

I. Introduction

This tutorial is intended to show the usage of the APEX extension *Flows for APEX* 21.1 as well as explaining the integration of a flow into an APEX application. It also incorporates some good practices. After this tutorial and with some training/coaching, you are ready to build your own process-based applications!

We assume that you have basic understanding about Oracle APEX. If not, we suggest you follow the steps outlined in this blog post: <http://nielsdebr.blogspot.com/2019/08/getting-up-to-speed-with-oracle.html>

Also, this tutorial is not meant to explain the BPMN 2.0 standard. There are good books and online materials already available explaining this.

For further information about Flows for APEX and documentation you may visit the GitHub repository: <https://github.com/mt-ag/apex-flowsforapex>

II. Contents

Exercise No.	Abstract
Exercise #01: Installation of Flows for APEX	Install the newest version of Flows for APEX inside your APEX workspace
Exercise #02: Model your first flow	Model an example flow with the Flows for APEX bpmn modeler to define a simple business process
Exercise #03: Create data model	Create the underlying SQL tables for storing the explicit process data
Exercise #04: Create application	Create the application with all pages, reports, and forms, that are needed within your business process
Exercise #05: Link application to flow	Link the application elements to your flow to have the process steps controlled by your bpmn model
Exercise #06: Define user roles	Define the APEX user roles and authorization schemes
Exercise #07: Testing	Test your application by creating test cases and observe the process using the Flows for APEX monitoring plugin
Exercise #08: Versioning	Deal with changes and multiple versions of your flow
Exercise #09: Process Variables	Use process variables on different places in your business process to enhance working with your flow
Exercise #10: Error handling	Detect and handle errors in your flow by using the Flow Monitor tools

1. Exercise #01: Installation of Flows for APEX

If not already available, apply for a free APEX workspace on apex.oracle.com.

Download the newest version of Flows for APEX from <http://flowsforapex.mt-ag.com>

Install Flows for APEX by importing the file

FLOWSFORAPEX_<apex-Version>_<application_id>_UTF8_<version>.sql

from the *Applications* folder into your APEX workspace. During the installation process select the default options including:

- choose *UTF.8* as file character set
- select *Auto Assign New Application ID (default)*
- install the *supporting objects*

Additionally, to enable the timer functionality for your flow, you need to add a certain DBMS Scheduler job to your workspace. Install it by importing and executing the file **enable_timers.sql** from inside the folder *Enable_Timers* using the **SQL Scripts** function of the **SQL Workshop**.

Notice that you might not have the right to create new jobs inside your own private workspace. In this case, the previous execution of the SQL statement “grant create job to <workspace_schema>”, invoked by the workspace sys, is necessary. Inside your free APEX workspace at apex.oracle.com, this right is already granted.

If you want, you can also install the Sample Process Flow Application on the same way by importing the file

FLOWSFORAPEX<version>_SAMPLE_APP_EN_<apex-Version>_<application_id>_UTF8.sql

from the download folder. This app provides a demonstration for integrating a modelled flow into an APEX application and can help understanding the integration of Flows for APEX.

In this tutorial, we will use the plug-ins for processing the single tasks and displaying the process diagram inside the developed application. The plug-ins are bundled inside the two applications of the distribution.

Open the **plug-ins** page under the *Shared Components* section of the installed *Flows for APEX* or *Sample Process Flow* application and export the four listed plug-ins

- Flows for APEX - Manage Flow Instance
- Flows for APEX - Manage Flow Instance Step
- Flows for APEX - Manage Flow Instance Variables
- Flows for APEX - Viewer

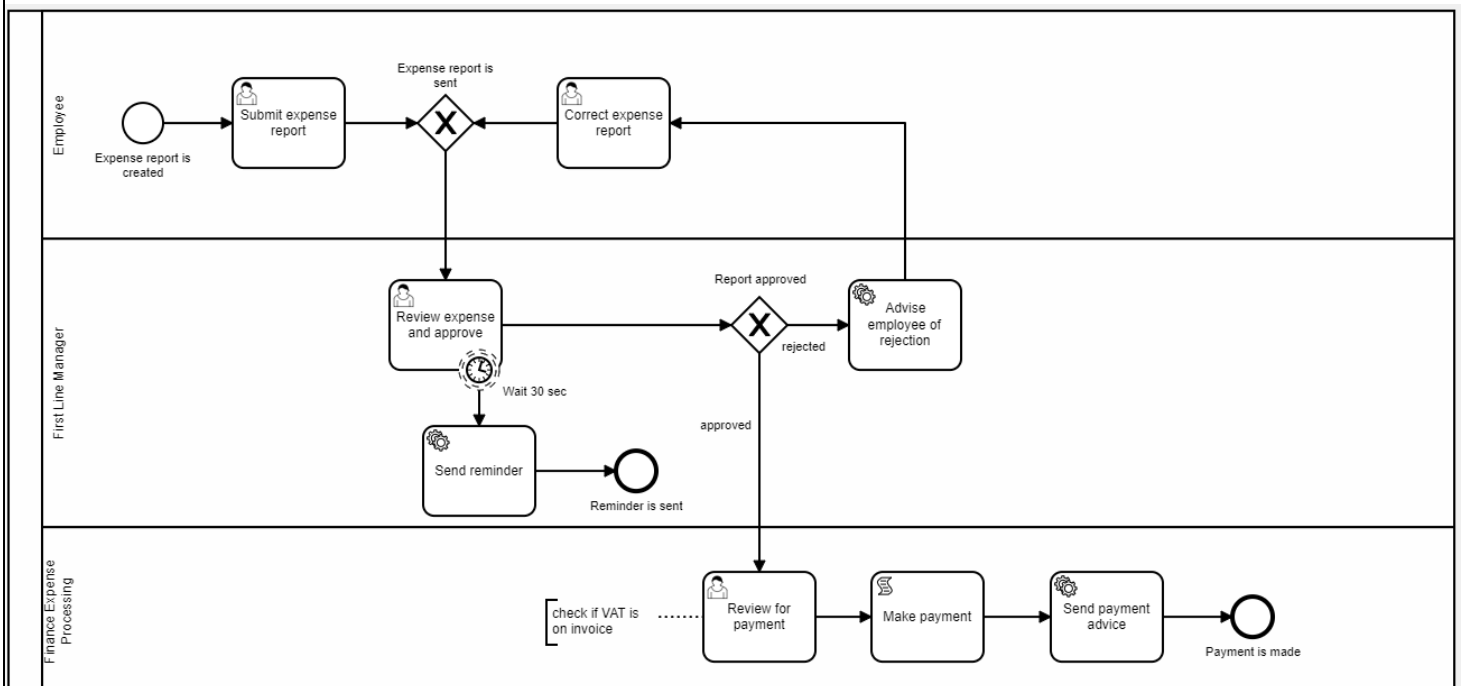
These plug-ins will be imported to our new application later in chapters #04 and #05.

2. Exercise #02: Model your first flow

Start the previously installed *Flows for APEX* application. Switch to the **Flow Management** page (second entry in the navigation menu) and click the **Create Model** button at the top right.

Enter *ExpenseFlow* as Name and *0* as Version. If you want, you can name a *Category*, to help organizing your different flows inside your workspace. Create your new flow by clicking the button **Create**. At the top of the page, you can see the attributes of the current process as well as stats about the process instances. Click the **Modify Diagram** button below to open the flow modeller tool.

