# 12 Standard Process SDK: Description and Usage


- 12.1 SDK fieldAction: Sum
- 12.2 SDK fieldAction: Difference
- 12.3 SDK fieldAction: Multiplication
- 12.4 SDK fieldAction: Division
- 12.5 SDK fieldAction: vte_json_column_fields
- 12.6 SDK fieldAction: vte_json_string
- 12.7 SDK fieldAction: vte_json_field_string
- 12.8 SDK fieldAction: vte_json_record
- 12.9 SDK fieldAction: Format Date
- 12.10 SDK fieldAction: Now Date
- 12.11 SDK fieldAction: Diff Date
- 12.12 SDK fieldAction: Set lead converted
- Nuova pagina12.13 SDK fieldAction: vte_formula

# 12.1 SDK fieldAction: Sum

This SDK function allows you to perform the sum between 2 or more values that must be passed as parameters.
Being a fieldAction type function, it can be called within the individual fields of modules or dynamic forms.

EXAMPLE OF USE

To better understand how it works, below is an example of using the vte_sum() function to perform the sum between the values of two fields called "Estimated Development Hours" and "Estimated Training Hours" and save the result in the "Total Estimated Hours" field (Figure 1)

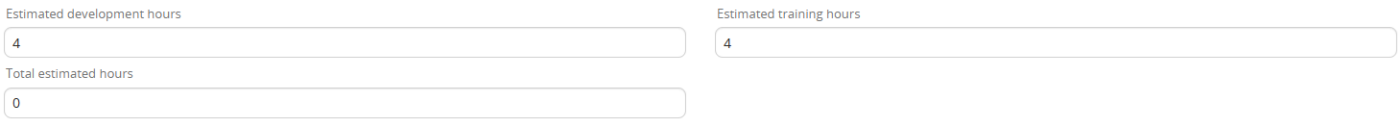


Figure 1

Within an Action Task we proceed with the configuration of an Update Entity action involving the "Total Estimated Hours" field.
Specifically, we are going to call the interested SDK function through the "Option Selection" picklist and accessing the "SDK Functions" section (Figure 2)

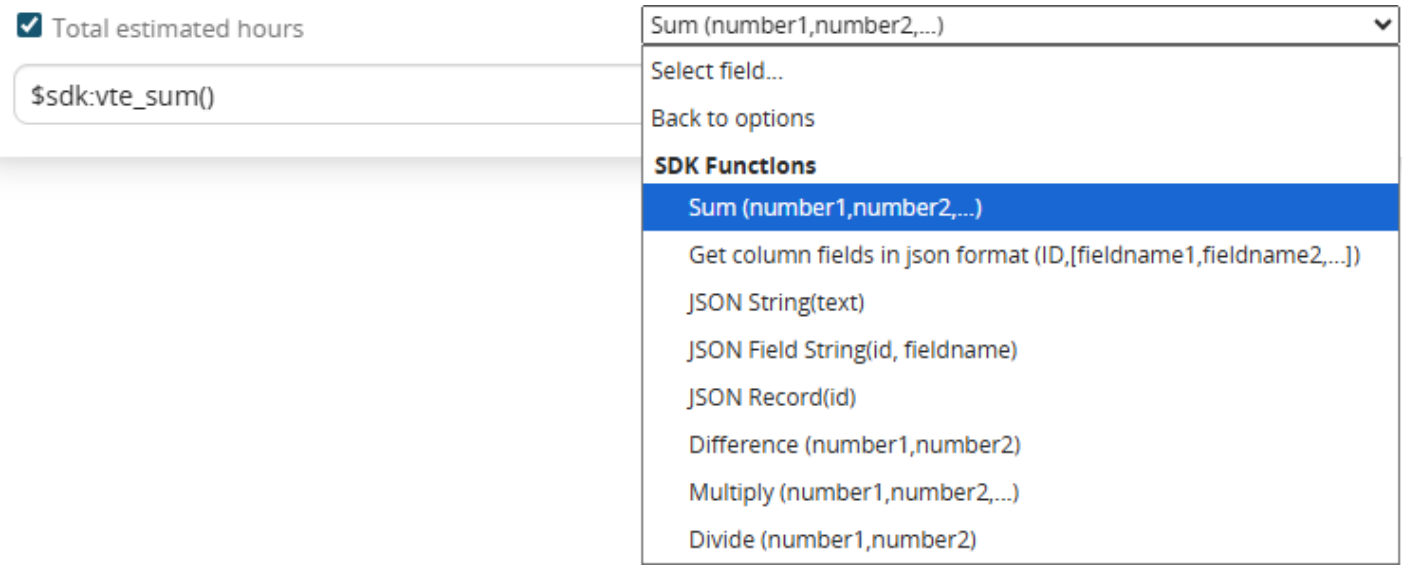

Figure 2

Finally we pass as parameters (separated by commas) the contents of the two fields to be added (Figure 3)

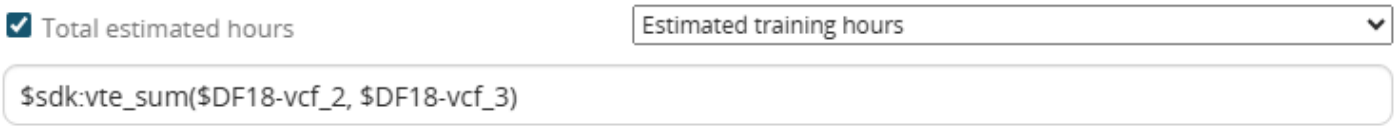


Figure 3

# 12.2 SDK fieldAction: Difference

This SDK function allows you to perform the difference between 2 or more values that must be passed as parameters.
Being a fieldAction type function, it can be called within the individual fields of modules or dynamic forms.

EXAMPLE OF USE

To better understand how it works, below is an example of using the vte_diff() function to perform the difference between the values of two fields called "Expected hours" and "Used hours" and save the result in the "Remaining hours" field (Figure 1)

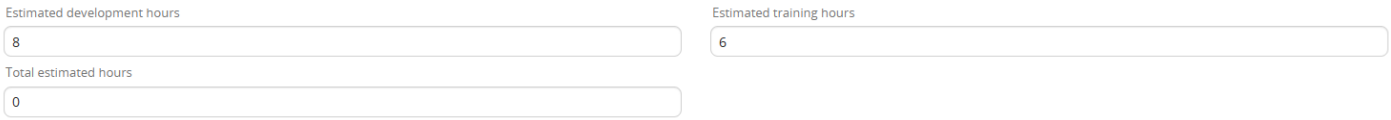


Figure 1

Within an Action Task we proceed with the configuration of an Update Entity action involving the "Remaining Hours" field.
Specifically, we are going to call the interested SDK function through the "Option Selection" picklist and accessing the "SDK Functions" section (Figure 2)

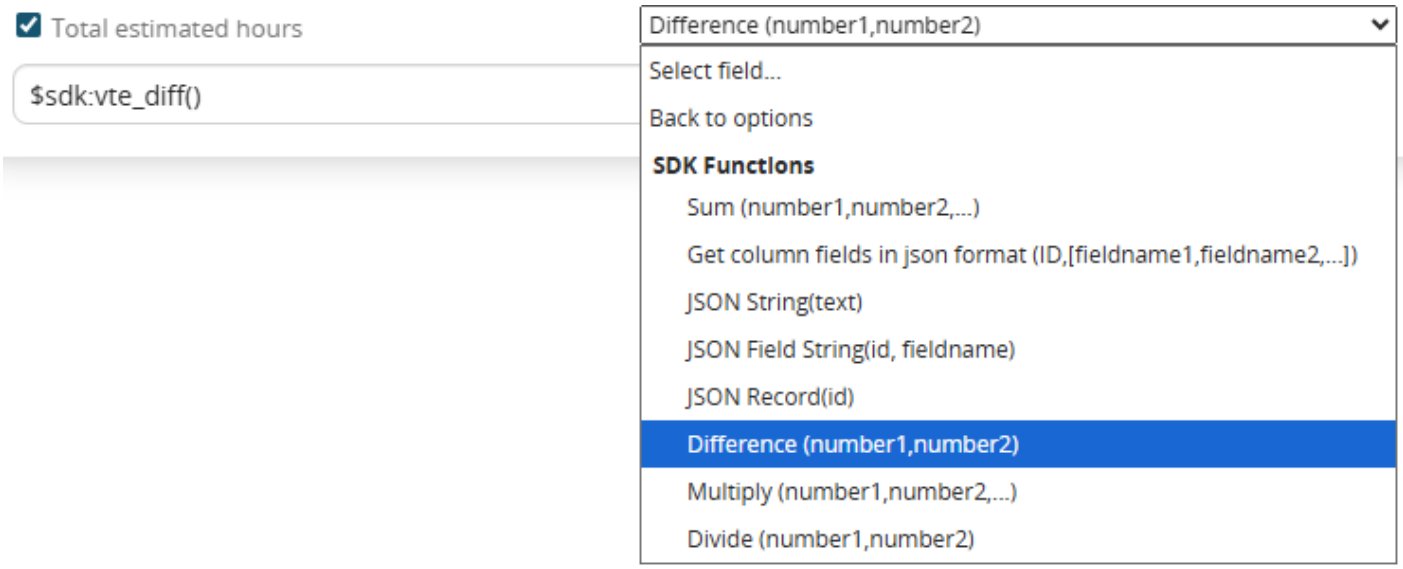


Figure 2

Finally we pass as parameters (separated by commas) the contents of the two fields to be subtracted (Figure 3)

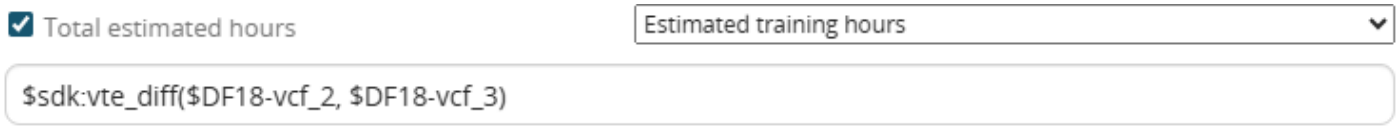


Figure 3

# 12.3 SDK fieldAction: Multiplication

This SDK function allows you to perform the multiplication between 2 or more values that must be passed as parameters.
Being a fieldAction type function, it can be called within the individual fields of modules or dynamic forms.

