Recurring Payments contest submission

github: github.com/Arseny271/FreeTON-RecurringPaymentsContest
email: arsendmitr@gmail.com **telegram**: @successful17k

This document describes a smart contract system for implementing a recurring payment mechanism. The system uses the implementation of tokens from broxus, but can be adapted for other implementations.

1. Basic smart contracts

1.1 MultitokenWallet.

This contract is proposed to be used as the main wallet for all tokens available to the user. The contract stores information about all wallets with TIP-3 tokens tied to it, as well as about all subscriptions used by the user.

1.1.1 Subscription management methods:

approve(address spender, SpendTerms terms) - Allows the contract with the specified address to spend funds according to the conditions. The conditions can limit the frequency of spending funds, their amount, set a specific recipient of funds, set a specific currency

disapprove(address spender) - Revokes a previously issued permission

spend() - Spends tokens if all conditions are met

1.1.2 Methods for managing token wallets:

walletDeployTonTokenWallet(address root_token_wallet) - Deploys a new TIP-3 wallet

walletAddTonTokenWallet(address root_token_wallet, bool need_receive_callback, bool allow_non_notifiable, bool need_bounce_callback) - Binds a previously deployed wallet and configures it depending on the function arguments.

walletSendValue and **walletSendTokens** functions are used to send TON Crystal or TIP-3 tokens by user request.

1.2 OffchainTimer.

This contract is responsible for calling another contract at a specific time. When the contract wants to be called at the right time, it sends a message to the address **:4242424242424242** with the time at which it should be called and also with the size of the reward for the caller. A special off-chain program should track incoming messages and call timer contracts when required, receiving a reward for this. Thus, there is an additional opportunity to earn money in the TON network and a universal cheap way of periodically calling the contract.

1.2.1 Methods:

setTimerParams(address notify_address, uint128 new_reward, uint64 new_period, uint64 new_start, bool need_repeat) - Sets the parameters of the timer. *new_notify_address* - timer callback address, *new_reward* - fee for the caller of the contract, *new_period* - frequency of calls, *new_start* - the origin, used for alignment, *need_repeat* - automatic repeat requests.

startTimer(address reward_address) - Starts the timer according to the previously specified settings, reserves funds for rewarding the caller, sends a request to the **:4242424242424242** address.

timerWakeUp(address reward_address) - This method can be called by an unsigned external message if it is time to call. Sends the reward to the *reward_address*. Calls a callback for a *notify_address* contract.

1.3 AccessController.

Is a simple example of a subscription service. This is a simple example of a subscription service. Mediator between the user and the service provider. When a user wants to subscribe to a service, he must independently or with the help of debot deploy this contract and provide him with access to the funds of his wallet using the **approve()** method.

1.3.1 Work algorithm.

1) When creating a contract, it checks for the presence of an associated TIP-3 wallet (if necessary), creates a timer contract and waits for access to the user's funds.

2) Once the previous conditions are met. The contract creates a request for spending funds and starts a timer for future calls.

3) If the request for spending is fulfilled, the time for using the service is extended. Part of the funds is spent on maintaining the timer, the rest of the funds are kept by the intermediary until the next timer is triggered (one subscription period).

4) When the timer is triggered (the first timer is triggered after one subscription period), the reserved funds are sent to the service provider and a new request is created to spend user funds.

5) If the user unsubscribes, the contract will find out about this through a callback and part of the reserved funds will be returned to the user, the remainder will go to the supplier, the timer is stopped.

1.4 SubscriptionTerms

This contract stores the terms of the subscription, as well as the code of the intermediary contract. The user must trust the intermediary's code, but even if the intermediary's code turns out to be malicious, the wallet will not allow you to write off funds at a time for more than one period of using the subscription.

1.4.1 Description of subscription terms

address debot_address - debot address, currently not used

uint256 verifier_pubkey - public key of the verifier, currently not used. It is assumed that the owner of this key can have access to any intermediary contract and resolve disputes between the user and the supplier.

address send_value_to - service provider address for sending funds there

address root_token_wallet - root TIP-3 token wallet address. Can be 0 if TIP-3 is not used.

uint128 value - payment amount.

uint128 gas_value - The amount of gas attached to the payment when spending funds, as well as when calling a callback. Recommended value 350_000_000 nanotokens.

uint64 period - payment period in seconds

uint32 max_payment_delay - the time that the service can be provided without receiving payment

string/bytes name - Service name.

string/bytes info - More detailed description.

1.5 RootOffchainTimer - Helper contract for deploying timers.

2. Contracts address net.ton.dev

debot:	0:3eee93d82002c1e2e44a208fdbc6430241cda93fc43e91b35891e1fd96e029ec
root timer:	0:b4a2d34eec146505e4166988a35ada1776faa7447974e9786bbdfd16cd8287f4
token 1 RTW	0:2397e4d02332dd23108974e6103d56864ae0571db86cb195f542159ea5754344
token 2 RTW	0:dccb2920d677e2587c79cb9b479d28d8ddd2bbfe0202dc7dc537b5406d32569a
user wallet:	0:309566fec98404c16b4121216b31009a4944f6752f56c9cc17dc3bbdcafbac6c
user TTW (token 1):	0:fc25273db06fbe99ab1bde017abb1b3302bd5d3de23ac1fa608b5aa5da5eb6b5
user TTW (token 2):	0:2c7c776e716f360e84e3d9e49e2cb30477b278743f64d119f3336daa5f0ac00e
terms 1 (without TIP-3):	0:7697311f0d5bb8814bbc19f9ae644e3845a2548d60951f74bcf36af7ff2c74db
terms 1 intermediary:	0:c4db4014a1520ad33c14b60b157ffb130a158292539983e6ed7f2e27ccdd3da3
terms 1 timer:	0:9a66398cff7354653366a3312fe37daed2e11d0696ae427dcb218a4d1a72e4bb
terms 1 provider wallet	0:d94edfa7023629a8aaad684fa08484138a887113274ed3fcef93e531131fdade
terms 2 (with TIP-3 token 1):	0:80699d6baeebf9b56df0dcb30c008de14342d650edb16fddbb068efd7826ca57
terms 2 intermediary:	0:14224dfdfd1b44bb5672276a9404cca8cfea3680fe9bff5111511f4eaa62bbb3
terms 2 intermediary TTW:	0:854ebab4f1d479c5b2945c4d57d65cb033c828a0c21a24a49f988d51226efaad
terms 2 timer:	0:621b5b45290836f724d59807e914a7b2b57f33b437f3ddde31bac3db2071d6db
terms 2 provider TTW:	0:1b2896c1f26458e88e98451c12173b1069f053ba20198ebcc94d8234662731f0
terms 3 (with TIP-3 token 2):	0:3902209bf22f330b37ea902eaf04c21d5d405f9c9f24a4b820c4d5c334930ff6
terms 3 intermediary:	0:cbc37de673d065e5a18503f335411b7602e9db1333f3acd4d87ce1db03b56d34
terms 3 intermediary TTW:	0:95ba092e254d34e4d4897ef9cfc4de8160093ca3a6ca2b38113e7a8f9e2bedeb
terms 3 timer:	0:fa759dbdbd6ca1f9980cc057904de95ecff0ba306bf8a39d35f11d5ba15aa4dc
terms 3 provider TTW:	0:3bd2d902f6c5ec4d5f5b39da3449bca43f87d302346f99dbbda46cd8be494f4b