Zoho Corporation

Data Mapping

what is data mapping?

Data mapping is the process of defining how data will be used and transformed within your RPA flows. It allows you to connect the outputs of one action or trigger to the inputs of another, ensuring a smooth and efficient flow of information throughout your automation process. This process is crucial for managing diverse data types commonly found in RPA workflows.

Why is data mapping important?

Data mapping plays a crucial role in several key aspects of RPA:

Dynamic Data Processing: Utilize the output data from one step to dynamically process information based on previous actions. For example, extracted timestamps or data can be used in subsequent actions.
Example: Extract departure times from emails and dynamically update travel schedules in calendars and to-do lists.

Flexibility: Connect diverse data formats and structures, enabling automation across various applications and systems.
Example: Extract invoice information from Excel files, web applications, and cloud app integrations, then transform the data into a unified format for reporting and data processing.

Streamlined Workflows: Eliminate manual data manipulation, leading to faster and more efficient automation processes.
Example: Data from new hire documents automatically populates employee profiles and system access, eliminating manual data entry.

Enhanced Accuracy: Clear data connections minimize errors and inconsistencies, ensuring data integrity throughout your flows.
Example: Mapping product codes to corresponding descriptions ensures accurate pricing and inventory management.

Understanding data and syntax

Zoho RPA talks to other applications using various data formats, most commonly JSON—a neat and organized way to package and share information. This format uses curly braces ({}) and brackets ([]) to organize information. Here's how it works:

Objects: These entities hold multiple pieces of information, such as names and emails, organized into key-value pairs. Think of them as boxes with labeled compartments, where each compartment holds specific information. The labels on those compartments are called "keys," and the items inside are called "values."
Example:

```
invoice: {
"company": "Zylker",
"amount": "$3000"
}
```

Arrays or lists: These are ordered collections of data, holding multiple objects in a specific sequence. Each item is assigned a position, starting at 0 (the first item). Visualize them as shopping carts with multiple items.
Example:

```
contacts": [
{ "name": "Angela", "email": "angela@zylker.com"},
{ "name": "Emma", "model": "emma@zylker.com"},
]
```

Tip:

- Curly braces ({}) enclose objects.
- Square brackets ([]) enclose arrays.

How do we extract data?

Extract data from an object:

Use **${source.object.key}** to access a specific value within an object.

where,
source: name for the trigger or action containing the data.
object: name of the object you want to extract data from.

key: name of the specific key holding the desired value.

Example :
Action output: getinvoicedetails

```
{
 "invoice": {
 "company": "Zylker",
 "amount": "$3000"
 }
}
```

Formula to extract the company name: **${getinvoicedetails.invoice.company}**
This will output "**Zylker**" in the target field.

Extract data from a list or array:

Use **${source.list[index].key}** to access a specific element and its corresponding key within a list or array.

where,
source: variable name for the trigger or action containing the list.
list: name of the list you want to extract data from.
index: position of the element within the list (starts at 0).
key: name of the specific key holding the desired value.

Example:

```
// List output (e.g., from a trigger):
"contacts":[
 {"name": "Angela", "email": "angela@zylker.com" },
 {"name": "Richard", "email": "richard@zylker.com" }
]
```

