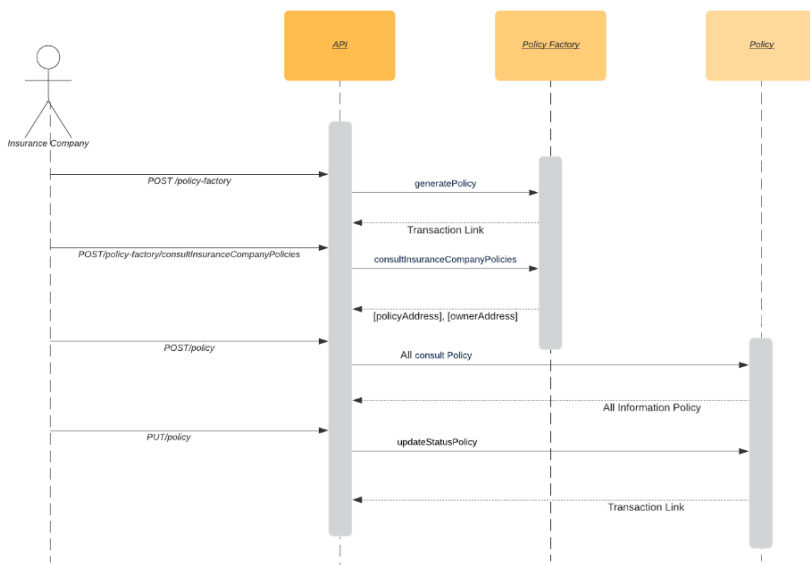


# Policy Manual

## Overview

Policy API provides the services necessary to administer and execute the functions of the Policy Factory and Policy smart contract.

The Policy Factory smart contract has the functions of creating policy smart contracts (Policy). A smart contract is created for each policy, recording its information on the blockchain. Each smart contract Policy can perform query and status change functions.



## Smart Contract Administration

To manage the policy smart contract, it is necessary to have a BFT wallet. With the address of that wallet the contract owner can add it as a smart contract administrator.

owner	The address of the wallet that published the smart contract on the network. This address is inserted in all smart contract roles at the time of publication.
carteira	Wallets allow BFT users to store BFT and interact with smart contracts on the BFT network.
address	The BFT address is a unique identifier derived from public keys. Private Key -> Public Key -> BFT Address

smart contract	A smart contract is the application code published on the BFT blockchain network. Executes the business rules defined for the policy.
----------------	---

## Administrative functions

The roles in a smart contract define the roles performed by each entity. The policy's smart contract has the following roles: Insurance Company and Admin.

### Admin role

- Responsible for controlling the functions of **adding** and **removing** other admins.

API Method	Smart Contract Function
POST/role/admin	addAdmin
DELETE/role/admin	removeAdmin

- Responsible for controlling the functions of **adding** and **removing** insurance companies.

API Method	Smart Contract Function
POST/role/insurance-company	addInsuranceCompany
DELETE/role/insurance-company	removeInsuranceCompany

- Responsible for **pausing** / **activating** the smart contract when necessary

API Method	Smart Contract Function
POST/lifecycle/pause	pause
POST/lifecycle/unpause	unpause

### Insurance Company role

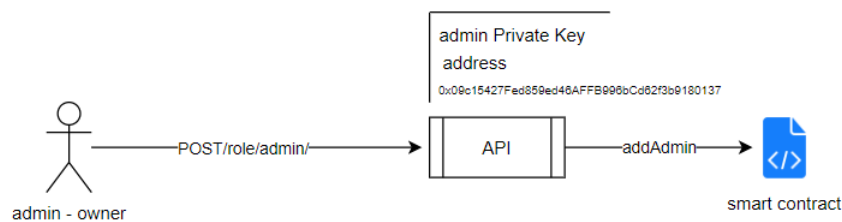
This role gives you permission to create policies, change a policy's status, and perform policy queries.

