



Jedox Microsoft Integration

MS Power Apps, MS Power Automate, MS Teams

Technical documentation



Table of Contents:

1. Introduction.....	2
2. Building the Power App.....	3
1.1 Header	3
2.2 Variable fields	3
2.3 Approval user gallery	5
2.4 Request Button.....	6
3. Building the Power Automate Flow:.....	7
3.1 Setting up all variables that should be passed to Jedox	7
3.2 MS Teams approval request.....	9
3.3 Condition	10
3.4 Jedox OData Connection	10
4. How to dynamically link back to the Jedox report via a MS Teams message	12
4.1 Extending the Power App link with a parameter	12
4.2 Using the parameter in the Power Automate Flow	12
5. Run an existing Power App with approval workflow in own environment.....	14
5.1 Setting up the PowerApp and PowerAutomate functionality in the Jedox Model	17
6. ETL process to receive the variables of the example in a Jedox cube	18

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.



New Employee Form

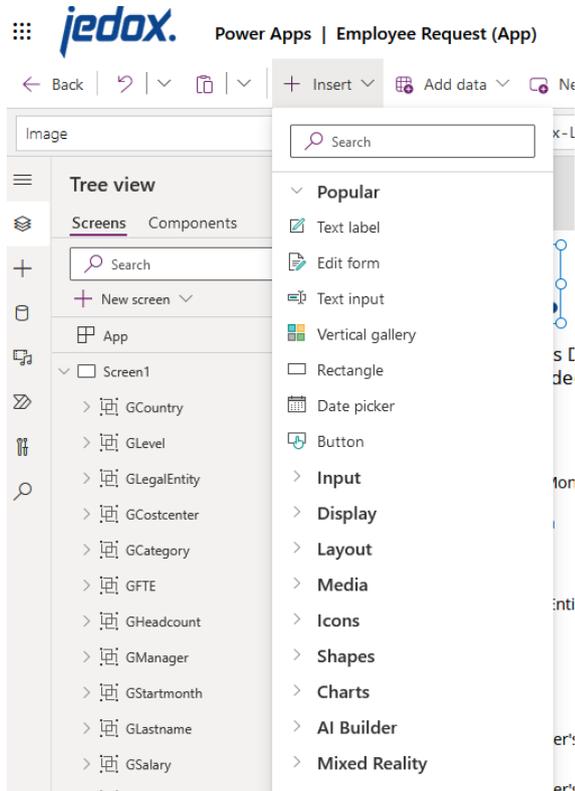
Marius Damerou, please fill in the required fields to request a new employee to be added to Cost Center in Jedox.

Name	<input type="text" value="TestFirstname"/>	Lastname	<input type="text" value="TestLastname"/>
Start Month	<input type="text" value="03"/> <input type="text" value="2023"/>	Costcenter	<input type="text" value="111100"/>
Region	<input type="text" value="DE"/>	Category	<input type="text" value="Full Time"/>
Level	<input type="text" value="Senior"/>	FTE	<input type="text" value="1.00"/>
Legal Entity	<input type="text" value="11"/>	Headcount	<input type="text" value="1"/>
Salary	<input type="text" value="3000"/>	Manager	<input type="text" value="Test Manager"/>

Approver's name	<input type="text" value="John Johnson (Dummy)"/>	<input type="button" value="Request Approval"/>
Approver's email	<input type="text" value="john.johnson@jedox.com"/>	

Search for another approver

2. Building the Power App



1.1 Header

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

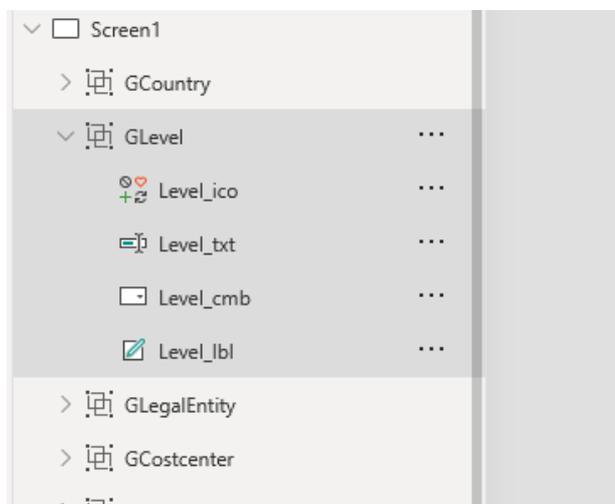
These can be added in the menu via "Insert" under "Media/Image" and "Display/Text label".

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.



These variables, please fill in the request form, be added to Cost Center in Jedox.

Name	<input type="text" value="TestFirstname"/>
Start Month	<input type="text" value="03"/> <input type="text" value="2023"/>
Region	<input type="text" value="DE"/>
Level	<input type="text" value="Senior"/>
Legal Entity	<input type="text" value="11"/>
Salary	<input type="text" value="3000"/>

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.

