FLOWX.AI

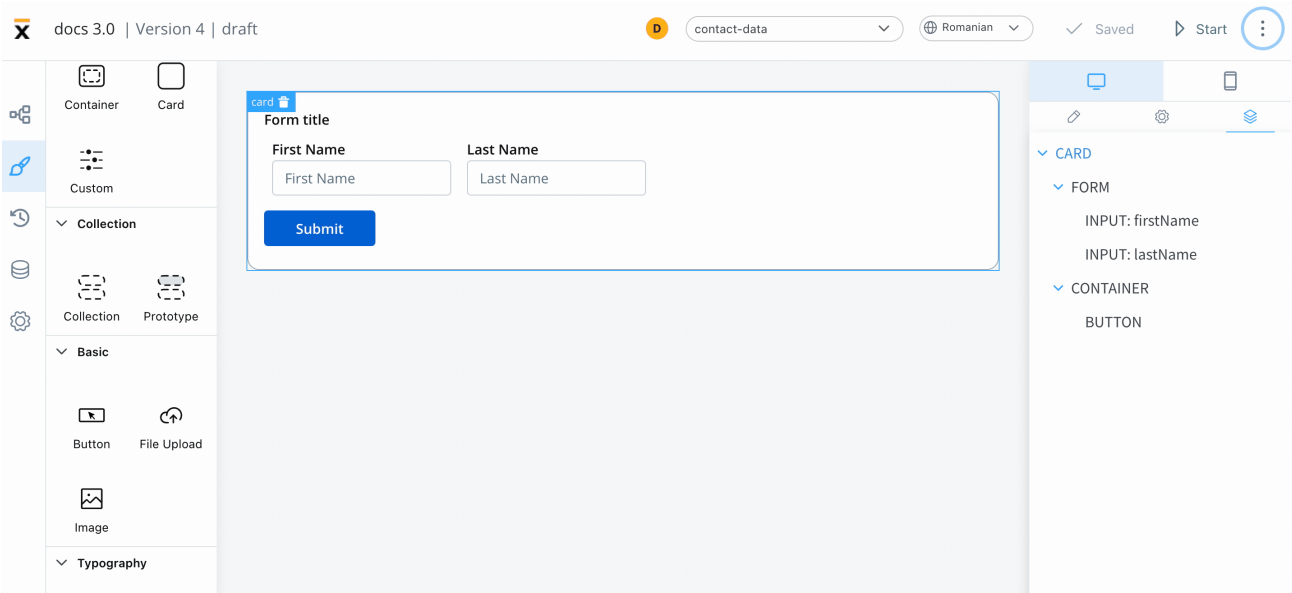
BUILDING BLOCKS / UI Designer / UI component types / Root components

Contents

- BUILDING BLOCKS / UI Designer / UI component types / Root components / Container
- BUILDING BLOCKS / UI Designer / UI component types / Root components / Card
- BUILDING BLOCKS / UI Designer / UI component types / Root components / Custom