EXAMPLE OF USE

To better understand how it works, below is an example of using the vte_mul() function to perform the multiplication between the contents of the two fields called "Fixed cost per km: (€)" and "Number of km traveled" and save the result in the "Total cost: (€)" field (Figure 1)


Figure 1

Within an Action Task we proceed with the configuration of an Update Entity action involving the "Total Cost: (€)" field.
Specifically, we are going to call the interested SDK function through the "Option Selection" picklist and accessing the "SDK Functions" section (Figure 2)

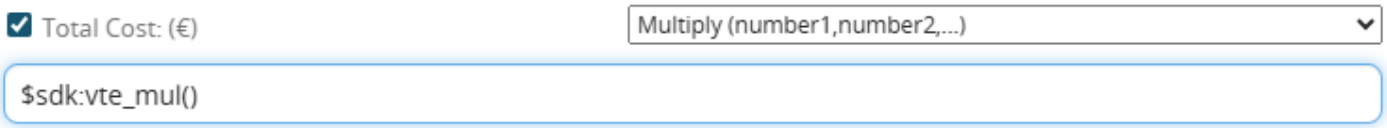


Figure 2

Finally we pass as parameters (separated by commas) the two values to be multiplied (Figure 3)

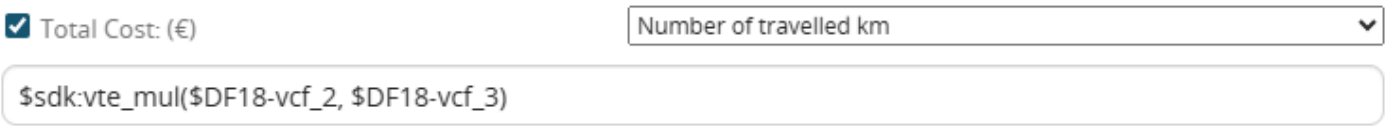

Figure 3

# 12.4 SDK fieldAction: Division

This SDK function allows you to perform the division between 2 or more values that must be passed as parameters.
Being a fieldAction type function, it can be called within the individual fields of modules or dynamic forms.

EXAMPLE OF USE

To better understand how it works, below is an example of using the vte_div() function to perform the division between the "Estimated Development Hours" field and a fixed value of 8 (which represents the working hours in a day) and save the result in the "Estimated Development Days" field (Figure 1)


Figure 1

Within an Action Task we proceed with the configuration of an Update Entity action involving the "Estimated Development Days" field.
Specifically, we go to call the interested SDK function through the "Option Selection" picklist and accessing the "SDK Functions" section (Figure 2)

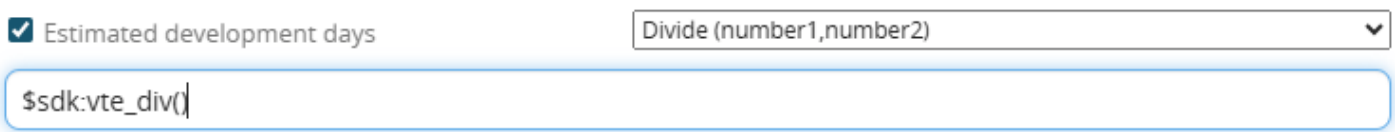


Figure 2

Finally we pass as parameters (separated by commas) the two values to divide (Figure 3)

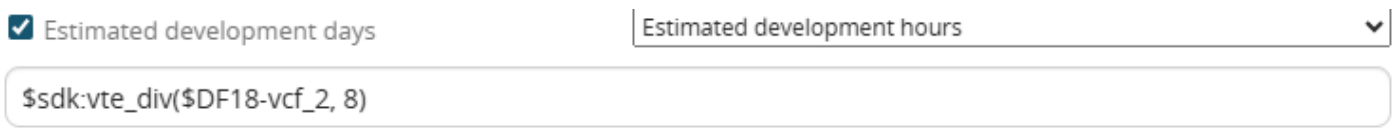
Estimated development days
Estimated development hours
$sdk:vte_div($DF18-vcf_2, 8)

Figure 3

# 12.5 SDK fieldAction: vte_json_column_fields

This SDK function allows you to generate a JSON code containing the labels and values of the 2 or more fields passed as parameters.

As the first parameter, you must pass the crmid of the record from which you want to extract the information, instead as subsequent parameters you must pass the "fieldname" of the fields to be included, or the names in which those fields are registered in the Database.

NOTE: even if in the description of the function it is indicated to pass the parameters following the id by inserting square brackets, the latter must not be inserted.

It is mainly used to format a subset of data in order to perform REST-type Web Service calls (always configurable by process through the dedicated standard action "Call External Web Service").

For more information on configuring API calls from the process, please refer to chapter 3.15 of the process manual.

EXAMPLE OF USE

To better understand how it works, below is an example of using the SDK function vte_json_column_fields() to generate a JSON code containing the following fields and values of an instance of the Customer Service module involved in the process:

Title
Status

Inside the dynamic form of a process helper, we proceed with the creation of a text area field called "Body JSON Format" in which the result of the function will be saved.
Then we call the interested SDK function through the "Option Selection" picklist and access the "SDK Functions" section (Figure 1)

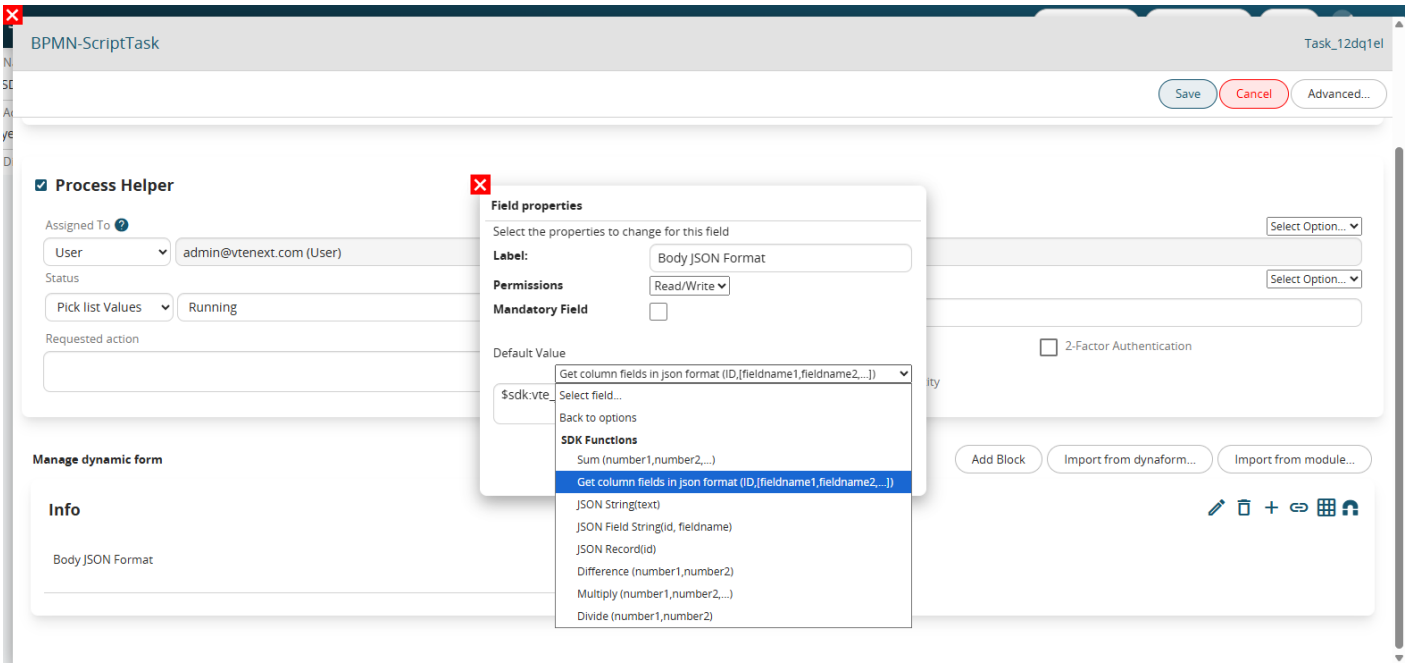


Figure 1

Finally we pass the parameters required by the function, all separated by commas.
Specifically, as the first parameter we insert the crmid of the record from which we want to extract the information, so in our case the ticket id. (Figure 2)

