Appian Academy

Build an Application: Step-by-Step #5

Exercise to Accompany
Design Record Types: Model and Configure Your Data

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.

This exercise was developed for **Appian 25.2**. If Appian Community Edition is on a later Appian version, functionality might be different. Go to academy.appian.com to download the latest exercise.

Exercise 5: Design Record Types

In this exercise, you will create two record types and their associated views and actions so business users can view information about the vehicle fleet and take action. You will complete a few incremental steps to create your record types:

1. Create the W#SA Vehicle record type.
2. Configure vehicle record events.
3. Create a custom record field.
4. Configure the W#SA Vehicle record list.
5. Add a user filter.
6. Create the W#SA Maintenance record type.
7. Add record type relationships.
8. Generate a summary view interface to display vehicle data.
9. Generate an action to add vehicles to the fleet.

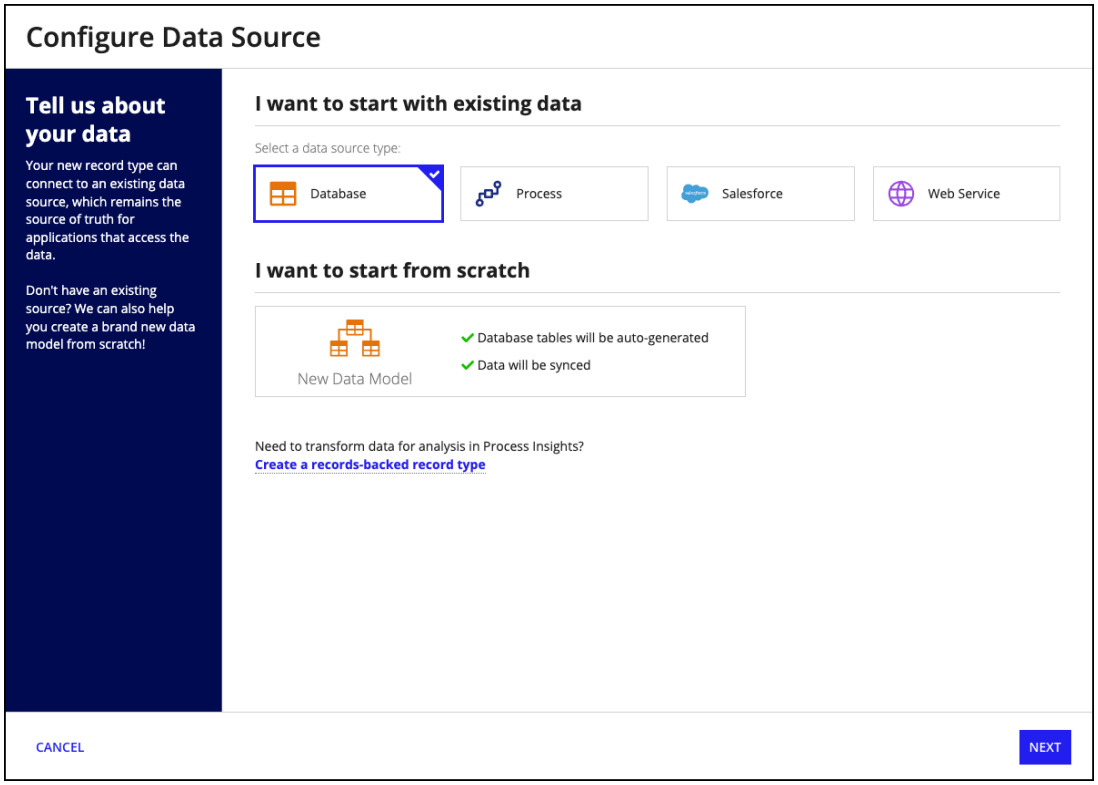
Create the W#SA Vehicle Record Type

Follow the steps below to create the W#SA Vehicle record type.

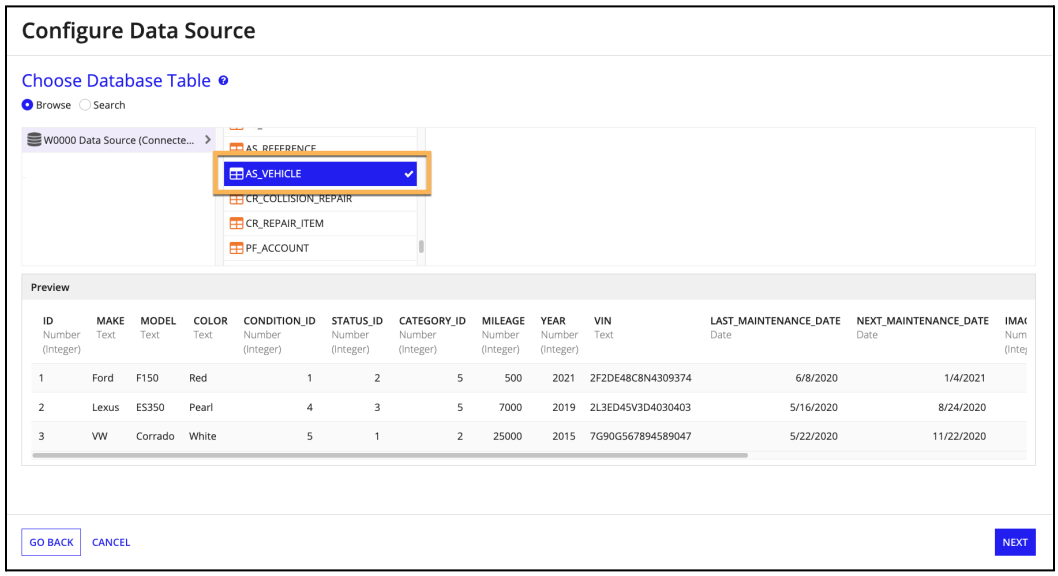
1. In the **Build** view of your application, click **NEW** > **Record Type**.
2. In the **Create Record Type** dialog, configure the following properties:
    - **Name**: Enter `W#SA Vehicle`.
    - **Plural Name**: Enter `W#SA Vehicles`. The plural name is what business users see on their sites, so the name should make sense to them. Note that application prefixes are not typically used for this field. In this exercise, you are using a prefix because a Vehicle record type already exists in the Acme Automobile Reference Application (AA).
    - **Description**: Enter `The list of vehicles managed by the W#SA application.`
3. Click **CREATE**.
4. In the **Review Record Type Security** dialog, click **SAVE**. You will use the default security, which should have W#SA Users as Viewers and W#SA Administrators as Administrators.

