Client Design mode

The client design mode is created to make it possible for end users to edit the possitions and sizes of elements on a form.

When the form enters client design mode, the user can use drag & drop to relocate elements or change the size of elements.

In This Chapter

- Putting a Form in Client Design mode
- Responding to user actions
- Limiting user actions
- Restrictions

Putting a Form in Client Design mode

To enter client design mode using:

```
controller.setDesignMode(true)
```

To exit client design mode:

```
controller.setDesignMode(false)
```

⚠️ There can only be one form in client design mode at any time.

Responding to user actions

When the Form is in Client Design mode, users can move the elements or resize them. In order to respond to the user actions, the function that puts a Form in Client Design mode optionally takes a set of event handlers as parameters:

- **onDrag**: fires when the user starts dragging an element
  The method should return a DRAGNDROP constant for what mode is supported or a combination of 2 constants:
  DRAGNDROP.MOVE if only a move can happen,
  DRAGNDROP.COPY if only a copy can happen,
  DRAGNDROP.MOVE|DRAGNDROP.COPY if a move or copy can happen,
  DRAGNDROP.NONE if nothing is supported (drag should not start).
  To be able to cancel the drag if the user sets the element somewhere it should not go, the start positions should be saved here so they can be restored in the onDrop.
- **onDrop**: fires when the user drops an element that was being dragged
  This event handler can be used to retrieve the new position of the Element for persisting it
- **onSelect**: fires when a user selects an element
  Every element that is selected will get this event. If the event handler returns false the element will not be selectable by the user.
  It is also possible to apply this restriction on the element beforehand, see Limiting user actions. This option is preferred, especially in the Web Client, as using the onSelect event handler requires a callback to the server, thus a delay in the user experience
- **onResize**: Fires when a user has resized an element
  This event handler can be used to retrieve the new size of the Element for persisting it

Using these events the developer can respond to the changes made by the user, for example storing the modifications, as the changes made by the user are not automatically persisted.

Limiting user actions

It's possible to "annotate" elements with certain UIProperties that will affect their behavior when the Form they are on is put in client Design mode.

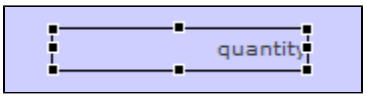
Through these properties, it's possible to restrict elements from being selectable and to provide information on which resize handles should be available on the element when selected

Making an element not selectable in Client Design mode:

```
elements.\{elementName}.putClientProperty(CLIENTDESIGN.SELECTABLE, false);
```

Restricting the available Resize handlers

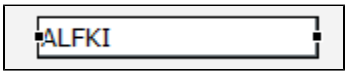
The selected element on a Form in Client Design mode by default shows resize handlers for all directions.



If the directions in which the element is resizable should be restricted, this is possible using the following code:

```
elements.\{elementName}.putClientProperty(CLIENTDESIGN.HANDLES, ['r', 'l']);
```

The code sample above will result in the following resize handlers:



The second argument in the code above is an Array with all the supported handlers. These are the available values:

Value	Resize handler
't'	Top
'b'	Bottom
'r'	Right
'l'	Left
'bl'	Bottom left corner
'br'	Bottom right corner
'tl'	Top left corner
'tr'	Top right corner

⚠ These are runtime properties and will be lost by use of controller.recreateUI().

Restrictions

The following restrictions are applicable when working with Client Design mode:

- Elements cannot be moved or sized outside of the dimensions of the Form part they are located on
- Elements need to have a name in order to be able to address them in code
- The changes made by the user are not automatically persisted. The developer has to take care of this. The changes will be lost when the fform is unloaded or when controller.recreateUI() is called on the Form