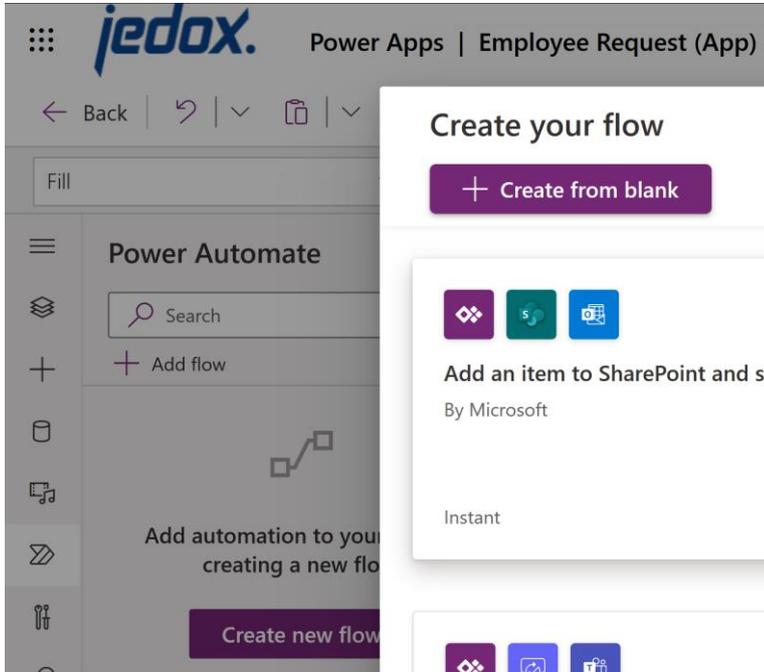
If the simple text fields are populated, run the Power Automate Flow '[Sendapprovalandfollowupviaemail-1](#)' with variables.

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.

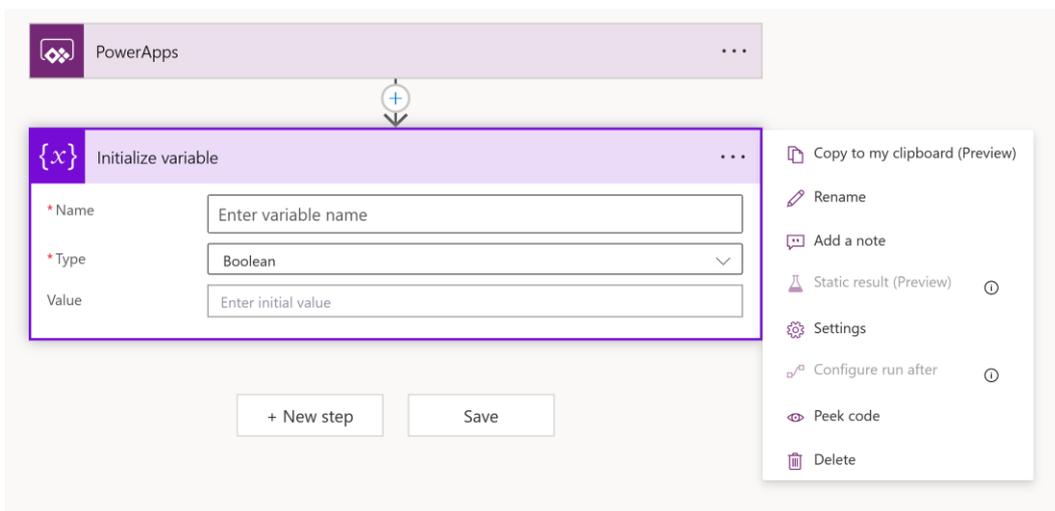
3. Building the Power Automate Flow:

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.

The image shows a screenshot of the PowerApps 'Peek code' view. The code is a JSON object representing a variable's schema. It includes a 'kind' of 'PowerApp', an 'inputs' object with a 'schema' for 'TestVariable_Value' (type: string, description: 'Enter initial value', isPartial: false), and a 'required' array containing 'TestVariable_Value'. Below the code is a purple 'Done' button. An arrow points from the 'Done' button to a variable configuration card for 'TestVariable'. The card shows the variable's name as 'Test1', type as 'String', and value as 'TestVariable_Va...'. The value field has a small icon and an 'x' to clear it.

```
1 {
2   "kind": "PowerApp",
3   "inputs": {
4     "schema": {
5       "type": "object",
6       "properties": {
7         "TestVariable_Value": {
8           "type": "string",
9           "description": "Enter initial value",
10          "isPartial": false
11        }
12      },
13      "required": [
14        "TestVariable_Value"
15      ]
16    }
17  }
18 }
```

Done

{x} TestVariable ...

*Name Test1

*Type String

Value TestVariable_Va... x

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.