API Method	Smart Contract Function
POST /policy-factory	generatePolicy
POST/policy-factory/consultInsuranceCompanyPolicies	consultInsuranceCompanyPolicies

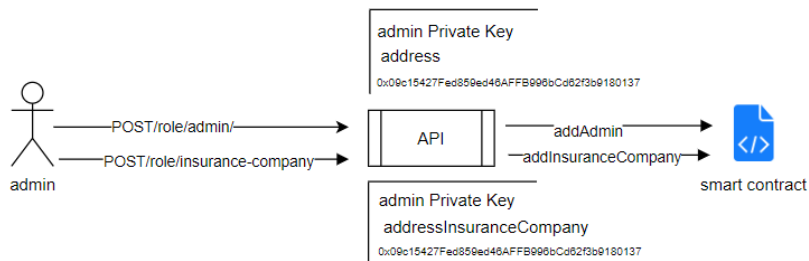
POST/policy	consultPolicyCoverage; consultDriver; consultPolicyHolder; consultPolicyInformation; consultPolicyInsuranceCompany; consultVehicle; consultVehicleDocuments
PUT/policy	updateStatusPolicy

## Create policy

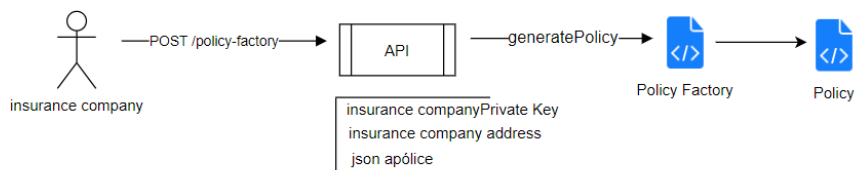
After publishing the smart contract on the BFT network, the owner of the contract that is automatically added to the role admin, and can choose who will be the administrator of that smart contract and add them to the role admin.



This admin role can control who is added or removed from both the admin role and the insurance company role. Then the admin adds the insurers to the role insurance company. Every insurer must have an BFT wallet in order for the address of that wallet to be added to the role insurance company by the admin.



Policy is created and registered on the blockchain once authorized insurance companies submit all parameters.



## Run a method using swagger

swagger	<a href="http://54.90.5.188:8091/swagger/">http://54.90.5.188:8091/swagger/</a>
Rinkeby	<a href="https://rinkeby.etherscan.io/address/0x059369f9866f9a4dd74a86c068f676e4d76856b">https://rinkeby.etherscan.io/address/0x059369f9866f9a4dd74a86c068f676e4d76856b</a>

On the swagger page you can select a GET / POST / PUT method and click on the button ***try it out***.

GET methods can be performed by entering the parameters in the required fields.

POST / PUT methods can be executed by editing the body with the necessary parameters.

After editing, click **Execute** and the method will be executed.

Method	Description
<b>POST</b>	The method HTTP <code>POST</code> is used to create resources or to send information that will be processed. For example, creating a claim or report, consulting the claim and report information.
<b>PUT</b>	The method HTTP <code>PUT</code> is used to update an existing resource. For example, changing the status.
<b>GET</b>	The method HTTP <code>GET</code> is used to query existing resources. For example, query for roles.

## Methods

### Policy Factory

Policy Factory methods must be signed by addresses that have the permission of insurers within the smart contract. This signature is called a private key. The smart contract admin first needs to add the insurer's address to the Insurance Company role, from there the insurer will be allowed to create policies and consult policies using their private key.

## POST /policy-factory

---

Through this method it is possible to create and register a policy on the blockchain. You must be on the role Insurance Company.

### Parameters

- private Key - Whoever carries out this transaction must be in the role "insurance Company". This private key must be from an insurance company
- insurance Company - Addresses of the insurers participating in this policy.
- Parameters:

### ▼ Json exemplo

```
{
  "privateKey": "7FDE0137C2E831A713A7415FB1256E7605EBFF04020345EB4E043B0C95BF27BD",
  "insuranceCompany": {
    "wallets": ["0x71fC0e20F2DA5853CE9174A54093f5b2918cCfa2",
"0x17cA6A08758F4A078B9c53ca25E6F6736dF34094"]
  },
  "coverage": {
    "prizeAmount": "1000",
    "fiipePercentage": "5",
    "app": "1",
    "glasses": 1,
    "rcfMaterials": "1",
    "rcfBodily": "1",
    "reserveCar": 1,
    "franchise": 1,
    "productCoverage": [
      "1", "2"
    ]
  },
  "policyInformation": {
    "proposal": "1",
    "apolice": "13334",
    "startValidity": 1586305780,
    "endValidity": 1617841780,
    "apoliceStatus": 1
  },
  "policyHoldedrData": {
    "nameComplete": "client",
    "dateOfBirth": 492308980,
    "maritalStatus": 1,
    "pocket": "1",
    "cnpjCpf": "111111111111",
    "gender": 1,
    "relationshipPolicyHolder": 1
  },
  "vehicleData": {
    "typeParam": 1,

