Fomo Coin Documentation

Fomo, short for *Fear Of Missing Out*, is an exponentially increasing coin whose value is determined in USD.

Throughout this paper we will often be referring to "tokens". Tokens represent the smallest possible unit of Fomo. One Fomo coin is equivalent to exactly 100 tokens and one token is equivalent to 0.01 Fomo coins. By default, Fomo is represented in coins, hence any values appended with the `FOMO` symbol are given in coins.

- 1 FOMO = 100 Tokens
- 0.01 FOMO = 1 Token

Initial Coin Offering

Fomo provides a total token supply of **100 000 000** tokens and a starting price of **$0.01**. While the price of the token is determined in USD, tokens are purchased in ETH, respective to their USD price. The ETH price per token is regularly updated by the Fiat Contract, actively maintained by Hunter Long.

> The Fomo contract is equipped with emergency functions which can update the ETH price even if the Fiat contract ceases to be maintained.

Value

The Fomo value can be analogized to a wave. As soon as an expected amount of tokens (10 0000 tokens or 1 000 coins) has been purchased, whether it is from the minter or from sell offers (see Trading), the wave "breaks" and the USD value exponentially increases by *10%*.

Due to overflow reasons, the price is incremented recursively and floors at the 4th USD decimal. The tokens value growth can be modeled by the following equation:

$$new\ price = old\ price + \frac{\lfloor old\ price \cdot 1000 \rfloor}{10000}$$

Where the initial price per token is `$0.01`. Hence, the price will increment as it follows: `$0.01 + $0.011 + $0.0121 + $0.0133 + ...`, rather than relying on an exponential equation as such as `0.01 * 1.1 ^ w` where w is the wave.

Trading

Fomo may not be directly traded from wallet to wallet. Instead, Fomo relies on "sell offers" that are initiated by token holders. Once a sell offer has been proposed, it will remain active until it has been accepted and there is no way to cancel it. Only after an offer has been accepted, the seller will receive his ETH.

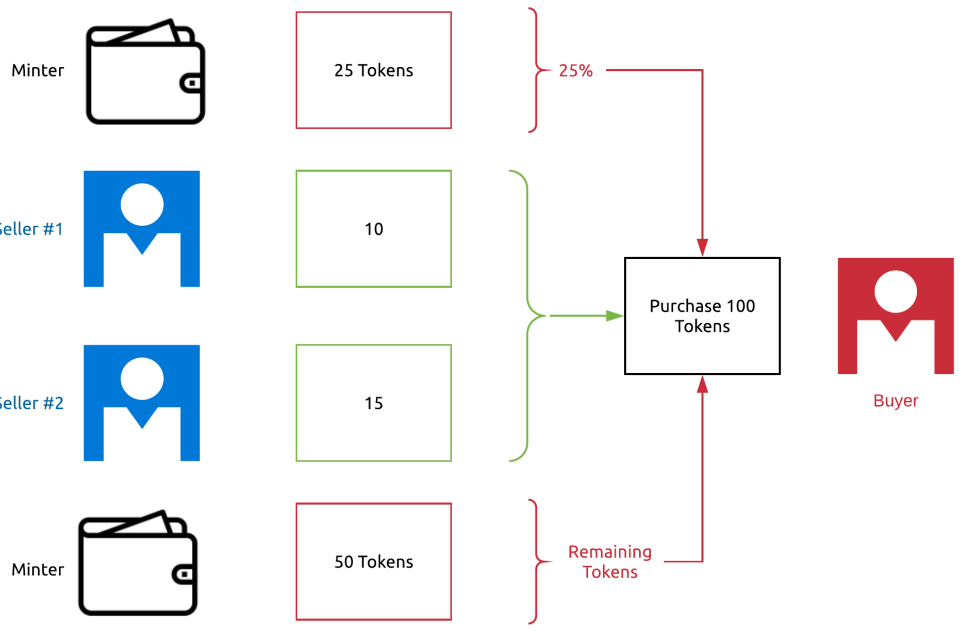
Sell offers are "accepted" whenever someone purchases tokens. A purchase is not limited to a single sell offer, but can span across multiple offers. Contrarily, if insufficient sell offers are available, the remaining tokens are purchased directly from the minter wallet. If the minter has been depleted, the buyer must wait until enough tokens are offered.

A buyer does not get to choose from which offers to purchase tokens, instead, the contract automatically purchases tokens from the oldest standing offer(s). Each offer has a "priority value" which indicates how many offers have a higher priority. An offer priority of 0 means that the offer is guaranteed to be accepted with the next purchase. Offers may also only be partially depleted if the number of purchased tokens does not exceed the offers value. Should a token holder decide to sell more tokens after already initiating a sell offer, the tokens will simply be added to the existing sell offer, meaning the offer priority will remain unaffected.

