Arun Pareek, Principal Consultant, Rubicon Red

DETERMINE BOOKING FRAUD - SOLUTION

PROBLEM STATEMENT

The problem statement was shared on the Decision Management Community website in February 2017 and an abstract of it is presented here:

Your decision model should determine potential fraud during an online product booking process. It's based on an automatically calculated Fraud Rating Score that should be less than 200 to allow automatic booking. Fraud Rating Score depends on the following factors:

- If Booked Product Type is POST-PAID-HOTEL add 5 to the score
- If Booked Product Type is INTERNAL-FLIGHT add 100 to the score
- If Booked Product Type is INTERNATIONAL-FLIGHT add 25 to the score
- If Booked Product Type is CAR add 10 to the score
- If Booked Product Type is PRE-PAID-HOTEL add 5 to the score
- If there were no previous orders from this customer add 100 to the score
- If Number of Orders from this customer between 1 and 10 including bounds add (100 – Number of Orders * 10) to the score
- If Customer has previous disputes add 190 to the score

Solution Submitted by: Arun Pareek, Principal Consultant, RubiconRed

http://blog.rubiconred.com

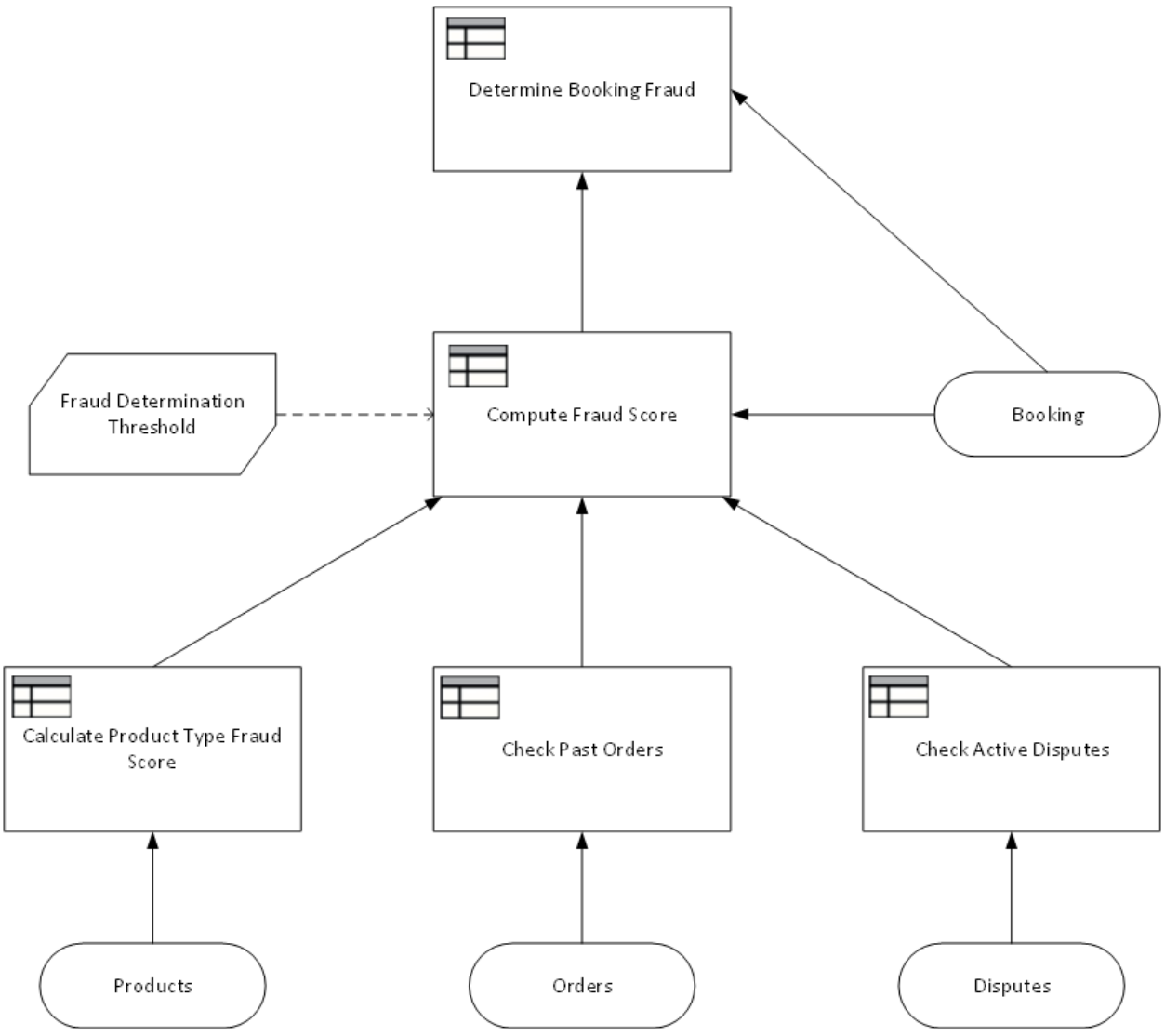
http://beatechnologies.wordpress.com

Solution implemented using Oracle Process Cloud Service DMN Engine

SOLUTION

Decision Requirements Diagram

The complete solution is presented in form of both a decision model diagram and a decision logic representation. To evaluate the multiple scenario as part of this use case, the overall decision model is broken down into decisions and sub decisions as shown below:

