

# Rating Tests Guide

---

The purpose of this user guide is to help testers use execute and analyze rating tests in Apollo.

## Table of Contents

---

Rating Tests Guide .....	1
Table of Contents .....	1
Description .....	2
How it works .....	2
Rating Engine (Visual) .....	3
Execution.....	4
Test result analysis .....	5

# Description

---

## How it works

### 1. Scenario Setup:

To run any tests, we first need a fully elaborated quote (risk, coverage, question answers & premium), that is the purpose of the step “**Given** quote for ‘state’ and ‘algorithm’ is set to Quoted”.

Using two parameters: State & Algorithm (e.g. VA00029).

```
@ratingTests
Scenario: Test Rating Algorithm
    Given quote for '<State>' and '<Algorithm>' is set to Quoted
    When expected values are gathered
    Then expected values should match the system output

Examples:
| State | Algorithm |
| IL    | VA00029  |
| SC    | VA00058  |
| CA    | VA00058  |
| GA    | VA00058  |
```

Details: this step will use Rest API to go through the quote process in order to get a premium meeting the both of the parameters. This uses default question answers from “ApolloQA.Data.TestData.AnswersHydrator”.

Alternatively, we can use the desired quote to run tests on by using the quote Id. (ApplicationId)

```
@ratingTests
Scenario: Test Rating Algorithm Static Quote
    Given quote with Id 13901 is loaded
    When expected values are gathered
    Then expected values should match the system output
```

### 2. Gathering expected premium along with all its factors

There is a Rating Engine in place (ApolloQA.Data.Rating.Engine), this class uses Factors(e.g. Class Code Factor) which uses Knownfields (E.g. Class Code) both resolving to a single value to then be used on the algorithm formula(E.g. VA00029) in the rating engine.

The manual used to output a premium comes from the PDF manual which is parsed into tables. Each state’s manual can be found in “ApolloQA.Data.RatingManual”.

Details: The system uses Rest API, Cosmos & SQL to gather all the data points directly from their source. The source of each known field is configured in the source property of each object in “ApolloQA.Data.Rating.Knownfields.json”.

### 3. Testing that expected results are identical to actual results from the system.

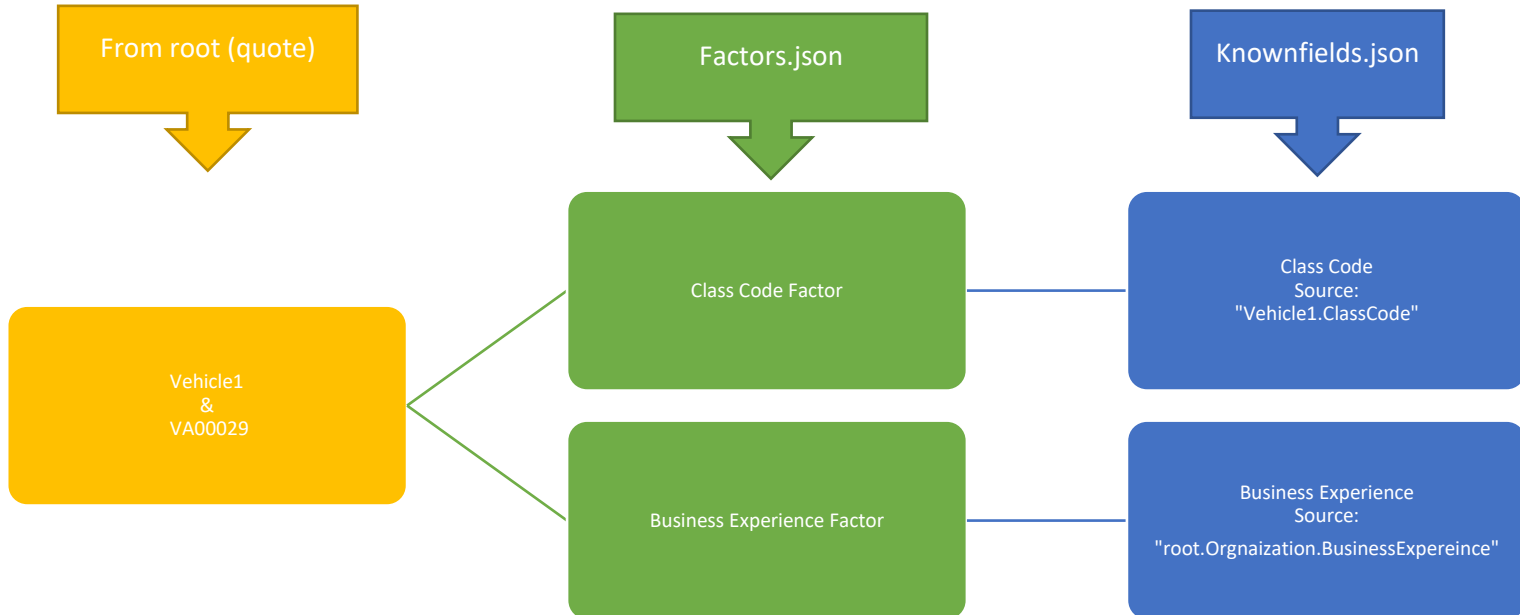
Expected Results: Json object generated by “ApolloQA.Data.Rating.Engine.Run()”.

Actual Results: Json object generated by the system upon rating. (Rating Worksheet object)

The step iterates through every Vehicle & Coverage from the expected results object, Failing if the premium is different.

