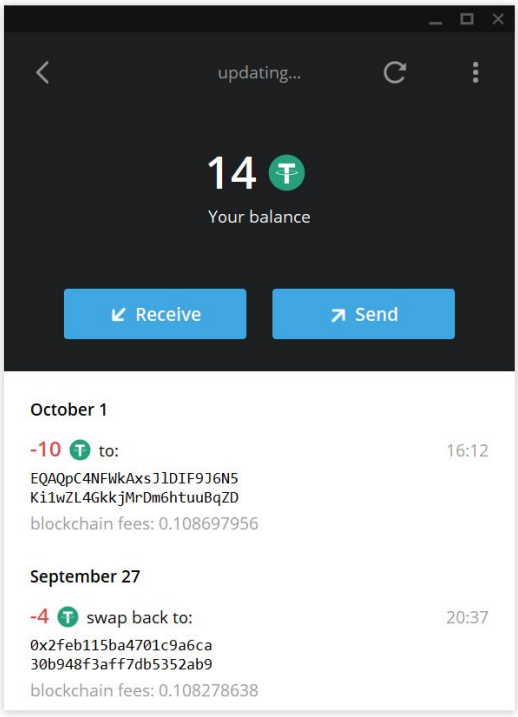
Abstract

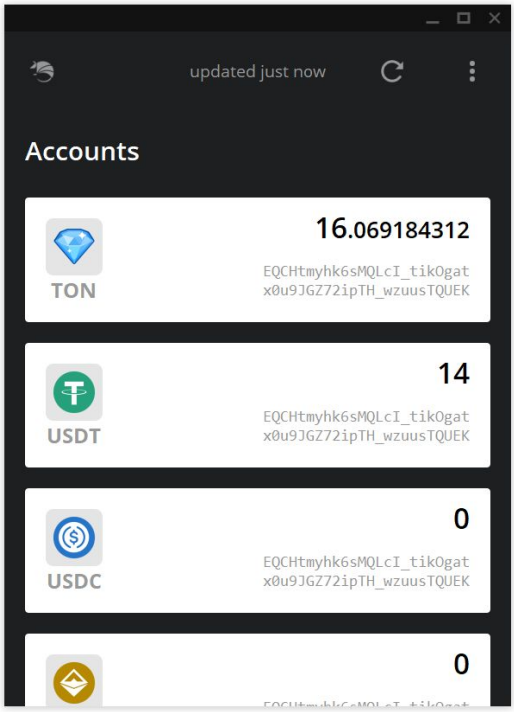
The document describes the basics of a two-way gate, which allows you to transfer data from Ethereum to FreeTON and back. An example of gate use is the transfer of tokens between networks. The functionality of a gate is supported by relays, which are rewarded for their participation in the form of commissions from the performed actions. Gate is based on a set of on-chain actions on the FreeTON platform, as commissions in this network are much lower than in Ethereum. The functionality of the gate does not impose any restrictions on data transferred between networks. Users can add new types of data by specifying the details of their transmission. Thanks to this broad functionality, a huge variety of applications can be integrated into a gate.

Introduction

This document describes the main details of bilateral gate implementation between Ethereum and FreeTON. The implementation of the described scheme will allow to extend possibilities, improve security level and introduce decentralized management for the existing gate - gate.broxus.com. The gate is already integrated with Crystalwallet (https://l1.broxus.com/freeton/wallet).