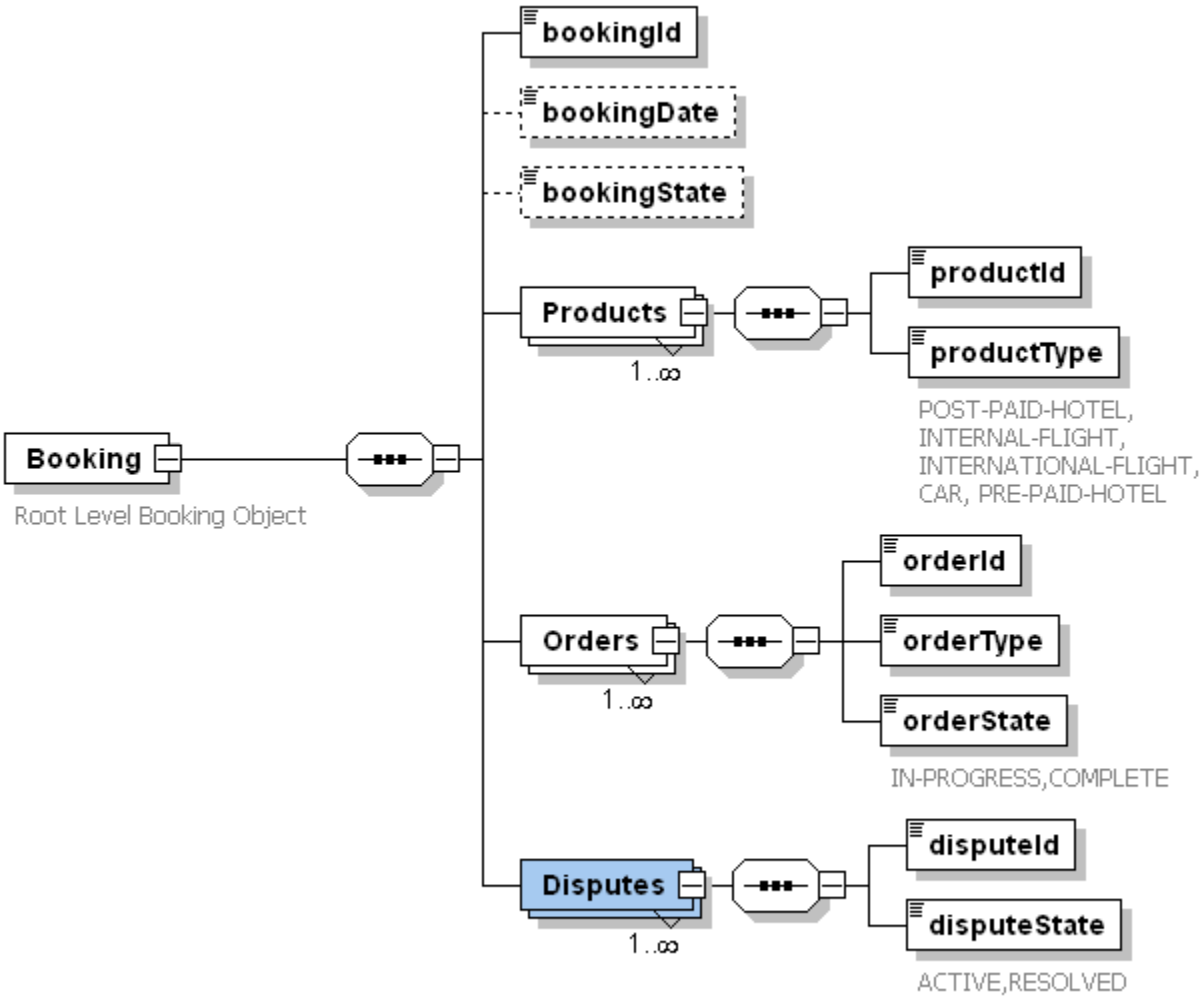


Decision Requirements Diagram

- **Determine Booking Fraud** is the top level decision that asserts a Boolean flag of **true|false** as a final interpretation of the sub decisions. The flag is based on the **Compute Fraud Score** decision value (**true** for fraud score greater than 200 and **false** for fraud score less than 200)
- **Check Active Disputes** is a sub decision that iterates over all Dispute elements of a booking request and if any dispute is valid asserts a constant fraud score (190 in this case)
- **Check Past Orders** is a sub decision that iterates over all Order elements of a booking request and if there are any past orders, asserts a calculated fraud score (100 – Number of Orders * 10)
- **Calculate Product Type Fraud Score** is a sub decision that loops through all Products in a booking request and based on the product type, assigns a fraud score. It also rolls up a sum of all assigned fraud score for each product type.
- The **Compute Fraud Score** sub decision invokes the above sub decisions and sums up the evaluated fraud score from each.

Decision Input Definition

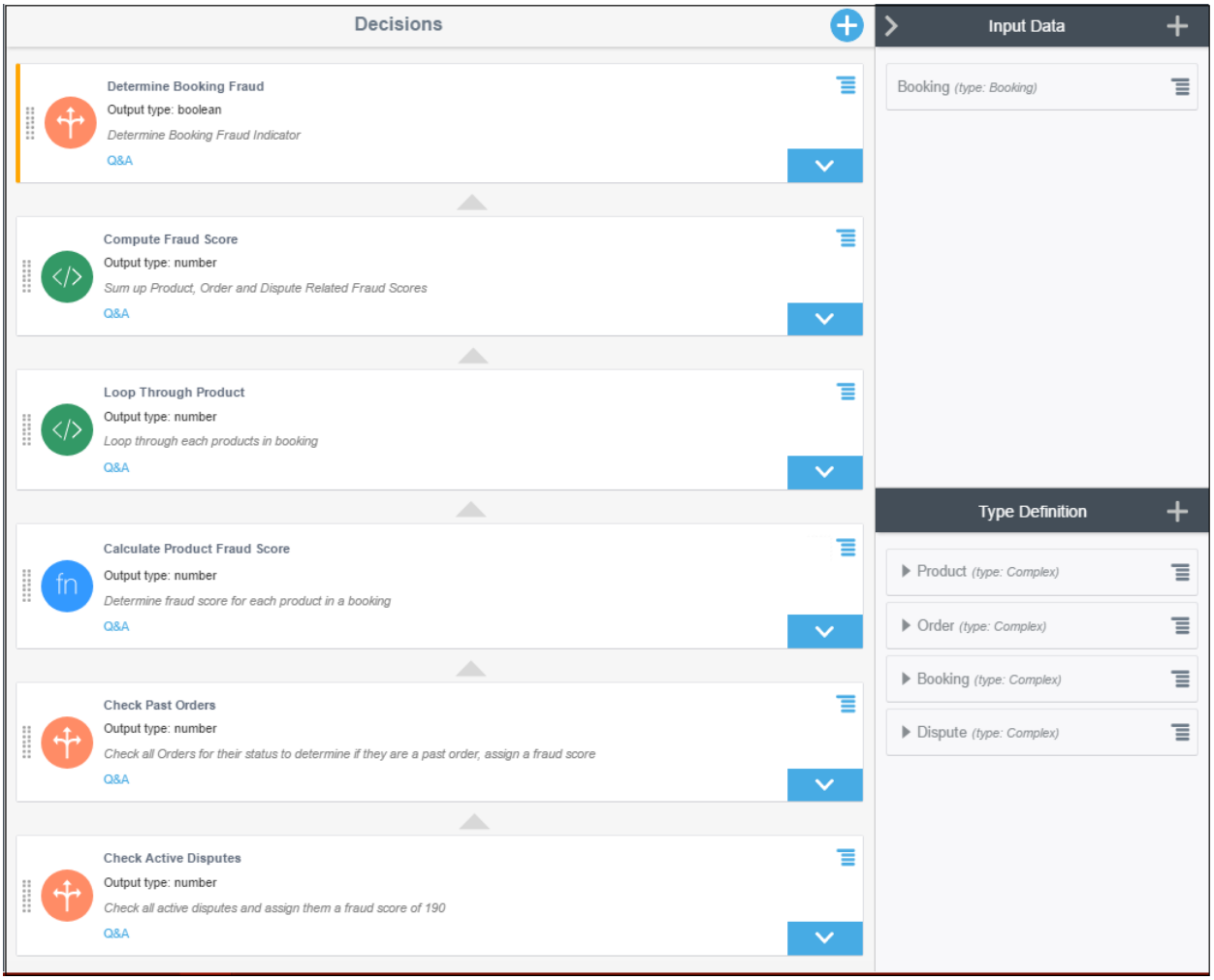
In order to implement the decision logic for this scenario, we would need to create a root level **Booking** request input data type The data structure for the decision model is depicted below:



- A **Booking** can have one or more **Products**. Each product has a unique **productId**. A product element also has a sub-element representing the **productType**.
- A **Booking** can have one or more **Orders**. Each order has a unique **ordertId**. The **bookingState** sub element represents if it is a current order or a completed one**.**
- A **Booking** can have one or more **Disputes**. Each dispute has a unique **disputetId**. The **disputeState** sub element represents if it is a confirmed or an active dispute. Active disputes are currently under investigation.

Decision Logic Level

The overall decision model is broken down into individual decisions that are modular with the top level decision determine if the booking is a fraudulent one or not. The sub decisions uses different types of boxed expression to compute a fraud score for every scenario covered in the use case.

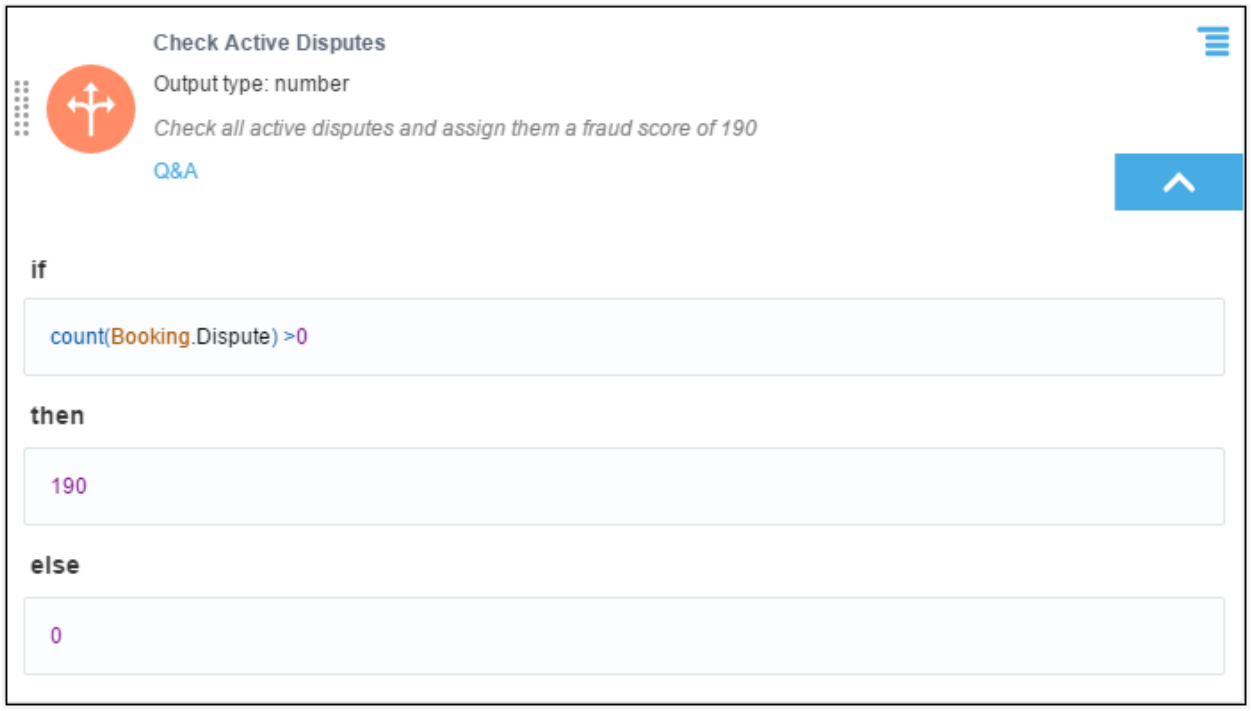


The top level and each of the underlying decisions are explained in more details in the following section.

Decision 1 - Check Active Disputes

Decision Type	Sub Decision
Boxed Expression Type	If-then-Else
Decision Input Type	Booking.Disputes[] (List)
Decision Output Type	Dispute Fraud Score (integer)
Question	Does the booking request has any unresolved disputes?
Allowed Answers	190,0

The expression simply checks if the count of Dispute elements in the Booking request is greater than 0. If a dispute is found, then it assigns a fraud score of 190 else the fraud score is 0.