Next, configure the data source.

1. In the record type, click **TELL US ABOUT YOUR DATA**.
2. In the **Configure Data Source** dialog, ensure that **Database** is selected. Click **NEXT**.



3. Ensure **Build Apps Faster with Data Fabric** is selected. Click **NEXT**.
4. Under **Choose Database Table**, find and select **AS_VEHICLE**. Click **NEXT**.



5. On the **Configure Sync Filters** page, click **NEXT**. You will not create sync filters in this exercise.
6. Preview the fields and configure the following:
    - Using the dropdown, change the Record Field Type for **ADDED_BY** and **MODIFIED_BY** from **Text** to **User**.
7. Click **Finish**.
8. Click **SAVE** to save the record type before you build additional features.
9. Next to the record type name, click **View Record List** to view the record type from the business user perspective.

Tip: Use Security Rules to Configure Your Record Security Requirements

- You might have strict guidelines around who can see what data. With record-level security and field-level security, you can configure who can see which rows of your record data, and which fields.
- By default, any user with Viewer permission on the record type can see all available records.
- To configure security rules, click **Record-Level Security** or **Field-Level Security** in the left menu.
- For this exercise, you do not need to configure any security rules.

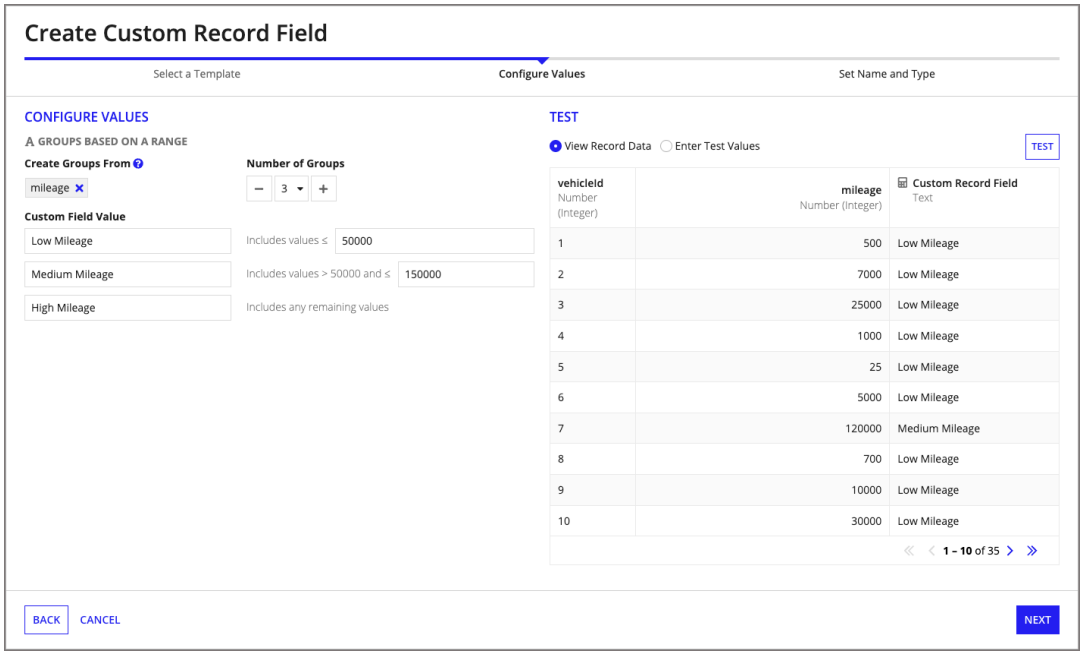
Create a Custom Record Field

Before you configure the record list, you will first create a custom record field to display the vehicle mileage category: Low Mileage, Medium Mileage, or High Mileage. Follow the steps below to create a custom record field.

1. On the **Data Model** page, click **NEW CUSTOM RECORD FIELD**.
2. Under **Select a Template**, select **Groups Based on a Range**. Click **NEXT**.
3. In the **Configure Values** section, under **Create Groups From**, select **mileage**.
4. Change **Number of Groups** to **3**.
5. Under **Custom Field Value**, configure the group names and values:
    - Name Group 1 `Low Mileage`, and set Upper Limit 1 to `50,000`.

- Name Group 2 `Medium Mileage`, and set Upper Limit 2 to `150,000`.
- Name Group 3 `High Mileage`.

6. Click **TEST** to preview the custom record field.



7. Click **NEXT**.
8. Under **Field Properties**, change **Record Field Name** to `mileageCategory`.
9. Click **CREATE**, then **SAVE**. Saving your changes syncs the vehicle data in Appian.

Configure Vehicle Record Events

Follow these steps to configure record events for the W#SA Vehicle record type.

1. In the left menu, go to the **Events** page.
2. Click **GENERATE EVENT RECORD TYPES**.
3. In the **Generate Event Record Types** dialog, include common event types by leaving the **Created Vehicle, Updated Vehicle**, and **Commented on Vehicle** checkbox selected.
4. Keep **Other Event Types** blank.
5. Keep the default record type names for **Event History**, **Event Type Lookup**, **Reply Thread**, and **Subscriber**.
6. Uncheck **Download database script**.

7. Click **GENERATE**.

Configure the Record List

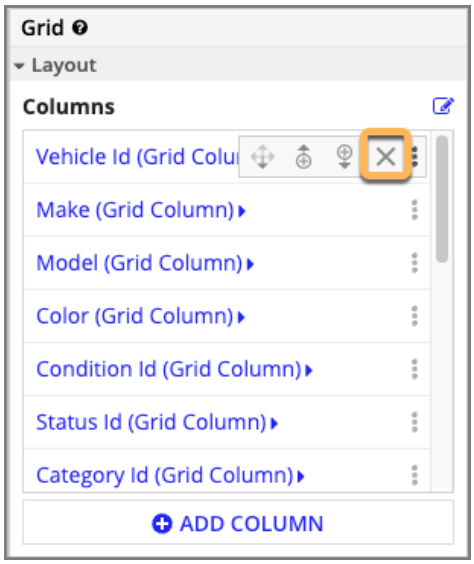
Next, you will edit the record list to display the following columns: Vehicle VIN, Make, Model, Year, Next Maintenance Date, Status, Added By, and Image. You will sort the list by the vehicle make, condition, category, and status. You will also add a clickable link to the vehicle VIN. Later, you will connect this link to a summary interface for each vehicle.

