

Appian Step-by-Step #9

Exercise to Accompany
Interfaces 101

The Appian Step-by-Step series consists of 13 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.

- 1** Welcome to the Appian Developer Learning Path
- 2** Create an Application
- 3** Manage Users and Groups
- 4** Expressions
- 5** Records Part 1: Accessing Your Data
- 6** Records Part 2: Record Type Relationships
- 7** Sites
- 8** Query Your Data
- 9** **Interfaces**
- 10** Process Modeling 101: Part 1
- 11** Process Modeling 101: Part 2
- 12** Reports
- 13** Task Report

Exercise 9: Interfaces	2
Update the Summary View Interface	3
Add a Vehicle Image	4
Add a Read-Only Grid	6
Update Category, Status, and Condition Fields	7
Edit the Add Vehicle Form	8
Update Fields and Labels	8
Add a File Upload Component	10
Add Dropdown Components	10
TIP: Use the Visibility Setting to Conditionally Show Interface Components	11
Configure the Submit Button	12
Create the Supervisor Approval Form	12
Create a Reusable Interface	16
Troubleshooting Resources	17

Notice of Rights

This document was created by Appian Corporation, 7950 Jones Branch Dr, Tysons, Virginia 22102. Copyright 2023 by Appian Corporation. All rights reserved. Information in this document is subject to change. No part of this document may be reproduced or transmitted in any form by any means, without prior written permission of Appian Corporation. For more information on obtaining permission for reprints or excerpts, contact Appian Training at academyonline@appian.com.

This exercise was developed for **Appian 23.2**. If you are using a different version, please see **Appian Community** for information about updating your environment to the current version.



Exercise 9: Interfaces

In this exercise, you will modify or create the following interfaces:

1. AX_VehicleSummary
2. AX_AddVehicle
3. AX_SupervisorForm

AX_VehicleSummary is used in the Summary View to display read-only vehicle information.

AX_AddVehicle is used to start the AX Add Vehicle process model to add a new vehicle to the fleet.

AX_SupervisorForm is used by supervisors in the Add Vehicle process model to review a new vehicle.

To create efficient, intuitive, and attractive interfaces, it is important to follow UX design best practices and recommendations. Visit the SAIL Design System in [Appian Documentation](#) to learn more.

Update the Summary View Interface

When you generated the vehicle summary view in an earlier exercise, the AX_VehicleSummary interface was created for you. In this section, you will update this interface to better display the vehicle summary information.

Follow the steps below to update this interface.

1. In your application, open **AX_VehicleSummary**.
2. In the **Rule Inputs** pane, click **record**.



3. In the **Edit Rule Inputs** dialog, configure the following properties:
 - **Name:** Change the name to **vehicle**.

- **Description:** Enter Accepts a single vehicle of the AX Vehicle record data type.
 - Click **OK**.
4. Click **SAVE CHANGES** to check for dependencies. One object requires a manual update.
- In a new tab, open the **AX Vehicle** record type.
 - Go to **Views and Header**.
 - Next to the **Summary** view name, click **Edit**.
 - Update the underlined record to vehicle.

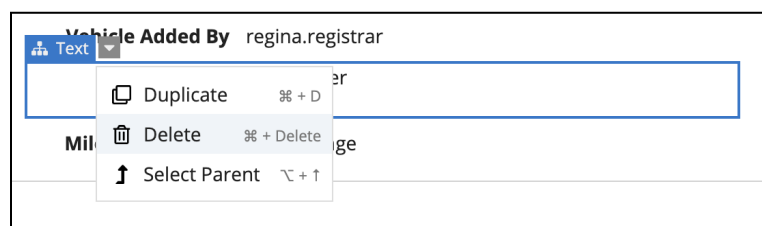


```
rule!AX_VehicleSummary(vehicle: rv!record)
```

- Click **OK**, then **SAVE CHANGES**.

Next, you will edit the interface components.

