Appian
Academy

Appian Step-by-Step #8

Exercise to Accompany
Process Modeling 101: Automate Your Business Processes

The Appian Step-by-Step series consists of 12 exercises that accompany the courses in the Appian Developer learning path. Exercises build upon each other. Complete exercises in order and keep the app and all objects until you are done with the project.

Exercise 8: Process Modeling 101, Part 1

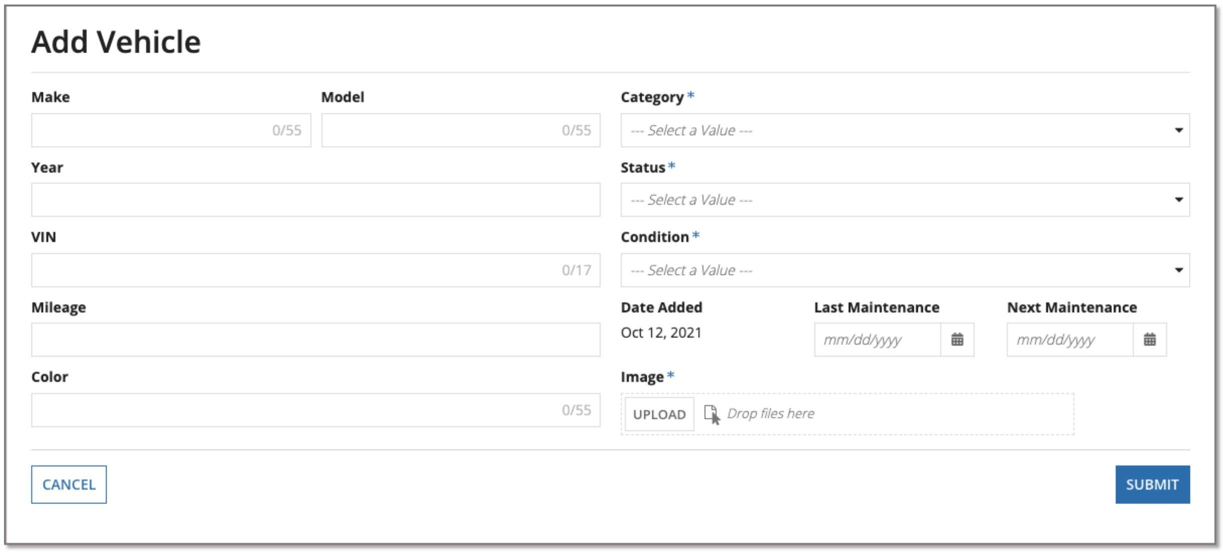
In this exercise, you will edit a process model for adding new vehicles that was generated for you, along with other objects, when you created a record action. You will edit both the process model and related interface to ensure that they function correctly and meet application requirements. These requirements are:

- The process allows the Registrar to fill out the Add New Vehicle form
- The process allows the Supervisor to approve the vehicle
- If the vehicle is approved, the process writes all vehicle details to the database

Before you proceed to extend your auto-generated process model, create the interfaces that this process model will need.

Edit the Add Vehicle Form

In this section, you will edit the auto-generated AX Add Vehicle interface that the AX New Vehicle process model uses to kick off the addition of the vehicle to the fleet. Your goal is to improve the overall UX of this form and to update a couple of fields, so that they display the information correctly. After edits, this interface will look like the image below.



Complete the steps below to edit the Add Vehicle form.

Update Fields and Labels

1. Open **AX_AddVehicle**, and update the form title to **Add Vehicle**.
2. Click the **record** rule input in the top right corner, and rename it to **vehicle**.
3. Delete the following fields: Vehicle Status, Vehicle Added By, Vehicle Last Modified By, Vehicle Condition, Vehicle Category, Vehicle Image, and Vehicle Last Modified Date.

4. Remove redundant labels. All default labels in this form contain **"vehicle,"** which is unnecessary in the Add Vehicle form. Also, remove the word **"date"** from the labels for the Last Maintenance and Next Maintenance fields.

5. Drag and drop a **side-by-side** layout at the top of column one. Then drop the **Make** and **Model** fields inside of the side-by-side layout. The side-by-side layout will ensure that the related Make and Model fields stay next to each other on a narrower screen.

   Keep in mind that columns layouts are flattened into a single column on mobile devices. If you need certain related fields to stay together, use side-by-side layouts.

6. Add another **side-by-side** layout in column two, and place the **Date Added**, **Last Maintenance**, and **Next Maintenance** inside of it. For reference, use the image at the top of this section.

7. Next, add a validation to the **Next Maintenance** field. Let's create a validation so that the Next Maintenance date has to be after the Last Maintenance date. Click **Next Maintenance**, and scroll down to **Validations** in the **Component Configuration** pane.

8. Click **List of Text** or the **Expression Editor** icon. Click **Add Item**.

9. In the **Expression Editor**, type the following expression:

   ```
   if(ri!vehicle.vehicleNextMaintenanceDate <
   ri!vehicle.vehicleLastMaintenanceDate, "The next maintenance
   date must be after the last maintenance date.", null)
   ```

   This expression checks to see if the next maintenance date is before the last maintenance date. If it is, the field displays an error message. If not, the field is valid and displays no error message.

10. Click **OK**, then **Save Changes**.

Add a File Upload Component

Next, add the Image field back as a File Upload component.

1. Drag and drop a **File Upload** component onto the canvas. Edit the field label to **Image**.

2. In the **Component Configuration** pane, scroll to the **Target Folder** field**.** Type and select the constant **AX_DOCUMENTS_FOLDER_POINTER**.

3. For the **Selected Files** and **Save Files To** fields, select **ri!vehicle.vehicleImage** using the dropdown menu.

4. Scroll down, and select the **Required** checkbox.

Add Dropdown Components

In this section, you will add the Category, Status, and Condition fields back as Dropdown components. You will then populate these dropdowns using data stored in a reference database table. For the sake of exercise, you will draw on data and objects that already exist in the Acme Automobile Reference app. Complete these steps to add dropdown components.

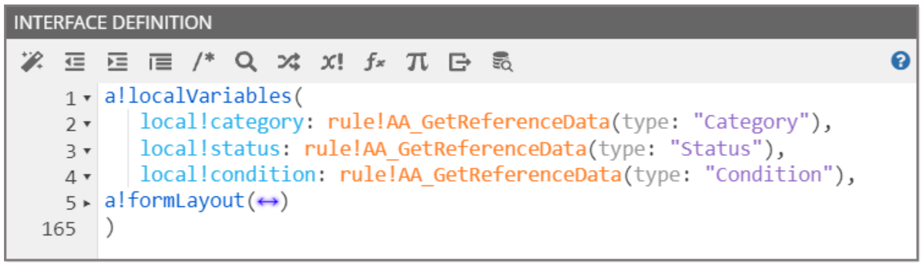
1. Drag and drop three **Dropdown** components onto the canvas. Edit their labels to **Category**, **Status**, and **Condition**.

2. To draw on the values stored in a reference table, you will need to create three local variables. Click **Expression Mode**

to access the expression for your interface.

