Appian
Academy

Appian Step-by-Step #9

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 9: Process Modeling 101, Part 2

In this exercise, you will create a process model for requesting vehicle maintenance, along with other objects. You will create both the process model and related interfaces to ensure that they function correctly and meet application requirements. These requirements are:

- The process allows Mechanics to fill out the Request Maintenance form
- The process allows the Supervisor to approve maintenance requests
- If the maintenance request is approved, the process writes all vehicle details to the database

Before you start building the process model, let's create the interfaces and other objects that this process model will need.

Create the Maintenance Custom Data Type

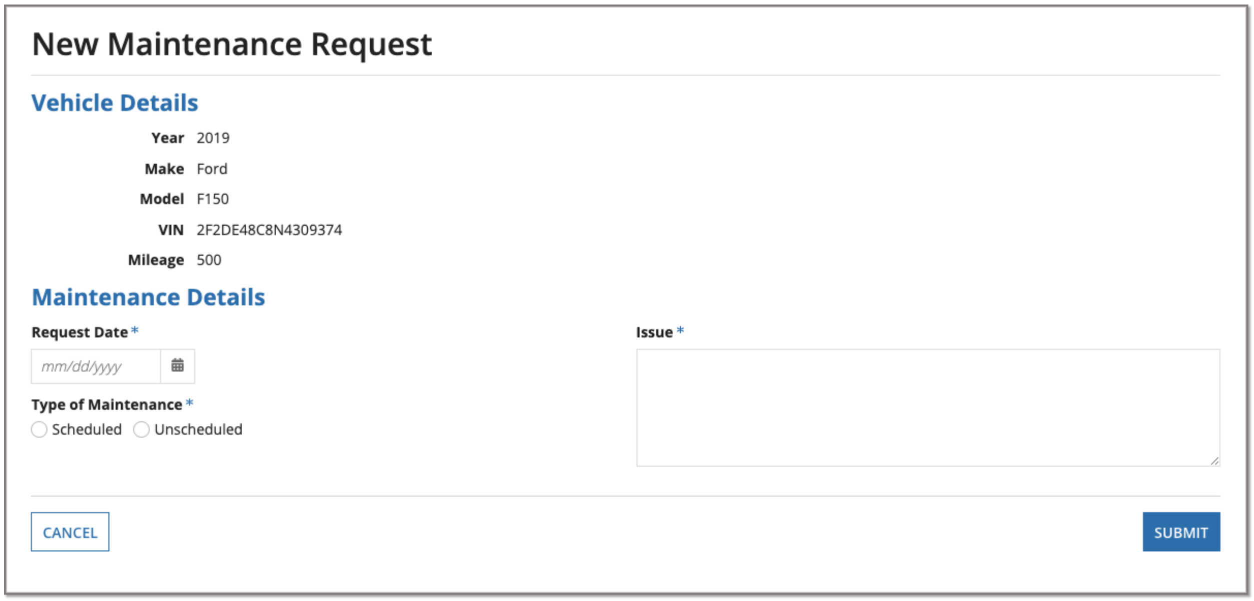
Since this process model involves writing maintenance data to the database, you will need a maintenance data entity. You will use this maintenance data entity in both your interfaces and process model.

Follow the steps below to create a maintenance data entity.

1. In your application, click **New**, then **Data Type**.
2. Select the **Create from database table or view** radio button.
3. Select **Appian (Tomcat)** as the **Data Source** and **aamaintenance** as the **Table or View**.
4. Name the data type **AX_Maintenance**, and add a description: "Base data structure for a single Maintenance record."
5. Edit the field names to use camel casing.
6. Scroll down, and if necessary, rename the **Entity Name** to **AX_Maintenance**.
7. Click **Create**.

Create the Request Maintenance Form

In this part, you will create the AX_RequestMaintenance form that mechanics will use to request maintenance. This interface will contain a section with read-only vehicle details and an editable section for Mechanics to enter the maintenance-related details. At the end of this exercise, your interface will look like the image below.



Follow the steps below to create the request maintenance form.

Add Read-Only Vehicle Details

1. In your application, click **New** and create a new interface **AX_RequestMaintenance**. Add a simple description, and save this interface in the **AX Interfaces** folder.

2. Select the **One-Column Form** template, and update the form label to **New Maintenance Request**.

3. In the **Rule Inputs** pane in the top right corner, click the **plus** sign and add two rule inputs. Name the first rule input **vehicle**, and select the **AX_Vehicle** CDT as the **Type**:

4. Name the second rule input **maintenance**, and select the **AX_Maintenance** CDT as the **Type**.

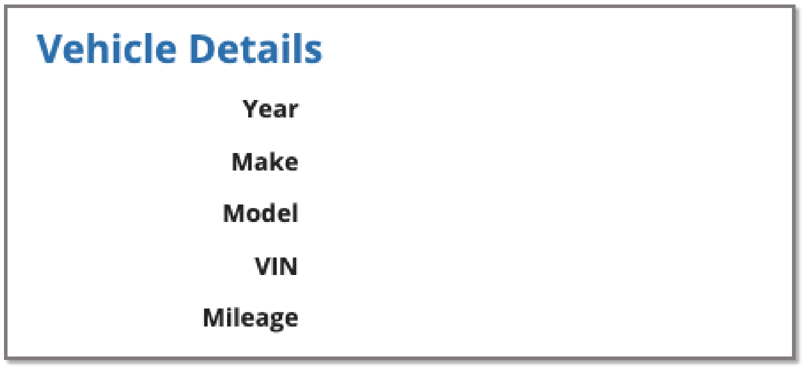
5. Select the first section, and type **Vehicle Details** in the **Label** field.