1. Return to **AX_VehicleSummary**.
2. Double-click the **Section Layout** title, and delete the **Vehicle** text.
3. In the **Component Configuration** pane, under **Divider Line**, select **None**.
4. In the **Components** tab of the **Palette**, find the **CARD** layout. Drag and drop it above vehicle details, outside of the section layout.
5. Drag and drop the **Section Layout** with the vehicle details into the **Card Layout**.
6. Click the **Card Layout**, and configure the following:
 - Delete the following fields: **Vehicle Last Maintenance Date**, **Vehicle Last Modified Date**, **Vehicle Added By**, **Vehicle Image**, and **Vehicle Last Modified By**. To remove fields, click an individual text box, and select **Delete** from the dropdown menu.



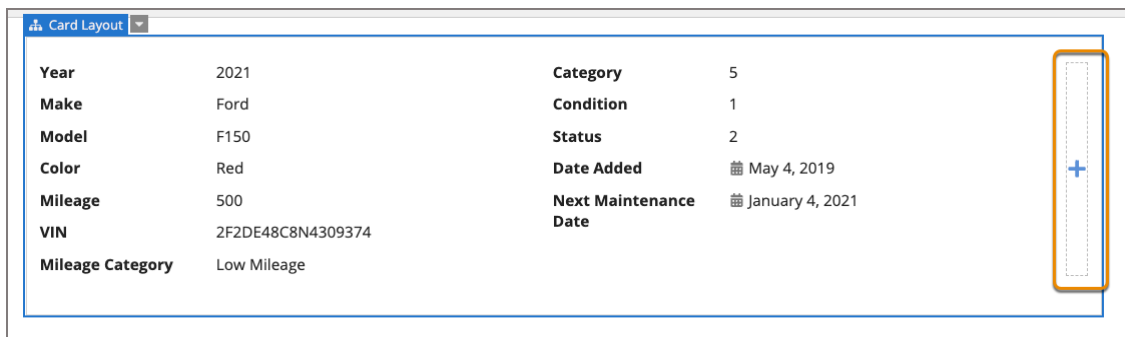
- Practice rearranging components by dragging and dropping the text boxes. Rearrange the fields to match the following layout:

Vehicle Year	2021	Vehicle Category	5
Vehicle Make	Ford	Vehicle Condition	1
Vehicle Model	F150	Vehicle Status	2
Vehicle Color	Red	Vehicle Date Added	📅 May 4, 2019
Vehicle Mileage	500	Vehicle Next Maintenance Date	📅 January 4, 2021
Vehicle Vin	2F2DE48C8N4309374		
Mileage Category	Low Mileage		

- Delete **Vehicle** from the labels.
- Click **SAVE CHANGES**.

Add a Vehicle Image

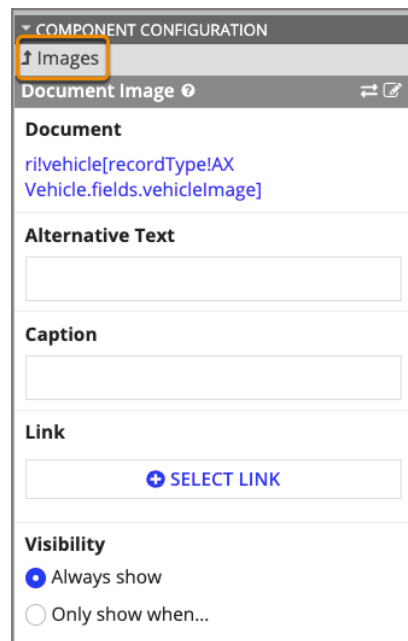
- Next to the second column, click **+** to add a third column.



- Drag and drop an **IMAGE** component into the new column.
- Click **+** in the **Image** component, and select **DOCUMENT IMAGE**.
- In the **Component Configuration** pane, under **Document**, click **a!EXAMPLE_DOCUMENT_IMAGE()**. Delete the expression, and enter:


```
ri!vehicle[recordType!AX Vehicle.fields.vehicleImage]
```
- Click **OK**.
- On the canvas, delete the **Image** label. Alternatively, you can remove the label in the **Image Component Configuration** pane by deleting the text in **Label**.