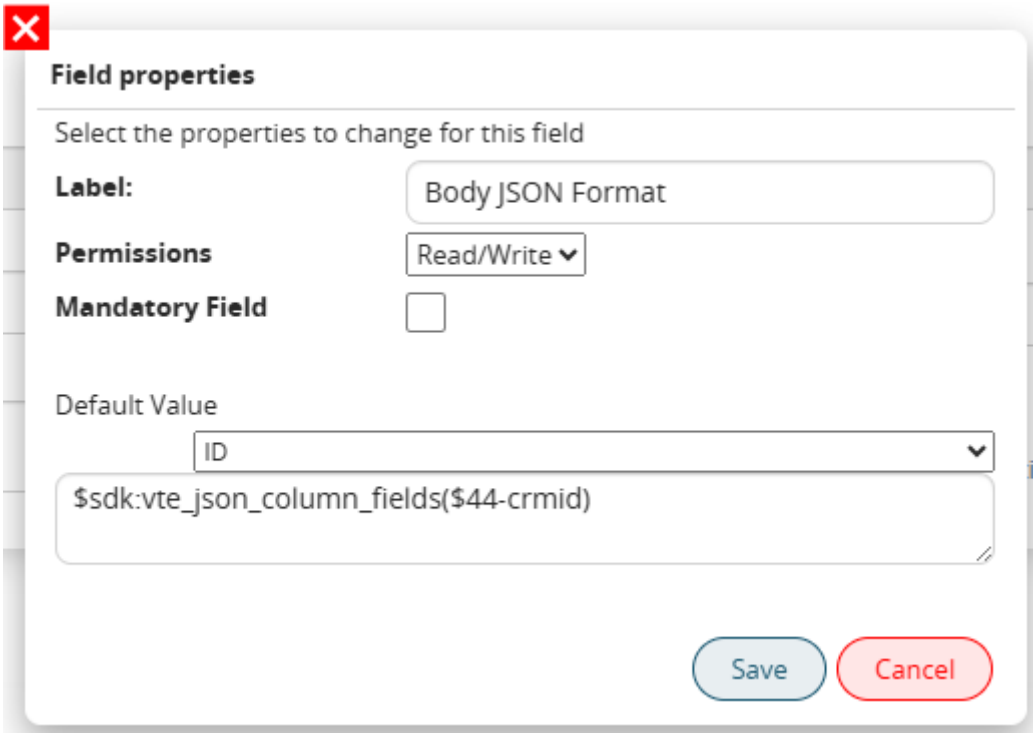


Figure 2

Instead, as subsequent parameters, we insert the "fieldnames" of the fields to be involved, that is, the names with which those fields are registered in the Database.
In our specific case, they will be "ticket_title", "ticketstatus" and "ticketpriorities". (Figure 3)

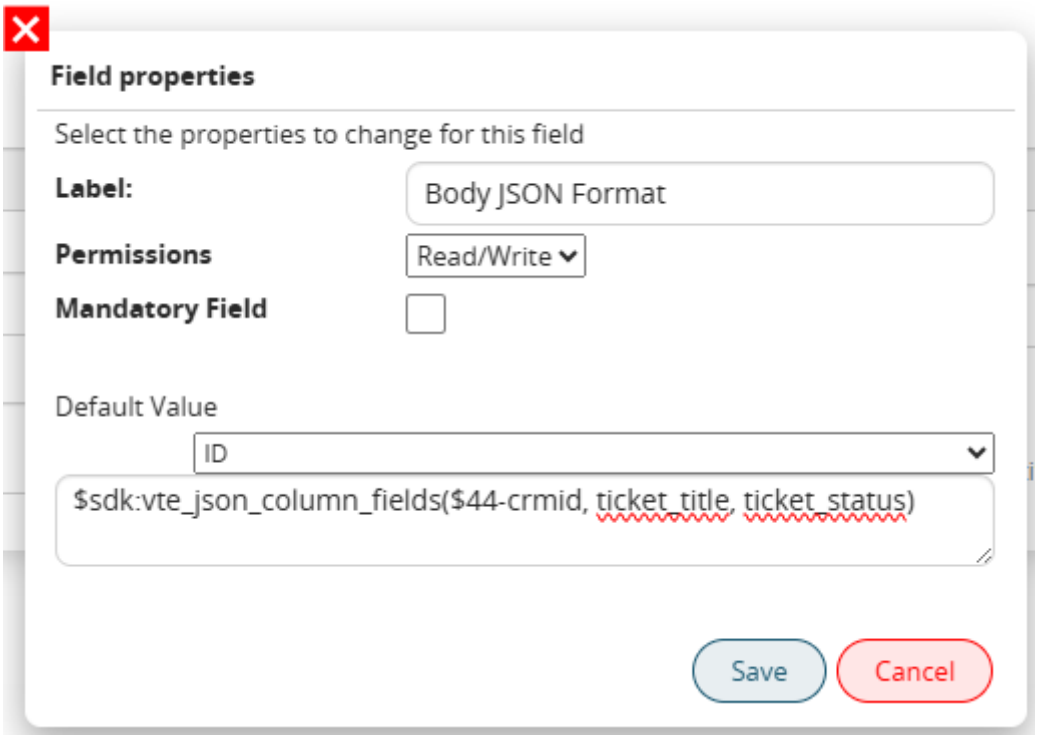


Figure 3

To easily obtain them, simply select the relevant field from the "Option Selection" picklist and delete the reference to the instance involved in the process, i.e. the metaid (Figure 4).

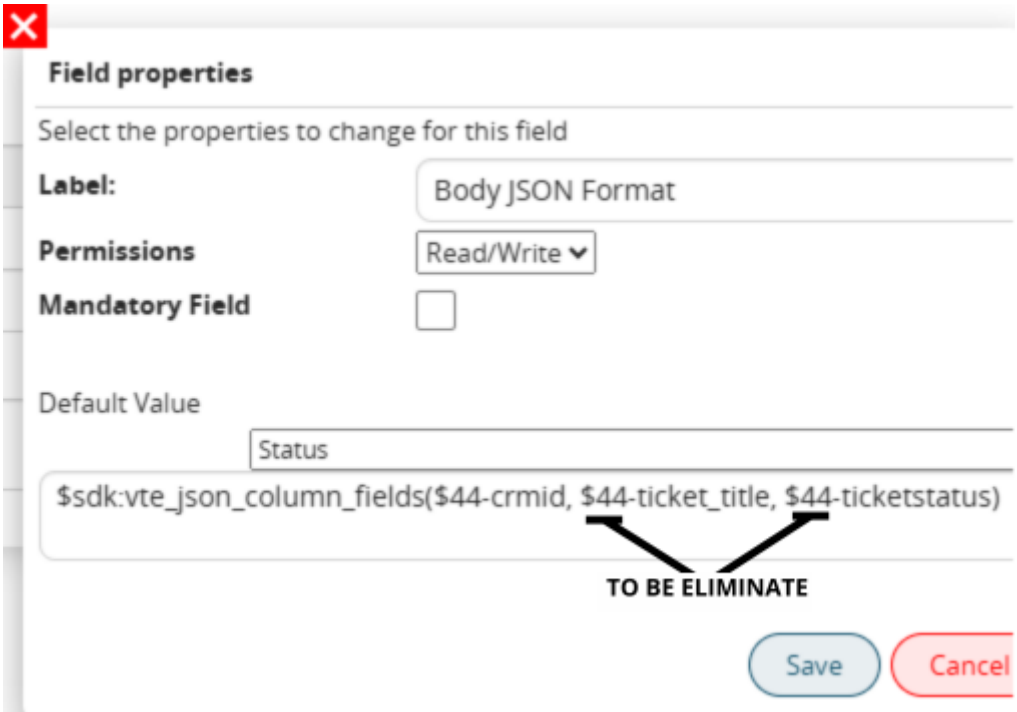


Figure 4

The generated JSON code will be as shown in Figure 5

Body JSON Format

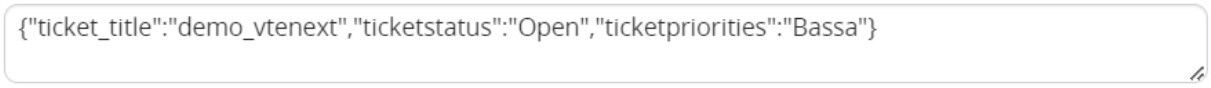


Figure 5

# 12.6 SDK fieldAction: vte_json_string

This SDK function allows you to convert a string into JSON format by managing compatibility with special characters, so as to format them correctly and avoid conflicts with it.

