Possible Values: Value and Label

Special Feature of the Pair: Value & Label

Some form elements do not contain only one value, but a value pair. This currently applies to the following elements:

- **Drop-down Field**: The element must be preset with possible values so that users can select a value from the defaults.
- **Radio Button**: The element must be preset with possible values so that users can select a value from the defaults.
- **Search Field**: This element also references a list of possible values, but in contrast to a drop-down field or a radio button, these values are specified using JavaScript code.
- **URL**: This element can contain two entries: In addition to the URL, a display name can be saved.

All these elements have a special feature: Input is saved as **value pair**.

When possible values are listed, as for example in the settings of a radio button, the designer can enter a value and a label - separated by a semicolon. The input in these four elements is always stored in the format **value;label.** The first value (**value**) is stored as a key in the data container, the second value (**label**) is used for displaying to the user. The contents of **value** and **label** are stored in the container as strings.

> Even if only simple values are specified in the element's setting **Possible Values**, the input is still saved as **value;label** pair.

This particular feature must be taken into account when configuring columns of instance tables and defining conditions in EPCs with XOR branching.

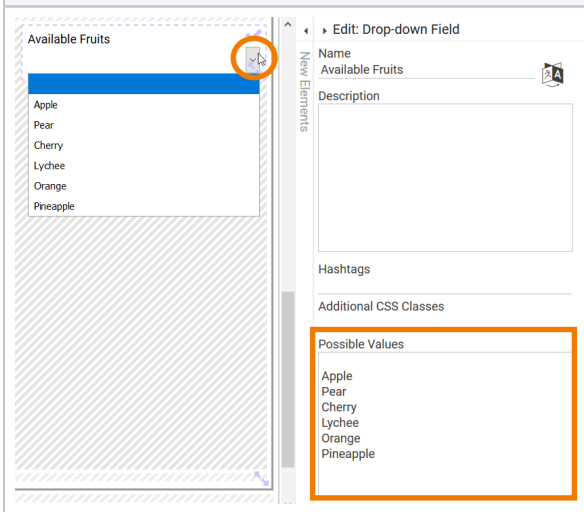
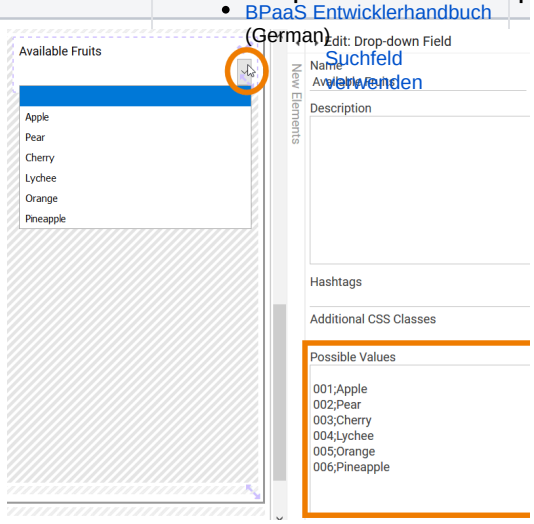
Why use Value Pairs?

Saving a pair of values is particularly useful if you want to use technical data in a form, for example from another system, but want to present the data in an understandable language to the user.

> It is not possible to assign the same value to different labels.

Example: Drop-down field with and without value pairs

In a form, the drop-down field Available Fruits has been inserted. The values Apple, Pear, Cherry, Lychee , Orange and Pineapple shall be displayed.

Variant 1: Drop-down field with simple input	Variant 2: Drop-down field with value p
	

On this Page:

- Special Feature of the Pair: Value & Label
  - Why use Value Pairs?
  - Handling Possible Values in Column Configuration
  - Value Pairs in EPCs with XOR Branching

Related Pages:

- Modeling EPCs
  - Configuring EPC Branching
- Form Elements
  - Drop-down Field
  - Radio Button
  - Search Field
- Creating Forms
  - Configuring Tables
- Process Apps
  - Configuring Overview Tables
- Service
  - Procurement Process

Related Documentation:

- BPaaS Entwicklerhandbuch (German)
  - Suchfeld verwenden

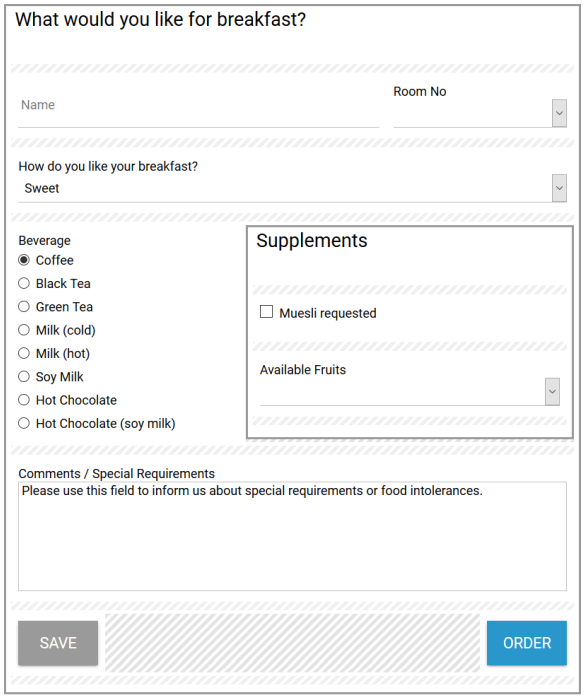
Variant 1: All fruit varieties are listed in the **Possible Values** field. The selection list shows the names of the fruit varieties from which the user can select one.	**Variant 2**: All fruit varieties are listed in the **Possible Values** field as a Just as in variant 1, the drop-down field for the user only sh varieties, the respective **label**. At the same time, the associated number (=**value**), is store Let's assume that in an ERP system the fruit varieties are s numbers: When an order is placed, it is now possible to dire thus the numbers that the ERP system "understands".