- Configure the image's visibility. You only want the image to display when the image field is neither null nor empty. If there is no image to display, there will be an error.
 - In the **Component Configuration** pane, click on **Images** twice to return to the Image component.



- Under **Visibility**, select **Only show when**.
- Click **Edit Condition**.
- In the Expression Editor, enter the following expression:

```
a!isNotNullorEmpty(ri!vehicle[recordType!AX  
Vehicle.fields.vehicleImage])
```

- Click **OK**.
- Click **TEST** to review the test data used to populate the interface.
 - The expression in the Expression Editor uses `a!queryRecordType` to query the vehicle record type and return a row of vehicle data. Update the expression to use the query rule you created in a previous exercise. Delete the expression, and enter:

```
rule!AX_QR_GetVehicleById(vehicleId: 1)
```

- Click **SET AS DEFAULTS AND TEST**, then **SAVE CHANGES**.

Add a Read-Only Grid

In this section, you will add a read-only grid with the vehicle's maintenance data to the AX_VehicleSummary interface.

1. In **AX_VehicleSummary**, drag and drop a **READ-ONLY GRID** component under the **Card Layout** with the vehicle details.
2. Click the **Read-only Grid**, and make the following changes in the **Component Configuration** pane:
 - **Data Source:** Select **Record Type**, and search for the **AX Maintenance** record type.
 - Click **FILTER RECORDS**. To display the maintenance requests for the selected vehicle, filter the record based on **vehicleId**.
 - In the **Filter Records** dialog, click **Add Filter**. For **Field**, select **vehicleId**. For **Condition**, select **=**.
 - For **Value**, click **123 > Expression**.
 - Click the **null** link. In the Expression Editor, enter:


```
ri!vehicle[recordType!AX Vehicle.fields.vehicleId]
```



```
1 ▾ ri!vehicle[ri!AX Vehicle.vehicleId]
```

- Click **OK**. You will see that the read-only grid now only displays the maintenance requests that are associated with the vehicle record's vehicle ID.
 - At the bottom of the **Component Configuration** pane, click **LAYOUT**. Change **Label** to **Maintenance Requests**.
3. Click **SAVE CHANGES**.

NOTE: The Maintenance Requests grid is based on the current configuration of the AX Maintenance record list. If you want, you can adjust the record list. Refer to Exercise 5: Records Part 1 to review configuring a record list. If you do make changes, re-add this read-only grid to see your changes.

Year	2021	Category	Convertible	
Make	Ford	Condition	Like New	
Model	F150	Status	Inactive	
Color	Red	Date Added	May 4, 2019	
Mileage	500	Next Maintenance Date	January 4, 2021	
VIN	2F2DE48C8N4309374			
Mileage Category	Low Mileage			

Maintenance Requests

Q Search AX Maintenances SEARCH ▼ ↺

Request ID	Request Date	Issue	Status	Start Date	Actual Complete Date	Duration	Cost	Requester	Mechanic	Last Modified By	Last Modified Date
1	5/3/2020	Flat Tire	Completed	5/5/2020	5/8/2020	4	50	fred.fixit	mike.mechanic	mike.mechanic	5/5/2020
42	5/5/2020	Burning smell after driving for more than 15 minutes. Verified it's not from the oil leak.	Completed	5/11/2020	5/18/2020	7	50	mike.mechanic	fred.fixit	suzanne.supervisor	5/8/2020
43	5/7/2020	Seat belt problem on driver's side. Problem is with latch.	Completed	5/11/2020	5/15/2020	4	200	mike.mechanic	mike.mechanic	suzanne.supervisor	5/8/2020
59	5/12/2020	Water leaking inside car	Completed	5/12/2020	5/14/2020	2	900	regina.registrar	mike.mechanic	regina.registrar	5/12/2020
60	5/12/2020	The brakes are noisy.	Completed	6/1/2020	6/1/2020	1	70	mike.mechanic	mike.mechanic	suzanne.supervisor	5/12/2020
114	6/8/2020	Oil leak	Completed	6/8/2020	6/8/2020	1	120	mike.mechanic	mike.mechanic	suzanne.supervisor	6/8/2020

6 items

Update Category, Status, and Condition Fields

Next, you will update the Category, Status, and Condition fields in the AX_VehicleSummary interface. Currently, these fields display numeric IDs. You will use record type relationships to replace them with the user-friendly values in the associated reference tables.