It is mainly used to format strings in order to perform REST Web Service calls (always configurable by process via the dedicated standard action "Call External Web Service") without encountering syntax errors on the JSON.

Specifically, if the string contains special characters used in JSON syntax such as double quotes ("), slash (/) and backslash (\), the function will automatically insert additional backslashes to prevent the JSON from generating an error.

As the only parameter, therefore, we pass the content of any field (as long as it is a string) or a simple static value.

EXAMPLE OF USE

To better understand how it works, below is an example of using the SDK function vte_json_string() to format the "Description" field containing the following string:

And \ at that point / I told him: "Goodbye!" (Figure 1)

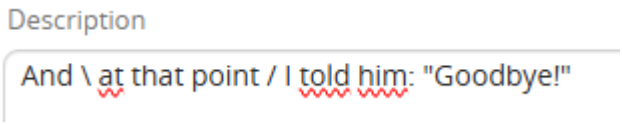


Figure 1

Within an Action Task we proceed with the configuration of an Update Entity action involving the "Description" field.
Specifically, we go to call the interested SDK function through the "Option Selection" picklist and accessing the "SDK Functions" section (Figure 2)

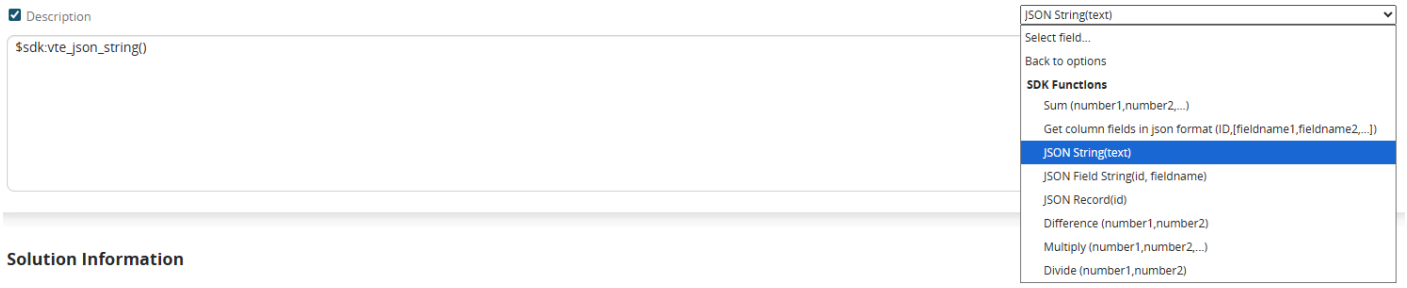


Figure 2 (click on the image for a higher graphic resolution)

Finally we pass as the only parameter the content of the field to be formatted (Figure 3)

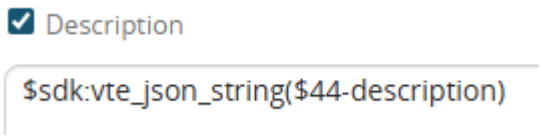


Figure 3

The result generated by the function will be as shown in Figure 4

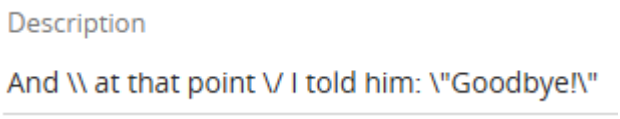


Figure 4

# 12.7 SDK fieldAction: vte_json_field_string

It is a function similar to vte_json_string(), in fact it allows you to convert a string in JSON format in the same way by managing compatibility with special characters, so as to format them correctly and avoid conflicts with it.

It is also mainly used to format strings in order to perform REST type Web Service calls (always configurable by process via the dedicated standard action "Call External Web Service") without encountering syntax errors on the JSON.

Specifically, if the string contains special characters used in JSON syntax such as double quotes ("), slash (/) and backslash (\), the function will automatically insert additional backslashes to prevent the JSON from generating an error.

For more information on configuring API calls from the process, please see chapter 3.15 of the process manual.

The real difference lies in the parameters that can be entered in input.
The first parameter is the crmid of the record from which you want to extract the information, while the second parameter is the "fieldname" of the field to be included, i.e. the name in which that field is registered in the database.

EXAMPLE OF USE

To better understand how it works, below is an example of using the SDK function vte_json_field_string() to format the "City (Invoicing)" field of an instance of the Company module involved in the process containing the following string in JSON code:

"Verona" (Figure 1)

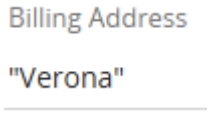


Figure 1

The result will then be saved inside the "Description" field.

Inside an Action Task we proceed with the configuration of an Update Entity action involving the "Description" field.

Specifically, we are going to call the interested SDK function through the "Option Selection" picklist and accessing the "SDK Functions" section (Figure 2)

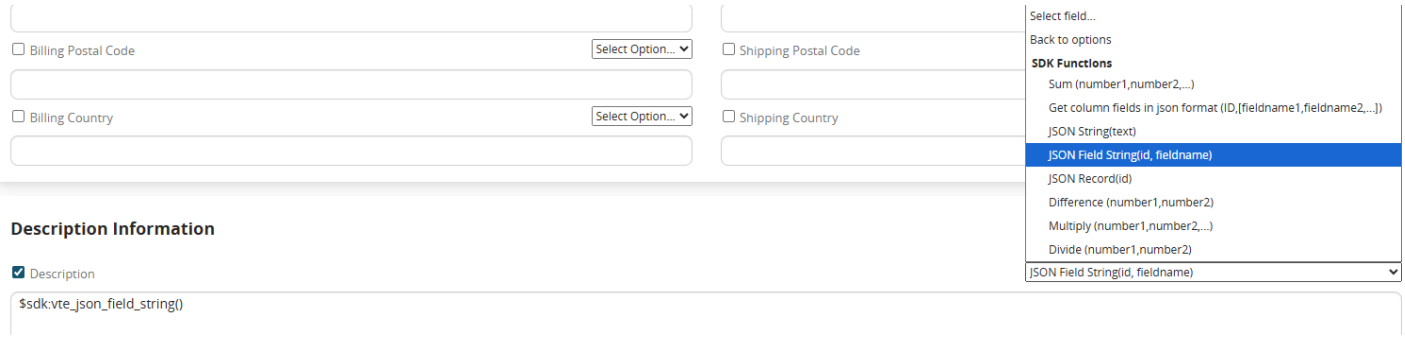


Figure 2 (click on the image for a higher graphic resolution)

Finally we pass the parameters required by the function, all separated by commas.
Specifically, as the first parameter we insert the crmid of the record from which we want to extract the information, so in our case the company id. (Figure 3)

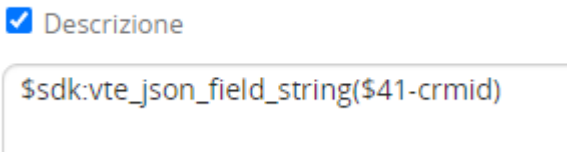


Figure 3

Instead, as a second parameter, we insert the "fieldname" of the field to be involved, or the name with which that field is registered in the Data Base.
In our specific case it will be "bill_city". (Figure 4)

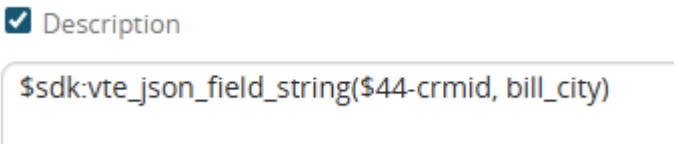


Figure 4

To easily obtain it, simply select the relevant field from the "Option Selection" picklist and delete the reference to the instance involved in the process, i.e. the metaid (Figure 5).

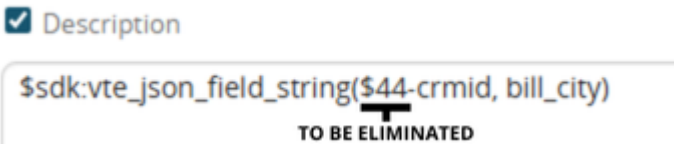

Figure 5

The result generated by the function will be as shown in Figure 6

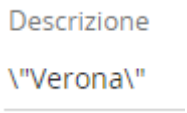


Figure 6

# 12.8 SDK fieldAction: vte_json_record

This SDK function allows you to generate a JSON code containing the labels and values of all the fields of a module instance (record) passed as input.

The only parameter to be passed is the crmid of the record from which you want to extract the information to be inserted into the JSON code.

It is mainly used to easily generate a JSON code in order to perform REST type Web Service calls (always configurable from the process via the dedicated standard action "Call External Web Service").

For further information on configuring API calls from the process, please consult chapter 3.15 of the process manual.

EXAMPLE OF USE

To better understand how it works, below is an example of using the SDK function vte_json_record() to generate a JSON code containing all the fields and values of an instance of the Companies module involved in the process.