6. Drag and drop three **Text** fields for the **Make**, **Model**, and **VIN** into the **Vehicle Details** section. Additionally, add two **Integer** fields for the **Year** and **Mileage**. Update the labels for each field.

7. For each field, configure the values in the **Display Value** and **Save Input To** fields. For example, use the dropdown menu for the Year field to select ri!vehicle.vehicleYear in both the Display Value and Save Input To fields.

8. For each field, change the label position to **Adjacent**, and select the **Read-only** checkbox.

9. Rearrange the fields to match the image below.



Add User Input Maintenance Details

1. Select the bottom section, and rename it to **Maintenance Details**.

2. From the **Components Palette**, drag and drop a **Columns** layout into the **Maintenance Details** section. Delete **one column**.

3. Drag and drop a **Date** component into the first column.

   - Update the label to **Request Date**.
   - Use the dropdown menu to select **ri!maintenance.maintenanceRequestDate** in the **Display Value** and **Save Input To** fields.
   - Select the **Required** checkbox.

4. Drag and drop a **Radio Buttons** component into the first column under Request Date.

   - Update the label to **Type of Maintenance**.
   - Click the **Expression Editor** icon next to **Choice Labels**, and type **{"Scheduled", "Unscheduled"}**.
   - Click the **Expression Editor** icon next to **Choice Labels**, and type **{true, false}**.
   - Select **ri!maintenance.maintenanceScheduled** from the dropdown for the **Display Value** and **Save Input To** fields.
   - Select the **Required** checkbox.
   - Scroll down to the **Choice Layout** field, and select **Compact**.

5. Drag and drop a **Paragraph** component into the second column.

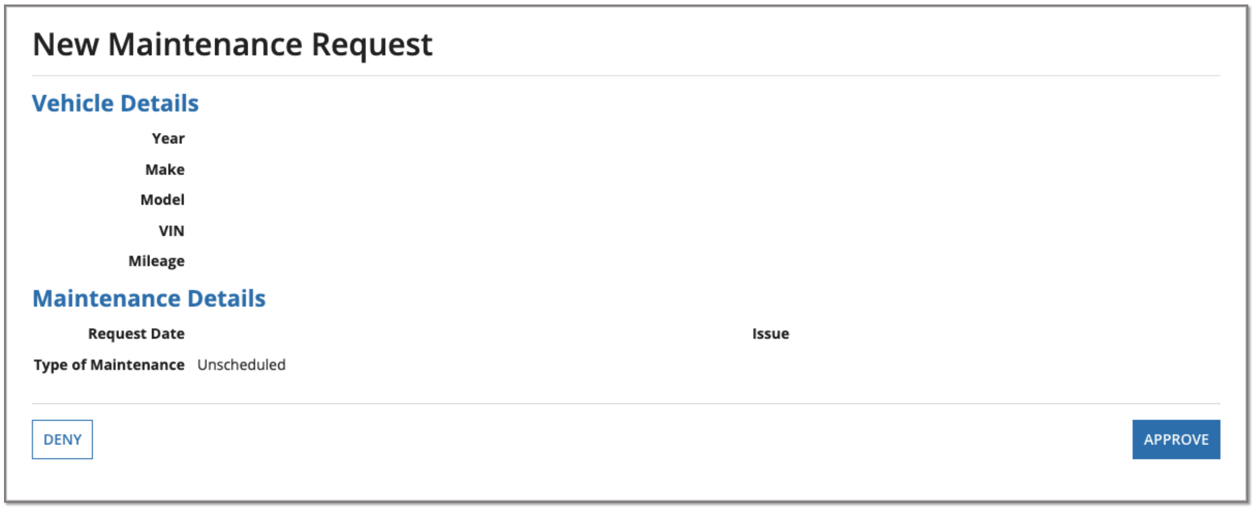
    - Update the label to **Issue**.
    - Select **ri!maintenance.maintenanceIssue** from the dropdown for the **Display Value** and **Save Input To** fields.
    - Select the **Required** checkbox.

6. Let's add some testing values to this interface. Click **Test**, and then type **rule!AX_GetVehicleByID(1)** into the **Expression** field for the vehicle. Click **Set as Default Test Values**, then **Test Interface.** You will see that the interface is now pre-populated with vehicle values.

7. Click **Save Changes**.

Create the Approve Maintenance Form

In this part, you will create the AX_ApproveMaintenance interface for the AX Request Maintenance process model. Once a maintenance request is submitted, Supervisors will use this form to approve or deny the maintenance request. Your goal is to create an interface with one section for read-only vehicle details and one for read-only maintenance details. At the end of this part, your interface will look like the image below.


New Maintenance Request

Vehicle Details

Year

Make

Model

VIN

Mileage

Maintenance Details

Request Date

Issue

Type of Maintenance Unscheduled

Follow the steps to create the approve maintenance form.

1. In your application, click **New**, and select **Interface**.

2. Select the **Duplicate existing interface** radio button, and then select the **AX_RequestMaintenance** interface.

3. Name this interface **AX_ApproveMaintenance**, and add a description: "Supervisor form for approving or denying a vehicle maintenance request." Save it in **AX Interfaces**, then click **Create**.

4. In the **Rule Inputs** pane, rename the **cancel** rule input to **approvalDecision**. Leave **Boolean** in the **Type** field.