Model the depicted BPMN process by combining the specific elements as shown below. Also observe the following hints:



- All BPMN elements can be accessed via the toolbar on the left. Click the needed element and position it on the modelling area. After placing an element, you can access the different types by clicking the **wrench** icon displayed next to it after selection. In this example you need *User*, *Service* and *Script* tasks. Apart from that, this flow contains *Start*, *Boundary* and *End Events*, *Exclusive Gateways* and *Swim Lanes* (modelled by using the **Create Pool/Participant** function on the toolbar).
- After selecting a placed element, additional settings are shown in a separate tab at the right. For this tutorial, set each element's **ID** to its name in underscore notation (e.g., Submit expense report -> Submit_expense_report). These IDs are used to identify the process steps from inside your application. Proceed for all tasks and events as well as the gateway routes. For *non-labelled* sequence or message flows or collaborations, you do not have to assign a specific ID.
- To enable role authorization for multiple lanes in your flow, the lane IDs must match the corresponding user roles static IDs. Since they consist of capital letters, use capital letters for the IDs as well or convert them to uppercase in your views (as done

later in Exercise 03). *Note: for large flows, you might want to avoid the usage of lanes to preserve readability.*

- The shown *Timer Boundary* Event can be modelled by dragging a boundary event to the border of a task and changing its type afterwards. Set the **Timer Definition Type** to *Duration* and enter *PT30S* for the **Timer Definition** to define a 30 second duration using the ISO 8601 syntax. The source of the depicted arrow pointing to the *Send Reminder* task must be set to that Timer Boundary Event.
- The **Text Annotation** as show at the *Review for payment* task can be used, to give extensive information about elements in the process. Alternatively, you can use the **Element Documentation** field inside the element's settings tab to provide a reference, for example a documentation page. All those documentation elements have no further usage inside an end user application depending on that flow.

When done with modelling, click the **Apply Changes** button to save your process.

First, we need to create a **table** to store the relevant user data during the execution of a single process instance. Open the **SQL Commands** page inside the **SQL Workshop** of your workspace and execute the following statement:

In addition, we create two page-specific **views**, that we will use for our business application later. Create those views by executing the following statements:

```
create or replace force view expe_p0002_vw
as
select expe.expe_id
       , expe.expe_name
       , expe.expe_value
       , expe.expe_status
       , expe.expe_created_by
from expenses expe
```

4. Exercise #04: Create application

In this chapter we build the application for executing the defined business process.

Start by creating a new APEX application named *ExpenseApp*, keeping all the default settings.

We will deal with the automatically created page 1 later and start with the first additional page. For that we need a form page, that serves as entry point to the process by creating a new expense report. Create a new modal dialog page of the type **Form** called *Create Expense Report*. Select the default *Navigation Preference*, so no menu entry will be created. Choose the view **EXPE_P0002_VW** as *Data Source* and select **EXPE_ID** as *Primary Key Column*.

After creation, change the type of the items

- **P2_EXPE_STATUS**
- **P2_EXPE_CREATED_BY**

to *hidden* since they are not to be manually set by the end user.

Rename the labels of the displayed items

P2_EXPE_NAME and **P2_EXPE_VALUE** to *Report name* and *Value*.

Next create a *Computation* on the item

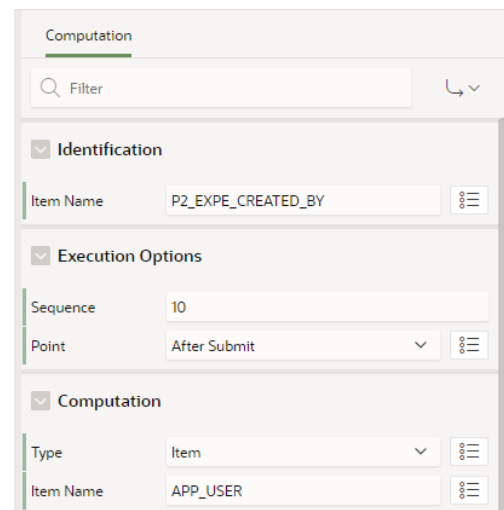
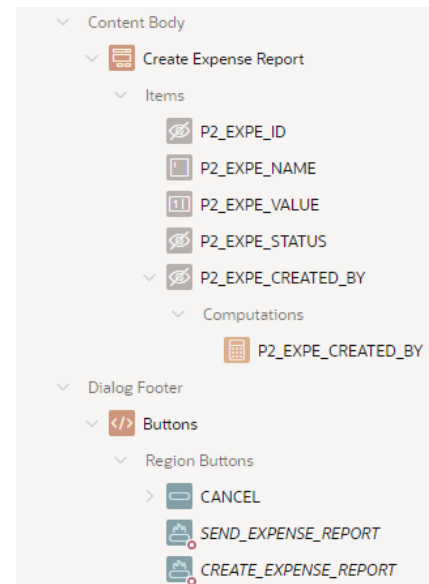
P2_EXPE_CREATED_BY item to retrieve the current user's name. Use the settings shown on the right.

In the buttons section of the dialog

- delete the button **DELETE**
- rename the button **SAVE** to *SEND_EXPENSE_REPORT*
- rename the button **CREATE** to *CREATE_EXPENSE_REPORT*.

The code for processing the submitted forms is added later in chapter #05.

Save this page and **go back to Page 1**.



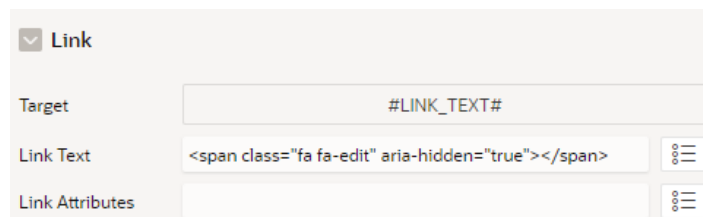
This page will contain a report showing all current process instances for a specific user and provide the links to the next page respective to the following user tasks.

On page 1, move the automatically created region *ExpenseApp* to the **Content Body** of the page and change its type to **Interactive Report**. Rename the region to *Expense Reports* and change the source to the previously created view **EXPE_P0001_VW**. Additionally, add the following *Where Clause*:

```
user_name = :APP_USER
```

This ensures, that the report shows only process instances inside the accessible lanes for a specific user.

Change the type of the columns **EXPE_ID**, **SBFL_PRCS_ID** and **USER_NAME** to *Hidden Column* and the type of the column **LINK_TEXT** to *Link*. Enter the following values under the *Link* section (**target type** set to *URL*):



Link Text: ``

Rename all the displayed column's headings accordingly to their content (*Report name*, *Value*, *Status*, *Created by* and *Link*).

Create a button named and labelled **Create** in this region and set its position to *Right of Interactive Report Search Bar*. Change its **behaviour** to *Redirect to Page in this Application* and set the recently created Page 2 as **target**.

Add a **Dynamic Action** which is executed when the *Dialog Closed* event occurs and use it to refresh the region *Expense Reports*.

Finally save your changes to this page.

In addition to the previous form to create and modify expense reports, we need another form for the approving and reviewing steps of the process.

Create another modal dialog page of the type **Form** called *Review Expense Report*. Again, do not create a navigation menu entry and choose the view **EXPE_P0002_VW** as *Data Source* as well as the column **EXPE_ID** as *Primary Key Column*.

After creation, change the type of the items

- **P3_EXPE_NAME**
- **P3_EXPE_VALUE**
- **P3_EXPE_CREATED_BY**

to *Display Only* and arrange them in a row.

The type of **P3_EXPE_STATUS** should be set to *Select List* to ensure process conform input values. Under the *List of Values* section insert the following input static values:

- Display: approve, Return: approved
- Display: reject, Return: rejected

The return values should match with the corresponding gateway routes in the modelled process.

