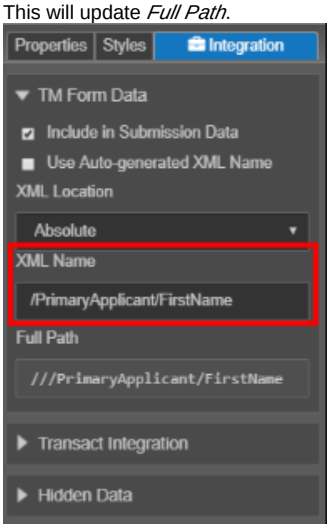
1. Select the component and switch to the Integration panel
2. Deselect *Use Auto-generated XML Name*
   This will allow you to change the "XML Location".



3. Select *Absolute* from the *XML Location* dropdown



4. Change the XML Name to include the component you want to bind the selected component to This will update *Full Path*.



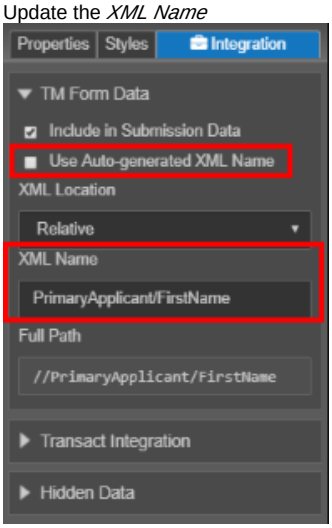
You should also use "Absolute" binding when you are binding to externally generated XML. It is safer to use "Absolute" binding to ensure the structure remains as you expect. For more information see the *Data Mapper* section on the The Data Model page.

Change XML Name

To change the XML name of a component, go to the Integration panel and under the Submission Data section, uncheck the 'Use Auto-generated XML Name'. You may then update the XML Name property.

1. Select the component
2. Switch to the Integration panel
3. Deselect *Use Auto-generated XML*

4. Update the *XML Name*



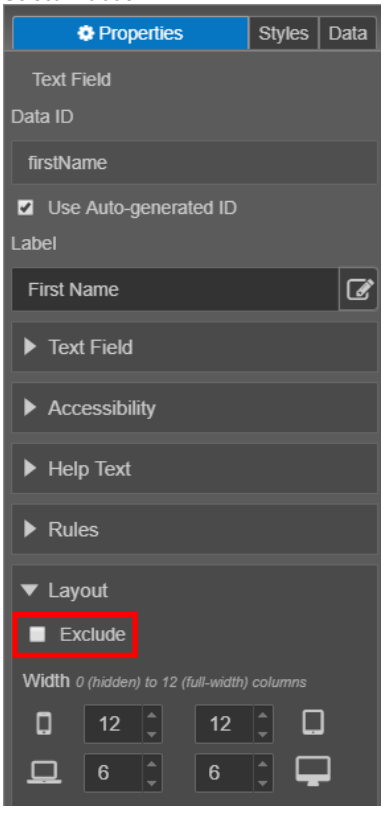
When changing the XML Name, you can type a single level name (FirstName), or you can create additional structure in the XML using a two-level name (personalDetails.FirstName or personalDetails /FirstName).

> You can update the XML Name for both "Relative" and "Absolute" binding.

Exclude a Component

You can exclude a component from the form's XML if needed.

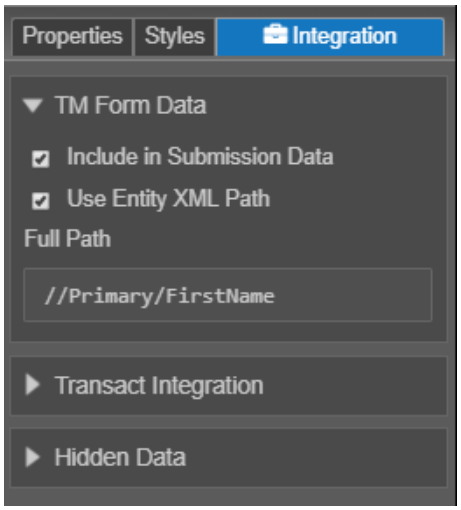
1. Select the component
2. Switch to the *Properties* panel (if needed
3. Expand *Layout* (if needed)
4. Select *Exclude*



Components Using Entity XML Path

When a component ID contains an entity the Integration panel options will be different.

The XML Location and XML Name properties will not be available. The location will be based on the entity included in the ID property as displayed in the Full Path property. You can deselect *Use Entity XML Path* to enter a different Full Path.


Properties
Styles
Integration
TM Form Data
Include in Submission Data
Use Entity XML Path
Full Path
//Primary/FirstName
Transact Integration
Hidden Data

Resolve Duplicate Bindings

Unknown macro: 'redirect'

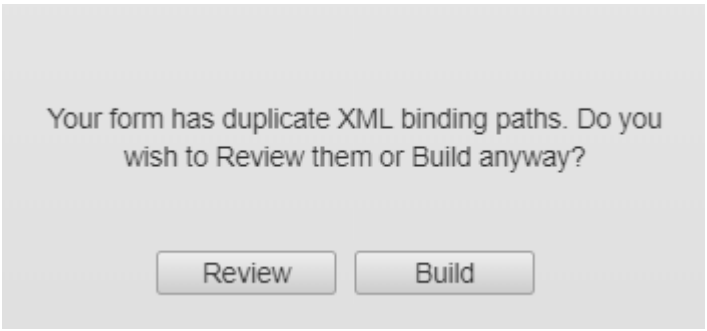
Related Pages:

- Binding
- Submission Data Extract
- The Data Model

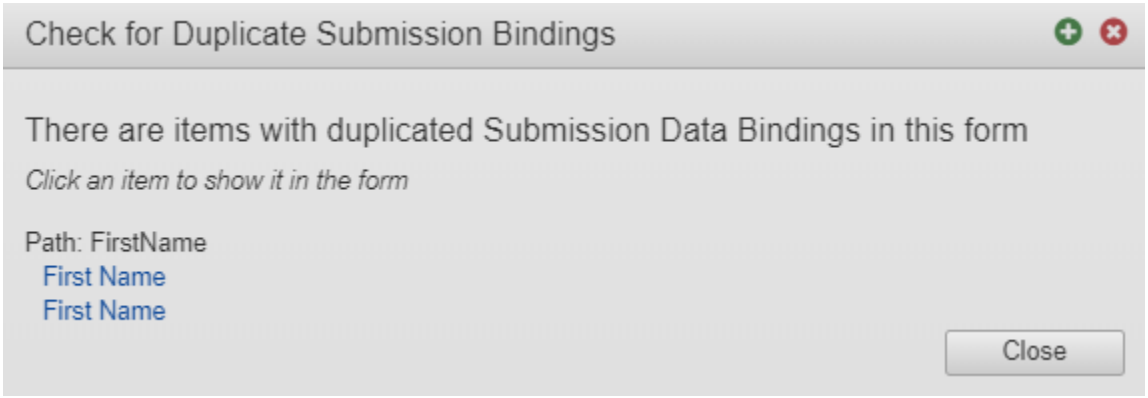
Resolve Duplicate Bindings

Duplicate bindings occur when more than one component shares the same XML Name. This can be caused by using the Accept button when a duplicate ID occurs, or when the XML Name is manually updated by editing the XML Name from the Integration panel. If you enter the same XML Name as another component you will *not* receive an error message until you use the Check for Duplicate Binding Errors option or build your TM form version.

When you add components to a form, Maestro will automatically add this component to the underlying XML layer. Parent containers are not automatically added so the default XML structure is flat. Therefore you may inadvertently, or purposely, bind two components to the same location. The dialog shown below displays when Maestro detects that two components have the same XML binding location when you build the TM form version.



Review will allow you to view the components that contain the same ID.



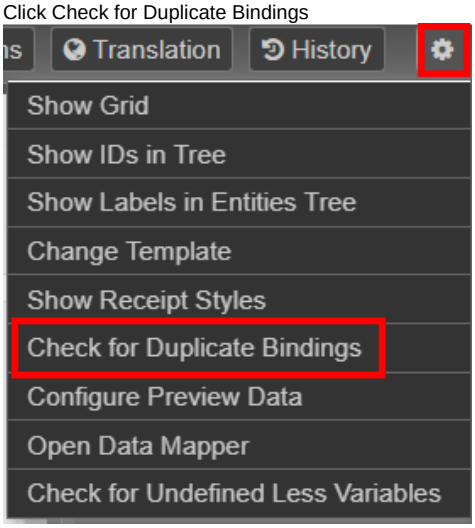
Build will allow you to continue building the TM form version. If you build anyway, please be aware that unexpected issues may occur.