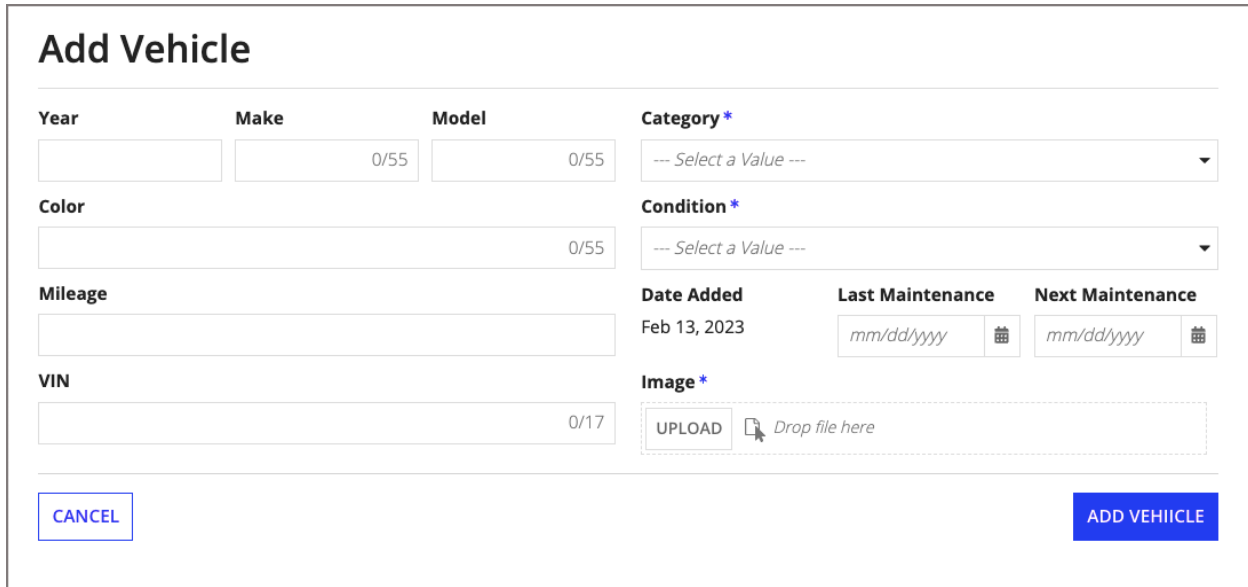
1. Click the **Category** text component.
2. Under **Display Value**, click the link.
3. In the Expression Editor, delete the text, and enter the following expression:

```
a!defaultValue(ri!vehicle[recordType!AX
Vehicle.relationships.toCategoryRecord.fields.value], "-")
```

4. Click **OK**.
5. Follow steps 1–4 to update the **Status** and **Condition** text components. When selecting the relationship, for Status, select **toStatusRecord**. For Condition, select **toConditionRecord**.
6. Click **SAVE CHANGES**.
7. Go to the AX Vehicle record type, and next to the record type name, click **View Record List**. Click on a **VIN** to view the updated summary view interface.

Edit the Add Vehicle Form

In this section, you will edit the generated AX_AddVehicleForm interface that starts the AX Add Vehicle process. Your goal is to improve the overall UX design of this form and to update a few fields so that they display the information correctly. When you are done, the interface will look like the image below.