5. Navigate to the **Maintenance Details** section, and select the **Request Date** field. Make the following changes to this field:

   - Change the **Label Position** to **Adjacent**.
   - Deselect the **Required** checkbox.
   - Select the **Read-only** checkbox.

6. Implement the same changes for the **Issue** field.

7. Delete the **Type of Maintenance** field, and replace it with a **Text** component. Make the following changes to the new text field:

   - Update the label to **Type of Maintenance**.
   - Set the **Label Position** to **Adjacent**.
   - Click the **Expression Editor** icon next to **Display Value**, and type the following expression:

     ```
     if(ri!maintenance.maintenanceScheduled, "Scheduled",
     "Unscheduled")
     ```

     This expression formats the field, so that true values display as "Scheduled" and false values display as "Unscheduled."

   - Select the **Read-only** checkbox.

8. Click the **Cancel** button, and make the following changes:

   - Update the label to **Deny**.
   - Under **Value**, replace **true** with **false**.
   - Make sure the **Submit** checkbox is selected.

9. Click the **Submit** button, and make the following changes:

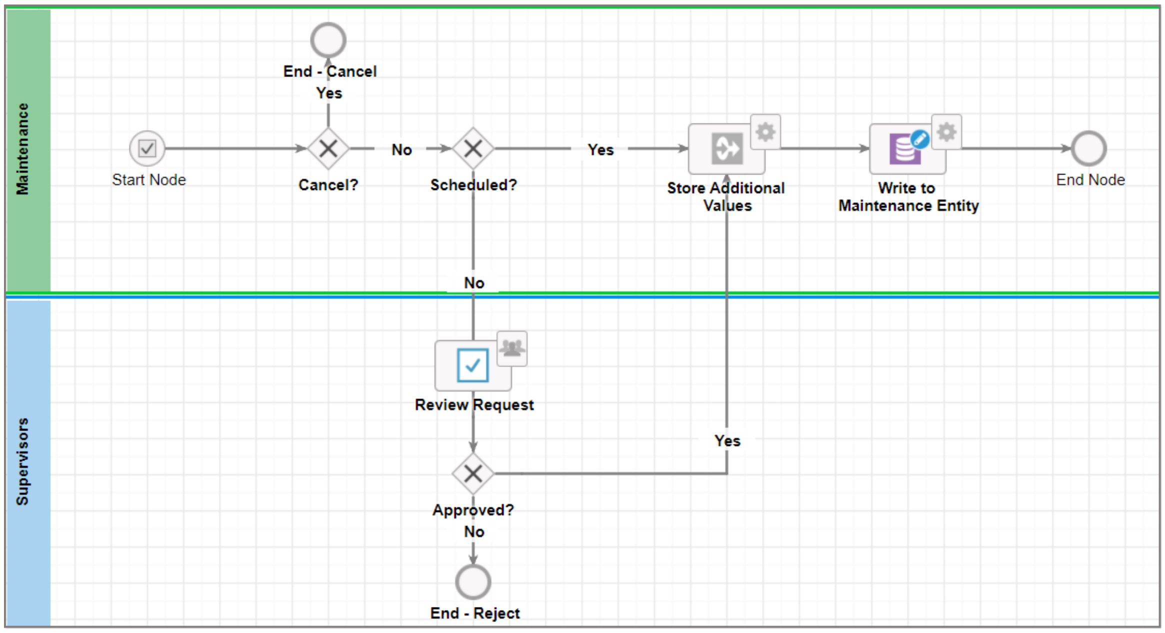
   - Update the label to **Approve**.
   - Click the **Expression Editor** icon next to the **Value** field, and type **true**.
   - In **Save Value To**, select **ri!approvalDecision**.
   - Make sure the **Submit** checkbox is selected.

10. Click **Save Changes**.

Create the Request Maintenance Process Model

In this section, you will create a new process model that will automate the steps for requesting vehicle maintenance at Acme Automobile. The process starts with a Mechanic filling out a maintenance request. Once a request is submitted, the process checks whether a given request is a scheduled maintenance event or an unscheduled event that requires a Supervisor's approval. While a scheduled maintenance event is written to the database without any further delay, an unscheduled request is first sent to a Supervisor for approval.

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



Create the Process Model

1. In your application, click **New** and select **Process Model**. Name this process **AX Request Maintenance**, and add a brief description: "Process model for requesting maintenance for a vehicle in the fleet." Click **Create**.

2. Unlike with the auto-generated process model, you will be prompted to secure this new process. Add **AX Administrators** as **Administrators** and **AX All Users** as **Viewers**. Click **Save**.

3. Add the swimlanes for Maintenance and Supervisors. Click the **Add Horizontal Lane** button on the toolbar to add two swimlanes for Maintenance and Supervisors. Click **Lane 1** and **Lane 2** labels, and rename them to **Maintenance** and **Supervisors**.

Configure Process Properties

Now you can start configuring the process model properties, such as variables, alerts, and data management. Follow the steps below to configure the process properties.

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

2. Navigate to the **Process Start Form** tab. Type and select the **AX_RequestMaintenance** interface. Click **OK**, then **Yes** to automatically create process parameters.

3. 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**.

