API Token

API information

Title	HBTrust.domain.token
Description	Provides services for the generating PoP wallet, PoP consultation and management of the Insurance token.

PIT Token

The available Insurance token management services are:

- Management of Minter, Burner and Admin functions (add, remove and consult)

Function	**Description**
Minter	Address / Person responsible for generating new Tokens.
Burner	Address / Person responsible for burning tokens.
Admin	Address / Person responsible for maintaining Roles - Add or Remove members from all roles in the Contract. Admin will also be responsible for pausing / unpausing the contract.

- PIT Token management

mintTo (address _to, uint256 _value)	Allows Minter to generate new tokens. Emits the mint event to the declared address. Increases totalSupply.
burnFrom (address _from, uint256 _value)	Allows Burner to "burn" tokens. Emits the burn event for the declared address. Decreases totalSupply.
transfer (address _to, uint _value)	Function that allows tokens to be transferred from a declared address.
balanceOf (address_owner)	Allows any address to view the balance in Tokens.

API Methods

Input Parameters

Below is the list of parameters expected for the API to perform its main function. This input information will be provided by the platform that will consume this API - so any error in sending the parameters will directly affect the functioning of the same.

Content type: `application/json`

Method	**Input parameters**	**Mandatory?**	**Token Function**	**Descrição**
POST/mintTo	`privateKey, _sender, address _to, _value`	Yes	mintTo	Token creation
POST/burnFrom	`privateKey, _sender, address _to, _value`	Yes	burnFrom	Token destruction
POST/generateWallet	`quantity (integer)`	Yes		Wallet generation
POST/transfer	`privateKey, _sender, address _to, _value`	Yes	transfer	Transfer PIT between wallets
GET/balance	`address_owner`	Yes	balanceOf	Check PIT balance of wallet
GET/balanceEther	`address_owner`	Yes		Return PoP balance of hotwallet
GET/verifyRole	`address _account / role`	Yes	• isAdmin; • isBurner; • isMinter	Check roles of an adress inside a smart contract
POST/addRole	`privateKey, _sender, _account, _role`	Yes	• addAdmin; • addBurner; • addMinter	Add address to role

POST/removeRole	privateKey, _sender, _account, _role	Yes	• removeAdmin; • removeBurner; • removeMinter	Remove address from role

Output Parameters

Below is the list of parameters returned:

Service	Parameter	Description
GenerateWallet	Address	Hash (public key) generated to represent the wallet address on the blockchain. Example:0xFE...9dC8Cb84E7d6Fac
GenerateWallet	Seed Phrase (Mnemonic)	"Phrase" with 12 words randomly generated to recover the Private Key and access the wallet.
GenerateWallet	Private Key	Private key to access wallet and sign transactions.
addRole / removeRole	hash	Transaction info
verifyRole	true / false	Informs if the address has the role of admin, minter or burner
balancePop	balance	PoP Balance
balance	balance	PIT Token balance
mintTo	hash	Successful transaction information
burnFrom	hash	Successful transaction information
transfer	hash	Successful transaction information

Field dictionary

Variable	Description	Type	Example
Address	Wallet address	string address	0x09c15427Fed859ed46AFFB996bCd62f3b9180137
quantity	Integer value. Number of wallets to be created.	int	12

`privateKey`	Private key of who is signing the transaction msg.sender	`string`	A9AC29CBEB110215AE6D5AF5D8731A848160A3A7DABEF198B83
`_sender`	Wallet address of the person signing the transaction	`string address`	0x09c15427Fed859ed46AFFB996bCd62f3b9180137
`_to`	Wallet address that will receive the tokens	`string address`	0x09c15427Fed859ed46AFFB996bCd62f3b9180137
`_from`	Wallet address where tokens will be burned from	`string address`	0x09c15427Fed859ed46AFFB996bCd62f3b9180137
`_amount`	Number of tokens that will be sent or deleted	`uint256`	200
`_account`	Wallet address that will be added or removed from the role	`string address`	0x09c15427Fed859ed46AFFB996bCd62f3b9180137
`_role`	Role name: admin, minter or burner	`string`	admin / minter or burner

Requests / Responses

Authors	Gabriela Quero Pereira (Unlicensed) Flavio Gouveia

GET / Version

Through this method it is possible to check the version of this API

The method receives the following parameters in return:

```
{
    "title": "HBTrust.Domain.Token",
    "version": "0.0.1"
}
```

GET / Balance

Through this method it is possible to check the PIT Token balance of a wallet.

