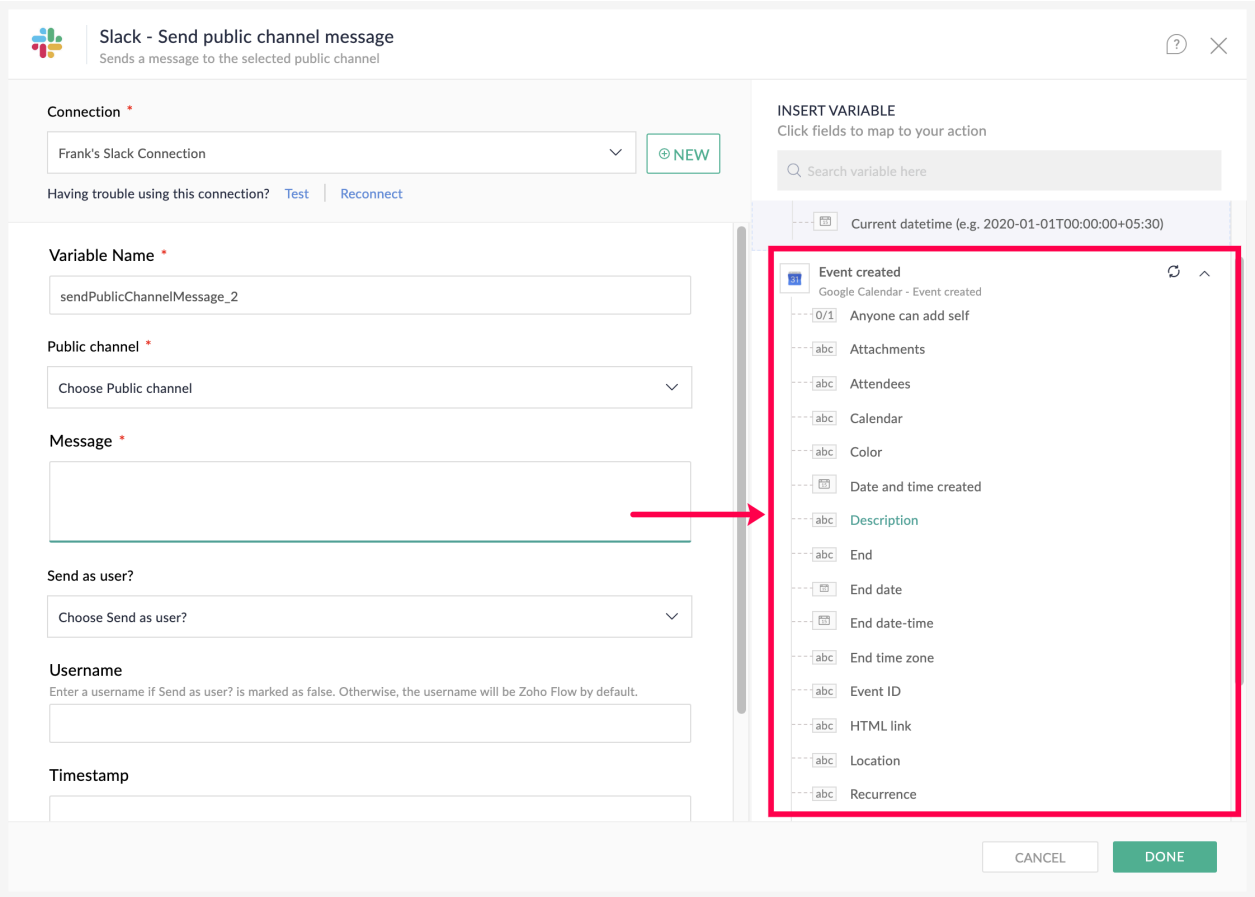
Zoho Corporation

Data mapping

Introduction

Data mapping is using the variables of a trigger or action as input for the next steps of the flow. It lets you decide how data should be organized when the flow is processing. With mapped data you can create a consistent flow of information through your apps.