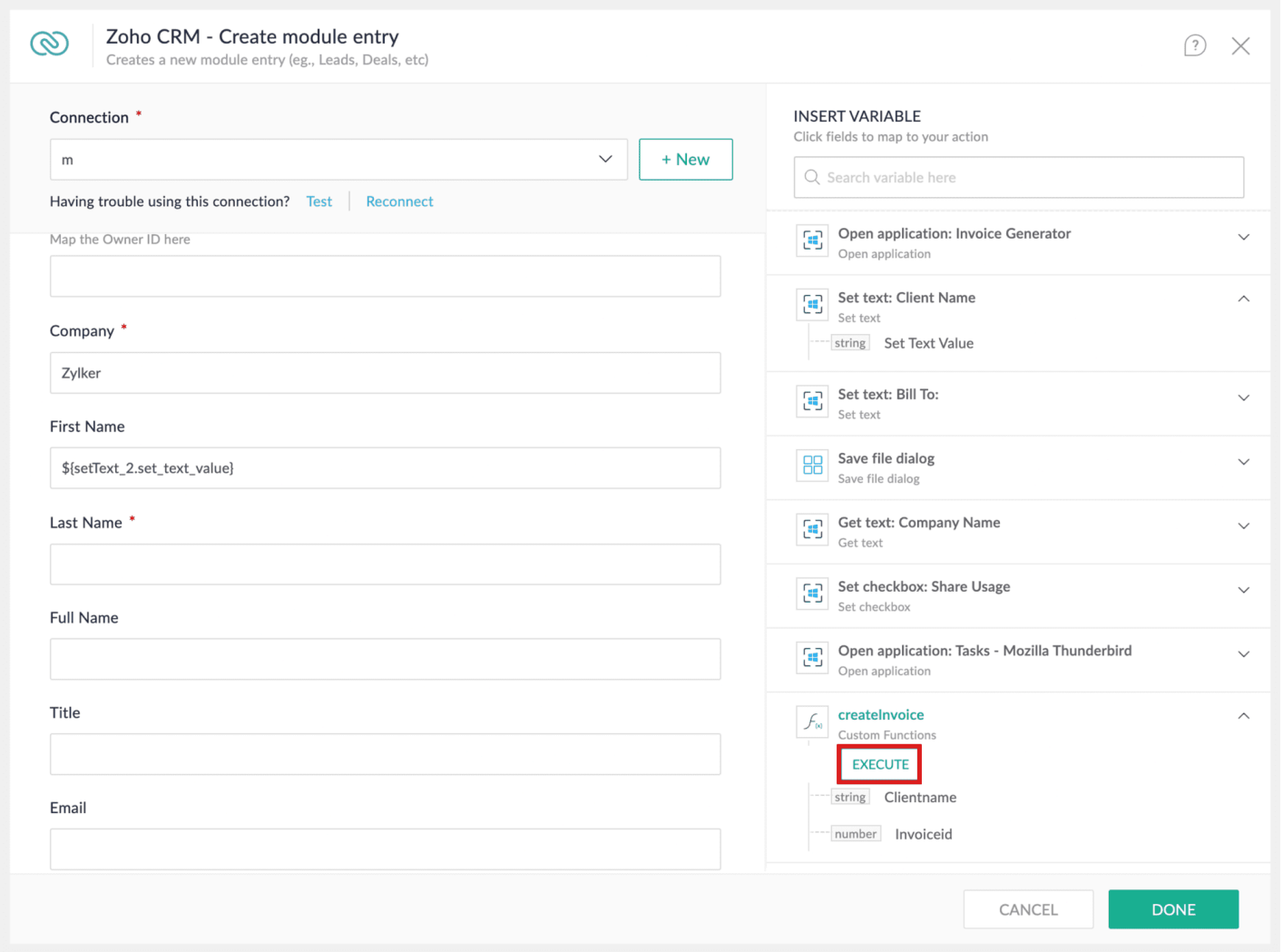
Formula to extract the email of the second element: **${trigger.contacts[1].email**}
This will output "richard@zylker.com" in the target field.

Note:

- Indexes in a list always start at 0.
- The variable name assigned to a trigger is 'trigger' by default.

Extract data from the output of a custom function:

1. Locate the custom function in the variable pane.
2. Click the Execute button.



3. Enter test data, if required.
4. Once executed, the function's outputs will appear in the variable pane.
5. Use these outputs seamlessly as inputs for other fields or functions within your RPA flow.

Format date and time values in fields:

Apps: When assigning variables to date fields, you have a format date time option that allows you to choose the date format that the variable uses. The date format will then be automatically converted to the required format the app expects.

RPA: RPA actions do not have any specific set date fields. However, when using the Set Text / Type into action to assign date values to Windows app fields or Web browser fields, you will need to make sure the date format matches the required format in the target application.

You can use custom functions to convert the date to the required format and then assign the value to the field.

Sample Custom Function

```
String convertDate(input dateValue)
{
convertedDate=dateValue.toDate(dd-mm-yy);
return convertedDate.toString();
}
```

Learn more about the toDate deluge function. You can also reference the date formats that you can use in this link.

Using system variables in fields:

You can use predefined date and time variables in field configurations. These date and time formats are fixed and can't be changed.

Current Date: Provides the date when the task is executed and passed on to the mapped field.
Format: yyyy-MM-dd (e.g., 2024-01-23)

Current Date-Time: Provides the date and time at the time of task execution to the mapped field.
Format: yyyy-MM-ddThh:mm:ssZ (e.g., 2024-01-19T23:30:30+05:30)

How to map data in your RPA flow

1. Find your variables.

When configuring an action, look for the 'Insert variable' section on the right. This lists all available variables:

- **System variables:** current date, current date and time.
- **Output variables:** from triggers and actions that support outputs.
- **Custom variables:** created using the 'set variable' task.