4. Navigate to the **General** tab, and click the **Expression Editor** icon next to the **Process Display Name** field. You will now edit the display name to show the specifics of each maintenance request. Enter the following expression:

   ```
   "Request Maintenance: " & pv!vehicle.vehicleYear & " " &
   pv!vehicle.vehicleMake & " " & pv!vehicle.vehicleModel
   ```

   Insert the three process variables by selecting them from the **Data** tab. Click **Save and Close.**

5. Navigate to the **Alerts** tab. Select the **Use custom error alert settings** radio button. Also, select the **Send alerts to the following users & groups** checkbox, and type **AX Administrators** into the adjoining field**.**

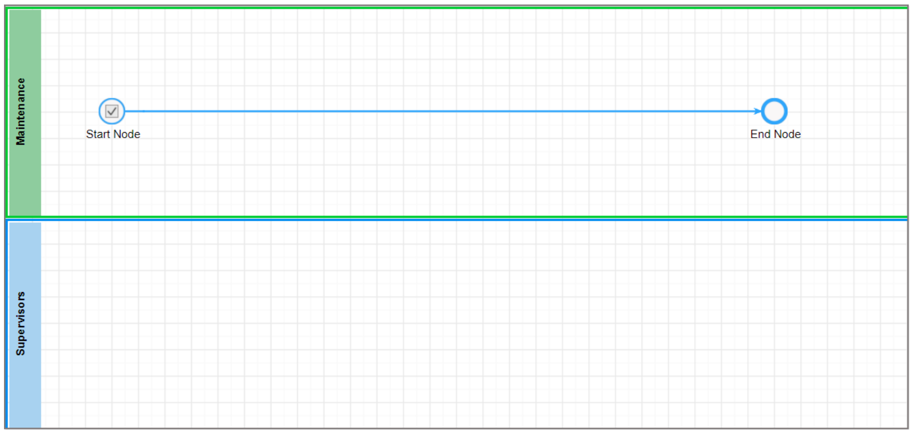
6. Navigate to the **Data Management** tab. Select **Archive processes**, and change it to **3** days.

7. Click **OK**, then **File > Save & Publish**.

8. Now that you have wired the form to the process, test your process model to check that everything works correctly and the process advances to completion. Follow the steps below to test the process:

   - Click **File > Start Process for Debugging**.
   - Complete the form, and click **Submit**. You will notice that a new process instance started in the **Monitoring Mode** and on a new tab in the Process Modeler.

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



- Click the **Process Details** button Process Details 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.
- Close the **Monitoring** tab, but keep the Process Model open to configure our next sequence of activities.

As you continue working on your process model, test frequently to ensure that the nodes are configured correctly, and all data is captured as expected.

Configure the Cancel XOR Gateway

In this process, if a Mechanic cancels the start form, the process will end. If a Mechanic submits the form, the process will continue to the next process node. Follow the steps below to configure the Cancel? gateway.

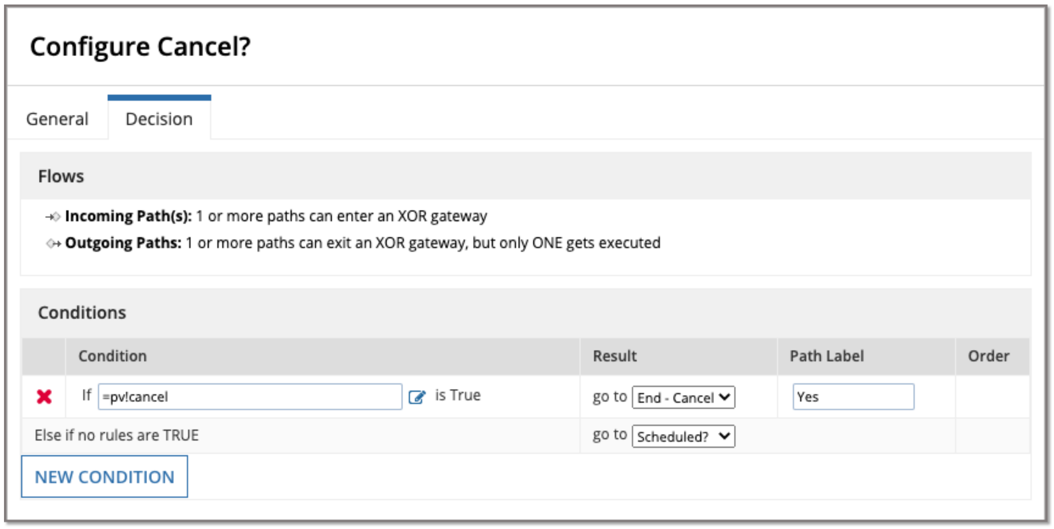
1. Drag and drop an **XOR Gateway** into the Maintenance swimlane to the right of the Start Node. Rename it to **Cancel?**
2. Drag and drop an **End Event** into the Maintenance swimlane above Cancel?. Rename it to **End - Cancel**. Connect the **Cancel?** and **End - Cancel** nodes.

   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.
3. Drag and drop an **XOR Gateway** into the Maintenance swimlane to the right of Cancel? Rename it to **Scheduled?**
4. Double-click the **Cancel?** gateway, and navigate to the **Decision** tab.

5. Click the **New Condition** button, and then click the **Expression Editor** icon next to the first condition. Select the **cancel** process variable from the **Data** tab, or simply type **pv!cancel**. Click **Save and Close**.

6. For the first condition, select **End - Cancel** in the **Result** column. Put **Yes** in the **Path Label** column.

7. For the second condition, select **Scheduled?** in the **Result** column.


Configure Cancel?