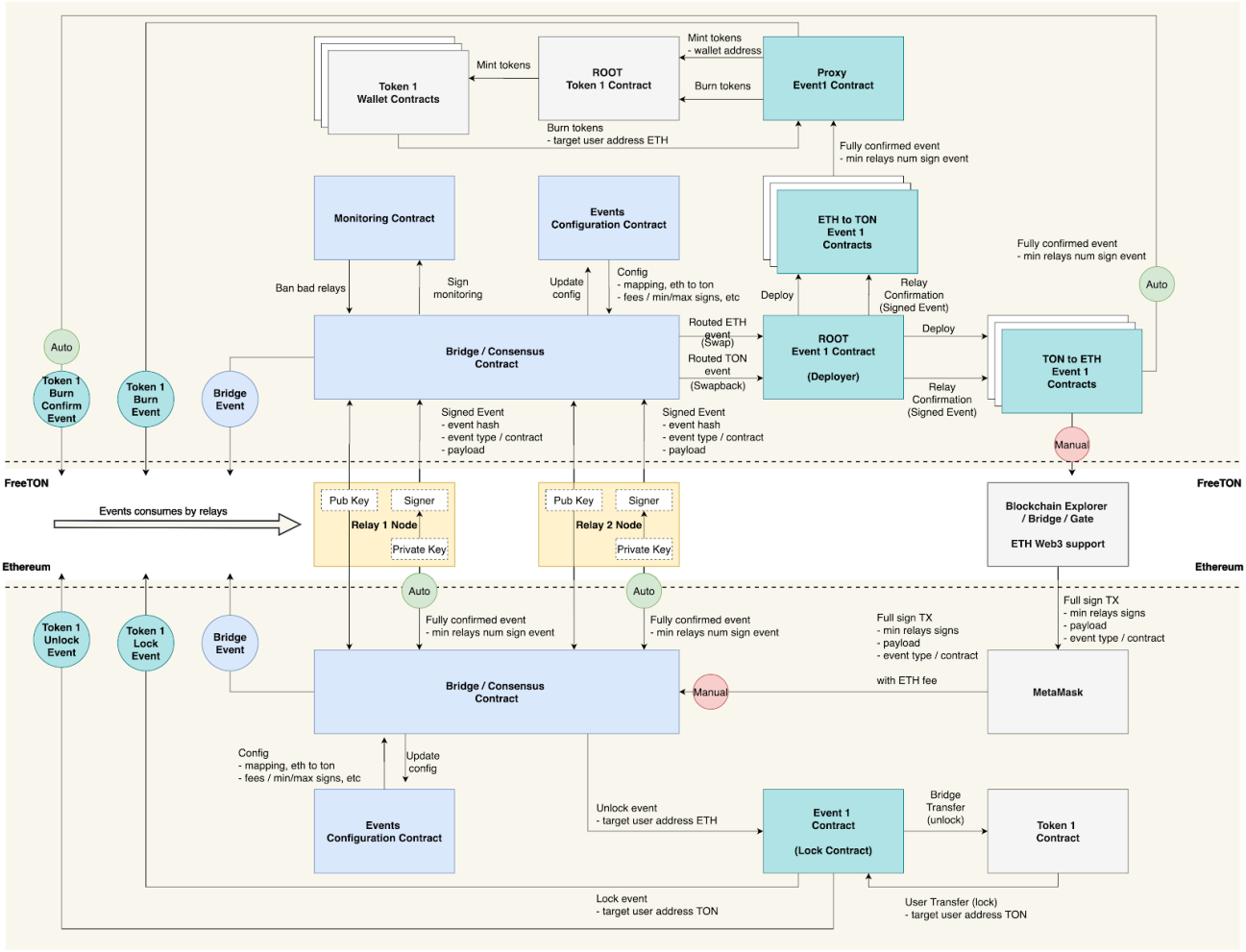
Sources for current gate smart contracts and wallet are available on Broxus Github (https://github.com/broxus/eth-gate-contracts, https://github.com/broxus/wallet-desktop)





Gate schema

The scheme below illustrates how a gate works in case of transferring Ethereum tokens to FreeTON and back.



Gate Components

This paragraph describes the main components and terms used in the text to describe how a gate works.

Relays (relays)

A set of users that provide gate functionality. The task of relays is to monitor and respond to actions performed in Ethereum and FreeTON networks. Relays are uniquely identified by their public keys. The list of public keys of relays is available through a smart contract in both networks.

Bridge contract

Main smart contract available on both networks. Responsible for achieving consensus between relays. Contains lists of public keys of relays.

Monitoring contract

The contract in the TON network accepts calls only from bridge contracts. Responsible for storing a certain number of recent events that were sent by other relays, so relays can vote on each event individually. It can be

implemented inside the bridge of the contract. The contract stores only a certain number of recent events, in order to avoid an increase in fees due to the cost of data storage.

Configuration contract

Smart contract available on the FreeTON network. Serviced by relays. Contains processing details for each supported Ethereum event. For example - required number of relay confirmations.