```

```

        "marker": 1,
        "model": 1,
        "numSlides": 5,
        "yearManufacture": 2000,
        "yearModel": 2001,
        "licensePlate": "11111",
        "chassis": "1111111111",
        "renavam": "1111111111",
        "fuel": 1,
        "newVehicle": 1,
        "vehicleFinaced": 1,
        "color": 3
    },
    "driverData": {
        "nameComplete": "client",
        "dateOfBirth": 492308980 ,
        "maritalStatus": 1,
        "pocket": "1",
        "cpfCnpj": "111111111111",
        "gender": 1,
        "profession": 1,
        "cnh": "111111",
        "dateFirstCnh": 955153780 ,
        "garage": 1,
        "usesWork": 1,
        "vehicleUse": 1
    }
}

```

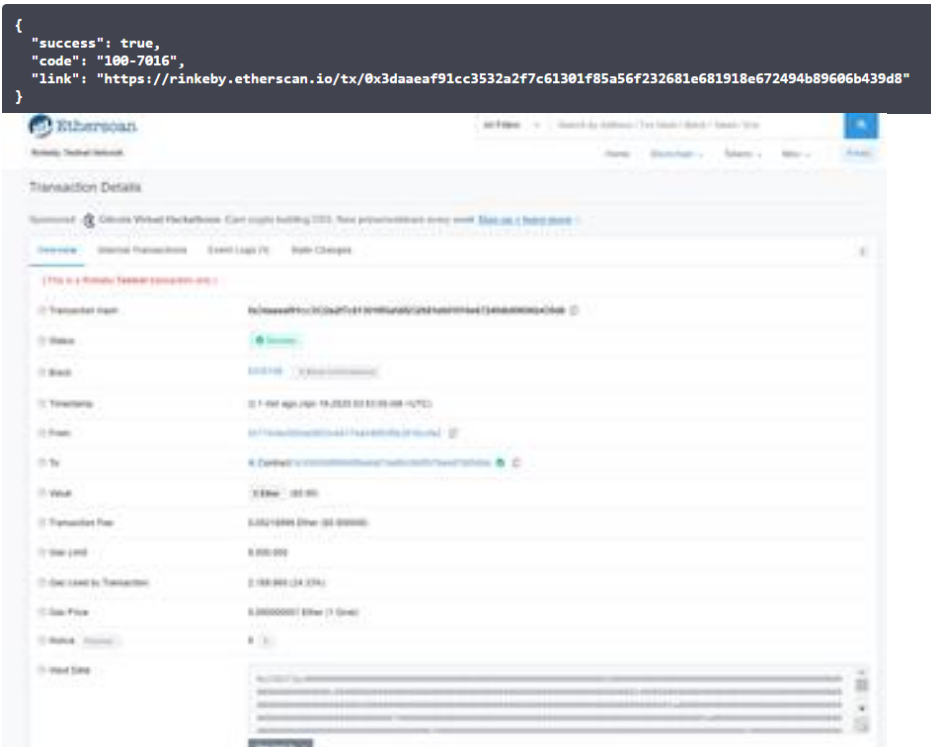
```

{
  "privateKey": "7FDE0137C2E831A713A7415F81256E7605EBFF04020345EB4E043B0C958F278D",
  "insuranceCompany": {
    "wallets": ["0x1fC0c20F2DAS853CE9174A54093f5b2918cCfa2", "0x17cA6A0B75BF4A07889c53ca25E6F6736dF34094"]
  },
  "coverage": {
    "prizeAmount": "1000",
    "fipePercentage": "5",
    "app": "1",
    "glasses": 1,
    "rcfMaterials": "1",
    "rcfBodily": "1",
    "reserveCar": 1,
    "franchise": 1,
    "productCoverage": [
      "1", "2"
    ]
  },
  "policyInformation": {
    "proposal": "1",
    "apolice": "13334",
    "startValidity": 1586305780,
    "endValidity": 1617841780,
    "apoliceStatus": 1
  },
  "policyHolderData": {
    "nameComplete": "client",

```

## Response

- link - Blockexplorer link for the transaction



## POST/policy-factory/consultInsuranceCompanyPolicies

Through this method it is possible to list all insurance company policies and owners registered on Blockchain. You must be on the role Insurance Company.