Within an Action Task, we proceed with the configuration of an Update entity action involving the "Description" field.
Specifically, we are going to call the interested SDK function through the "Option Selection" picklist and accessing the "SDK Functions" section (Figure 1)

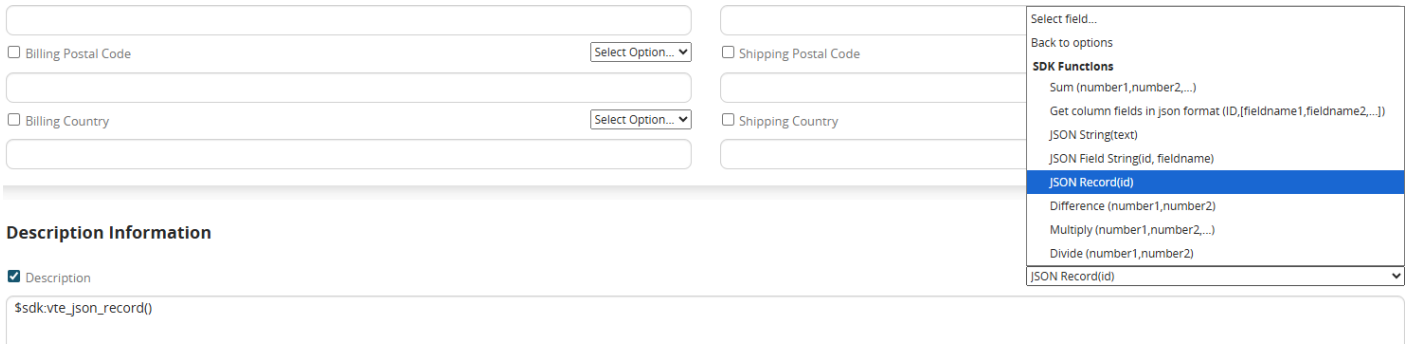


Figure 1 (click on the image for a higher graphic resolution)

Finally we pass as the only parameter the crmid of the record from which we want to extract the information, in this case of the Company (Figure 2)

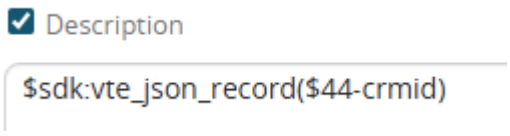


Figure 2

The result generated by the function will be as shown in Figure 3

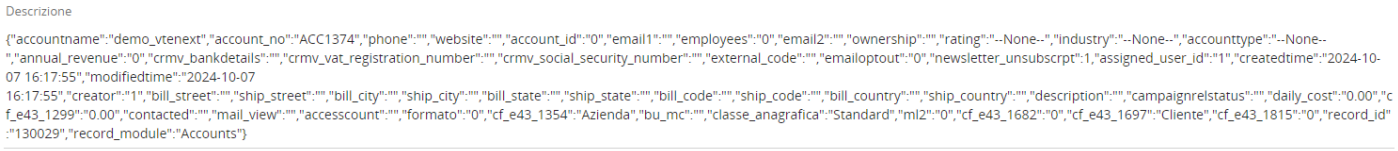


Figure 3 (click on the image for a higher graphic resolution)

# 12.9 SDK fieldAction: Format Date

This SDK function allows you to convert any date passed as input into a specific format.

As the first parameter, you must pass the value of a field of a module or dynamic form containing the date you want to format, instead as the second parameter you must pass the format according to the following syntax:

day / month / year (complete) -> d-m-Y
Example: 25-10-2024

month / day / year (complete) -> m-d-Y
Example: 10-25-2024

year (complete) / month / day -> Y-m-d
Example: 2024-10-25

year (complete) / day / month -> Y-d-m
Example: 2024-25-10

day / month / year (partial) -> d-m-y
Example: 25-10-24

month / day / year(partial) -> m-d-y
Example: 10-25-24

year(partial) / month / day -> y-m-d
Example: 24-10-25

year(partial) / day / month -> y-d-m
Example: 24-25-10

day / month / year(complete)  hours / minutes / seconds -> d-m-Y H:i:s
Example: 25-10-2024 16:32:20

month / day / year(complete)  hours / minutes / seconds -> m-d-Y H:i:s
Example: 10-25-2024 16:32:20

year(complete) / month / day  hours / minutes / seconds -> Y-m-d H:i:s
Example: 2024-10-25 16:32:20

year(full) / day / month  hours / minutes / seconds -> Y-d-m H:i:s
Example: 2024-25-10 16:32:20

hours / minutes / seconds -> H:i:s
Example: 16:32:20

day only -> d
Example: 25

month only -> m
Example: 10

year(full) -> Y
Example: 2024

year(partial) -> y
Example: 24

hours only -> H
Example: 16

minutes only -> i
Example: 32

seconds only -> s
Example: 20

N.B: a partial result will not be manageable within date or datetime fields, this is because they only accept values with at least one day, month and year.

EXAMPLE OF USE

To better understand how it works, below is an example of using the SDK function formatDate() to format the value of the "Creation Period" field of an instance of the Companies module and save it within a field of a process helper.

Within the dynamic form we proceed with the creation of a date field called "Creation Date" in which the result of the function will be saved.
Then we call the interested SDK function through the "Option Selection" picklist and access the "Date Functions" section (Figure 1)

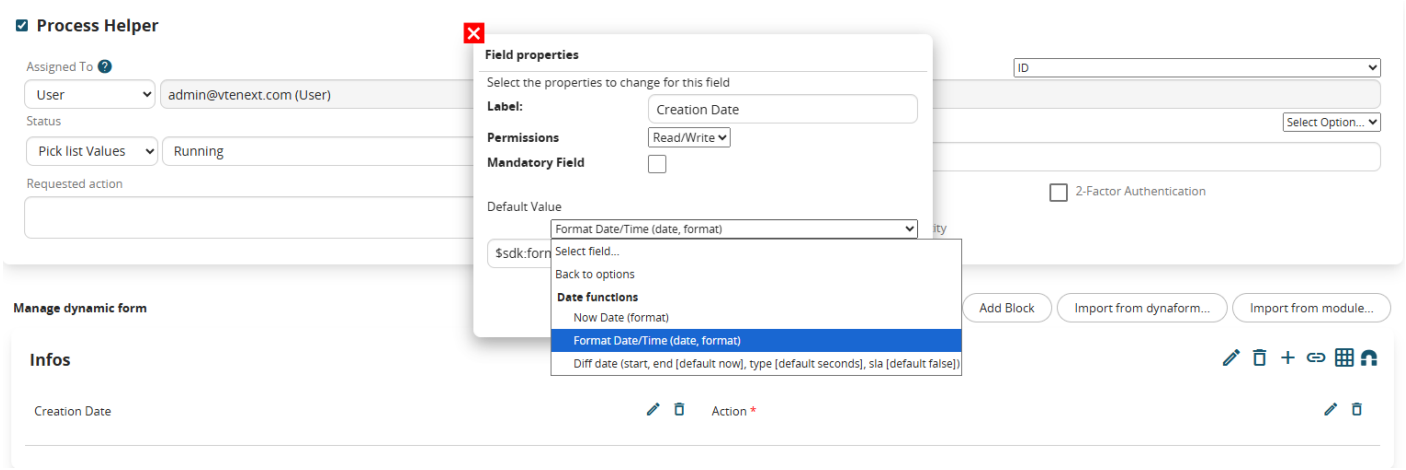


Figure 1

Finally we pass the parameters required by the function, all separated by commas.
Specifically, as the first parameter we insert the value of the "Creation Period" field (Figure 2)

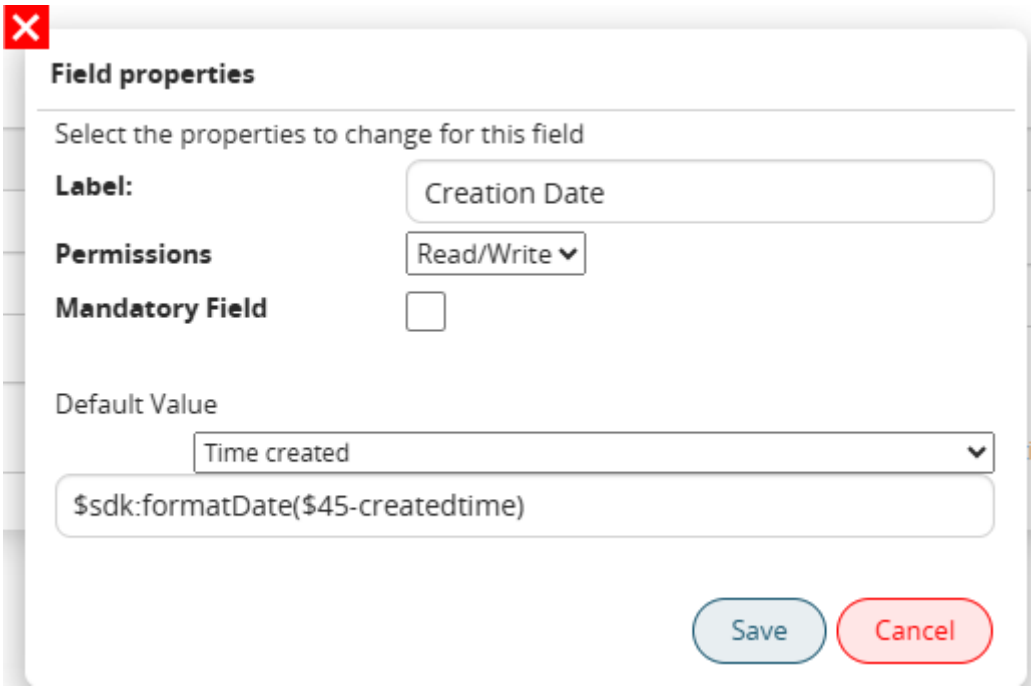


Figure 2

Instead, as a second parameter, we pass the format with which the function must return the date, so in our case d-m-Y (Figure 3)