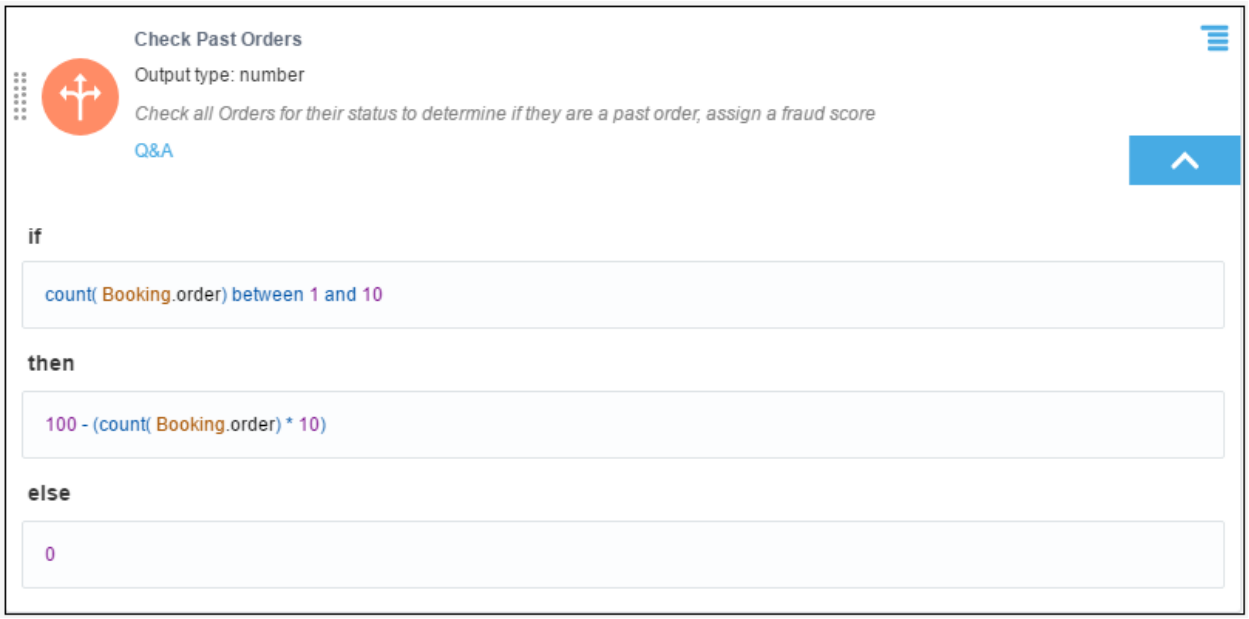
Decision 2 - Check Past Orders

Decision Type	Sub Decision
Boxed Expression Type	If-then-Else
Decision Input Type	Booking.Orders[] (List)
Decision Output Type	Dispute Fraud Score (integer)
Question	Does the booking request has any past orders?
Allowed Answers	0-100

The conditional expression checks if the count of Order elements in the Booking request is greater than 0. If a previous orders are found, then it calculates the number of order to determine the fraud score using the formulae.

100 – Number of Orders * 10

The higher the number of previous orders, the lesser is the fraud score. For example, if a booking has 9 previous orders, then the fraud score would be 10 as opposed to a fraud score of 90 if a booking has 1 past order.

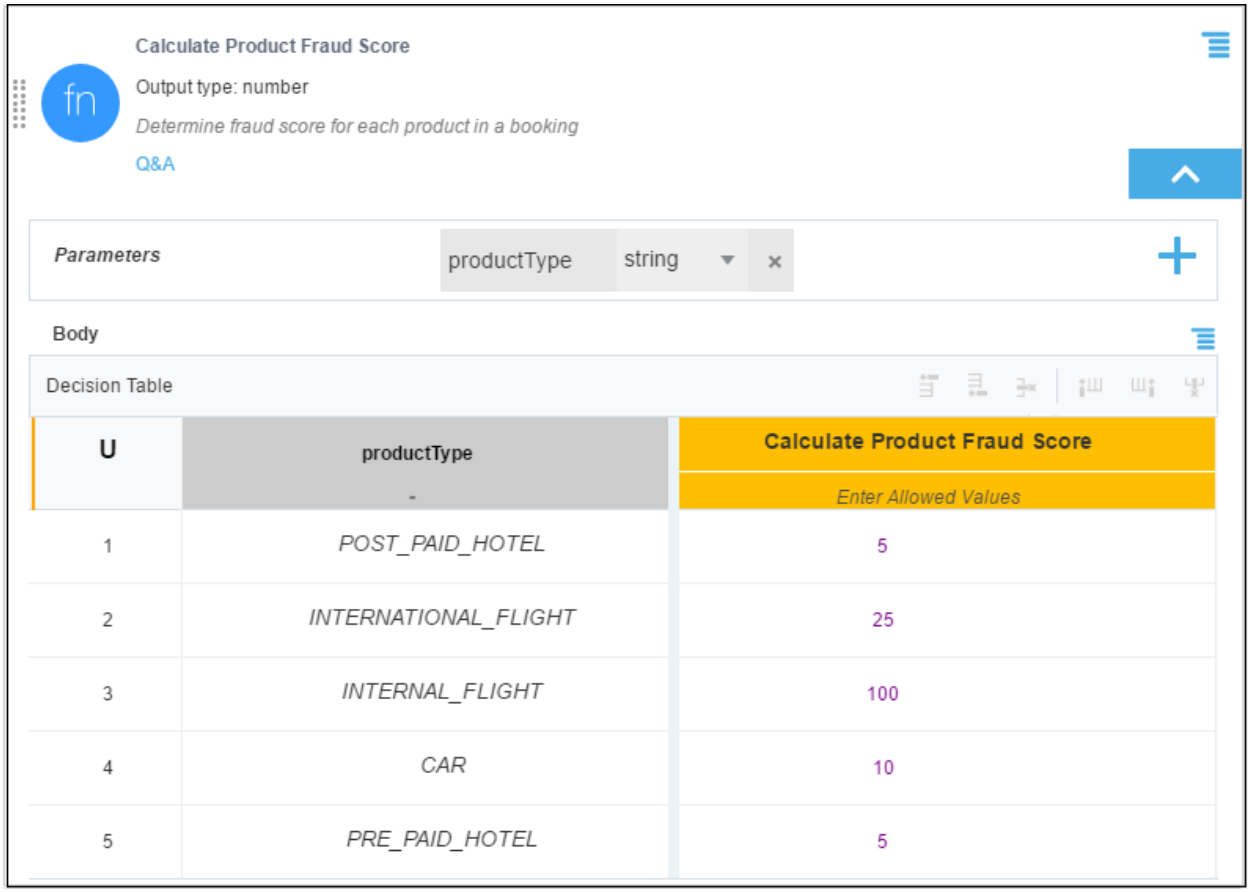


Decision 3 - Calculate Product Fraud Score

Decision Type	Sub Decision
Boxed Expression Type	Function with a Decision Table
Decision Input Type	Product Type (String)
Decision Output Type	Dispute Fraud Score (integer)
Question	What is the fraud score for a particular product type?
Allowed Answers	0,5,25,100,10,5

This is a tricky sub decision to implement. There can be multiple products within a booking request and a fraud score has to assigned based on the product type for each product.