Check for Duplicate Bindings

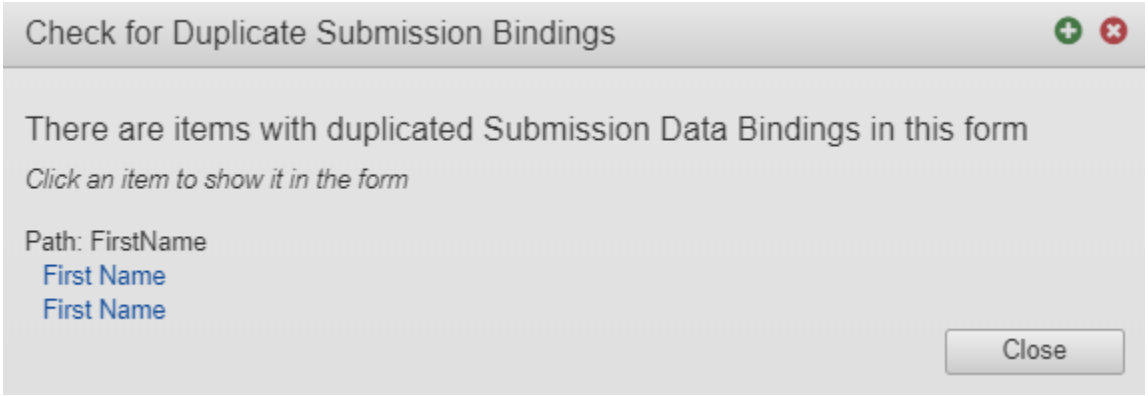
You can check for duplicate bindings any time while developing the form.

1. Select the Settings Menu icon

2. Click Check for Duplicate Bindings



This will display the duplicate bindings window, which will list the components that contain a duplicate binding.



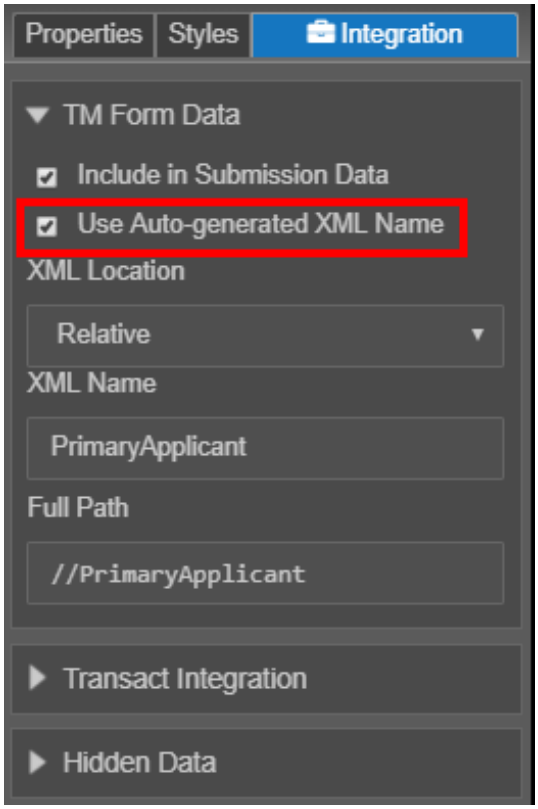
Resolve Duplicate Binding

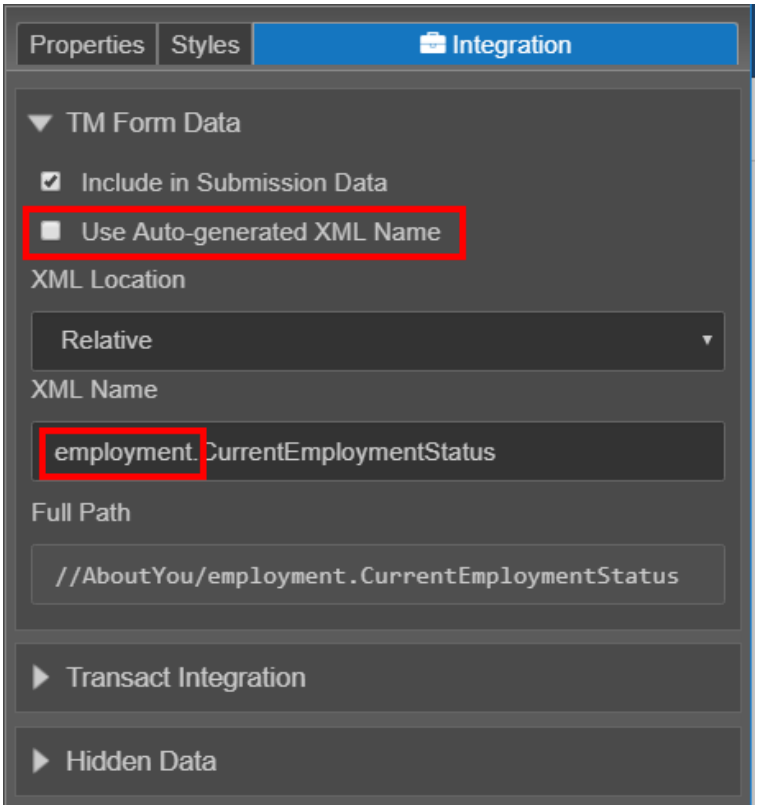
One way to resolve this issue, is to modify the component's parent's binding to ensure the components with the conflicting binding have different paths.

To achieve this, select the parent container of the component, select the Integration tab and select the "Include in Submission Data" option and ensure the XML Location option is set to Relative.

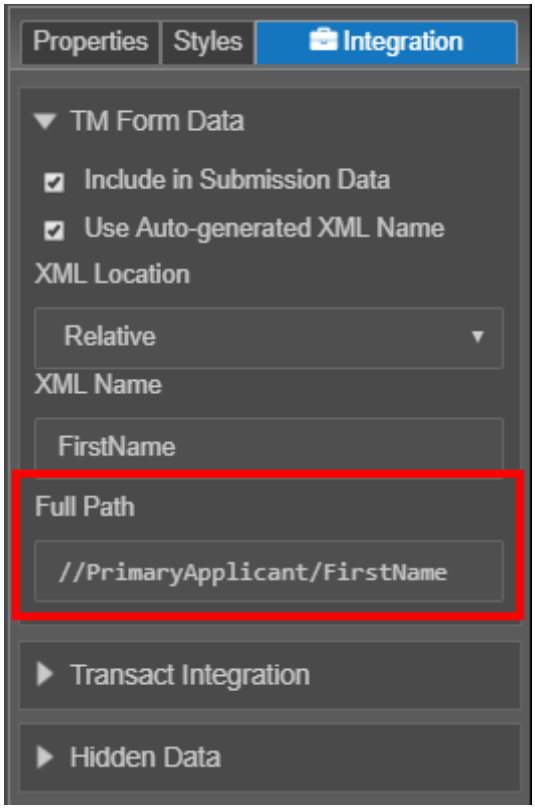
Updating the container component will update the Full Path of components within the container that are set to a Related XML Location.

Another option is to select the component and manually update it's XML Name property to include an additional parent level.





You may use either a dot (.) as above or a backslash (/) to create the additional parent in the component's hierarchy.


Properties
Styles
Integration
TM Form Data
Include in Submission Data
Use Auto-generated XML Name
XML Location
Relative
XML Name
FirstName
Full Path
//PrimaryApplicant/FirstName
Transact Integration
Hidden Data

Submission Data Extract

Unknown macro: 'redirect'

Page Contents:

Related Pages:

- Binding
- Submission Data Extract
- The Data Model

Overview

Submission Data Extracts allow the Form Builder to configure certain components to be extracted from the XML payload and placed in an easy to read format, such as an Excel spreadsheet. This extract may then be used for integration with an external system or simply emailed for further processing.

Submission data extracts usually represent some of the data, not all of it. To select which components are included in the data extract you need to configure them in Maestro, or through Transact Manager. This documentation will only discuss the configuration process using Maestro.