General | Decision

Flows

Incoming Path(s): 1 or more paths can enter an XOR gateway
Outgoing Paths: 1 or more paths can exit an XOR gateway, but only ONE gets executed

Conditions

	Condition	Result	Path Label	Order
✖	If =pv!cancel is True	go to End - Cancel	Yes	
	Else if no rules are TRUE	go to Scheduled?		

8. Click **OK.** Add the label **Yes** to the connector leading from **Cancel?** to **End - Cancel**, and the label **No** to the connector leading from **Cancel?** to **Scheduled?** To add labels, right-click each individual connector, and select **Add Label**.

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

Configure the Scheduled XOR Gateway

Next, configure the Scheduled? XOR gateway. This gateway will direct scheduled maintenance events to the Store Additional Values node and unscheduled events to the Review Request node. Follow the steps below to configure this gateway.

1. Drag and drop a **User Input Task** into the Supervisor swimlane below Scheduled? Rename it to **Review Request**. Connect the **Scheduled?** and **Review Request** nodes.

2. Drag and drop an **XOR Gateway** into the Supervisor swimlane below Review Maintenance Request. Rename it to **Approved?** Connect the **Review Request** and **Approved?** Nodes.

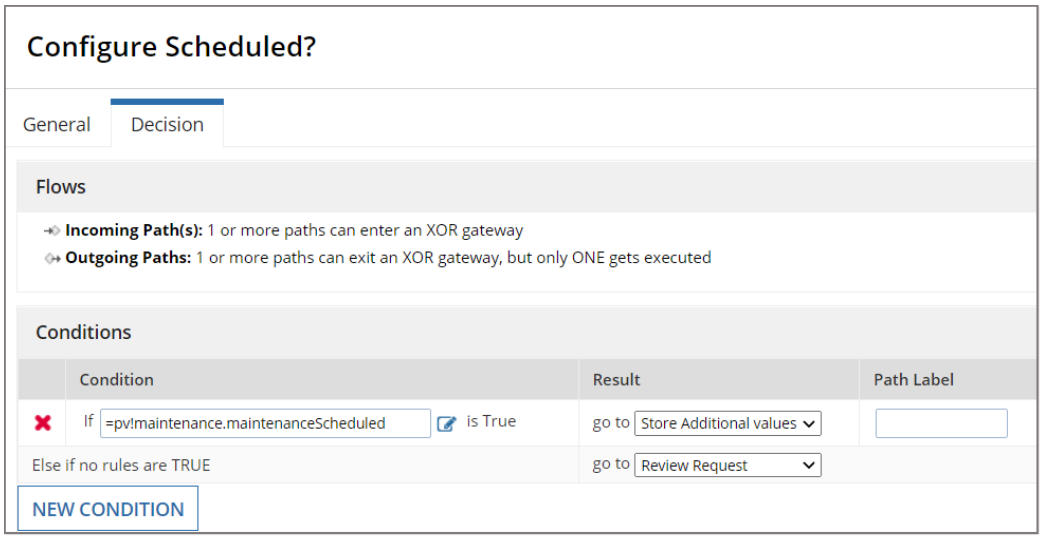
3. Drag and drop a **Script Task** into the Maintenance swimlane to the right of Scheduled? Rename it to **Store Additional Values**. Connect the **Approved?** and **Store Additional Values** nodes.

4. Double-click **Scheduled?**, and navigate to the **Decision** tab.

5. Click the **New Condition** button, and then click the **Expression Editor** icon next to the first condition. Click the **+** sign next to **maintenance** to expand it, and then select the **maintenanceScheduled** process variable. The expression should now be populated with **pv!maintenance.maintenanceScheduled**. Click **Save and Close**.

6. For the first condition, select **Store Additional Values** in the **Result** column. Put **Yes** in the **Path Label** column.

7. For the second condition, select **Review Request** in the **Result** column.


Configure Scheduled?

Flows

Incoming Path(s): 1 or more paths can enter an XOR gateway

Outgoing Paths: 1 or more paths can exit an XOR gateway, but only ONE gets executed

Conditions

	Condition	Result	Path Label
✖	If =pv!maintenance.maintenanceScheduled is True	go to Store Additional values	
Else if no rules are TRUE		go to Review Request	

8. Click **OK**. Add the label **Yes** to the connector leading from **Scheduled?** to **Store Additional Values**, and the label **No** to the connector leading from **Scheduled?** to **Review Request**. To add labels, right-click each individual connector, and select **Add Label**.

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

Configure the User Input Task

Next, let's configure the Review Request node. You will use this node to add the Approve Maintenance form to this process. Complete the steps below.

1. Double-click **Review Request**. 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 sert them into the expression. Your expression should look as follows:

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

Click **Save and Close**.

2. Navigate to the **Forms** tab. Type and select **AX_ApproveMaintenance** as the interface. Click **Yes** to automatically create node inputs.
3. Navigate to the **Assignment** tab. Assign this node to **AX Supervisors**.
4. Navigate to the **Data** tab. Under **Inputs**, click the first node input **vehicle**. For the **Value** field, select **vehicle**.
5. Click the second node input **maintenance**. For the **Value** field, select **maintenance**.
6. Click the third node input **approvalDecision**. For the **Save into** field, select **approvalDecision**. Note that you make your selection under the **Save into** field because this is the new information passed back from the user that you want to write to the database table.
7. Click **OK**, then **File > Save & Publish**.

