

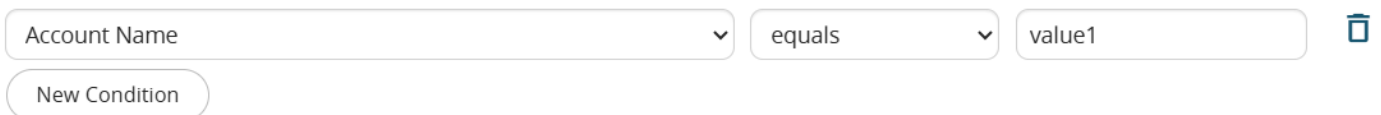
# 2.8 Comparison Operators: Description and Use in Conditional Tasks.

The system provides several comparison operators that can be used within Conditional Tasks, which are the types of tasks that allow you to perform checks on the fields of the dynamic forms/forms involved within the process.

WARNING! -> the system is "Case Sensitive," so the same values with different upper or lower case are treated as if they were different.

Example: "VTENEXT" and "vtenext" are seen by the system as different values.

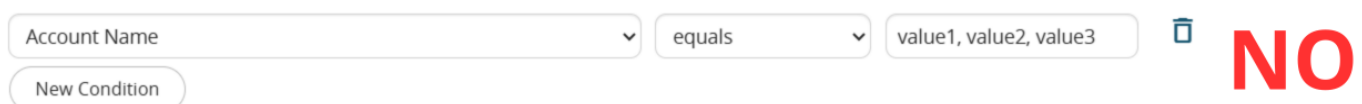
- "**equals**" -> allows checking whether the contents of a field are equal to a static value specified within the condition. (Figure 1)



The screenshot shows a configuration interface for a conditional task. It consists of three main components: a dropdown menu on the left containing the text "Account Name", a middle dropdown menu containing the text "equals", and a text input field on the right containing the text "value1". To the right of the text input field is a small trash icon. Below these three components is a rounded rectangular button with the text "New Condition".

Figure 1

PLEASE NOTE: the system does not allow you to compare multiple values at the same time on the same row, so in case you need to perform a multiple check you will have to create separate conditions (Figure 2 and 3)



The screenshot shows a configuration interface similar to Figure 1, but with an invalid condition. The dropdown menu on the left contains "Account Name", the middle dropdown menu contains "equals", and the text input field contains "value1, value2, value3". To the right of the text input field is a trash icon. To the right of the trash icon is the word "NO" in large, bold, red capital letters. Below these components is a rounded rectangular button with the text "New Condition".

Figure 2

Account Name equals value1 or  
Account Name equals value2 or  
Account Name equals value3  
New Condition YES

Figure 3

- “**different**” -> allows to check whether the contents of a field are NOT equal to a static value specified within the condition. (Figure 4)

Account Name not equal to value1  
New Condition

Figure 4

- “**contains**” -> allows checking whether the static value specified within the condition is present within a field, regardless of where it is located. (Figure 5)

Taking the example given in Figure 5 as a reference, in case we have a company named “demo\_vtenext”, the condition will be verified, since, this tringa contains “vte”.

Account Name contains vte  
New Condition

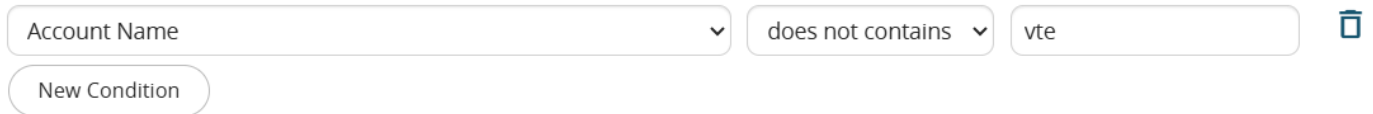
Figure 5

In contrast to the “equal” operator, the system allows multiple values to be compared simultaneously on the same row (Figure 6)

Account Name contains vte, test, demo  
New Condition

Figure 6

- **“does not contain”** -> allows checking whether the static value specified within the condition is NOT present within a field, regardless of where it is located. (Figure 7)

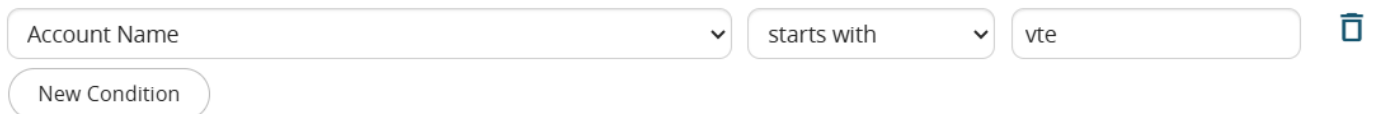


The screenshot shows a condition builder interface. It consists of three main input fields in a row: a dropdown menu with 'Account Name' selected, a dropdown menu with 'does not contains' selected, and a text input field with 'vte' entered. To the right of these fields is a trash icon. Below the first dropdown is a button labeled 'New Condition'.