Submission Data Extracts can also be used for the following:

- Collaboration jobs. Data extracts are used to pass information from the form to the collaboration job in order to customize email messages, such as taking the user's first name and adding it to the email greeting.
- Overlay receipts. Forms used to implement pixel perfect receipts use data extract values for binding to the AcroForm.
- Card. A card is a repeating element in the portal which can be customized from data in the form using data extracts.
- Portal Searches. You can use data extracts when searching portals. For example, you can search for a task that has been assigned to you by using the data extracts and not the entire XML file.

Best Practices for Data Extracts

Form data extracts are a great way to quickly see key data elements from Transact that will enable you to make your business decisions or meet reporting requirements. There are some best practices when using form data extracts:

1. *Do not extract Personally Identifiable Information*
   Extracting this information using Transact, means that your customer's personal data may be downloaded in bulk from the TM server into a csv file. Using data extracts in this way is not the intended design, rather it is used for reporting purposes for generalizing on a particular set of data, or for testing, etc. It is best to keep any Personally Identifiable Information out of the data extracts, and rather send this data to your secure back end systems for processing.
2. *Do not extract every single piece of data in your form using Data Extracts*
   There are many other ways of integrating data into systems and sending your form data to your processing areas. Extracting all the data in your form using Submission Data Extracts adds a level of development and ongoing maintenance overhead, which is not necessary given the many other options to retrieve your data.

Configure a Component in Maestro

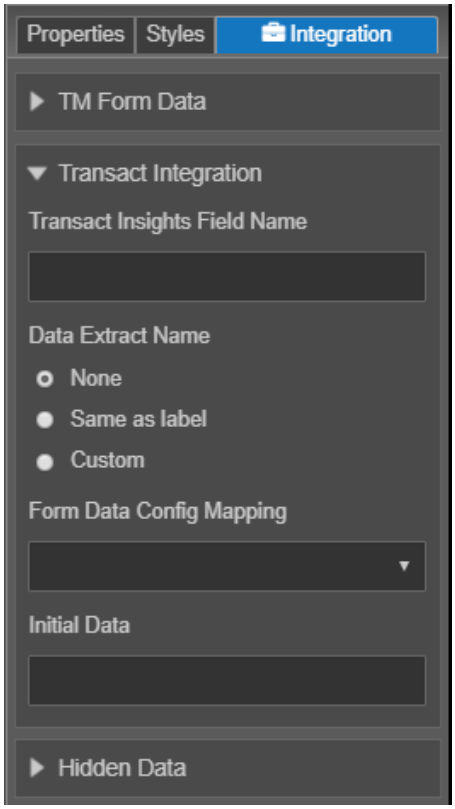
1. Select the component
2. Select the *Integration* tab
3. Expand *Transact Integration*

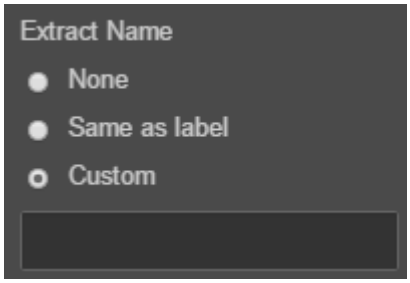
You will see the *Data Extract Name* options.

None is the default and this means that the component will not be included in the data extract.

Same as label will include the selected component in the data extract and it will use the label as the extract name in Transact Manager also known as TM.

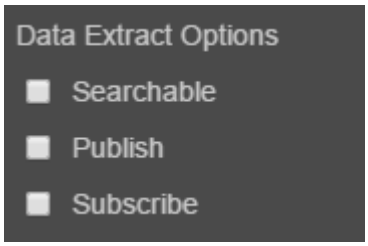
Custom will include the selected component in the data extract, but it will allow you to enter a specific name to be used in TM. Once you select *Custom* you will see an entry box displayed.





Data Extract Options

If **Same as label** or **Custom** is selected you will see the *Data Extract Options*.



Searchable means that the whole data extract value can be used for as search criteria in Transact Manager.

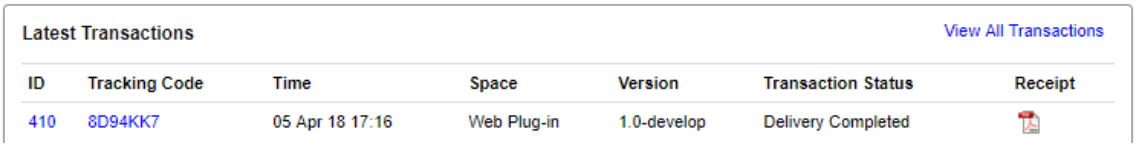
Publish is used by Collaboration Jobs in conjunction with the *Subscribe* option. Using *Publish* and *Subscribe* is an alternative mechanism to share data between tasks in a Form Bundle - Job Step.

Subscribe consumes the publish event using the published value.

Once all the required fields have been configured for data extract, build and render your form. Test out the form data extracts by completing and submitting a form.

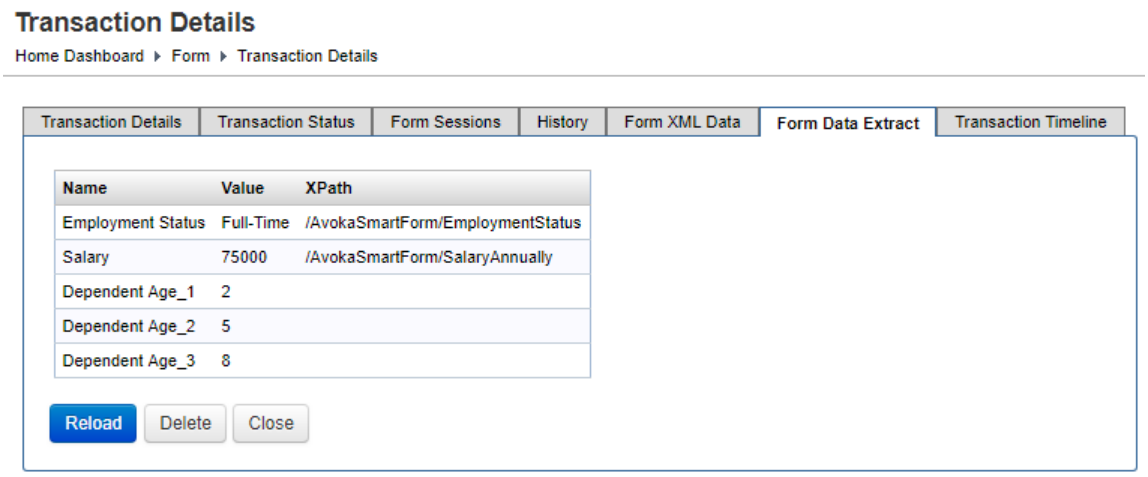
View Form Submission Data in Transact Manager (TM)

From the form dashboard, select your transaction from the *Latest Transactions* section by clicking on the recent transaction's *ID* or *Tracking Code*. This is the same as viewing a form submission from the appropriate *Operations* menu options.



Then switch to the *Form Data Extract* tab.

The screenshot below displays the entered *Employment Status* and *Salary* values (*Full-Time* and *75,000)* plus the ages of each of the applicant's three dependents (*2, 5* and *8*), as was extracted by TM for this submission in the *Value* column.



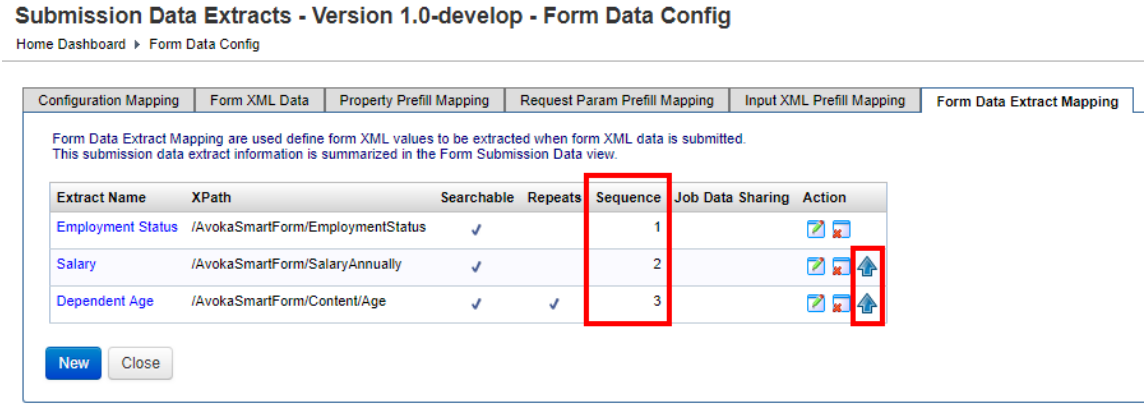
Rearrange Data Extract Columns

In Transact Manager you can configure and/or rearrange the data extract mappings.