Add a condition for the **Read Only** attribute of the item *P3_EXPE_STATUS* using the following settings:

Type	Item	Value
Item = Value	P3_EXPE_STATUS	approved

Like on the previous pages, rename the form field labels to *Report name*, *Value*, *Created by* and *Status*.

In the buttons section of the dialog, delete the **DELETE** Button and rename the **SAVE** button to *REVIEW_EXPENSE*. Enter the **Server-side Condition** shown on the right.

Type	Item	Value
Item != Value	P3_EXPE_STATUS	approved

Rename the **CREATE** button to *REVIEW_PAYMENT* and set the **Database Action** field under the **behaviour** section to *SQL UPDATE action*. Enter the displayed **Server-side Condition**.

Type	Item	Value
Item = Value	P3_EXPE_STATUS	approved

Save the changes to this page and start with adding the processing elements for the created forms in the next section.

5. Exercise #05: Link application to flow

Prerequisites

To control the modelled flow and its single tasks from within your application, you need to be able to reference the current process ID and subflow ID. Create two Application Items **PROCESS_ID** and **SUBFLOW_ID** under *Shared Components > Application Items* so you can use the items on all your pages. Be sure to choose *Unrestricted* or *Checksum Required* for the **Security** setting of the items to allow them being set by the flow engine.

The functions, that shall be executed on the script and service tasks, are bundled inside a custom PL/SQL package. To install that package to your workspace, upload and execute the downloaded file **expense_pkg.sql** inside the **SQL Scripts** section.

Next, import and install the four exported plugins from chapter #01 in the **Plug-ins** section of your application's *Shared Components*. While installing the **Flows for APEX – Manage Flow Instance** plug-in, you can enter a value for the component setting *Global Flow*. Enter the name of the modelled flow **ExpenseFlow** to set the needed reference.

Monitor process instance

To be able to monitor the current state of a specific flow instance, the viewer plug-in can be used to embed a diagram view inside the application.

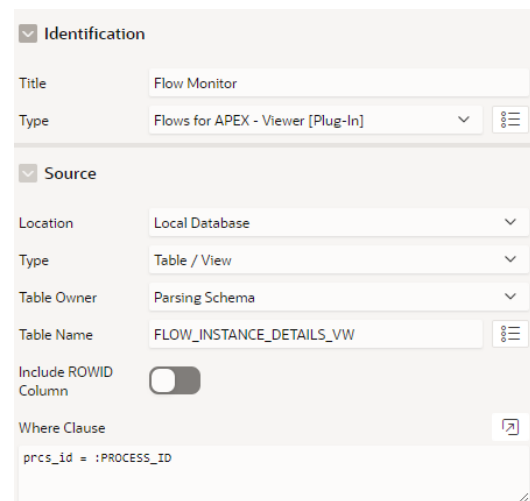
Create a new empty Modal Dialog page named *Flow Monitor*. Add a new region to this page named equally and choose the installed **Flows for APEX - Viewer** plug-in as the region's *type*. Select the Flows for APEX specific view **FLOW_INSTANCE_DETAILS_VW** as the region's *source* and enter the following *Where Clause*, so the process ID application item is used to identify the current process instance:

```
prcs_id = :PROCESS_ID
```

Under the attributes tab of the plugin enter the values displayed on the right.

Finally enable the *Stretch to fit window* setting under the page's **Template Options**, so the diagram is shown in an appropriate size.

Save your changes to this page and **return to page 1**.



Identification

Title: Flow Monitor

Type: Flows for APEX - Viewer [Plug-In]

Source

Location: Local Database

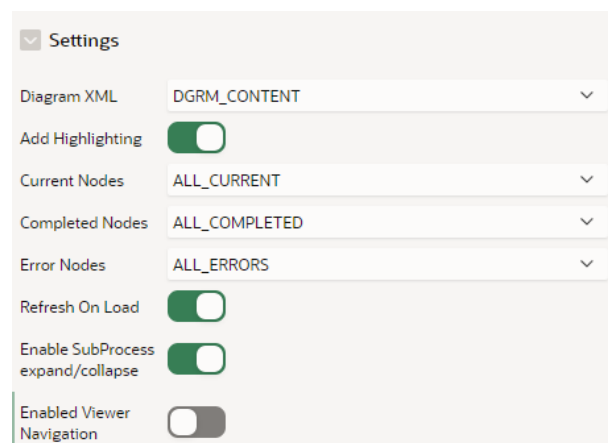
Type: Table / View

Table Owner: Parsing Schema

Table Name: FLOW_INSTANCE_DETAILS_VW

Include ROWID Column: ☐

Where Clause: prcs_id = :PROCESS_ID



Settings

Diagram XML: DGRM_CONTENT

Add Highlighting: ☒

Current Nodes: ALL_CURRENT

Completed Nodes: ALL_COMPLETED

Error Nodes: ALL_ERRORS

Refresh On Load: ☒

Enable SubProcess expand/collapse: ☒

Enabled Viewer Navigation: ☐

Now we want to add a link to the previously created dialog page inside the report. For that we use the column **SBFL_PRCs_ID** which we set to *Hidden Column* before. Change its type to *Link*, enter *Flow Monitor* as the column's heading and set the following options inside the **Target** dialog:

Link Builder - Target

Target

Type: Page in this application

Page: 4

Set Items

Name	Value
PROCESS_ID	#SBFL_PRCs_ID#

Clear / Reset

Clear Cache:

Action: **None** Clear Regions Reset Regions Reset Pagination

Cancel Clear OK

Additionally, instead of displaying the process ID, add the following **Link Text**:
``

Link

Target: Page 4

Link Text: ``

Link Attributes:

OK Cancel

Finally save your changes to Page 1.

Processing forms

Switch back to the first created form **page 2** and open the processing tab. Under the *Processing* section you can find the automatically generated process for storing the expense report input data to the region source. Here we must add the flow-specific processes to control the single tasks inside our modelled flow.

Create a new process named **Flow – Create and Start** of the type *Flows for APEX – Manage Flow Instance*, the previously installed plug-in, and enter the settings show on the right. This provides a correct creation and initialization of the referenced flow.

For clearly identifiable process instances, you can use a combination of static text and Item Values as the *Instance Name*:

Expense Report -
&P2_EXPE_CREATED_BY. -
&P2_EXPE_NAME.

Change the **Server-side Condition** to *When Button Pressed* and choose the button *CREATE_EXPENSE_REPORT*. Ensure the correct execution order by putting the process element below the *Process Form* element.

The screenshot shows the configuration form for a process named 'Flow - Create and Start'. It is categorized under 'Identification' and 'Settings'. The 'Type' is set to 'Flows for APEX - Manage Flow Instance [Plug-I]'. The 'Action' is 'Create and Start'. The 'Flow Instance Name' is 'Expense Report - &P2_EXPE_CREATED_BY. - &P2_EXPE_NAME.'. The 'Select Flow using' is 'Component Setting'. The 'Flow (Diagram) selection based on' is 'Name'. The 'Set Business Reference' is 'P2_EXPE_ID'. The 'Return Instance ID into' is 'PROCESS_ID'. The 'Set Process Variables?' is 'No Process Variables'.

Create a second process named **Flow - Complete Step** and put it below the previous one. This process is used for moving one step forward inside the flow. Change its type to the previously installed *Flows for APEX – Manage Flow Instance Step* plugin and set the displayed values under the settings tab.

Like the first process, set the **Server-side Condition** to *When Button Pressed* and enter the second button's name *SEND_EXPENSE_REPORT*.

Remove the **Server-side Condition** from the *Close Dialog* process so it is always executed at the end.