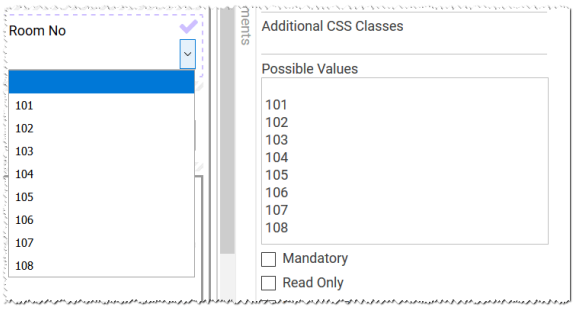
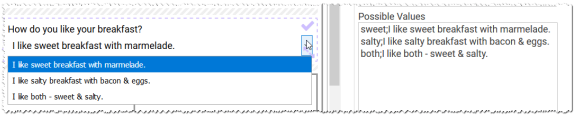
Handling Possible Values in Column Configuration

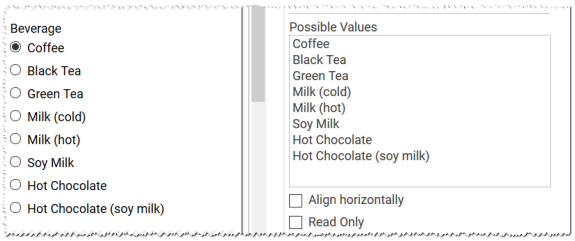
A form element with possible values must be correctly referenced in the Column Configuration of an instance table. Since the value pair consists of two specifications, you must specify in the column configuration which of the two values is to be displayed: **value** or **label**. Therefore, you have to extend the entry in the column **Field Name in Container** with the corresponding information:

- `ElementName.value`
- `ElementName.label`

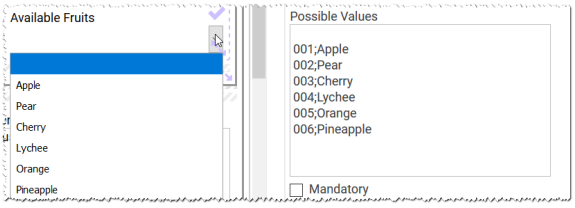
Example: Breakfast App

	A hotel offers its guests to order breakfast via an app.

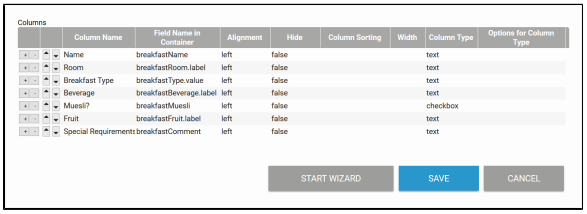
	The following fields containing value pairs can be filled out in the form: • Room number (**Drop-down Field** with possible values) • **Field Name in Container:** breakfastRoom
	• How do you like your breakfast? (**Drop-down Field** with value pairs) • **Field Name in Container:** breakfastType



- Beverage (**Radio Button** with possible values)
- **Field Name in Container:** breakfastBeverage

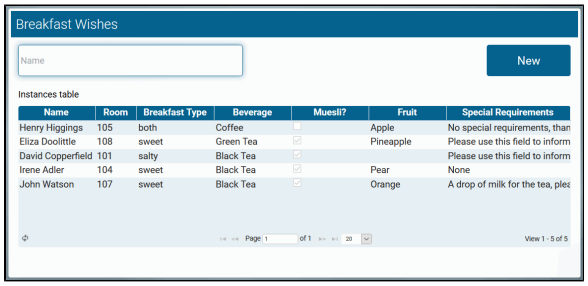


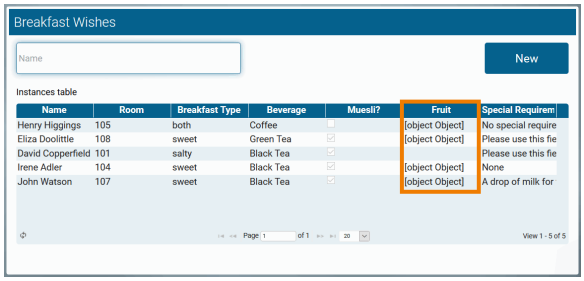
- Available Fruits (**Drop-down Field** with value pairs)
- **Field Name in Container:** breakfastFruit



During the configuration of the overview, the designer needs to specify, what data he wants to display - the value or the label of the element.