Also, it will iterate through every single Factor and compare both resulting factor. (e.g. 1.05)

## Rating Engine (Visual)



### Vehicle & Coverage:

- engine to iterate through every vehicle and all coverages associated. (policy & vehicle level)

### Factors:

- factors are read from algorithm formula (e.g. VA00029). The system will attempt to find each factor listed in `ApolloQA.Data.Rating.Factors.json`.
- Each factor is instantiated as a `ApolloQA.Data.Rating.Factor` object.
- Each factor is resolved one of two ways:
  1. **(Default)** resolving each Knownfield associated (must exist in `knownfields.json`), then it will find the factor in its corresponding table (e.g. `VA00029.ClassCodeFactor`) from the Rating Manual.
  2. CustomCalculation (triggered if property "CustomCalculation"=true in `Factors.json`) The system will attempt to find a the source property value in `Apollo.Data.Rating.Factor` as a private member.

### Knownfields:

- Each Factor contains a set of knownfield, and each knownfield is resolved by resolving the value of its source property. (e.g. `root.Organization.BusinessExperience`)
- Knownfield are matched by the column name on the Factor table from the rating manual.

## Execution

1. Because rating tests are too extensive to run on a regular basis, therefore they're tagged with "@ratingTests" which is being ignored in default.srprofile.

Ignored (with '!'):

```
<Filter>!@ignore & @; !@bugReported & @; !@broken & @; !@ratingTests /Filter>
```

Not Ignored (without '!'):

```
<Filter>!@ignore & @; !@bugReported & @; !@broken & @; @ratingTests /Filter>
```

note: make sure to rebuild solution for the above changes to take effect

**Important:** never push to master without ignoring the rating tests because execution is really extensive when triggered in the pipeline.

2. Select the desired test to run

The screenshot displays the Visual Studio Test Explorer interface on the left and the code editor on the right. The Test Explorer shows a tree view of tests under the 'Algorithms' feature. The test 'Test Rating Algorithm, Variant 0 in Algorithms' is selected and highlighted with a red box. The 'Test Detail Summary' pane at the bottom shows the source of the test as 'Algorithms.feature' line 19. The code editor on the right shows the 'KnownFields.json' file, which contains a list of test data examples. The example for 'IL' is highlighted with a red box, showing the 'State' and 'Algorithm' fields.

State	Algorithm
IL	VA00029
IL	VA00036
IL	VA00043
IL	VA00054
SC	VA00058
SC	VA00063
SC	VA00043
SC	VA00050
CA	VA00058
CA	VA00065
CA	VA00043
CA	VA00050
GA	VA00058
GA	VA00065
GA	VA00043
GA	VA00050

## Test result analysis

Test results are found in ./ApolloQA/TestResults/TestRunReport.html

### Step Results:

1. Given quote for 'State' and 'Algorithm' is set to quoted.

#### Output:

Quote Id

Link to Rating Worksheet

Given quote for 'IL' and 'VA00029' is set to Quoted	[DEBUG] cover name: Bodily Injury Property Damage (BIPD) [DEBUG] Quote Id: 13986 [DEBUG] Rating Group Id (rating worksheet): https://biberk-apollo-qa.azurewebsites.net/rating/ratings-worksheet/4006bf13-56a3-4995-994e-fdf415d016ca  done: AlgorithmsSteps.GivenQuoteForIsSetToQuoted("IL", "VA00029") (31.5s)	Succeeded in 31.538s
-----------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	----------------------

2. When expected values are gathered.

Visual of the entire expected object that was used for the test

Output:

Matched Row in the rating worksheet.

Resulting Value for each Factor.

Resulting Value for each Knownfield.

When expected values are gathered	[DEBUG] Current Coverage: Bodily Injury Property Damage (BIPD) [DEBUG] C:\Users\macosta\Desktop\Biberk Apollo Automation\ApolloQA\ApolloQA\Data\RatingManual\DR.2.csv [DEBUG] Points: 0 [DEBUG] C:\Users\macosta\Desktop\Biberk Apollo Automation\ApolloQA\ApolloQA\Data\RatingManual\DR.2.csv [DEBUG] { "CoverageCode": "VA00063", "Factors": { "VA00063.BaseRateFactors": { "KnownFields": [], "matchedRow": null, "matchedNextRow": null, "interpolated": false, "displayOnly": false, "Value": 806.76, "parsedValue": null, "Name": "BaseRateFactors", "NameUI": { "Base Rate Factor", "VA00063 Base Rate Factor" }, "CustomCalculation": true, "currentPremium": 806.76 }, "VA00063.IsAccidentPreventionFactors": {	Succeeded in 15.331s
-----------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	----------------------

3. Then expected values should match the system output  
result of comparison of both objects

#### Output:

Difference in premiums.

Difference in Factors.

Any factors expected but not found in the rating worksheet.

e.g.

Then expected values should match the system output	[DEBUG] Expected Premium: 725 does not equal actual Premium: 2474 for Vehicle: 2020 Honda Accord & Coverage Code: VA00063, Expected Factor Value for factor Increased Limit Factor value: 0.8500 did not equal 2.9014, Expected Factor Value for Policy Term Factor value: 365 / 365 did not match value in (TermFactorPremium: 2474) [ERROR] Expected Premium: 725 does not equal actual Premium: 2474 for Vehicle: 2020 Honda Accord & Coverage Code: VA00063 error: Expected Premium: 725 does not equal actual Premium: 2474 for Vehicle: 2020 Honda Accord & Coverage Code: VA00063 (2.8s) <b>Expected Premium: 725 does not equal actual Premium: 2474 for Vehicle: 2020 Honda Accord &amp; Coverage Code: VA00063</b>	Failed in 2.776s
-----------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	------------------