Root event contract

Smart contract available on the FreeTON network. It is the root contract of the event. For one type of event, one such contract must be created. An example of an event is, for example, the transfer (freezing) of tokens in the Ethereum network for their subsequent release in the Free TON network. In addition, the same contract can serve reverse events. At the implementation level, it is a separate smart contract. For each new event detected by relays, the contract creates a child smart contract (Event contract ETH-TON). The relays send their event confirmations to the Root event contract, which proxies their voice to a specific Event contract ETH-TON, if one has already been created, or creates a new one.

Event contract ETH-TON

Smart contract available on the FreeTON network. Used to perform a key action, if a sufficient number of relays have confirmed the event in Ethereum. Each event corresponds to a unique Event contract. The contract can only be sealed with a Root event contract. Contains the relays of the original event confirmation. When the minimum number of confirmations is reached, it triggers a bulb for subsequent token rallying or other action.

Event Proxy Contract

Smart contract available on the FreeTON network. It’s a contract adapter for token contracts, can be integrated into a child event contract or root token contract.

Event contract TON-ETH

Smart contract available on the FreeTON network. Used to collect signatures of relays that confirm the validity of the action transmitted from FreeTON to Ethereum. Example of action - move tokens from FreeTON to Ethereum. If a sufficient number of signatures of relays is received, the user can perform an action in Ethereum with the provision of these signatures. The contract is created in the FreeTON network because of the lower cost of transactions.