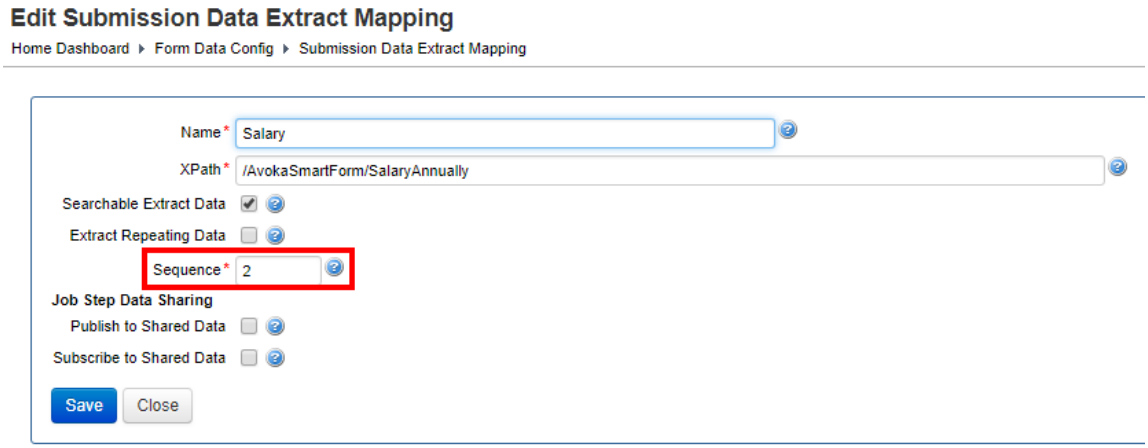
1. Select *Data Config* next to the form version
2. Select the *Form Data Extract Mappin*g tab

The *Sequence* column will indicate the order of the components within the extracted data table.

You can use the arrows to move components up in the sequence.



You can also edit the extract field and use the *Sequence* field to change the order.



> You can only change the *Sequence* to a number that is not currently being used. It will not automatically rearrange the component order on the Maestro form as displayed to the user.

Reference

Unknown macro: 'redirect'

This section contains reference information on various elements of Maestro.

- Components

Components

Unknown macro: 'redirect'

This section contains property options for specific components.

- Card Content Template
- Repeating Block Template
- Section
- Text Field

Card Content Template

Unknown macro: 'redirect'

Overview

The Card Content Template is used to simplify the process of collecting details and representing these details in a card display.

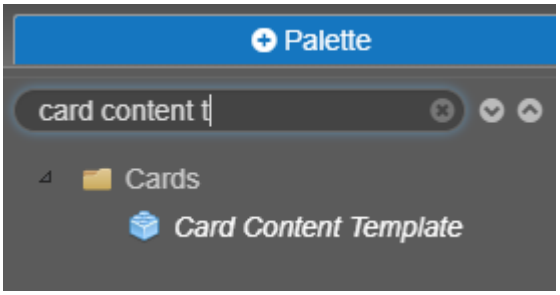
The Card Content Template is used to:

- Add one or many card details to forms
- Select and edit card details
- Add advanced card details by adding multiple components to the template

The Card Content Template is used to build the details that make up a card (a card is a layout type that collects and displays details about an applicant/form user). For cards-based layout forms, the Card Content Template is the recommended starting point as it allows form users to enter details for one or many cards. Once entered, the Card Content Template provides form users with the ability to navigate between different entered card details for selecting and editing purposes. The Card Content Template is quite similar to the Repeating Block Template in that it allows multiple instances of the same set of components to be added to a Maestro form.

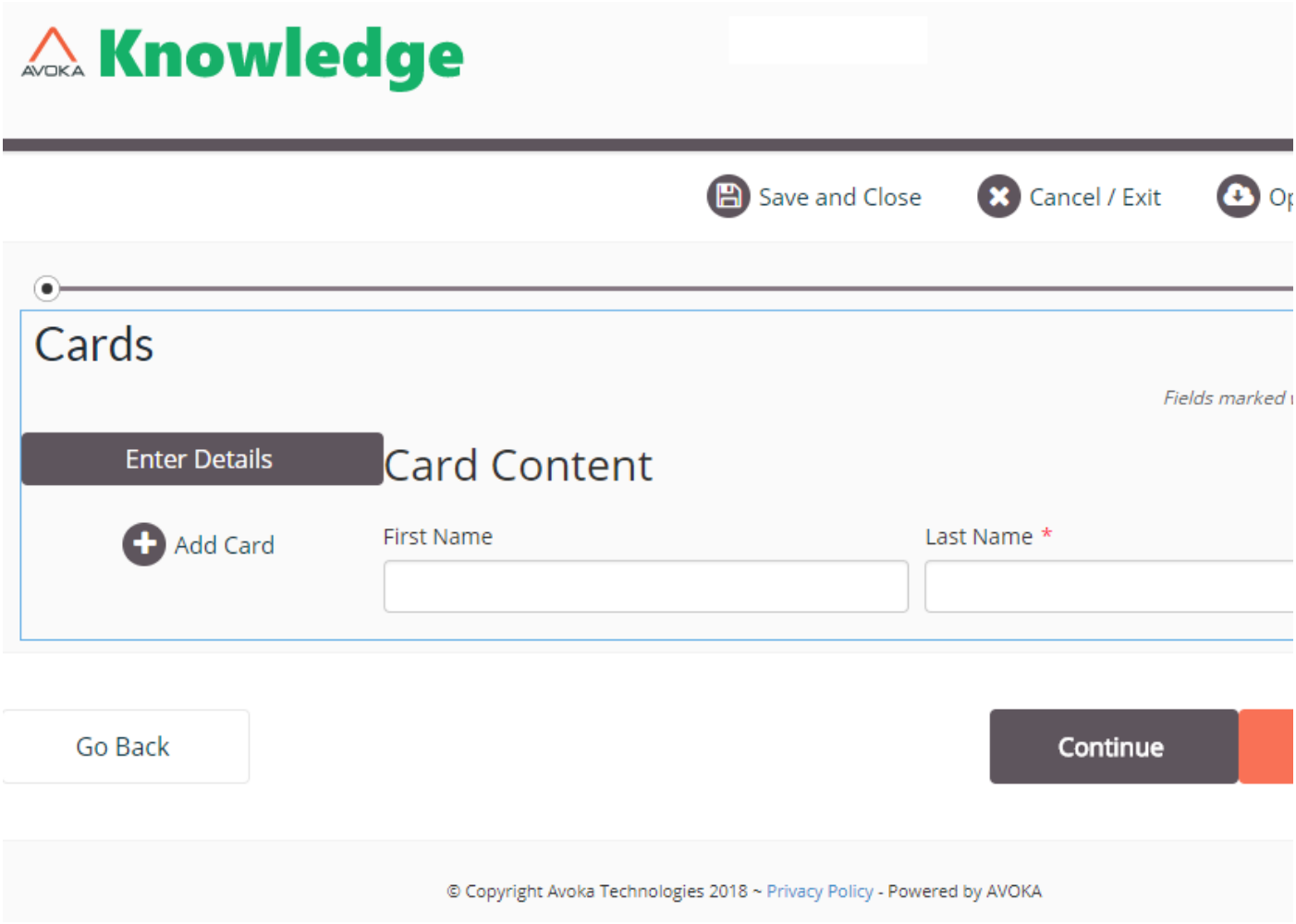
The Card Content Template component is made available from the Cards library and like all components is added to a Maestro form from the Maestro Palette.

The screenshot below displays the Card Content Template component in the Maestro Palette.