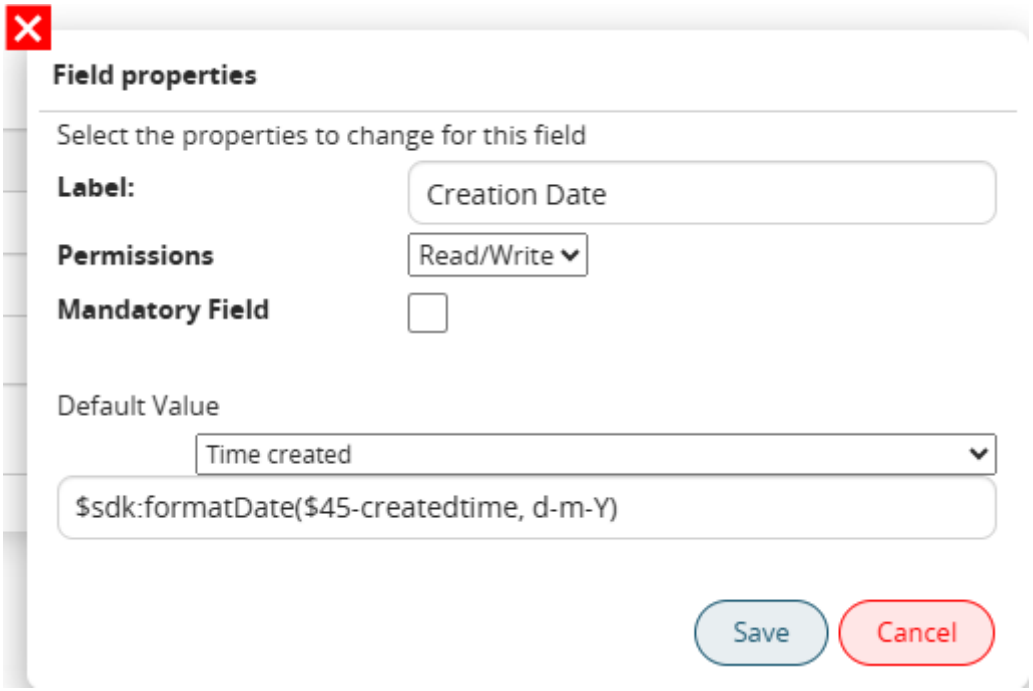


Figure 3

The result will be as shown in figure 4


Figure 4

# 12.10 SDK fieldAction: Now Date

This SDK function allows you to obtain and save today's date within any text, text area, date and datetime field.

By default (so without passing any parameter), the result will always be returned in the "d-m-Y" format, to obtain the date in a different format, the format will be passed as the only parameter of the function, to view the syntax of each available case in detail, consult the chapter ... of the process manual.

EXAMPLE OF USE

To better understand how it works, below is an example of using the SDK function date_now() to save today's date within a field of a process helper.

Inside the dynamic form we proceed with the creation of a date field called "Today's date" in which the result of the function will be saved.

Then we call the interested SDK function through the "Option Selection" picklist and access the "Date functions" section (Figure 1)

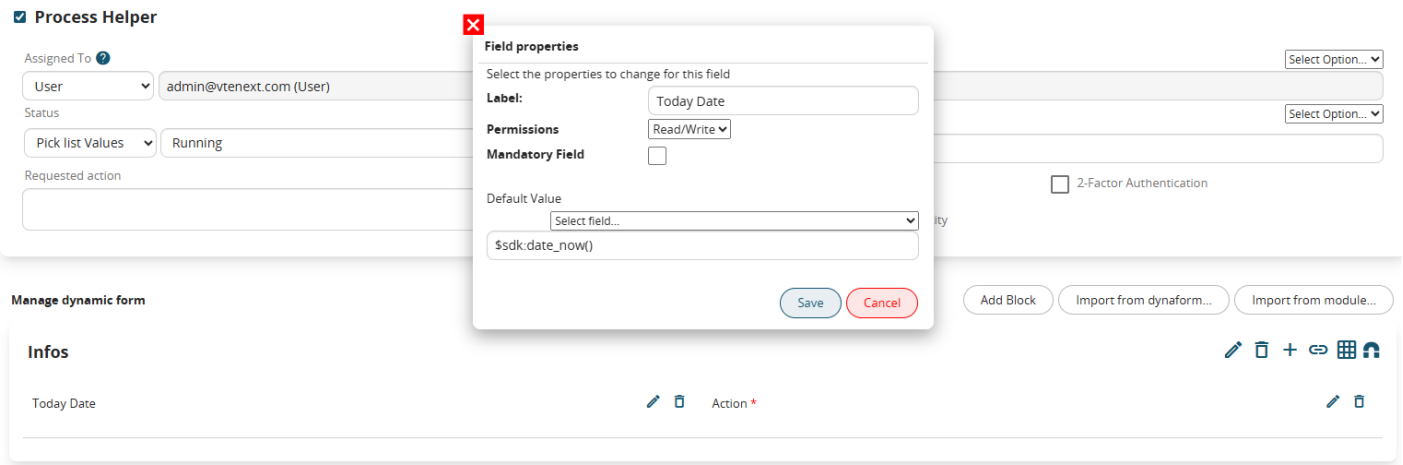


Figure 1

Finally, if necessary, we pass the format with which the date must be returned (Figure 2)

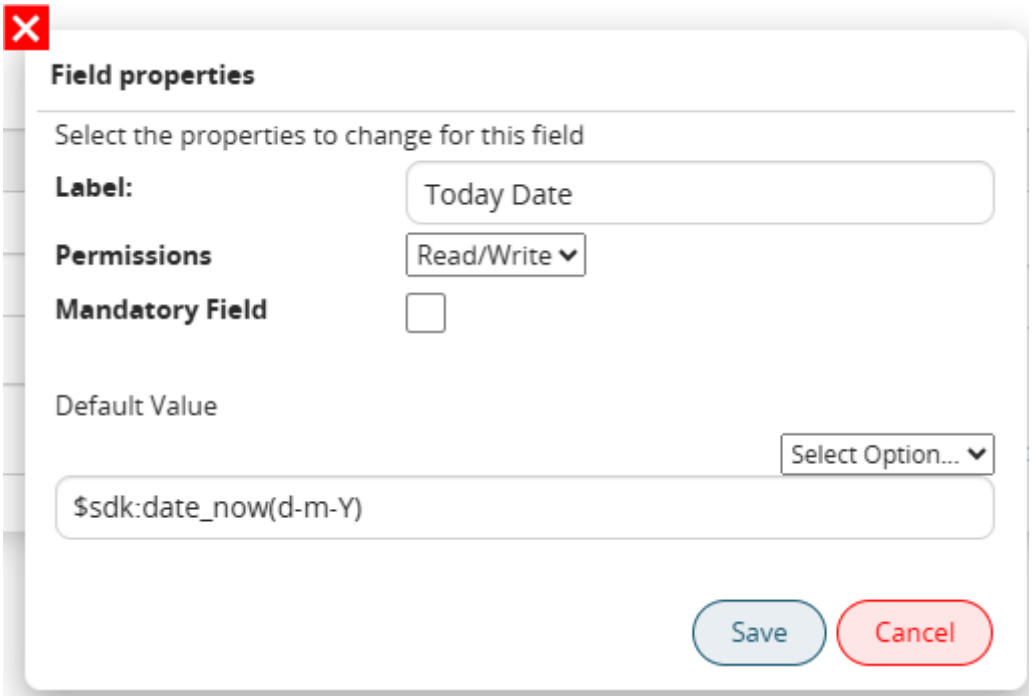


Figure 2

The result will be as shown in figure 3


Figure 3

# 12.11 SDK fieldAction: Diff Date

This SDK function allows you to perform the difference between two dates passed as input parameters.

As the first two parameters, separated by a comma, you will need to pass the values of the two date type fields to be involved in the difference, but keep in mind that the function performs the following calculation:

parameter2 - parameter1

The second date parameter is not mandatory, so if you only pass a date value (parameter1), the calculation will be performed automatically taking today's date-time as the second parameter (parameter2).

By default (so without passing a third parameter), the function will always return the result of the calculation in seconds.
Instead, to obtain the result in days, you will need to pass the fixed string "days".

ATTENTION! by default the function will always return only positive results, to enable the return of negative results you will need to apply a change to the php code.