### Parameters

- private Key - Whoever carries out this transaction must be in the role "insurance Company". This private key must be from an insurance company
- address insurance Company - Address of the insurer participating in this policy

```
{
  "privateKey": "7FDE0137C2E831A713A7415FB1256E7605EBFF04020345EB4E043B0C95BF27BD",
  "addressInsuranceCompany": "0x71fC0e20F2DA5853CE9174A54093f5b2918cCfa2"
}
```

### Response

- result- lists *contract address policy* and *address insurance company*

[[contract address policy], [address insurance company]]

```
{
  "success": true,
  "code": "100-1000",
  "result": [
    [
      "0xD8fDe9b9c0fda8eb3665A468542Dc591d478C958"
    ],
    [
      "0x71fC0e20F2DA5853CE9174A54093f5b2918cCfa2"
    ]
  ]
}
```

## Policy

Policy methods must be signed by addresses that have the permission of insurers within the smart contract. This subscription is called a *private key*. The smart contract admin first needs to add the insurer's address to the Insurance Company role, from then on the insurer will be allowed to change the status and consult policies using their *private key*.

## POST/policy

---

Through this method it is possible to return all information from the Policy registered on the Blockchain. You must be on the role Insurance Company.

### Parameters

- private Key - Whoever carries out this transaction must be in the role "insurance Company". This private key must be from an insurance company
- contract address - Policy smart contract address that wants to retrieve information.

```
{
  "privateKey": "7FDE0137C2E831A713A7415FB1256E7605EBFF04020345EB4E043B0C95BF27BD",
  "contractAddress": "0xD8fDe9b9c0fda8eb3665A468542Dc591d478C958"
}
```

### Response

#### ▼ Exemplo

```
{
  "success": true,
  "code": "100-1000",
  "data": {
    "insuranceCompany": [
      "0x71fC0e20F2DA5853CE9174A54093f5b2918cCfa2",

```



```
"0x17cA6A08758F4A078B9c53ca25E6F6736dF34094"
],
"coverages": [
  "1000",
  "5",
  "1",
  1,
  "1",
  "1",
  1,
  1,
  [
    "1",
    "2"
  ]
],
"policyInformation": [
  "1",
  "13334",
  {
    "_hex": "0x5e8d1af4"
  },
  {
    "_hex": "0x606e4e74"
  },
  1
],
"policyHoldedrData": [
  "client",
  {
    "_hex": "0x1d5809f4"
  },
  1,
  "1",
  "11111111111",
  1,
  1
],
"vehicleData": [
  1,
  {
    "_hex": "0x01"
  },
  {
    "_hex": "0x01"
  },
  5,
  2000,
  2001,
  1,
  1,
  3
],
"vehicleDocuments": [
  "11111",
  "1111111111",
  "1111111111",
```

```
1
],
"driverData": [
  "client",
  {
    "_hex": "0x1d5809f4"
  },
  1,
  "1",
  "111111111111",
  1,
  {
    "_hex": "0x01"
  },
  "111111",
  {
    "_hex": "0x38ee7d74"
  },
  1,
  1,
  1
]
}
```

```
{
  "success": true,
  "code": "100-1000",
  "data": {
    "insuranceCompany": [
      "0x71fC0e20F2DA5853CE9174A54093f5b2918cCfa2",
      "0x17cA6A08758F4A078B9c53ca25E6F6736dF34094"
    ],
    "coverages": [
      "1000",
      "5",
      "1",
      1,
      "1",
      "1",
      1,
      1,
      [
        "1",
        "2"
      ]
    ],
    "policyInformation": [
      "1",
      "13334",
      {
        "_hex": "0x5e8d1af4"
      }
    ]
  }
}
```

## PUT/policy

---

Through this method it is possible to change the value of the policy status on the blockchain. You must be on the role Insurance Company.

### Parameters

- private Key - Whoever carries out this transaction must be in the role "insurance Company". This private key must be from an insurance company
- contract address - Address of the Policy smart contract that will have its information retrieved
- status - New status



### Response

## Lifecycle

The so-called lifecycle methods must be signed by an address that has admin permission within the smart contract. This signature is called a private key. A smart contract admin first needs to add the address in the role admin, from there they will be allowed to pause or activate the smart contract Policy Factory using their private key.

### POST/lifecycle/pause

---

Through this method, it is possible to pause the smart contract to create a policy factory policy. You must be in the Admin role.