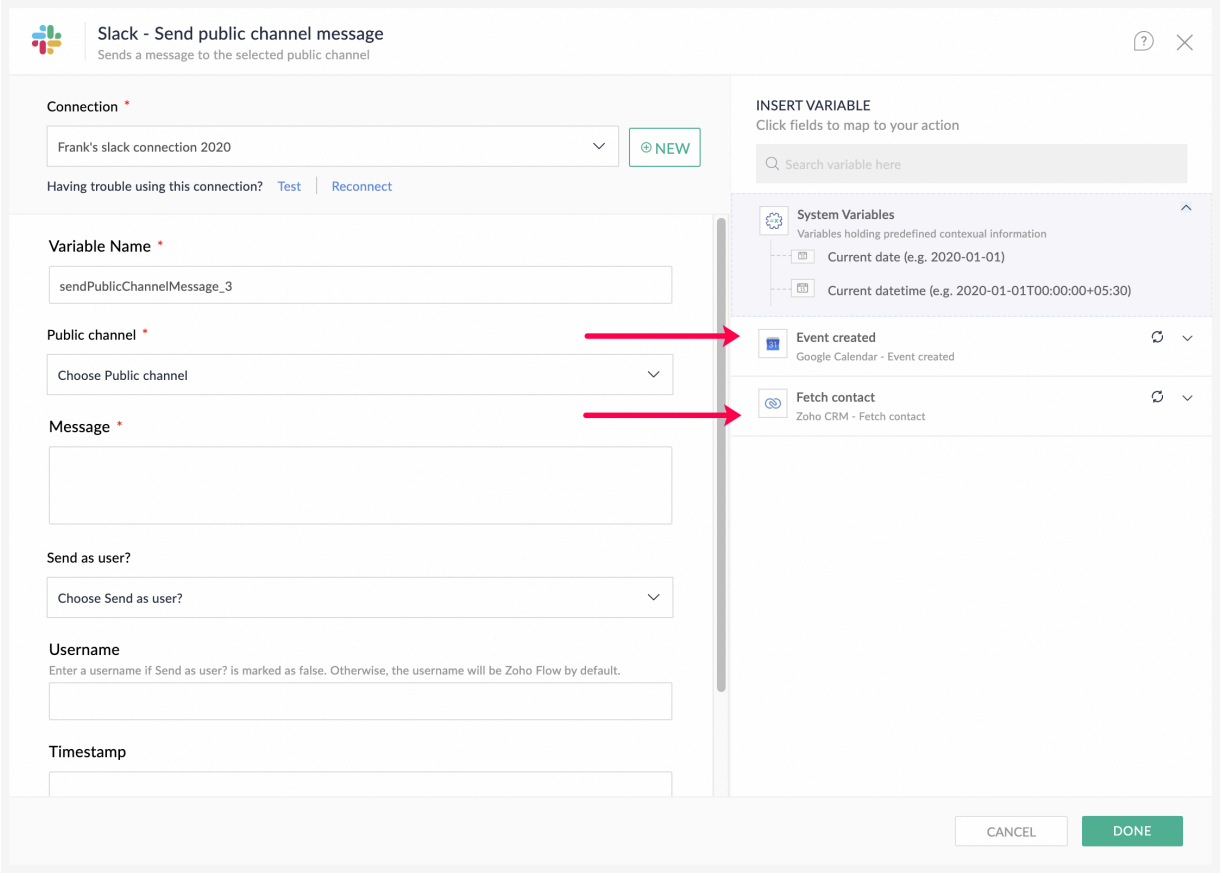
Variables are fields provided by the trigger or action as output after an execution. The value of these variables may change with each execution.



For example, every time an event is created in Google Calendar, you want to send a message to a Slack channel to notify your team. You can have a standard message every time: 'A new event has been created'. This does not provide your team with any details of the event and they have to keep switching between Slack and Google Calendar. Instead, you can map variables to the Slack message and provide a helpful text such as 'Content Strategy Meeting at 10 am'.