BUILDING BLOCKS / UI Designer / UI component types / Root components / Container

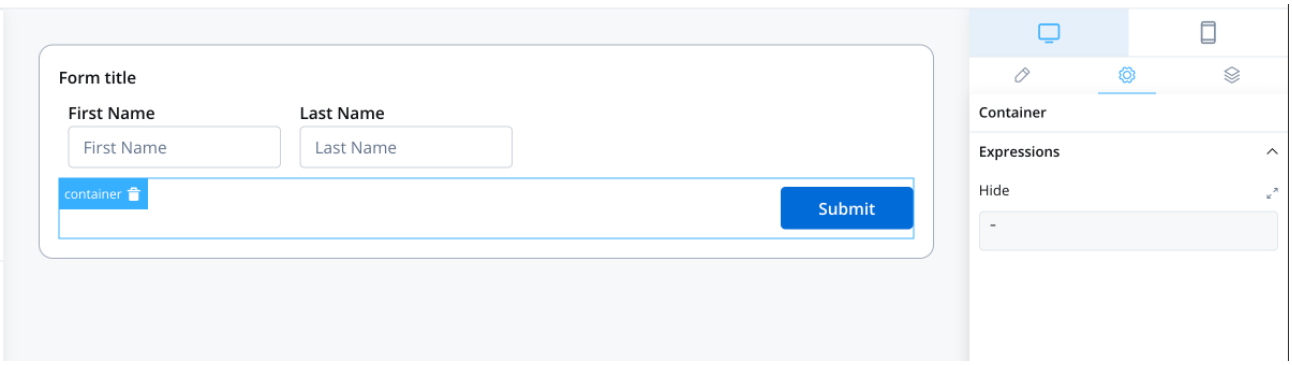
A container is a versatile element that allows you to group components and align them as desired.



The following properties can be configured in the container:

Settings

- **Expressions (Hide)** - JavaScript expressions used to hide components when they evaluate to true



Styling

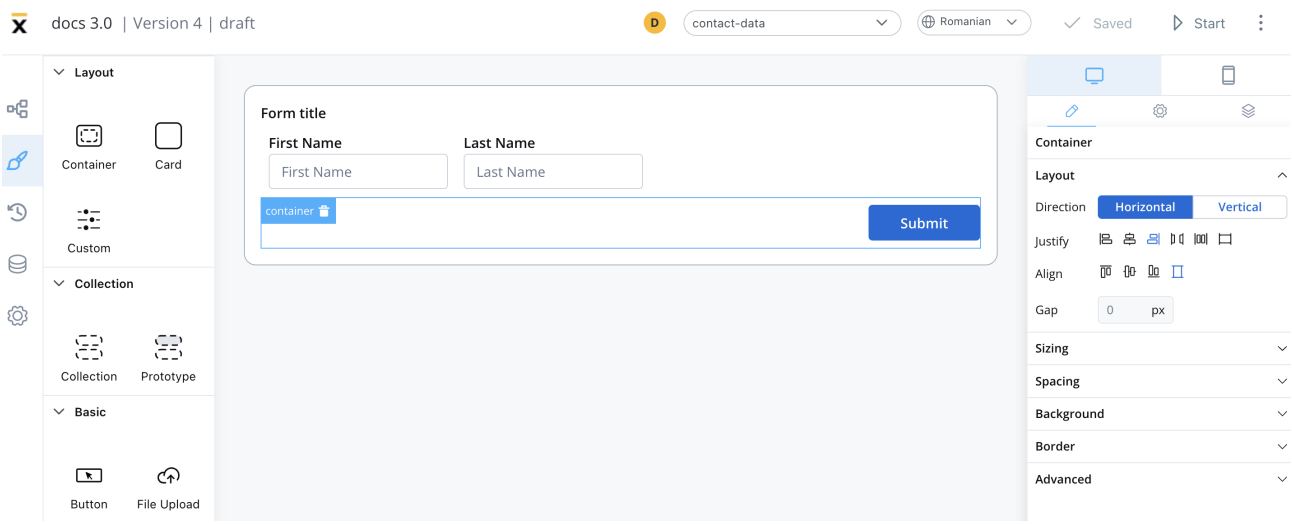
- **Layout** - this property is available for components that group children and includes the following options:
  - Direction - Horizontal / Vertical (for example, select *Horizontal*)
  - Justify (H) - (for example, select *end*)
  - Align (V) - this option allows you to align components vertically
  - Gap - you can set the gap between components

More layout demos available below:

» Layout Demos

When you apply the above properties, you can generate the following output, with the button appearing on the right side of the container, underneath the form with

three form elements:



For more information about styling and layout configuration, check the following section:

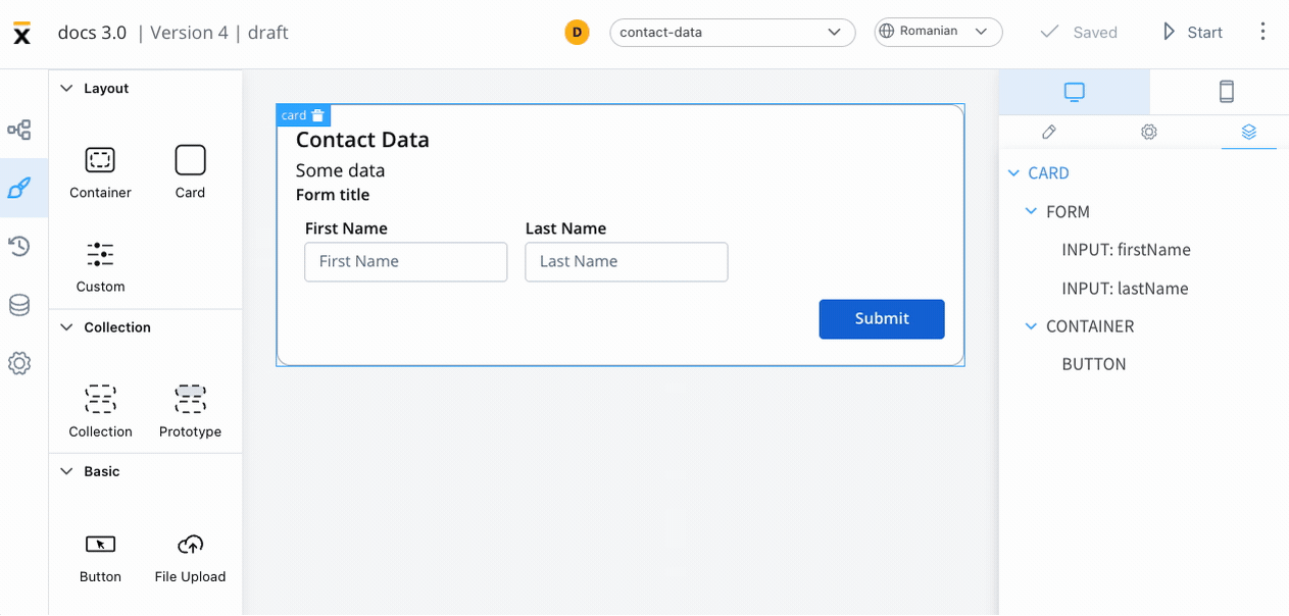
» UI Designer

Was this page helpful?

BUILDING BLOCKS / UI Designer / UI component types / Root components / Card

A card is a graphical component that allows grouping and alignment of other components. It can also include an accordion element for expanding and

collapsing content.