EXAMPLE

To better understand how it works, here is an example of using the SDK function diffDate() to calculate the difference between the values of two fields called "Start Date" and "End Date" and save the result (in days) in the "Time Range" field (Figure 1)


Figure 1

Within an Action Task we proceed with the configuration of an Update Entity action involving the "Time Range" field.
Specifically, we are going to call the interested SDK function through the "Option Selection" picklist and accessing the "Date Functions" section (Figure 2)

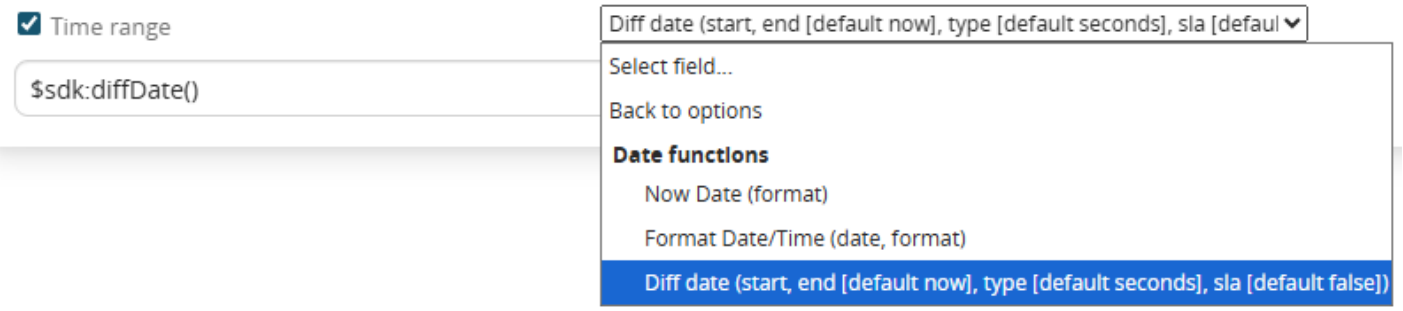


Figure 2 (click on the image for a higher graphic resolution)

N.B: in the label of the selectable function in the picklist "Select option.." the possibility of passing an additional parameter called "sla" is indicated, which by default is set to false, in fact it is only a typo, therefore it is an unmanageable parameter.

As the first and second parameters (separated by commas) we pass the values of the "Start Date" and "End Date" fields in the order just mentioned (Figure 3)


Figure 3

Finally as a third parameter we pass the fixed string "days" (Figure 4)


Figure 4

The result will be as shown in figure 5

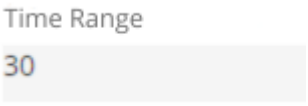


Figure 5

# 12.12 SDK fieldAction: Set lead converted

This SDK function allows the system to perform a series of automations that allow the correct and complete conversion of the lead.

In fact, this function can be used exclusively within the standard lead conversion process available in the "Process manager" section with the wording "Lead conversion". (Figure 1)

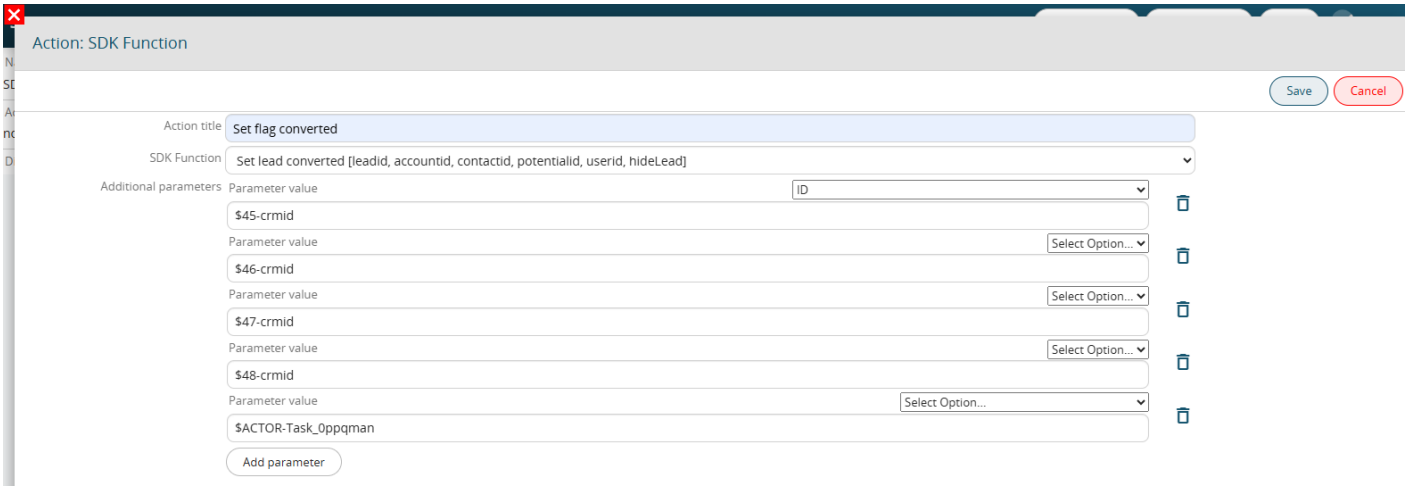


Figure 1 (click on the image for a larger graphic resolution)

The following operations are performed by the function:

1) References to the lead in the vte_email_directory table are deleted

2) If the lead is set among the favorites of some user, the function replaces it with the company or contact involved in the process depending on the value selected in the "Transfer elements linked to" field.
N.B: if the "empty" value is selected, the lead will not be replaced.

3) If the lead is present among the static newsletters of some campaign, the function replaces it with the company or contact involved in the process depending on the value selected in the "Transfer elements linked to" field.
N.B: if the "empty" value is selected, the lead will not be replaced.

4) The value of the "converted" column of the vte_leaddetails table is set to "1".
This allows you to completely hide the visibility of the lead on the interface side (as if it had been deleted).

5) The vte_leadconvrel table is populated to track the Conversion Date, lead crmid, company crmid, contact crmid and opportunity crmid.

The first parameter is the crmid of the lead being converted.

The second parameter is the crmid of the company (created or existing).

The third parameter is the crmid of the contact (if created).

The fourth parameter is the crmid of the opportunity (if created).

The fifth parameter is the id of the user who triggered the process.

To prevent the system from executing points 4) and 5) and therefore the lead from being hidden on the interface side, a sixth parameter must be added in which to insert the static value "false". (Figure 2)

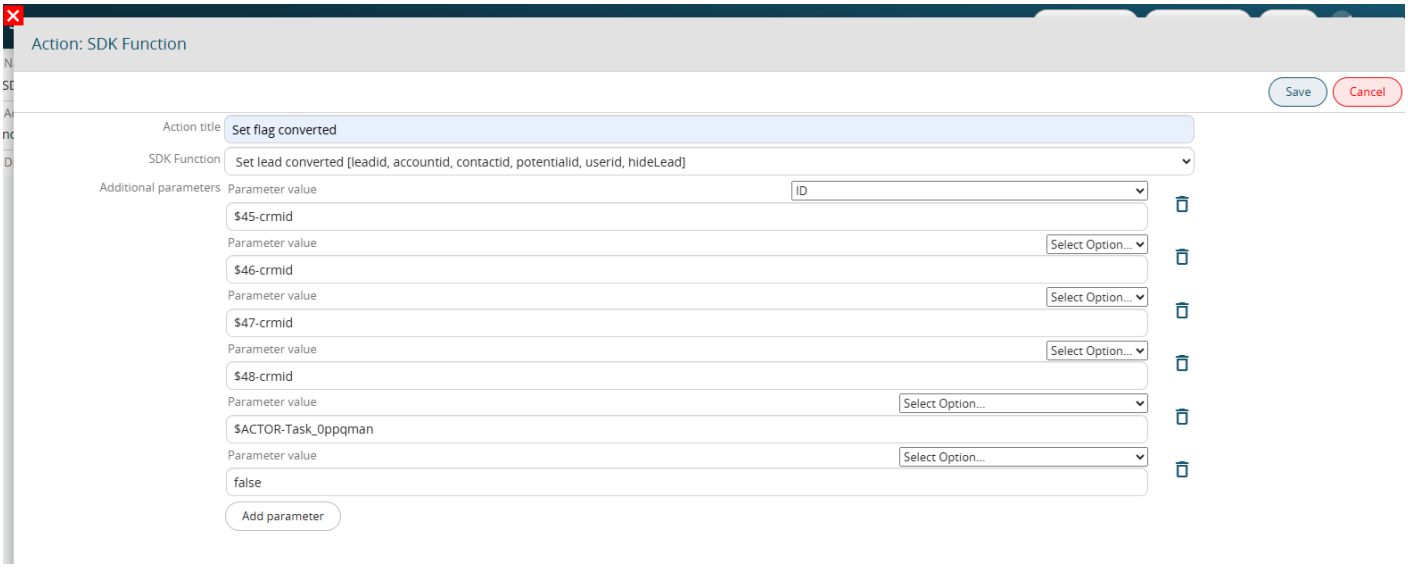

Figure 2 (click on the image for a higher graphic resolution)

# Nuova pagina12.13 SDK fieldAction: vte_formula

This SDK function allows you to replicate the functionality of formula fields.

