BUSINESS ROLE BookingAgent

INTERACTIONS

/bookTrip
  / from, to: airport
  / out, in: date
  / fconf: fcode
  / hconf: hcode
  / amount: moneyvalue

/ log(username, password): bool
/ getData(username): usrd data
/ getCard(username): paydata
/s& / bookFlight
  / from, to: airport
  / out, in: date
  / traveller: usrd data
  / fconf: fcode
  / amount: moneyvalue
  / beneficiary: account
  / payService: serviceId
/s& / payment
  / amount: moneyvalue
  / beneficiary: account
  / originator: usrd data
  / cardNo: paydata
  / proof: pcode
/s& / bookHotel
  / checkin, checkout: date
  / traveller: usrd data
  / hconf: hcode
/snd / payAck
  / proof: pcode
  / status: bool
/rcv / ackRefundRcv
  / amount: moneyvalue
/snd / ackRefundSnd
  / amount: moneyvalue

SLA VARIABLES

BOOKFEE: [0..100], KD: [1..30]
**ORCHESTRATION**

local s:[START, LOGGED, QUERIED, FLIGHT_OK, HOTEL_OK, CONFIRMED, END_PAID, END_UNBOOKED, COMPENSATING, END_COMPENSATED], logged:bool, traveller:usrdata, travcard:paydata

**transition** HotelAnswer

<table>
<thead>
<tr>
<th>triggeredBy</th>
<th>bookHotelReply</th>
</tr>
</thead>
<tbody>
<tr>
<td>guardedBy</td>
<td>s=FLIGHT_OK</td>
</tr>
<tr>
<td>effects</td>
<td>bookHotelReply ⊃ s'=HOTEL_OK</td>
</tr>
<tr>
<td></td>
<td>¬bookHotelReply ⊃ s'=END_UNBOOKED</td>
</tr>
<tr>
<td>sends</td>
<td>bookHotelReply ⊃ bookTrip</td>
</tr>
<tr>
<td></td>
<td>bookTrip.fconf=bookFlight.fconf</td>
</tr>
<tr>
<td></td>
<td>bookTrip.amount=bookFlight.amount + BOOKFEE</td>
</tr>
<tr>
<td></td>
<td>bookTrip.hconf=bookHotel.hconf</td>
</tr>
<tr>
<td></td>
<td>¬bookHotel.Reply ⊃ bookFlight</td>
</tr>
<tr>
<td></td>
<td>¬bookTrip.Reply=False</td>
</tr>
</tbody>
</table>

**transition** TripCommit

<table>
<thead>
<tr>
<th>triggeredBy</th>
<th>bookTrip</th>
</tr>
</thead>
<tbody>
<tr>
<td>guardedBy</td>
<td>s=HOTEL_OK</td>
</tr>
<tr>
<td>effects</td>
<td>s'=CONFIRMED</td>
</tr>
<tr>
<td>sends</td>
<td>bookFlight ⊃ bookHotel ⊃ payment</td>
</tr>
<tr>
<td></td>
<td>payment.amount=bookFlight.amount + BOOKFEE</td>
</tr>
<tr>
<td></td>
<td>payment.beneficiary=bookFlight.beneficiary</td>
</tr>
<tr>
<td></td>
<td>payment.originator=traveller</td>
</tr>
<tr>
<td></td>
<td>payment.cardNo=travcard</td>
</tr>
</tbody>
</table>

**transition** PaymentAnswer

<table>
<thead>
<tr>
<th>triggeredBy</th>
<th>paymentReply</th>
</tr>
</thead>
<tbody>
<tr>
<td>guardedBy</td>
<td>s=CONFIRMED</td>
</tr>
<tr>
<td>effects</td>
<td>payment.Reply ⊃ s'=END_PA</td>
</tr>
<tr>
<td></td>
<td>¬payment.Reply ⊃ s'=END_UNBOOKED</td>
</tr>
<tr>
<td>sends</td>
<td>payAck</td>
</tr>
<tr>
<td></td>
<td>payAck.proof=payment.proof</td>
</tr>
<tr>
<td></td>
<td>payAck.status=payment.Reply</td>
</tr>
</tbody>
</table>

... 