3. Click the little arrow in line 1 to collapse the interface expression, and then press **Enter**. Type and select **a!localVariables**, and then cut the **closing parenthesis**, and paste it at the end of the expression. Press **Enter** again, and type the following expression in lines 2 through 4:

```
local!category: rule!AA_GetReferenceData(type: "Category"),
local!status: rule!AA_GetReferenceData(type: "Status"),
local!condition: rule!AA_GetReferenceData(type:
"Condition"),
```



```
  1 a!localVariables(
  2   local!category: rule!AA_GetReferenceData(type: "Category"),
  3   local!status: rule!AA_GetReferenceData(type: "Status"),
  4   local!condition: rule!AA_GetReferenceData(type: "Condition"),
  5 a!formLayout(↔)
165 )
```

You now have three local variables that you will use to define the values in the Category, Status, and Condition fields. Click **Save Changes**.

4. Switch to the design mode by clicking the **Design Mode** button

, and save changes.

5. Next, configure the Category field to display correct choices in the dropdown menu. Make sure the Category field is selected, and scroll to **Choice Labels** in the **Component Configuration** pane. Click the **Expression Editor icon** next to it, and enter the following expression:

```
local!category[recordType!Category.fields.label]
```

6. Scroll to **Choice Value**s, and click the **Expression Editor** icon next to it. Enter the following expression:

```
local!category[recordType!Category.fields.id]
```

7. Scroll to **Selected Value** and **Save Selection To**. Use the in-line arrow to select **ri!vehicle.vehicleCategory** in both fields.

8. In the same way, configure the **Status** and **Condition** fields.

    - For the **Status** field, use `local!status[recordType!Status.fields.label]` in **Choice Labels**, and `local!status[recordType!Status.fields.id]` in **Choice Values**.
    - For the **Condition** field, use `local!condition[recordType!Condition.fields.label]` in **Choice Labels** and `local!condition[recordType!Condition.fields.id]` in **Choice Values**.

9. Configure the **Selected Value** and **Save Selectio**n **To** fields for Status and Condition. Make sure to select the corresponding fields for Status and Condition.

10. Scroll down and select the **Required** checkbox for all three dropdowns.

11. The dropdown fields should now display the correct choices. Review, and click **Save Changes** to save your work.

TIP: Use the Visibility Setting to Conditionally Show Interface Components

- In some cases, you may want to only display a component based on a condition.
- To do this, navigate to the **Visibility** setting in the **Component Configuration** pane, then select **Only Show When**. Use the **Expression Editor** to write an expression.
- For example, imagine that you want to display a Comments box if a vehicle's mileage is above 150,000 miles. You can do this by adding the expression **ri!vehicle.vehicleMileage > 150000**. This expression determines whether the Comments component is displayed on the interface or not. When set to false, the component is hidden and not evaluated. The default is set to true.

Configure the Submit Button

In this step, you will update the Date Added value to display the current date. While the Registrars won't fill out the vehicle added date using the Add Vehicle form, this important date still needs to be recorded to the database table for each new vehicle in the fleet. Since it is going to be a read-only field and not a user input field, you will need to configure the Submit button to auto-save the Date Added value.

1. Click the **Date Added** field.

2. In the **Component Configuration** pane, scroll down to **Display Value**, and click the **Expression Editor**.

3. Clear the existing expression, and type **today()**. Click **OK**.

4. Scroll down, and select the **Read-only** checkbox.

5. Click the **Submit** button.

6. Click the **Expression Editor** for **Save Value To**, and type the following expression:

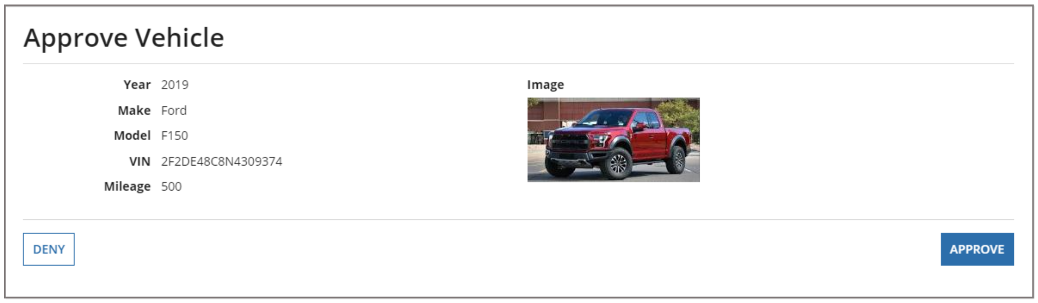
   ```
   a!save(ri!vehicle.vehicleDateAdded, today())
   ```

   In this expression, **ri!vehicle.vehicleDateAdded** is the target and **today()** is the value being saved into the target.

7. Click **OK**, then **Save Changes**.

Create the Supervisor Approval Form

In this part, you will create an interface for the Supervisor to approve new vehicles. After the Registrar submits a new vehicle, the Supervisor will receive a task to approve or reject the vehicle. The interface that you create in this exercise will look like the image below:



Approve Vehicle

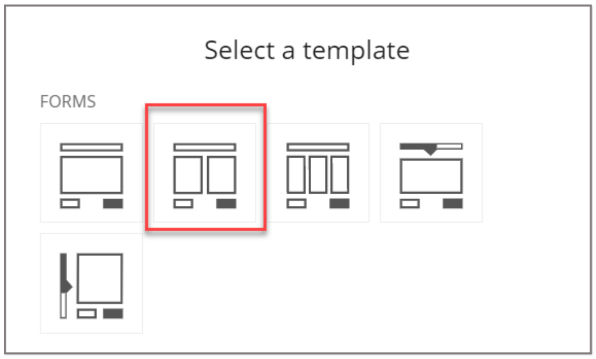
Year	2019
Make	Ford
Model	F150
VIN	2F2DE48C8N4309374
Mileage	500

Follow the steps below to create a Supervisor approval form.

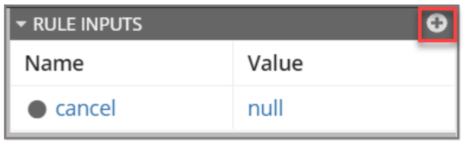
1. In your application, click **New** and select **Interface**. Name it **AX_SupervisorForm**, and add a simple description: "Displays read-only details about a selected vehicle for Supervisor approval."

2. Save the interface in **AX Interfaces**, and click **Create**.

3. Choose the **Two-Column Form** template:



4. Click the **plus** sign in the Rule Inputs, and add the new rule input **vehicle**. Select the CDT **AX_Vehicle** in the **Type** field, and click **Create**.



Name	Value
● cancel	null

5. Rename the **cancel** rule input to **approvalDecision**, but leave **Boolean** as the type.

6. Rename the form to **Approve Vehicle**, and delete the top **Section Layout**. To delete, select this section, and use the dropdown arrow to navigate to **Delete**.

7. Delete the **blue Section label** for the bottom section layout.

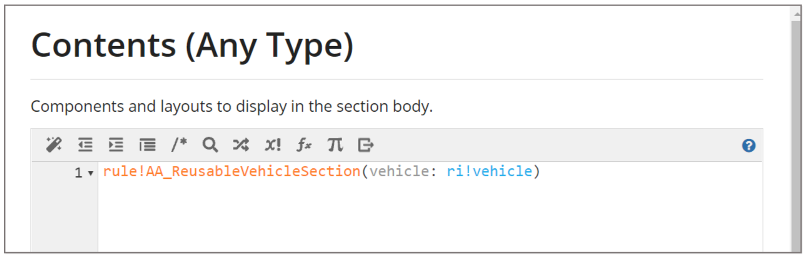
8. Drag and drop **three text** and **two integer components** into the first column.

