Avoka Transact

How-to videos

Unknown macro: 'redirect'

Step by step demo videos

- All about repeating data
- Custom Component Properties
- Getting Started with Data Driven Components
- How to access external data from a Maestro Form
- How to Use Preview data
- The Fluent SDK

All about repeating data

Unknown macro: 'redirect'

A typical requirement within a form is the ability to enter multiple instances, such as, with details of one or more persons. This is known as repeating data which may also be nested, such as, along with each person, a list of his/her assets is required to be entered.

In this video, David uses the above scenario to display in Maestro:

- The position in the list - as in the index for each person entered.
- The number of items in a repeat - as in the number of persons entered.
- The related parent details in a repeat - as in the name of each person owning the list of assets being entered.
- The summed value - as in the total value of assets for each person.

In this video, David shows in Maestro how:

- The form data object works and its structure.
- To access a repeats index, repeats parent data object and repeats array. That is, show the use of *data,$i*, *data.$p* and *data.$r* keys, respectively.
- To use the get repeat data method for summing a list of assets. That is, show how to use the API method for summing a person's listed asset values (*Form.getRepeatData*).

Custom Component Properties

Unknown macro: 'redirect'

In this suite of videos, David shows you how to create and use custom *Component Properties* within Maestro (video1).

He then shows you how to create some popular types of *Component Properties* (video 2).

What are custom Component Properties?

These are components that are customized for specific purposes and can be reused and shared between forms.

Why configure and use custom Component Properties?

Custom *Component Properties* can help drive your business logic and, once published, can be shared between forms.

Let's Start!

Video 1: Custom Component Properties

Here David uses a typical business requirement to start you with using custom *Component Properties*. On the first page, the user enters his/her first and last names of which the full name equivalent is displayed on the second page.

This is achieved by:

- Creating a custom component for a section having:
  - Custom *Component Properties* of type *Field Ref* for the first name and last name.
  - A *Calculation Rule* to concatenate the first and last name as entered on the first page
- Referencing the full name to dynamically display on the second page.

Video 2: Custom Component Properties – Part 2

David shows more on custom *Component Properties*:

- How to configure and edit the different types, such as *Integer*, *Boolean*, *Option*, *Option List* and *Field Ref Map*.
- How to reorder.
- How to delete.

Getting Started with Data Driven Components

Unknown macro: 'redirect'

In this suite of videos, David will show you how to configure and source data driven components within Maestro (video 1). The other option is to source the data driven components from Transact Manager, and two ways are demonstrated. One way is, to use a prefill Groovy service (video2) and the other is using a JSON definition (video 3).

What are Data Driven Components?

These are fields that are populated from a data source. For instance, when displaying the data list for dropdowns, checkboxes and radio button groups. The data list may at times be a static list of items, such as with the states of a specific country. But at other times, a more dynamic list of items is required.

Let's Start!

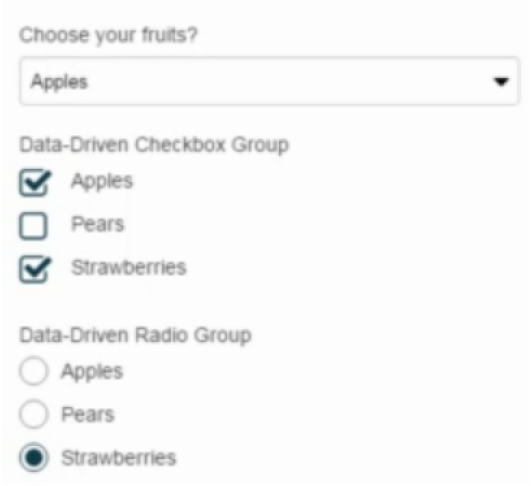
Video 1 - Maestro: Exploring Data Driven Components

Maestro: Exploring Data Driven Components from Avoka on Vimeo.

In this video, David creates a form with a simple dropdown to choose your fruit, as well as, creating this example as a checkbox group and as a radio button group.

