Short description

Develop smart contracts to recurring payments ('subscriptions') on the Free TON, both (1) for TIP-3 tokens, and (2) for TON Crystal.

Contest dates

July 16, 2021 00:00 UTC — August 31, 2021 23:59 UTC

Voting cycle

15 days

Motivation

Monthly subscriptions are a key monetization channel for legacy web, and arguably they are the most healthy monetization channel for businesses (especially when compared to ad/surveillance) based models. But in the blockchain world there haven't been any successful full-featured crypto subscription payment systems yet. For these reasons, I think it's worth creating smart contracts to do 'subscriptions' on Free TON to facilitate user experience in crypto for both users and product owners (service providers).

- Users don't have to read a complex whitepapers to assess service in Free TON (as opposed to utility tokens)
- Users should have ability to subscribe and cancel anytime, without having to own any specific tokens
- Don't have to understand the product owners vesting schedules, crypto-economics, or anything more complex than the DeBot/Smart Contracts use case.
- Product owners can get a consistent, ongoing stream of cash flow while assessing the health of their business (subscribers, churn, growth)
- Product owners can focus on making customers happy, as opposed to splitting time between speculators and users
- Product owners utilize a proven, time-tested business model

Task

Implement on-chain Recurring Payments ('Subscriptions') smart contract system by which FreeTON users will be able to pay to unlock access to special DeBots/Smart Contracts/Off-chain service for a certain amount of time. Users must be able to recurly pay by TON Crystal or TIP-3 token (e.q. USDT) to subscribe for DeBots/Smart Contracts

and be able to cancel or pause subscription anytime. In turn, DeBots/Smart Contracts owners must be able to track subscribers to unlock DeBots/Smart Contracts usage or some of its features. Existing examples:
Solc:
https://github.com/tonlabs/ton-client-js/blob/master/packages/tests/contracts/abi_v2/Subscription.sol
C++: https://github.com/tonlabs/samples/tree/master/cpp/Wallet

Also some kind of validation (for off-chain services) and verification (for on-chain services) should be introduced also. It's necessary to ensure that the end user receives the service which he subscribed to. In the case of subscribing to on-chain services, some special contract can collect metrics about service and determine the subscription agreement fulfilled. In case of off-chain service, the system should be able to receive this information from external messages and validate it.

Requirements

- To develop a Recurring Subscription smart contract system using either Solidity or C++ languages
- Distributed smart contracts are always preferable, the participants should avoid operations on large data sets as much as possible (distributed programming paradigm) (github, youtube)
- Should allow product owners (service providers) to deploy a smart contract on FreeTON with parameters for a subscription including destination address, TIP-3 token address (if necessary), token amount, and period of recurrence.
- The subscriber should be able to control subscription status (starting, stopping, pausing, etc)
- Product owners should be able supply a link to the subscriber that is presented with the terms of the subscription to sign an transaction that is replayed on the defined period
- Should support the ability to subscribe with TON Crystal tokens as well as any other TIP-3 Token
- Should include DeBots for all system user interfaces
- Some kind of validation (for off-chain services) and verification (on-chain services) should be implemented
- A system should be deployed and tested on any test network(net.ton.dev, fld.ton.dev) and Jury should be able to access it for testing
- Solution should be committed to the Free TON community repo in accordance with the following document - https://github.com/freeton-org/readme
- Should include auto-tests designed as a smart contract or a script to test scenarios
- A solution should have a Free Software license (https://www.gnu.org/licenses/license-list.html 1)
- Your repo should include README with introduction and usage manual and deploy instructions

*any TIP-3 realisation allowed to use

Evaluation criteria

- **Safety first.** If the architecture is not able to keep user funds secure, it is a non-starter
- **Minimize UX Complexity.** A user should have closest as possible experience to legacy web but with all blockchain benefits
- **Opt-Out First.** A user should not need to perform actions on an on-going basis. It should be a “set it and forget it” experience. A subscriber performs one transaction to set the subscription in motion and transactions should happen “automatically” between the parties
- **Minimizes Gas Use**: The smart contract(s) should be as efficient of a consumer of on-chain resources as possible
- **No Staking.** A user should not have to stake funds. i.e. lock 1200 TON for a subscription that pays out 100 TON per month.
- Bonus: **Extensibility.** It’d be great if the smart contract could be extended to any recurring action (not just TIP3/TON Crystal transfers)
- Bonus: **Notifications.** It’d be great if the smart contract could notify users regarding payment date or insufficient balance on the wallet, send bills (e.q. with TON Surf)

Submission format

Submission must be published to a public FreeTON org github repository (https://github.com/freeton-org/readme)

PDF should be attached to the submission with the link to the repository’s contest branch along with the telegram id of the participant so that jury members can access the participant and ask questions.