An offers value is determined by the latest USD price, meaning that a sell offer will increase in value if it has not been accepted by the time the token price has increased. Further, any offered tokens are technically still part of the sellers wallet until the offer has been accepted. This means that sellers still receive dividends (see Dividents) for any offered tokens.

Finally, with every purchase, 25% of purchased tokens are minted. This way, dividends are provided with every transaction until the minter wallet has been fully depleted (See Dividents).

See Figure 1

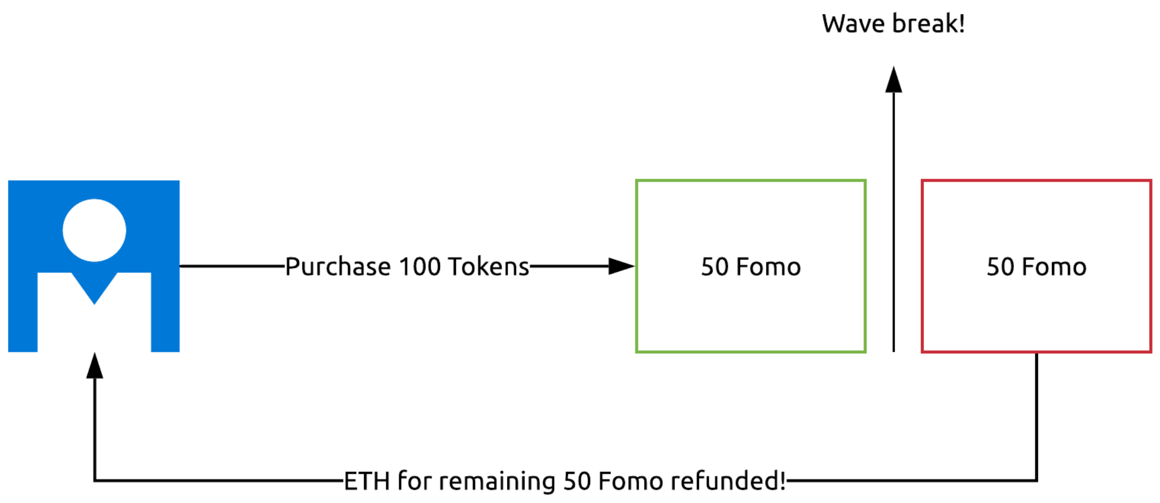


Exceptions

Should a purchase be responsible for the increase in price (aka. a "wave break"), then the buyer will only receive, and be charged, for the tokens of the previous "wave". This exception is rare to occur for any user, but will nearly always occur at the end of a wave unless the transaction features the exact required amount of tokens to increment the price.

On the official FOMO exchange portal, the buyer will be notified of this event. Developers that wish to implement FOMO should listen for the `Increase` event after any purchase to handle this exception!

See Figure 2



Dividends

An additional touch that makes Fomo an attractive token is the presence of dividends. A part of every minted ETH income (5%) is shared to all Fomo holders. Dividends are split in percentiles, meaning whoever held most Fomo at the time of transaction, will get the largest percentile of the dividend share.

Due to gas limits, it is impossible to keep track of all Fomo holders. Instead, holders are required to fetch/retrieve their dividends manually. Dividends are fetched according to an accounts balance history, meaning a change in balance will not affect the dividends related to the prior balance.

> Fetching dividends requires the holder to pay a gas fee, which further increases with every transaction that took place. **In some circumstances (depending on how much FOMO were trasfered and what the current gas price is, etc.), the retrieved dividends may be overshadowed by the gas price!**

Contract Reference

This section provides a reference that documents the publicly exposed views and transactions of the Fomo Coin contract

Sections

- Contract Info
- Transactions
- Dividends
- Offers

Quick Views Reference

View	Parameters	Return Type
name		string
symbol		string
decimals		uint256
totalSupply		uint256
minter		address
usd		uint256
wave		uint256
mintedTokens		uint256
onTrade		uint256
oldestOfferIndex		uint256
fiat		address
minter		address
balance	address _owner	uint256
offers	uint256 _at	address
offer	address _of	Offer
getOfferPriority	address _for	uint256 priority
tokensToWei	uint256 _value	uint256 price
expectedDividends	address _for	uint256 expected

Quick Transactions Reference

Transaction	Parameters
sell	uint256 _value

Transaction	Parameters
purchase	uint256 _value
getDividends	

Contract Views

View	Parameters	Return Type	Returns	Notes
name		string	Token name	
symbol		string	Token identifier	
decimals		uint256	The tokens decimal units	
totalSupply		uint256	Total Supply of tokens	
oldestOfferIndex		uint256	Index of the oldest active seller	
wave		uint256	Current wave	
minter		address	Minter address	
usd		uint256	The current USD value of a token	Uses 4 decimals (100 = 0.01$)
mintedTokens		uint256	The amount of minted tokens	
fiat		address	Fiat contract address	
onTrade		uint256	Number of tokens being offered	Does not take minter wallet into account
balance	address _owner	uint256	Returns accounts Fomo balance in tokens	

Transactions

This section documents all contract functions that are required to process and complete transactions.