Create a Maestro form:

- Configure a data driven dropdown, data driven checkbox group and radio button group.
- Configure a Data Field for holding the data values, the list of fruits, in two ways.
    a. An array of objects.
    b. A string delimited by pipe and commas - converted into an array of objects using an API method Calc.delimitedSelection (David's preferred, as requires less code).
- From the fruit dropdown, checkbox group and radio button group, reference the array of objects, the data field



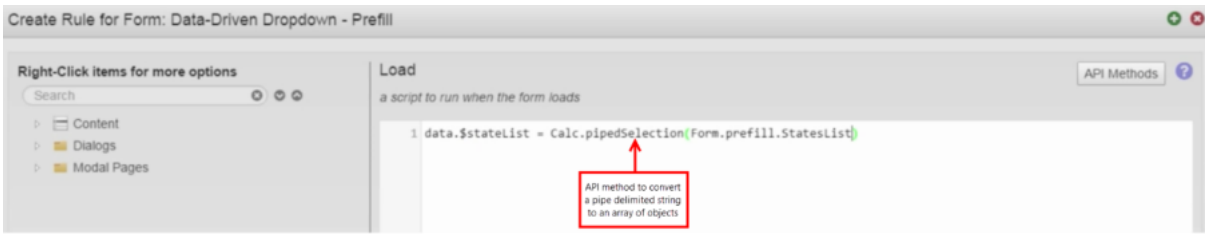
Video 2 - Maestro: Populating Data Driven Components Using a Form Prefill Data Service

Maestro: Populating Data Driven Components using a Form Prefill Data Service from Avoka on Vimeo.

David shows you how to populate a dropdown of states using a string in a prefill Groovy service.

General steps are:

1. Create the Maestro form with the data driven dropdown for choosing your state.
2. Create a Groovy service in TM which defines the data as a list of states delimited by pipes.
3. Create a form Load rule in Maestro which defines a data field, states list, and at start, loads the states using the new Groovy service.
4. From the state dropdown, reference the data field to obtain the states list.



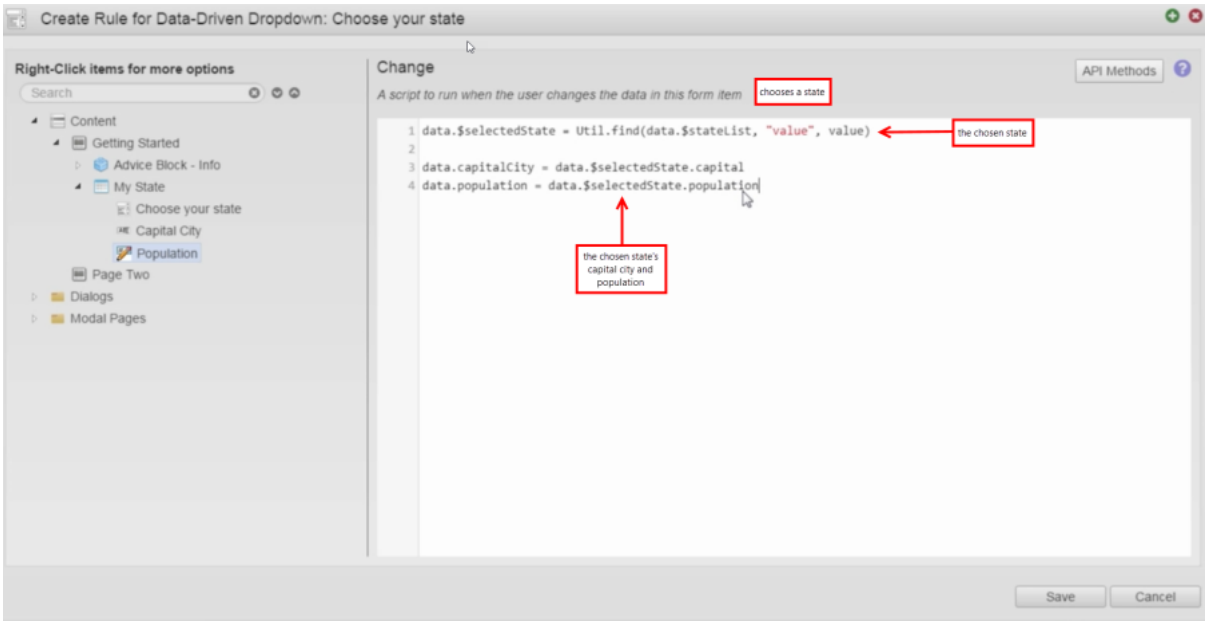
Video 3 - Maestro: Populating Data Driven Components Using JSON in a Form Prefill Data Service

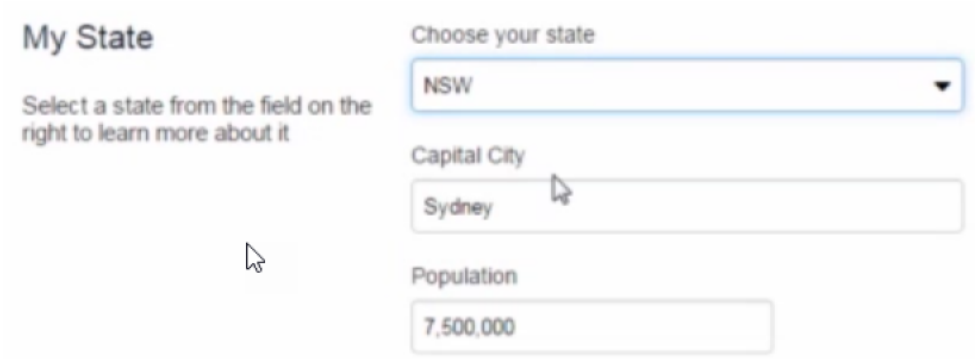
Maestro: Populating Data Driven Components using JSON with a Form Prefill Data Service from Avoka on Vimeo.