9. Change the labels for the text fields to **Make**, **Model**, **VIN**, and for the integer fields to **Year** and **Mileage**. For each of these fields, make the following changes in the **Components Configuration** pane:

    - Select **Adjacent** in the **Label Position** dropdown.
    - Select the correct values for **Display Value**. Use the dropdown menus to select the values. For example, for Vehicle Year, select **ri!vehicle.vehicleYear** from the dropdown menu.
    - Check the **Read-only** checkbox.

TIP: Save Time by Creating Reusable Interfaces

- As you plan your first application, keep in mind that interfaces can be built as reusable blocks. Simply create a reusable interface, and then call it from other interfaces in your app. For example, the read-only vehicle details that you just added to the supervisor approval form could have been set up as its own interface. To call a reusable interface from a different interface, use **rule!** and the name of the interface you want to reuse:



Contents (Any Type)

Components and layouts to display in the section body.

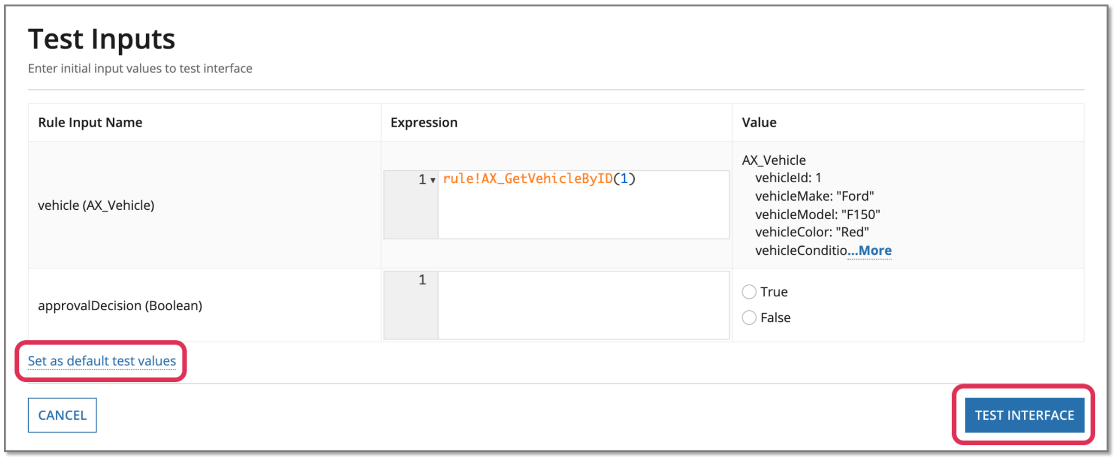
```
rule!AA_ReusableVehicleSection(vehicle: ri!vehicle)
```

- This is a particularly useful technique for reporting interfaces. You can create each chart or grid as its own interface, and then combine them into different larger reporting interfaces that you may need for the business users of your app.

10. Next, drag and drop an **Image** component into the second column. Delete the label **Image**, and click the **+** in the Image field. Select **Document Image**. In the **Component Configuration** pane, click **a!EXAMPLE_DOCUMENT_IMAGE()**, and delete all text in the **Expression Editor**. Type the following expression:

```
ri!vehicle.vehicleImage
```

11. You will see an error message. This error appears because the value for the document cannot be null. Let's fix this by adding test values to this interface. Click **Test**, and type the following expression into the vehicle row: `rule!AX_GetVehicleByID(1)`. Click the **Set as default test values** link, and then click **Test Interface**.



Test Inputs

Enter initial input values to test interface

Rule Input Name	Expression	Value
vehicle (AX_Vehicle)	rule!AX_GetVehicleByID(1)	AX_Vehicle vehicleId: 1 vehicleMake: "Ford" vehicleModel: "F150" vehicleColor: "Red" vehicleConditio...More
approvalDecision (Boolean)		○ True ○ False

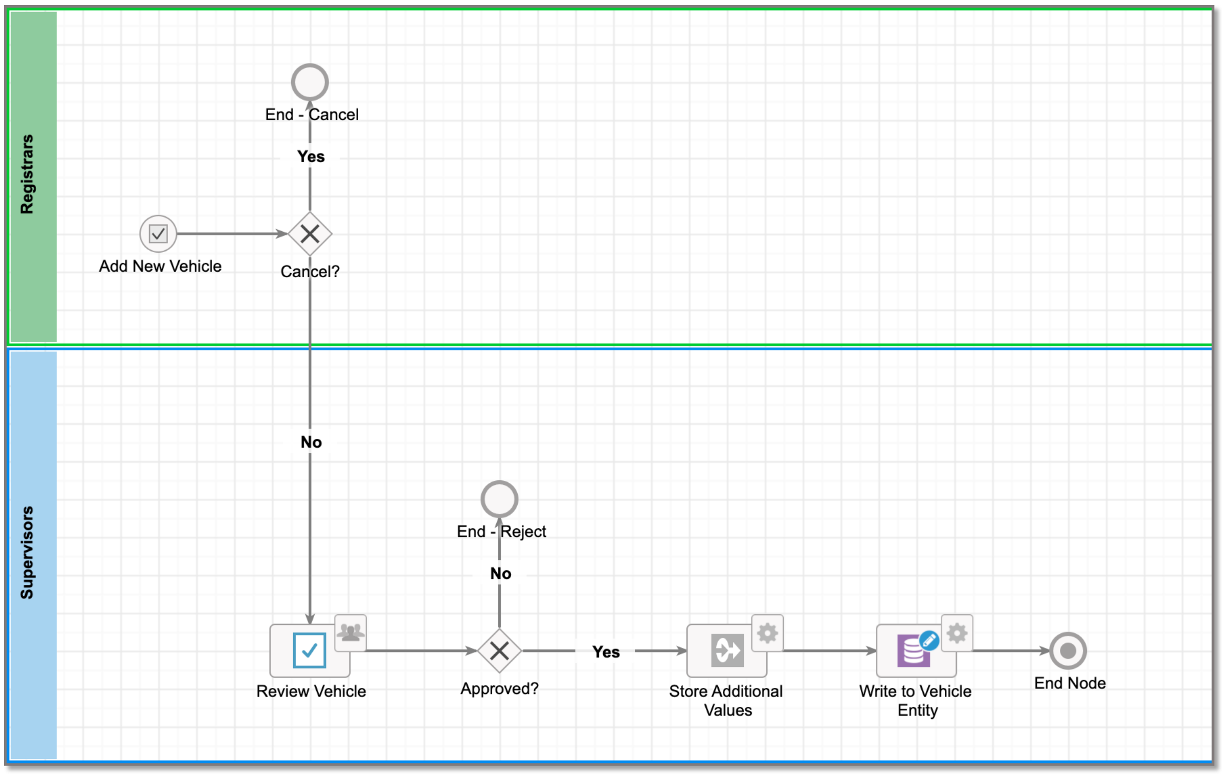
You will see the error disappear, and the interface will display the sample data.