Start an approval

* Approval type: Approve/Reject - First to respond

* Title: Approval Request for new Employee {x} ename x {x} elastname x

* Assigned to: {x} approversemail x ;

Details: {x} ename x {x} elastname x

Starting month: {x} emonth x - {x} eyear x

Category: {x} ecategory x

Region: {x} ecountry x

Costcenter: {x} ecostcenter x

LegalEntity: {x} elegalentity x

Salary: {x} esalary x

Headcount: {x} eheadcount x

FTE: {x} efte x

Level: {x} elevel x

Manager: {x} managerName x

Item link: Add a link to the item to approve

Item link description: Describe the link to the item

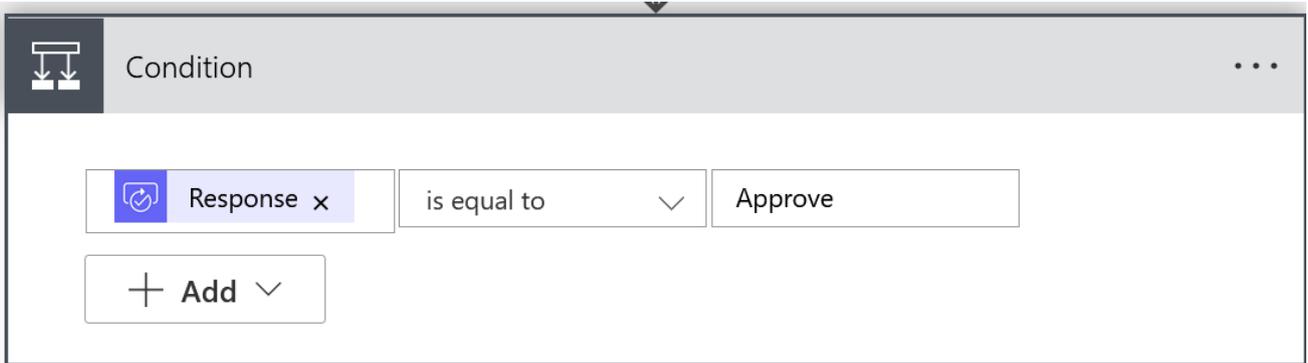
[Show advanced options](#) ▾

3.3 Condition

The reason of the condition, only if the authorization is approved there should be data in Jedox.

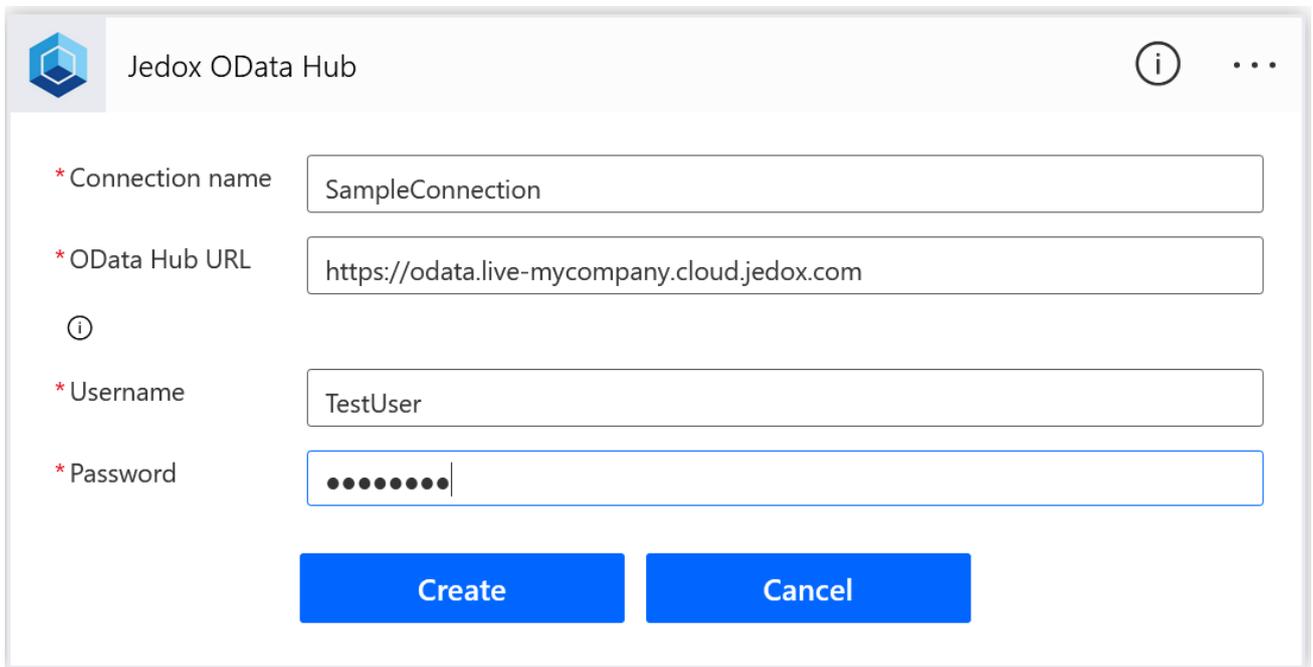
The other case, if the permission was denied, an action is not mandatory in this example.

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.

Condition

Response x is equal to Approve

+ Add

If yes

If no

Run job with variables

- * Group Identifier: MSFT Integration
- * Project Name: MSFTPowerAppIntegration
- * Job Name: Main_Job
- * Variables:


```

manager="{x} managerName x ",email="{x} ename x .
{x} elastname x @jedox.com",firstname="{x} ename x ",lastname="{x}
{x} elastname x ",Month="{x} eyear x _{x} emonth x
",employeeecategory="{x} ecategory x ",region="{x} ecountry x
",CostCenter="{x} ecostcenter x ",LegalEntity="{x} elegalentity x
",BaseSalary="{x} esalary x ",HeadcountChange="{x} eheadcount x
",FTEChange="{x} efte x ",SeniorJunior="{x} elevel x "

```

Add an action

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 approval 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.