Consequently, you can use all algebraic operators within the formulas: plus (+), minus (-), divided by (/), and multiplied by (*).

EXAMPLE 1

To better understand how it works, an example of using the `vte_formula()` SDK function is provided below. This example calculates the content of the "Delta Budget Invoiced Open Order Value" field by applying the following formula:
"Budget" - ( "Current Year Invoiced" + "Open Order Value" ) (Figure 1)

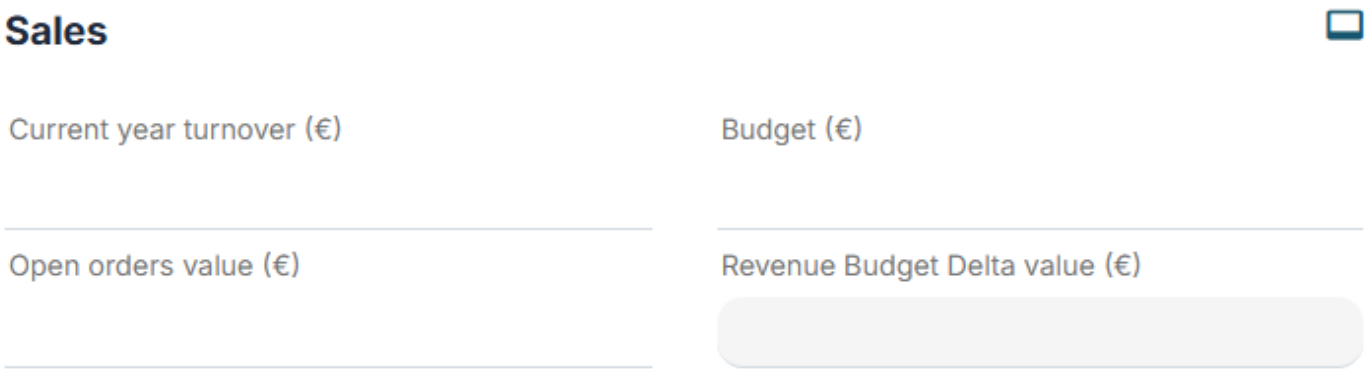


Figure 1

Within an Action Task, we proceed to configure an "Update Entity" action involving the field "Delta Budget Invoiced Open Order Value."
Specifically, we call the relevant SDK function via the "Option Selection" picklist by accessing the "Date Functions" section (Figure 2).

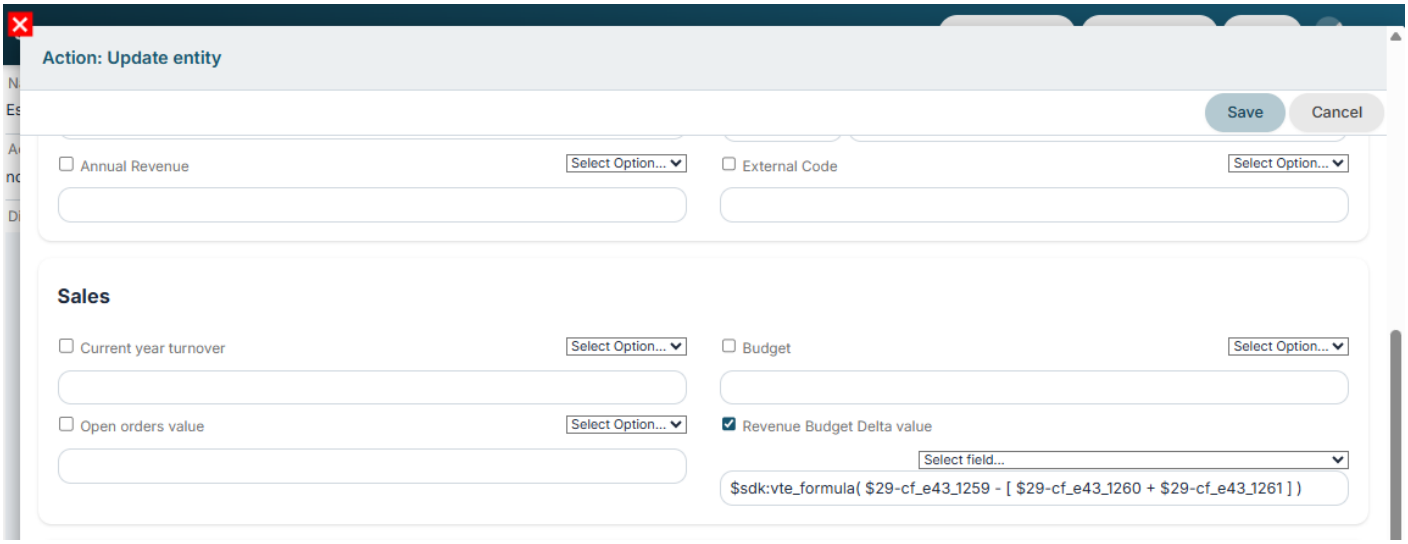


Figure 2

The result will be like the one shown in Figure 3.


Figure 3

It is also possible to configure If/Else controls using the following structure:

**if** condition **then** true_case **else** false_case **end** (in this case, >, <, and == operators can be used; the != operator is not permitted).

EXAMPLE 2

To better understand how it works, an example is provided below showing the use of the SDK function `vte_formula()` to insert the value "Mr." into the "Salutation Formula" field if the "Gender" field is set to "Male," or the value "Ms." otherwise. (Figure 4)

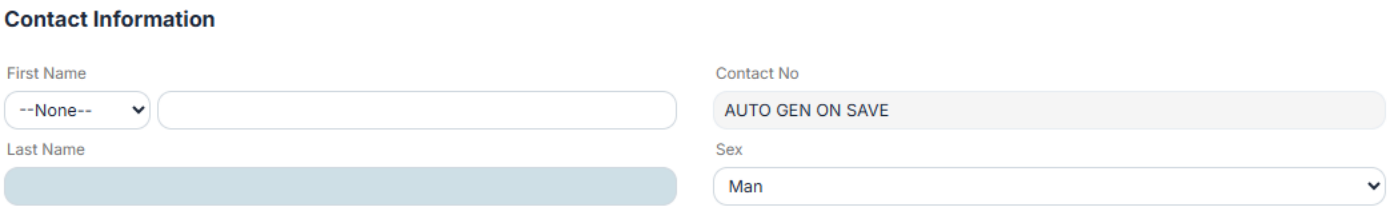


Figure 4

Within an Action Task, we proceed to configure an "Update Entity" action involving the "Salutation Formula" field.
Specifically, we select the relevant SDK function via the "Option Selection" picklist by accessing the "Date Functions" section (Figure 5).

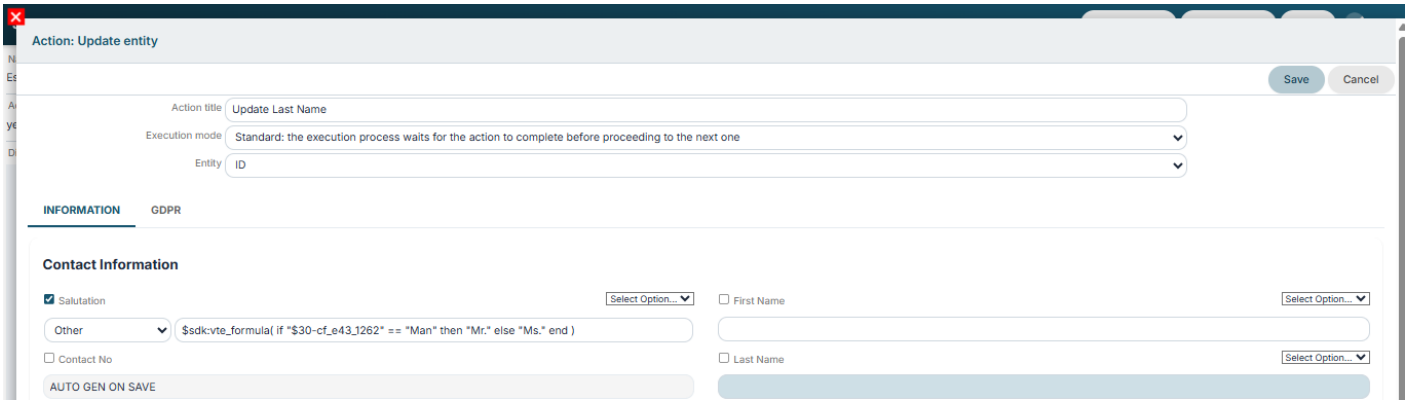


Figure 5

Selecting Gender = Male when adding a contact will result in the display shown in Figure 6.


Figure 6

It is also possible to perform operations using date-type fields; specifically, there are two operators:

**time_diffdays**: returns the difference in days between two dates

**time_diff**: returns the difference in seconds between two dates.

**Note**: If only a single parameter is entered for the time_diff and time_diffdays functions, the value returned in the formula field is the difference between the current date and the date provided as the parameter.