Follow the steps below to configure the record list.

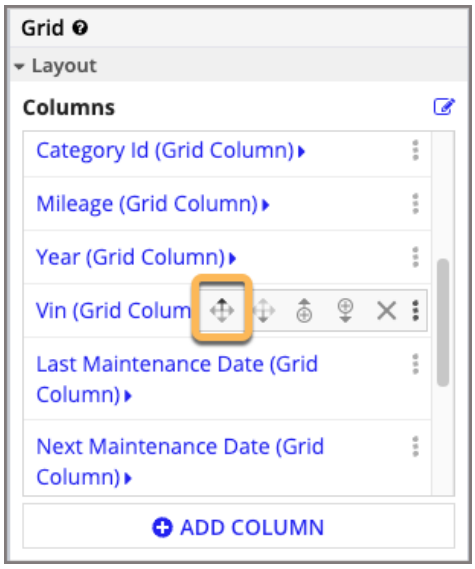
1. In the left menu, go to the **List** page.
2. Click **EDIT LIST**.
3. In the **Edit Record List** dialog, delete fields that do not need to be shown in the record list. To delete fields, click the in-line **X** next to those fields.

   Delete the following fields: Id, Color, Condition Id, Status Id, Category Id, Mileage, Last Maintenance Date, Added On, and Modified.

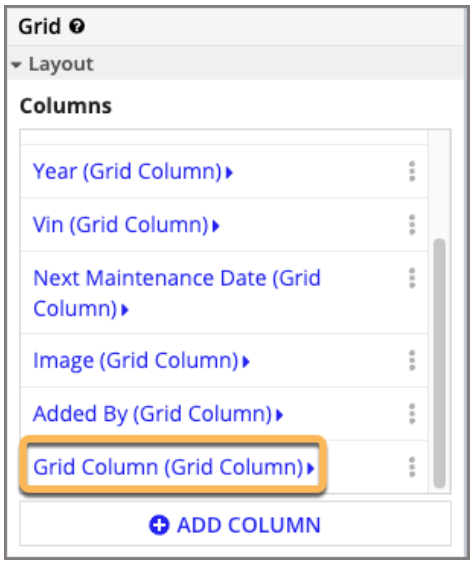
   You should be left with: Make, Model, Year, Vin, Next Maintenance Date, Image, and Added By.



4. Move the **Vin** column up so that it is the first column in the grid. Click the in-line **arrow** to move this column.



5. Add a new column for the mileage category. Click **ADD COLUMN**, and click the new **Grid Column** link to drill into the column.



6. Configure the following properties:

    - **Label**: Change the label to `Mileage Category`.
    - **Sort Field**: Select **mileageCategory**.
    - **Display Value**: Select **mileageCategory**.

7. In the left navigation, click **Grid**.

8. Move the **Image** column so that it is the last column in the grid. Use the in-line **arrow** to move this column.
9. Click **OK**, then **SAVE**.

Edit Record List Columns

Now that you have the columns you need for your record list, you will edit them. Follow the steps below to transform the VIN field into a clickable link.

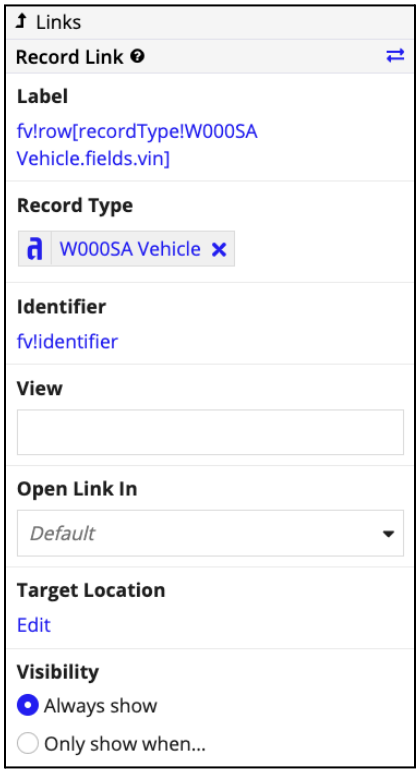
1. Click **EDIT LIST**.
2. In the left pane, under **Columns**, click **Vin**.
3. Change the **Label** to **VIN**.
4. Under **Display Value**, click **DISPLAY OPTIONS**.
5. In the **Display Options** dialog, select **RECORD LINK**.
6. Under **Display Value**, click **Link**.
7. Under **Links**, click **List of Links**.
8. Click **Record Link (Record Link)**.
9. Next to **Label**, click the **Edit as Expression** icon.
    - In the Expression Editor, delete the text.
    - Enter `fv!`, and select **row**.
    - Add a square bracket `[`, and start entering `vin`. Select **vin** when it appears.
    - Your expression will look like this:

      ```
      fv!row[W#SA Vehicle.vin]
      ```

    - Click **OK**.
10. In the **Record Link** section, under **Record Type**, enter and select **W#SA Vehicle**.
11. In the **Record Link** section, next to **Identifier**, click the **Edit as Expression** icon. In the Expression Editor:
    - Delete the text.
    - Enter `fv!`, and select **identifier**.
    - Click **OK**.

Your configurations will look like the image below:



Follow the steps below to format the Image column.

1. Under **Columns**, click **Image**.
2. Under **Display Value**, click **DISPLAY OPTIONS**.
3. Select **DOCUMENT IMAGE**.
4. Under **Display Value**, click **Image**.
5. Under **Images**, click **Document Image**.
6. Under **Document**, click the **a!EXAMPLE_DOCUMENT_IMAGE()** to edit as expression.
    - In the Expression Editor, delete all text, and enter the following expression:

```
if(
  a!isNullOrEmpty(
    fv!row[recordType!W#SA Vehicle.fields.image]
  ),
  a!EXAMPLE_DOCUMENT_IMAGE(),
  fv!row[recordType!W#SA Vehicle.fields.image]
)
```

Replace the record type in the example with your Vehicle record type.