Here David shows you how to extend the state list dropdown. In addition to the states dropdown, any time a state is selected its related capital city and population is displayed. This is achieved by the JSON array holding the capital city and population along with each state.

General steps are:

1. Add two additional fields for capital city and population.
2. Create a Groovy service in TM which defines prefill using a JSON array of states plus their corresponding capital city and population.
3. Create a form Load rule in Maestro which defines a data field, states list, and at start, loads the states using the new Groovy service.
4. From the state dropdown, reference the data field to obtain the states list.
5. Create a Change rule for the State dropdown to obtain the chosen state's capital city and population whenever the user chooses another state.





How to access external data from a Maestro Form

Unknown macro: 'redirect'

Transact Manager: How to access external data from a Maestro Form

In these videos, you will learn how to access external content from a Transact Maestro form, using two different methods, Property Prefill (video 1) and Groovy Dynamic Data Services (video 2).

Transact Manager uses Property Prefill to inject property values into forms when they are rendered, and Groovy Dynamic Data Services can be called from a Maestro form to provide on-demand data at runtime.

Video 1 - Transact Manager: Property Prefill

Many forms present information that is best kept outside of the form design. This protects the form from the changing nature of information, and allows the same form to present different information for different circumstances.

One method Transact Manager uses to support externalizing content is *Form Property Prefill*. This method allows properties of a form to be defined in Transact Manager, which are then injected into the form when it is rendered.

In this video, Bill shows how to use Transact Manager to define a Property Type, create a Form Property of that Property Type, and how to map the Form Property to a Maestro Form field so it appears in the form at runtime.

Video 2 - Transact Manager: Services for Maestro Forms

A Transact Maestro Form can leverage Transact Manager Dynamic Data Services to access external data. Just about every form imaginable needs access to some form of external data to present a better user interface, or perform validation.

Lists such as vehicle manufacturers, models and colors are just one example of dynamic data that is best kept external to the form, and accessed on demand at runtime. You don't want to have to change your form each time a manufacturer offers a new color.

In this video, Bill introduces the concept of Service Connections, uses the Transact Manager dashboard to create Fluent Groovy Services, then shows how to access data from those services in a Maestro form.

He creates two Groovy services which in turn call public web services to provide a list of countries and currencies, and to lookup the latest exchange rate between two currencies. He then shows how to use Transact Maestro to call those Transact Manager Groovy services, by replacing hard-coded data in the demonstration form with calls to fetch live data.

During the presentation, you will also learn how to import and export services and service connections by using Application Packages.

To download the code used in the videos for groovy services and unit tests, and in the Maestro form, click below to download a single zip file with all the code inside.


Transact Manager... Groovy Code.zip

How to Use Preview data

Unknown macro: 'redirect'

How to use preview data from Avoka on Vimeo.

In this video, David shows you how to use the *Configure Preview Data* feature within Maestro.

Why use preview data?

You may wish to try out your form in Maestro before publishing it to Transact Manager. This is handy for previewing the form to see what it would look like with entered details from a fictitious user.

The *Configure Preview Data* feature in Maestro allows you to preview your form with a sample of data as may have been entered by a user and thus facilitating the:

- Testing of business logic, such as with the form's business rules.
- Previewing of receipts.

There are two main configuration options:

- *Generate Default XML* – to automatically generate the XML from the form and then manually add the data values. This is great for a quick preview.
- *Generate From File* – to use a file containing the XML structure and already filled with some sample of data. This is great for reuse in testing forms especially when requiring a more structured testing and where the test data is to represent more realistic scenarios.

The Fluent SDK

Unknown macro: 'redirect'

The Fluent SDK is the recommended approach to developing integration with the Avoka Transact Platform.

Please refer to the links below for more information about the Fluent SDK.

The full Fluent SDK documentation is available here:

SDK51

Configuring the Fluent SDK in your IDE

Desktop IDE Development with The Transact Fluent API

A Simple Example

Hello World Service Example