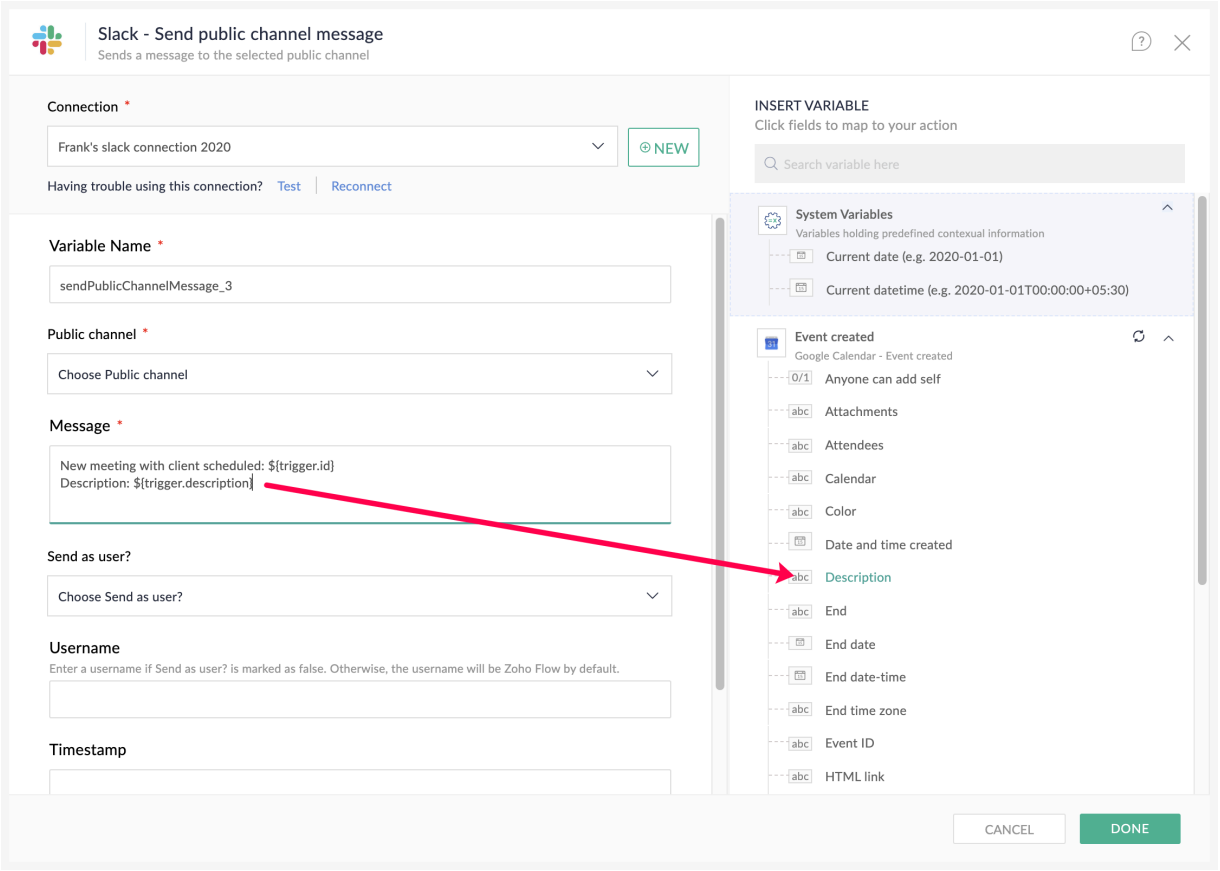
How to map

1. When you configure an action, all the available variables will be listed on the right of the configuration window under **Insert Variable**. You can view variables from all previous steps. They will be classified by the trigger or action they came from.



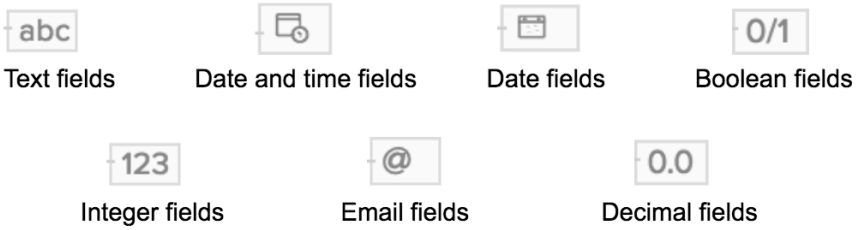
2. Click the trigger or action to view all variables it provides.

3. **Click the field in the action** that you are configuring. **Once your cursor is there**, click the variable you want to map.



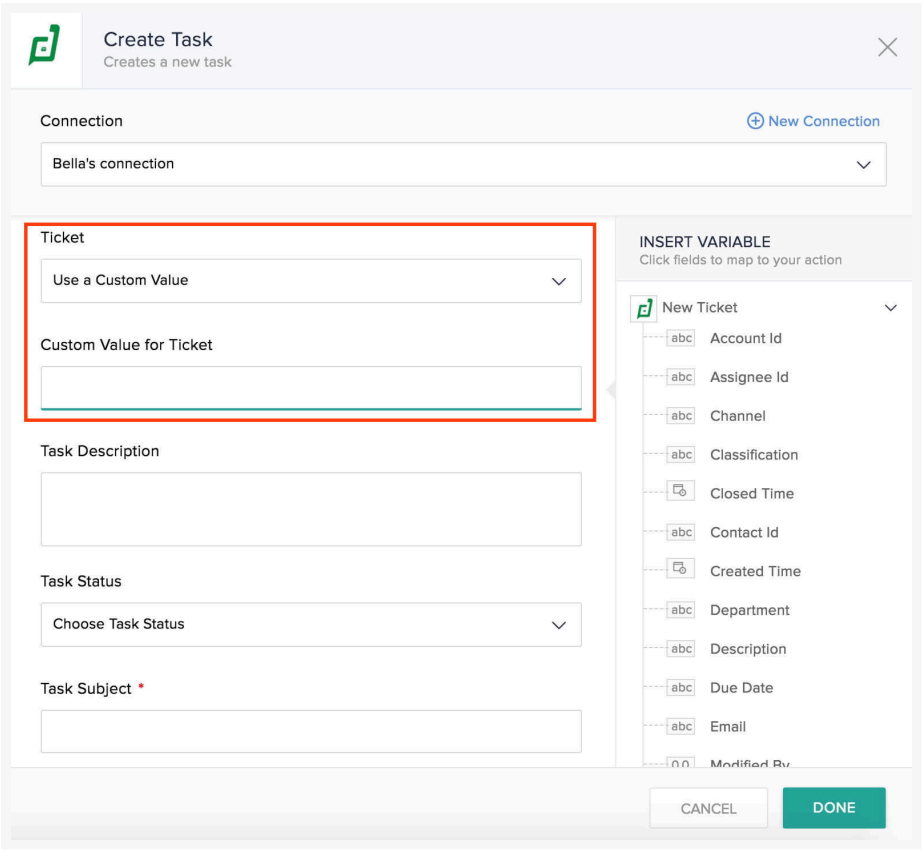
4. You can map as many variables as you need.

Note: Make sure that you map the variables to the right fields. The icons to the left of the variable indicate what type of data each variable provides.