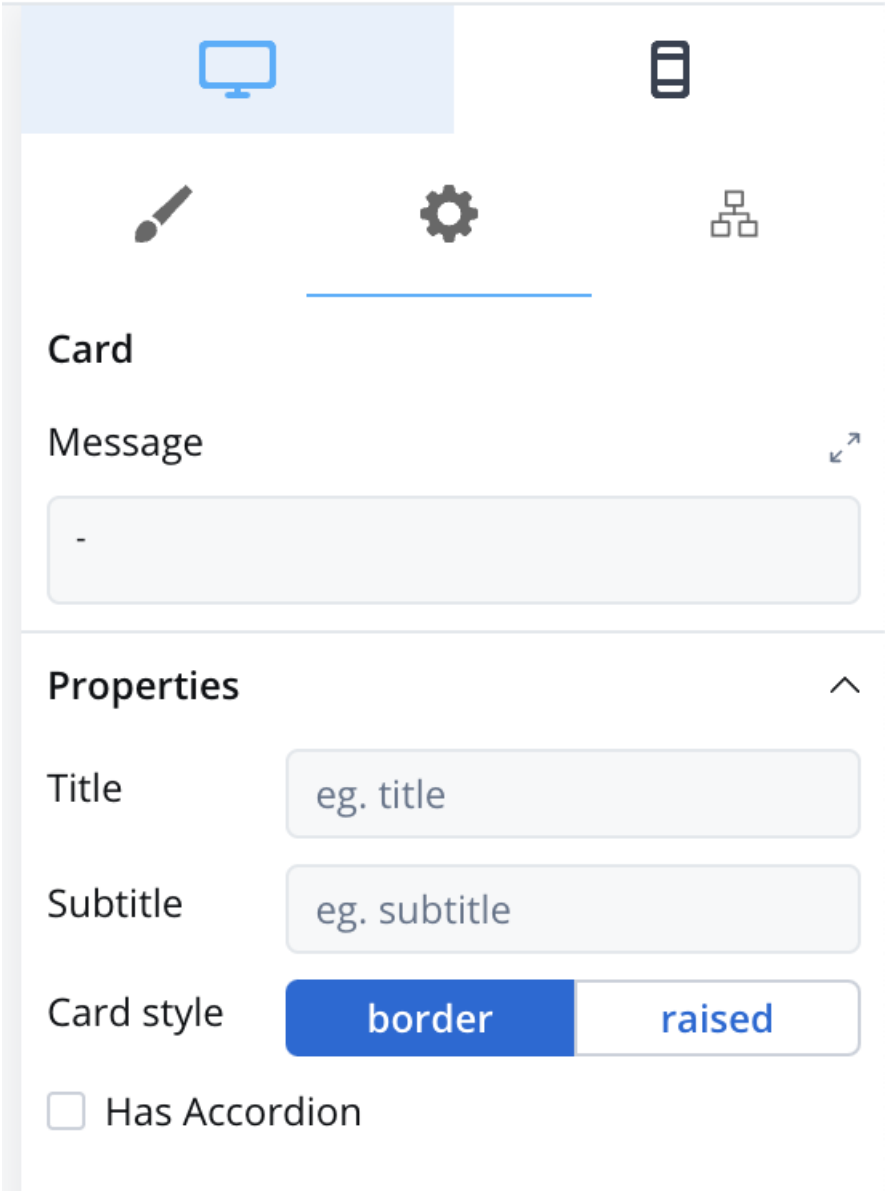
The following properties that can be configured:

Settings

- **Message** - a valid JSON that describes the data pushed to the frontend application when the process reaches a specific user task
- **Title** - the title of the card
- **Subtitle** - the subtitle of the card
- **Card style** - you can choose between a border or raised style
- **Has accordion?** - this feature allows you to add a Bootstrap accordion, which organizes content within collapsible items and displays only one collapsed item at a time

CAUTION

> Accordion element is not available for mobile.



