**SIMULATION OF A POSSIBLE SCENARIO**

In the following lines we are going to provide you a simulation of how our prototype works. We assume to have four players: **two individuals** that create their own personal profile and buy a car insurance, the **Police** submitting accident’s reports as an official source, and the **Insurer** which collects the premia from the individuals and provides them coverage.

We have three solidity-based *Smart Contracts* that can be instantiated in order to represent our entities: **PersonalProfile,** which manages all the individual’s personal information, **InsurerContract**, which is used by the Insurer to manage the relationships with individuals, and **PoliceReport**, which enables the police to document traffic accidents. Note that every Smart Contract should be deployed using a different account, since each is used by a different entity.

**Assumptions:**

1. Individuals’ homogeneity: all individuals belong to the same risk class, pay the same price to the insurer and have the same maximum damage coverage. In our simulation, we fix the individual premium to 5 ethers and the maximum coverage to 8 ethers.
2. There is only one insurer in the market: the two individuals are covered by the same entity.
3. Accidents are standardized: Only two cars are involved in the accidents: there are no pedestrians and no casualties. Moreover, the fault is binary and is attributed to an individual by the police.
4. Police is the only authorized source of information.
5. Individuals do not behave opportunistically: This assumption has to do with all the limitations about building a prototype in the Remix console. Since accounts available change as the web page is reloaded, it is impossible to restrict function modifications to a single, verified account as the Police: in order to relax these restrictions we forced some functions not to be called by the owner of the contract. However, opportunistic behaviours are still technically feasible in our prototype. For instance, the owner of a Personal Profile cannot modify a Police Report pushed by the police in its own contract, but another individual can. An opportunistic behaviour could be two individuals involved in an accident that agree on modifying the other party’s Police Report to be both refunded by the Insurer. For the scope of this project we simply assume in our simulation that this behaviour will not take place; however, in a more advanced prototyping we would be able to specify inside modifiers which verified account can call some functions.

**Additional Considerations**:

* Both individuals pay a minimal amount of ether and gas as transaction fees when they pay the insurance premium or when they request a refund.
* In case of accident, under the assumption of *binary fault*, the faulty person will have to pay his damages on his own. Only the damages of the person without fault are displayed inside the Police Report, and only them are covered in the reimbursement procedure.
* In Italy the new regulatory procedure implies that if there are no cases of disability exceeding 9%, in most cases the *direct reimbursement procedure* is applied even without agreement between the parties. It means that the person who took the damage can apply to his own insurer and, if eligible, he will get the reimbursement directly from it. Then it will be up to his insurance company to assert its rights against the insurance company of the person who caused the accident, in order to get the money back. Therefore, we will cover all the direct reimbursement procedure, without touching the “dialogue” between the two insurances.

**Procedure**

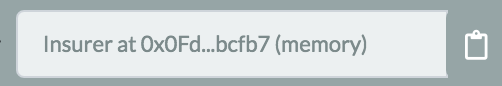
In a very logical way we start with the deployment of the insurance company through the InsurerContract:

* *Compile* InsurerContract.
* Select an account and *Deploy* the contract.
* Click on Balance, as you can see it is equal to 0, it means that the insurance company in this moment has no money. Try to click on *recorded\_accounts\_array* (it is a collection of all insured people’s accounts)*,* and as you can see it does not return anything, because no one has yet bought a policy.

Initialize Person\_1 with PersonalProfile:

* *Compile* PersonalProfile.
* Select an account (different from the previous one) and *Deploy* the contract.
* Fill the function *InsertPersonalInfo*: name, surname, birth, licenseID, vehicle (*license plate*). This function records all the information about a person, as if it were an official and uncorruptible document. Click on transact.

This information -  ideally - should be useful for estimating the risk premium, but in our simulation for simplicity we have assumed a homogeneous risk premium for all individuals

* Now if you click on *viewPersonalInfo*, you can see all the information.
* Try to use the function *UpdateHistory*: this function is used by the police to push a report - about a car accident - to a personal profile. As you can see the transaction is reverted, since the individual can just initialize his state but can’t update it, only the police can.
* If you click on *howManyAccidents* you can see that now we have 0 accidents.
* Now copy the InsurerContract address from the symbol (the one circled in red) 
* Paste it in the function *payAndgetInsurance*, select a value of 5 ethersand click on transact, in this way the Person\_1 provides all his personal information to the insurance company (be sure that the account is the one that you used to deploy the PersonalContract) and pays the risk premium.
* Go to the InsurerContract, in *recorded\_accounts\_array* select 0 (the index of the first person who passed his own information), as you can see now there is one address that is the account of Person\_1. Paste this address in the *viewInsured* function and as you can see now you have all the information about Person\_1, pay attention at the line 5 where you have *bool: true*, it means that the person has paid the risk premium and he is insured.
* Stay in the InsurerContract and click on *Balance*, as you can see now there are 5 ethers, this is a prof that the policy has been payed.