12. Select the **Cancel** button, and rename it to **Reject**. Click the **link** under **Value**, and change it to **false**. Select **ri!approvalDecision** in the **Save Value To** field.
13. Select the **Submit** button, and rename it to **Approve**. Click the **link** under **Value**, and make sure it is set to **true**. Select **ri!approvalDecision** in the **Save Value To** field. Click **Save Changes**.

Edit the Add Vehicle Process Model

The Add Vehicle record action has already created a preliminary process model for you. In this exercise, you will edit this auto-generated process model to include the steps for the Supervisor approval of the vehicle addition to the fleet. Keep in mind that because this process model was auto-generated, security is already set for this model. In other instances, you will need to secure each new process model individually.

At the end of this exercise, your process model will look like the image below.



Follow the steps below to complete this exercise.

Configure Process Properties

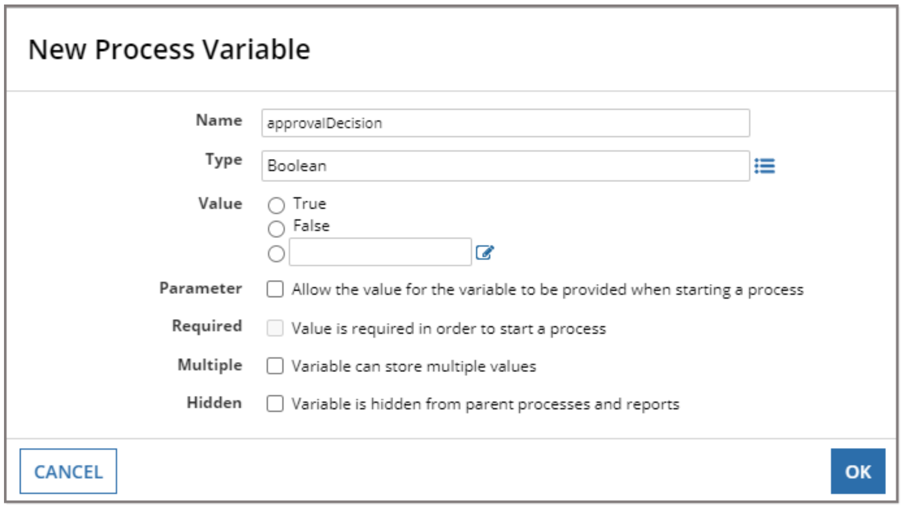
Let's start by configuring the process model properties, such as variables, alerts, and data management. Because this process model was auto-generated, some of the properties are already configured. Follow the steps below to review and edit process properties:

1. Open the **AX New Vehicle** process model.

2. If you are in the Analyst View, switch to **Designer View**. Simply click the button in the upper right corner of the process modeler.

3. Click the **Properties** button in the toolbar.

4. Navigate to the **Variables** tab. You need to add a new process variable **approvalDecision** that you will use to configure the Approved? XOR node later. Click **Add Variable**, and name the new variable **approvalDecision**. Type **Boolean** into the **Type** field, and click **OK**.



New Process Variable

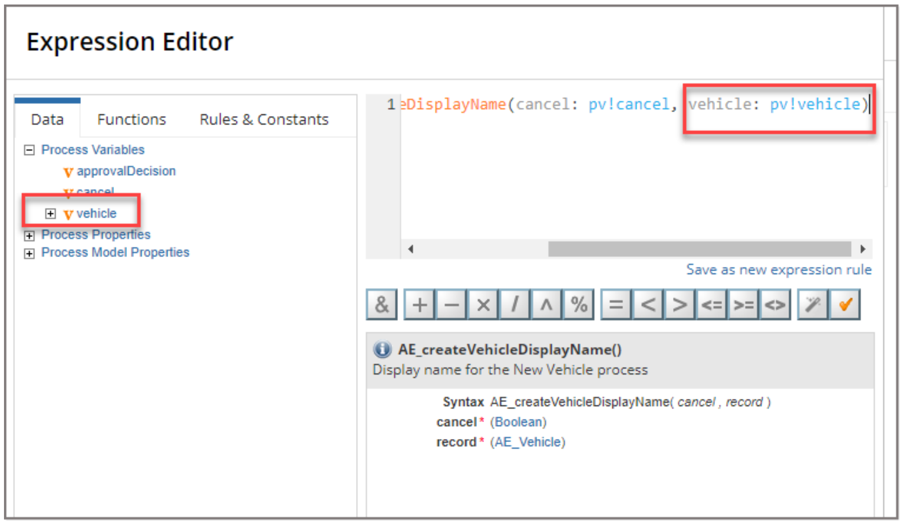
Name	approvalDecision
Type	Boolean
Value	○ True ○ False ○
Parameter	☐ Allow the value for the variable to be provided when starting a process
Required	☐ Value is required in order to start a process
Multiple	☐ Variable can store multiple values
Hidden	☐ Variable is hidden from parent processes and reports

5. Click the **record** variable, and update the name to **vehicle**.

6. In the **General** tab, click the **Expression Editor icon** next to the **Process Display Name** field. The AX_createVehicleDisplayName expression was created alongside the record action. This expression rule creates a dynamic display name that users can see after a process has been started.

7. Before you use this expression rule, open the rule in a new tab. This can be done by holding the ctrl/command key and clicking the name of the rule within the Expression Editor. Once the expression rule is open, change the name of the **record** rule input to **vehicle**. Click **Save Changes**.

8. Click in the expression, and delete **record: pv!record**. Type **vehicle:**, and insert **pv!vehicle** by selecting it in the **Data** tab.



```
=DisplayName(cancel: pv!cancel, vehicle: pv!vehicle)
```

AE_createVehicleDisplayName()
Display name for the New Vehicle process

Syntax AE_createVehicleDisplayName(*cancel* , *record*)
cancel* (Boolean)
record* (AE_Vehicle)

Click **Save and Close.**

9. Navigate to the **Process Start Form** tab to review the interface that starts the process. Make sure the rule inputs for this interface are **vehicle** and **cancel**.

10. Navigate to the **Alerts** tab to review the alert settings. Make sure that the **AX Administrators** are selected to receive alert notifications. Update the group if necessary.

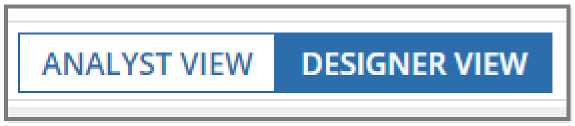
11. Navigate to the **Data Management** tab to adjust the timeline for process archiving. The system default is to archive processes 7 days after completion. Change this to **3** days to improve utilization of server resources. This setting will archive process history, metrics, and variables from memory after three days.

