Jedox Microsoft Integration

MS Power Apps, MS Power Automate, MS Teams

Technical documentation

23.03.2023 | Jedox

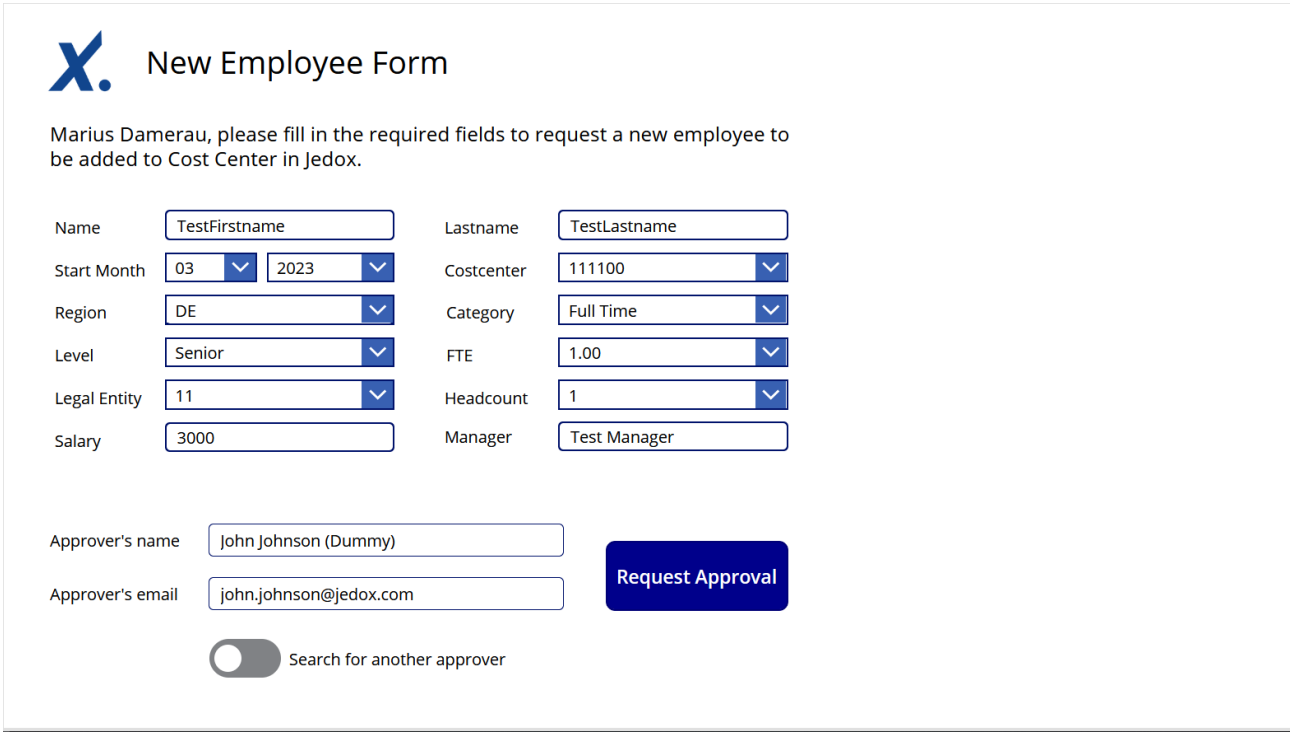
Table of Contents:

1. Introduction

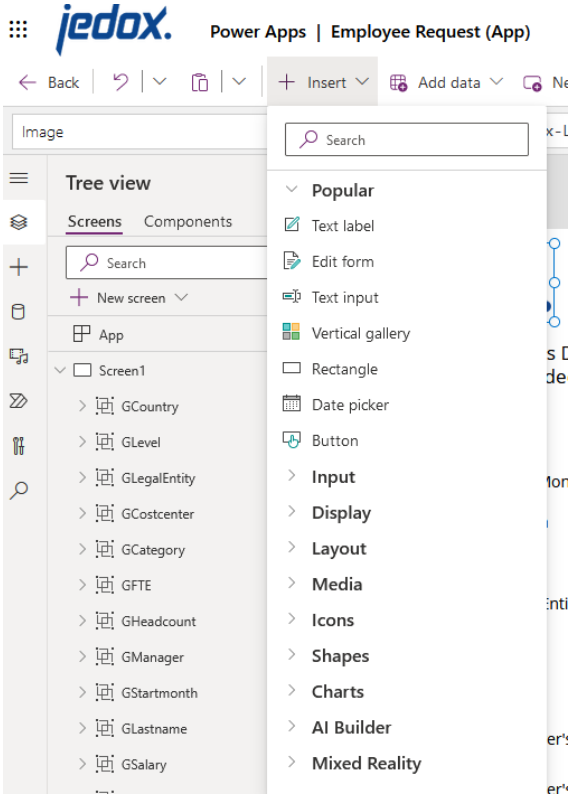
Jedox and Microsoft have a great connectivity through their integration of Microsoft Excel and Office 365 with the Jedox platform. It is a powerful and flexible solution for organizations looking to improve their planning and analysis processes. Wouldn't it be awesome to also optimize workflows with the help of MS Power Apps and MS Power Automate to directly react in MS Teams and having everything stored in a Jedox database?

The goal of this project is to add a new employee to an organization and enter them into a Jedox database via an approval process in Microsoft Teams.

An example structure of a Microsoft Power App, which triggers such an approval process, is shown below.



2. Building the Power App



1.1 Header

The header area consists of a logo, a headline and a description.

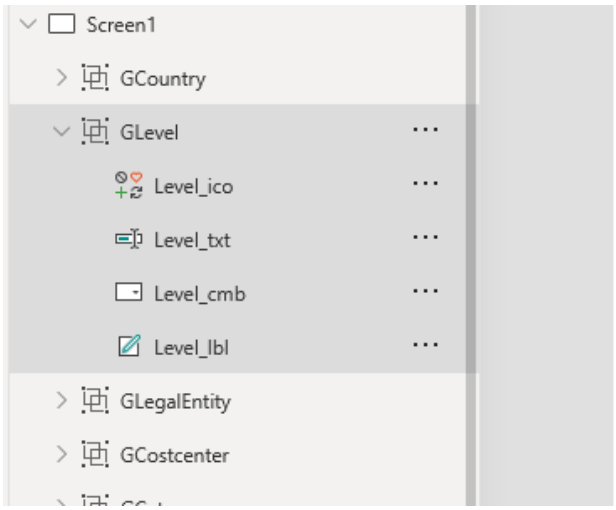
These can be added in the menu via “Insert” under “Media/Image” and “Display/Text label”.

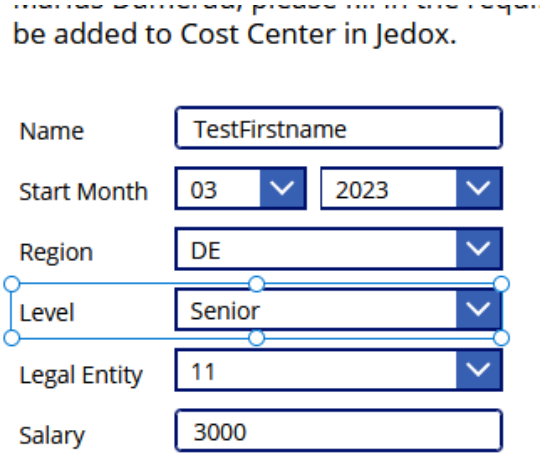
The name of the current user in the description is written with "User().FullName& " in the text label.

2.2 Variable fields

For the structure of the fields, it is important that all variables that are to be mapped in Jedox are defined in advance. In the example: Name, Lastname, Start month, Costcenter, Region, Category, Level, FTE, Legal Entity, Headcount, Salary, Manager, Approver's name, Approver's email.

Text fields can be easily created under “Input/Text input “and labeled with a “Display/Text label”. In the fields with a dropdown menu, it is possible to store default values and to enable the option of free input.



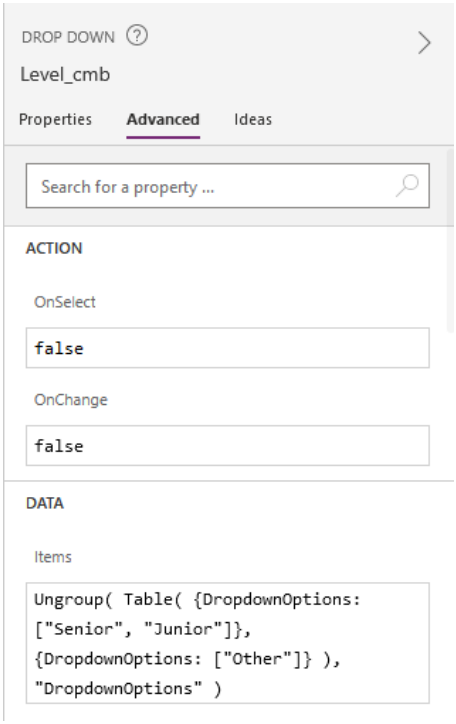


In the example of the Level field, four objects are required from the "Insert" option:

Display/Text label (Level_lbl), Input/Drop down (Level_cmb), Input/Text input (Level_txt) and Icons/Cancel (Level_ico).

The label Level_lbl is static and only describes the name of the field to be edited.

For the dropdown field Level_cmb needs a table with values, which is added under the properties of the field "Advanced/DATA/Items".




```
Ungroup(
    Table(
        {DropdownOptions: ["Senior", "Junior"]},
        {DropdownOptions: ["Other"]}
    ),
    "DropdownOptions"
)
```

Note: Tables can also be pulled from SharePoint

In addition, the visibility property "Advanced/Visible" must be edited under Advanced in order to no longer display a dropdown for the "Other" option.

```
Self.Selected.Value<>"Other"
```

The text input field Level_txt also needs the visibility property to ensure that it only appears when the dropdown menu disappears.

```
!Level_cmb.Visible
```

Also a "Cancel" symbol to be able to undo the "Other" option if necessary, is needed.

Advanced/OnSelect:

```
Reset(Level_cmb); Reset(Level_txt);
```

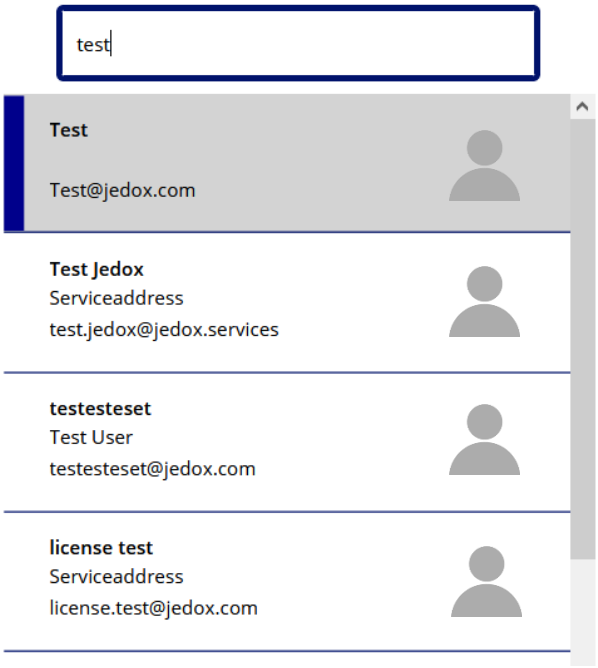
Advanced/Visible:

```
!Level_cmb.Visible
```

The behavior of the dropdown can be tested at any time on the play button in the menu at the top right. In order to maintain clarity in the "Tree View", it is advisable to mark all four elements, then right-click to group and name the group.