Save this page and **switch to page 3**.

The screenshot shows the configuration form for a process named 'Flow - Complete Step'. It is categorized under 'Identification' and 'Settings'. The 'Type' is set to 'Flows for APEX - Manage Flow Instanc'. The 'Action' is 'Complete Step'. The 'Flow Instance info' is 'In Page Items'. The 'Process ID Item' is 'PROCESS_ID'. The 'Subflow ID item' is 'SUBFLOW_ID'. The 'Set Gateway Routing?' and 'Auto-Branching?' are both set to 'Off'. The 'Return Flow Instance and Subflow ID' is empty.

For this form page, two separate *Complete Step* processes are needed. Add the process elements using the *Flows for APEX – Manage Flow Instance Step* plugin like before and use the following settings:

Identification

Name

Flow - Complete Step with Gateway

Type

Flows for APEX - Manage Flow Instance Step [PI]

Settings

Action

Complete Step

Flow Instance info

In Page Items

Process ID Item

PROCESS_ID

Subflow ID item

SUBFLOW_ID

Set Gateway Routing?

☒

Gateway ID

Report_approved

Route ID

&P3_EXPE_STATUS.

Auto-Branching?

☐

Return Flow Instance and Subflow ID

Execution Options

Success Message

Error

Server-side Condition

When Button Pressed

REVIEW_EXPENSE

Type

- Select -

Identification

Name

Flow - Complete Step

Type

Flows for APEX - Manage Flow Instance Step [PI]

Settings

Action

Complete Step

Flow Instance info

In Page Items

Process ID Item

PROCESS_ID

Subflow ID item

SUBFLOW_ID

Set Gateway Routing?

☐

Auto-Branching?

☐

Return Flow Instance and Subflow ID

Execution Options

Success Message

Error

Server-side Condition

When Button Pressed

REVIEW_PAYMENT

Type

- Select -

Again, remove the **Server-side Condition** from the *Dialog Closed* process and save your changes to the page.

bpmn modeller settings

Inside the bpmn modeller you can use additional information and settings to call APEX pages or run scripts inside your application. Open your modelled *ExpenseFlow* inside the **Flow Management** section of the Flows for APEX application (using the *Show Details* row action) and click the **Modify Diagram** button.

User Tasks

User Tasks are designed to open a specific APEX page when reached inside the process flow by providing a generated link to that page. Click the first user task **Submit expense report** and open the *APEX* tab in the settings panel on the right.

Here you can specify the Application where your Diagram is used as well as the page to execute the User Task. Be sure to use the correct application ID of the *ExpenseApp* here. Alternatively, you can leave the field blank to use the currently opened application. *Note: Sometimes the usage of the alias causes conversion errors when querying the generated link. We recommend using the application ID or leaving it empty in this case.*

By setting the **Page Item** and **Item Values**, references to the current process ID and subflow ID are granted. The ID to the explicit expense entry has been set as the business reference variable during the creation of the process instance inside the plugin and can be accessed via the substitution string *&F4A\$BUSINESS_REF.*

Enter the following values to create a connection to page 2 (enter your specific application id):

- Application: <app_id>
- Page: 2
- Page Items: P2_EXPE_ID,PROCESS_ID,SUBFLOW_ID
- Item Values: &F4A\$BUSINESS_REF.,&F4A\$PROCESS_ID.,&F4A\$SUBFLOW_ID.

Similarly change the settings for all existing User Tasks:

Correct expense report

- Application: <app_id>
- Page: 2
- Page Items: P2_EXPE_ID,PROCESS_ID,SUBFLOW_ID
- Item Values: &F4A\$BUSINESS_REF.,&F4A\$PROCESS_ID.,&F4A\$SUBFLOW_ID.

Review expense and approve

- Application: <app_id>
- Page: 3
- Page Items: P3_EXPE_ID,PROCESS_ID,SUBFLOW_ID
- Item Values: &F4A\$BUSINESS_REF.,&F4A\$PROCESS_ID.,&F4A\$SUBFLOW_ID.

Review for payment

- Application: <app_id>
- Page: 3
- Page Items: P3_EXPE_ID,PROCESS_ID,SUBFLOW_ID
- Item Values: &F4A\$BUSINESS_REF.,&F4A\$PROCESS_ID.,&F4A\$SUBFLOW_ID.

Script Tasks / Service Tasks

On Script and Service Tasks you can trigger the execution of PL/SQL code by invoking functions and procedures of custom packages. Again, the settings can be entered under the *APEX* tab after clicking a task in the modeller.

Leave the setting for Engine on *No* and use the following procedure calls as **PL/SQL code** for the Script and Services Tasks in the model:

- Send reminder: *expense_pkg.send_reminder;*
- Advise employee of rejection:
expense_pkg.advise_employee;
- Make payment: *expense_pkg.make_payment;*
- Send payment advice: *expense_pkg.finish_expense;*

Send_reminder

General APEX Variables

Service Task

Engine

No

Use APEX_EXEC

PL/SQL Code

expense_pkg.send_reminder;

Enter the PL/SQL code to be executed.

Finally save your changes to the flow diagram by clicking the *Save Flow* button.

6. Exercise #06: Define user roles

The mapping of the three lanes inside the modelled flow can be realised by using the APEX user role mechanism. By that you can ensure that each user gets access only to the current steps inside its permitted area.

Open the **administration** page inside your workspace and click **Manage Users and Groups**. Create three new users as end users using an accessible mail address of yours and free to choose names and passwords. To maintain consistency when assigning the roles, you can name them accordingly to the lanes inside the modelled flow or by using shortcuts (e.g., EMP, FLM, FEP).

Return to the *ExpenseApp* AppBuilder and go to *Shared Components > Application Access Control*. Create three new **Roles**, named identically to the lanes in the model:

- Employee (Static ID: EMPLOYEE)
- First Line Manager (Static ID: FIRST_LINE_MANAGER)
- Finance Expense Processing (FINANCE_EXPENSE_PROCESSING)

Then add the user role assignments by adding each role to one of the previously created users.

*Note: By using this way of role management, all users sharing a lane can see the related process instances. So, in this case, each employee can edit all existing expense report entries. This is due to the logic behind lanes in BPMN processes. Additionally, it allows us to combine the employee's and manager's tasks to one single form page since there are no accessibility issues between them. To enable further authentication, you can use the **CREATED_BY** attribute in combination with a strictly implemented hierarchy between employees and managers with proper page and region visibilities as well as the APEX authorization schemes. For bigger and more complex business applications, Flows for APEX allows the reservation of single process tasks to show other users, that an instance is already being processed internally. Thus, concurrent operations on a single process instance can be prevented. Notice that this functionality as well as further authorization techniques are not included as part of this tutorial for the sake of convenience.*

7. Exercise #07: Testing

In this chapter we want to test our application in combination with the modelled flow.

Start by logging in into the **ExpenseApp** with a user from the *Employee* group.

The image shows the ExpenseApp interface. At the top is a blue header bar with the text 'ExpenseApp' on the left and a user profile icon labeled 'mamani' on the right. Below the header is a white navigation bar containing a search icon, a 'Go' button, an 'Actions' dropdown menu, and a 'Create Expense Report' button. The main content area is white and contains the 'Expense report creation' section. This section has a title 'Expense report creation' and a paragraph stating: 'By clicking on the **Create** button, the following Modal Dialog appears.' Below this text is a modal dialog titled 'Create Expense Report' with a close button (X) in the top right corner. The dialog contains two input fields: 'Report name' and 'Value'. At the bottom of the dialog are two buttons: 'Cancel' and 'Create Expense Report'.