The screenshot shows the configuration for a parameter named 'EParam'. It has three fields: 'Name' set to 'Instance', 'Type' set to 'String', and 'Value' set to 'EParam_Value'. The 'Value' field includes a small icon and a close button (x).

EParam	
* Name	Instance
* Type	String
Value	EParam_Value x

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.

The screenshot displays a Power Automate flow configuration. The first step is "Run job with variables". Below it, a plus sign icon indicates a new step is being added. The second step is "Post message in a chat or channel".

The configuration for the "Post message in a chat or channel" step is as follows:

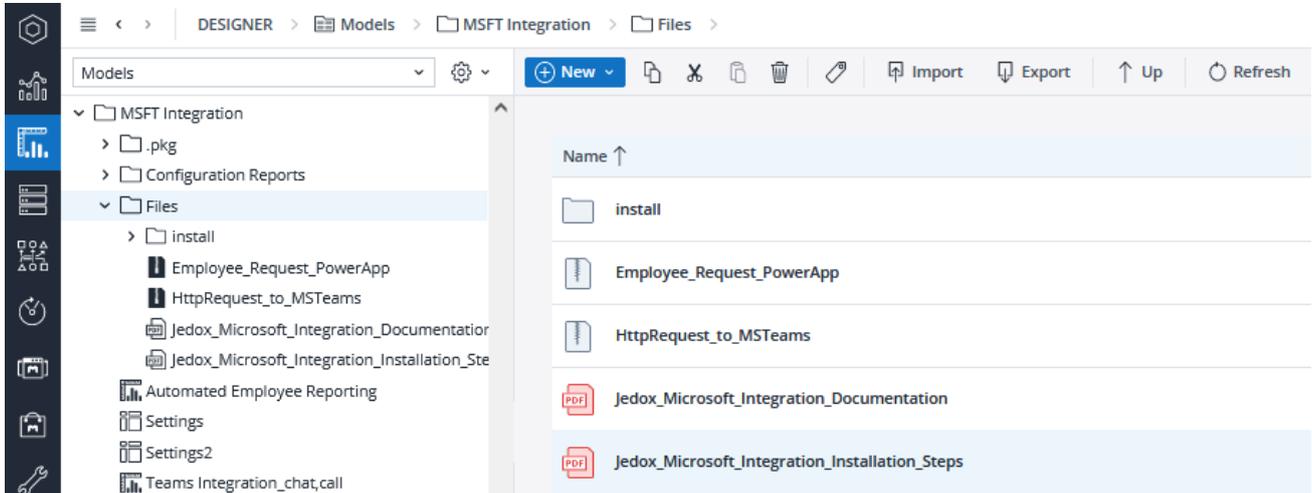
- Post as:** Flow bot
- Post in:** Chat with Flow bot
- Recipient:** {x} approversemail x ;
- Message:**

```
</>
<p>Hi {x} approversName x <br>
You have successfully approved a new Employee.<br>
Here is the link to the Employee Report:<br>
<br>
<a href="https://{x} Instance x .cloud.jedox.com/ui
/lnk/?_eJxFUEFOWzAQ%2Fkrkc5VKHCuExAEkDlwKN4KQiaeJwc5a601LqMLbWUeFnDy745nR7Nmccja7
szl6nMAFiZcAszMTstkYIqrvlv%2Fn3AOSedZFyxTW8ZsoXoZ5VIZG56nYOhzsGKRARiKWJbZjGpP%2BT59
dUfcebLntJ13tEWAz3lqK9QccfdWRHEld80H8IOjYiqfh589tYwblVThcKU5WesXNttk%2BFlnW9%2Bn%2Bu
XpYldX1JeKm2d6OQtEKXHUXU6AJqPaLrx86dTuWC7y8qi1Dy6PVHslj9D5TKplZCeuW42jvef4FcOR3rQ
%3D%3D">Employee Report</a> <br>
</p>
```

At the bottom of the configuration, there is a link to "Show advanced options" with a downward arrow.

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.