To execute this method it is necessary to inform the PATH the address of the wallet, for example:

```
0xf9c744832a2EE4D6f2256DC7BBaAb5f38273De76
```

The method receives the following parameters in return:

```
{
    "success": true,
    "code": "100-1000",
    "message": "Success!",
    "balance": 7548
}
```

GET / Balance PoP

Through this method it is possible to check the PoP balance of a wallet.

To execute this method it is necessary to inform the PATH the address of the wallet, for example:

```
0xf9c744832a2EE4D6f2256DC7BBaAb5f38273De76
```

The method receives the following parameters in return:

```
{
    "success": true,
    "code": "100-1000",
    "message": "Success!",
    "balance": "18.6554607026"
}
```

GET / Verify Role

Through this method it is possible to check the roles (admin, minter or burner) of a wallet.

To execute this method, it is necessary to inform PATH the address of the wallet being verified, such as:

```
0xf9c744832a2EE4D6f2256DC7BBaAb5f38273De76
0xf9c744832a2EE4D6f2256DC7BBaAb5f38273De76
```

The method receives the following parameters in return:

```
{
    "success": true,
    "code": "100-7013",
    "message": "Endereço da carteira possui as funções Admin, Minter e 
Burner.",
    "admin": true,
    "minter": true,
    "burner": true
}
```

POST / Generate Wallet

Through this method it is possible to create Wallets.

To execute this method it is necessary to inform the following parameters in the request body, for example:

```
{
  "quantity": 1 // quantidade de wallets que serão criadas
}
```

The method receives the following parameters in return:

```
{
  "success": true,
  "result": [
    {
      "index": 1,
      "mnemonic": "month leave someone zebra clap explain account page tired 
put robot ancient",
      "address": "0xC0fDcfc1E468c6dB915D12379670a6914ED6e758",
      "privateKey": 
"9798fee0fca3f8d7a5cc36fb8ae822180170ed96164eb8acc8dad30bb60246ee"
    }
  ]
}
```

POST / Transfer

Through this method it is possible to transfer PIT Tokens from one wallet to another.

To execute this method it is necessary to inform the following parameters in the request body, for example:

```
{
    "privateKey":
"9798fee0fca3f8d7a5cc36fb8ae822180170ed96164eb8acc8dad30bb60246ee", //
Private Key da Wallet que irá transferir
    "_sender": "0xf9c744832a2EE4D6f2256DC7BBaAb5f38273De76", // Address da
Wallet que irá transferir
    "_to": "0x17cA6A08758F4A078B9c53ca25E6F6736dF34094", // Address da Wallet
que irá receber
    "_value": 1000 // Quantidade de HBT Tokens
}
```

The method receives the following parameters in return:

```
{
    "success": true,
    "code": "100-1000",
    "message": "Sucess!",
    "hash":
"0x06cb0c8189bebdd1c40e339bfeb928287f606e64ea2c0970000361b8fc8ddb66"
}
```

POST / Mint To

Through this method it is possible to mint, that is, create PIT Tokens in a wallet.

To execute this method it is necessary to inform the following parameters in the request body, such as:

```
{
    "privateKey":
"9798fee0fca3f8d7a5cc36fb8ae822180170ed96164eb8acc8dad30bb60246ee", //
Private Key do Minter
    "_sender": "0xf9c744832a2EE4D6f2256DC7BBaAb5f38273De76", // Address do
Minter
    "_to": "0x17cA6A08758F4A078B9c53ca25E6F6736dF34094", // Address da Wallet
que irá receber
    "_amount": 10000 // Quantidade de HBT Tokens
}
```

The method receives the following parameters in return:

```
{
    "success": true,
    "code": "100-1000",
    "message": "Sucess!",
    "hash":
"0x5b8dfa628668f2ef59ad80aa73b39000cc57bc6fe3d60e6ac566b335781cb2d7"
}
```

POST / Burn From

Through this method it is possible to burn, that is, destroy HBT Tokens from a wallet.