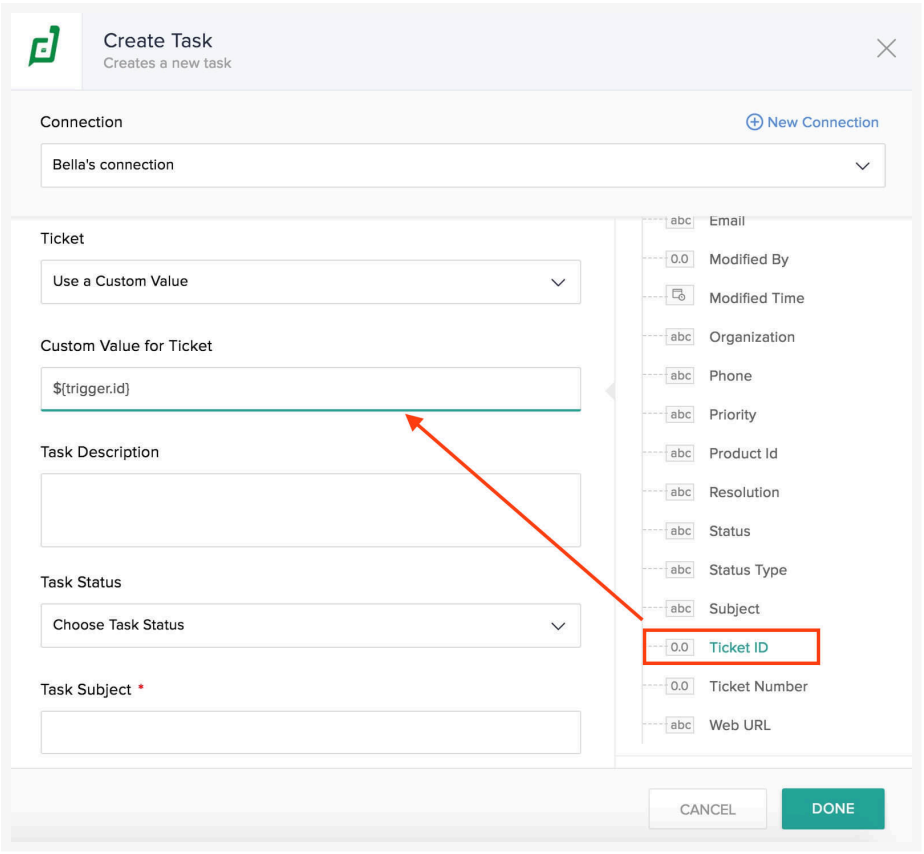
Mapping custom values

When configuring a dropdown in an action, you can pick from the available values or use a custom value.



For example, every time a ticket is created, you want to create a task in it. You can't define a particular ticket to create a task in while setting up the flow. If you do so, a new task will be created in the selected ticket every time instead of in the new ticket you want to create the task in.

In cases like these, you can use a custom value. Map the ticket ID from the previous step to this field. When a ticket is created, the ticket ID will be fetched and a task will be created for that ticket.



Note that not all dropdowns can be configured with a custom value. If the other fields in the configuration change depending on the value selected in the dropdown, then you can't use a custom value. For example, consider the Form field in Zoho Forms. The fields in the form in Zoho Forms appear as configuration fields in Zoho Flow. These change with each form, so they won't have the option of using a custom value.

Understanding data and syntax

Zoho Flow sends and receives data between apps as payloads. The payload can be sent or received in various formats—the most common being **JSON**. This format displays data, using **,** **{}** and **[]** to separate and group the data together. You can use **Test and Debug** to view the JSON payload received for the triggers or actions in your flows. JSON defines only two data structures: **objects** and **arrays**.

Note: Variables and field names must be formed of English characters and numbers. Other languages are not supported.

Objects

When you see multiple fields appear within one set of **{ }** it is an **object**. JSON objects are written in **key/value pairs**. For example:

```
contact: {
"name" : "Frank",
"email" : "frank@zylker.com"
}
```

Here, **contact** is the object, **name** and **email** are the keys, and **Frank** and **frank@zylker.com** are their respective values. If you want to map a key to a field, you can use the format:

```
${source.object.key}
```

where:
source - the variable name assigned to your trigger or action from where you receive the payload containing the required information
object - the name of the object
key - the name of the key that holds the value that you need to map to your field

Lists or Arrays

What you see within [ ] is called an **array**, otherwise called a **list**. There are simple arrays of strings that contain values that are comma separated. There are also arrays of objects that contain multiple objects with one or more **key-value** pairs.

For example:

```
"cars" : [
{"name" : "BMW" , "model" : "X3"},
{"name" : "Audi" , "model" : "Q5"},
{"name" : "Ford" , "model" : "Mustang"}
]
```

Mapping list or array elements

If your trigger or action returns a payload with a variable that is a list or an array, you can use the formula editor to specifically extract a particular array element instead of the whole thing. You just need to know the nature of the array and how the elements are distributed within it. Use the **Test and Debug** feature in your flow builder to view the JSON payload received for the triggers or actions in your flows.