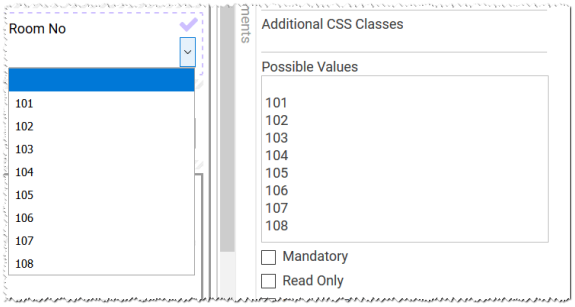
In our example, the overview is configured to show the following data:

	• The label of field Room No (`breakfastRoom.label`) • The value of field How do you like your breakfast? (`breakfastType.value`) • The label of field Beverage (`breakfastBeverage.label`) • The label of field Available Fruits (`breakfastFruit.label`)
	During execution of the overview, the instance data is displayed as configured.



If you forget to specify to which part of the value pair you want to address, **[object Object]** will be displayed.

[object Object] indicates, that an object exists. But which content of the value pair is to be displayed was not defined correctly.



Even if the possible values have not been entered as value pairs, their content is still saved as **value; label** pair. It therefore doesn't matter whether you use the label or value extension to reference the content.

Example: Room No

The possible values of the drop-down field Room No contain only a single entry for each line.

		Column Name	Field Name in Container
+	-	Name	breakfastName
+	-	Room Value	breakfastRoom.value
+	-	Room Label	breakfastRoom.label
+	-	Breakfast Type	breakfastType.value

During column configuration, you may use both extensions to the **Field Name in Container**:

- `breakfastRoom.value`
- `breakfastRoom.label`

Instances table

Name	Room Value	Room Label	Breakfast Type
Henry Higgings	105	105	both
Eliza Doolittle	108	108	sweet
David Copperfie	101	101	salty
Irene Adler	104	104	sweet
John Watson	107	107	sweet

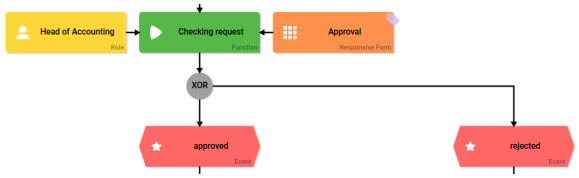
Either entry leads to the desired result.

Value Pairs in EPCs with XOR Branching

In EPCs with XOR Branching, conditions are defined. The conditions are checked when the process is executed. Depending on the result of the check, the user runs through a different branch of the process.

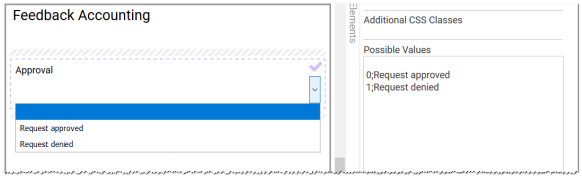
If conditions are defined that refer to the possible values of form elements, the distinction between **value** and **label** must also be taken into account to make sure that the check delivers a valid result.

Example: ACME's Procurement Process



In ACME's Procurement Process the Head of Accounting has to approve a procurement request if it exceeds the specified budget.

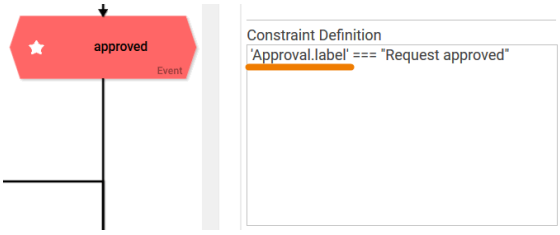
He checks the request and then decides whether to approve or reject the order.



In the approval form the Head of Accounting finds the drop-down field Approval.

The field contains the **Possible Values**

- 0; Request approved
- 1; Request denied



The constraint definitions for the next process steps are saved in the two event elements:

- approved: **'Approval.label' === "Request approved"**
- rejected: **'Approval.label' !== "Request approved"**

When the process reaches this step, the system therefore checks whether the drop-down field Approve was set to Request approved in the Approval form or not.

Sinc

e
a
v
a
l
u
e
p
a
ir
w
a
s
e
n
t
e
r
e
d
f
o
r
t
h
e
d
r
o
p
-
d
o
w
n
fi
e
l
d
,
t
h
e
c
o
n
d
it
i
o
n
s
c
o
u
l
d
a
ls
o
b
e
c
h
e
c
k
e
d
w
it
h
t
h
e
v
a
l

	u e: ' A p p r o v a l. v a l u e ' = = = 0 ' A p p r o v a l. v a l u e ' ! = = 0

If you forget to specify which part of the value pair you want to check in the constraint definition, the process can be run through. However, the same branch is always run, regardless of which condition is fulfilled.