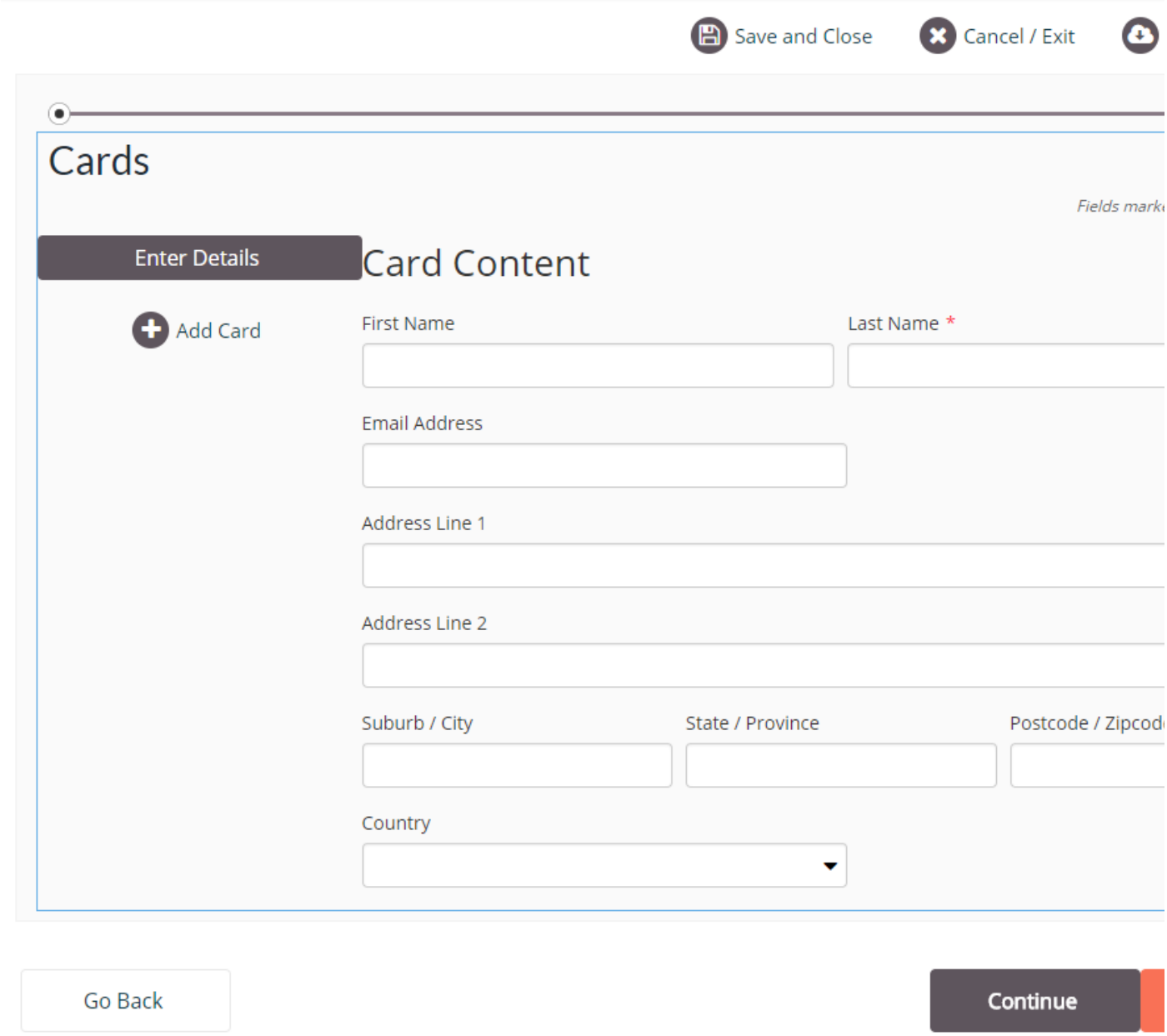


The Card Content Template combines the functionality of both the Card Selector and Card Content components.

You can add components as required. The screenshot below displays the Card Content Template with two text fields.



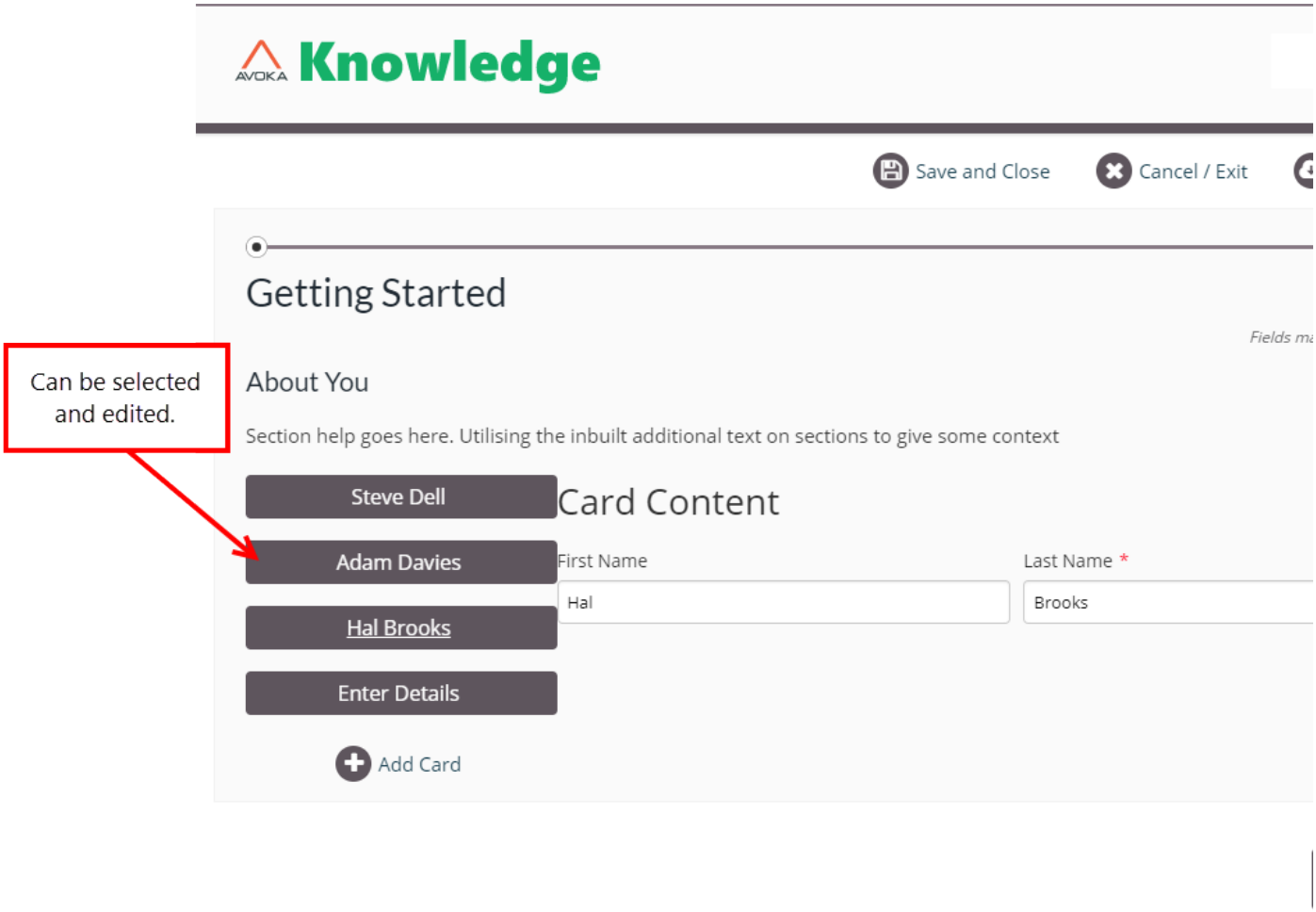
The screenshot below displays the Card Content Template after multiple components have been added to the template.



How to use the Card Content Template

Once details have been added by a form user, the *Enter Details* tab will display the first and last name of each card entered (by default but this can be changed by updating the labels). After an initial instance of details has been entered, the form user can click the Add Card button to enter details for another card, this process can be repeated as many times as the form user requires (a maximum number of repeats/instances can be configured using the Properties panel in Maestro).

The screenshot below displays a form that has had multiple card details entered.



In the above screenshot, details for 3 users (three cards) have been collected. Each of these options can be selected and edited as needed. If a form user selects a component in the Card Content Template, and then saves and resumes, the component that was selected before the form was saved will still be selected when the user returns to the form.