Configure the Approved Gateway

If the Supervisor approves the maintenance request, the process will continue to the script task. If the Supervisor denies the maintenance request, the process will end. Follow the steps below to configure the Approved? gateway added earlier.

1. Drag and drop an **End Event** into the Supervisor swimlane below **Approved?** Rename it to **End - Reject**, and then connect these two nodes.
2. Double-click **Approved?**, and navigate to the **Decision** tab.
3. Click the **New Condition** button, and then click the **Expression Editor** icon next to the first condition. In the **Data** tab, select the **approvalDecision** process variable. Click **Save and Close**.
4. For the first condition, select **Store Additional Values** in the **Result** column. Put **Yes** in the **Path Label** column.
5. For the second condition, select **End - Reject** in the **Result** column.
6. Click **OK**. Add the label **Yes** to the connector leading from **Approved?** to **Store Additional Values**, and the label **No** to the connector leading from **Approved?** to **End - Reject**. To add labels, right-click each individual connector, and select **Add Label**, or double-click the connector to edit the label in the **Flow Properties**.
7. Click **File > Save & Publish**.

Configure the Script Task

For this process, you will use a script task to store six variables: Maintenance Last Modified Date, Maintenance Review Date, Maintenance Requester, Maintenance Last Modified By, Maintenance Status, and Vehicle ID. Follow the steps below to set up the Store Additional Values script task.

1. Double-click **Store Additional Values**, and navigate to the **Data** tab. Click the **Outputs** tab.

2. Click **New Custom Output**. You will want to set the **Maintenance Last Modified Date** as the current date. Click the **Expression Editor** icon, and type the function **today()**. Click **Save and Close**. For the **Target**, click the **inline arrow**, and hold your cursor over **maintenance**. Select **maintenance.maintenanceLastModifiedDate**.

3. Repeat the same steps for the **Maintenance Review Date**. Be sure to select **maintenance.maintenanceReviewDate** as the **Target**.

4. Click **New Custom Output**. You will now want to set the **Maintenance Requester** as the initiator of the process model. Click the **Expression Editor** icon, and type the function **pp!initiator**. Alternatively, click **initiator** under **Process Properties**. Click **Save and Close**. For the **Target**, select **maintenance.maintenanceRequester**.

5. Repeat the same steps for **Maintenance Last Modified By**. Be sure to select **maintenance.maintenanceLastModifiedBy** as the **Target**.

6. Click **New Custom Output**. You will now want to set the **Maintenance Status** as Scheduled. Click the **Expression Editor** icon, and type **"Scheduled"**. Click **Save and Close.** For the **Target**, select **maintenance.maintenanceStatus**.

7. Click **New Custom Output**. The last value to save is the **Vehicle ID** associated with the maintenance request. Click the **Expression Editor** icon, and then find **vehicle** under **Process Variables** on the left. Click the plus sign, then select **vehicleId**. The expression should be populated to look like this:

   ```
   pv!vehicle.vehicleId
   ```

   Click **Save and Close**. For the **Target**, select **maintenance.vehicleId**.

8. Click **OK**, then **File > Save & Publish**.

9. Test your process model using the steps below:

   - Click **File > Start Process for Debugging**.
   - Complete the **New Maintenance Request** form, selecting the unscheduled maintenance, and click **Submit**.
   - In the new **Monitoring** tab, click the **Refresh** button in the toolbar.
   - Right-click the **Review Request** task once it is highlighted in green, and select **View Form.**
   - Accept the task, and then click **Approve**.
   - Click **Refresh** in the toolbar. You will see the process advance to the **End Node**.
   - Click **Process Details**, and select the **Process Variables** tab to view the data that was collected.

- Repeat the testing steps, but using different selections. For example, request an unscheduled maintenance, and then click **Deny** instead of Approve. Make sure that the process flows correctly.

Configure the Write to Maintenance Entity Service

Now that you have two forms and a script task in your process, add and configure the Write to Data Store Entity smart service to ensure that all maintenance data is written to the database table. Follow the steps below to complete this task.

1. Drag and drop a **Write to Data Store Entity** node into the **Maintenance** swimlane, to the right of Store Additional Values. Rename it to **Write to Maintenance Entity**.
2. Double-click **Write to Maintenance Entity**.
3. Navigate to the **Data** tab, and then the **Inputs** tab.
4. Click **Data Store Entity**, and then the **Directory icon** next to the **Value** field.
5. Click **AX Data Store**, then select **AX_Maintenance**. Click **OK**.
6. Click **New Input**, and name it **maintenance**.
7. In the **Type** field, type and select **AX_Maintenance** once it appears.
8. In the **Value** field, use the dropdown arrow to select the process variable **maintenance**.
9. Navigate to the **Assignment** tab. Select the **Run as whoever designed this process model** radio button, and click **OK**.
10. Click **File > Save & Publish**.

