Elepig Crowdsale

Deploying:

1. Deploy the crowdsale contract. To deploy, the following parameters are required:

**uint256 \_startTime,** // start of pre-ico

**uint256 \_endTime,** // end of ico

**uint256 \_rate,** // original rate of ETH to EPG

**address \_wallet,** // the wallet address where ETH gets sent

**uint256 \_goal,** // softcap in ETH

**uint256 \_cap,** // Hardcap in ETH

**address \_communityAddress,** // address where tokens that aren’t sold will get minted

**address \_token,** // address of Elepig Token

**address \_signer** // address of signer. (used for KYC)

1. Once the crowdsale contract is deployed, the owner address of the Elepig Token contract, must call the transferOwnership function within the Elepig Token Contract (can be done in remix) The **newOwner** Parameter must be the address of the newly deployed crowdsale contract. This is so the crowdsale contract can call the mint function within the Token contract. At the end of sale, we can transfer ownership back if needed. Using the transferTokenOwnership in the crowdsale contract.

function transferOwnership(address newOwner) public onlyOwner {

require(newOwner != address(0));

emit OwnershipTransferred(owner, newOwner);

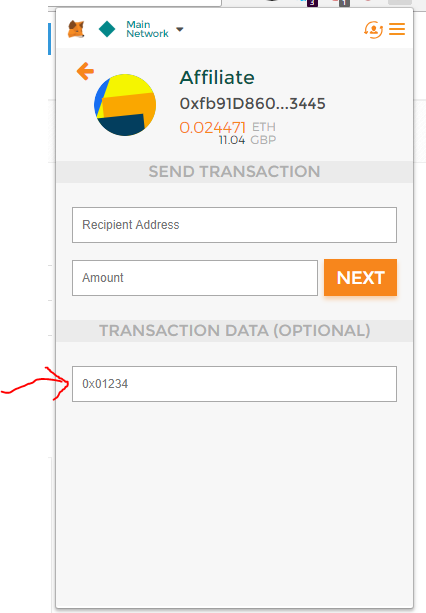
owner = newOwner;

}

1. Any money sent to the crowdsale contract before the start date will fail.

How people will buy EPG:

1. Once KYC’d. There will be a web3 function that will get called. This function has an input of the wallet address. Their address gets signed by the **signer** address (set during deployment) and they get given an encoded message (which can be displayed on this website.) This will only work for their address. They will input this encoded data as shown below:



1. Depending on the crowdsale stage, the total EPG to be sent is calculated and the EPG’s are then minted in their wallet.
2. Any value under 0.15 ETH will fail.
3. If we later need to remove access from someone, the owner of the crowdsale contract will call:

// adding an address to the blacklist, addresses on this list cannot send ETH to the contract

function addBlacklistAddress (address \_address) public onlyOwner {

blacklist[\_address] = true;

}

// removing an address from the blacklist

function removeBlacklistAddress (address \_address) public onlyOwner {

blacklist[\_address] = false;

}

Sale Rounds:

1. Each rate for the rounds are hardcoded on the contract as:

if (stage == CrowdsaleStage.PreICO) {

setCurrentRate(2380); // 1 EPG = 0.00042 ETH

} else if (stage == CrowdsaleStage.ICO1) {

setCurrentRate(2040); // 1 EPG = 0.00049 ETH

} else if (stage == CrowdsaleStage.ICO2) {

setCurrentRate(1785); // 1 EPG = 0.00056 ETH

} else if (stage == CrowdsaleStage.ICO3) {

setCurrentRate(1587); // 1 EPG = 0.00063 ETH

} else if (stage == CrowdsaleStage.ICO4) {

setCurrentRate(1503); // 1 EPG = 0.000665 ETH

}

Where the ratio is EPG to ETH

1. The Hard caps for each round are as follows:

// 150MM of Elepig are already minted.

uint256 public totalTokensForSale = 150000000000000000000000000; // 150 EPGs will be sold in Crowdsale (50% of total tokens for community)

uint256 public totalTokensForSaleDuringPreICO = 30000000000000000000000000; // 30MM out of 150MM EPGs will be sold during Bonus Round 4

uint256 public totalTokensForSaleDuringICO1 = 37500000000000000000000000; // 37.5MM out of 150MM EPGs will be sold during Bonus Round 4

uint256 public totalTokensForSaleDuringICO2 = 37500000000000000000000000; // 37.5MM out of 150MM EPGs will be sold during Bonus Round 4

uint256 public totalTokensForSaleDuringICO3 = 30000000000000000000000000; // 30MM out of 150MM EPGs will be sold during Bonus Round 4

uint256 public totalTokensForSaleDuringICO4 = 15000000000000000000000000; // 15MM out of 150MM EPGs will be sold during Bonus Round 4

If someone tries to send ETH to the contract which will put it over the cap for that round, then the transaction will revert. We then need to decide if we go onto the next round. The safest way to transition into the next round is manually. The owner of the crowdsale contract will call:

function setCrowdsaleStage(uint value) public onlyOwner

The input for value will be : 0, 1, 2, 3 or 4. (for each round)

Refunds and SoftCap:

1. All funds below the soft cap are forwarded into an ETH vault that the contract sets itself. No one has the private key for this vault, not even us. One we have reached the softcap, the owner is able to call a releaseVault function which sends the funds from the vault directly to the wallet address we set during the deployment.
2. Once we finalise the crowdsale (requires we have reached our hardcap or we have reached the end of the sale) The ETH within the vault will either be sent to us or refunded to the users if we haven’t reached the softcap. Any ETH below the softcap will therefore be unavailable until the end of the sale.
3. The Contract can be finalised by the owner by calling:

function finish() public onlyOwner