12. Click **OK**, then **File > Save**.

TIP: Use the Analyst View to Draft High Level Diagrams

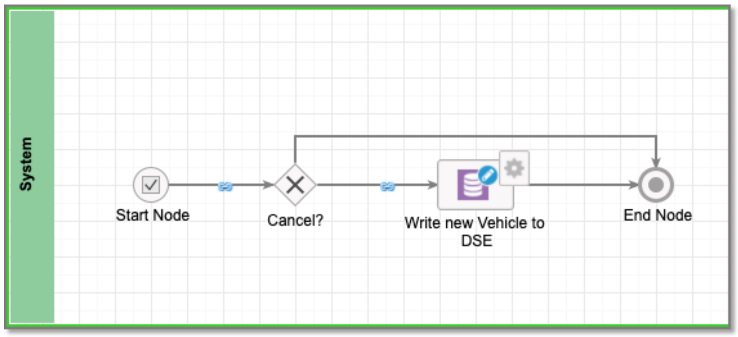
Note that you can select to view a process model using the Analyst or Designer views. The process Analyst View allows you to draft high-level diagrams, while the process Designer View allows you to configure the nodes and publish process models. Use the Analyst View when drafting a process, for example if you want to share a process diagram with stakeholders or developers. This will suppress any errors and validation messages.

You can select the views from the upper right corner in the Process Modeler:



Configure the Cancel XOR Gateway

Gateways are added to the process when you need to branch the workflow. In this process, if the Registrar cancels the start form, the process will end. If the Registrar submits the form, the process will continue to the next process node. Appian has created a template for you that you will need to edit before it becomes fully functional.



Follow the steps below to configure the Cancel? Gateway.

1. Switch to **Designer View** in the top right corner.
2. Click **System** in the first swim lane, and rename it to **Registrars**. Add a swimlane for **Supervisors**. Click the **Add Horizontal Lane** button in the toolbar to add one swimlane, and rename it to **Supervisors**.
3. Rename the **Start Node** to **Add New Vehicle.**
4. Drag a new **End Event** above the Cancel? XOR gateway in the Registrars swimlane. Rename it to **End - Cancel**.
5. Delete the **line** connecting the **Cancel? XOR gateway** and the **End Node**.
6. Connect the **Cancel? XOR gateway** and **End - Cancel** nodes. To connect, hold down **SHIFT** on your keyboard, and then click from one node to another. SHIFT will automatically toggle your pointer to the connector tool. You can also use the **Connect** icon to connect the nodes.

7. Double-click **Cancel?**, and navigate to the **Decision** tab.

8. For **pv!cancel**, select the **End - Cancel** in the **Result** column.

9. For the **Else if none are TRUE** condition, select **Write new Vehicle to DSE** in the **Result** column, and click **OK**.

10. Click **File > Save**.

Configure the Write to Vehicle Entity

Before you test your preliminary process model, configure the Write to Data Store Entity smart service to ensure that all vehicle data is written to the database table. Follow the steps below to complete this task.

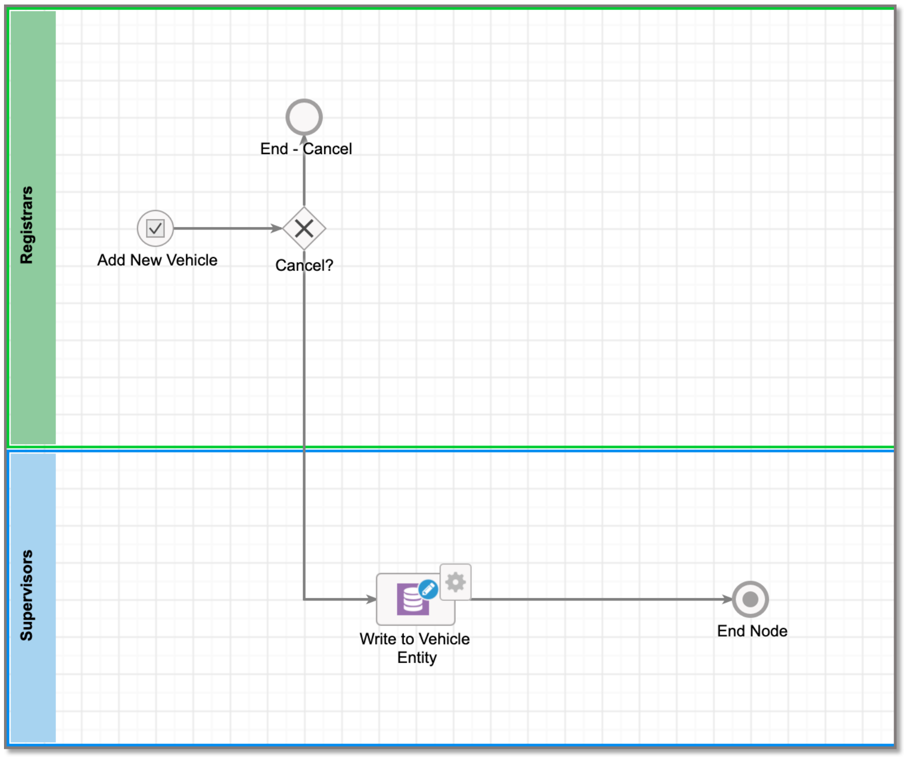
1. Rename the **Write new Vehicle to DSE** node to **Write to Vehicle Entity**.

2. Delete the **line** connecting **Cancel?** and **Write to Vehicle Entity**.

3. Move the **Write to Vehicle Entity** and **End Node** nodes to the Supervisors lane.

4. Connect the **Cancel?** And **Write to Vehicle Entity** nodes.

5. Double-click **Cancel?**, and navigate to the **Decision** tab. Make sure **Write to Vehicle Entity** is selected for the **Else if none are TRUE** condition in the **Result** column. Click **OK**:



6. Double-click **Write to Vehicle Entity**.

7. Navigate to the **Data** tab, and then to the **Inputs** tab.

8. You will notice that two node inputs have been already created for you. This was done when you created the Add Vehicle record action. Click **record**, and update the name to **vehicle**.

9. Select **vehicle** from the **Value** dropdown field. It will display as pv!vehicle when selected.

10. Navigate to the **Assignment** tab. Use this tab to assign the Write to Vehicle Entity activity to process designers. This is done to elevate the security of this node since writing to the application's Data Store Entity is not something that should be done by basic users, and it typically requires a higher level of permission. Select the **Run as whoever designed this process model** radio button, and click **OK**.

11. Click **File** > **Save and Publish**.

Test the Process

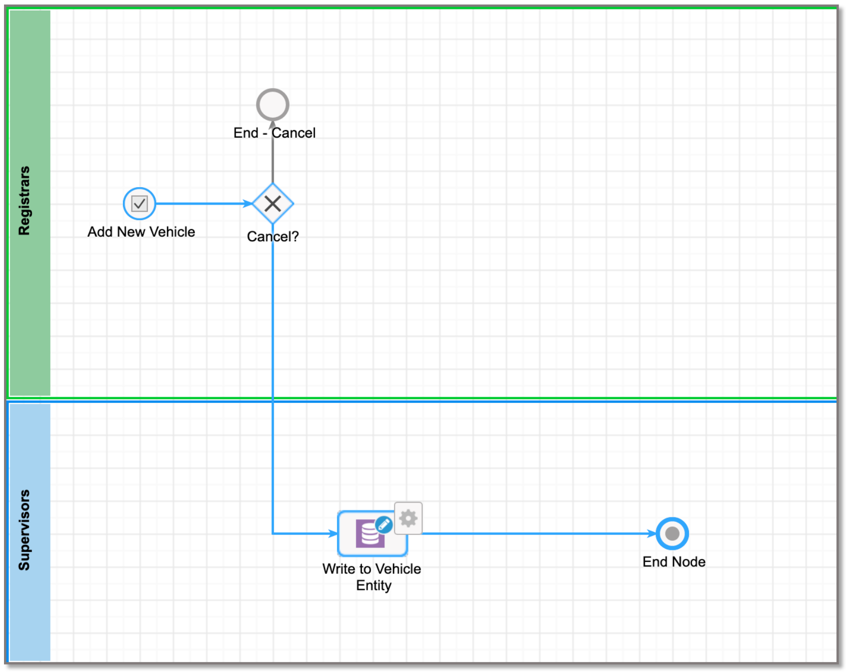
Testing should be performed each time you add a new piece of functionality to the process model. Be sure to test frequently to make sure your process model works as expected. Follow the steps below to test the process.