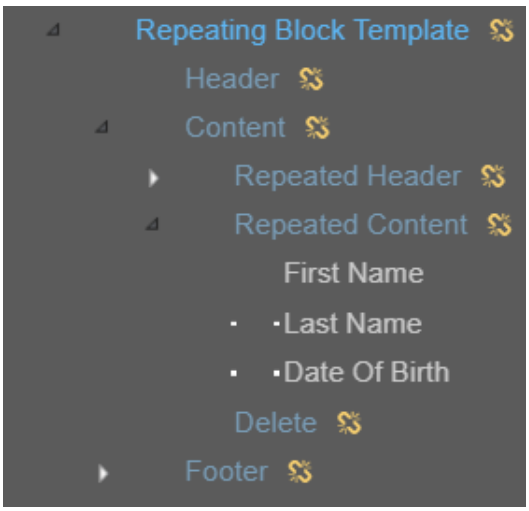
Repeating Block Template

Unknown macro: 'redirect'

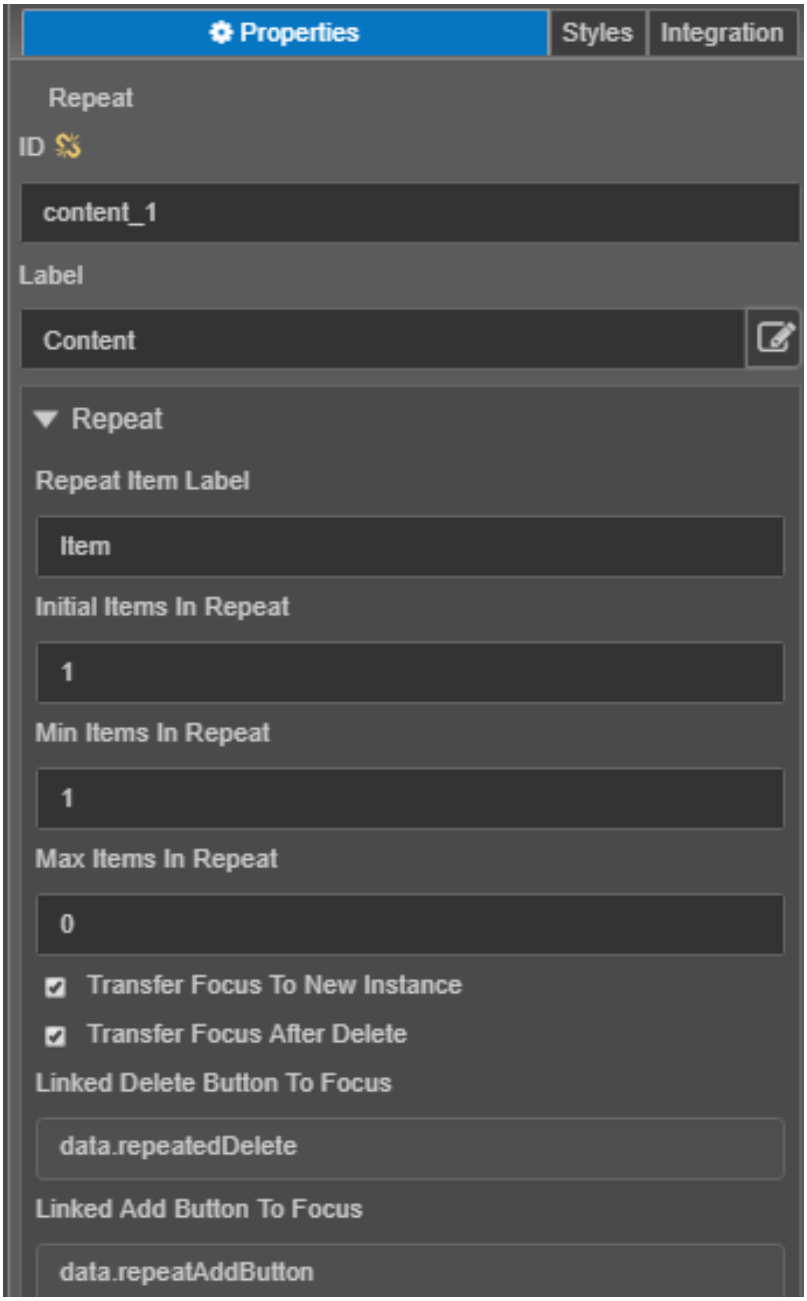
The Repeating Block Template allows users to add multiple instances of an item. This is a useful component when the number of instances will vary for each user. For example, the Repeat Block Template can be used to add the full names of multiple family members.

Before Components are Added

Components that you want repeated need to be added to the Repeated Content block within the Repeating Block Template. To change the properties of the Repeating Block Template, select Content. Selecting Content will allow you to navigate to the properties tab and change any of the properties associated with the selected Repeating Block Template.



Repeating Block Template Properties



Repeat Item Label - This will change the heading within the repeating section, as well as the button labels.

Initial Items in Repeat -This is the starting number of repeating items. The default option is 1, but you can change this number to whatever you like. However, you must have at least one item in the repeat.

Min Items in Repeat - The user has the ability to add or remove items. This is the minimum number of items that must be displayed on the form. You must have at least one item. Once the minimum is reached, the Delete button can disappear.

Max Items in Repeat - You set a maximum number of repeating items for the form user. Once the maximum is reached the Add button can disappear. Enter zero to indicate that there is no maximum number of items.

Transfer Focus To New Instance - When the user adds a new instance/item, the focus will be on the first component of the new instances/items.

Transfer Focus After Delete - When the user deletes an instance/item, the focus will be on the first component of the previous instance/item.

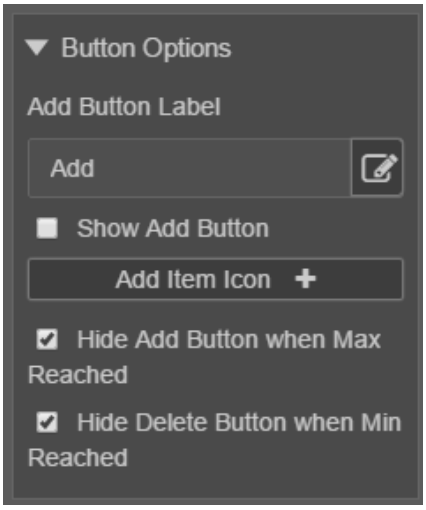
Linked Delete Button To Focus - When the user deletes an instance/item, the focus will be on the delete button of the previous instance/item in the repeat.

Linked Add Button To Focus - When the user deletes all instances/items in the repeat, the focus will be on the remaining add button. This setting is only relevant where the **Min Items in Repeat** is set to 0.

Button Options

Add Button Label - This allows you to change the wording on the add button. The default is "Add" plus the Repeat Item Label. You can use the Edit icon to display the text editor window which will allow you to use properties from other elements of the repeating block.

Hide Add Button when Max Reached - When this option is selected, it will hide the add button when the Max Items In Repeat is reached. This option is selected by default.



Hide Delete Button when Min Reached - When this option is selected, this will hide the delete button when the Min Items In Repeat is reached. This option is selected by default.

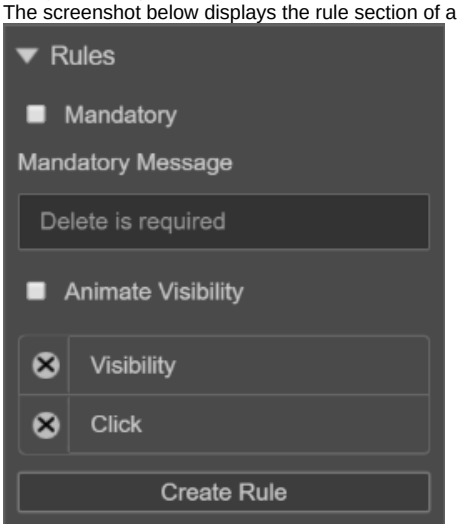
Hide Delete Button for a Single Item in the Repeating Block

Each time a new item in the repeat is created, an *Add Item* button and a *Delete Item* button will be created for the item.These buttons are used to allow the form user to continue to add items until they have entered all items that are required of them. However, there may be times where a Form Builder will not want to give the form user the option to delete an item until a certain validation rule has passed. The Form Builder may also only want the delete button to appear on certain repeat items.

An example of how to hide a delete button for a single item in a Repeating Block is described below.

To hide a delete button for a single item in a Repeating Block:

1. Select the Delete Item button from the Repeating Block
   Delete Item buttons in the Repeating Block Component have their own visibility rules that can be modified to create expanded validation and at times added functionality.

2. Navigate to *Properties* and scroll down to Rules
   The screenshot below displays the rule section of a default Delete Item button.
   