The screenshot shows a form titled "Add Vehicle" with the following fields and controls:

- Year:** Text input field.
- Make:** Text input field with a character count of 0/55.
- Model:** Text input field with a character count of 0/55.
- Category*:** Dropdown menu with the text "-- Select a Value --".
- Color:** Text input field with a character count of 0/55.
- Condition*:** Dropdown menu with the text "-- Select a Value --".
- Mileage:** Text input field.
- Date Added:** Text input field containing "Feb 13, 2023".
- Last Maintenance:** Text input field with a date format mask "mm/dd/yyyy" and a calendar icon.
- Next Maintenance:** Text input field with a date format mask "mm/dd/yyyy" and a calendar icon.
- VIN:** Text input field with a character count of 0/17.
- Image*:** File upload area with an "UPLOAD" button and a "Drop file here" instruction.

At the bottom of the form, there are two buttons: "CANCEL" on the left and "ADD VEHICLE" on the right.

Follow the steps below to update the Add Vehicle form.

Update Fields and Labels

1. Open **AX_AddVehicleForm**.
2. In the **Rule Inputs** pane, click the **record** rule input. Configure the following properties:
 - **Name:** Change the name to `vehicle`.
 - **Description:** Enter a simple description.
3. Click **SAVE CHANGES** to check for dependencies after changing the rule input name. A manual change is required in the AX Add Vehicle process model. You will make this change in the next exercise.
4. Change the form title to `Add Vehicle`.
5. Delete the following text components: Vehicle Status, Vehicle Added By, Vehicle Last Modified By, Vehicle Condition, Vehicle Category, Vehicle Image, and Vehicle Last Modified Date.

6. Remove redundant labels. All default labels in this form contain **vehicle**, which is unnecessary in the form. Also, remove the word **date** from the Last Maintenance and Next Maintenance field labels.
7. Drag and drop a **SIDE BY SIDE** layout inside of and at the top of the first column. Move the **Year**, **Make**, and **Model** fields inside of the Side By Side layout. The Side By Side layout will keep these fields next to each other on a narrower screen. For reference, use the image at the beginning of this section.

NOTE: On mobile devices, columns layouts are flattened into a single column. If you need certain fields to stay together, use side-by-side layouts.

8. Drag and drop a **SIDE BY SIDE** layout into the second column. Move the **Date Added**, **Last Maintenance**, and **Next Maintenance** fields inside of it.
9. Next, add a validation to the Next Maintenance field so that the next maintenance date entered must be after the last maintenance date.
 - Click **Next Maintenance**.
 - In the **Component Configuration** pane, find **Validations**. Next to **Validations**, click the **Edit as Expression** icon.
 - In the Expression Editor, enter the following expression:

```
if(todate(ri!vehicle[recordType!AX
Vehicle.fields.vehicleNextMaintenanceDate]) <
todate(ri!vehicle[recordType!AX
Vehicle.fields.vehicleLastMaintenanceDate]), "The next
maintenance date must be after the last maintenance date.",
null)
```

This expression checks whether the next maintenance date is before the last maintenance date. If it is, the field is invalid and displays an error message.

10. Click **OK**, then **SAVE CHANGES**.

Add a File Upload Component

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

1. Drag and drop a **FILE UPLOAD** component into the second column. Change the **Label** to Image.
2. In the **Component Configuration** pane, configure the following:
 - **Target Folder:** Enter and select the constant **AX_DOCUMENTS_FOLDER_POINTER**.
 - **Maximum Selections:** Enter 1.

- **Selected Files** and **Save Files To:** Select **ri!vehicle.vehicleImage**.
- Select the **Required** checkbox.

Add Dropdown Components

In this section, you will add the Category and Condition fields as Dropdown components. You will populate these dropdowns using data stored in the reference database tables from the Acme Automobile Reference Application.

Follow the steps below to add dropdown components.