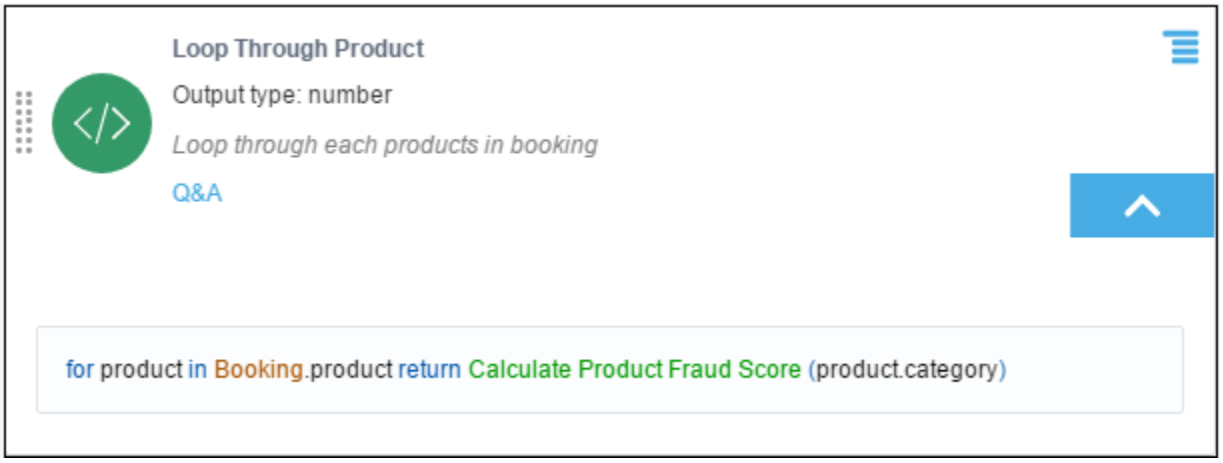
The sub decision is implemented as a **function (**a boxed expression that creates a parameterized logic to apply multiple times with different parameter values**)**. The function accepts **productType** as an input and evaluates the fraud score based on a unique hit policy decision table.



Decision 4 – Loop through Product

Decision Type	Sub Decision
Boxed Expression Type	Friendly Enough Expression Language (FEEL)
Decision Input Type	Booking.Products[] (List)
Decision Output Type	Fraud Score List (integer)
Question	-
Allowed Answers	-

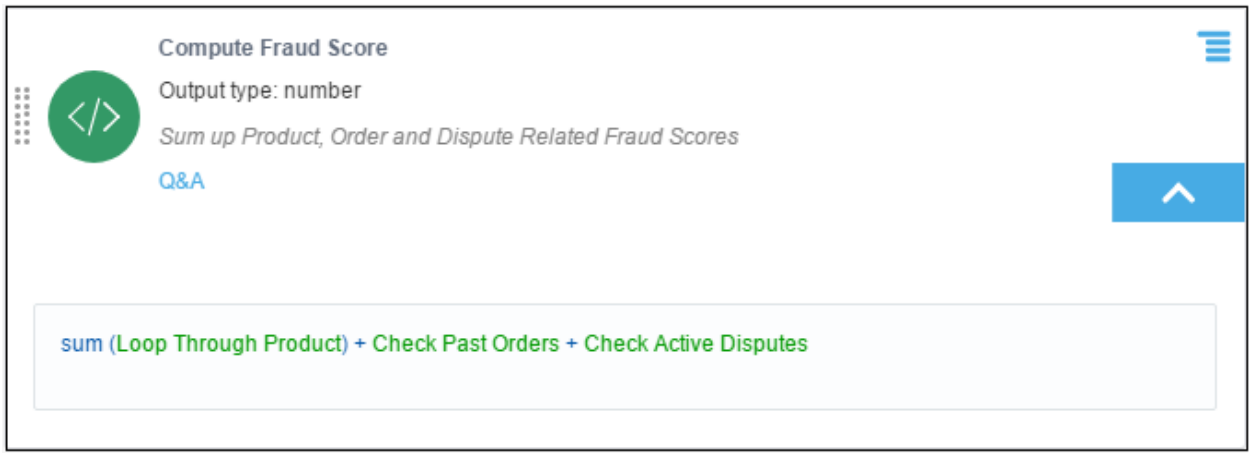
The sub decision is implemented as a **friendly enough expression language (FEEL)** to loop over the Products in a booking and return a list of fraud score for each product type. This sub decision invokes the parameterized **Calculate Product Fraud Score** by passing the product type for each product in the loop.



Decision 5 – Compute Fraud Score

Decision Type	Sub Decision
Boxed Expression Type	Friendly Enough Expression Language (FEEL)
Decision Input Type	Booking
Decision Output Type	Overall Fraud Score (integer)