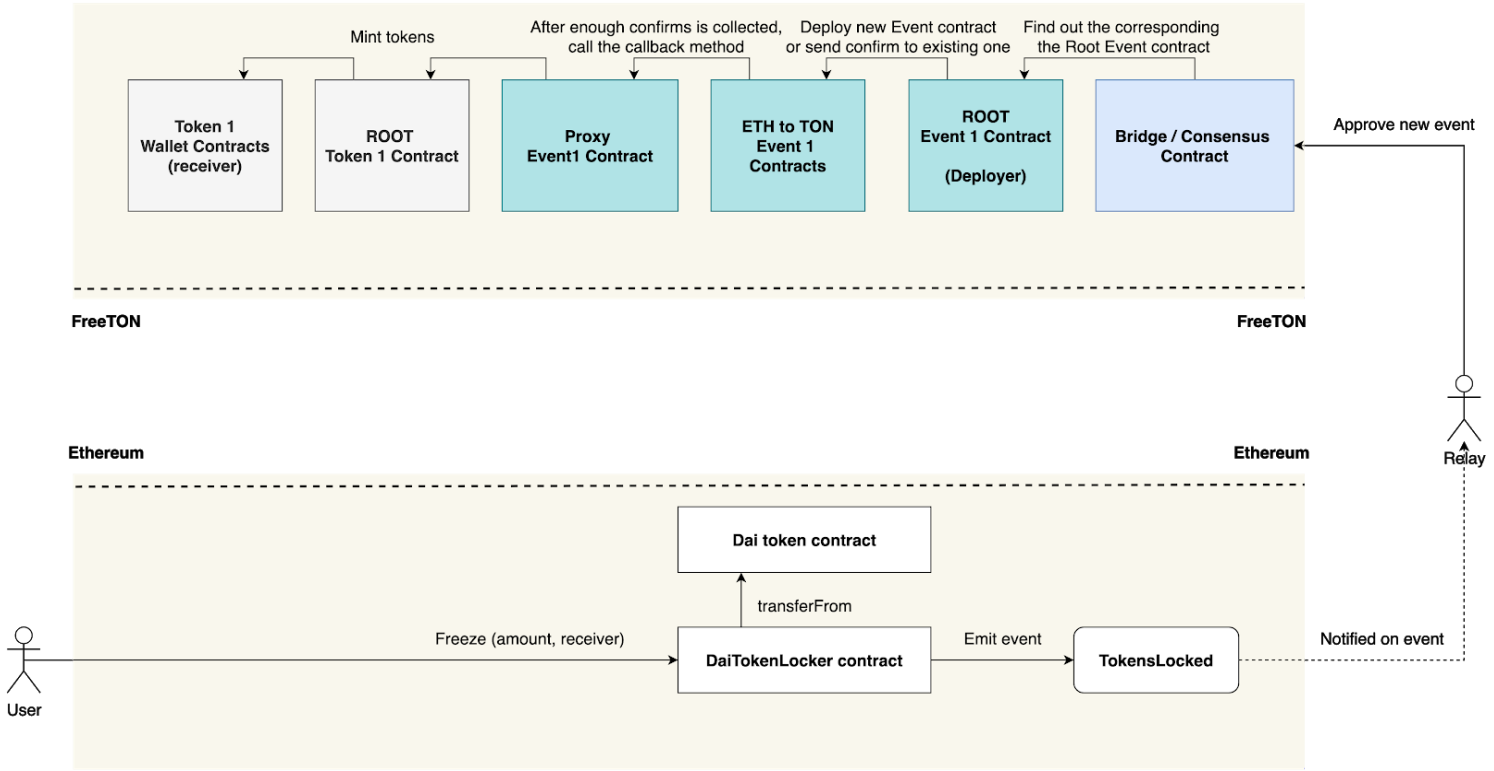
Key action ETH-TON

An action taken in the FreeTON network when a sufficient number of Ethereum event relay confirmations are received. Implemented as a smart contract method which is available to call by the Event proxy contract.

Key action TON-ETH

An action taken in the Ethereum network when a sufficient number of FreeTON event relay confirmations are received. Event body signatures are used as confirmations.

Using gate to respond to Ethereum event in FreeTON



This paragraph gives an example of how the described gate can serve for transferring Ethereum events in the FreeTON network. As an example we use the transfer of Dai tokens from Ethereum to FreeTON.

To transfer Dai to FreeTON, the user freezes Dai tokens on a separate contract called **DaiTokenLocker** in the Ethereum network. When freezing tokens, the user specifies the number of tokens to be transferred and the address to obtain tokens in the FreeTON. When freezing tokens, the contract creates event **TokensLocked (uint amount, bytes receiver)**.