1. Drag and drop two **DROPDOWN** components into the second column. Change the **Labels** to Category and Condition.

To use the values stored in the reference tables, you need to create two local variables. Complete the following steps to add local variables:

1. In the **Local Variables** pane, click **New Local Variable**.
2. Configure fields for the first local variable
 - **Name:** category
 - **Value:** rule!AA_QR_REF_getVehicleCategory
3. Click **CREATE AND ADD ANOTHER**.
4. Configure fields for the second local variable
 - **Name:** condition
 - **Value:** rule!AA_QR_REF_getVehicleCategory
5. Click **CREATE**.

Next, configure the Category field to display correct choices in the dropdown menu.

1. Click the **Category** field, and in the **Component Configuration** pane, configure the following properties:
 - **Choice Labels:** Click the **Edit as Expression** icon. Enter `local!category[recordType!AA REF Vehicle Category.fields.value]`.

An error message will appear because the choiceLabels and choiceValues arrays are currently different lengths. This message will disappear after the next step.
 - **Choice Values:** Click the **Edit as Expression** icon. Enter `local!category[recordType!AA REF Vehicle Category.fields.id]`.
 - **Selected Value** and **Save Selection To:** Select **ri!vehicle.vehicleCategory**.
 - Select the **Required** checkbox.
2. In the same way, configure the **Condition** field:

- **Choice Labels:** Enter `local!condition[recordType!AA REF Vehicle Condition.fields.value]`.
 - **Choice Values:** Enter `local!condition[recordType!AA REF Vehicle Condition.fields.id]`.
 - **Selected Value** and **Save Selection To:** Select `ri!vehicle.vehicleCondition`.
 - Select the **Required** checkbox.
3. Check that the dropdown selections display the correct choices. Click **SAVE CHANGES**.

TIP: Use the Visibility Setting to Conditionally Show Interface Components

- In some cases, you might want to display a component based on a condition.
- To do this, use the **Visibility** setting in the **Component Configuration** pane. Select **Only Show When** and enter an expression in the Expression Editor.
- For example, imagine that you want to display a Comments field if a vehicle's mileage is above 150,000 miles. You can enter the expression `ri!vehicle[recordType!AX Vehicle.fields.vehicleMileage] > 150000`.
- This expression determines whether the Comments component is displayed on the interface. When set to false, the component is hidden and not evaluated. The default is set to true.

Configure the Submit Button

Next, you will update the Date Added field to display the current date. Although the registrars will not manually fill out the vehicle added date, this date still needs to be recorded and saved to the database table. Because it will be a read-only field and not a user input field, you will need to configure the Submit button to save the Vehicle Date Added value. You will also configure the Submit button to save the Vehicle Added By value.

1. Click the **Date Added** field. In the **Component Configuration** pane, configure the following properties:
 - **Display Value:** Delete the text, and click the **Edit as Expression** icon. Enter `today()` to return the current date.
 - Select the **Read-only** checkbox.
2. Click the **CREATE** button. In the **Component Configuration** pane, configure the following properties:
 - **Label:** Change the label to `Add Vehicle`.
 - **Save Value To:** Click the **Edit as Expression** icon, and enter the expression below. As you are typing, use the suggested fields to ensure proper formatting.

```

{
  a!save(ri!vehicle[recordType!AX
Vehicle.fields.vehicleDateAdded], today()),
  a!save(ri!vehicle[recordType!AX
Vehicle.fields.vehicleAddedBy], loggedInUser())
}

```

```

1 {
2   a!save(
3     ri!vehicle[AX Vehicle.vehicleDateAdded],
4     today()
5   ),
6   a!save(
7     ri!vehicle[AX Vehicle.vehicleAddedBy],
8     loggedInUser()
9   )
10 }

```


In this expression, `ri!vehicle[AX Vehicle.vehicleDateAdded]` is the target and `today()` is the value being saved into the target. Similarly, `ri!vehicle[AX Vehicle.vehicleAddedBy]` is the target and `loggedInUser()` is the value being saved into the target.