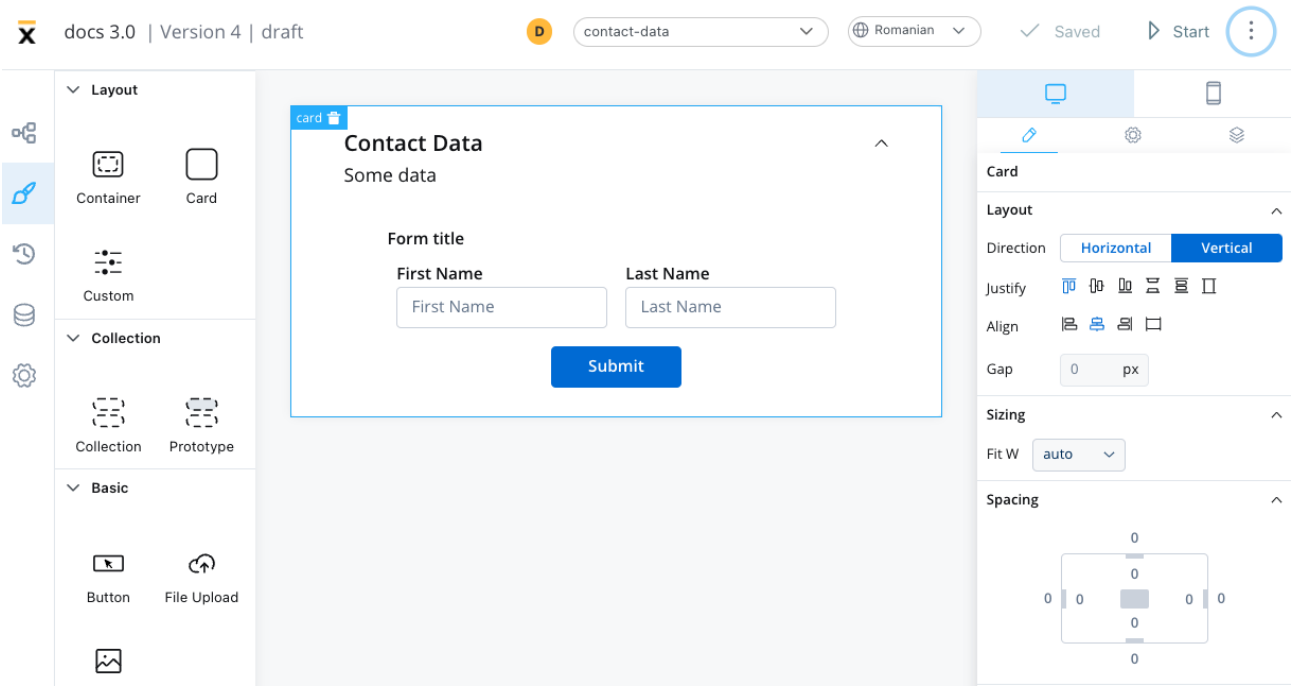
Styling

- **Layout** - This property is available for components that group children and includes the following options:
    - Direction - Horizontal / Vertical (for example, select *Vertical*)
    - Justify (H) - (for example, select *center*)
    - Align (V) - this option allows you to align components vertically
    - Gap - you can set the gap between components

More layout demos available below:

» Layout Demos

This example will generate a card with the following layout configuration:

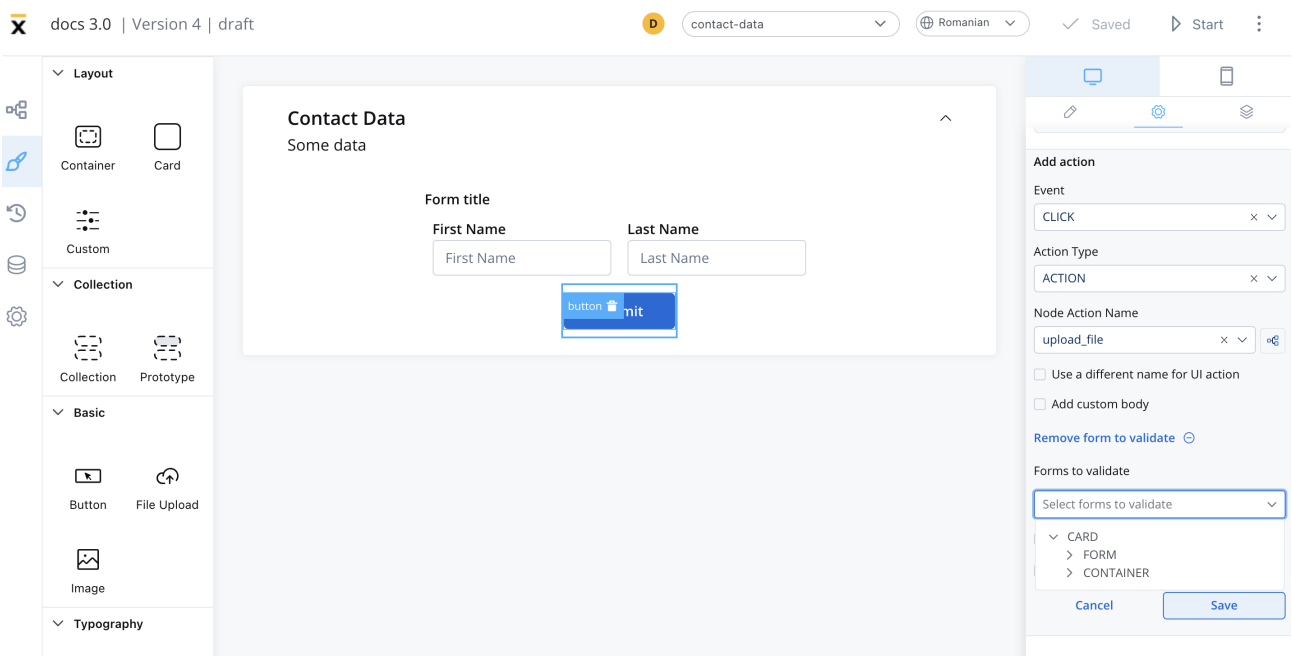


For more information about styling and layout configuration, check the following section:

» UI Designer

Validating elements

To validate all form elements under a card, you need to set the key of the form/element on the property of the button: *Forms To Validate*.

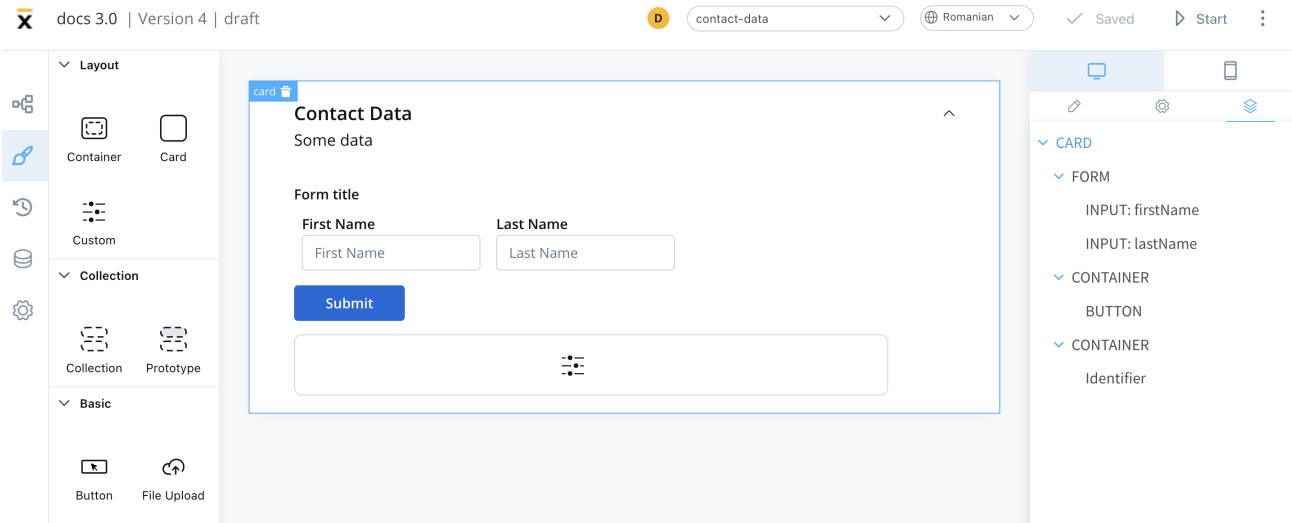


Was this page helpful?

BUILDING BLOCKS / UI Designer / UI component types / Root components / Custom

Custom components are developed in the web application and referenced here by component identifier. This will dictate where the component is displayed in the component hierarchy and what actions are available for the component.