Arrays are **zero-indexed**. This means the array elements are numbered starting with **0**. [0] is the first element, [1] is the second element, and so on. As a general rule of thumb, to map an element from an array variable, enter the formula in the following format:

```
${source.array[n].key}
```

where,
source - the variable name assigned to your trigger or action from where you receive the payload containing the required array
array - the name of the array that contains the required value
n - the index of the array element
key - the name of the key that holds the value that you need to map to your field

Suppose, in the example array given below (say, from the trigger output), you want to map the model of a car ("Audi") to a field in an action.

```
"cars" : [
{"name" : "BMW" , "model" : "X3"},
{"name" : "Audi" , "model" : "Q5"},
{"name" : "Ford" , "model" : "Mustang"}
]
```

Here, you need to map the value of the field, "model" of the 2nd item in the array. Therefore, the key is "model" and n is 1. So you should enter the formula as:
${trigger.cars[1].model}

A few use cases

Zoho Books - Extracting a line item entry

A common use case of extracting values from an array is when you want to map a field with an entry from line items in Zoho Books trigger response data. For example, if you want to extract the value of *Description* for the second item under line items, while mapping, modify the formula as:

```
${trigger.line_items[1].description}
```

Calendly - Pulling individual responses from Questions and Answers

Suppose you have a flow with the Calendly trigger, *Event scheduled by invitee*. You might have configured custom questions for which the answers are stored in the array, *Questions and answers*. If you want to pull the answer of the first question, say *Twitter ID* from *Questions and answers*, map the value using the key:

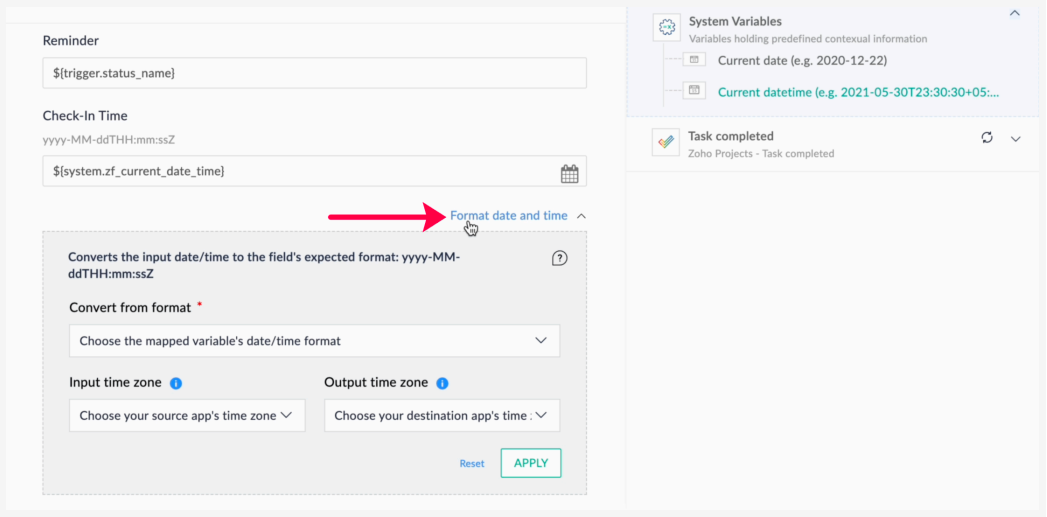
1. `${trigger.questions_and_answers[0].answer}`

Mapping and formatting date and time values

When working with different apps that handle date and time values in different formats, it's important to ensure that the data is accurately interpreted and processed by the receiving app. You can configure the actions in your flows to format the date/time values automatically to match the expected formats of the receiving apps.

To format date/time in Zoho Flow:

1. Map your variable to the date/time field.
2. Click **Format date and time** under the field.



3. Configure the input date and time format to match the format used in your source app. If you're unsure, consult your app's API documentation.
4. Click **Apply**.

If your flow involves teams or apps in different time zones, you can also configure the time zones to ensure that the time difference is also accounted for.

By taking these steps, your date/time values will be automatically converted to the correct format whenever your flow is executed, ensuring smooth communication between your apps.

Note:

- If you do not configure date formatting, then the value coming from the source will be passed as it is to the destination app.
- If you do not configure the time zones, the default time zone of your Zoho Flow organization will be used.