3. Double click the existing Visibility Rule
   This rule is created by default for all Delete Item buttons in a Repeating Block.
   To hide the delete button, we need to add additional logic to this visibility rule. Double-clicking the visibility rule will open the rule editor dialog. The screenshot below displays the default logic associated with a Delete Icon visibility rule.
   

4. Edit the visibility rule.
   Depending on your needs, you can edit the visibility rule in many different ways. In this example, we will use a checkbox to decide which instances of the repeat block will display a delete icon button.
   The screenshot below displays an example of how you may wish to edit the visibility rule. In this example, when the Hide Delete checkbox is selected, the Delete Item button will not display for the following item

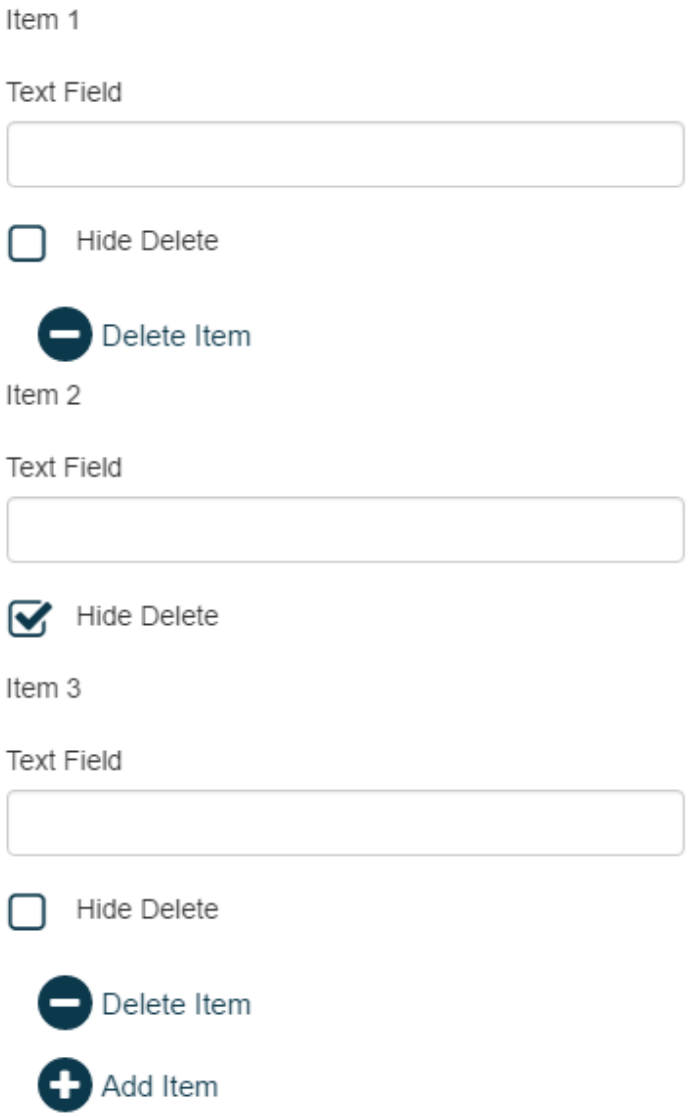
Show if...

true to show this item or false to hide it

```
var repeatParent = Util.findParentRepeat(item);
if(repeatParent && !Util.isBlank(data.$r) && repeatParent.properties.hideDeleteButtonOnMin)
  return (data.$r.length) > (repeatParent.properties.minItems || 0) && !data.hideDelete;
}

return !data.hideDelete;
```

5. Click *Save*
   The screenshot below displays an example of when this visibility rule is applied to a Repeating Block. In this example, the Hide Delete checkbox has been selected after the first Text Field has displayed and the result of this selection is that the Delete Item button has been removed from the second item.

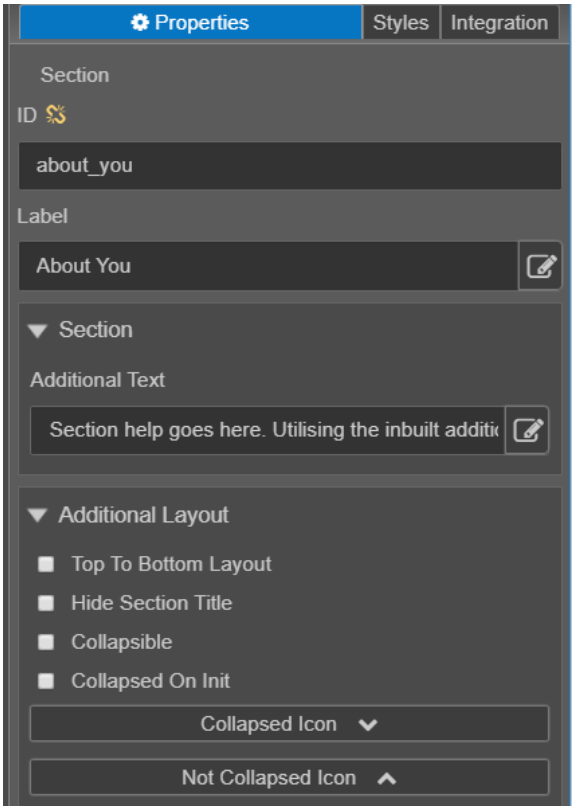


Section

Unknown macro: 'redirect'

Section

A section is used to break up a large number of fields in a long page into logical sections. A Section has a section header, and optional explanatory text. Sections also have additional features such as changing their default layout characteristics, and allowing them to be collapsed and expanded.



Section Header - this will display at the top of the section on the form.

Additional Text - this will display below the header within the section.

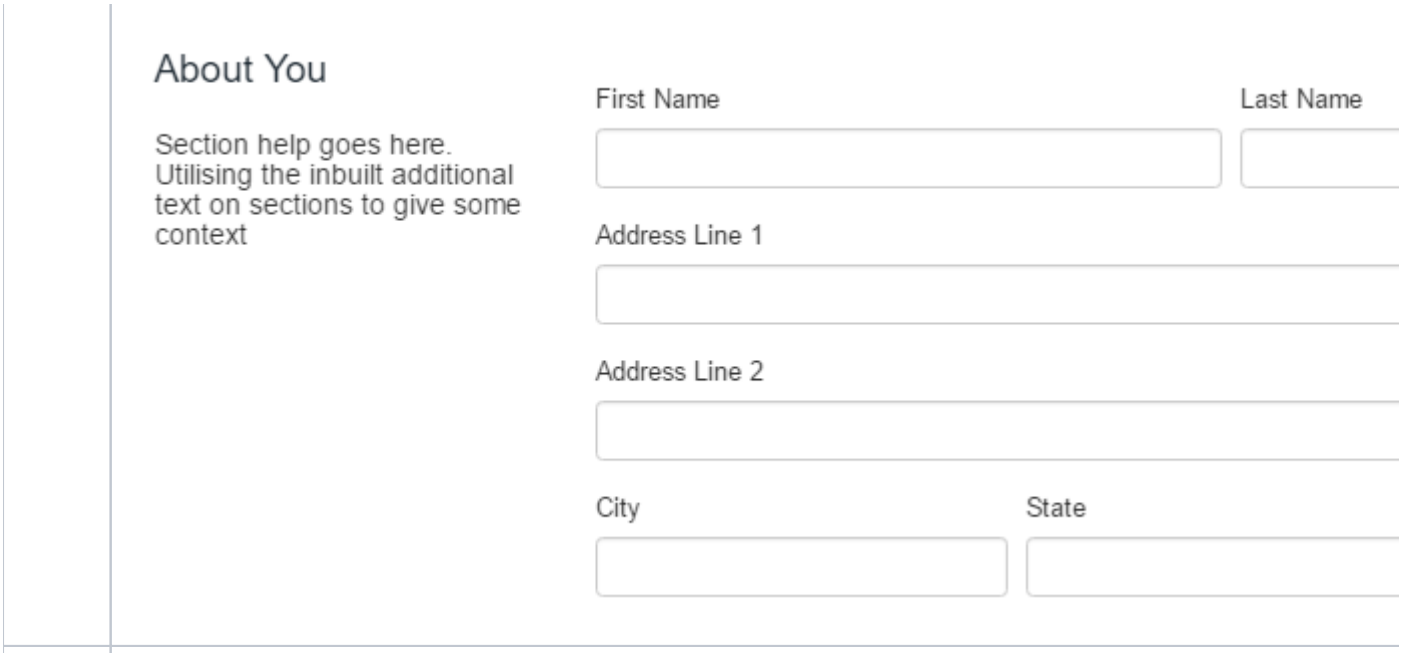
Top To Bottom Layout - this will change the layout of the section so that the header and additional text display above the components, rather than to the left.