This expression checks whether the image field is null. If it is null, it displays an example document image. The example document image is displayed for existing vehicle data; you will be able to upload vehicle images when you add new vehicles to the fleet in later exercises.

- Click **OK**.

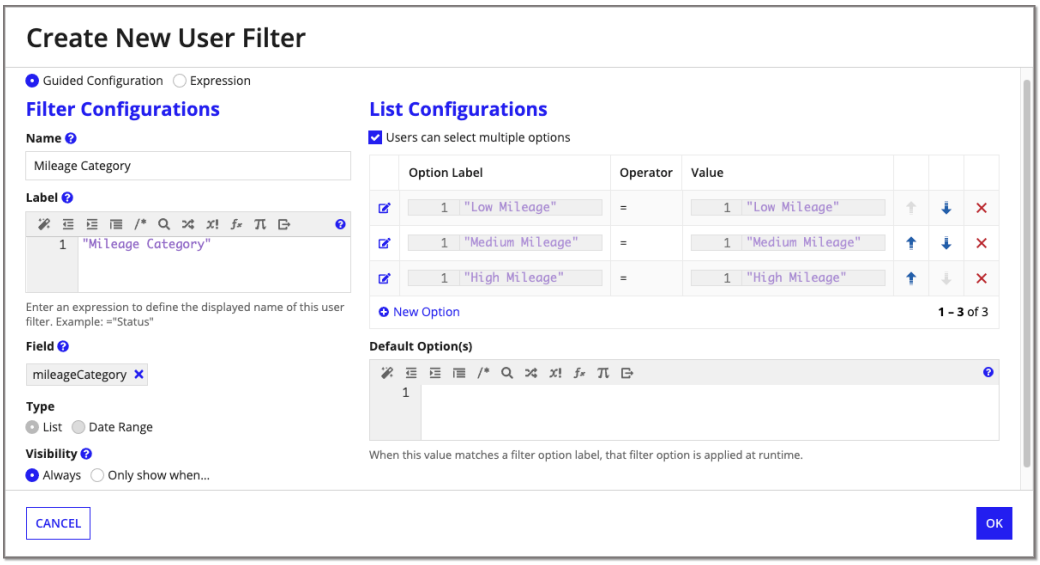
7. Click **OK**.
8. Click **SAVE**.

Add a User Filter

Next, you will add a user filter to the record list so business users can filter for vehicles by mileage category. Follow the steps below to create this user filter.

1. In the left menu, go to the **Filters** page.
2. In the **User Filters** section, click **New User Filter**.
3. Configure the following properties:
    - **Name**: Enter `Mileage Category`.
    - **Label**: Enter `“Mileage Category”`. Use quotation marks because it is an expression.
    - **Field**: Select **mileageCategory**.
    - Under **List Configurations**, click **New Option**.
        - **Option Label**: Enter `“Low Mileage”`.
        - **Value**: Enter `“Low Mileage”`.
        - Scroll down, and in the **Operator** field, leave the **=** option.
        - Click **SAVE FILTER OPTION**.

- In the same manner, create the **Medium Mileage** and **High Mileage** options.



4. Click **OK**, then **SAVE**.

5. Next to the record type name, click **View Record List** to test your user filter.

Add an Expression-Based User Filter

In this exercise, you will create an expression-based user filter to filter the W#SA Vehicle record type by vehicle make.

Follow the steps below to create an expression-based user filter.

1. In the **W#SA Vehicle** record type, go to the **Filters** page.

2. Click **New User Filter**.

3. Select **Expression**.

4. In **Name**, enter `Make`.

5. In **Filter Expression**, enter the expression below. This expression should be typed. If you copy and paste it, you will need to edit the lines with `recordType`.

```
a!localVariables(
  local!vehicleMakes: a!queryRecordType(
    recordType: recordType!W#SA Vehicle,
    fields: a!aggregationFields(
      groupings: {
        a!grouping(
          field: recordType!W#SA Vehicle.make,
```

```
            alias: "make"
          )
        },
        measures: {
          a!measure(
            field: recordType!W#SA Vehicle.make,
            function: "COUNT",
            alias: "count"
          )
        }
      ),
      pagingInfo: a!pagingInfo(
        startIndex: 1,

            batchSize: 5000
          )
        ),
        a!recordFilterList(
          name: "Make",
          options: a!forEach(
            items: local!vehicleMakes.data,
            expression: a!recordFilterListOption(
              id: fv!index,
              name: fv!item.make,
              filter: a!queryFilter(
                field: recordType!W#SA Vehicle.make,
                operator: "=",
                value: fv!item.make
              ),
              dataCount: fv!item.count
            )
          )
        )
      )
```

First, this expression creates a local variable for the record type query, so that it can be referenced later in the expression. In this case, `a!queryRecordType()` executes a query on the W#SA Vehicle record type and returns the vehicle makes. Instead of manually creating an option for each make, the `a!forEach()` function writes these options for you. It takes the array of vehicle makes returned in the query and passes them to an expression one at a time, creating the `a!recordFilterListOption` function for each item in the array.

Tip: Use Local Variables to Define and Store Temporary Values

- Local variables define and store temporary values within an expression. Use them when you only need values within a particular expression. For example, if you need to temporarily store user-inputted search terms in an interface, you can add a local variable, such as `local!search`.
- A local variable temporarily holds a value until you save it. For example, in an interface you can save a local variable using the relevant rule input.
- Use the following syntax to define a local variable: `a!localVariables(localVar1, localVarN, expression)`. `LocalVarN` means that you can have multiple local variables.

6. Click **OK**, and then **SAVE**.
7. Next to the record type name, click **VIEW RECORD LIST**. At the top of the record list, you will see the **Make** filter. Test that it works as expected.