ExpenseApp

mamani

Go Actions

Create Expense Report

Expense report creation

By clicking on the **Create** button, the following Modal Dialog appears.

Create Expense Report

Report name

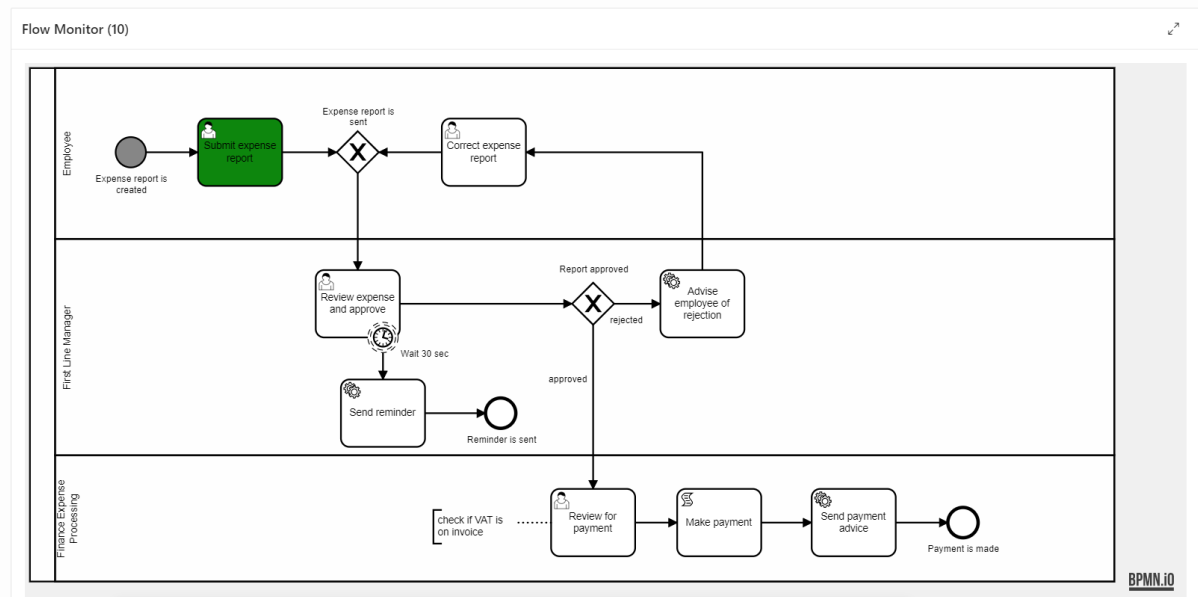
Value

Cancel Create Expense Report

Fill the required input fields and click the button **Create Expense Report** to submit the data and create a new entry.

The diagram of the process instance can be displayed by clicking the *eye icon* in the corresponding row.

The diagram for a newly created process instance will look as follows:

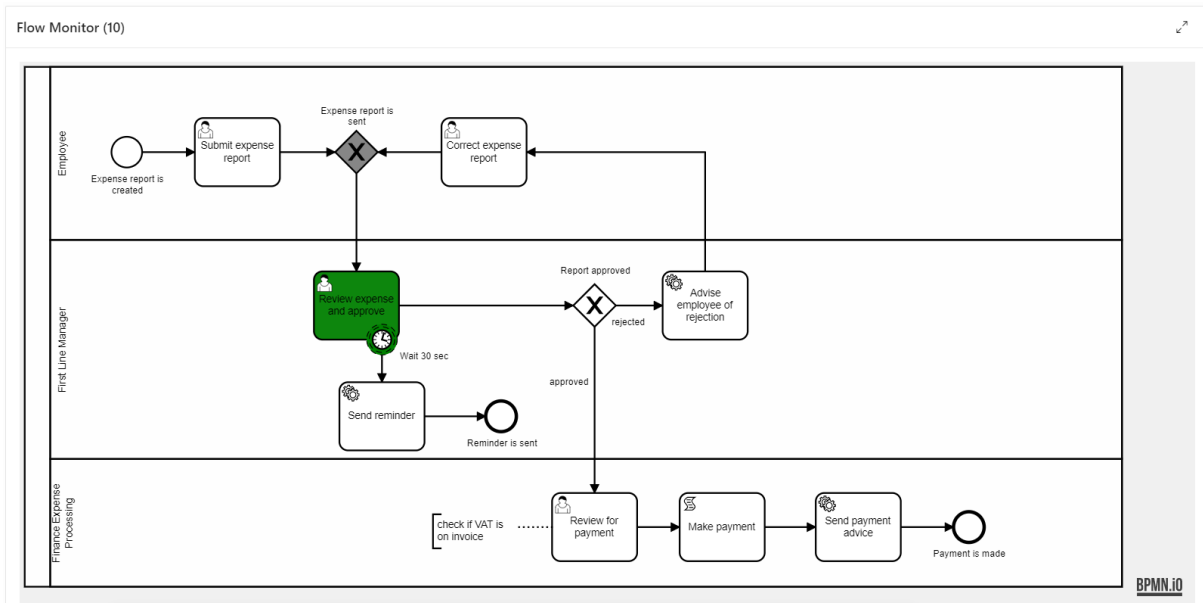


To complete this user task, you need to edit the expense report entry inside the *Expense App* by clicking the *Pencil icon* inside the row. You can keep the entered values and click on **Send Expense Report** to finish this step.

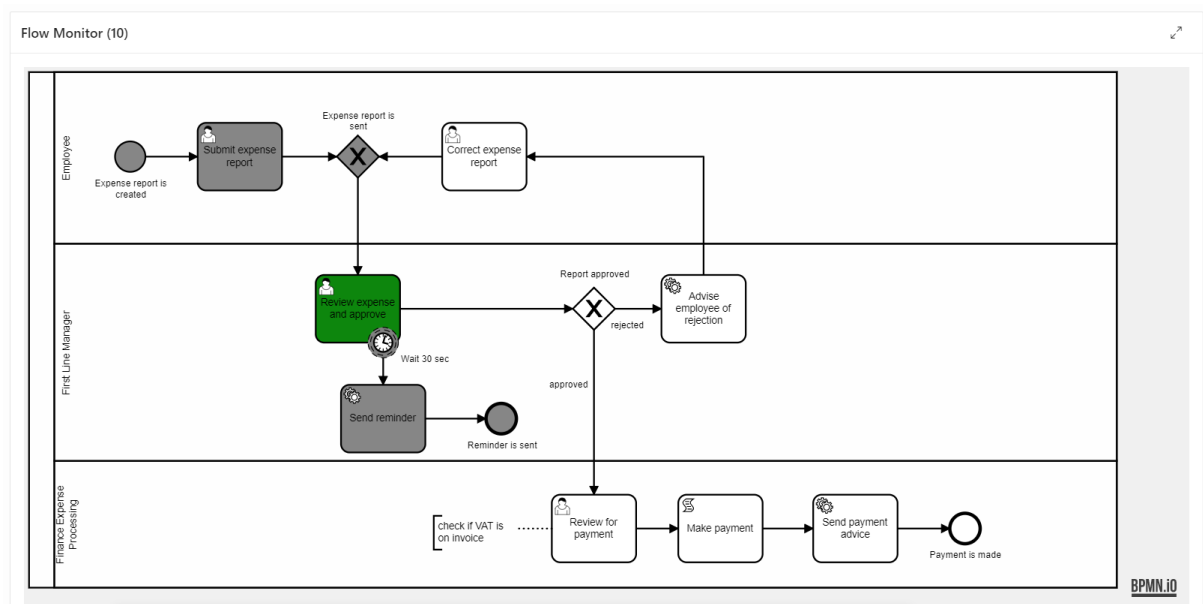
The screenshot shows a web form titled "Create Expense Report" with a close button (X) in the top right corner.