3. Click **OK**, then **SAVE CHANGES**.

Create the Supervisor Approval Form

In this section, you will create an interface for the supervisor to review the new vehicles. After the registrar submits a new vehicle, the supervisor will receive a task to approve or reject it. Your interface will look like the image below:

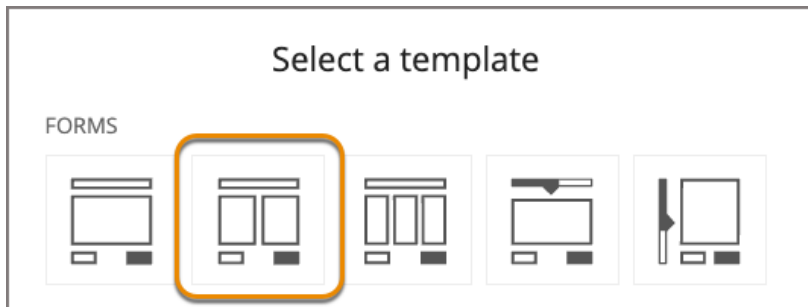
Approve Vehicle

<p>Year 2019</p> <p>Make Ford</p> <p>Model F150</p> <p>VIN 2F2DE48C8N4309374</p> <p>Mileage 500</p>	
--	---

REJECT
APPROVE

Follow the steps below to create the supervisor approval form and configure the rule inputs.

1. In the **Build** view of your application, click **New > Interface**. In the **Create Interface** dialog, configure the following properties:
 - **Name:** Enter `AX_SupervisorForm`.
 - **Description:** Enter `Displays read-only details about a selected vehicle for Supervisor approval.`
 - **Save In:** Select **AX Interfaces**.
2. Choose the **Two Column Form** template:



3. In the **Rule Inputs** pane, click the **New Rule Input** icon. Configure the following properties for the new rule input:
 - **Name:** Enter `vehicle`.
 - **Description:** Enter a simple description.
 - **Type:** Select the **AX Vehicle** record type.
4. Click the **cancel** rule input, and configure the following properties:
 - **Name:** Change the name to `approvalDecision`.
 - **Description:** Enter a simple description.
 - **Type:** Keep **Boolean** as the type.

Next, update the layout and configure the components.

1. Rename the form `Review Vehicle`.

2. Delete the top **Section Layout**. To delete it, click this section, then click the dropdown arrow > **Delete**.
3. For the bottom section layout, delete the blue **Section** label.
4. Drag and drop **three TEXT** and **two INTEGER input components** into the first column.
5. For the text fields, change the labels to **Make, Model, VIN**. For the integer fields, change the labels to **Year** and **Mileage**. For each of these fields, make the following changes in the **Components Configuration** pane:

- **Label Position:** Select **Adjacent**.
- **Display Value:** Select the correct values. For example, for Year, select **ri!vehicle.vehicleYear**.
- Select the **Read-only** checkbox.

6. Drag and drop an **IMAGE** component into the second column. Delete the label **Image**, and click + in the Image field. Select **DOCUMENT IMAGE**.
7. In the **Component Configuration** pane, under **Document**, click **a!EXAMPLE_DOCUMENT_IMAGE()**. In the Expression Editor, delete all text, and enter the following expression:

```
ri!vehicle[recordType!AX Vehicle.fields.vehicleImage]
```

You will see an error message. This error appears because the value for the document cannot be null.

8. To fix this error, add test values to the interface.
 - Click **TEST**, and enter the following expression into the vehicle row:

```
rule!AX_QR_GetVehicleByID(vehicleId: 1)
```
 - Click **SET AS DEFAULTS AND TEST**. You will see the error disappear, and the interface will display the sample data.
9. Click the **CANCEL** button, and rename it **Reject**. In the **Component Configuration** pane, under **Value**, click the **link**, and change it to **false**. Under **Save Value To**, keep **ri!approvalDecision**.
10. Click the **SUBMIT** button, and rename it **Approve**.
11. Configure the Approve button to save values for the following fields: Vehicle Last Modified By, Vehicle Last Modified Date, and Vehicle Status.
 - In the **Component Configuration** pane, for **Save Value To**, click the **Edit as Expression** icon.