Contents

tokensToWei

Calculates the exact price in wei for `_value` tokens. This function should be used to determine the required wei input when purchasing tokens.

Parameters

Type	Name	Description
uint256	_value	Amount of tokens

Returns

`uint256 price`

Example

See tokensToWei example

purchase

Purchase `_value` tokens. This function takes the **exact required** amount of wei as input, which can be calculated by tokensToWei.

Parameters

Type	Name	Description
uint256	_value	Amount of tokens to be purchased

Example

```
/* Purchase 100 Tokens */
Fomo.tokensToWei.call(100, function(err, wei) {
  Fomo.purchase(100, {value: wei.toString()});
});
```

sell

Offer `_value` tokens for sale. Re-calling this function will add tokens to the previous offer. After a offer has been made, it waits to be purchased. As soon as a purchase has accepted the offer, the expected ETH will be transfered to the seller.

> See the Offers section for further understanding of offers

Parameters

Type	Name	Description
uint256	_value	Amount of tokens to be sold

Example

```
/* Purchase 100 tokens, then sell 50 and finally 25 more */
Fomo.tokensToWei.call(100, function(err, wei) {
  Fomo.purchase(100, {value: wei.toString()});
});

Fomo.sell(50); // Offering 50 tokens
Fomo.sell(25); // Offering 75 tokens (50 + 25)
```

Dividends

With every mint of tokens, a certain percentage of ETH income is reserved for all momentary holders of Fomo.

The distribution of dividends is calculated by comparing the holders balance at the time of transaction, with the total amount of minted Fomo during the time of transaction. This way, the contract determines the percentage of Fomo the account has held at the time of transaction, and the percentile of the dividends that should be given to that account.

Contents

expectedDividends

Calculates the expected ETH to be received by `_for` if `_for` was to call getDividends.

Parameters

Type	Name	Description
address	_for	Address to be checked for expected dividends

Returns

`uint256 expected`

Example

```
/* Output accounts[1]'s dividends */
Fomo.expectedDividends.call(web3.eth.accounts[1], function(err, expected) {
  console.log("Expected dividends for account 1: " + expected.toString());
});
```

getDividends

Fetches the dividends for the transaction caller.

Exceptions

Reverts if no dividends are available

Example

```
/* Receive dividends */
Fomo.getDividends();
```

Offers

Offers are generated every time an account decides to sell tokens. Addresses of accounts with active offers are appended to an offers array which is read from the oldest offer to the newest offer. With the given address, the contract can then access an accounts offer instance which lists the total amount of tokens that are being offered by the address.

Should a account attempt to sell tokens while an existing offer is still active, then the active offer will simply be updated.

> Upon purchase, older offers are always prioritized!

Contents

Offer Structure

Every account stores a instance of the offer structure, which indicates whether an offer is active, the assigned index in the and number of Fomo that is being offered by the account offers array.

Index	Type	Property	Description
0	bool	active	Defines whether an offer is active
1	uint256	index	Index of the assigned offer entry in offers
2	uint256	value	Fomo being offered

If a offer is no longer active, it is not deleted. Instead the `active` property is set to `false`, which will tell the contract to ignore the offer when iterating through all offers. As a result, the `value` and `index` properties may be set, even if the offer is inactive.

Example

See offers example

offers

A list of addresses that offer tokens. After a offer has been accepted and depleted, its address is not deleted from the array. Instead, oldestOfferIndex is incremented, which indicates on which index the oldest active offer is located. All offers following oldestOfferIndex are active.

Parameters

Type	Name	Description
uint256	_at	Index of offer to be returned

Returns

`address`

Example

```
/* Fetch 1st offer in offers */
Fomo.oldestOfferIndex.call(function(err, index) {
  Fomo.offers.call(index.toNumber(), function(err, address) {
    Fomo.offer.call(address.toString(), function(err, offer) {
      console.log("Oldest active offer offers " + offer[1] + " tokens!");
    });
  });
});
```

oldestOfferIndex

Return Index of the oldest offer. All offers following the oldest offer are active.

> oldestOfferIndex is incremented with every offer completion.

Returns

`uint256 oldestOfferIndex`

Example

See offers example

getOfferPriority

Returns the priority of an accounts offer (ie. the number of active offers that are older)

Parameters

Type	Name	Description
uint256	_for	Address holding an active offer

Returns

`uint256 priority`

Exceptions

Reverts if offer at address is inactive

Example

```
/* Fetch 1st offer in offers */
Fomo.getOfferPriority.call(web3.eth.accounts[1], function(err, priority) {
  console.log("Offer priority: " + priority.toString());
});
```