**BUSINESS PROTOCOL**  
Customer is

**INTERACTIONS**

rs login  
   ⊳ usur:username, pwd:password  
rs bookTrip  
   ⊳ from,to:airport,  
out,in:date  
   ⊳ fconf:fcode,  
   hconf:hcode,  
   amount:moneyvalue  
snd payNotify  
   ⊳ status:bool  
snd refund  
   ⊳ amount:moneyvalue  

**SLA VARIABLES**

BOOKFEE:[0..100], KD:[1..30]

**BEHAVIOUR**

initiallyEnabled login?  
   (login! ∧ login.Reply) enables bookTrip?  
   (bookTrip? ∧ bookTrip!? ) ensures payNotify!  
   (payNotify!? ∧ payNotify! status) enables bookTrip!?  
until today=bookTrip.out+KD  
   (bookTrip!? ∧ today+KD=bookTrip.out) ensures refund?  
   ∧ refund.amount=bookTrip.amount-BOOKFEE
Define, for the module `TravelBooking`, an external policy on the SLA variables:

- **CR.PROCODE**: is a variable associated to `CR` that denote the promotional code used by the customer to obtain discounts,
- **CR.PERC, FA.PERC**: is a variable associated to `CR` that denote the percentage of refund,
- **CR.FEE**: is the variable associated to `CR` that denote the forfait price for each booking,

The external policy must contain a number of constraints that ensure:

a. That the percentage of refund concessed to the customer must always be between 50% and 100% and the flight agent must support this.

b. The degree of satisfaction is inversely proportional to the percentage of refund (PERC) concessed to the customer.

c. If the promotional code of the customer is “VIP” then **CR.FEE** is the zero, if it is “MEMBER” the satisfaction is directly proportional to **CR.FEE** but less than 10£ and if it is “OTHER” then it 11£.

**EXTERNAL POLICY**

**SLA variables**

```
CA.PERC, FA.PERC, CR.FEE, CR.PROCODE
```

**CONSTRAINTS**

```
C_1: \{CA.PERC, FA.PERC\}
\quad d(x,y) = \begin{cases} 
1 & \text{if } 50 \leq x \leq 100 \land x \leq y \\
0 & \text{otherwise}
\end{cases}
```

```
C_2: \{CA.PERC\}
\quad d(s) = \begin{cases} 
1/x & \text{if } x > 0 \\
0 & \text{otherwise}
\end{cases}
```

```
C_3: \{CR.FEE, CR.PROCODE\},
\quad def(f,p) = \begin{cases} 
1 & \text{if } (p = \text{"VIP"} \land f = 0) \lor (p = \text{"OTHER"} \land f = 11) \\
f/10 & \text{if } p = \text{"MEMBER"} \text{ and } f \leq 10 \\
0 & \text{otherwise}
\end{cases}
```

---

Continues on the next page
BUSINESS PROTOCOL Customer is

INTERACTIONS

r&s login
   user:username, pwd:password
r&s bookTrip
   from,to:airport,
  out,in:date
  fconf:fcode,
  hconf:hcode,
  amount:moneyvalue
snd payNotify
   status:bool
snd refund
   amount:moneyvalue

BEHAVIOUR

initiallyEnabled login? (login!  login.Reply) enables bookTrip?
  (bookTrip?  bookTrip!?) ensures payNotify!

Define the following statement for the business protocol Customer:

- That the percentage of refund ($Refund.amount$) concessed to the customer must always be between 50% and 100%.

\[50 \leq refund.amount \land refund.amount \leq 100\] after refund! !

- That the compensation of $bookTrip$ is always allowed after a $payNotify$ with a positive status (one statement) but that the amount refund (which is ensured after $payNotify$) will be zero on or after the day of the trip (one statement).

1- payNotify! !  payNotify.status enables bookTrip??
2- Refund.amount=0 after bookTrip.out \geq today
   (bookTrip?? ensures refund!) this was assumed.
- That (alternative to the previous) the compensation of bookTrip is not allowed on or after the day of the trip (one statement) but the amount of the refund is always as the one agreed with the SLA variable PERC (one statement).

1- payNotify! ∧ payNotify.status enables bookTrip? until bookTrip.out ≥ today

2- refund.amount=bookTrip.amount*PERC after refund!

(we should declare the SLA variable PERC in Customer)