Initialize Person\_2 with PersonalProfile:

* Select a different account from the two previous ones.
* Repeat the same procedure as Person\_1 with different personal information and then pay the risk premium to be insured.
* Check if everything worked: go to *InsurerContract*, in the function *recorded\_accounts\_array* select 1 (the index of the second person that passed the information), copy that address and paste it in *viewInsured,* now you can see all the information about Person\_2 and the line 5 is *bool true*.
* Check also the balance of the Insurer, it is 10 Ether, that are the two risk premia now.

**NOTE**: Both Person\_1 and Person\_2 for each transaction pay a little amount of ether, but it is a necessary cost to buy the insurance policy.

Now suppose that Person\_1 did not give priority to Person\_2 and caused an accident.

Since it is fault of Person\_1 while Person\_2 has no fault, the latter will be eligible to get a refund while the former should pay with his own money his own damage. 

Deploy PoliceReport: this smart contract, as we have already said, is an official source and it is authorized by the police:

* *Compile* the PoliceReport
* Select a different account that this time represents the police and click on *Deploy*.
* Select the function *updateReport* and insert all the information:

\_address1: address of the PersonalProfile of Person\_1;

\_address2: address of the PersonalProfile of Person\_2;

  first\_vehicle: the license plates of Person\_1 car;

second\_vehicle: the license plates of Person\_2 car;

fault: the ID licence of Person\_1, since in this case it’s his fault;

description: just a brief description of the accident;

damage: the estimated amount of damage of the car ‘without fault’, let’s select for this case 2 ethers.

* Now, if you click on *howManyReports,* it returns the number 1 and if you select 0 in the function *viewReport* you can see all the information about the report that we have just recorded.

**NOTE**: In this way the police updates both the history of PersonalProfile of Person\_1 and Person\_2, even though in this simulation we have not included the computation of the risk premium, information about ‘history’ would be very useful for the insurer in order to compute it.

The police pays a little amount of ether in order to update the history of both people, but we assume that they are essential costs that fall under the Public Administration.

Go to the personalprofile of Person\_2 (select the right account!)

* Click on *howManyAccident*, as you can see now it returns 1, and if in the function *getAccident* you write 0 (the index of the first accident) the function returns all the information about the first car accident.
* In the function *getRefund* copy paste from  both the address of the InsurerContract and Person\_2 PersonalProfile address, in the index field select the index of the accident for which you want to request a refund, in this case is 0, then click on transact.
* Now click on *MyBalance*, as you can see now it amounts at 2 Ether! That is the amount of the damage written in the PoliceReport.

Before of providing the reimbursement, the insurer has checked two conditions related to Person\_2 that must be True:

1. Person\_2 must have paid the risk premium and therefore he must be insured;
2. With regard to the accident, it’s not his fault.

If these two conditions are satisfied the Insurer automatically provide the refund.

* Now if you go in the InsurerContract and click on Balance, it is 8 ether not anymore 10 ehter, it means that the Insurer took the money from its *Balance* and paied its insured.
* In *RefundQueue* select 0 and click, it returns an address, copy and paste that address in the function *ShowRefundRequest* and you can observe the information about the refund that that person requested.

Go to the Personalprofile of Person\_1, selecting the right account:

* Click on *howManyAccident*, as you can see now you have 1, and if in the function *getAccident* you write 0 it returns all the information about the first car accident.
* In the function *getRefund* copy paste both the address of the InsurerContract and Person\_1 PersonalProfile address, in the index select the index of the accident for which you want to request a refund - in this case is 0 – click on transact.
* This time it’s different from the previous procedure, indeed the transaction is reverted, because Person\_1 – since it’s his fault – is not entitled to get a refund.
* If you click on *MyBalance* you have 0 and if you go to the InsurerContract you can observe that anything has changed. The insurer Balance is still 8 ether.

**NOTE**: Both people in order to request the refund pays a little amount of ether, but we decided that the cost of notifying the accident is at their expense. It represents a sort of effort that even in practice they should have taken to notify the accident to their insurance company.