Configure a Record Title

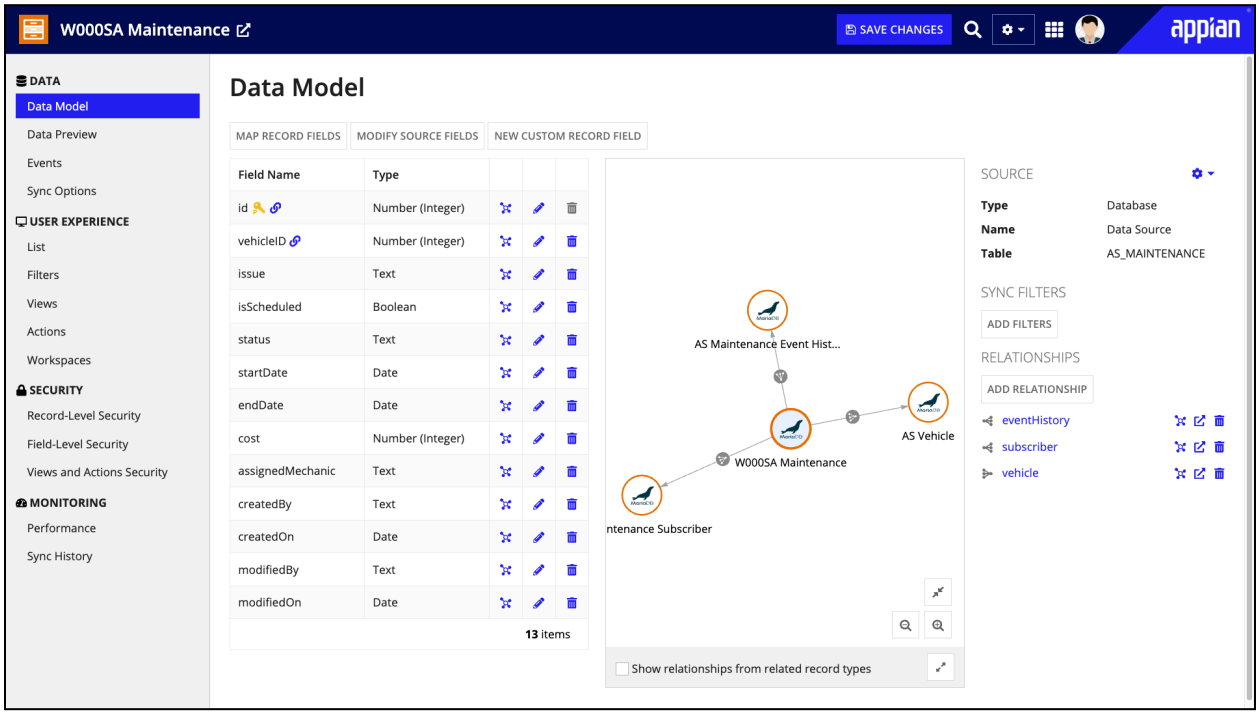
Next, you will configure the title of a vehicle's summary view.

1. In the **W#SA Vehicle** record type, go to the **Views** page.
2. Under **Header**, go to **Configurations** > **Record Title**.
3. Click the **X** next to **Make** to remove the current field being used as the title.
4. Using the dropdown, select **vin.**
5. Click **SAVE**.
6. Preview the vehicle summary view with the new record title.
    - Next to the record type name, click **View Record List**.
    - Select any vehicle to access its summary view. Your summary view should now display the vehicle's VIN.

    NOTE: The summary view will be blank. You will create the summary view later in this exercise.

Create the W#SA Maintenance Record Type

Next, you will create a second record type for the vehicle maintenance requests. Follow the steps below to create the W#SA Maintenance record type.

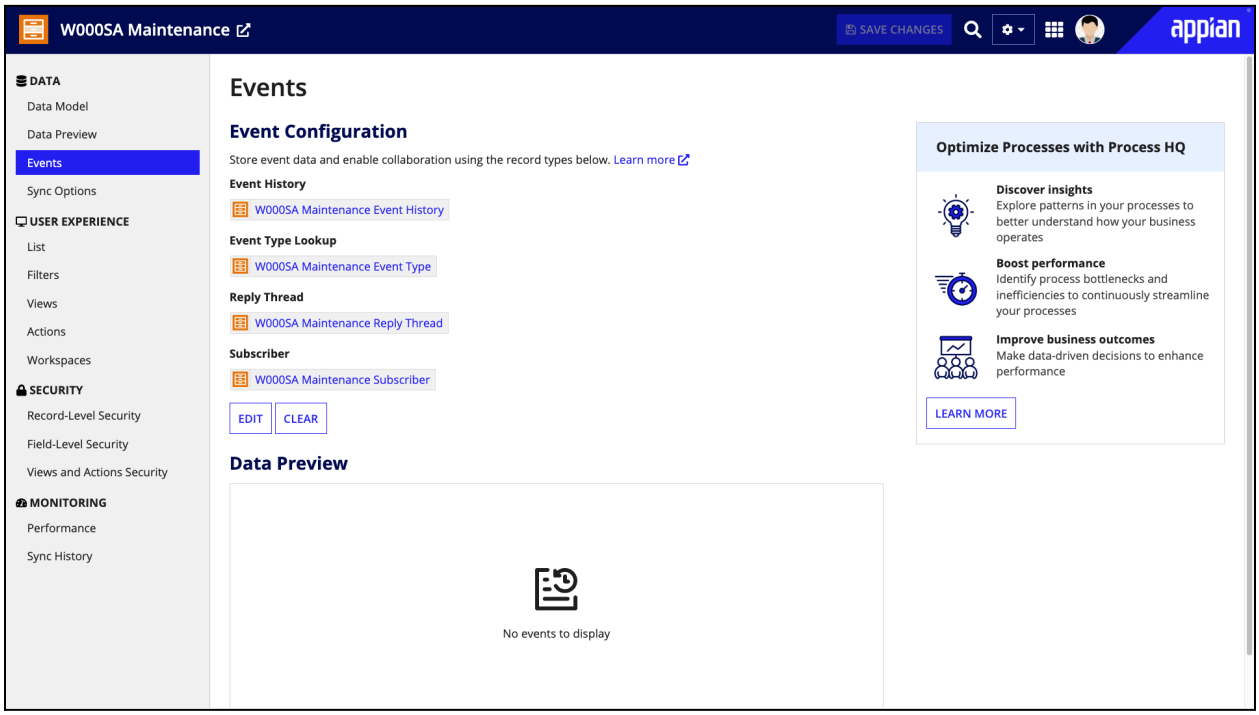


1. In the **Build** view of your application, click **New** > **Record Type**.
2. In the **Create Record Type** dialog, configure the following properties:
    - **Name**: Enter `W#SA Maintenance`.
    - **Plural Name**: Enter `W#SA Maintenance Requests`.
    - **Description**: Enter `The list of maintenance requests for vehicles managed by the W#SA application.`
3. Click **CREATE**.
4. In the **Review Record Type Security** dialog, click **SAVE**.
5. In the record type, click **TELL US ABOUT YOUR DATA**.
6. In the **Configure Data Source** dialog, ensure that **Database** is selected, and click **NEXT**.
7. Ensure **Build Apps Faster with Data Fabric** is selected. Click **NEXT**.
8. Under **Choose Database Table**, find and select **AS_MAINTENANCE**. Click **NEXT**.