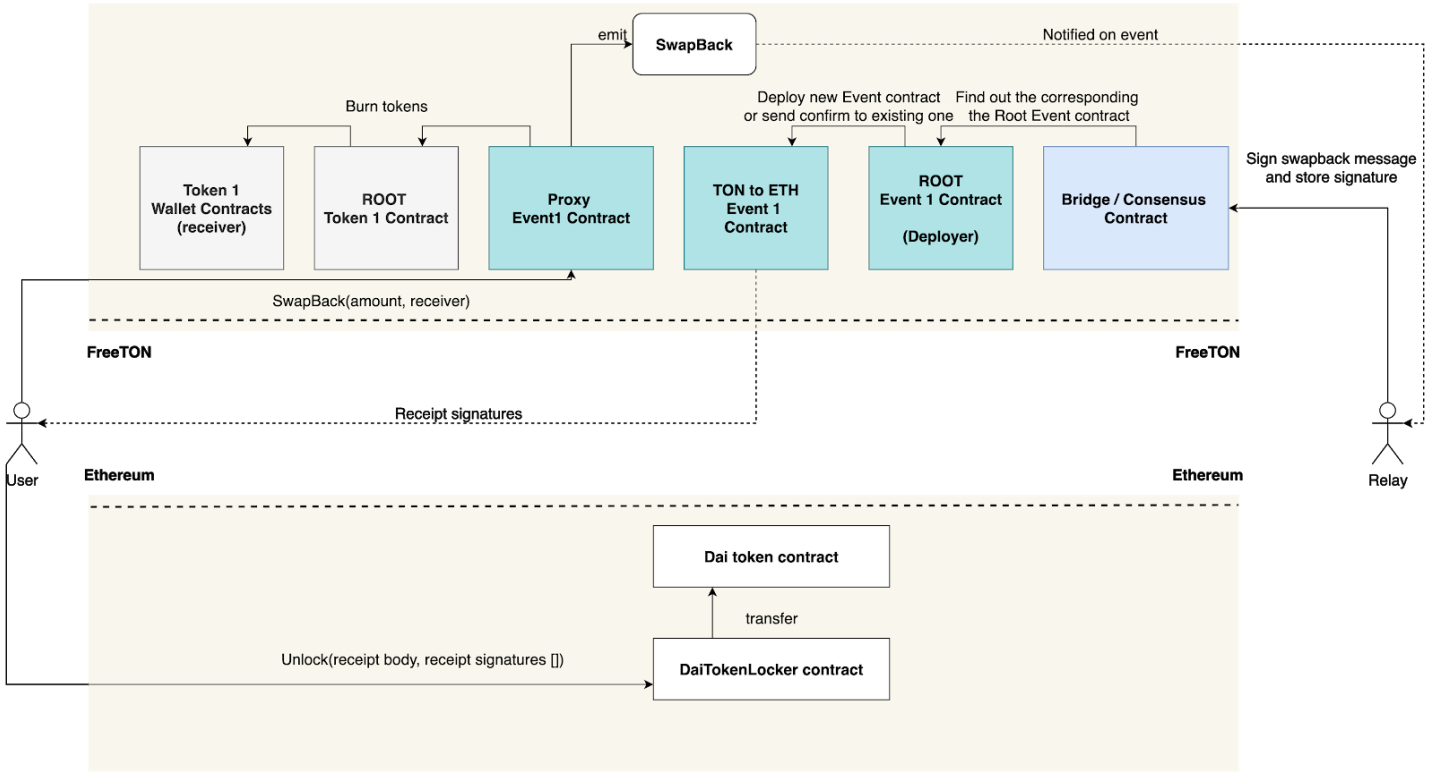
Adding event support

The basic action to support Ethereum events in the FreeTON network is to configure the event in the contract in the FreeTON network. When configuring an event, the following are specified

- Event Ethereum details (address DaiTokenLocker, TokensLocked Event)
- required number of relay confirmations
- Event Proxy Contract Address

When adding event support, the user deposits the amount to TON Crystal to cover the cost of sending transactions to the FreeTON network. The decision to add an event to the list of supported events is made by the relays, in a voting format.

Using the gate to respond to FreeTON events on the Ethereum network



This paragraph describes how a gate can serve to display FreeTON events in the Ethereum network.

The key difference from the scheme described above is that the fees in Ethereum are significantly higher than in the FreeTON network, which in most cases makes it impossible to send transactions by relays. Instead, relays send the signature of the action to Event contract TON-ETH as an endorsement of the action. When a sufficient number of approvals are received, the user sends the body of the action and the set of signatures to the Ethereum contract himself.

Using FreeTON - ETH gate in automatic mode

In the scheme described above, the gate works in manual mode, as it is required to send a transaction with the target action. If necessary, the gate in the direction FreeTON-ETH can work without user participation. In this case, the transaction with the target action body and signatures of the action by relays is sent by any relay. The difference is that ETH has to be deposited to Ethereum to cover the costs of the relays.

Change of gate parameters

All gate contracts are implemented considering the possibility of changing configurations. Changes are accepted or rejected by relays in a decentralized way, in a voting format. Changing the list of relays is one example of a gate configuration.

Changing the list of relays

Any user can send a request to the Bridge contract on the FreeTON network for relay rights. The decision to grant relay privileges is made by the current set of relays in voting format.

Staking

In order for an address to become a relay, it must have a stake in the gate system. Any relay can accuse any other relay of not following the rules and demand to take away its stake. This decision is made in a voting format, and if the decision is confirmed, the malicious relay is banned with confiscation of the stake. The number of votes required for the ban of a relay is part of the configuration and can be changed by a separate voting.

Specification of events

This paragraph gives more details about the message design used for gate operation.

Ethereum-FreeTON

After an event has occurred in the FreeTON network and the necessary number of signatures from relays for this event has been received, the user is able to form and send a transaction for its processing in the Ethereum network.

The user enters a special web-page of the event in one of the FreeTON Explorer block explorers using a browser with built-in web3-provider.

Through web3 a user's purse is accessed, where the user is offered to send the prepared transaction to the Ethereum network.

This transaction is a call to the processEvent method of the Bridge smart contract with transfer to it as parameters:

id	event identifier
tonRootEventContractAddress	RootEventContract address in the FreeTON network
payload	event data
created	date and time of the event in the TON network

relayPublicKeys	keys of the signed relays
relaySigns	signatures of relays that confirm the authenticity of the event (in the same order as relayPublicKeys).

The structure of payload depends on the specific type of event. For example, in case of token transfer, payload may contain:

recipient	ETH address of the recipient
lockAmount	amount frozen in the FreeTON network
unlockAmount	amount for receiving in the network Ethereum

Users can send this transaction for a limited period of time.