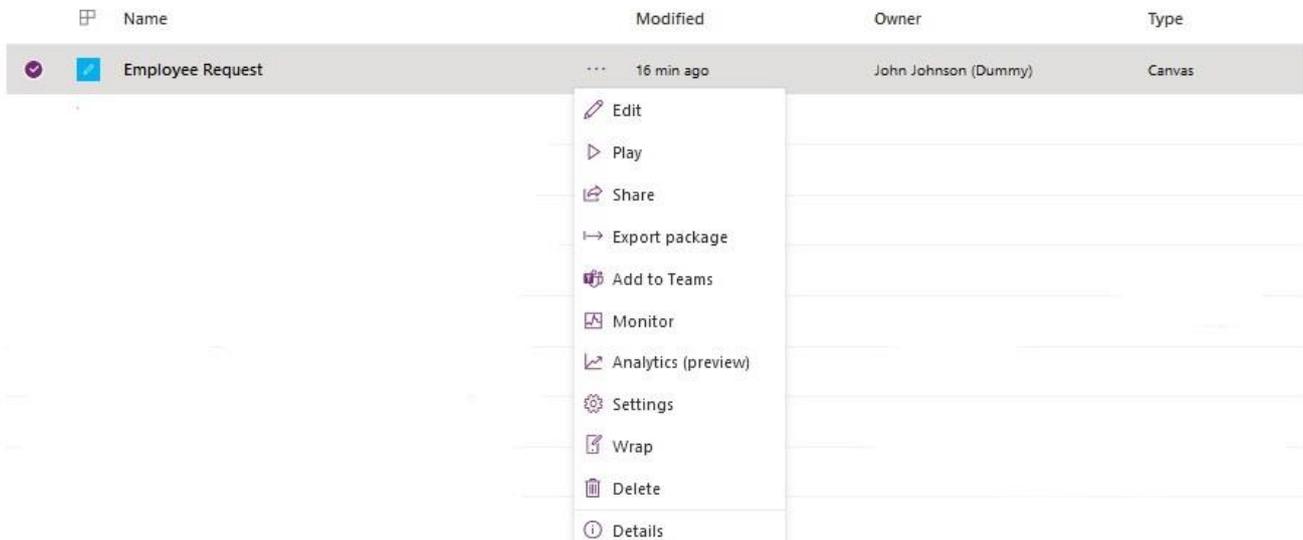
To import that .zip package in the own environment, go to Apps in MS Power Apps / Import canvas apps. The Premium User Plan of Power Automate is required for the OData Connection.

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.

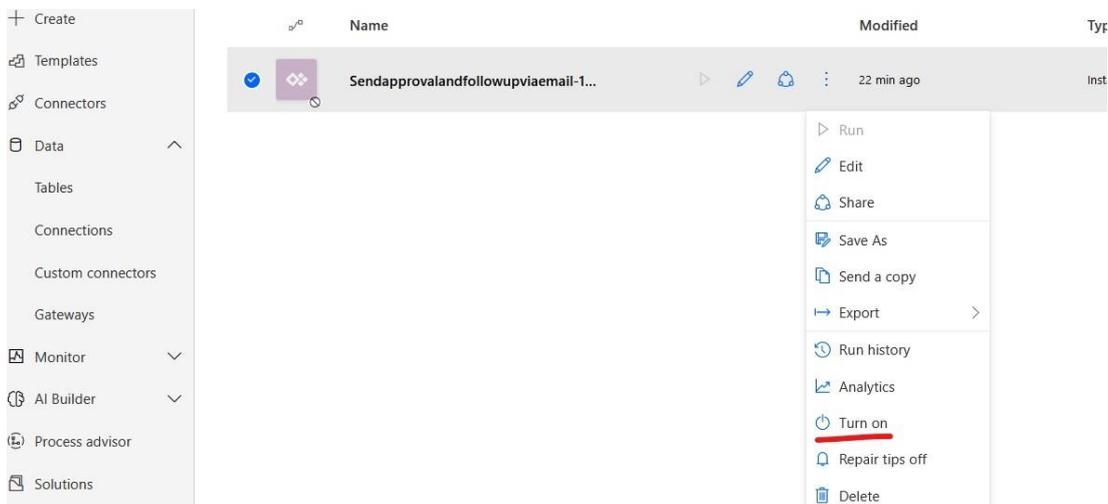
For save creation, especially for the OData Connection, go to (<https://make.powerautomate.com/>) Select "Data/Connections" in the right pane "New connection".

Review Package Content			
Choose your import options.			
NAME	RESOURCE TYPE	IMPORT SETUP	ACTION
Employee Request	App	Create as new	
Related resources			
NAME	RESOURCE TYPE	IMPORT SETUP	ACTION
Sendapprovalandfollowupviaemail-1	Flow	Create as new	
Approvals	Approvals Connection	Select during import	
SampleODataConnection	Jedox OData Hub Connection	Select during import	
john.johnson@jedox.com	Microsoft Teams Connection	Select during import	

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.