2.3 Approvement user gallery

The user gallery from the example consists of the gallery element, the profile picture, the name, the job title and the email address.



First, a vertical gallery is selected under "Insert/Layout". The Image element and the other predefined elements can be positioned, duplicated or deleted anywhere in the gallery row.

In order to create the link to the address book of the current environment, a text input field to search (in the example: ApproversSearch) is needed.
Then the function below is entered in the item event of the user gallery.

```
Filter(Office365Users.SearchUserV2({searchTerm:ApproversSearch.Text,top:100}).value,AccountEnabled=true)
```

The text fields can each be set in the DATA/Text event with "ThisItem.(field name of the 365 user)".
In the example: "ThisItem.Displayname", "ThisItem.Mail", "ThisItem.Jobtitle"
For the profile picture, the DATA/Image event of the image element is edited:

```
If(!IsBlank(ThisItem.Id),If(Office365Users.UserPhotoMetadata(ThisItem.Id).HasPhoto =
true,Office365Users.UserPhoto(ThisItem.Id),"profilepic-generic-user"),"")
```

The complete gallery is only visible in the example if the user sets the toggle "Search for another approver" to true.
The toggle is added via "Insert" "Input/Toggle" and is given the name "ApproversToggle".
Finally edit the visible event of the gallery and the search field and the gallery is ready.

```
ApproversToggle.Value
```

2.4 Request Button

The button is added via "Insert" "Input/Button"

The button will later trigger the Power Automate Flow, which must be created before giving the button a function. For the creation continue to the next page “3. Building the Power Automate Flow” in this document.

In the example, the following script is stored in the OnSelect event of the button.

```
If(Or(IsBlank(Name_txt.Text),IsBlank(Salary_txt.Text),IsBlank(Lastname_txt.Text),IsBlank(Manager_txt.Text)),Notify("Error. Every field must be populated",NotificationType.Error),'Sendapprovalandfollowupviaemail-1'.Run(ApproversEmail.Text,Name_txt.Text,Lastname_txt.Text,Month_cmb.Selected.Value,Year_cmb.Selected.Value,If(Category_cmb.Selected.Value="Other",Category_txt.Text,Category_cmb.Selected.Value),Country_cmb.Selected.Value,If(Costcenter_cmb.Selected.Value="Other",Costcenter_txt.Text,Costcenter_cmb.Selected.Value),If(Level_cmb.Selected.Value="Other",Level_txt.Text,Level_cmb.Selected.Value),If(FTE_cmb.Selected.Value="Other",FTE_txt.Text,FTE_cmb.Selected.Value),If(Legalentity_cmb.Selected.Value="Other",Legalentity_txt.Text,Legalentity_cmb.Selected.Value),If(Headcount_cmb.Selected.Value="Other",Headcount_txt.Text,Headcount_cmb.Selected.Value),Salary_txt.Text,ApproversName.Text,Manager_txt.Text);Navigate(Screen2))
```

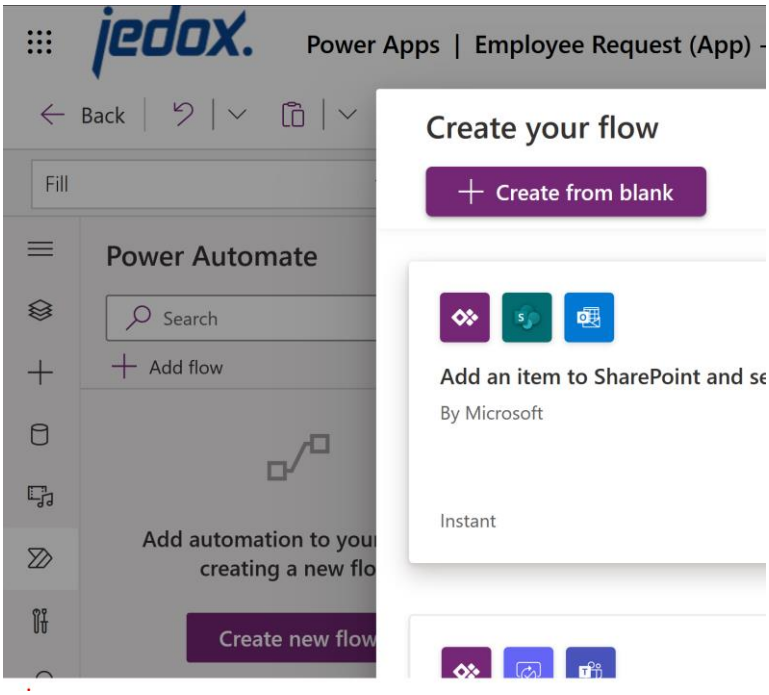
The function of the request button is composed as follows:

If the simple text fields are blank, send an error message explaining that every field must be populated.
If the simple text fields are populated, run the Power Automate Flow 'Sendapprovalandfollowupviaemail-1' with variables.
Every variable that is written in a dropdown menu or a text field is checked if the “Other” option is selected in the dropdown. If it is the “Other” option, take the variable from the text field, if not, take it from the dropdown.
After everything, navigate to “Screen2”.

Screen2 is displaying a Button with a Exit() function, to close the Power App.

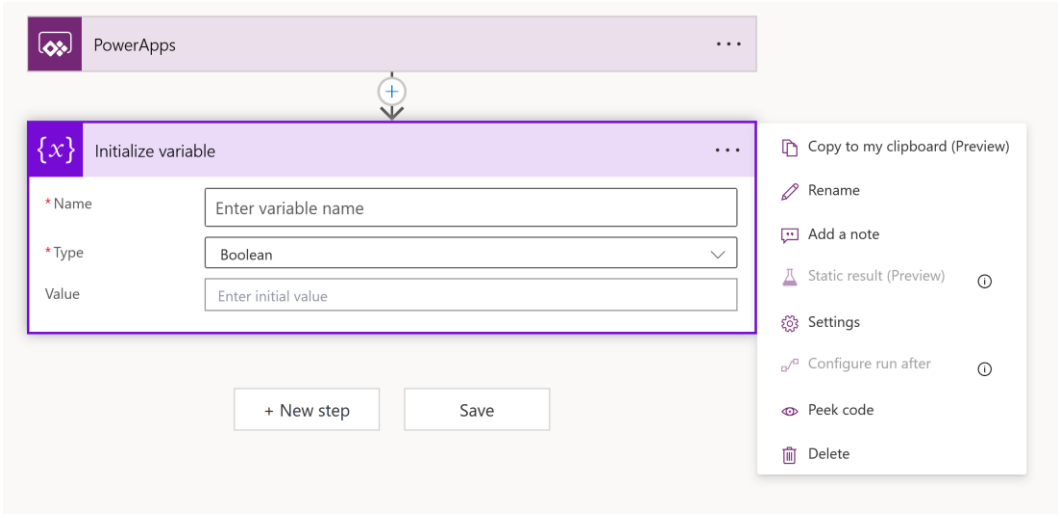
3. Building the Power Automate Flow:

In the editing environment of the Power App there is the opportunity to link a new flow at “Power Automate” in the right pane, “Create new flow”, “Create from Blank” and give the flow a Name.



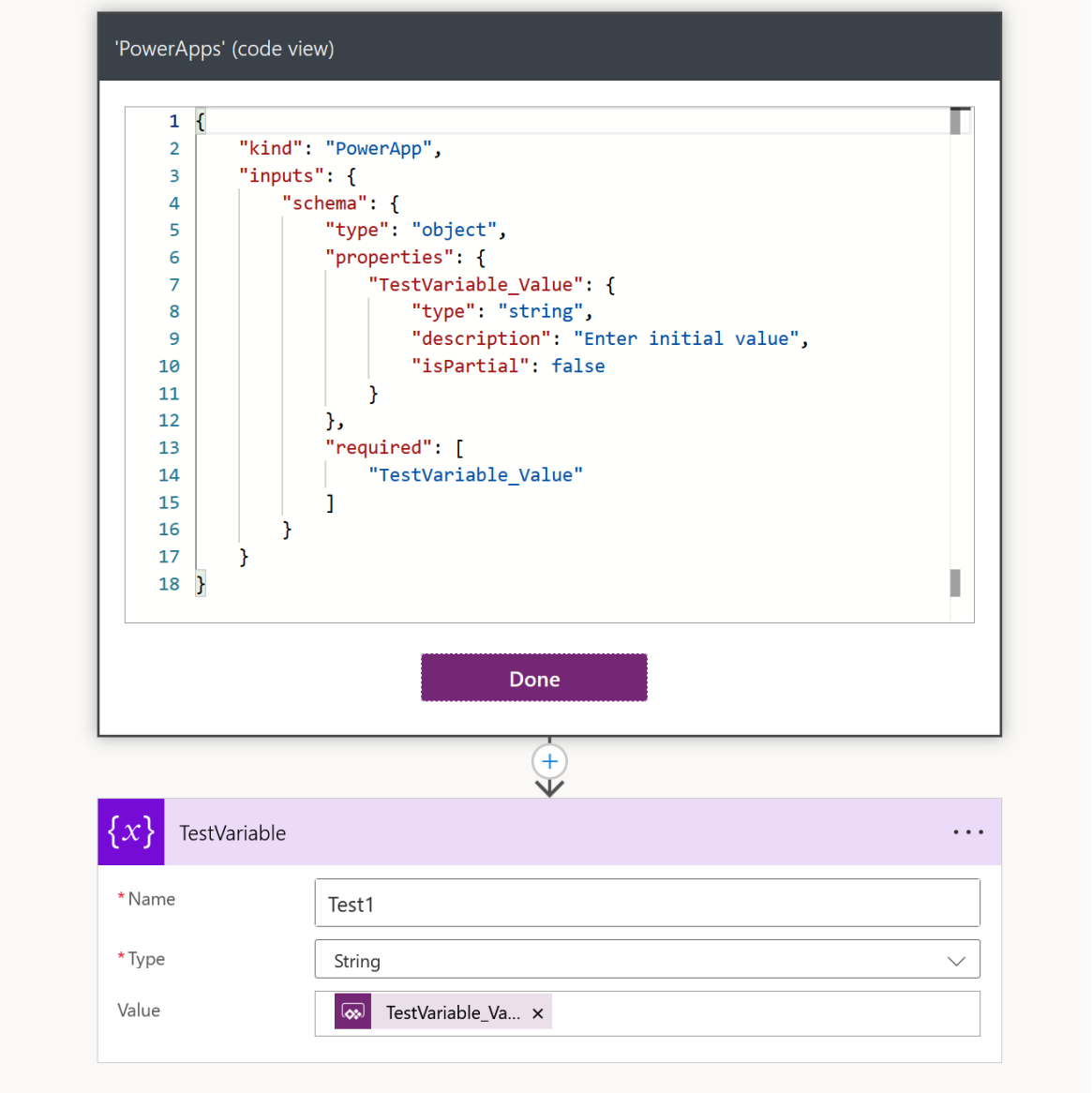
3.1 Setting up all variables that should be passed to Jedox

Choose “new step”, search for “variable” and choose initialize variable. Be sure to rename the “initialize variable” step at the three dots at the left “Rename”, to a specific name. Also give the variable itself a name and choose the datatype string. Then continue to the value field and select “Ask in PowerApps”. It is automatically creating a variable coming from PowerApps with part of the name of the actual “initialize variable” step. That’s also why this was renamed first.