1. Click **File**, and select **Start Process for Debugging**.

2. Complete the **Add Vehicle** form, and click **Submit**. Since these are test values, feel free to make up vehicle details to enter into the form.

3. Once you submit, you will notice that a new process instance has started in the **Monitoring Mode** and on a new tab in the Process Modeler.

4. Click the **Refresh** button in the toolbar to update the process flow. You will notice that the process will advance to the **End Node** node:



5. Click the **Process Details** button in the toolbar, and navigate to the **Variables** tab to verify that the data you entered into the form was captured by the process model.

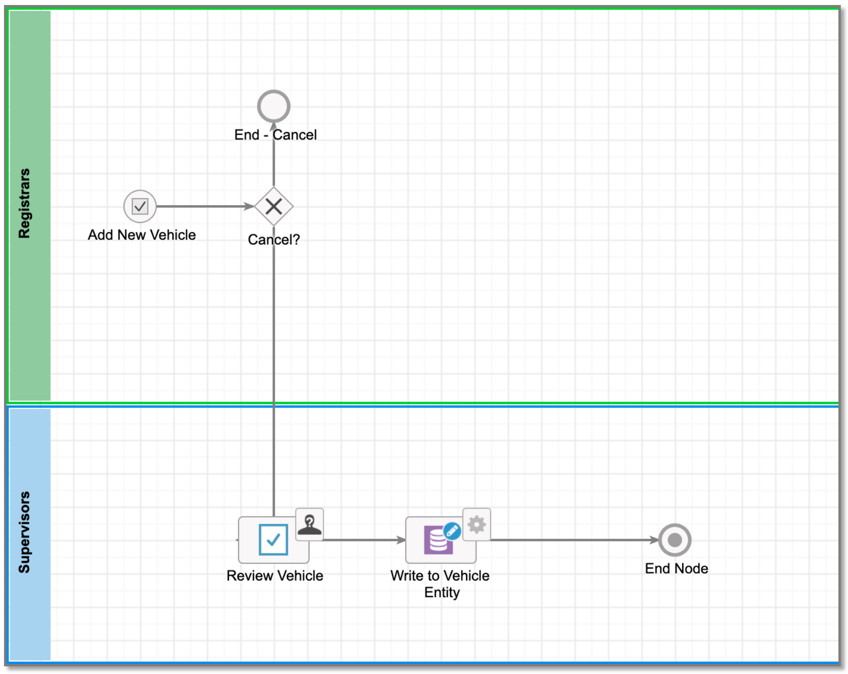
6. Exit out of the **Monitoring** tab to return to your process model.

Configure the User Input Task

Next, let's configure the Review Vehicle node. You will use this node to add the Supervisor approval form to this process. Complete the steps below to add and configure a user input task.

1. Drag a new **User Input Task** node to the Supervisor swimlane to the left of Write to Vehicle Entity. Rename it to **Review Vehicle**.

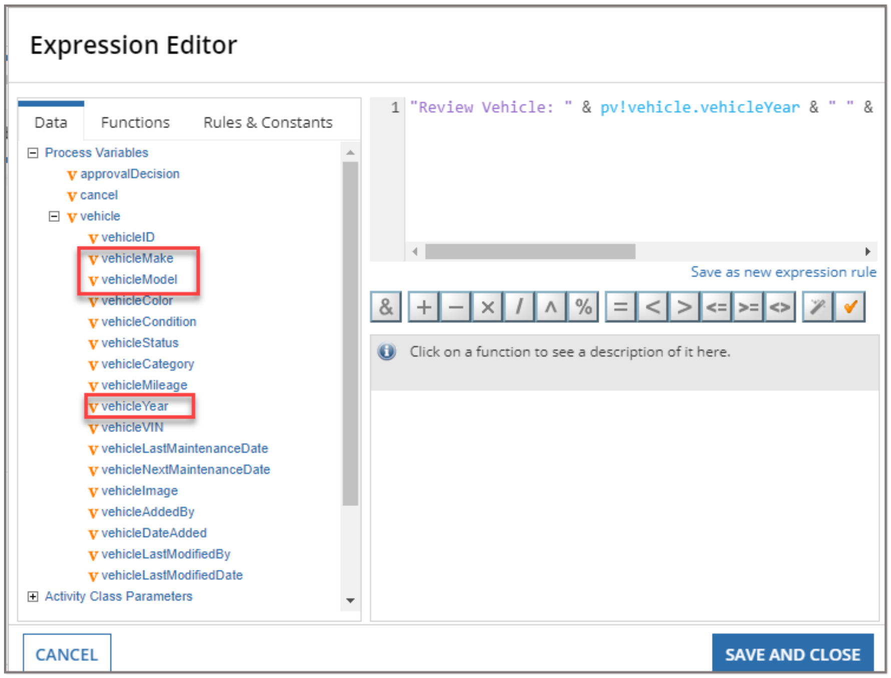
2. Reconnect the nodes so that **Review Vehicle** falls in between **Cancel?** and **Write to Vehicle Entity**:



3. Double-click **Cancel?**, and navigate to the **Decision** tab. Make sure **Review Vehicle** is now selected for the **Else if none are TRUE** condition in the **Result** column. Click **OK**.

4. Double-click **Review Vehicle**. First, edit the **Task Display Name** to make it dynamic. Click the **Expression Editor icon**, and select the process variables for the **year**, **make**, and **model** to insert them into the expression. Your expression should look as follows:

```
"Review Vehicle: " & pv!vehicle.vehicleYear & " " &
pv!vehicle.vehicleMake & " " & pv!vehicle.vehicleModel
```