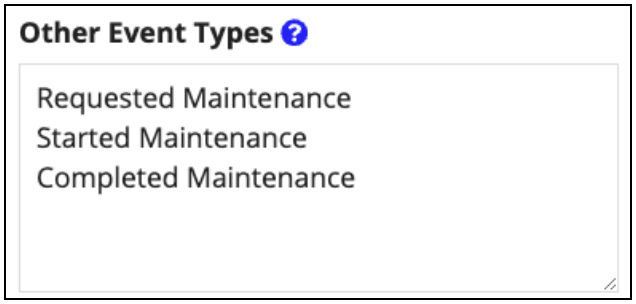
9. Click **NEXT**. You will not create sync filters for this exercise.
10. Preview the fields and configure the following:
    - Using the dropdown, change the Record Field Type for **CREATED_BY** and **MODIFIED_BY** from **Text** to **User**.
11. Preview the fields, and click **FINISH**.
12. Click **SAVE**.

Configure Maintenance Record Events

Finally, you will generate the requested, started, and completed events from the W#SA Maintenance record type.



1. In the left menu, go to the **Events** page.
2. Click **GENERATE EVENT RECORD TYPES**.
3. In the **Generate Event Record Types** dialog, deselect the **Created Maintenance, Updated Maintenance**, and **Commented on Maintenance** checkbox.
4. In the **Other Event Types** input, add the following list:
    - Requested Maintenance
    - Started Maintenance
    - Completed Maintenance



5. Keep the default record type names for **Event History**, **Event Type Lookup**, **Reply Thread,** and **Subscriber**.
6. Uncheck **Download database script.**
7. Click **GENERATE**.

Generate the Maintenance Summary View Interface

In this section, you will create the interface to display the vehicle maintenance summary information.

1. In the left menu, go to the **Views** page.
2. Click **GENERATE RECORD VIEW**.
3. Ensure **Maintenance** is selected, then click **NEXT**. Leave **Subscribers** unselected.
4. Keep the **View Name** as **Summary**, then click **NEXT**.
5. Next to **W#SA Rules & Constants** folder, click the **Edit** icon. Replace this folder with the existing **W#SA Interfaces**.
6. Click **GENERATE VIEW**, then **CLOSE**. Click **SAVE**.

In this exercise, you do not need to configure this record type further. For more practice, try customizing the record list, adding user filters, updating the summary view, and adding a record title.

Add Record Type Relationships

In this exercise, you will add relationships to the record types you just created. Once you build record type relationships, you will be able to easily access data from related records.

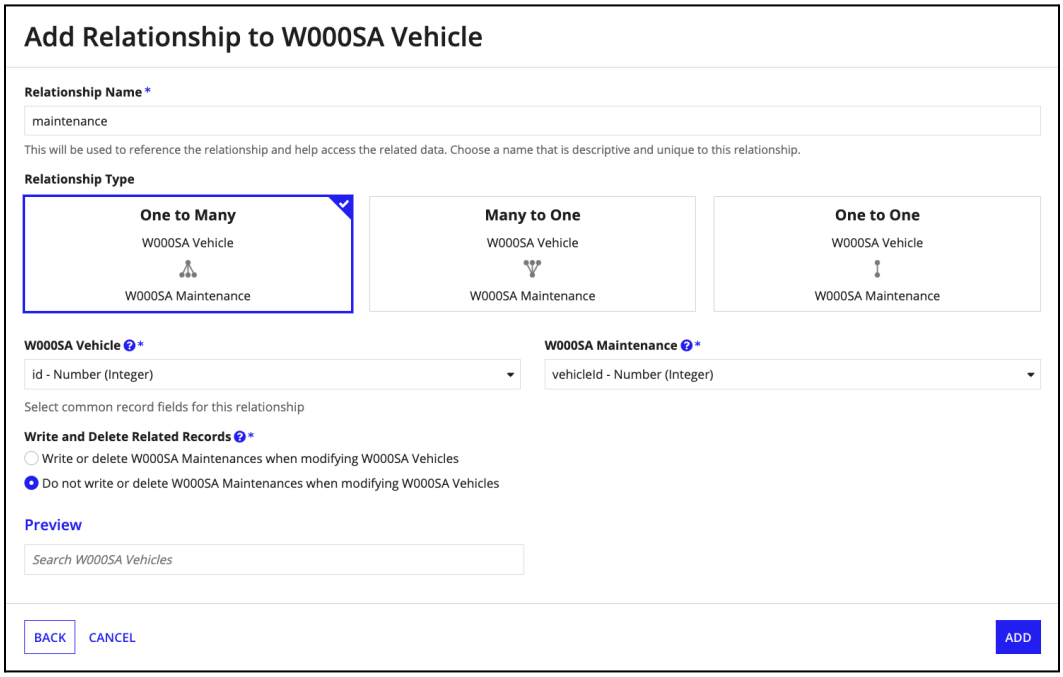
First, you will add four relationships to the W#SA Vehicle record type: Maintenance, Category, Status, and Condition. You will use the existing Category, Status, and Condition record types from the Acme Automobile Reference Application (W#AA).

The Category, Status, and Condition record types are reference, or lookup, tables that contain the unique vehicle category, status, and condition values. A *reference table* contains set,

categorical data, which are often used as static dropdown values. Think of it as a “cumulative list” that uses an identifier, or ID number, (1) to look up a value, such as vehicle category (sedan).

Follow the steps below to add the W#SA Maintenance record type relationship.

1. In the **W#SA Vehicle** record type, go to the **Data Model** page. Click **ADD RELATIONSHIP**.
2. In **Related Record Type**, enter and select **W#SA Maintenance**. Click **NEXT**.
3. Configure the following properties:
    - **Relationship Type**: Select **One to Many**. One vehicle record can be related to multiple maintenance records.
    - **W#SA Vehicle**: Select **id**.
    - **W#SA Maintenance**: Select **vehicleId**.



