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](#_Toc71284587)

[Table of Contents 1](#_Toc71284588)

[Description 2](#_Toc71284589)

[How it works 2](#_Toc71284590)

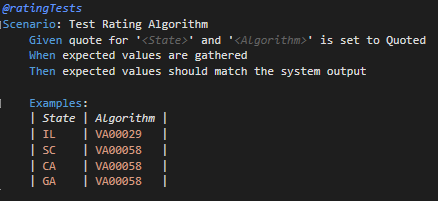
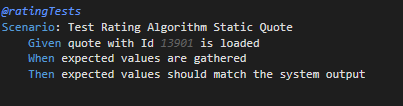
[Rating Engine (Visual) 3](#_Toc71284591)

[Execution 4](#_Toc71284592)

[Test result analysis 5](#_Toc71284593)

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).  
     
     
   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)  
   
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)

Factors.json

From root (quote)

Knownfields.json

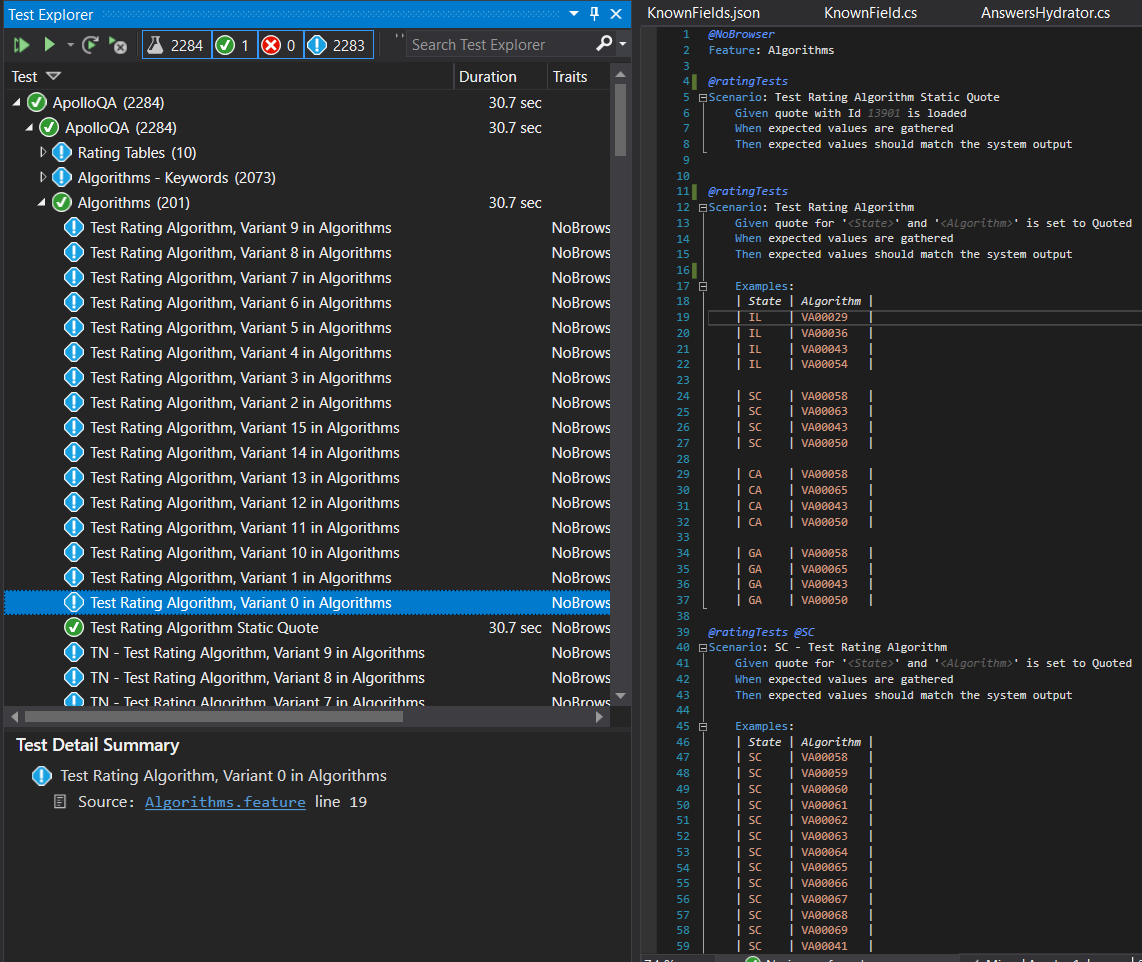
Vehicle & Coverage:

* 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.
* Each knownfield is resolved by resolving the value of its source property. (e.g. root.Organization.BusinessExperience)

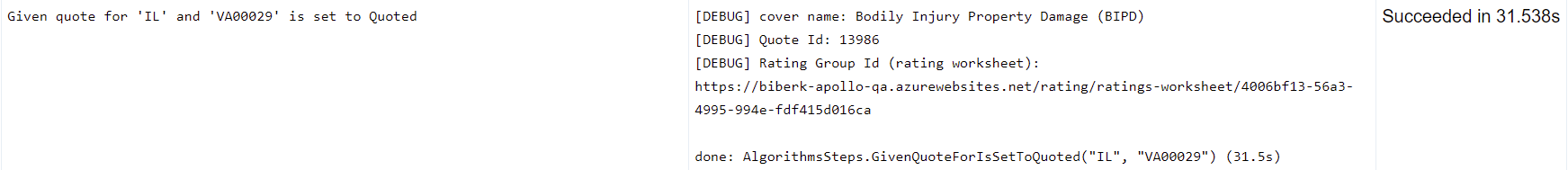
# Execution

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 ‘!’):   
     
     
   Not Ignored (without:):  
     
     
   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   
   

# 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  
   
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.  
   
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.  
   