It is necessary to repeat this initialize variable part for every single variable that should be written to Jedox.

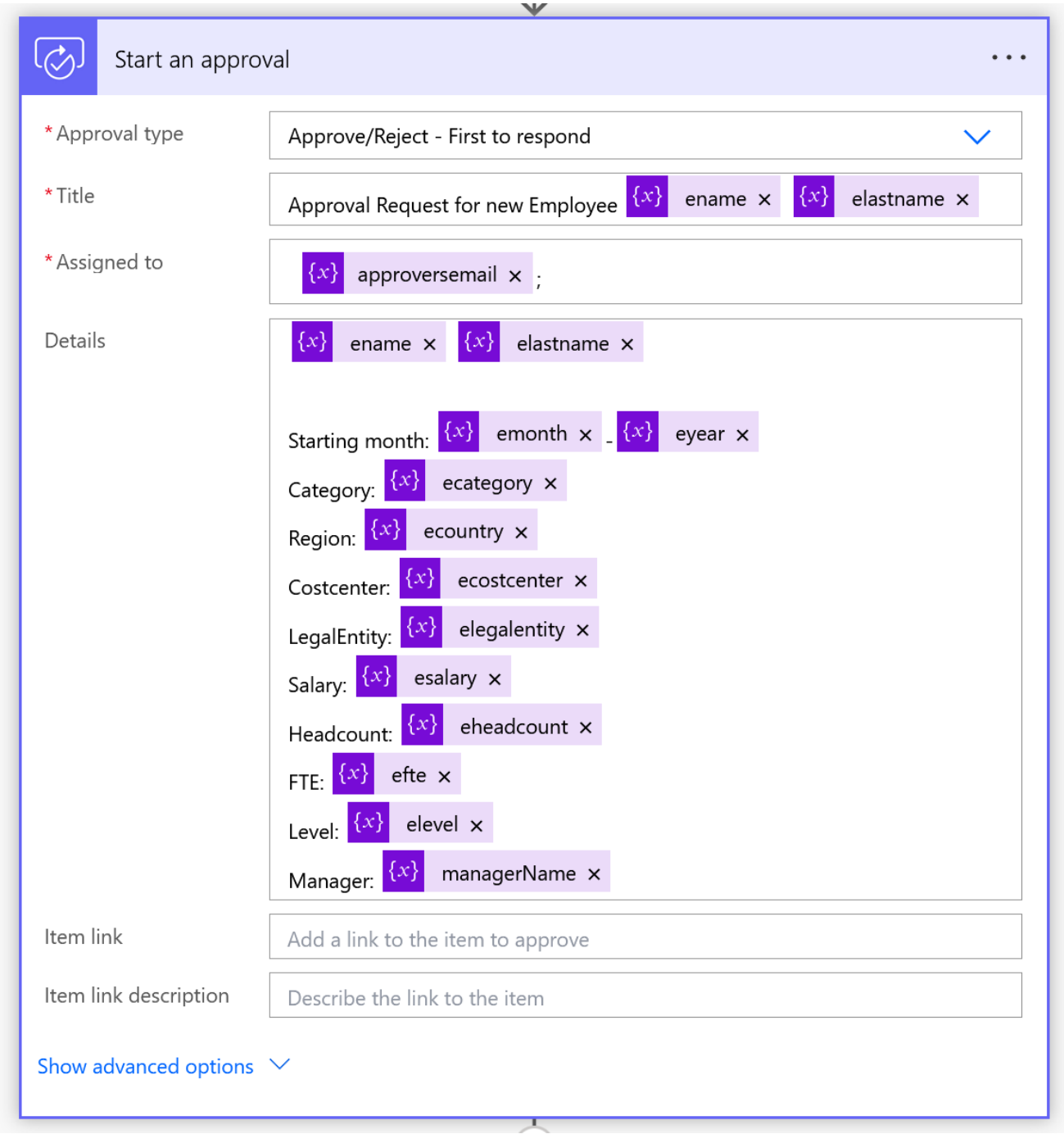
After creating every variable in the flow there is a way to doublecheck if the right (required) variables are chosen. In the first step “PowerApps” navigate to the three dots and click “Peek code”. Under required the variables should be listed.



Note: This PowerApps step is remembering every required variable step that was created. When deleting an existing “initialize variable” step for example because of improvement reasons, this variable will still be required for the flow. That means the only way for updating the required variables is to delete and recreate the “PowerApps” step.

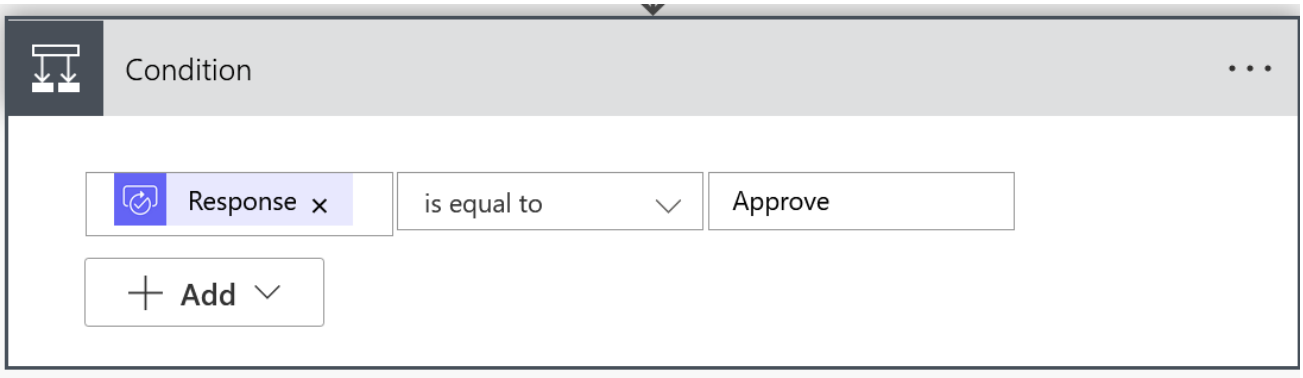
3.2 MS Teams approval request

The next step is, to start an approval request using MS Teams. Search for “Start and wait for an approval”. The Required information like “Assigned to” can now filled with the according variable. Also the rest of the information for the MS Teams approval can be edited with or without variables.



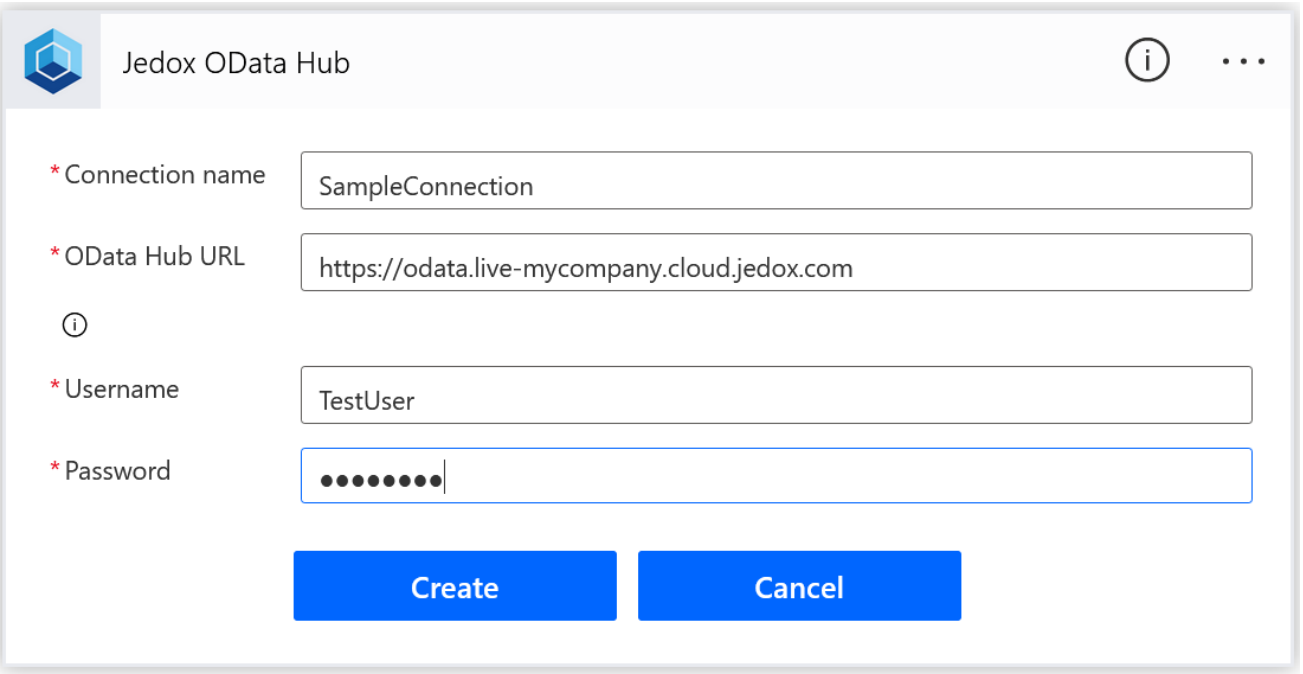
3.3 Condition

The reason of the condition, only if the authorization is approved there should be data in Jedox.
The other case, if the permission was denied, an action is not mandatory in this example.
So search for "condition", select it and choose "Response" in the first field. Then choose "is equal to" in the second dropdown field and type "Approve" in the third field like this:



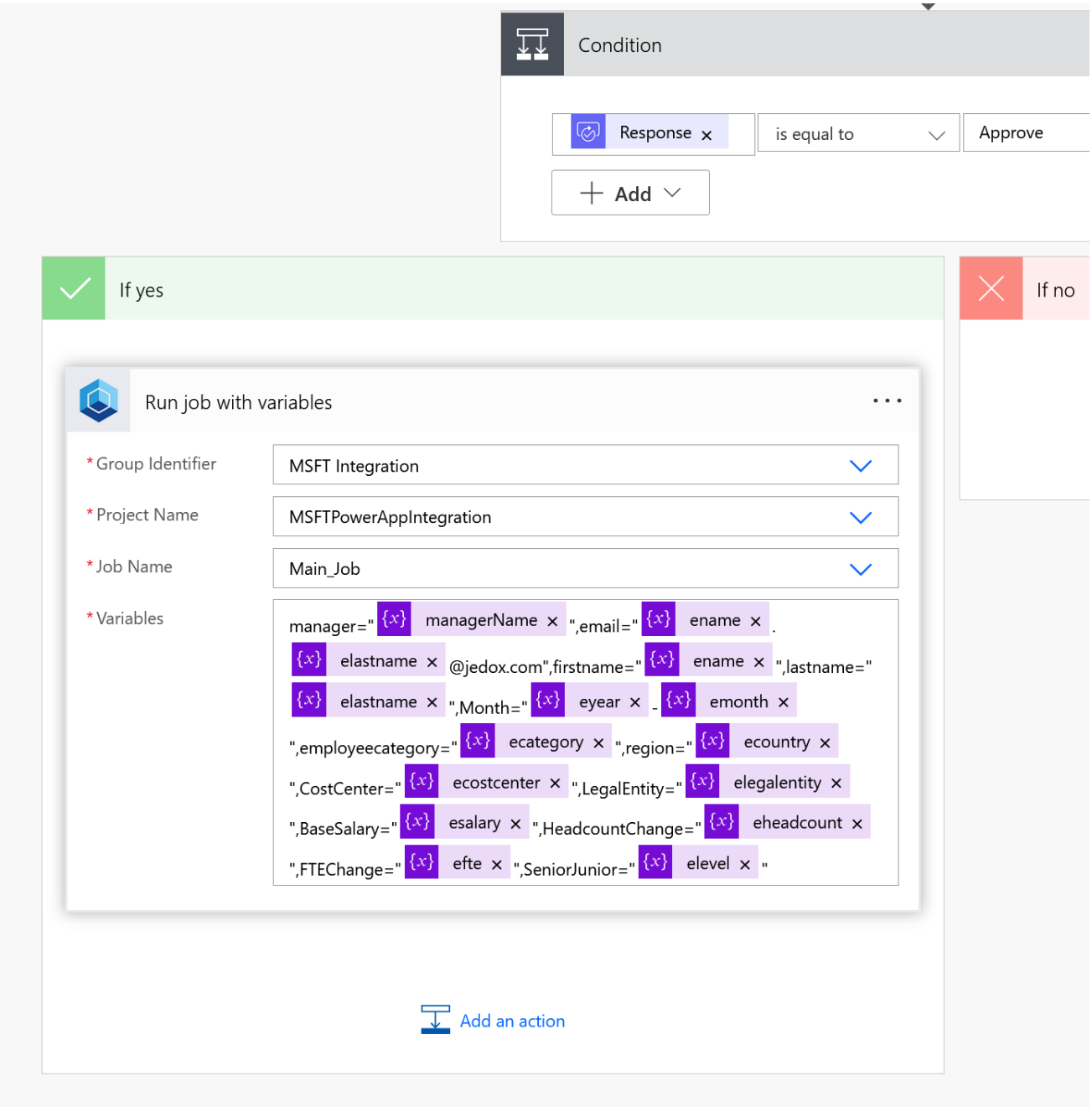
3.4 Jedox OData Connection

Moving on the "If yes" branch of the condition creating a new action. Choose "Run job with variables", For that a new Odata connection is needed. Choose "+Add new connection" and fill in the required fields to connect to a Jedox Instance.



Note: if the creation of the OData Connection throws an error, make sure that the OData service of your Jedox Instance is running. You can get this information from the cloud console or by asking cloud support.

When the Connection is set up, the Jedox ETL Job can be chosen by selecting the Model name and the ETL package name first. Next the variables can hand over by the variable name from the ETL project followed by an equal sign and the initialized variable from the Power App in quotation marks.
Several variables will be separated by a comma.



Note: When editing the Jedox Odata Connection, please make sure to use spaces in the package name and in the variable names of the integrator project.

4. How to dynamically link back to the Jedox report via a MS Teams message

Back to the Power Apps overview page clicking on the three dots of the created Power App, "Details", there is the Web link of the Power App. This link is useful to simply create a button with a hyperlink action in a Jedox report to open and edit the form that was successfully created before.

But how to get back to Jedox report after the whole approvement process?

4.1 Extending the Power App link with a parameter

In the Jedox report first step is to delete the hyperlink action in the button and replace it with a macro function.

```
function _Web_Link_Click ()
{
$instance = ActiveWorkbook()->names->item('instance')->value;
$url = 'https://apps.powerapps.com/play/e/default.......31f308'.'&instance='.$instance;
return __hyperlink($url);
}
```

Create the variable instance and fill it with the value of the Jedox cloud index name.
Then extend the URL with "&instance=" and the created variable ($instance=) and return this variable hyperlink.

Back in the Power App a visible text field is needed to cache the parameter.
Display/Text label (Param_lbl), toggle visible off and edit the text property.

```
Param("instance")
```

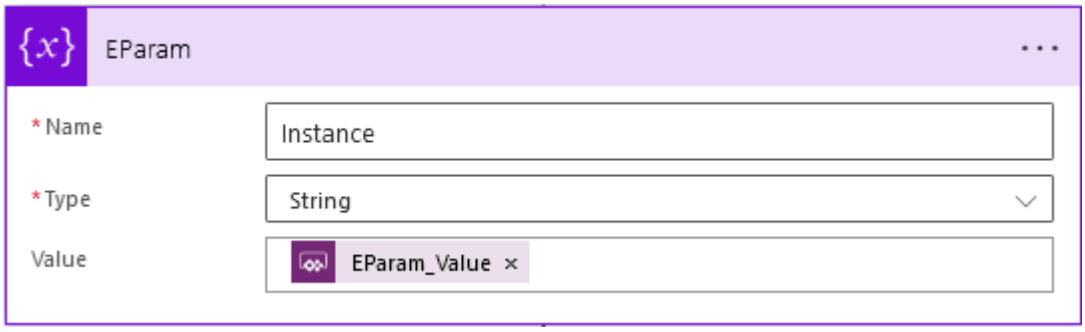
Also extend the request Button function to send the parameter to the flow.

```
Param_lbl.Text
```

After saving and publishing, move on to the Power Automate Flow.

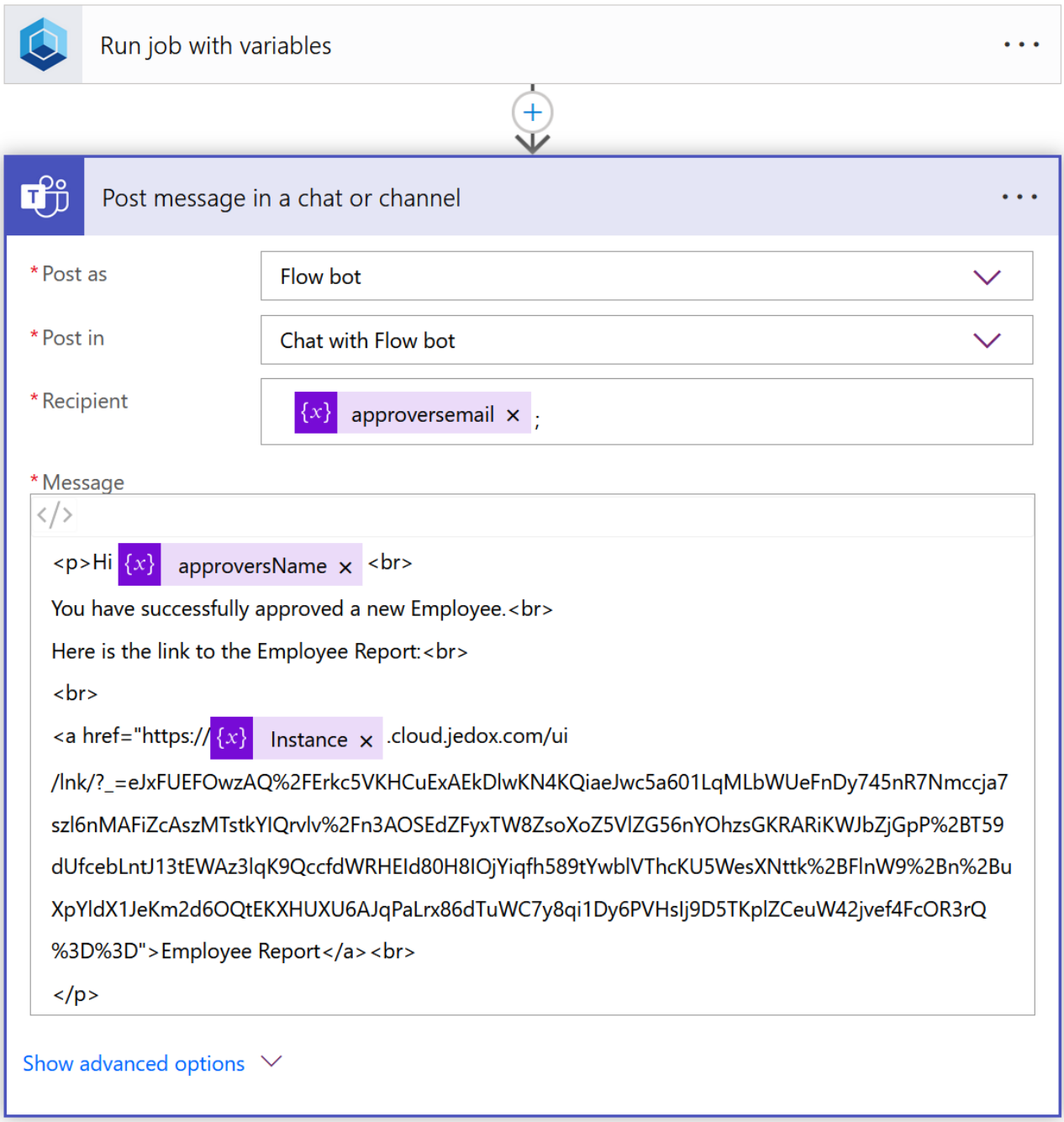
4.2 Using the parameter in the Power Automate Flow

Use "initialize variable" again, where the other variables were initialized, for the parameter. "Ask in PowerApps" to make this variable required.



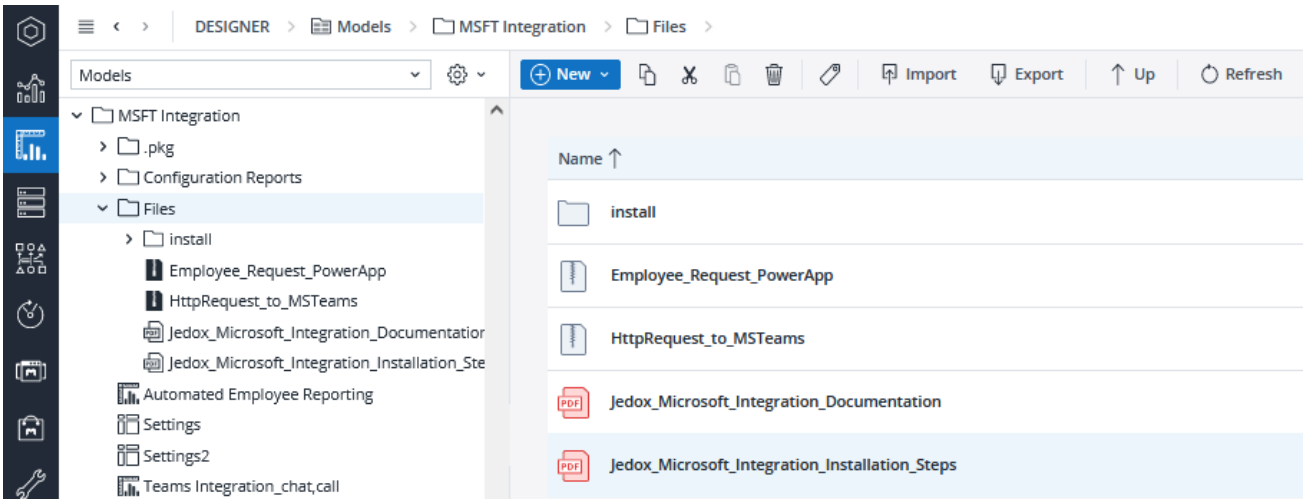
In the Flow, after “Run job with variables”, continue to make a new MS Teams step “Post message in a chat or channel”.
Post as “Flow bot”, Post in “Chat with Flow bot” and Recipient is the Email variable of the approver.
Then write a text and add a the Jedox report link (Right click on the Jedox report, Properties/Link).
Extend the Message field at the tools on the right “Code View”.
There is the opportunity to add the Jedox cloud instance parameter to the link.