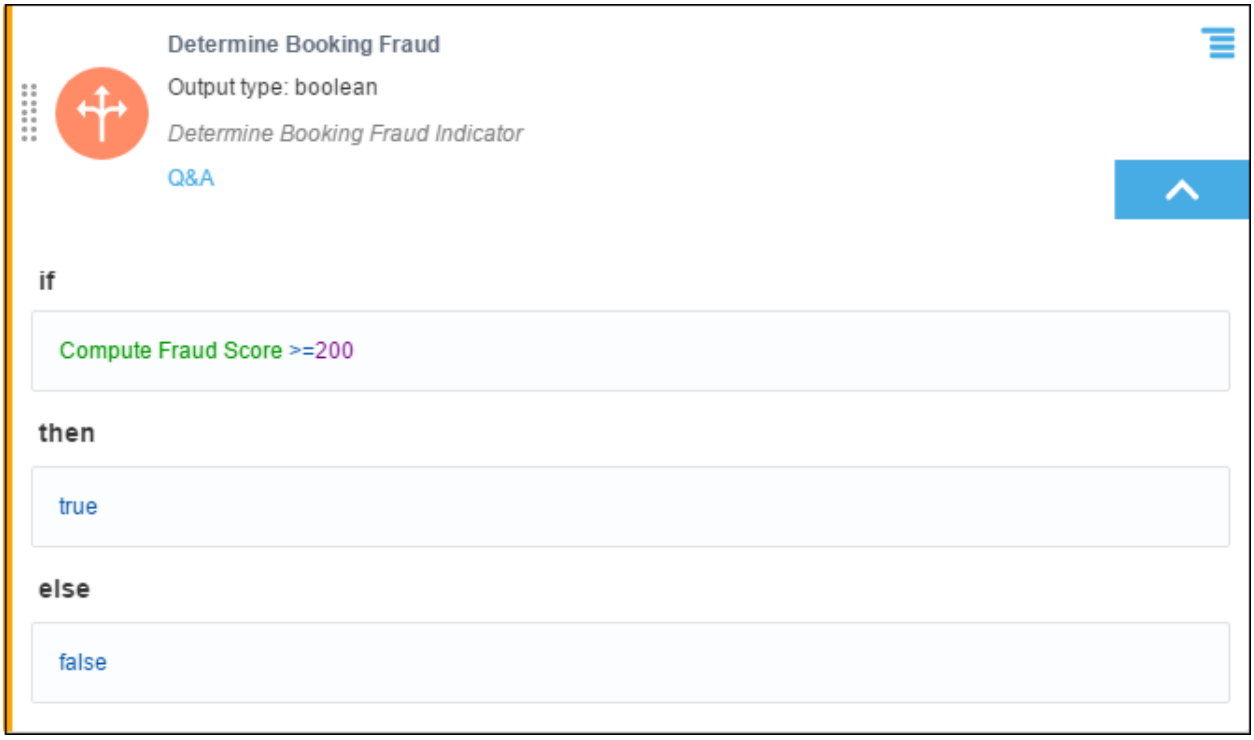
This sub decision is again implemented as a **friendly enough expression language (FEEL)** to sum all the product scores determined by the previous decisions. It uses a summation function to calculate the sum of all product type fraud score retrieved from the **Loop through Product** decision and adds the result of the **Check Past Orders** and **Check Active Disputes** decisions.



Decision 6 – Determine Booking Fraud

Decision Type	Sub Decision
Boxed Expression Type	If-then-Else
Decision Input Type	Computed Fraud Score (integer)
Decision Output Type	Booking Fraud Interpretation (boolean)
Question	Does the computed fraud score confirm it is a fraudulent booking?
Answers	True, False

The conditional expression checks if the value of the computed fraud score is greater than or less than 200. If it is greater than or equal to 200, then it asserts true otherwise false.



This completes the implementation of the decision logic level for the use case.

Arun Pareek, Principal Consultant, Rubicon Red

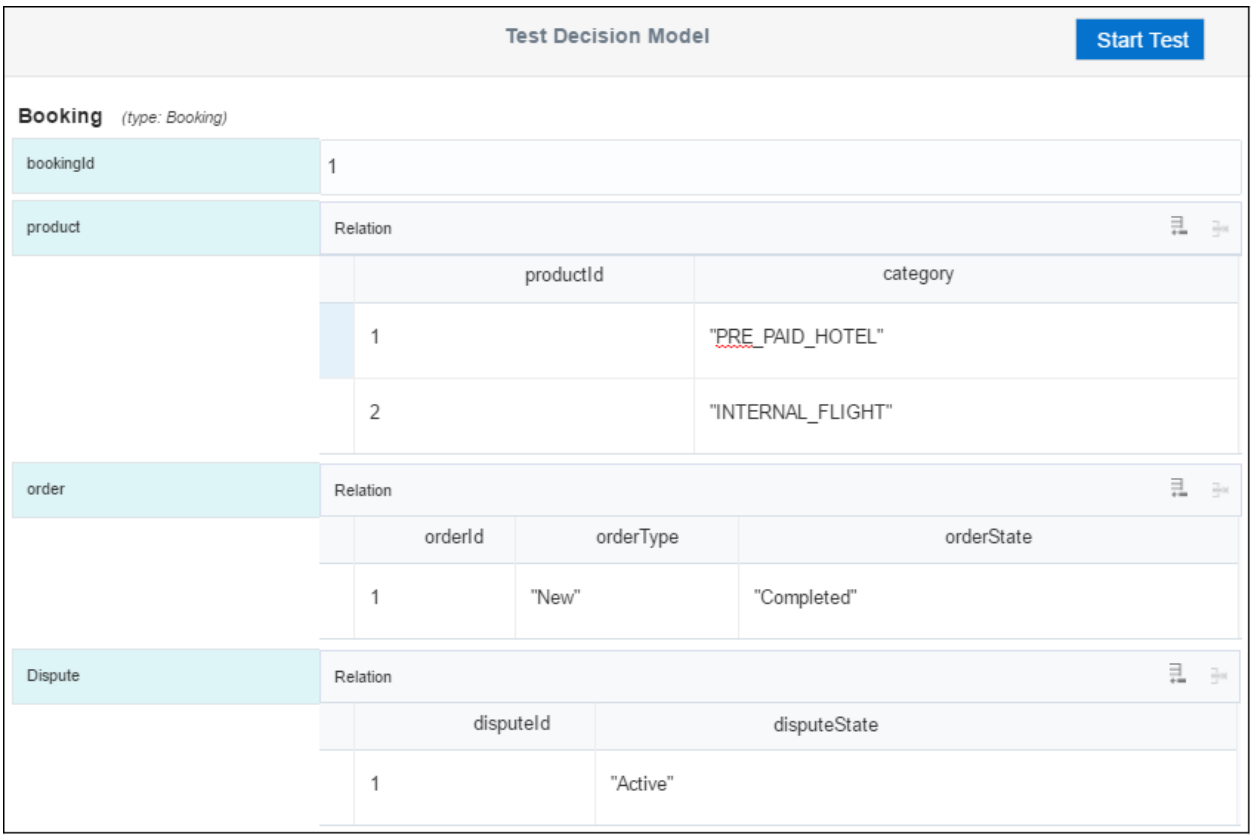
Testing the Final Decision Model

The following section shows how the overall solution is unit tested. It also shows how the decision model is exposed as a decision service rest operation which can accept a JSON type request and assert an outcome.

Unit Testing – Testing for Fraud

Request

Testing the decision logic with a booking comprising of two products (of type **PRE_PAID_HOTEL and INTERNAL_FLIGHT**), one order and one dispute.