4. Click **ADD**, then **SAVE**.

Follow the steps below to add three more relationships to the W#SA Vehicle record type.

1. Under **RELATIONSHIPS**, click **ADD RELATIONSHIP**.
2. In **Related Record Type**, enter and select **W#AA Vehicle Category**. Click **NEXT**.

3. Configure the following properties:

    - **Relationship Type**: Select **Many to One**. Many vehicles can have the same category, such as sedan.
    - **Common Fields**: For **W#SA Vehicle**, select **categoryId**. For **W#AA Vehicle Category**, select **id**.

4. Click **ADD**, then **SAVE**.

5. Follow steps 1–4 to add two more relationships to the W#SA Vehicle record type: **W#AA Vehicle Status** and **W#AA Vehicle Condition**. Use the following configurations:

    - **Relationship Type**: Select **Many to One** for both relationships.
    - **Common Fields**: For the **W#AA Vehicle Status** relationship, select **statusId** as the **W#SA Vehicle** field. For the **W#AA Vehicle Condition** relationship, select **conditionId** as the **W#SA Vehicle** field.

NOTE: User filters are automatically created when you add Many-to-One relationships. Check them out by going to the **Filters** page or previewing the record list.

Next, add a relationship from W#SA Maintenance to W#SA Vehicle.

1. Open the **W#SA Maintenance** record type.

2. In the **RELATIONSHIPS** section, under **Suggested Relationships**, click **Add All** to use the suggested relationship to W#SA Vehicle.

Use Record Type Relationships in Custom Record Fields

Record type relationships are useful when you want to aggregate and display data from a related record type. In this exercise, you will create two custom record fields that aggregate maintenance-related data: the total cost of all maintenance per vehicle and the total count of all maintenance requests per vehicle.

Follow the steps below to create two custom record fields in the Vehicle record type.

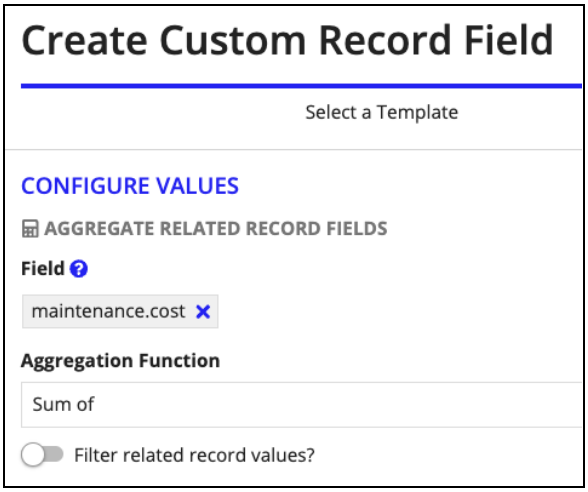
1. Go back to the **W#SA Vehicle** record type.

2. In the **Data Model** page, click **NEW CUSTOM RECORD FIELD**.

3. Select **Aggregate Related Record Fields**. Click **NEXT**.

4. In **Field**, select **maintenance.cost**.

5. In **Aggregation Function**, select **Sum of**.



6. Click **TEST** to preview the new field, and click **NEXT**.
7. Leave the name **costSum**, and click **CREATE**.
8. Follow steps 1–6 to create a custom record field that displays a count of all maintenance requests for a vehicle. Use the following configurations:
    - **Field**: Select **maintenance.id**.
    - **Aggregation Function**: Select **Count of**.
    - **Record Field Name**: Enter `countOfMaintenanceRequests`.
9. Click **SAVE**.

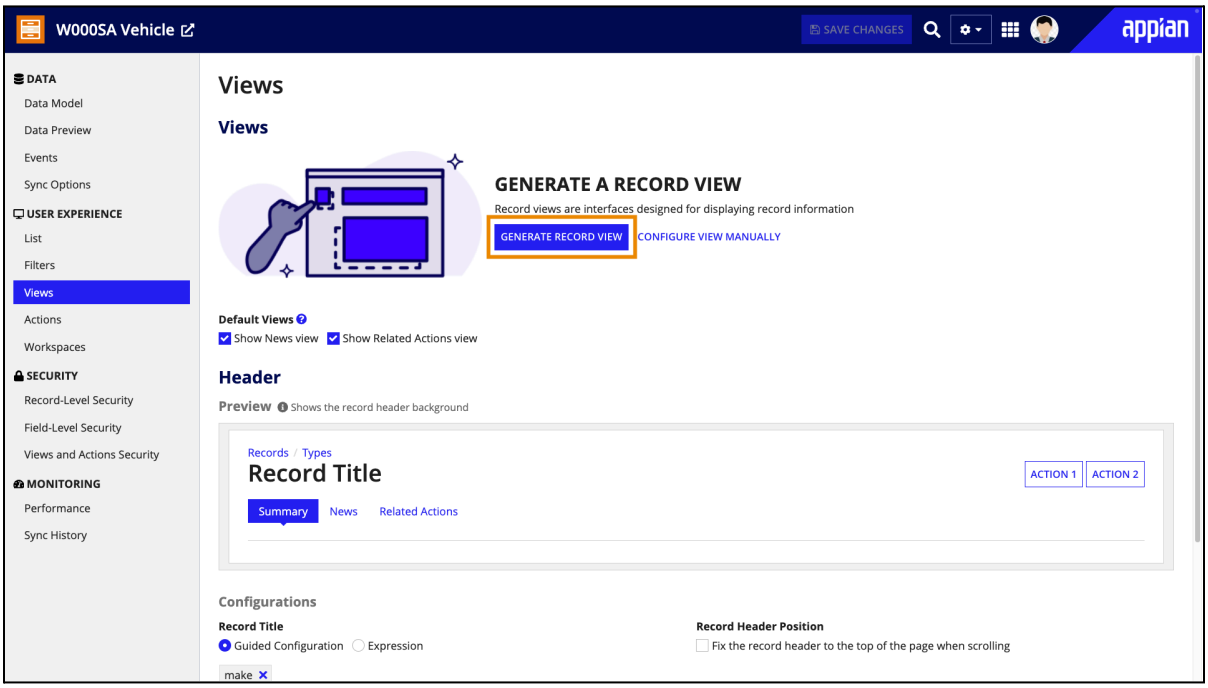
Generate the Summary View Interface

In this section, you will create the interface to display the vehicle summary information.