Now, after the employee request is approved, the approver is getting a message from Power Automate with the link to the Jedox report in which the whole process was triggered.



5. Run an existing Power App with approval workflow in own environment

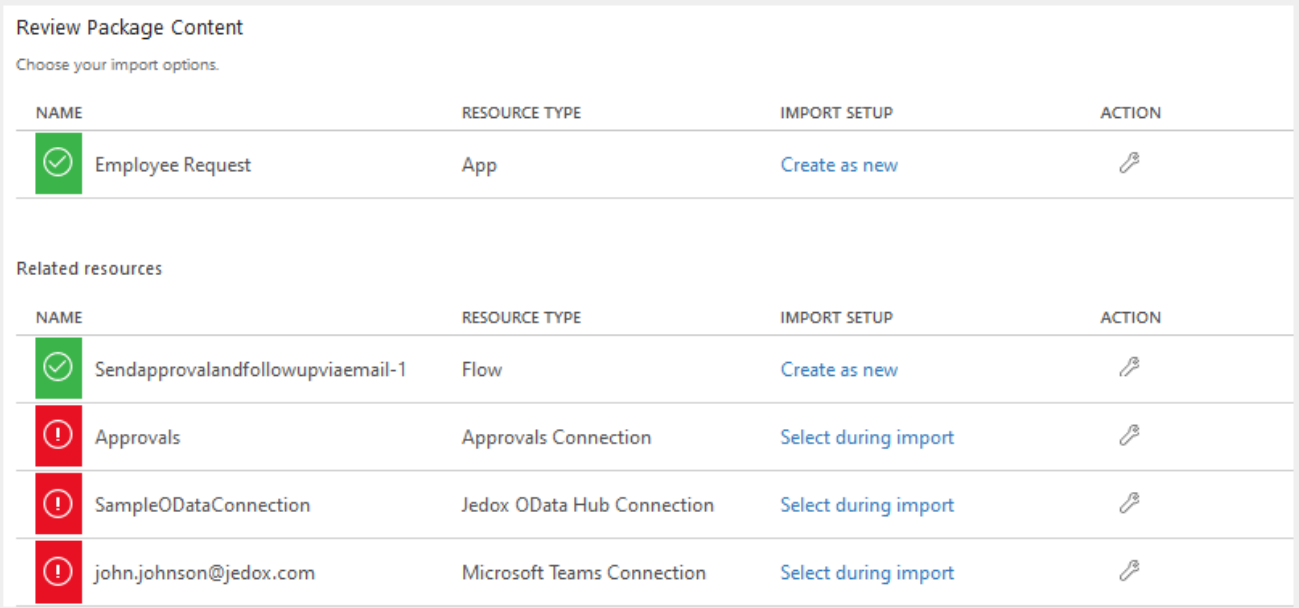
The Employee Request PowerApp and the HTTP Request Flow can be found in the Files Folder of the Model.



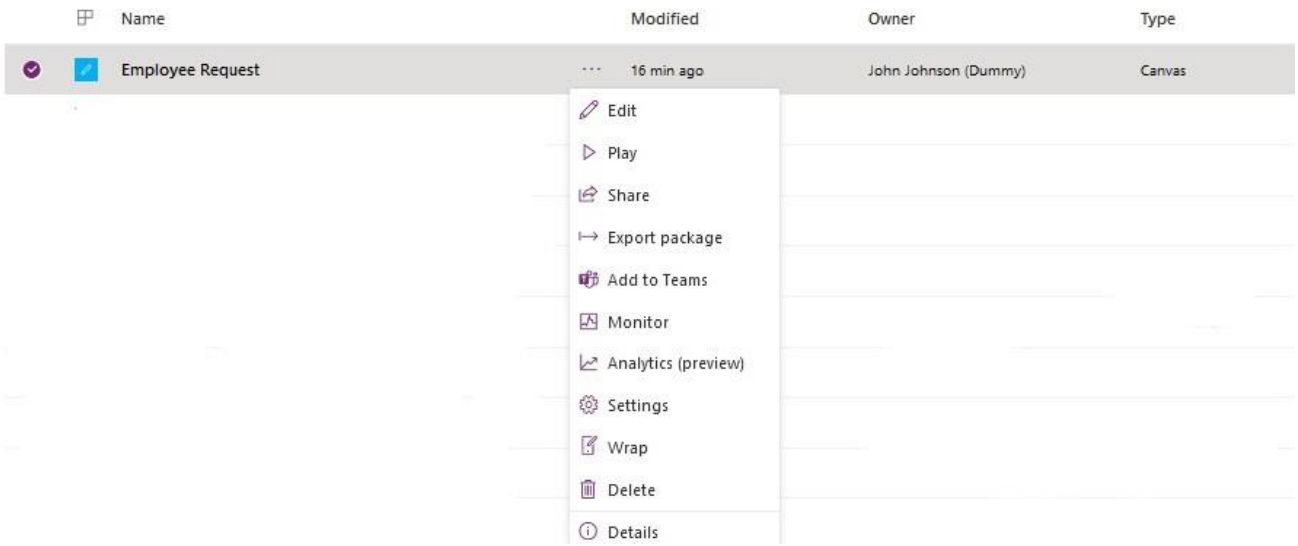
When exporting a Power App, there is the option to update the Power Automate Flow to this Power App. That means that it will also include the Power Automate Flow in this .zip package to download.

To import that .zip package in the own environment, go to Apps in MS Power Apps / Import canvas apps. The Premium User Plan of Power Automate is required for the OData Connection.
The App itself and the Flow can be taken over, but the Approvals Connection, MS Teams Connection and the OData Connection must be created and selected in the "Action" column on the right.

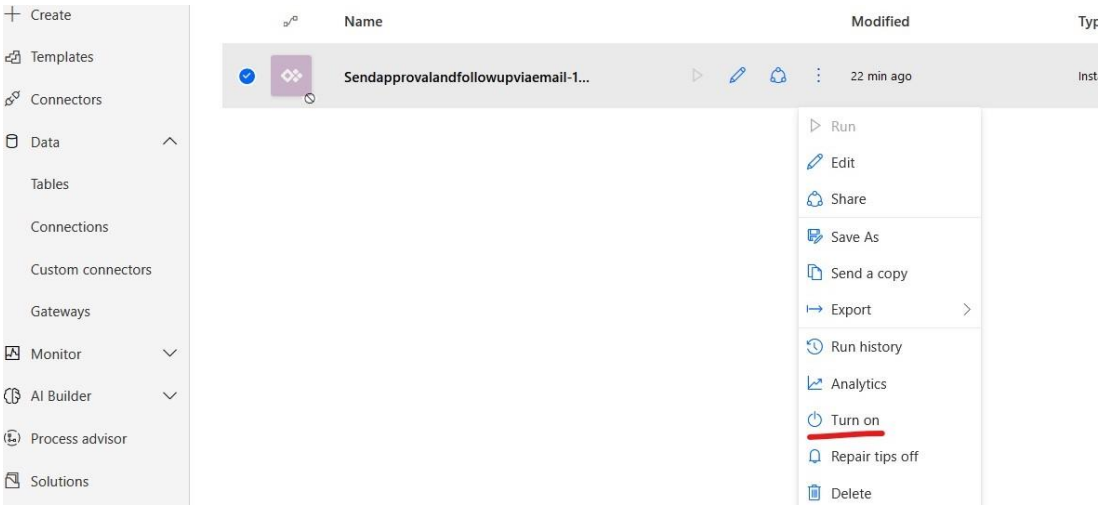
For save creation, especially for the OData Connection, go to (https://make.powerautomate.com/) Select "Data/Connections" in the right pane "New connection".



After editing the resources, the link of the Power App can be found in the Details of the Power App. (https://make.powerapps.com/apps) right click on the three dots “More Commands” of the app and select “Details” for showing the URL.

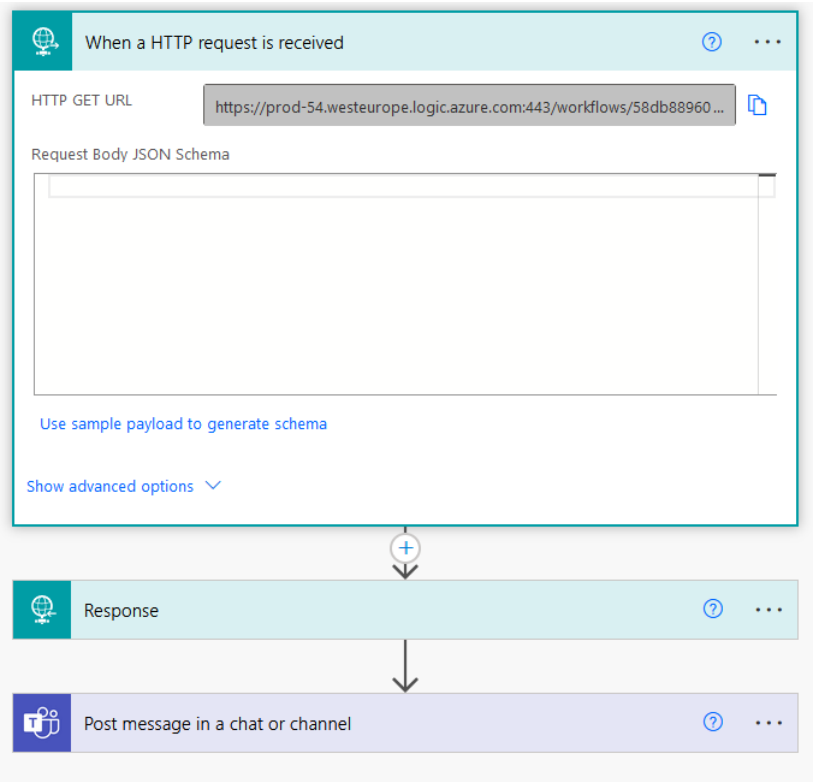


The taken over PowerAutomate Flow has to be turned on in the Flow overview. (https://make.powerautomate.com/flows)



In this Location also the “HttpRequest_to_MSTeams_20230421130831.zip” Flow can be Imported.
This is for another functionality of the Jedox MSFT Integration Model to directly send a message from Jedox to the PowerAutomate Flow Bot via HTTP request.