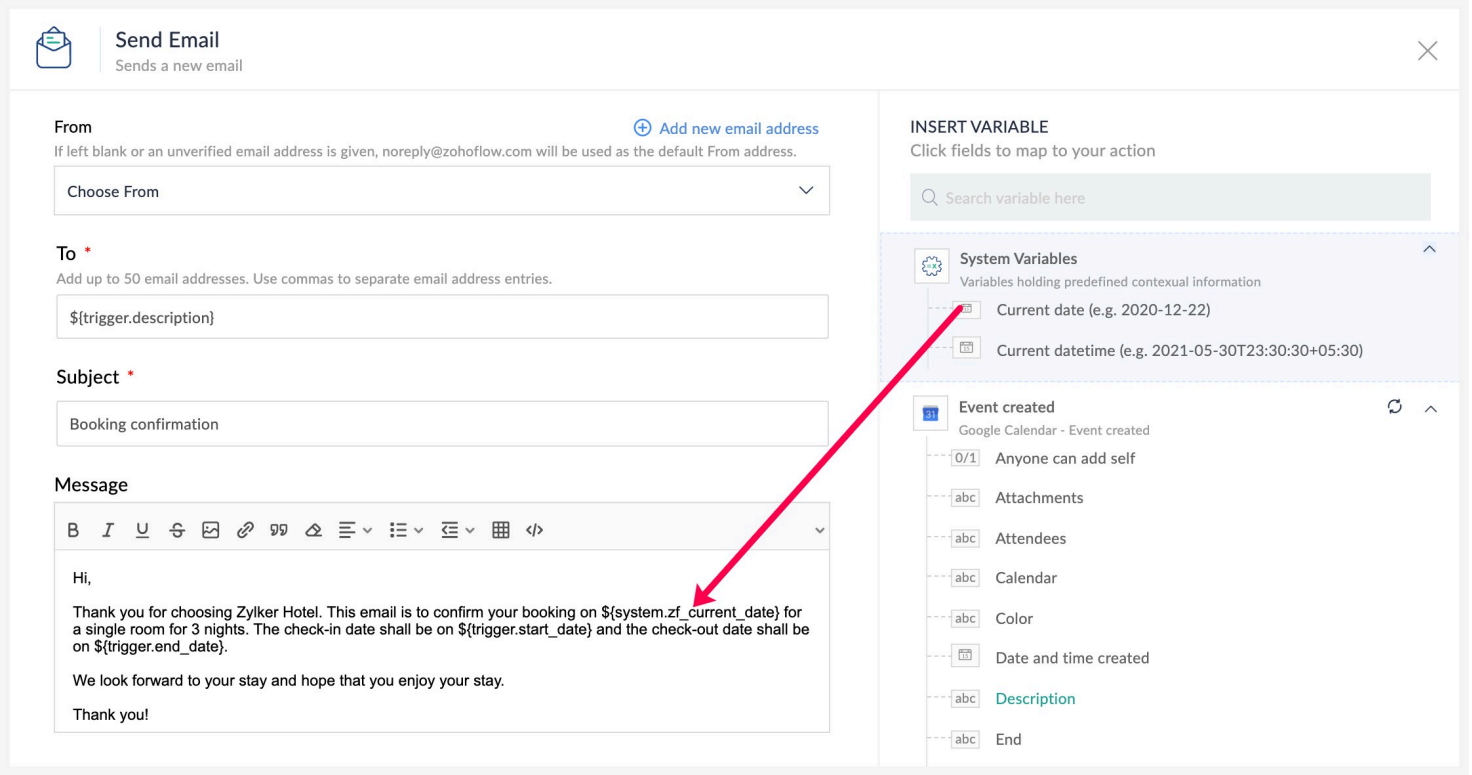
Mapping entire trigger (or action) variables

If you need to pass on all the variables received from the trigger (or an action) of your flow to a custom function or a key-value field, you can use the following format:
${source}
where, source is the variable name assigned to the trigger or the action.
For example, to refer to all the variables collected by your flow's trigger, use ${trigger}.

Note: The variable name assigned to a trigger is 'trigger' by default. Make sure you use the same name in the snippet.
Ensure that the field you are mapping the variables is of datatype Map (Key-value).

Mapping system variables

You can map contextual information like current date and current datetime to a field using system variables.



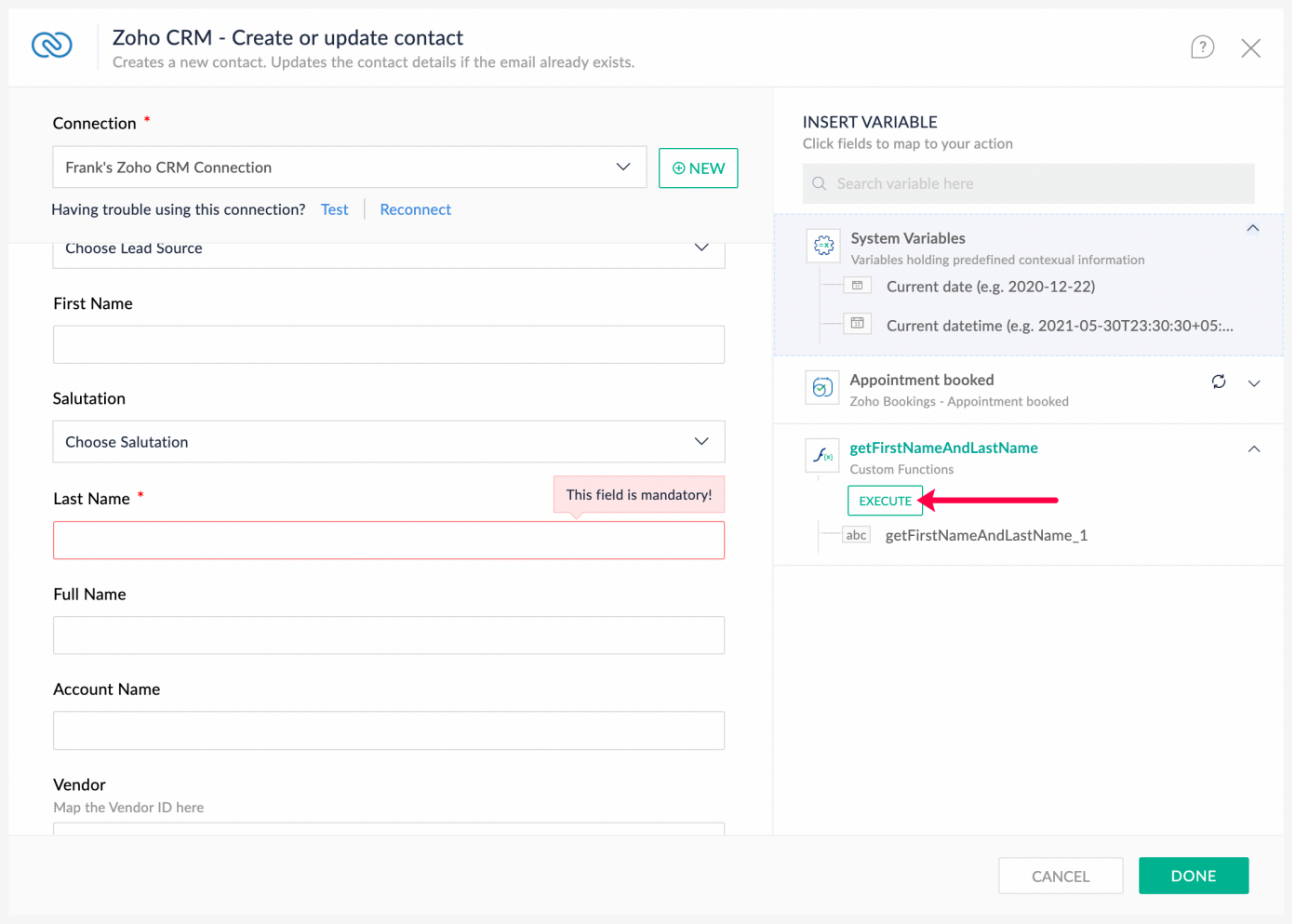
- **Current date**: The date when the task is executed will be passed on to the field it is mapped to. (Format: **yyyy-MM-dd**; e.g. 2021-11-29)
- **Current datetime**: The date and time when the task is executed will be passed on to the field it is mapped to. (Format: **yyyy-MM-ddThh:mm:ssZ**; e.g. 2021-12-19T23:30:30+05:30)