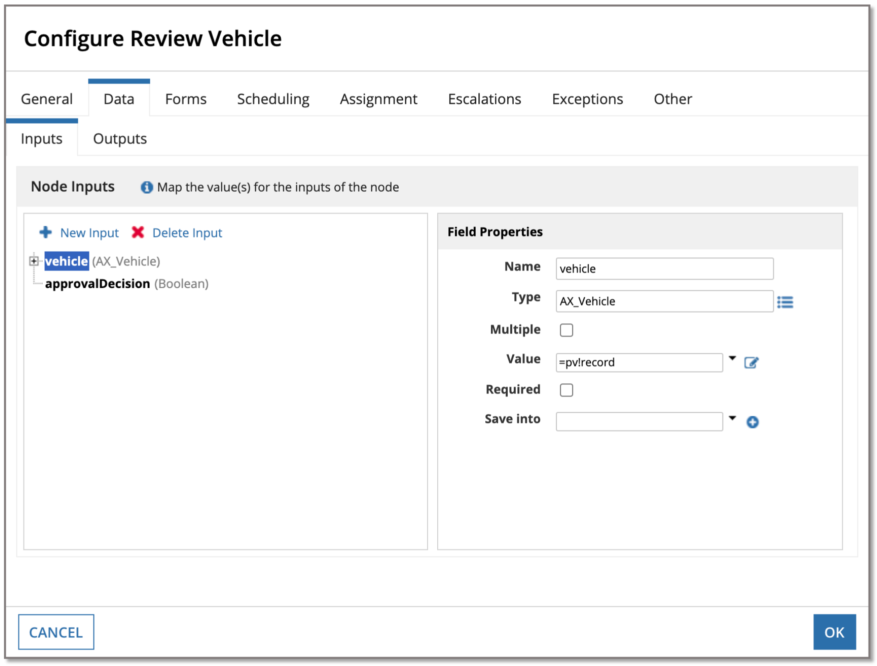
```
"Review Vehicle: " & pv!vehicle.vehicleYear & " " &
```

5. Navigate to the **Forms** tab, and use the **Directory** icon to select the **AX_SupervisorForm**. Allow the process model to automatically create node inputs to match your interface's inputs.

6. Navigate to the **Assignment** tab. In the **Assign to the following** field, type and select **AX Supervisors** from the list of auto-suggestions.

7. Navigate to the **Data** tab. You will need to configure the inputs in this tab to ensure that all existing vehicle data flows into the form, and the new data, entered by the Supervisor, is captured as well.

8. Select the **vehicle** node input. Click the dropdown menu next to the **Value** field, and select **vehicle**. This will allow you to display the existing vehicle data in the form. You don't need to update the **Save into** field because the Supervisor will not modify any vehicle information.



vehicle (AX_Vehicle)
approvalDecision (Boolean)

Field Properties

Name	vehicle
Type	AX_Vehicle
Multiple	☐
Value	=pv!record
Required	☐
Save into	

9. Select the **approvalDecision** node input. Click the dropdown menu next to **Save into**, and select the **approvalDecision** process variable. If the Supervisor clicks **Deny**, this will save the approvalDecision value as false into the process.

10. Click **OK**, and then save the process model by clicking **File** > **Save & Publish**.

Test the Process

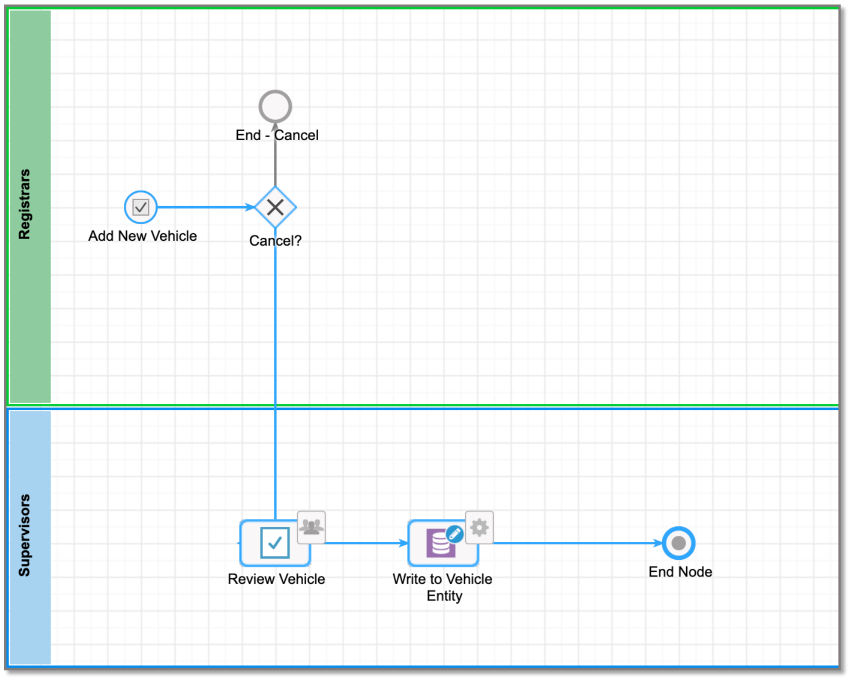
1. Click **File**, and select **Start Process for Debugging**.

2. Complete the **Add Vehicle** form, and click **Submit**. Since these are test values, feel free to make up vehicle details to enter into the form.

3. Click the **Refresh** button in the toolbar to update the process flow. You will notice that the process will advance to the **Review Vehicle** node.

4. Right-click the **Review Vehicle** node, and select **View Form**. Click **Accept** to accept the task, and then click **Approve**.

5. Click the **Refresh** button once again to update the process flow. You will notice that the process will advance to the **End Node** node:



6. Click the **Process Details** button in the toolbar, and navigate to the **Variables** tab to verify that the data you entered into the form was captured by the process model.

7. Exit out of the **Monitoring** tab to return to your process model.

Configure the Approved? XOR Gateway

In this process, if the Supervisor denies the AX_SupervisorForm form, the process will end. If the Supervisor approves the form, the process will continue to the next process node. Follow the steps below to configure the Approved? XOR node.

1. Drag and drop a **XOR** node to the right of Review Vehicle. Rename it **Approved?**

2. Drag and drop an **End Event** node above Approved? Rename it **End - Reject**.

3. Connect the **Approved?** and **End - Reject** nodes.

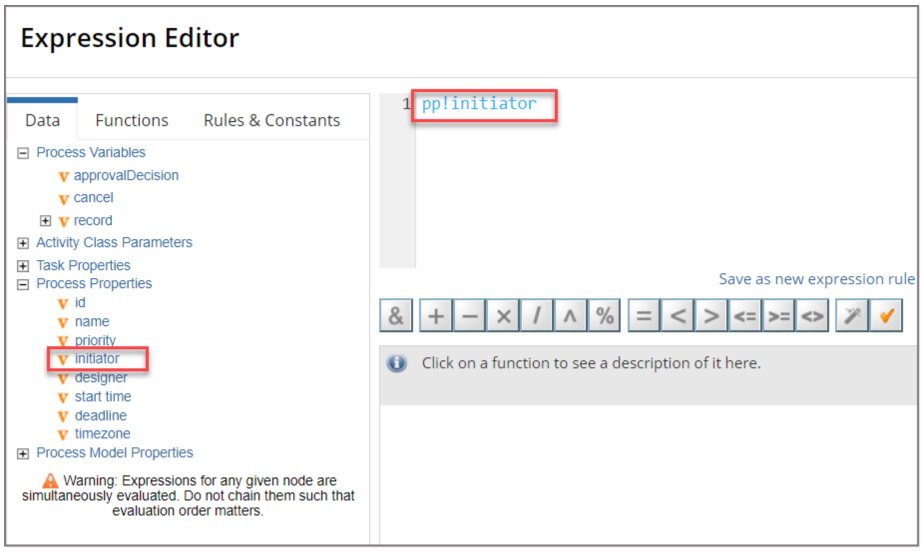
4. Double-click **Approved?**, and navigate to the **Decision** tab.
5. Click **New Condition**, and then click the **Expression Editor** icon next to the first condition. Select the **approvalDecision** process variable. Click **Save and Close**.
6. In the **Results** column, in the first dropdown, select **Write to Vehicle Entity**.
7. In the **Results** column, in the second dropdown for the **Else if none are TRUE** condition, select **End - Reject**.
8. Click **OK**, and save the process model by going to **File > Save and Publish**.

