Documentation

Use this documentation to interact with the crowdfunding smart contract. This contract allows creators and hosts to fund ideas and bring projects to life. This smart contract intentionally contains a public function to reset the pledged amount for any address, in order to dissuade other individuals from deploying this contract to the Ethereum mainnet.

You can use the deployed test token contract in the scenarios to test, although these smart contracts support any Ethereum token. The contracts are generalized to support a variety of uses and provide forward compatibility.

To begin, add a test balance to your address using this function in the test token contract:

```
function importBalance(uint256 amount) public; //Deposits to msg.sender
```

To create a crowdfunding proposal, use the function:

```
function makeProposal(string title,string description,uint256
minGoal,uint256 maxGoal,uint256 hostCut,uint256 duration,address
host,address tokenAddress) public returns(uint256);
```

Relevant parameter(s):

Parameter	Description
string title	Title of Idea
string description	A description of the project
uint256 minGoal	Minimum funding goal (10^-18 tokens)
uint256 maxGoal	Maximum funding goal (10^-18 tokens)
uint256 hostCut	Host percent cut (10^-18 %)
uint256 duration;	Funding duration (Ethereum blocks)
address host	Ethereum address of host
address tokenAddress	Ethereum token contract address used to raise funds

If a new Idea is created, this function returns the index of the new Idea.
To accept a proposal, the host uses this function:

```
function acceptProposal(uint256 index) public returns(bool);
```

Relevant parameter(s):

Parameter	Description
unit256 index	Index of Idea

The function returns a boolean value denoting whether the request has been accepted. After a crowdfunding proposal is accepted, it is now open for pledging. Pledges can only be made during the Idea duration, and cannot be made if they would put the total amount raised over the maximum funding threshold. Duration must be greater than four blocks. For Ideas in which a maximum goal is not desired, one can set maximum goal equal to the theoretical maximum supply (using the information in the statistics section of the elixirtoken.io homepage).

If a host wishes to reject a certain proposal, they can use the function:

```
function rejectProposalAsHost(uint256 index) public;
```

If a creator wishes to cancel a certain proposal, they use the function:

```
function cancelProposalByCreator(uint256 index) public;
```

When a user wishes to pledge tokens for an Idea, they must first approve the token contract to move tokens on their behalf using the ERC-20 standard token function:

```
function approve(address _spender, uint256 _amount) returns (bool
success);
```

Afterward, each user uses this function to pledge tokens:

```
function pledgeTokens(uint256 amount,uint256 index) public returns(bool);
```

Relevant parameter(s):

Parameter	Description
uint256 amount	Amount of tokens to pledge (in 10^-18 tokens)
uint256 index	Index of the Idea

Tokens can only be pledged during the Idea duration, and cannot be made which would put the total amount raised over the maximum funding threshold. The function returns true if the pledged amount is successfully transferred to the contract. If a campaign reaches its maximum goal, funds will automatically be distributed to the host and creator. If a sufficient amount of funding is raised but not reaching the maximum goal, the remainder of the fundraising period must be waited out, and then anyone can trigger the distribution of funds to the host and creator using the function:

```
function distributeSuccessfulCampaignFunds(uint256 index) public
returns(bool);
```

The smart contract will calculate the respective amounts using the amount raised and the host cut parameters, and will distribute them to the host and creator, returning true if this succeeds. After that time, it is the responsibility of the creator to execute and distribute their Idea.

If a campaign fails to receive sufficient funding during a funding round, anyone can notify the contract using the function:

```
function stateFail(uint256 index) public;
```

This will set the Idea status to failed, and users can then reclaim funds individually using the function:

```
function reclaimTokens(uint256 index) public;
```

The following statuses are used in the crowdfunding contract:

Idea Status	Usage
1	Proposed
2	Underway
3	Sufficient
4	Failed
5	Proposal Canceled by Creator
6	Proposal Canceled by Host
7	Distributed

A crowdfunding Idea exists in smart contract storage as follows:

```
struct Idea {
      string title;
      string description;
      uint256 minGoal;
      uint256 maxGoal;
      uint256 hostCut;
      uint256 duration;
      uint256 startTime;
      uint256 status;
      uint256 amountRaisedSoFar;
      address host;
      address tokenAddress;
      address creator;
      mapping(address => uint256) amountPledged;
      mapping(address=>bool) reclaimed;
}
```