#### Parameters



#### Response

### POST/lifecycle/unpause

---

Through this method it is possible to activate the function of creating policy in the smart contract Policy Factory. You must be in the Admin role.

#### Parameters



#### Response

### GET/lifecycle/paused/{contract}

Through this method, it is possible to check the status of the Policy Factory smart contract.

#### Parameters

Just run the method.

#### Response

```
{
  "success": true,
  "code": "100-2030",
  "result": false
}
```

## Roles

The methods called roles need to be signed by address that have admin permission within the smart contract. This signature is called a private key. A smart contract admin first needs to add the address to the role admin, from there he will be allowed to add or remove the address of the smart contract Policy Factory using his private key.

### GET/role/admin

---

Through this method it is possible to check if an address is an administrator in the smart contract Policy Factory.

#### Parameters

- address - wallet address to be consulted

**address** \* required  
(query)

Informe o Endereço na Blockchain

0x71fC0e20F2DA5853CE9174A54093f5b29

#### Response

```
{
  "success": true,
  "code": "100-2030",
  "result": true
}
```

is admin = true

```
{
  "success": false,
  "code": "100-2031",
  "result": false
}
```

is admin = false

## POST/role/admin

Through this method, it is possible to add an administrator to the smart contract Policy Factory. You must be in the Admin role.

### Parameters

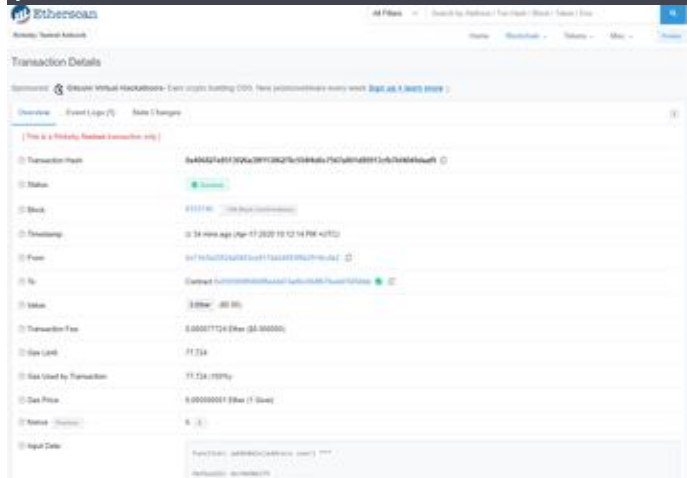
- private Key - Whoever carries out this transaction needs to be in the "admin" role. This private key must be from a Policy Factory smart contract administrator.
- address - Wallet address that will be added as an administrator

```
{
  "privateKey": "7FDE0137C2E831A713A7415FB1256E7605EBFF04020345EB4E043B0C95BF27BD",
  "address": "0xc9D668028A266FaFDFbEdD7A7ec54b0dFfEB2f61"
}
```

### Response

- link - Blockexplorer link for the transaction

```
{
  "success": true,
  "code": "100-2010",
  "link": "https://rinkeby.etherscan.io/tx/0x406827e8513026a38f11386278c554f4d6c7567a801d80913cfb7bf4049daaf9"
}
```



The screenshot shows the Etherscan interface for a transaction on the Rinkeby testnet. The transaction is successful, with a status of 'Success'. The transaction hash is 0x406827e8513026a38f11386278c554f4d6c7567a801d80913cfb7bf4049daaf9. The transaction was sent from the address 0x7fde0137c2e831a713a7415fb1256e7605ebff04020345eb4e043b0c95bf27bd to the contract address 0xc9d668028a266fafdfbedd7a7ec54b0dffeb2f61. The transaction fee was 0.00011214 Ether (35,300 Gwei) and the gas price was 11,734 Gwei. The transaction was mined at 11:23:14 PM on 11/23/2019.

## DELETE/role/admin

---

Through this method it is possible to remove an administrator in the smart contract Policy Factory. You must be in the Admin role.