- Report name:** A text input field containing "Test Expense Report".
- Value:** A text input field containing "1000".
- Buttons:** At the bottom, there is a "Cancel" button on the left and a blue "Send Expense Report" button on the right.

You are required to log in as a *First Line Manager* group user now to proceed. Looking on the flow monitor again you will see that the activity for the current instance has been updated. As shown below, a timer has been started.



After the timer's end, you will notice that all *First Line Manager* group users received a reminder email.



Expense report approval

Proceed the flow by clicking the link column again to review the expense report.

Review Expense Report

Report Name
Test Expense Report

Value
1000

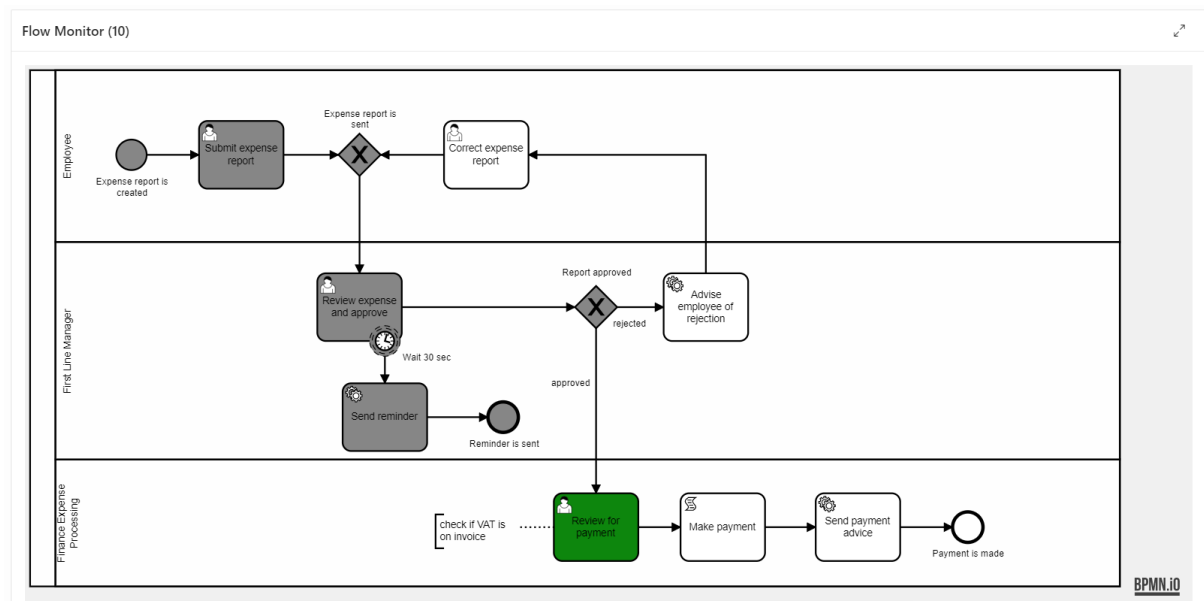
Created By
MAMARI

Status
approve

Cancel

Review Expense

After approving the expense report, the current activity on the Flow Monitor moved into the *Finance Expense Processing Line*.



Expense report payment review

You need to log in with a user from *Finance Expense Processing* group and review the payment as shown below.

Review Expense Report

Report Name
Test Expense Report

Value
1000

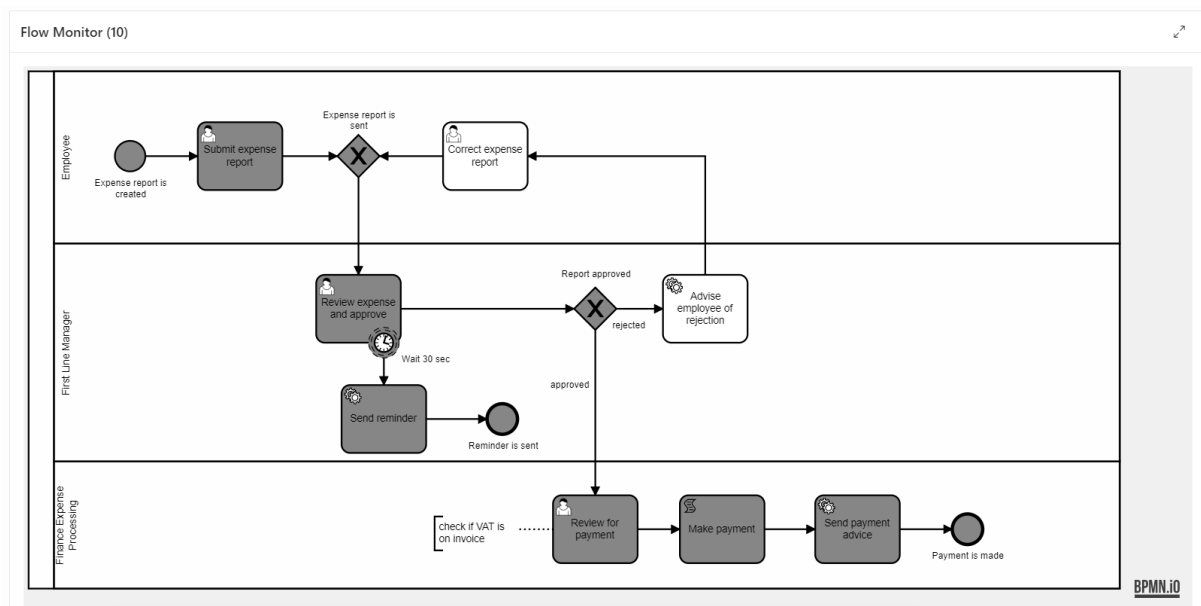
Created By
MAMARI

Status
approve

Cancel

Review Payment

That will end the flow and you will notice that an email was sent to confirm the payment.



Note: By using the view `FLOW_TASK_INBOX_VW` in the page specific view on page 1, only running instances that need to be worked on are displayed. Because of that, you can't see the completed (paid) expense report there. For documentation purposes this could be achieved by using the view `FLOW_INSTANCES_VW`, which includes all the instances and join it to the expense table using the business reference.

Expense report rejection

You can repeat the steps until the review of the expense report and choose to reject the report this time.