Follow the steps below to create the summary interface.

1. In the left menu, go to the **Views** page.

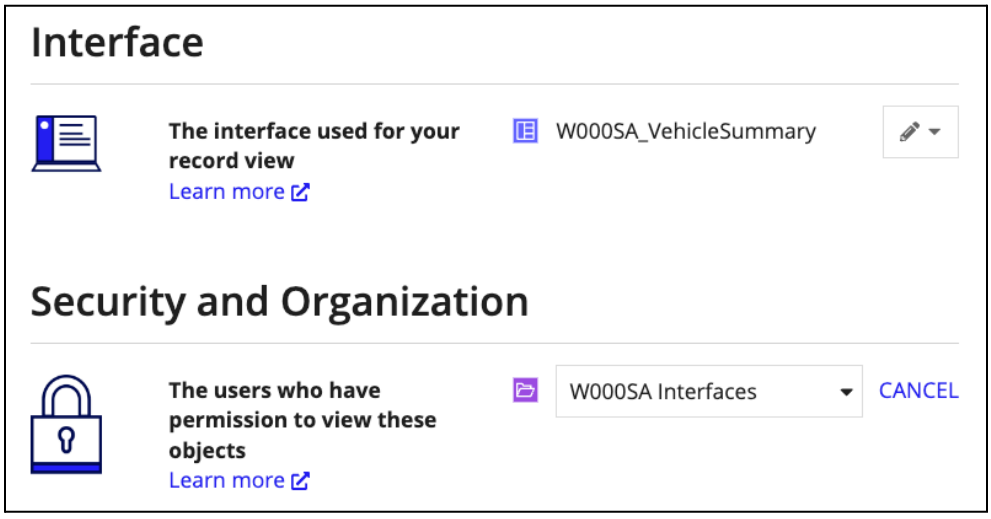
2. Click **GENERATE RECORD VIEW**.



3. Select **Vehicle** and **Maintenance**. Leave all other record types unselected. Click **NEXT**.

4. Keep the **View Name** as **Summary**, then click **NEXT**.

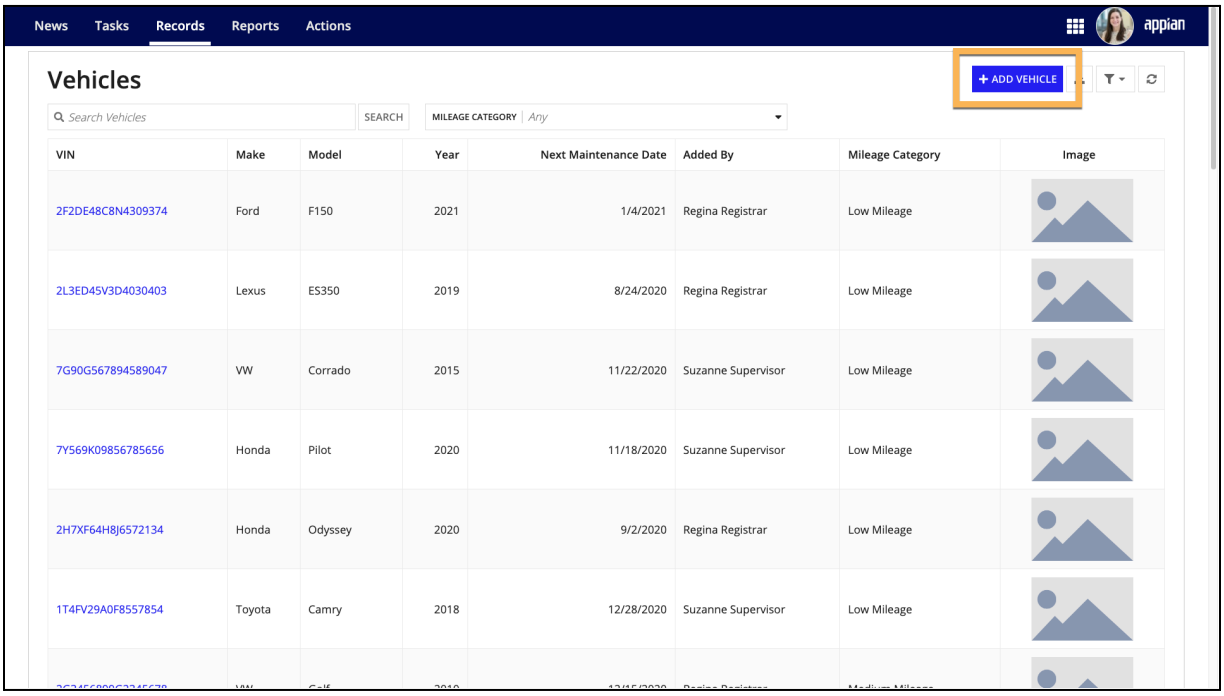
5. Under **Security and Organization**, click the **Edit** icon next to the **W#SA Rules & Constants** folder. Replace this folder with the existing folder **W#SA Interfaces**.



6. Click **GENERATE VIEW**, then **CLOSE**. Click **SAVE**.

Create a Record Action

In this exercise, you will create a record action that will allow registrars and supervisors to add a new vehicle to the fleet. They will be able to click a button on the record list to fill out a form.



Follow the steps below to create a record action.

1. In the left menu, go to the **Actions** page.
2. Click **GENERATE RECORD ACTIONS**.
3. In the **Choose Your Action** dialog, deselect **Update** and **Delete**. For this exercise, you will focus on the **Create** record action. Click **CUSTOMIZE**.
4. Configure the following properties:
    - **Display Name**: Enter `Add Vehicle`.
    - **Description**: Enter `Action to add a new vehicle to the fleet.`
    - Click **NEXT: REVIEW OBJECTS**.
5. Review the objects that Appian generates for this action, including a process model and interface for adding a new vehicle.
6. Click **GENERATE**.
7. Next to the record type name, click **View Record List** to preview the new Add Vehicle record action.

8. In the record type, click **SAVE**.

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 to the exercises you are following in the Step-by-Steps. You can use the AS application as a reference tool. Review it to see how specific objects are configured, or test the application to see how the features work from a business user's perspective. This application is preloaded into your workspace. If you do not see it in the list of applications in your workspace, you can deploy it from the App Catalog. Refer to **Build an Application: Step-by-Step #1** for more information on how to use the App Catalog.

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.