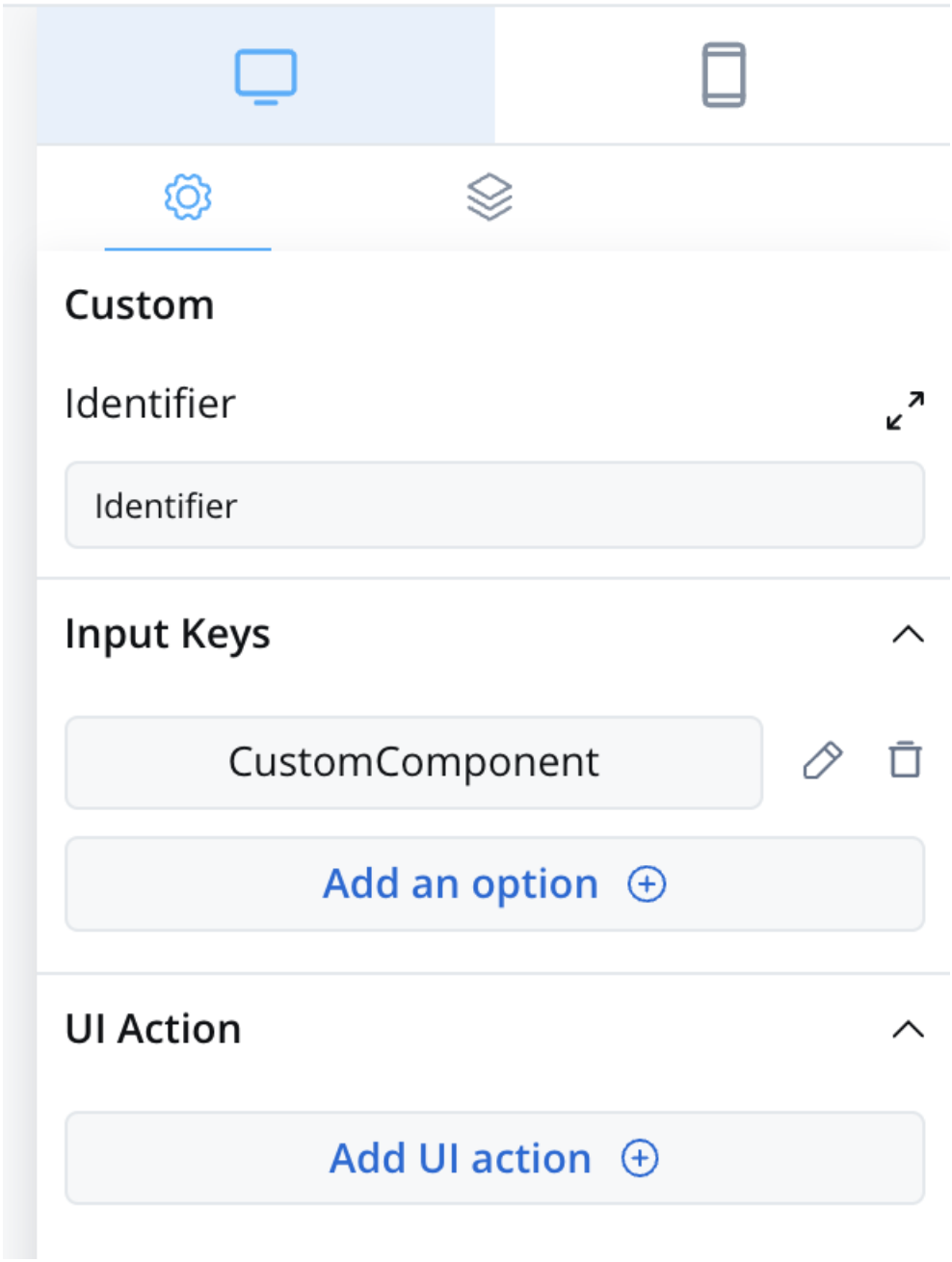
To add a custom component in the template config tree, we need to know its unique identifier and the data it should receive from the process model.