Review Expense Report

Report Name
Test Expense Report

Value
1000

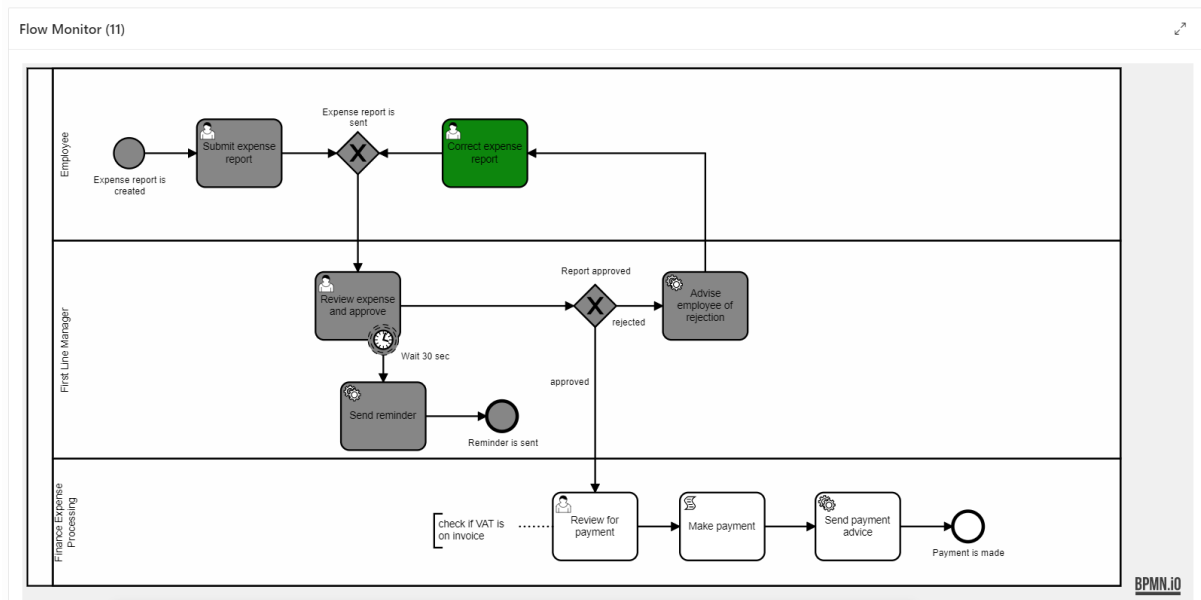
Created By
MAMARI

Status
reject

Cancel

Review Expense

This will bring you into the **Correct expense report** activity after receiving rejection email.



You can correct your report and send it again in to be reviewed by user from the *First Line Manager* group.

8. Exercise #08: Versioning

In Flows for APEX each process instance is based on a diagram name and a version. This allows you to have multiple versions of your flow while ensuring the runnability of the single instances. Additionally, the status attribute is used to mark the current version of your diagram and if changes to the diagram can be expected.

Each newly created diagram will start with the status **draft**. This status is typically used during the modelling phase, where you can edit and test your diagram at any time without changing the version. Once your flow is ready for production, you can set its status to **released**, which makes it immutable. For further changes you would have to create a new version of the diagram, starting with **draft** status again.

Open the *Flows for APEX* application and go to the **attributes** panel of your modelled flow.

Flow Management \

ExpenseFlow - Version 0

Apply Changes

Attributes

Category
Tutorial

Name
ExpenseFlow

Version
0

Status
draft

New Version

Release

Process instances per status

Status	Count
Created	0
Running	0
Completed	0
Terminated	0
Error	0

Click the button **Release** to change the status of this diagram to *released*. The **Modify Diagram** button is now hidden to prevent changes to your diagram.

Open the *Expense App* and start a new process instance by creating an expense report. If you do not specify a diagram version in the Component Settings, always the released version of the diagram will be used. Otherwise, if no *draft version 0* exists, errors might occur.

Now we want to create a new version of the diagram. Go back the *Flow Modeller* and click the button **New Version**. Enter *1* as the version identifier and click **Add Version** to close the dialog. A new version of the diagram has been created. Click the **Edit Flow** button and make some minor but visible change to the diagram, for example change the timer text and definition from 30 to 60 seconds.

To release this new version of the flow, the old version must be set to **deprecated** first. This can be done by clicking the button next to the status field on the attributes panel of the version 0 diagram. Now you can change the status of the new diagram to **released**.

Inside the *Expense App* create another process instance. By comparing the corresponding diagram windows using the viewer plugin you can see that the first entry still uses the deprecated diagram, while the second entry is based on the new version instead.

After all existing instances of a deprecated diagram have finished, you can set the diagram status to **archived** to indicate that it is not used for processing anymore.

9. Exercise #09: Process Variables

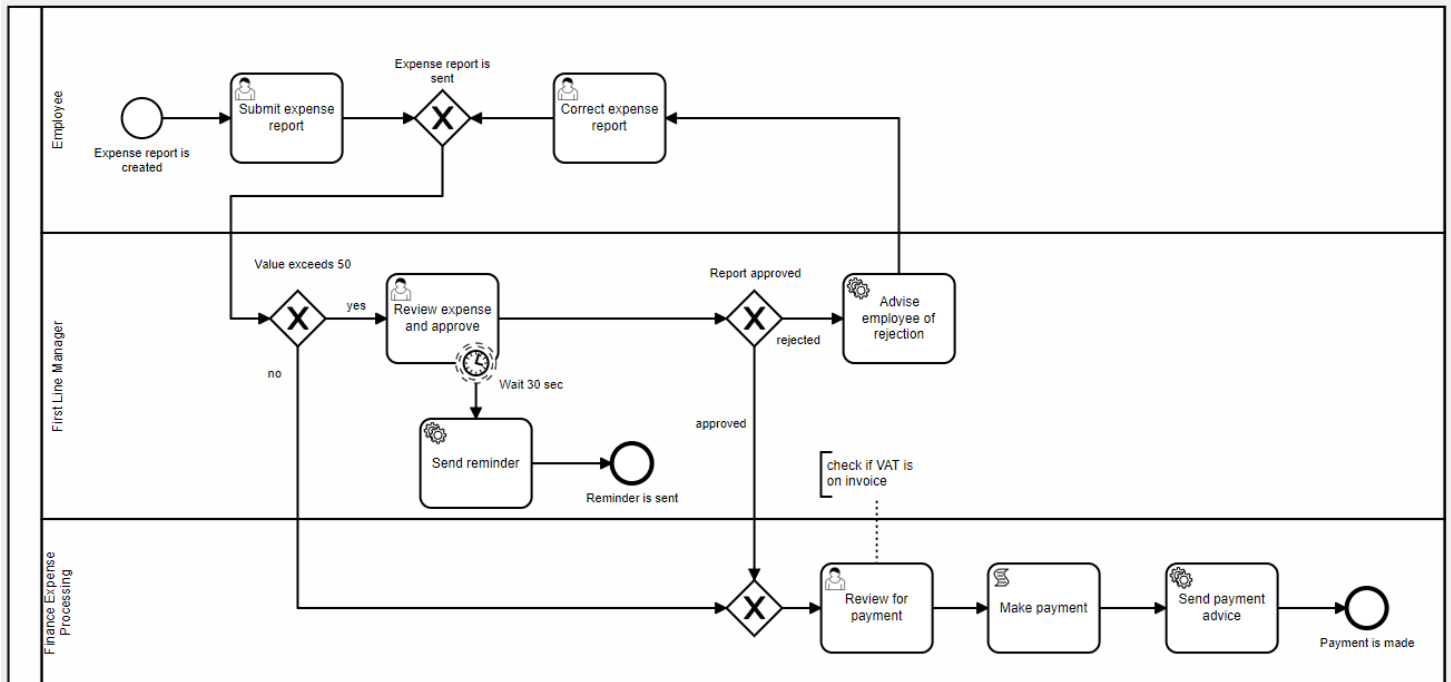
Process variables provide a way to store process relevant information throughout the whole execution of a single instance. They are stored persistently in the database and hence (in contrast to APEX items) can be used without an APEX session.

In this demo, process variables for a flow instance already have been used to store the business reference and the path to choose for the gateway, depending on the user's input. This works automatically by using the pre-configured attributes in the process plugins.

Starting with *Flows for APEX 21.1*, setting process variables is also possible in the flow modeler.

In this chapter we want to modify the approval process in a way, that the First Line Manager only has to approve expenses, when the value exceeds a certain limit.

First, open the modelled flow in the Flow Modeler. Extend the process flow as shown below by adding an additional gateway. Be sure to set the IDs of the gateway and the routes as before by using the underscore notation (e.g., Value exceeds 50 -> Value_exceeds_50).



Select the new gateway and open the *variables* tab on the properties panel on the right.