Mapping output keys of map functions

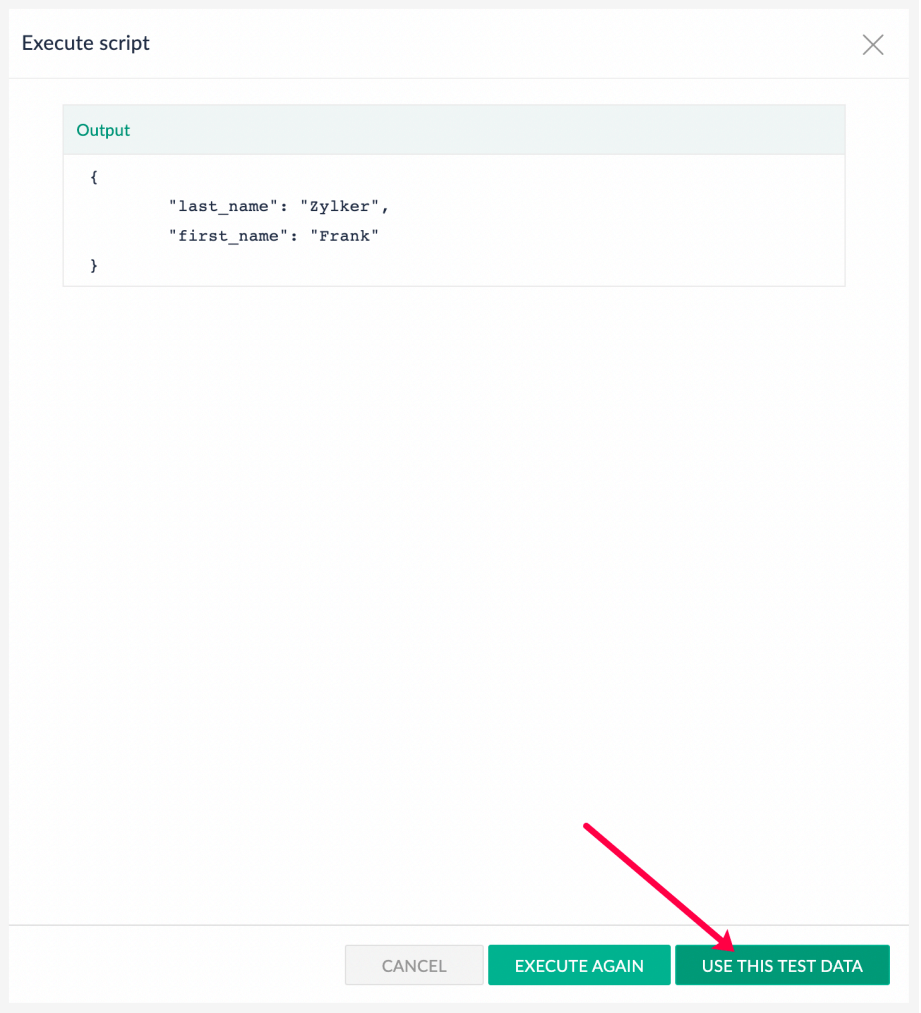
If you are using a custom function that returns key-value pairs (map datatype), your output variable may be an object with multiple key-value pairs. To map individual keys separately, you can use the **Execute** feature in the Insert Variable section.

