

Using transform function with smart fields

Short recipe to show how to use "transform" function with smart fields. See more about [Virtual functions](#) here

Consider having 2 forms

CONFIF
definition

ConfiForms Form (Definition) | formName = choices

CONFIF
virtual

ConfiForms Form Registration Control

CONFIF
definition

ConfiForms Form Field (Definition) | fieldName = choice | fieldL...

using choices in multi dropdown

CONFIF
definition

ConfiForms Form (Definition) | formName = f

CONFIF
virtual

ConfiForms Form Registration Control

CONFIF
definition

ConfiForms Form Field (Definition) | fieldName = purpose | field...

One form with a field "choice" and of type "text"

Another form with a field "purpose" and of type "smart dropdown"

Let's say we have these options in form nr.1

Choice	Edit
Wifi	<div>EditDelete</div>
Software	<div>EditDelete</div>
Hardware	<div>EditDelete</div>

These values are available as dropdown items in the form nr.2

Purpose

Hardware × Software ×

+

Save

Close

If we want to show the "labels" of the "choice" field (form nr.1) on the ListView which is configured to show values of form nr.2

And we want these values to be shown as

```
{ "value": "Hardware" }, { "value": "Software" }
```

Something, JIRA expects for multi-select fields

We will need to put the following expression into the field name

Edit 'ConfiForms Form Field' Macro

Field macro for ConfiForms dynamic forms. Use it with views or registration control [Documentation](#)

Field name *

purpose.transform(choice).asArray

Name of the field defined on the form or expression when used in TableView Merger macro

☐ Show value with label

Label from ConfiForms Field Definition will be added as message before the field value

Override label name to use for this field

When left empty the Field Definition label will be used

Preview

purpose.transform(choice).asArrayMultiSelect

Select macro

Save

Cancel

or, alternatively, if we need it without any styling and pure text value

```
[entry.purpose.transform(choice).asArrayMultiSelect]
```

This translates into the following:

- Take the value of field "purpose" (form nr.2)
- Apply **transform** function on the field "choice" (this is already a form nr.1, as field purpose is a smart field and it references values in another form)

- Apply another function (**asArrayMultiSelect**) on the values

Which will give us the result we need.



There is no limits on a number of functions you can chain, and there is no limits on the depth of the references to another forms (can navigate as much as you need)

More on [Virtual functions](#) here