Voting

- The juror must have a solid understanding of the described technology to provide a score and feedback. If not, the juror should choose to “Abstain”.
- Jurors or whose team(s) intend to participate in this contest by providing submissions lose their right to vote in this contest.
- Each juror will vote by rating each submission on a scale of 1 to 10 or can choose to reject it if it does not meet requirements or vote “Abstain” if they feel unqualified to judge.
- Jurors must provide feedback on submissions or lose their reward.
- The Jury will reject duplicate, sub-par, incomplete, or inappropriate submissions.
- The number of days for jury voting is hereby set at 10 day.

Contest Rewards

1. 💎 100’000
2. 💎 80’000
3. 💎 70’000
4. 💎 40’000
5. 💎 20’000

participants who receive 6th-10th place, receive points and do not have “rejected” votes will receive a consolation prize from 💎 10000 to 💎 2000 (with step equal to 2000 💎)

Minimum score to receive reward for 1 - 5 places is 6.

Total prizes: 350,000

Note: If the number of winning submissions is less than the number of rewards available, any remaining rewards are not subject to distribution and are considered void.

Jury Rewards

An amount equal to 5% of the total sum of all total tokens awarded to contest winners will be distributed among jurors who vote and provide feedback. This percentage will be awarded on the following basis:

- The percentage of tokens awarded to the jury will be distributed based on the number of votes each juror casts. For example, if one juror votes 50 times and another juror votes 5 times, the juror who votes 50 times will get 10 times more tokens than the juror who votes 5 times.
- Feedback is mandatory to collect any rewards.

Contest announcement and attracting new members rewards

An amount equal to 5% of the prize fund will be allocated to announcing partners who participates in announcing the contest in different media according with the following table: media list for technical contests announcements , to be distributed equally among them:
@renatSK
0:a2c66fbd01f0193c39127d1dd825e6d144d0581ca82a72a747d0af343b2c0b0b
@anovi
0:70759025778f37fd98ddb2b22aa8f6c54708a2917902cb3929e354d290b41d6a

@Alex770
0:ffc7897f7d234cb24f958aac097e59e16dad3e5ea147b4a214807f9274369be2
@moqub
0:a19bb3f75057490fd24c79f23c784db573bdf17dd75c955565b9ba3d439b5c3a
@lesnik13utsa
0:75a60f6c9aff6ecb6e2ce52ab16924248b09e3e11d5b4adb98893b7967591047
@Kronchs
0:a8b73825ef947fe5cc004761dc2eaf6400b23068df6c68859b7e7263dd7c02c5

Each participant of the contest, when submitting an application, will be asked through which announcing partner he/she learned about the contest. After the end of the contest, for each participant who won a reward, an amount equal to 5% of his/her reward will additionally be distributed:

- To the announcing partner who attracted him, if the referral was given during work submission;
- Or equally to all aforementioned partners of the announcement program, if the referral was not specified.

Procedural requirements

Accessibility. All submissions must be accessible for the jury to open and view, so please double-check your submission. If the submission is inaccessible or does not fit the criteria described, jurors may reject the submission.

Timing. Contestants must submit their work before the closing of the filing of applications. If not submitted on time, the submission will not count.

Contact. Each submission must have an identifiable contact that can be matched with your description. If you have not provided a forum description for discussion, then your application should contain links to your online persona, for example, a Telegram ID (preferred) or other direct contact information that can confirm that the submitted work is yours. In the absence of confirmation by the contestant of the authorship of the submitted work, the submission may be rejected.

Multiple submissions.

- Each contestant has the right to provide several submissions if they are all different from one another. If they are too similar, or in any way appear to be partially the same work done twice, or if they appear to be one whole body of work divided into parts to create the illusion of several submissions, jurors have the right to reject such submissions.
- If the contestant wants to make an additional submission to replace a previously published submission, the contestant must inform the jury about this fact and indicate which submission is the one to be judged. In this case, only the indicated

work will count. If the contestant fails to indicate which submission to judge, only the first submission that is uploaded by timestamp will count. The Jury will reject all others.