Figure 7

As with the “contains” operator, multiple values can be compared at the same time on the same row (which should be entered separated with a comma).

- **“starts for”** -> allows you to check whether the contents of a field begin with the static value specified within the condition. (Figure 8)

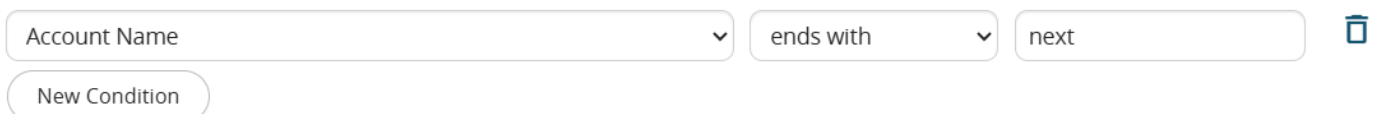


The screenshot shows a condition builder interface. It consists of three main input fields in a row: a dropdown menu with 'Account Name' selected, a dropdown menu with 'starts with' selected, and a text input field with 'vte' entered. To the right of these fields is a trash icon. Below the first dropdown is a button labeled 'New Condition'.

Figure 8

PLEASE NOTE: the system does not allow you to compare multiple values at the same time on the same row, so in case you need to perform a multiple check you will have to create separate conditions (take Figure 2 and 3 as reference)

- **“ends with”** ->allows you to check whether the contents of a field ends with the static value specified within the condition. (Figure 9)



The screenshot shows a condition builder interface. It consists of three main input fields in a row: a dropdown menu with 'Account Name' selected, a dropdown menu with 'ends with' selected, and a text input field with 'next' entered. To the right of these fields is a trash icon. Below the first dropdown is a button labeled 'New Condition'.

Figure 9

PLEASE NOTE: the system does not allow you to compare multiple values at the same time on the same row, so in case you need to perform a multiple check you will have to create separate conditions (take Figure 2 and 3 as reference)

- **“has changed to”** -> allows you to check whether the contents of the field has precisely changed to the value specified within the condition (Figure 10)

The screenshot shows a single condition row. On the left is a dropdown menu with 'Type' selected. To its right is another dropdown menu with 'has changed to' selected. Further right is a third dropdown menu with 'Partner' selected. To the far right is a trash icon. Below this row is a rounded button labeled 'New Condition'.

Figure 10

For all those types of fields that might have an undefined value (text, number, date, related to, etc. ), by combining the “has changed to” operator with the “different” operator, it is possible to configure a condition to capture a change in the content of a field to a consistent value that you are not aware of. (Figure 11)

The screenshot shows two condition rows. The first row has a dropdown menu with 'Billing Address' selected, followed by a dropdown menu with 'has changed to' selected, and an empty text input field. The second row has a dropdown menu with 'Billing Address' selected, followed by a dropdown menu with 'not equal to' selected, and another empty text input field. To the right of the second row is an 'And' dropdown menu and a trash icon. Below the rows is a rounded button labeled 'New Condition'.

Figure 11

The example shown in Figure 11 translates into the following sentence, “Expected closing date has changed to a value I cannot define but, at the same time, that value is different from empty, so it will necessarily be a consistent value.”

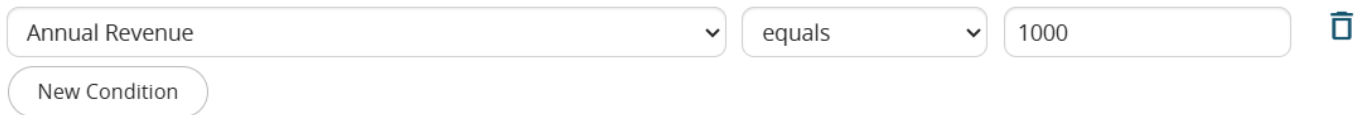
- **“has changed from”** -> allows you to check whether the contents of the field has changed from one value to another, both of which are specified within the condition (Figure 12)

The first slot should specify the value that the field had before the change, while the second slot should specify the new value that was entered.