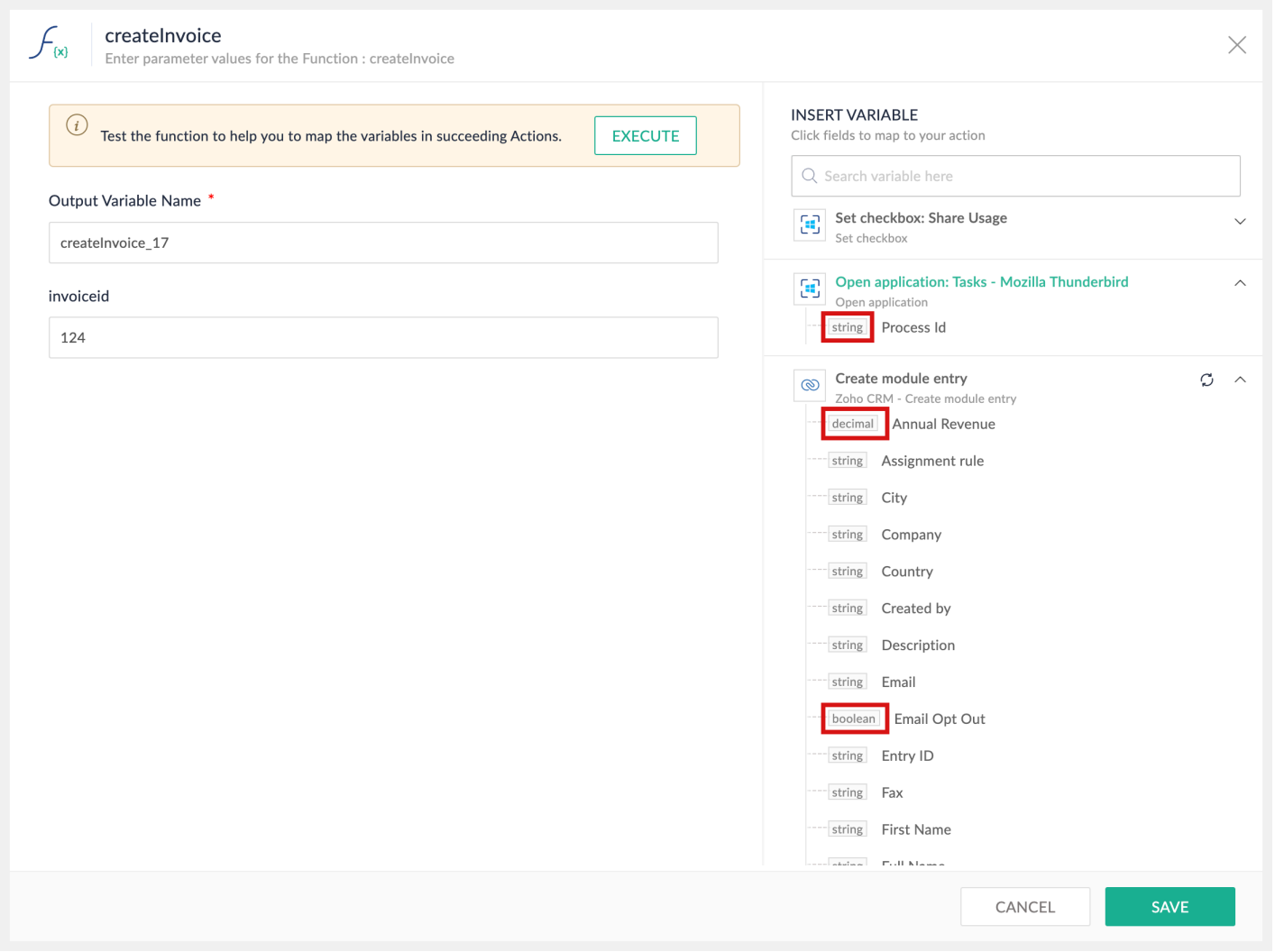
> **Note:** Variable and field names can only contain English letters and numbers. Other languages are not supported.

Click on an action name to see its specific variables.

2. Identify variable types.

Variable names are prefixed with icons indicating their type:

- **Text:** text fields.
- **Integer:** whole numbers.
- **Date:** dates.
- **DateTime:** dates and times.
- **Boolean:** true or false value
- **Email:** email addresses.
- **Decimal:** numbers with decimal places.



3. Map your variables.

Click the desired variable in the list, then click the corresponding input field. The variable will automatically populate the input field.

Tips

Consistent data in mandatory Fields:

Ensure that variables mapped to mandatory fields consistently return data. Empty mandatory fields during flow execution can lead to failure.

Review mappings after trigger/action changes:

Reassess mapped fields if you modify the trigger or action post-flow setup. Adjust mappings accordingly to maintain synchronization.

Caution with identical variable Names:

Exercise caution when variables share identical names across different steps (e.g. "name" or "date"). Verify mapping to the correct step to ensure the intended value usage, as errors during flow execution can result in failure.

Escape dollar sign:

To escape the dollar sign, use "$" twice. For example, $$40 in the field will give the output $40.