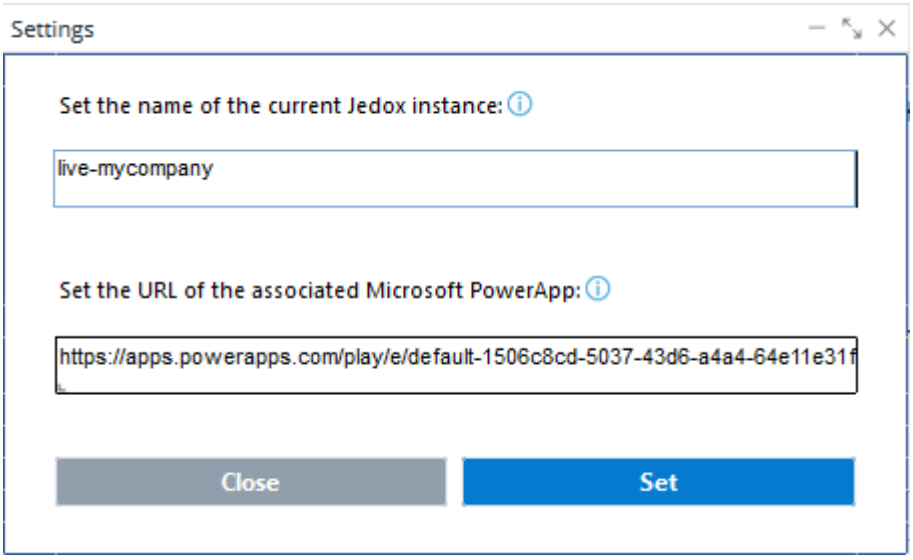
This Flow also must be turned on and the HTTP Link can be copied by editing it.



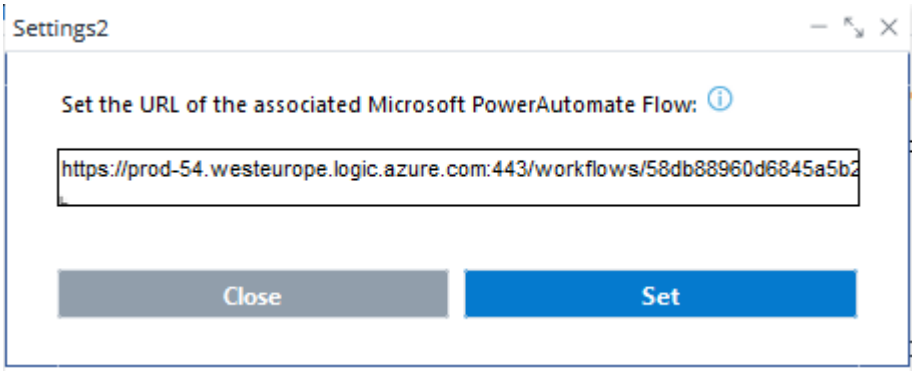
Note: If there are errors in this this process, reestablishing the used connections and reattaching to the associated flow solves most of them. Sometimes also refreshing the used flow in edit mode of the PowerApp can help.

5.1 Setting up the PowerApp and PowerAutomate functionality in the Jedox Model

To guarantee the functionality of the MSFT Integration Model it is important to not using a namespace when installing from the Jedox Marketplace.

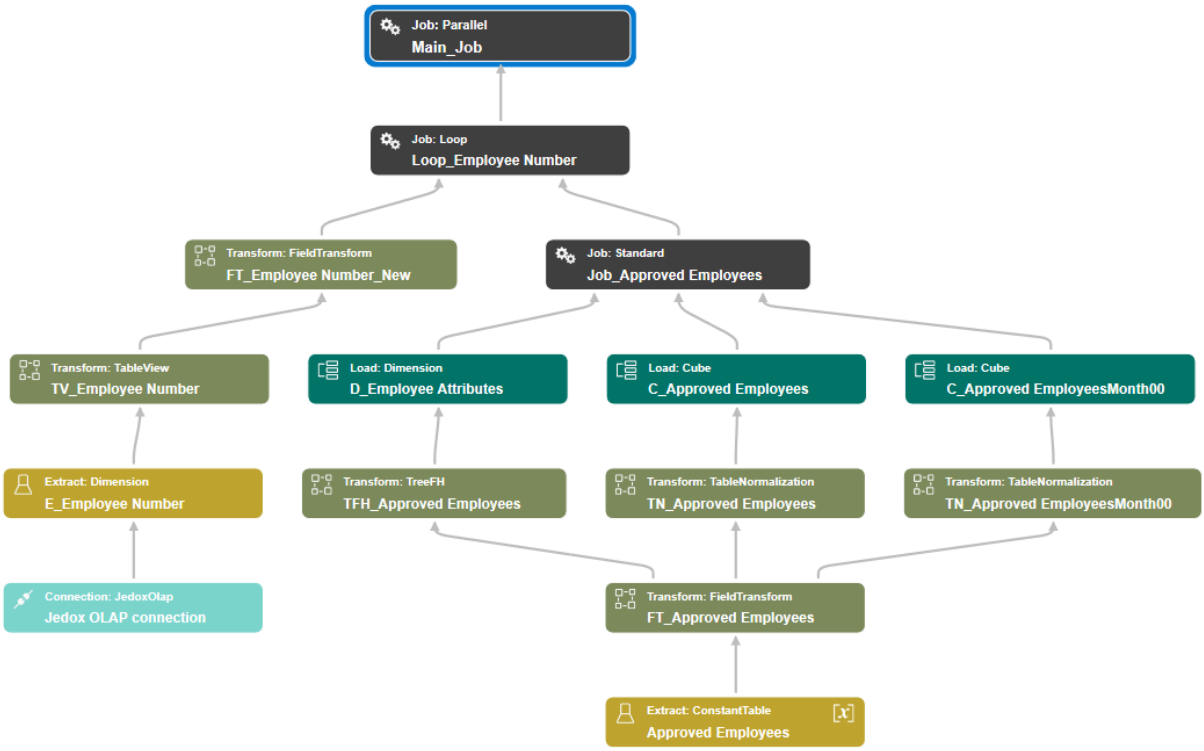


For the Employee Request PowerApp there is a settings box in the Jedox Model where the name of the Jedox instance and the PowerApp URL must be set.



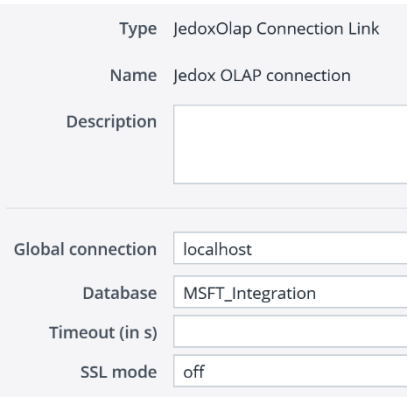
Also the HTTP request flow can be set in the Settings of the respective functionality.

6. ETL process to receive the variables of the example in a Jedox cube

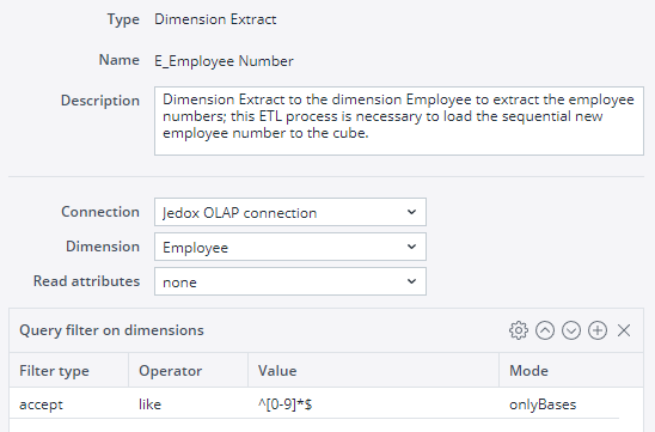


Create a new model/database/download OnePlatform if required.

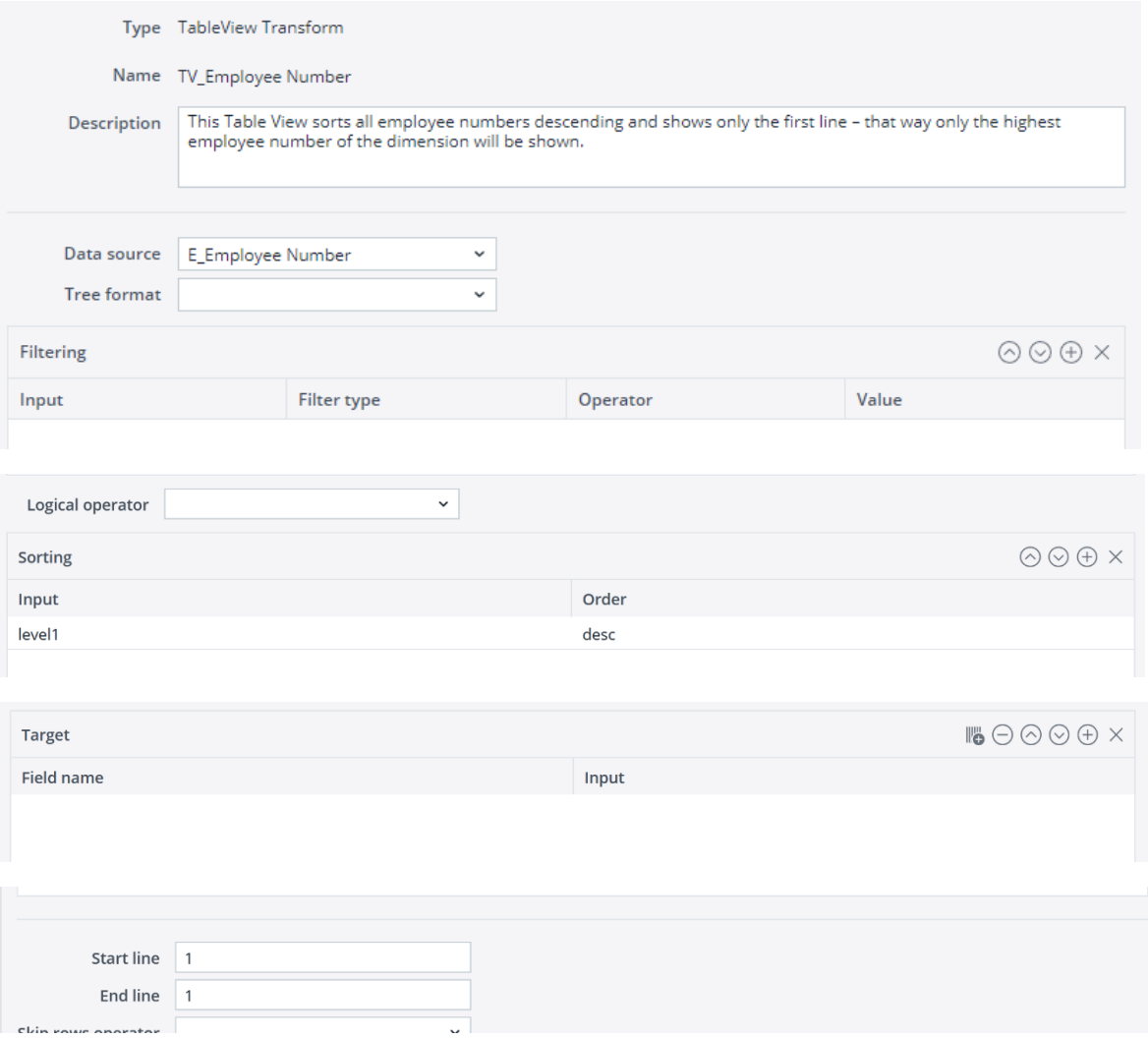
Create a JedoxOlap connection to the Database:



Create a Dimension Extract to the dimension Employee to extract the employee numbers; this ETL process is necessary to load the sequential new employee number to the cube:

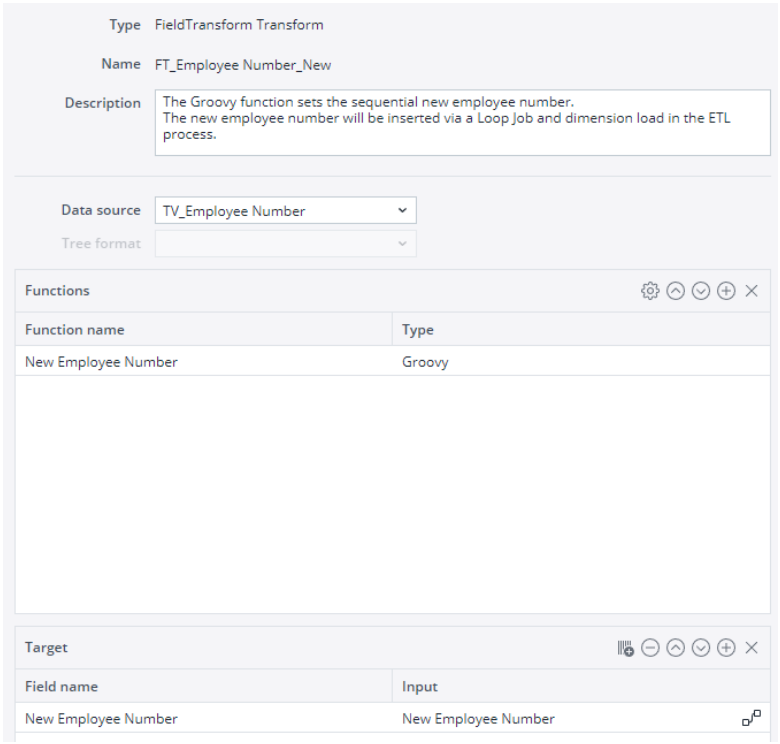


Create a Table View with the Dimension Extract as Data Source and consider the following inputs:

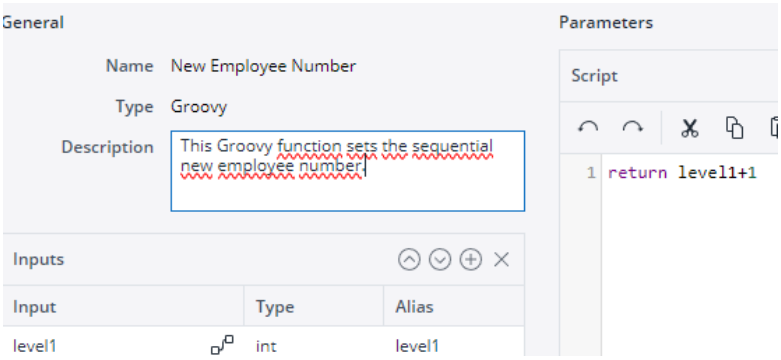


This Table View sorts all employee numbers descending and shows only the first line – that way only the highest employee number of the dimension will be shown.

Create a Field Transform with the Table View as Data source and a function with Type Groovy as Input in the Target section:



Input for the Groovy function:

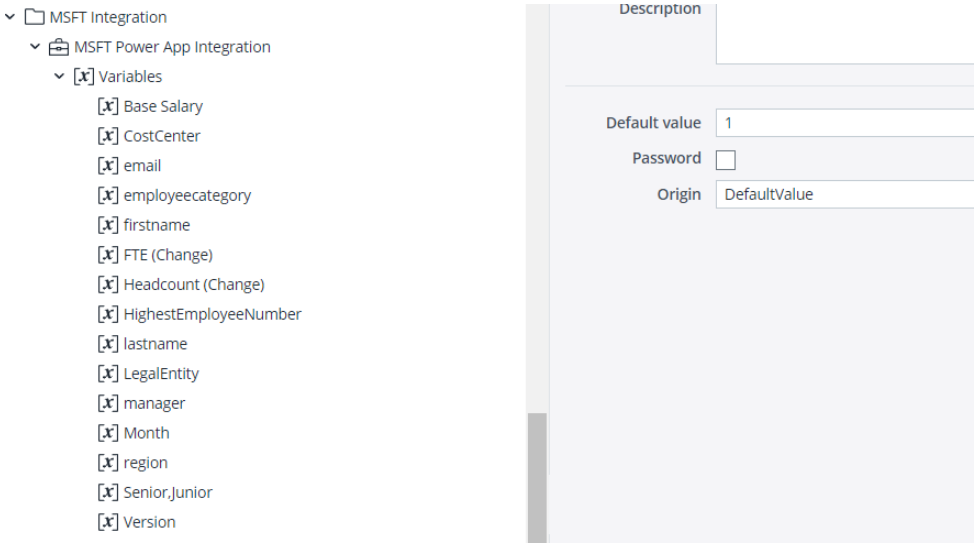


This Groovy function sets the sequential new employee number.

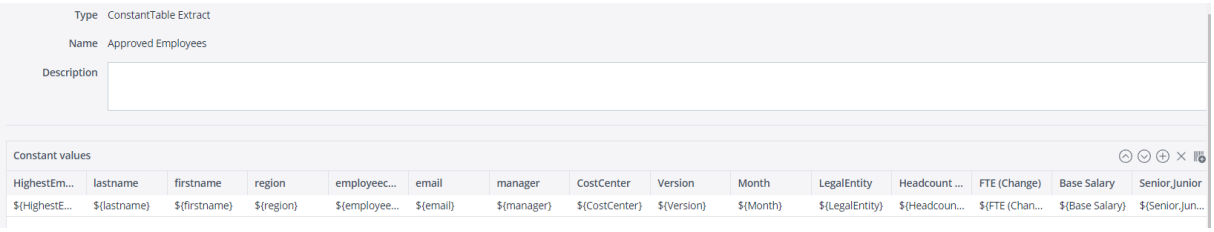
The new employee number will be inserted via a Loop Job and dimension load in the ETL process – for more details see overview on page 1.

Create the following variables in the integrator project:

Base Salary, CostCenter, email, employeecategory, firstname, FTE (Change), Headcount (Change), HighestEmployeeNumber, lastname, LegalEntity, manager, Month, region, Senior,Junior, Version



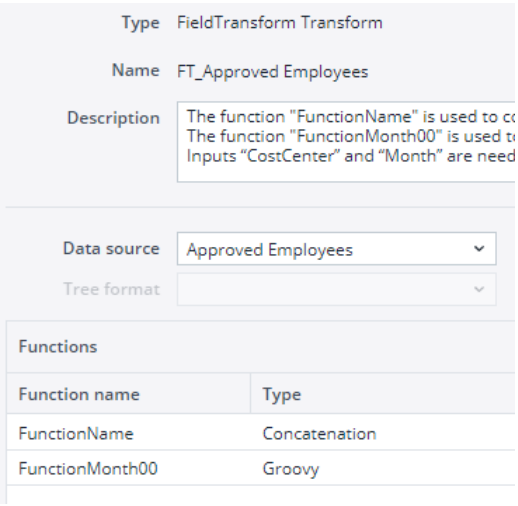
Create a Constant Table Extract containing all variables:



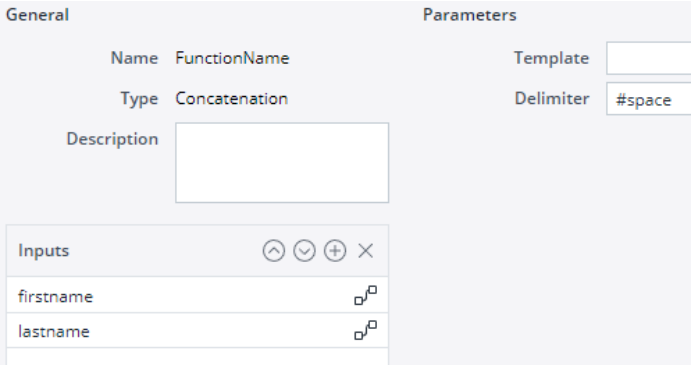
The variable "HighestEmployeeNumber" will come through the Loop Job. All other variables will be sent from the Power App Form via the Power Automate Flow.

Create a Field Transform to rename the Inputs.

Create two new functions within the Field Transform as input for the Target “Name” and the Target “Month00”:

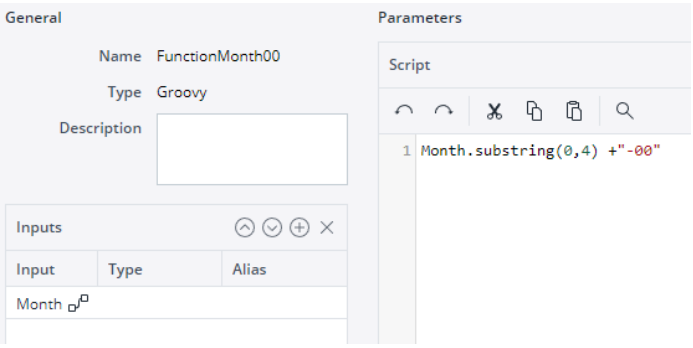


Input for the function “FunctionName”/Concatenation:



This function is used to combine first name and last name in one attribute for the Target “Name”.

Input for the function „FunctionMonth00”/Groovy:



This function is used to set Month to "-00" for the Target "Month00" and as input for the KPI "Full Time Base Salary (Change)".