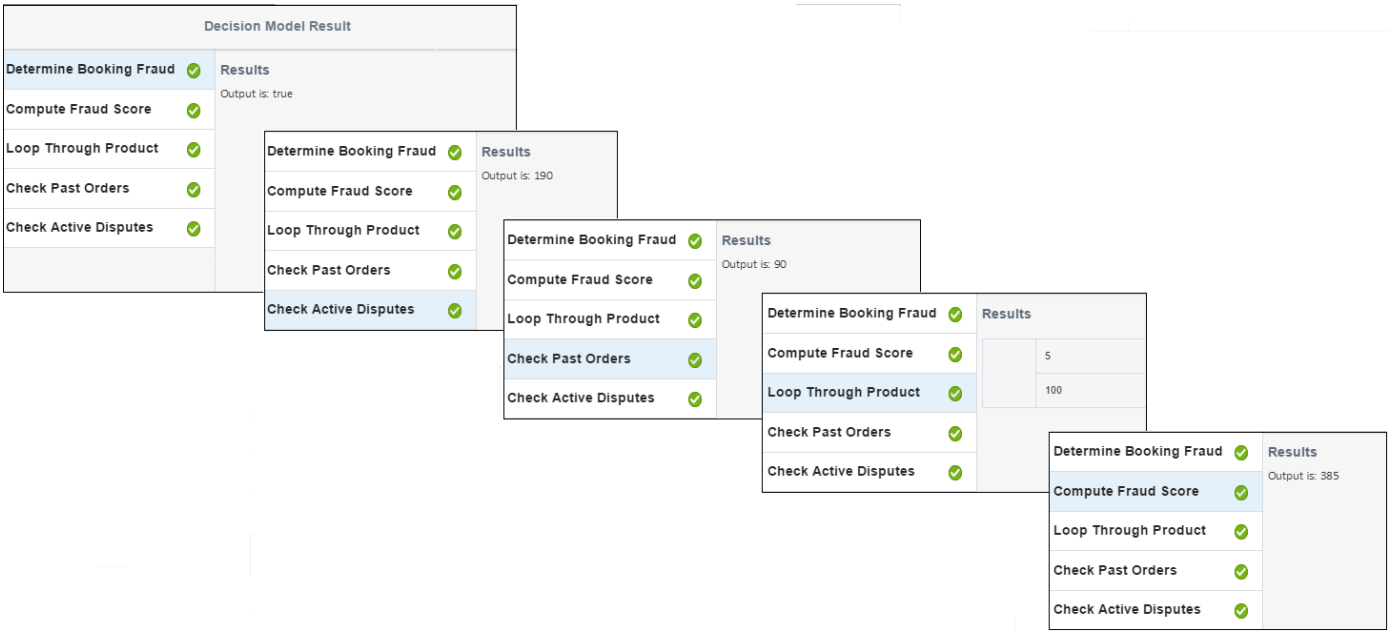


Response through Sub Decision Chains

The result for each of the decision functions for the above request through the decision functions is:

Check Active Disputes	190
Check Past Orders	90
Calculate Product Fraud Score	5,100
Loop Through Products	
Compute Fraud Score	

Determine Booking Fraud	**true**



Exposing the Decision Model as a Rest Service and Testing It

The decision model can also be deployed as a service which can be invoked as a REST POST operation (**determineBookingFraud**) from any external application.