To execute this method it is necessary to inform the following parameters in the request body, such as:

```
{
    "privateKey":
"9798fee0fca3f8d7a5cc36fb8ae822180170ed96164eb8acc8dad30bb60246ee", //
Private Key do Burner
    "_sender": "0xf9c744832a2EE4D6f2256DC7BBaAb5f38273De76", // Address do
Burner
    "_from": "0x17cA6A08758F4A078B9c53ca25E6F6736dF34094", // Address da
Wallet que estão os HBT Tokens
    "_amount": 10000 // Quantidade de HBT Tokens
}
```

The method receives the following parameters in return:

```
{
    "success": true,
    "code": "100-1000",
    "message": "Sucess!",
    "hash":
"0xde9b97536f43782fb465e65a8f2b50a603eeb02574c4ebc59f3d1aecaad66b0c"
}
```

POST / Add Role

Through this method it is possible to add a Role (Admin, Minter or Burner) to a wallet.

To execute this method it is necessary to inform the following parameters in the request body, such as:

```
{
    "privateKey":
"9798fee0fca3f8d7a5cc36fb8ae822180170ed96164eb8acc8dad30bb60246ee", //
Private Key do Admin
    "_sender": "0xf9c744832a2EE4D6f2256DC7BBaAb5f38273De76", // Address do
Admin
    "_account": "0x17cA6A08758F4A078B9c53ca25E6F6736dF34094", // Address da
Wallet receberá a autorização
    "_role": "admin" // role adicionado: admin, burner ou minter
}
```

The method receives the following parameters in return:

```
{
    "success": true,
    "code": "100-7015",
    "message": "Wallet address has been added to role admin.",
    "hash":
"0x702614b298ba637eb35dd757e5d87f5f85af3dcffe40c17aaf9dfdcff61c1752"
}
```

POST / Remove Role

Through this method it is possible to remove a Role (Admin, Minter or Burner) from a wallet.

To execute this method it is necessary to inform the following parameters in the request body, such as:

```
{
    "privateKey":
"9798fee0fca3f8d7a5cc36fb8ae822180170ed96164eb8acc8dad30bb60246ee", //
Private Key do Admin
    "_sender": "0xf9c744832a2EE4D6f2256DC7BBaAb5f38273De76", // Address do
Admin
    "_account": "0x17cA6A08758F4A078B9c53ca25E6F6736dF34094", // Address da
Wallet perderá a autorização
    "_role": "admin" // role removido: admin, burner ou minter
}
```

The method receives the following parameters in return:

```
{
    "success": true,
    "code": "100-7014",
    "message": "Wallet address has been removed from role admin.",
    "hash":
"0xfdb7b7adacd47b69d3fd3be5910850181ec7e00c58892bc00fa4633ca4794e51"
}
```

Implementation

Requirements

- NPM: `>=6.13.4`
- GIT: `>=2.21.1`
- Node.js: `>=10.0.0`

Clone repository

To clone this bitbucket repository you need access permission and a configured SSH key.

https://bitbucket.org/janusplatform/hbtrust.domain.token/src/master/

With the access and key you can clone the repository from the terminal:

```
git clone git@bitbucket.org:janusplatform/hbtrust.domain.token.git
```

Install dependencies

In the terminal access the root folder of the repository that was cloned, execute the command to install the dependencies:

```
npm install
```

All dependencies needed to run the API will be installed in your directory.

Create and configure the .env file

In order to run the API, it is necessary to create the .env file at the root of the directory

Clone the .env.example file with the following specifications:

```
MNENOMIC = // Your metamask's recovery words
INFURA_API_KEY = // Your Infura API Key after its registration
NETWORK_ID = // 1-Mainnet 3-Ropsten 4-Rinkeby 42-Kovan 1001-Development
TOKEN = // PDBToken Address
```

After creating and configuring .env, your directory is already configured to run the API.

Run nodemoon on localhost

In the terminal access the root folder of the repository that was cloned, execute the command to execute the API:

```
npm start
```

After executing this command the nodemoon will be initialized and the API will be working at the address http://localhost:3000/.

Access the API and execute a method via swagger

With nodemoon running you can access http: // localhost: 3000 / swagger and execute the methods available to interact with the PIT Token.

On the page you can select a GET / POST / PUT method and click the try button.

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 on execute and the method will be executed.

Run tests on the Test Network

To run API tests on the test network, you must configure the .env file with the following parameters:

```
MNENOMIC = ""
INFURA_API_KEY = "d7af4ca348a2460aadd341988fee82fd"
NETWORK_ID = "4"
TOKEN = "0xfB0E30F97c656b5bE9D92879FbCEC5A35195e346"
```