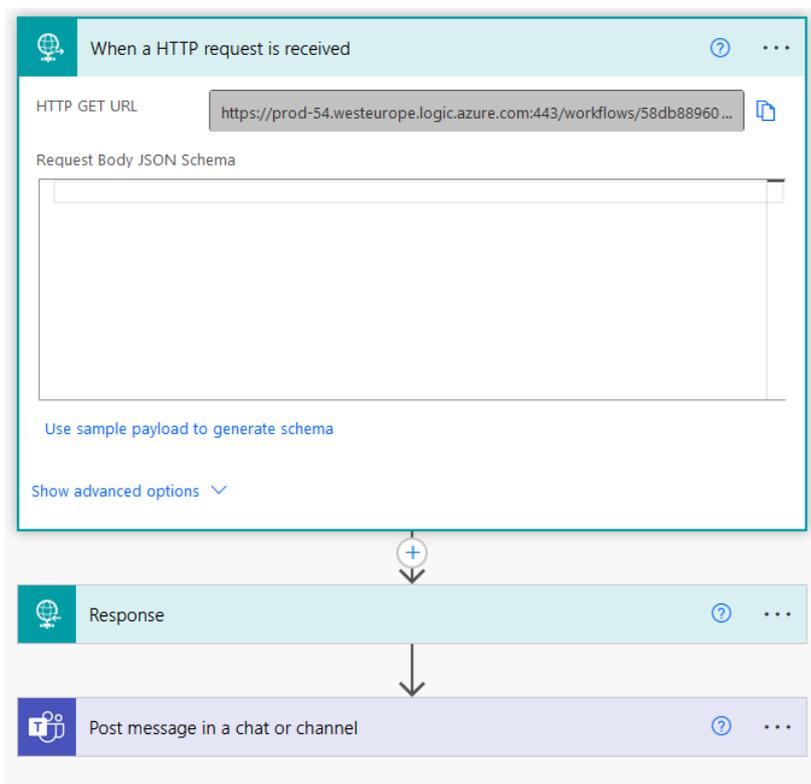


The taken over PowerAutomate Flow has to be turned on in the Flow overview. (<https://make.powerautomate.com/flows>)



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.

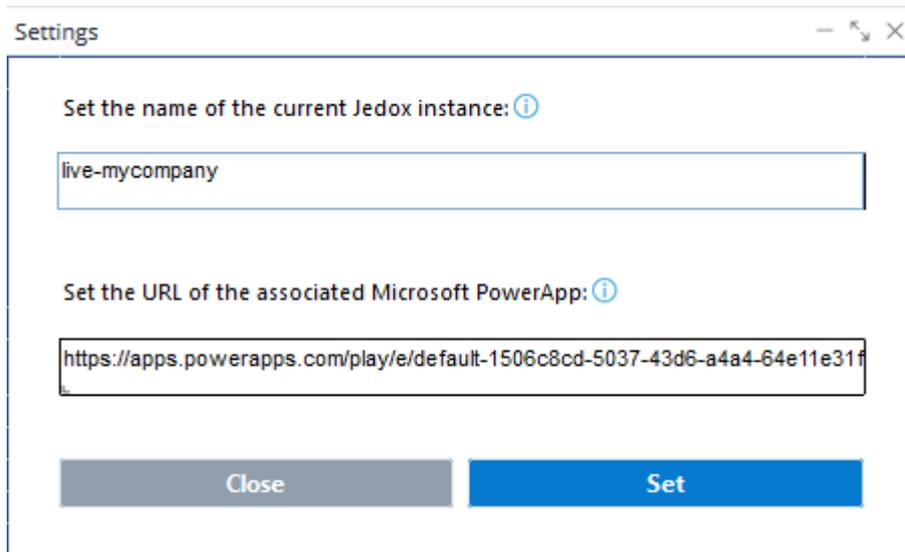
This Flow also must be turned on and the HTTP Link can be copied by editing it.



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.

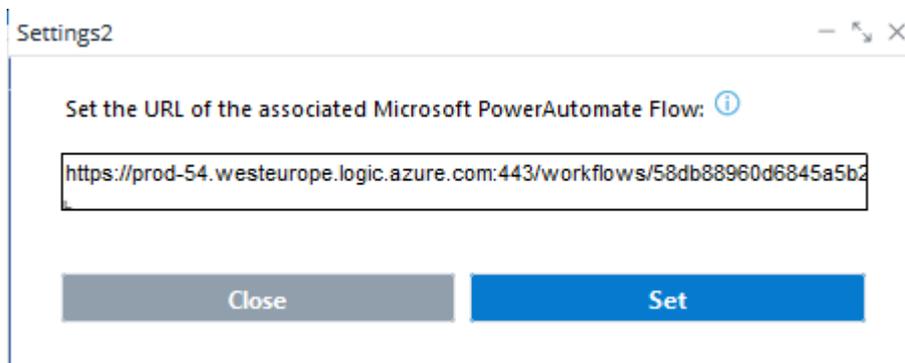
5.1 Setting up the PowerApp and PowerAutomate functionality in the Jedox Model

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



The screenshot shows a dialog box titled "Settings" with a close button in the top right corner. It contains two text input fields. The first field is labeled "Set the name of the current Jedox instance: ⓘ" and contains the text "live-mycompany". The second field is labeled "Set the URL of the associated Microsoft PowerApp: ⓘ" and contains the URL "https://apps.powerapps.com/play/e/default-1506c8cd-5037-43d6-a4a4-64e11e31f". At the bottom of the dialog, there are two buttons: a grey "Close" button and a blue "Set" button.