### Parameters

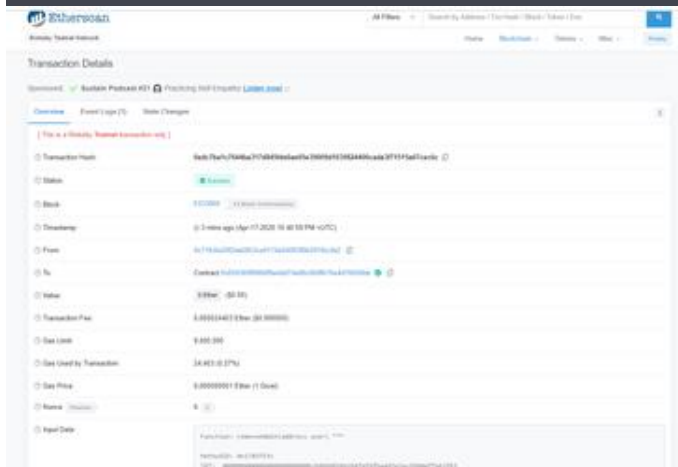
- private Key - Whoever carries out this transaction needs to be in the "admin" role. This private key must be from a Policy Factory smart contract administrator
- address - Wallet address that will be added as an administrator

```
{
  "privateKey": "7FDE0137C2E831A713A7415F81256E7605EBFF04020345EB4E043B0C958F278D",
  "address": "0xc9D668028A266FaFDfEdD7A7ec54b0dFfEB2f61"
}
```

### Response

- link - Blockexplorer link for the transaction

```
{
  "success": true,
  "code": "100-2020",
  "link": "https://rinkeby.etherscan.io/tx/0xdc7ba1c7644ba317d845fde0ae05e39009d1039824400cada3f71515a67cec6c"
}
```



## GET/role/insurance-company

---

Through this method it is possible to check if an address is in the insurance role in the policy factory smart contract.

### Parameters

- addressInsuranceCompany - Wallet address to be consulted

Name	Description
addressInsuranceCompany <small>* required</small> (query)	Informe o Endereço da companhia de seguro <div>0x71fC0e20F2DA5853CE9174A54093f5b29</div>

## Response

```
{
  "success": true,
  "code": "100-1000",
  "result": true
}
```

is InsuranceCompany = true

```
{
  "success": true,
  "code": "100-1000",
  "result": false
}
```

is InsuranceCompany = false

## POST/role/insurance-company

---

Through this method, it is possible to add an insurance company to the Policy Factory smart contract. You must be in the Admin role.

### Parameters

- private Key - Whoever carries out this transaction needs to be in the "admin" role. This private key must be from a Policy Factory smart contract administrator
- addressInsuranceCompany - Wallet address to be added to the role insurance company

```
{
  "privateKey": "7FDE0137C2E831A713A7415FB1256E7605EBFF04020345EB4E043B0C95BF27BD",
  "addressInsuranceCompany": "0xc9D66B028A266FaFDfEdD7A7ec54b0dFfEB2f61"
}
```

## Response



- link - Blockexplorer link for the transaction

```
{
  "success": true,
  "code": "100-1000",
  "link": "https://rinkeby.etherscan.io/tx/0x06f06bd05ea68fb2625189043009a2502c6e06c68f127685f2233e557c9b6148"
}
```

The screenshot shows the Etherscan interface for a transaction on the Rinkeby testnet. The transaction is successful, with a status of 'Success' and a link to the transaction details. The transaction hash is 0x06f06bd05ea68fb2625189043009a2502c6e06c68f127685f2233e557c9b6148. The transaction was sent from 0x7fde0137c2e831a713a7415fb1256e7605ebff04020345eb4e04380c95bf278d to a contract address 0x0c9d66b028a266fafdfbed7a7ec54b0dfefb2f61. The transaction value is 0 Ether, and the gas used is 77,746.

## DELETE/role/insurance-company

Through this method it is possible to remove an insurer from the Policy Factory smart contract. You must be in the Admin role.