The properties that can be configured are as follows:

- **Identifier** - this will enable the custom component to be displayed in the component hierarchy and what actions are available for the component
- **Input keys** - used to define the process model paths from which the components will receive their data

- **UI Actions** - actions defined here will be made available to the custom component


Custom
Identifier
Identifier
Input Keys
CustomComponent
Add an option
UI Action
Add UI action

Display of User Interface Elements

When a process instance is initiated, the web application receives all the UI elements that can be displayed in the process under the `templateConfig` key.

When a user task is reached in the process instance, the **events-gateway** receive requests, triggering it to display the associated UI element.

Example:

1. Starting a process:

- The following is an example of starting a process instance via a **POST** request to `{{processUrl}}/api/internal/process/DemoProcess/start`:


```
{
  "processDefinitionName" : "DemoProcess",
  "tokens" : [ {
    "id" : 662631,
    "startNodeId" : null,
    "currentNodeId" : 662807,
    "currentNodeName" : null,
    "state" : "ACTIVE",
    "statusCurrentNode" : "ARRIVED",
    "dateUpdated" : "2023-02-09T12:23:19.464155Z",
    "uuid" : "ae626fda-8166-49e8-823b-fe24f36524a7"
  } ],
  "state" : "CREATED",
  "templateConfig" : [ {
    "id" : 630831,
    "flowxUuid" : "80ea0a85-2b0b-442a-a123-2480c7aa2dce",
    "nodeDefinitionId" : 662856,
```

```
    "componentIdentifier" : "CONTAINER",
    "type" : "FLOWX",
    "order" : 1,
    "canGoBack" : true,
    "displayOptions" : {
      "flowxProps" : { },
      "style" : null,
      "flexLayout" : {
        "fxLayoutGap" : 0,
        "fxLayoutAlign" : "start stretch",
        "fxLayout" : "column"
      },
      "className" : null,
      "platform" : "DEFAULT"
    },
    "templateConfig" : [ {
      "id" : 630832,
      "flowxUuid" : "38e2c164-f8cd-4f6e-93c8-39b7cdd734cf",
      "nodeDefinitionId" : 662856,
      "uiTemplateParentId" : 630831,
      "componentIdentifier" : "TEXT",
      "type" : "FLOWX",
      "order" : 0,
      "key" : "",
      "canGoBack" : true,
      "displayOptions" : {
        "flowxProps" : {
          "text" : "Demo text"
        },
        "style" : null,
        "flexLayout" : null,
        "className" : null,
        "platform" : "DEFAULT"
      },
      "expressions" : {
```

```
        "hide" : ""
      },
      "templateConfig" : [ ],
      "dataSource" : {
        "processData" : {
          "parentFlowxUuid" : null
        },
        "nomenclator" : {
          "parentFlowxUuid" : null
        }
      }
    } ]
  } ],
  "uuid" : "d985d128-ae45-4408-a643-1dd026a644d3",
  "generalData" : null,
  "backCounter" : 0,
  "startedByActionId" : null,
  "subProcesses" : null,
  "subprocessesUuids" : null,
  "baseUrl" : null
}
```


2. The following is an example of a progress message:


```
{
  "progressUpdateDTO": {
    "processInstanceUuid": "5f24c66f-04a7-433a-b64a-
a765d3b8121a",
    "tokenUuid": "11c32ba6-b3e7-4267-9383-25d69b26492c",
    "currentNodeId": 662856

  }
}
```

3. **ProgressUpdateDto** will trigger the **SDK** to search for the UI element having the same **nodeId** as the one from the web socket progress event

4. Additionally, it will ask for data and actions that are required for this component via a GET request `{{processUrl}}/api/process/5f24c66f-04a7-433a-b64a-a765d3b8121aa/data/662856`

Was this page helpful?