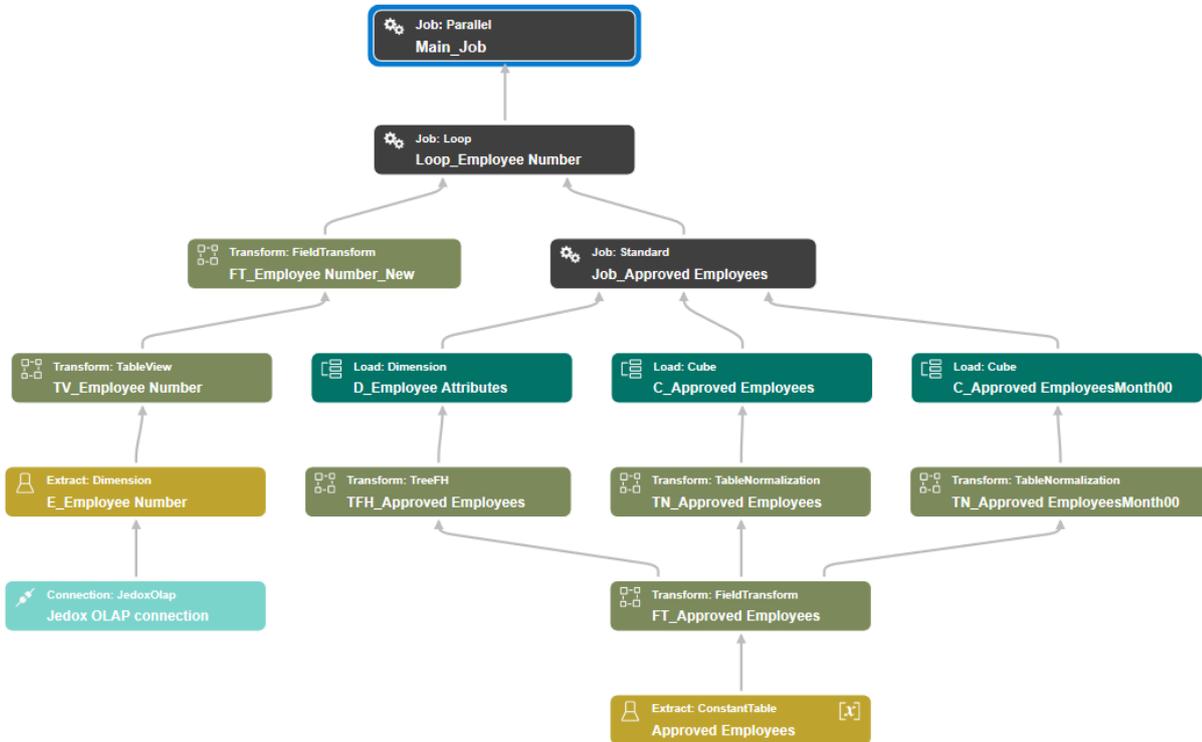
For the Employee Request PowerApp there is a settings box in the Jedox Model where the name of the Jedox instance and the PowerApp URL must be set.



The screenshot shows a dialog box titled "Settings2" with a close button in the top right corner. It contains one text input field labeled "Set the URL of the associated Microsoft PowerAutomate Flow: ⓘ" with the URL "https://prod-54.westeurope.logic.azure.com:443/workflows/58db88960d6845a5b2". At the bottom of the dialog, there are two buttons: a grey "Close" button and a blue "Set" button.

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:

Type	JedoxOlap Connection Link
Name	Jedox OLAP connection
Description	<input type="text"/>
Global connection	localhost
Database	MSFT_Integration
Timeout (in s)	<input type="text"/>
SSL mode	off

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:

Type Dimension Extract

Name E_Employee Number

Description 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.

Connection Jedox OLAP connection

Dimension Employee

Read attributes none

Query filter on dimensions

Filter type	Operator	Value	Mode
accept	like	^[0-9]*\$	onlyBases

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

Type TableView Transform

Name TV_Employee Number

Description 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.

Data source E_Employee Number

Tree format

Filtering

Input	Filter type	Operator	Value

Logical operator

Sorting

Input	Order
level1	desc

Target

Field name	Input

Start line 1

End line 1

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:

The screenshot shows the configuration for a Field Transform. At the top, the Type is 'FieldTransform Transform' and the Name is 'FT_Employee Number_New'. The Description states: 'The 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.'

Below this, the Data source is set to 'TV_Employee Number' and the Tree format is empty. The Functions section contains a table with one entry:

Function name	Type
New Employee Number	Groovy

The Target section contains a table with one entry:

Field name	Input
New Employee Number	New Employee Number

Input for the Groovy function:

The screenshot shows the configuration for the Groovy function. The Name is 'New Employee Number' and the Type is 'Groovy'. The Description is: 'This Groovy function sets the sequential new employee number'. The Parameters section shows a Script with the following code:

```
1 return level1+1
```

The Inputs section contains a table with one entry:

Input	Type	Alias
level1	int	level1

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

- MSFT Integration
 - MSFT Power App Integration
 - Variables
 - Base Salary
 - CostCenter
 - email
 - employeecategory
 - firstname
 - FTE (Change)
 - Headcount (Change)
 - HighestEmployeeNumber
 - lastname
 - LegalEntity
 - manager
 - Month
 - region
 - Senior,Junior
 - Version

Description

Default value

Password

Origin

Create a Constant Table Extract containing all variables:

Type ConstantTable Extract

Name Approved Employees

Description

Constant values

HighestEm...	lastname	firstname	region	employeec...	email	manager	CostCenter	Version	Month	LegalEntity	Headcount ...	FTE (Change)	Base Salary	Senior,Junior
\$(HighestEm...	\$(lastname)	\$(firstname)	\$(region)	\$(employeee...	\$(email)	\$(manager)	\$(CostCenter)	\$(Version)	\$(Month)	\$(LegalEntity)	\$(Headcoun...	\$(FTE (Chan...	\$(Base Salary)	\$(Senior,Jun...