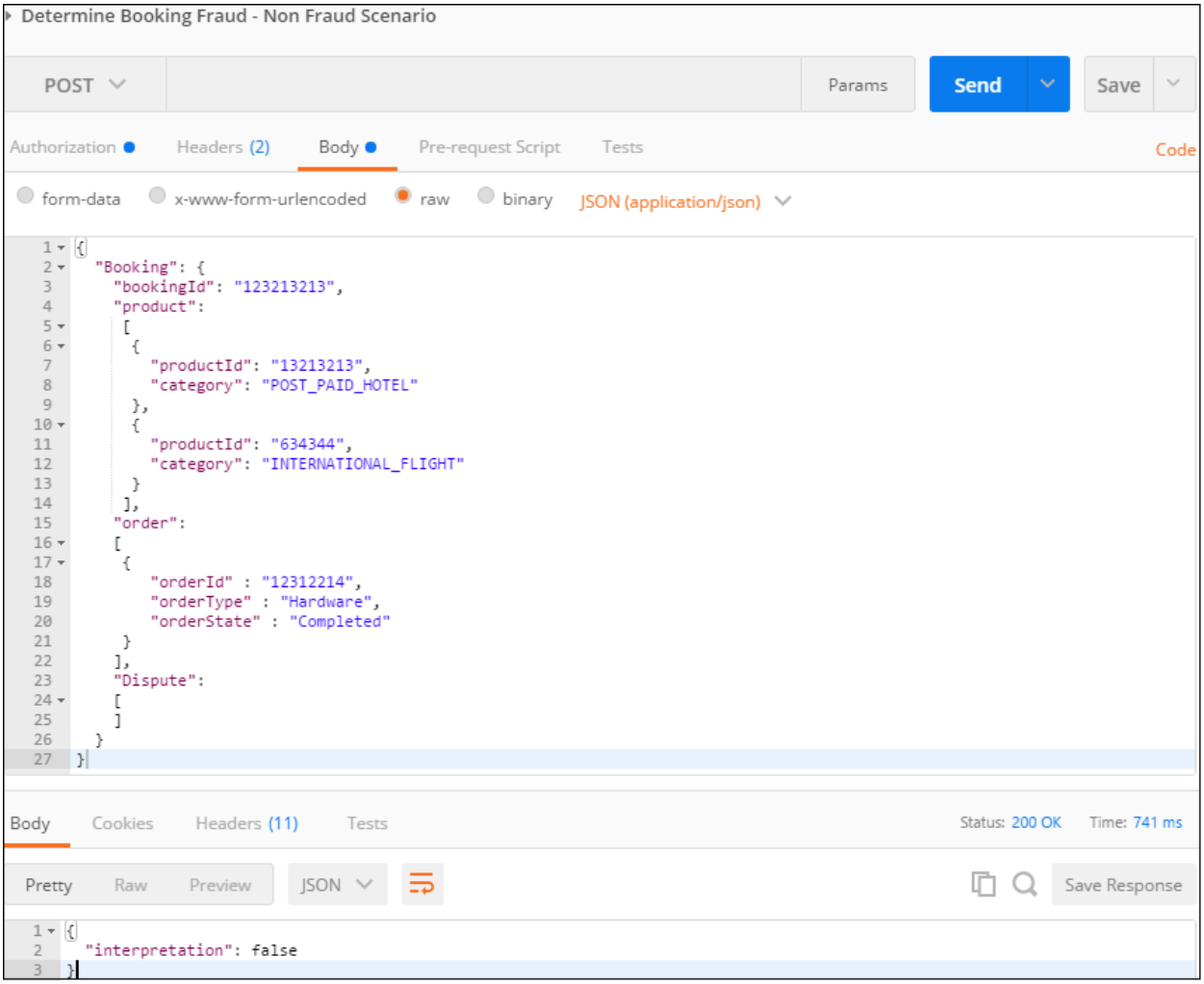
Scenario 1 – Non Fraud Scenario

Request

```
{
 "Booking": {
  "bookingId": "123213213",
  "product":
   [
    {
     "productId": "13213213",
     "category": "POST_PAID_HOTEL"
    },
    {
     "productId": "634344",
     "category": "INTERNATIONAL_FLIGHT"
    }
   ],
  "order":
  [
          {
                  "orderId" : "12312214",
                  "orderType" : "Hardware",
                  "orderState" : "Completed"
          }
  ],
  "Dispute":
  [
  ]
 }
}
```

Response

```
{
  "interpretation": false
}
```



The total score from the different combinations of products, orders and disputes is:

Products: 5+25 =30
Orders: 90
Disputes: 0
Total - 120

The Calculated Fraud Score is 120 which is less than the threshold value of 200 and hence it is not a case of fraud.

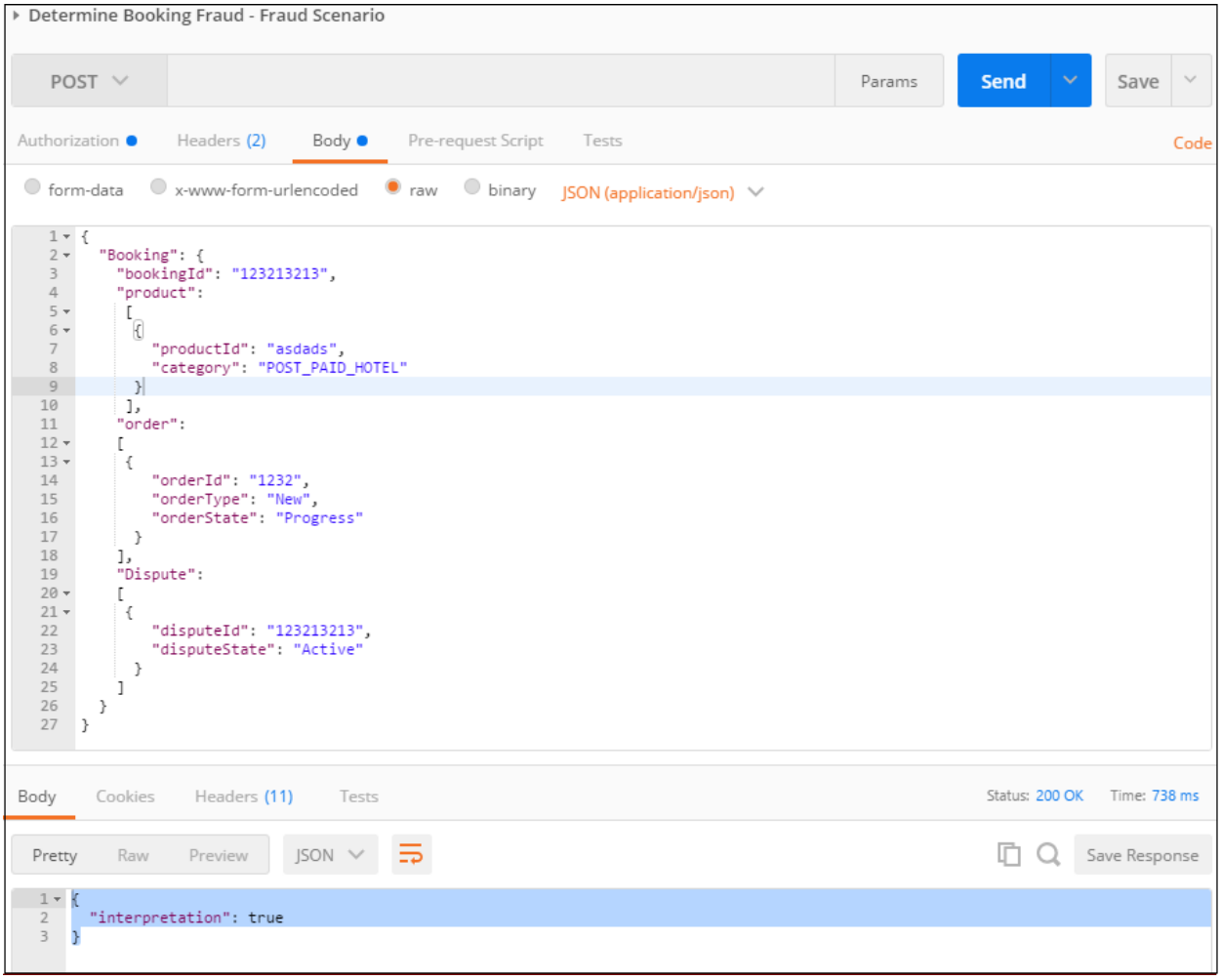
Scenario 2 – Fraud Scenario

```
{
 "Booking": {
  "bookingId": "123213213",
  "product":
  [
   {
```

```
      "productId": "asdads",
      "category": "POST_PAID_HOTEL"
    }
   ],
   "order":
   [
    {
      "orderId": "1232",
      "orderType": "New",
      "orderState": "Progress"
    }
   ],
   "Dispute":
   [
    {
            "disputeId": "123213213",
      "disputeState": "Active"
    }
   ]
  }
}
```


Response


```
{
  "interpretation": true
}
```



The total score from the different combinations of products, orders and disputes in this scenario is:

Products: 5
Orders: 90
Disputes: 190
Total - 285

The Calculated Fraud Score is 285 which is greater than the threshold value of 200 and hence it is a fraud.

Please post any feedback or questions regarding this solution to arun.pareek@rubiconred.com