The screenshot shows a single condition row. On the left is a dropdown menu with 'Type' selected. To its right is another dropdown menu with 'has changed from' selected. Further right are two dropdown menus: the first has 'Prospect' selected and the second has 'Partner' selected, with the word 'to' between them. To the far right is a trash icon. Below this row is a rounded button labeled 'New Condition'.

Figure 12

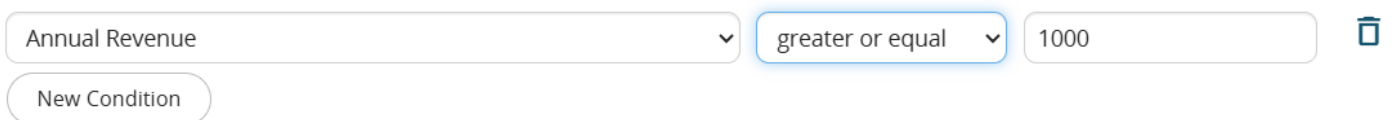
- **“greater than”** -> allows you to check whether the content of a number/currency field present is greater than an entered static value specified within the condition. (Figure 13)



The screenshot shows a condition builder interface. It consists of three main input fields: a dropdown menu containing 'Annual Revenue', a second dropdown menu containing 'equals', and a text input field containing '1000'. To the right of these fields is a trash icon. Below the first dropdown is a button labeled 'New Condition'.

Figure 13

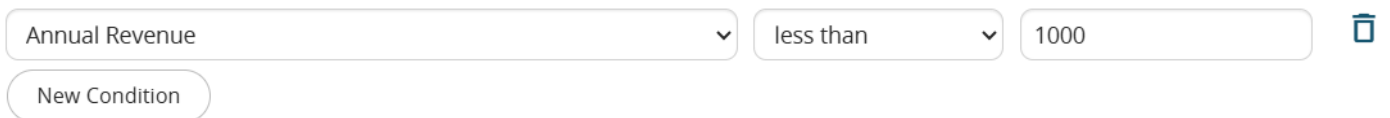
- **“greater than or equal”** -> allows checking whether the content of a number/currency field present is greater than or equal to an entered static value specified within the condition. (Figure 14)



The screenshot shows a condition builder interface. It consists of three main input fields: a dropdown menu containing 'Annual Revenue', a second dropdown menu containing 'greater or equal', and a text input field containing '1000'. To the right of these fields is a trash icon. Below the first dropdown is a button labeled 'New Condition'.

Figure 14

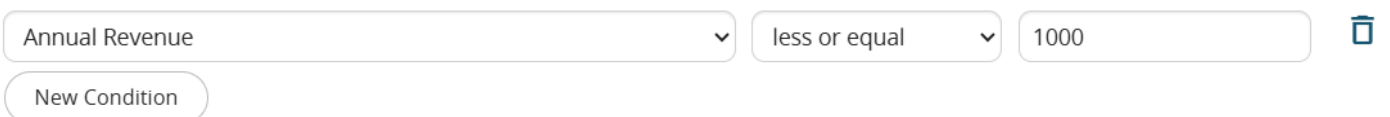
- **“less than”** -> allows you to check whether the content of a number/currency field present is less than an entered static value specified within the condition. (Figure 15)



The screenshot shows a condition builder interface. It consists of three main input fields: a dropdown menu containing 'Annual Revenue', a second dropdown menu containing 'less than', and a text input field containing '1000'. To the right of these fields is a trash icon. Below the first dropdown is a button labeled 'New Condition'.

Figure 15

- **“less than or equal”** -> allows checking whether the content of a number/currency field present is less than or equal to an entered static value specified within the condition. (Figure 16)



The screenshot shows a condition builder interface. It consists of three main input fields: a dropdown menu containing 'Annual Revenue', a second dropdown menu containing 'less or equal', and a text input field containing '1000'. To the right of these fields is a trash icon. Below the first dropdown is a button labeled 'New Condition'.

Figure 16

---

Revision #3

Created 2025-05-20 14:44:58 UTC by Riccardo

Updated 2025-05-29 13:31:14 UTC by Riccardo