- Enter the following expression:

```
{
    a!save(ri!vehicle[recordType!AX
Vehicle.fields.vehicleLastModifiedBy],
loggedInUser()),
a!save(ri!vehicle[recordType!AX
Vehicle.fields.vehicleLastModifiedDate], today()),
a!save(ri!vehicle[recordType!{AX
Vehicle.fields.vehicleStatus], 1),
a!save(ri!approvalDecision, true())
}
```

This expression saves:

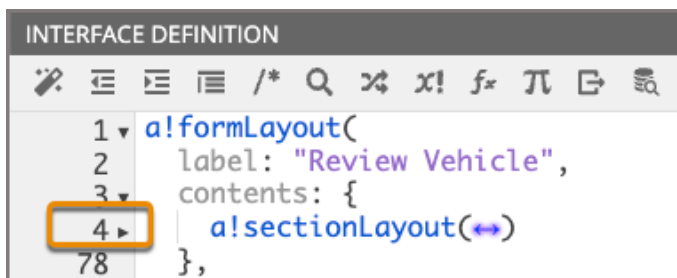
- The logged in user as the value for the Vehicle Last Modified By field,
- The current date as the value for the Vehicle Last Modified Date field,
- The active vehicle status (which has a value of 1 in the AA_REF_VEHICLE_STATUS table) as the value for the Vehicle Status field, and
- True() as the value for the Approval Decision rule input.

12. Click **SAVE CHANGES**.

Create a Reusable Interface

Keep in mind that interfaces can be built as reusable blocks. You can create an interface and then call it from other interfaces in your application. Try it out by creating a reusable interface from the Supervisor Review form.

1. Click **EXPRESSION MODE**.
2. In **line 4**, use the **arrow** to collapse the section layout:



3. Highlight the section layout:


```
INTERFACE DEFINITION
1 a!formLayout(
2   label: "Review Vehicle",
3   contents: {
4     a!sectionLayout(↔)
78  },
```

4. Click **Save Selected Expression As** in the toolbar:

```
INTERFACE DEFINITION
1 a!formLayout
2   label:
3   contents:
4   a!sectionLayout(↔)
78  },
```

5. In the **Save Expression As** dialog, configure the following properties:
 - **Save As:** Select **Interface**.
 - **Name:** Enter `AX_VehicleDetailsView`.
 - **Description:** Enter `Read only interface to display vehicle details`.
 - Click **SAVE**. The new interface will open.
6. In `AX_VehicleDetailsView`, in the top right corner, click the **Gear** icon > **Properties**.
7. Select the **Include in the design library** checkbox. Name the interface `AX_VehicleDetailsView`.
8. Click **OK**, then **SAVE CHANGES**. Close the `AX_VehiclesDetailsView` tab.
9. Go to the `AX_SupervisorForm`. You will notice the reusable interface has been added in place of the section layout and mapped to the appropriate rule input. Later, you will be able to reference this reusable interface anytime you want to display read-only vehicle details.

TIP: Creating reusable interfaces is useful for reporting interfaces. You can create each chart or grid as its own interface, and then combine them into different larger reporting interfaces for your business users.

Troubleshooting Resources

Stuck on a step, or need help troubleshooting? Appian provides several support resources that you can use as you build:

1. **Acme Auto Solution Application** - The Acme Auto Solution Application (AS) is the solution you are working to build in the Step-by-Steps, so you can use it as a reference tool. Review this application to see how specific objects are configured, or test the app to see how the features work from a business user's perspective. You can find this app in your Appian Community Edition environment.
2. [Community Discussions for New Users](#) - Check out the **New to Appian** thread in Community. Join our community of experts to ask questions and find answers from past discussions.
3. [Appian Documentation](#) - Appian's product documentation will provide you with an overview of key Appian features, newest release information, additional tutorials, and helpful patterns and recipes to implement in your app.