Inputs "CostCenter" and "Month" are needed twice (CostCenter as dimension and as attribute for the dimension "Employee"; Month as dimension and as attribute "Starting Month" for the dimension "Employee"):

Target	
Field name	**Input**
HighestEmployeeNumber	HighestEmployeeNumber
Last Name	lastname
Surname	firstname
Region	region
Employee Category	employeecategory
E-Mail	email

Manager	manager
Cost Center	CostCenter
Version	Version
Month	Month
LegalEntity	LegalEntity
Cost Center_Dim	CostCenter

Headcount (Change)	Headcount (Change)
FTE (Change)	FTE (Change)
Base Salary	Base Salary
Senior,Junior	Senior,Junior
Name	FunctionName
Starting Month	Month
Month00	FunctionMonth00

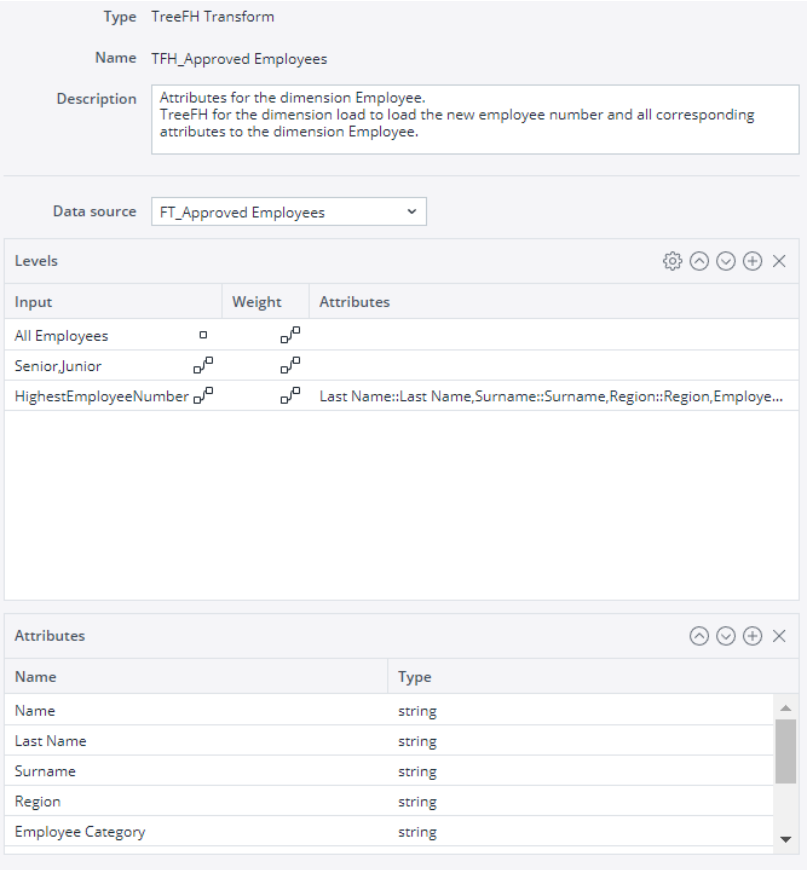
Create a TreeFH for the dimension load:

Select the created Field Transform as Data source.

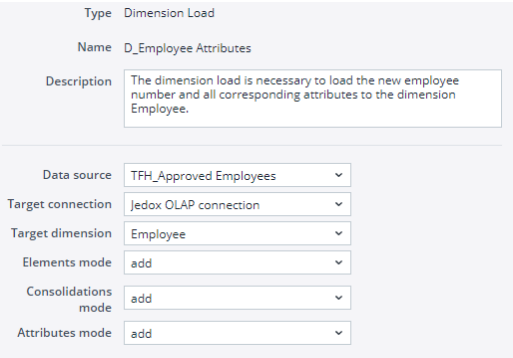
First add all the following Inputs as Attributes (attributes for the dimension Employee) as shown in the lower section of the following screenshot:

Last Name, First Name, Region, Employee Category, E-Mail, Manager, Cost Center, Name, Starting Month.

Add 3 levels as shown in the upper section of the screenshot and add all attributes to the lowest level (HighestEmployeeNumber).

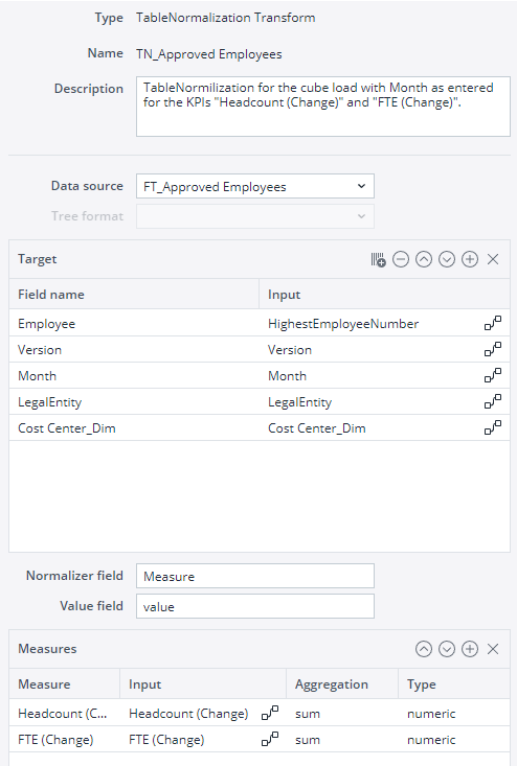


Create a new dimension load with the TreeFH as Data source:



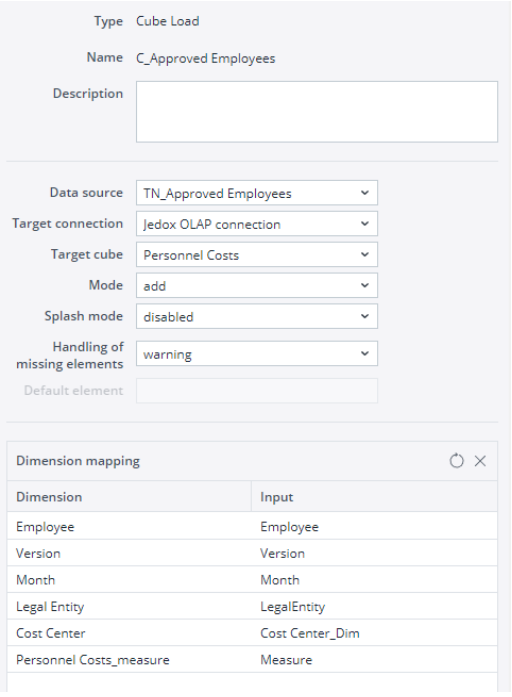
The TreeFH and the dimension load are necessary to load the new employee number and all corresponding attributes to the dimension Employee.

Create a TableNormalization (for one of the cube loads) as follows with the created Field Transform "FT_Approved Employees" as Data Source:



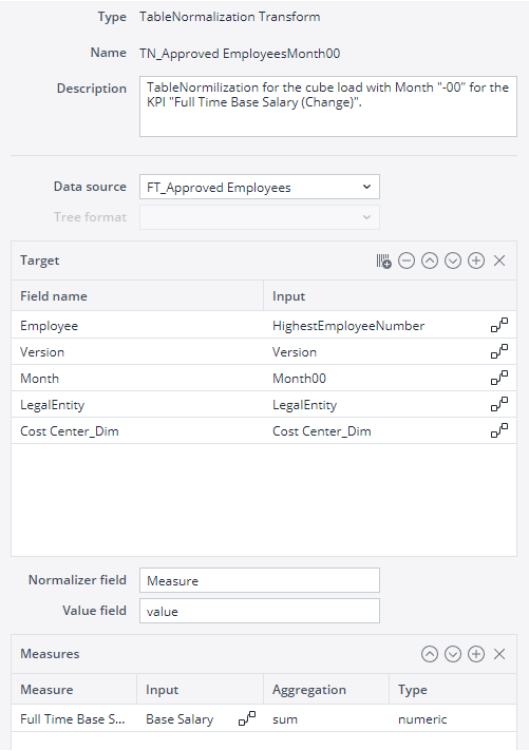
This TableNormalization Transform loads the KPIs "Headcount (Change)" and "FTE (Change)" to the given Month.

Create a Cube Load:



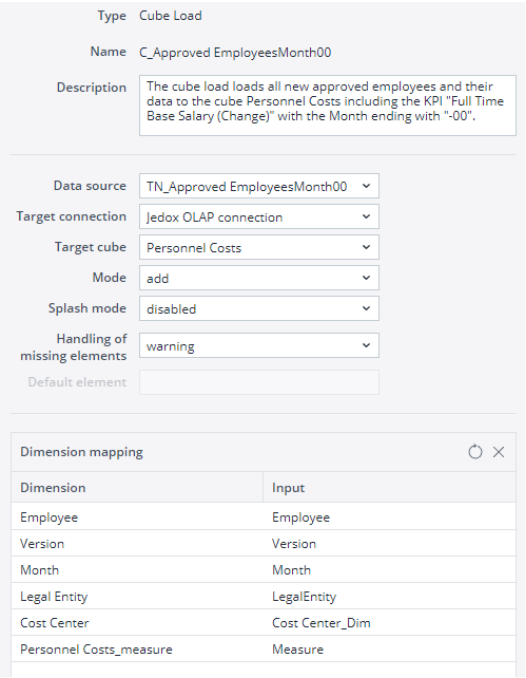
This cube load loads all new approved employees and their data including the KPIs "Headcount (Change)" and "FTE (Change)" to the cube Personnel Costs.

Create a TableNormalization (for one of the cube loads) as follows with the created Field Transform "FT_Approved Employees" as Data Source:



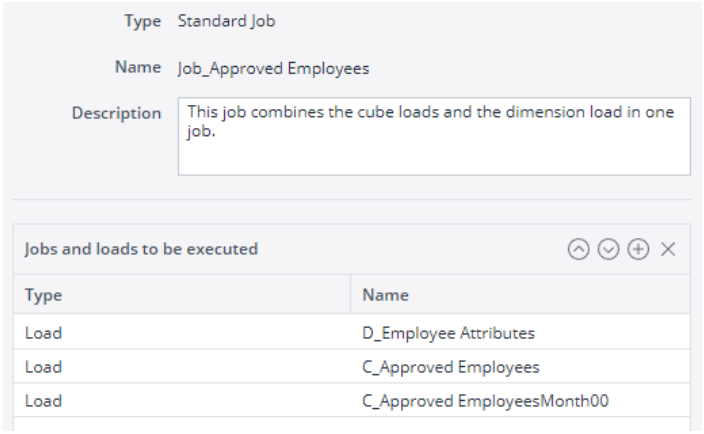
This TableNormalization Transform loads the KPI "Full Time Base Salary (Change)" to the Month00.

Create a Cube Load:



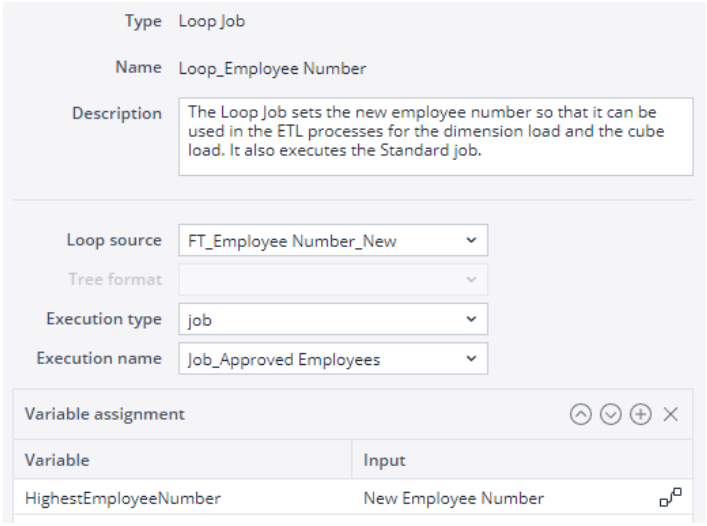
This cube load loads all new approved employees and their data including the KPI "Full Time Base Salary (Change)" to the cube Personnel Costs.

Standard Job Approved Employees:



This job combines the cube load and the dimension load in one job.

Loop Job Employee Number:



The Loop Job sets the new employee number so that it can be used in the ETL processes for the dimension load and the cube load. It also executes the Standard job.

Parallel Job:

This Job executes the Loop Job.