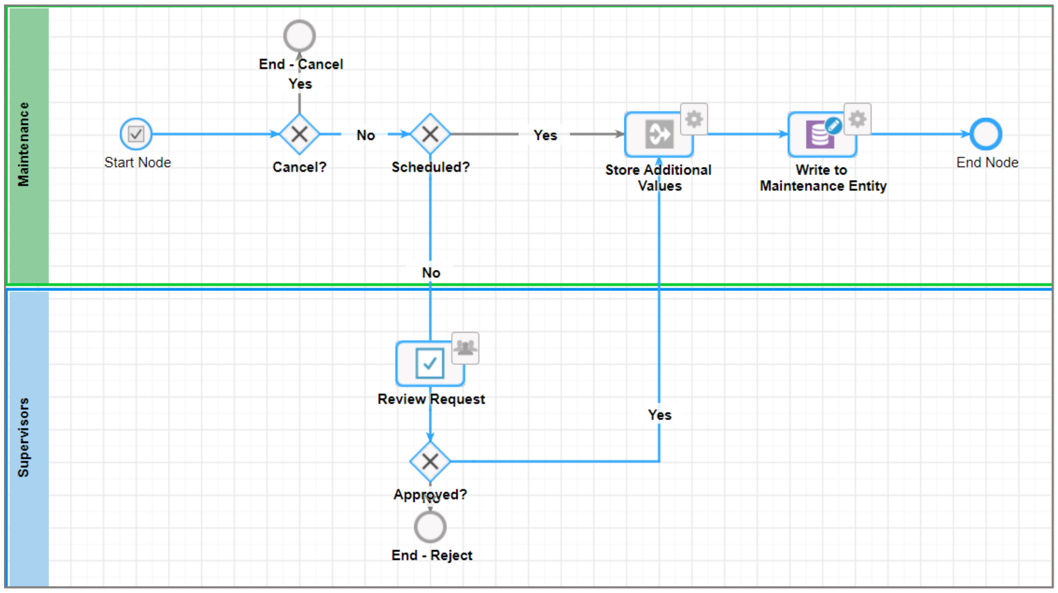
Test the Process

Let's test this process model one more time to ensure that it's working as expected. Keep in mind that when you work on an actual project, testing of this kind should be performed each time you add a new node to the process model. Follow the steps below to test the process model.

1. Click **File**, and select **Start Process for Debugging**.
2. Complete the **Request Maintenance** form, and click **Submit**.
3. Click the **Refresh** button in the toolbar to update the process flow. If you choose **Unscheduled** as the Maintenance Type, you will notice that the process will advance to the **Review Maintenance Request** node.

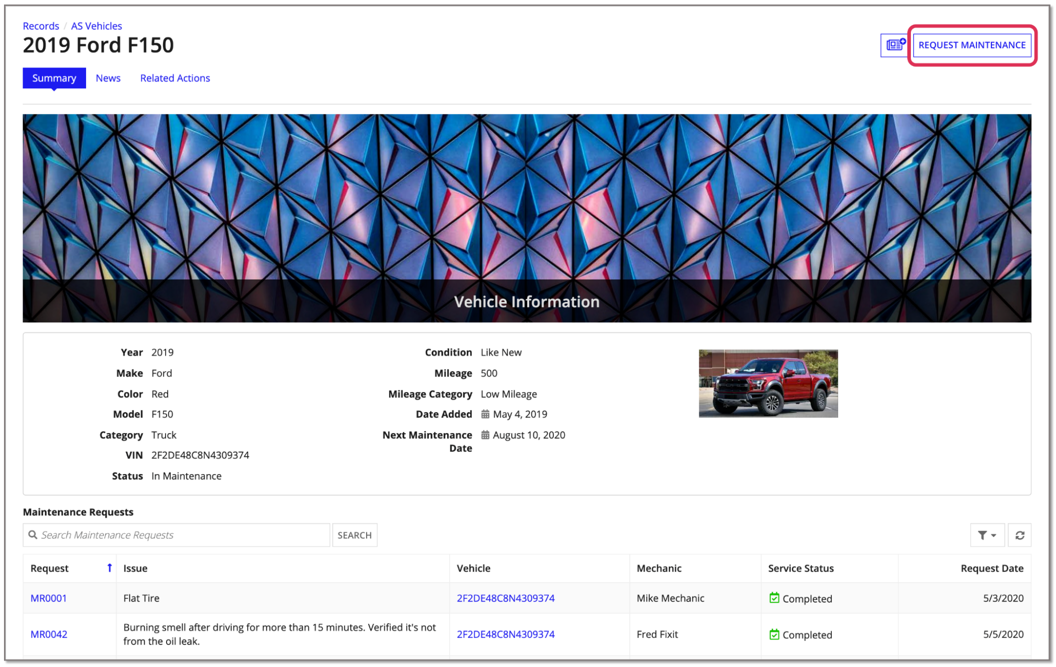
4. Right-click the **Review Maintenance Request** node, and select **View Form**. Click **Accept** to accept the task, and then click **Approve**.

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



Create a Related Action

In this exercise, you will create a related action that will enable mechanics to request maintenance from the summary view for a specific vehicle.


Records / AS Vehicles

2019 Ford F150

REQUEST MAINTENANCE

Summary News Related Actions

Vehicle Information

Year	2019	Condition	Like New
Make	Ford	Mileage	500
Color	Red	Mileage Category	Low Mileage
Model	F150	Date Added	May 4, 2019
Category	Truck	Next Maintenance Date	August 10, 2020
VIN	2F2DE48C8N4309374		
Status	In Maintenance		