Configure the Script Task

A script task is used to perform an automated activity. For this process, you will use a script task to store three variables: Vehicle Last Modified Date, Vehicle Last Modified By, and Vehicle Added By. In this example, the script task is used because we don't want the user to enter these values manually. Instead, we will lift the date and the name of the user from the process automatically. Follow the steps below to set up the script task.

1. Drag and drop a **Script Task** to the right of Approved? Rename it **Store Additional Values**.
2. Double-click **Approved?**, and navigate to the **Decision** tab.
3. Make sure **Store Additional Values** is selected in the first dropdown of the **Results** column. Click **OK**.
4. Double-click **Store Additional Values**, and navigate to the **Data** tab. Click the **Outputs** tab.
5. Click **New Custom Output**. You will want to set the last modified date as the current date. Click the **Expression Editor** icon, and type the function **today()**. Click **Save and Close**.
6. For the **Target**, click the **inline arrow,** and hold your cursor over **vehicle**. Select **vehicle.vehicleLastModifiedDate**.

7. Click **New Custom Output**. You will now want to set the last person to modify the record as the initiator of the process model. Click the **Expression Editor** icon, and type the function **pp!initiator**. Alternatively, click **initiator** under **Process Properties**:



```
pp!initiator
```

Warning: Expressions for any given node are simultaneously evaluated. Do not chain them such that evaluation order matters.

Click on a function to see a description of it here.

8. Click **Save and Close**.

9. For the Target, select **vehicle.vehicleLastModifiedBy**.

10. Click **New Custom Output** once again. You will now want to set the person who added the vehicle. Click the **Expression Editor** icon, and type the function **pp!initiator**. Click **Save and Close**.

11. For the target, select **vehicle.vehicleAddedBy**.

12. Click **OK**, save the process model by clicking **File > Save and Publish**.

Test the Process

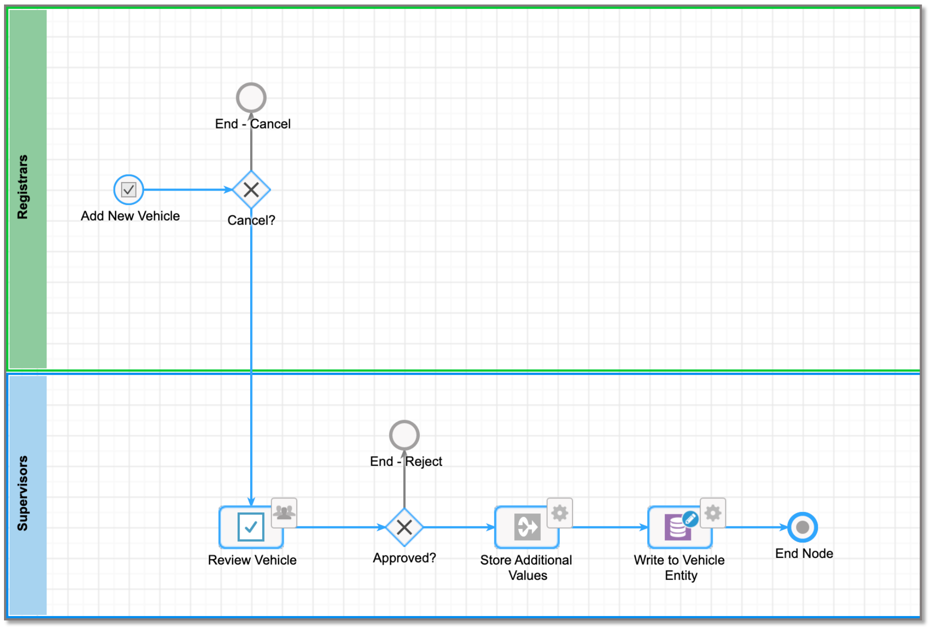
1. Click **File**, and select **Start Process for Debugging**.

2. Complete the **Add Vehicle** form, and click **Submit**. Since these are test values, feel free to make up vehicle details to enter into the form.

3. Click the **Refresh** button in the toolbar to update the process flow. You will notice that the process will advance to the **Review Vehicle** node.

4. Right-click the **Review Vehicle** node, and select **View Form**. Click **Accept** to accept the task, and then click **Approve**.

5. Click the **Refresh** button once again to update the process flow. You will notice that the process will advance to the **End Node** node:



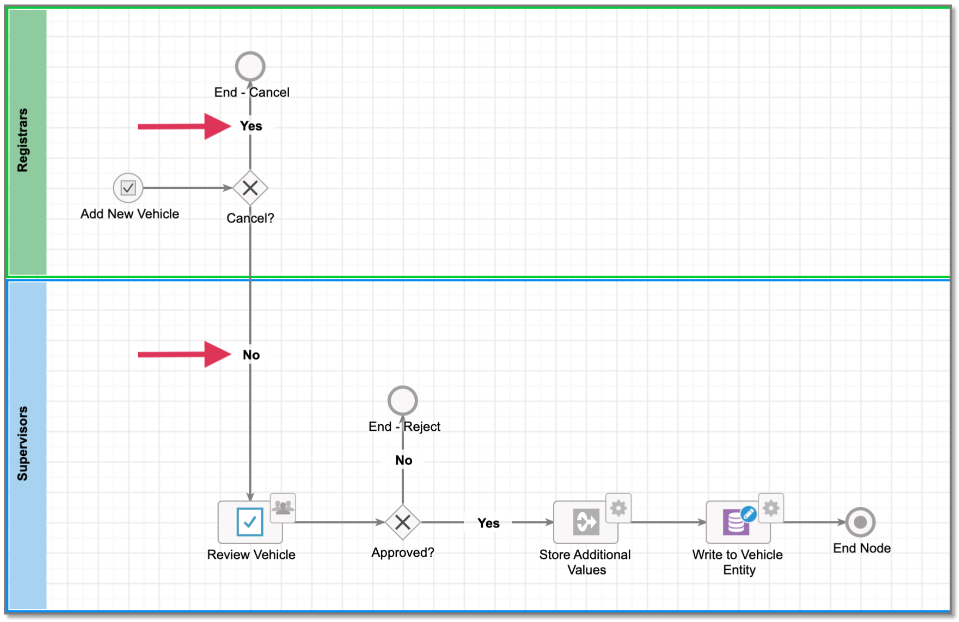
6. Click the **Process Details** button in the toolbar, and navigate to the **Variables** tab to verify that the data you entered into the form was captured by the process model.

7. Close the process monitoring tab, but keep the process model open. Debug the process model once again, but this time for the **Reject** flow. Check that the process arrives at the **End - Reject** flow.

Add Flow Labels

Flow labels help keep your process models clear and organized. Follow the steps to add flow labels to your process:

1. Add the labels **Yes** and **No** to the connectors leading from **Cancel?** To add labels, right-click each individual connector, and select **Add Label**. Rename labels to **Yes** and **No**:



2. Add the labels **Yes** and **No** to the connectors leading from **Approved?** Refer to the image above to see the labels.

3. Click **File > Save & Publish**.