For example, if you've created a custom function to parse a person's name into their first name and last name, the two keys (say, 'First name' and 'Last name') can be made available as individual variables in the Insert Variables section. To do so:

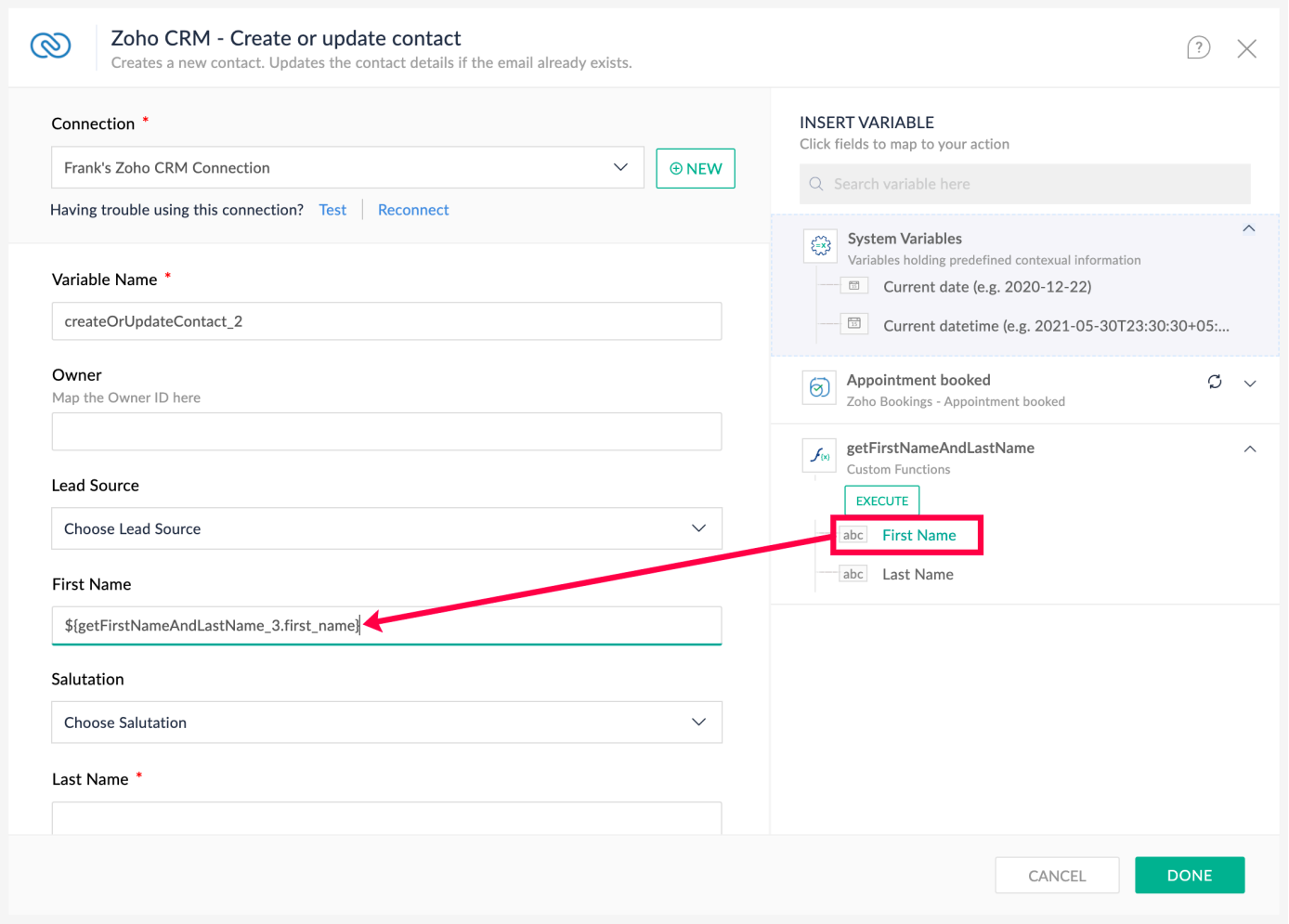
1. While configuring an action following your custom function, navigate to the required function under the **Insert Variables** section.



2. Click **Execute**.
3. Provide a sample input to test the function, then click **Execute**.
4. Once you've verified the output, click **Use Test Data** to make the keys available as variables under the custom function.



5. Map the required variables to the fields in any actions that follows.



You can also execute your function while configuring or editing the custom function, and use the test data to retrieve the output keys.

Common problems with mapping

Mapping variables to mandatory fields

When you map variables to mandatory fields, ensure that the variables always return data. If the mandatory fields do not contain data while the flow is processing, the flow will fail.

Modifying the trigger or action after the flow is set up

If you change the trigger or action after the flow is set up, modify the mapped fields accordingly.

Incorrect mapping

Sometimes, a variable with the same name will be provided by multiple steps in the flow. Common examples of this are name or date. Though the variable names are the same, they may provide different values. Make sure that you map the variable from the right step.

Using "$" as a text

Use $$ to include a $ symbol as a part of text. The flow will assume it to be a part of variable if only one $ symbol is used.

E.g: $$${trigger.price} will display the output with a $ symbol.