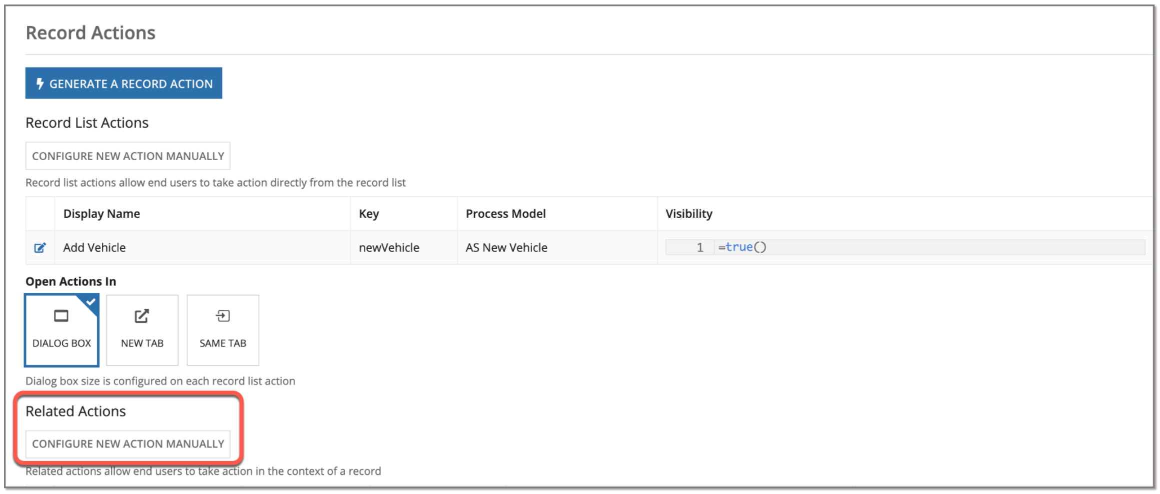
Maintenance Requests

Search Maintenance Requests SEARCH

Request	Issue	Vehicle	Mechanic	Service Status	Request Date
MR0001	Flat Tire	2F2DE48C8N4309374	Mike Mechanic	Completed	5/3/2020
MR0042	Burning smell after driving for more than 15 minutes. Verified it's not from the oil leak.	2F2DE48C8N4309374	Fred Fixit	Completed	5/5/2020

You will create this related action by adding the AX Request Maintenance process model to the AX Vehicle record. Follow the steps below to create a related action.

1. In the **AX Vehicle** record, navigate to the **Record Actions** tab.
2. Under **Related Actions**, click **Configure New Action Manually**:



Record Actions

GENERATE A RECORD ACTION

Record List Actions

CONFIGURE NEW ACTION MANUALLY

Record list actions allow end users to take action directly from the record list

	Display Name	Key	Process Model	Visibility
	Add Vehicle	newVehicle	AS New Vehicle	=true()

Open Actions In

Dialog box size is configured on each record list action

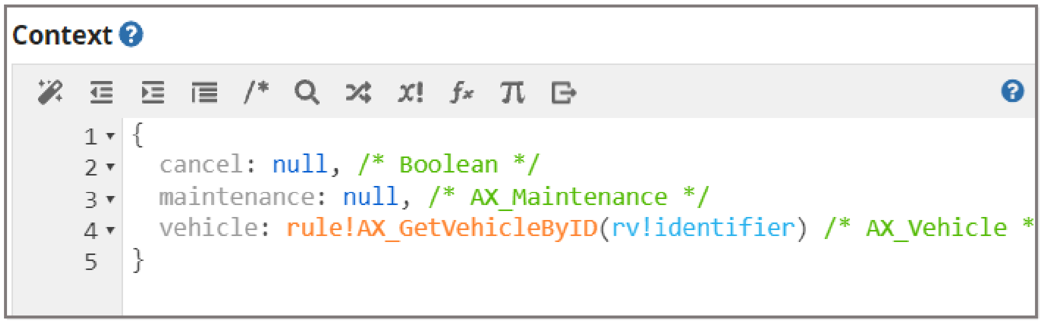
Related Actions

CONFIGURE NEW ACTION MANUALLY

Related actions allow end users to take action in the context of a record

3. Type **Request Maintenance** into the **Display Name** field, and add a simple description, "Request maintenance for a vehicle."
4. Select **Medium** for **Dialog Box Size**.
5. Select **AX Request Maintenance** for **Process Model**.
6. The Context variables will be generated for you. You use the Context box to pass in record information into the process model backing the related action. Note that the generated variables are the process variables from **AX Request Maintenance** that are set as parameters. Leave **cancel** and **maintenance** as **null**.
7. In the **Context** box, replace the **null** next to **vehicle** with the following expression:

```
rule!AX_GetVehicleByID(rv!identifier)
```



Context

```
{
  cancel: null, /* Boolean */
  maintenance: null, /* AX_Maintenance */
  vehicle: rule!AX_GetVehicleByID(rv!identifier) /* AX_Vehicle *
}
```

rv!identifier references a record's unique ID, which should always be the same as the primary key ID for the database table backing the record. In this case, since we are in the AX Vehicle record, this expression uses the unique vehicleId to pass vehicle information into the process model.

8. Now, you have to set the visibility of the related action. In the **Visibility** expression editor, type the following expression:

   ```
   a!isusermemberofgroup(loggedInUser(),
   cons!AX_MAINTENANCE_POINTER)
   ```

   This expression checks whether the logged in user is included in the AX Maintenance group or not. This related action is now only visible if the user is in the Maintenance group. Click **OK**.

9. Let's add the Related Action as a shortcut:

   - Navigate to the **Views** tab on the left hand side.
   - Click the **Edit** icon next to **Summary**.
   - Select the **Request Maintenance** checkbox under **Related Actions Shortcuts**.
   - Click **OK**, then **Save Changes**.

10. To test this feature from the business user perspective, click the link located next to the record name. Select any vehicle. The Request Maintenance related action button will be in the upper right corner. Test the flow to see how it works.