

Appian Step-by-Step #7

Exercise to Accompany
Query Entities

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.

- 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** **Query Entities**
- 8** Process Modeling 101: Part 1
- 9** Process Modeling 101: Part 2
- 10** Reports
- 11** Sites
- 12** Task Report

Exercise 7: Query Entities

3

Create a Query

3

Notice of Rights

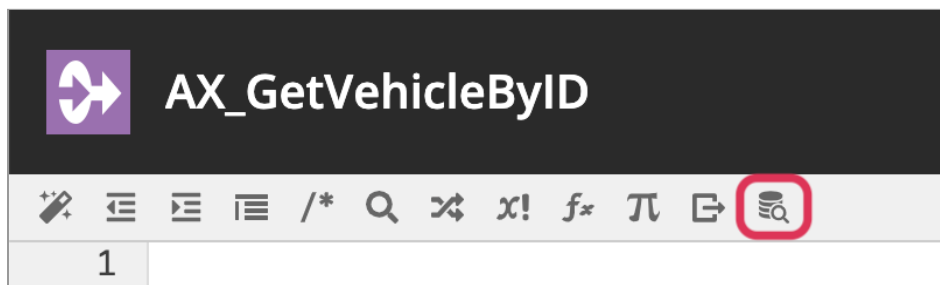
This document was created by Appian Corporation, 7950 Jones Branch Dr, Tysons, Virginia 22102. Copyright 2021 by Appian Corporation. All rights reserved. Information in this document is subject to change. No part of this document may be reproduced or transmitted in any form by any means, without prior written permission of Appian Corporation. For more information on obtaining permission for reprints or excerpts, contact Appian Training at academyonline@appian.com.

Exercise 7: Query Entities

Create a Query

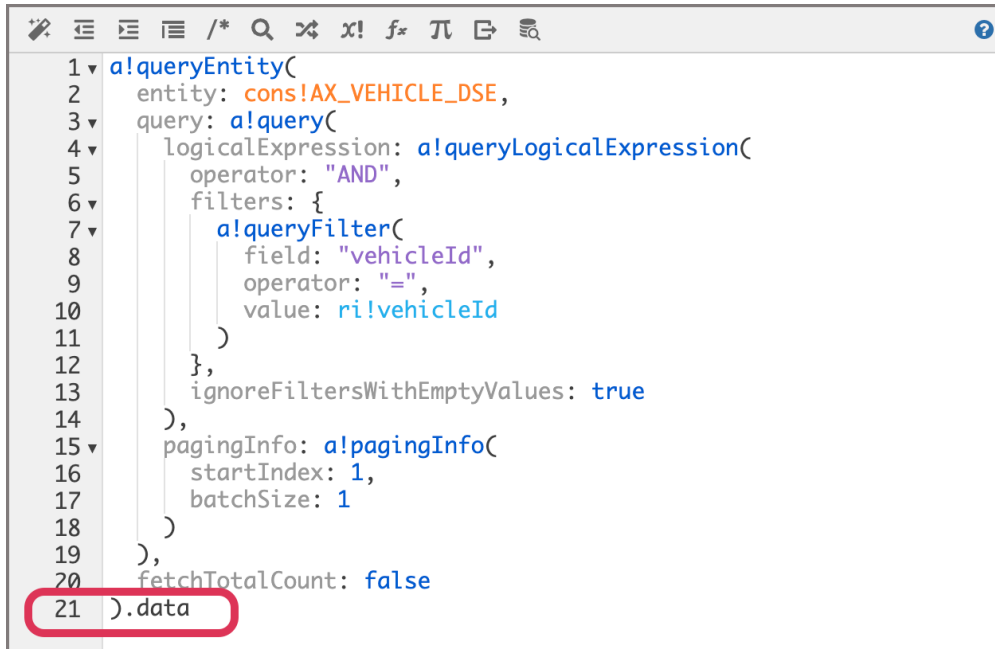
In this exercise, you will create an expression that will return information about a specific vehicle in the fleet given its ID. You will use the Query Editor to create this query. This query will be used in a later exercise to help you configure the Request Maintenance action.

1. In your application, click **New**, and select **Expression Rule** from the dropdown menu.
2. Name the rule **AX_GetVehicleByID**, and give it a description: "Return a vehicle by its unique ID number." Save the rule in **AX Expressions**, and click **Create**.
3. Click the **Query Editor** icon to launch the Query Editor:



4. Enter **AX_VEHICLE_DSE** in the **Data Store Entity** field. This constant was auto-generated for you when you created a record action. Click **Continue**.
5. Click the **Rule Inputs** button in the top right corner.
6. Click **Add New Rule Input**. Name this input **vehicleId**, set the **Type** to **Number (Integer)**, and click **Save**.
7. Under **Paging & Sorting**, replace 50 with **1** row per page.
8. Under **Filters**, click **Add Filter**, then set the following configurations:
 - Under **Field**, select **vehicleId**.
 - Under **Value**, use the **small arrow** next to 123 to select **Input/Variable**.
 - Under **Value**, use the dropdown menu to select **ri!vehicleId**.
9. Click **GENERATE QUERY**, and then **Save Changes**.
10. To test this query, type any vehicle ID into the Value field. For example, type **1**, and click **Test Rule**. You will see the entire data subset in the **Test Output**.

11. Add **.data** to the end of your expression to limit the query results only to the vehicle data:



```
1 a!queryEntity(  
2   entity: cons!AX_VEHICLE_DSE,  
3   query: a!query(  
4     logicalExpression: a!queryLogicalExpression(  
5       operator: "AND",  
6       filters: {  
7         a!queryFilter(  
8           field: "vehicleId",  
9           operator: "=",  
10          value: ri!vehicleId  
11        )  
12      },  
13      ignoreFiltersWithEmptyValues: true  
14    ),  
15    pagingInfo: a!pagingInfo(  
16      startIndex: 1,  
17      batchSize: 1  
18    )  
19  ),  
20  fetchTotalCount: false  
21 ).data
```

12. Click **Test Output** again to view the impact of the .data notation on the returned result.
Click **Save Changes**.