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":

Type: FieldTransform Transform

Name: FT_Approved Employees

Description: The function "FunctionName" is used to concatenate the first and last name. The function "FunctionMonth00" is used to format the month. Inputs "CostCenter" and "Month" are needed.

Data source: Approved Employees

Tree format: [dropdown]

Functions	
Function name	Type
FunctionName	Concatenation
FunctionMonth00	Groovy

Input for the function "FunctionName"/Concatenation:

General		Parameters	
Name	FunctionName	Template	[input]
Type	Concatenation	Delimiter	#space
Description	[input]		
Inputs [up] [down] [add] [remove]			
Input	Type	Alias	
firstname			[input]
lastname			[input]

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

Input for the function „FunctionMonth00"/Groovy:

General		Parameters	
Name	FunctionMonth00	Script	[undo] [redo] [copy] [paste] [search]
Type	Groovy		
Description	[input]		
Inputs [up] [down] [add] [remove]			
Input	Type	Alias	
Month			[input]

```
1 Month.substring(0,4) + "-00"
```



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	employeeecategory
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).

Type: TreeFH Transform
Name: TFH_Approved Employees
Description: Attributes for the dimension Employee. TreeFH for the dimension load to load the new employee number and all corresponding attributes to the dimension Employee.

Data source: FT_Approved Employees

Input	Weight	Attributes
All Employees	□	□
Senior,Junior	□	□
HighestEmployeeNumber	□	Last Name::Last Name,Surname::Surname,Region::Region,Employe...

Name	Type
Name	string
Last Name	string
Surname	string
Region	string
Employee Category	string

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

Type: Dimension Load
Name: D_Employee Attributes
Description: The dimension load is necessary to load the new employee number and all corresponding attributes to the dimension Employee.

Data source: TFH_Approved Employees

Target connection: Jedox OLAP connection

Target dimension: Employee

Elements mode: add

Consolidations mode: add

Attributes mode: add

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:

Type: TableNormalization Transform

Name: TN_Approved Employees

Description: TableNormalization for the cube load with Month as entered for the KPIs "Headcount (Change)" and "FTE (Change)".

Data source: FT_Approved Employees

Tree format:

Target

Field name	Input
Employee	HighestEmployeeNumber
Version	Version
Month	Month
LegalEntity	LegalEntity
Cost Center_Dim	Cost Center_Dim

Normalizer field: Measure

Value field: value

Measures

Measure	Input	Aggregation	Type
Headcount (C...	Headcount (Change)	sum	numeric
FTE (Change)	FTE (Change)	sum	numeric

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

Create a Cube Load:

Type: Cube Load

Name: C_Approved Employees

Description:

Data source: TN_Approved Employees

Target connection: Jedox OLAP connection

Target cube: Personnel Costs

Mode: add

Splash mode: disabled

Handling of missing elements: warning

Default element:

Dimension mapping

Dimension	Input
Employee	Employee
Version	Version
Month	Month
Legal Entity	LegalEntity
Cost Center	Cost Center_Dim
Personnel Costs_measure	Measure

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_Aproved Employees” as Data Source:

Type TableNormalization Transform

Name TN_Aproved EmployeesMonth00

Description TableNormalization for the cube load with Month "-00" for the KPI "Full Time Base Salary (Change)".

Data source FT_Aproved Employees

Tree format

Field name	Input
Employee	HighestEmployeeNumber
Version	Version
Month	Month00
LegalEntity	LegalEntity
Cost Center_Dim	Cost Center_Dim

Normalizer field Measure

Value field value

Measure	Input	Aggregation	Type
Full Time Base S...	Base Salary	sum	numeric

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

Create a Cube Load:

Type Cube Load

Name C_Aproved EmployeesMonth00

Description The cube load loads all new approved employees and their data to the cube Personnel Costs including the KPI "Full Time Base Salary (Change)" with the Month ending with "-00".

Data source TN_Aproved EmployeesMonth00

Target connection Jedox OLAP connection

Target cube Personnel Costs

Mode add

Splash mode disabled

Handling of missing elements warning

Default element

Dimension	Input
Employee	Employee
Version	Version
Month	Month
Legal Entity	LegalEntity
Cost Center	Cost Center_Dim
Personnel Costs_measure	Measure

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:

Type Standard Job

Name Job_Approved Employees

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

Jobs and loads to be executed ⏪ ⏩ + ×

Type	Name
Load	D_Employee Attributes
Load	C_Approved Employees
Load	C_Approved EmployeesMonth00

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

Loop Job Employee Number:

Type Loop Job

Name Loop_Employee Number

Description 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.

Loop source FT_Employee Number_New

Tree format

Execution type job

Execution name Job_Approved Employees

Variable assignment ⏪ ⏩ + ×

Variable	Input
HighestEmployeeNumber	New Employee Number <input type="checkbox"/>

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:

Type Parallel Job

Name Main_Job

Description This Job executes the Loop Job.

Jobs and loads to be executed ⏪ ⏩ + ×

Type	Name	Parallel
Job	Loop_Employee Number	

This Job executes the Loop Job.