For opening gateways, you can define variables and their value expressions here, that are set *before* the gateway splits up. For merging gateways, you can set variables *after merging* and for tasks you can set variables both *before* and *after* the task.

Create a new variable expression by clicking the **plus** icon on the upper right and enter the settings shown on the right. In this case we use a package function to determine the correct gateway route. Inside the function we use the package *flow_process_vars* to retrieve a prior set process variable, compare it to a static limit and return the name of the gateway route depending on that.

Save the changes to the model, go back to the App Builder and **open page 2**.

The screenshot shows the 'Value_exceeds_50' gateway configuration in the 'Variables' tab. Under 'Process Variables', there is a 'Before Split' section with a list containing 'Value_exceeds_50:route : return ex'. Below this, the 'Variable Details' section shows 'Name' as 'Value_exceeds_50:route', 'Data Type' as 'Varchar2', and 'Expression Type' as 'Function Body'. The 'Expression' field contains the code 'return expense_pkg.check_limit;', with a note below stating 'PL/SQL Function Body returning a value'.

In the processing section of the page, we now want to add a process to set a variable, that will be used inside the *check_limit* function of the package.

Create a new process and select the plugin *Flows for APEX – Manage Flow Instance Variables* as type. Enter the following settings and position it right after the *Flow – Complete Step* process.

The input value of an expense report will now be saved after creation with a reference to that process instance, so it can be used at the gateway we defined earlier.

Save your changes to the page and go back to application.

The screenshot shows the configuration for a new process. The 'Identification' tab is active, showing 'Name' as 'Flow - Set Variable' and 'Type' as 'Flows for APEX - Manage Flow Instance Variables'. The 'Settings' tab is also visible, showing 'Action' as 'Set', 'Flow Instance info' as 'In Page Items', 'Process ID Item' as 'PROCESS_ID', 'Manage Variable(s) using' as 'APEX item(s)', 'Process Variable(s) Name(s)' as 'VALUE', 'APEX item(s)' as 'P2_EXPE_VALUE', and 'Return Flow Instance ID into' as an empty field.

If you now create an expense report with a value below 50, the review step of the First Line Manager will be skipped, and the process continues at *Review for payment*.

10. Exercise #10: Error handling

In Flows for APEX v21.1 the Engine is able to detect errors while processing a modelled flow and display the error information to the user. Using the Flow Monitor of the Flows for APEX Engine App an admin can then analyse, where the error occurred and initiate the needed error handling routines or fix the error directly by using the Flow Modeler. After that, a single step can be restarted so that the Flow can continue properly.

In this exercise we will manually enter an incorrect PL/SQL code on a Script Task and fix it afterwards, to test out the logging and restart functionality.

Open the Flow Modeler and edit your modelled flow. Select the *Make payment* Script task and change the PL/SQL code to something, that will not execute, e.g.

`expense_pkg.make_payments;`

Save your changes and open the *Expense App*. Create a new expense report and move forward in the process until the task *Review for Payment* should have been completed. Now open the Flows for APEX Engine App and go to the **Flow Monitor** page (third entry in the navigation menu).

The *Flow Instances* report contains an overview of all the current instances. You will see that the *Status* column of your current instance is marked as **error** and has a red background. Click on the *Flow Name* of this instance to open the diagram in the viewer. You can see that the *Make payment* task is marked red as well.

Flow Instances

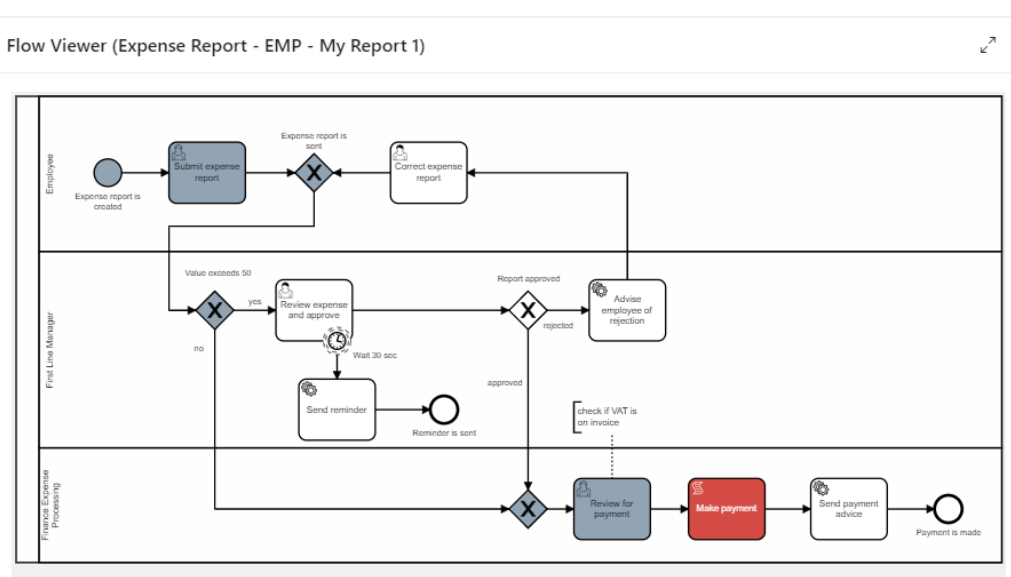
Q v Go Rows 5 Actions Reset

Flow Category

Flow Category :

	Quick Action	Flow Name	Business Reference	Name	Version	Status	Creation Date	Last Update
	Details	Expense Report - EMP - My Report 1	1	ExpenseFlow	0	error	09.09.2021 11:31:01	09.09.2021 11:33:20

1 - 1



If you click on the task, you can see detailed information like the PL/SQL code that gets executed on this Script Task. Close this dialog and click on the **Details** link in the *Quick Action* column for this instance. On this page you can see more information about the instance as well as the current existing subflows and process variables. Click on the **Audit** button in the top right corner. This will open the event log for the current instance, including all the logged information on instance, subflow and step level. In the *error stack* column, you can now see, what error has happened in the process.

Expense Report - EMP - My Report 1

Instance Events Current Steps Completed Steps Variable History				
<input type="text"/> <input type="button" value="Go"/> <input type="button" value="Actions"/>				
User	Timestamp	Event	Comment	Error Stack
FEP	09.09.2021 11:33:20	error	Process 1: ScriptTask Make_payment failed due to PL/SQL error - see event log.	ORA-06510: PL/SQL: unhandled user-defined exception ORA-06512: at "WKSP_DAMTHORFLOWS.FLOW_PLSQL_RUNNER_PKG", line 124 ORA-06550: line 2, column 13: PLS-00302: component 'MAKE_PAYMENTS' must be declared ORA-06550: line 2, column 1: PL/SQL: Statement ignored ORA-06512: at "WKSP_DAMTHORFLOWS.FLOW_PLSQL_RUNNER_PKG", line 39 ORA-06512: at "WKSP_DAMTHORFLOWS.FLOW_PLSQL_RUNNER_PKG", line 95
EMP	09.09.2021 11:31:01	started		
EMP	09.09.2021 11:31:01	created		

1 - 3

Close this dialog and fix the error in the **Flow Modeler** by changing the PL/SQL statement back to the correct spelling. Save the diagram, go back to the **Flow Monitor**, and open the *Details* page of the instance again. Now we want to **restart** the *Review Payment* task so the process can continue. This can be done by clicking the *Quick Action* in the corresponding row of the *Subflow* report on the left. Enter an optional comment on the restart action and **confirm** the dialog. You can now see that the task has been successfully executed and the process could be finished.

Flow Viewer