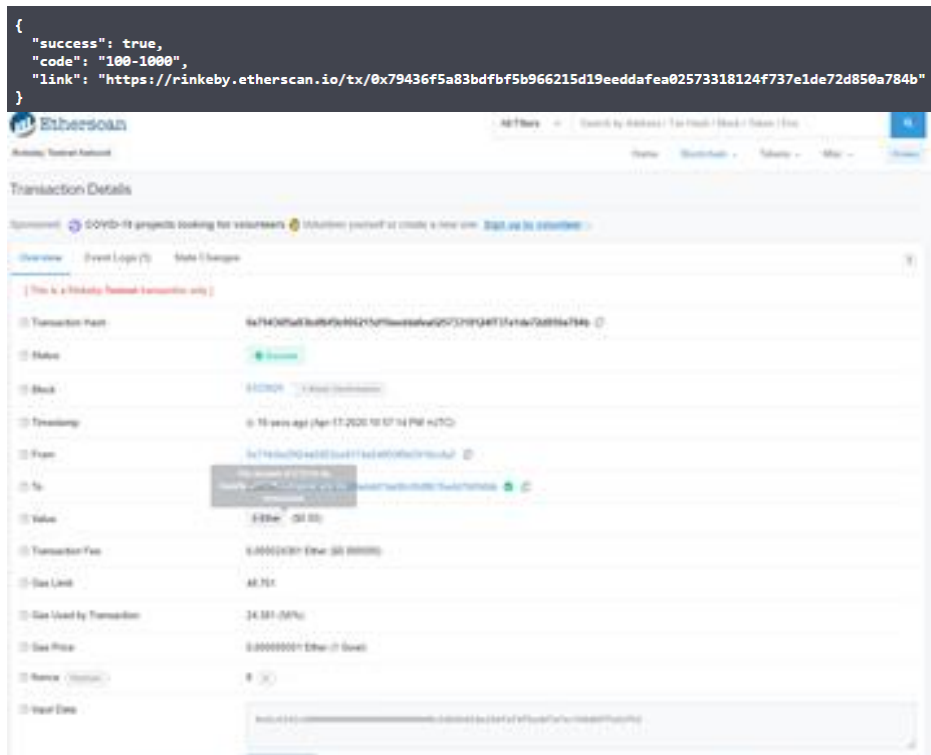
### Parameters

- private Key - Whoever carries out this transaction needs to be in the "admin" role. This private key must be from a Policy Factory smart contract administrator
- addressInsuranceCompany - Wallet address to be added to the role insurance company

```
{
  "privateKey": "7FDE0137C2E831A713A7415FB1256E7605EBFF04020345EB4E04380C95BF278D",
  "addressInsuranceCompany": "0xc9D66B028A266FaFDFbEdD7A7ec54b0dFfEB2f61"
}
```

### Response

- link - Blockexplorer link for the transaction



## Policy parameters

Category	Variable	Description	Type
privateKey		Private key of the signer (msg.sender)	string
mutual			
	wallets	Numbers of the portfolios of each insurance company belonging to the policy.	address[]
Coverage			
	prizeAmount	Coverage amount	uint256

	fipePercentage	Fipe table index	uint256
	app	Personal Passenger Accidents	string
	glasses	Glass coverage	uint256
	rcfMaterials	Optional Liability for Material Damage	string
	rcfBodily	Optional Liability for bodily harm	string
	reserveCar	If the paid coverage has a reserve car	uint256
	franchise	Mandatory participation of the insured in a claim	uint256
	productCoverage	Insured product	string[]
policyInformation			
	proposal	Identification of the Proposal that generated the policy	string
	apolice	Policy identification (internal)	string
	startValidity	Effective date	uint256
	endValidity	Effective end date	uint256

	apoliceStatus	Policy status	uint256
PolicyHolderData			
	nameComplete	Insured's name	string
	dateOfBirth	Date of birth	uint256
	maritalStatus	Marital status	uint256
	pocket		string
	cnpjCpf	CPF / CNPJ	string
	gender	Gender	uint256
	relationshipPolicyHolder	Degree of kinship	uint256
VehicleDataModel			
	typeParam	Type of vehicle	uint256
	maker	Maker	uint256
	model	Model	uint256
	numerSlides		uint256
	yearManufactur	Year of manufacture	uint256
	yearModel	Model year	uint256
	licensePlate	License plate	string
	chassis	Chassis	string
	renavam	Renavan	string
	fuel	Fuel type	uint256
	newVehicle	Brand new vehicle	uint256
	vehicleFinaced	Financed	uint256
	color	Color	uint256

DriverDataModel			
	nameComplete	Driver name	string
	dateOfBirth	Date of birth	uint256
	maritalStatus	Marital status	uint256
	pocket		string
	cpfCnpj	CPF / CNPJ	string
	gender	Gender	uint256
	profession	Profession	uint256
	cnh	Drive license	string
	dateFirstCnh	First license emission	uint256
	garage	Vehicle stays in garage	uint256
	usesWork	Vehicle is used for work	uint256
	vehicleUse	Type of vehicle use	uint256