Such transaction will lead to the next call chain:
1) Bridge.processEvent(id, tonTokenRootContractAddress, payload, created, relayPublicKeys, relaySigns)

1.1) Check: created < (now - time limit)

1.2) Calculation of relay public key list - trustedRelayPublicKeys

Note: Since the list of active relays may change, it is the active relays at the time of creation that are taken into account during the check.

1.3) Call EventsConfigurationContract.getEventConfig with the following parameters

tonRootEventContractAddress	RootEventContract address in the FreeTON network
created	date and time of the event in the TON network

EventsConfigurationContract.getEventConfig returns as a response:

minRequiredSigns	minimum number of required event signatures (value at the moment of creation)
eventContract	address of the contract EventContract responsible for event processing
config	[optional] additional parameters (value at the moment of creation)

1.4) The contract verifies the authenticity of relay signatures and whether there are enough signatures based on relayPublicKeys, relaySigns and trustedRelayPublicKeys:

minRequiredSigns <= number of matches (relaySigns[i] == signature(relayPublicKeys[i], message)) And (trustedRelayPublicKeys contains relayPublicKeys[i])

1.5) If the check is successful, the EventContract(eventContract) is called. process(id, payload, created, sonfig) is passed as parameters.

id	event identifier
payload	event data
created	date and time of the event in the TON network
config	[optional] additional parameters (value at the moment of creation)

the result of this method execution is saved in the result variable

2) In EventContract.process

2.1) There is a check that the event with id data has not been processed yet

2.2) The event is directly processed

For example, for a token transfer event, this method will perform the following actions:
i. Checks for id

ii. Decodes from payload
recipient - ETH address of the recipient
unlockAmount - amount for receiving in the network Ethereum

iii. Performs transfer of tokens to the recipient by calling ERC20(tokenAddress). transfer(recipient, unlockAmount).
tokenAddress - address of the token controlled by this ERC20

iv. Sends event TokenUnlock message with fields

id	event identifier
lockAmount	amount frozen in the FreeTON network
unlockAmount	amount for receiving in the network Ethereum
recipient	ETH address of the recipient

tokenAddress	ERC20 token address
locked	date and time of the event in the TON network (created)
unlocked	date and time of event processing in Ethereum network (processed)

2.3) Returns random data set as a result

3) Bridge sends EventProcessed event with fields

id	event identifier
payload	event data
config	[optional] additional parameters (value at the moment of creation)
result	result of EventContract.process execution
tonRootEventContractAddress	RootEventContract address in the FreeTON network
eventContractAddress	EventContract address (this contract)
created	date and time of the event in the TON network
processing	date and time of event processing in the Ethereum network

FreeTON-Ethereum

Relays monitor all the events occurring in contracts of the EventContract type, but only those that meet the following rules are transmitted to the FreeTON network:

1) All required fields are present:

id	event identifier
payload	event data
created	date and time of the event
crossChain: boolean	service field with value true

This is how a user can work with EventContract in case it is a token transfer contract:

1) The user executes the ERC20(tokenAddress) call (eventContractAddress, lockAmount).

tokenAddress	ERC20 token address
eventContractAddress	EventContract address corresponding to Token ROOT contract in the FreeTON network
lockAmount	amount for freezing in the Ethereum network

2) The user sends a transaction in the Ethereum network calling the EventContract method (eventContractAddress).lock(lockAmount, recipient) at the EventContract corresponding to the translation token, where:

lockAmount - the amount for freezing in the Ethereum network
recipient - address in the TON network, to which the transfer is made

In this method, it is called
2.1) if a commission is charged to ETH, the sufficiency of msg.value for its payment is checked
2.2) lockAmount freezing in favor of EventContract

ERC20(token_address).transferFrom(msg.sender, address(this), lockAmount)

2.3) calculation of the amount to be received in the FreeTON network (unlockAmount).

2.4) sending the TokenLock event

id	event identifier
payload	event data
created	date and time of the event
crossChain	a service field that makes it clear to the reality of the need to broadcast the event to the FreeTon network

The payload field may have the following data:

lockAmount	amount frozen in the Ethereum network
unlockAmount	amount for receiving in the network FreeTON
recipient	FreeTON address of the recipient