Hide Section Title - this option hides the section title so that it doesn't display on the form.

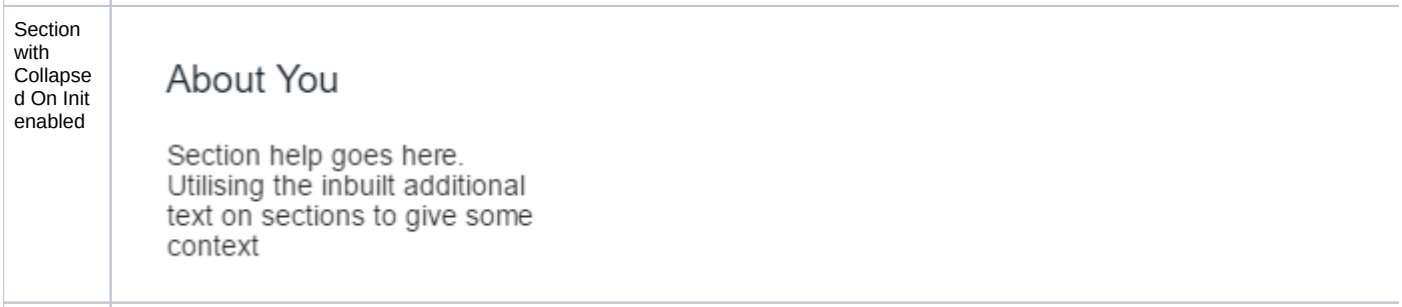
Collapsible - this option enables the user to collapse and expand the section on the form.

Collapsed On Init - this option has the section collapsed when first displayed. The user can expand and collapse the section as needed.

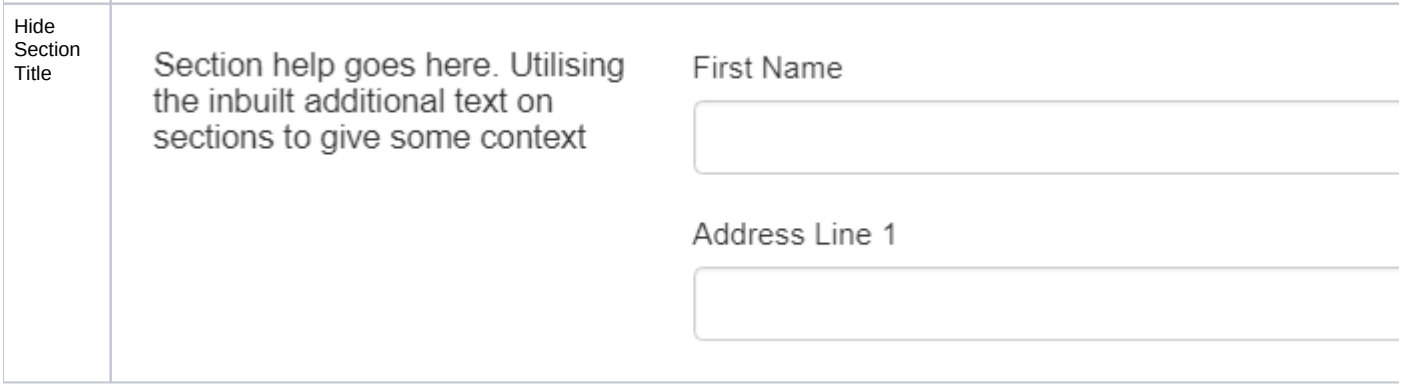
Details	Samples
Section Details	
Section with Collapsible enabled	



Section with Collapsed On Init enabled



Hide Section Title

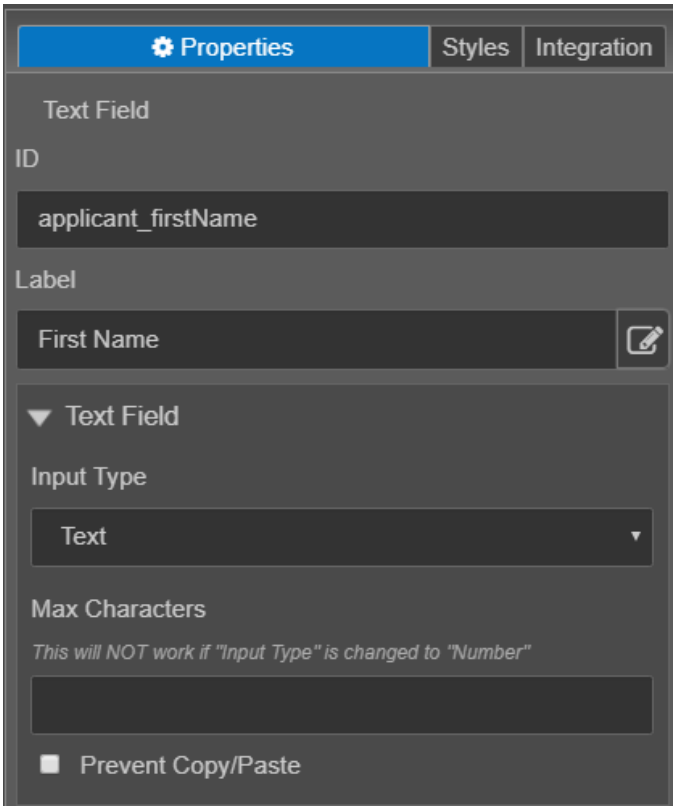


Text Field

Unknown macro: 'redirect'

The Text Field allows the user to create a text field within a Maestro form.

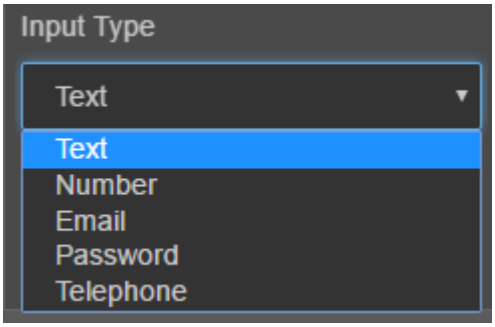
Properties



ID: The ID is the underlying Maestro name of the component currently selected. For more information see Understanding the Component ID.

Label - The label will be displayed to the user on the form. In the above example, the label is *First Name*.

Input Type - This dropdown is specific to text related components. By default, when you drag a text field component on to a form, the input type will default to Text. If you were to drag an email address component on to the form, the Email input type would be selected. Though you can change the input type, it is recommended that you change the component rather than the input type. For example, if you inserted a text field, but decide that you want an email address, you should delete the text field and add the email address component. The specific components have other pre-defined properties that are useful for form builders. For example, the email address component has a validation rule already created, whereas a text field (even if the input type is set to email), does not.



Each input type is described below.

Text - The Text input type should be used for text-based input.

Number - The Number input field is used for numeric input (0-9). For example, if you want a user to enter their age, use a text field and set it as a number input type.

Email - The Email input type is used for text fields that should contain an e-mail address. If you select the email type for a component that is not created for email input (e.g. a text field), the entered input will not be validated to ensure the user entered a valid email address. You can add a validation rule to the text field component, however, using the email address component (which already has the rule created) is much easier.

Password - The Password input type will mask any characters entered into the component. The image below displays an example of the characters being masked

Telephone - The Telephone input type is used for fields that should contain a telephone number. Similar to email addresses, you should use the specific phone number component, rather than changing the input type. The phone number component will contain the validation rule to ensure that the user is entering a valid phone number. You can add a validation rule to the text field component, however, using the phone number component (which already has the rule created) is much easier.

More information on input fields can be found at https://www.w3schools.com/html/html_form_input_types.asp

Though these options are available for selection, it is recommended that when possible you use a component tailored to suit the needs of the data that will be collected by the field. For example, if want to collect an email address from a user, use the Email Address component from the Maestro Palette.

Max Characters - This property allows you to set the maximum number of characters that the user can enter into the field. Letters, numbers, special characters and spaces are each counted towards the maximum number of characters.

Prevent Copy/Paste - If you wish to prevent this text field from being copied to or pasted from the clipboard, then select the ***Prevent Copy/Paste*** checkbox.

For large amounts of text, you should use the Multiline Text Area.