prps.io

Purpose is a not-for-profit project that is not looking for funding.
It is fully launched and operational.

The PRPS and DUBI tokens are integrated in multiple live commercial and charitable projects, most notably Clash of Streamers, a top grossing mobile game on the Google Play Store & Apple App Store.

Examples of integrations:

Clash of Streamers: top grossing mobile game
DubiEx: truly decentralized zero-fees exchange
KickstartNFT: not-for-profit NFT platform
Giving Works: crypto-based philanthropy

While not an official White Paper, this document provides an extensive technical overview.

Table of Contents

Overview **3**
Clash of Streamers 3
Technical Info **4**
Introduction 4
Purpose (PRPS) 5
Decentralized Universal Basic Income (DUBI) 6
Hodl 7
Finite Locking 7
Infinite Locking 8
Gasless Transactions 9
OptIn 9
Booster 9
Boosted Functions 10
Booster Message 11
Booster Fuel 11
EIP 712 12
Instant Transactions 13
Security Measures 13
Pending Transactions 13

Overview

The PRPS (Purpose) & DUBI (Decentralized Universal Basic Income) tokens are currently utilized in a number of products, and the team is not looking for funding. As such, this whitepaper seeks only to inform people about the technical aspects of the tokens, in addition to serving as a gateway to the projects that provide utility to the tokens.

Clash of Streamers

Clash of Streamers is a popular mobile game which features deep integration with PRPS & DUBI. Both of these tokens can be used on the Clash of Streamers crypto shop to buy in-game deals for better value than what is found in-game. Additionally, as the play-to-earn token of Clash of Streamers, DUBI can be used to trade, empower, and rent NFTs from within the game.

If you would like to learn more, feel free to check out any of the links below.

Clash of Streamers official links:

- Official Website
- Google Play Store
- Apple App Store
- Clash of Streamers PRPS & DUBI Crypto Shop

Community resources:

- Clash of Streamers Guides
- Clash of Streamers News

OpenSea NFT links:

- Clash of Streamers Hero NFTs
- Clash of Streamers Pet NFTs

Technical Info

Introduction

PRPS (Purpose) is an ERC20 compliant utility token on the Ethereum blockchain, and DUBI (Decentralized Universal Basic Income) is an ERC20 and BEP20 compliant utility token on Ethereum and Binance Smart Chain.

In addition to the standard ERC20 functionality, both tokens have additional features and optimizations that sets them apart from common ERC20 tokens.

The following optimizations have been made to make Purpose and Dubi as gas efficient as possible:

- On-chain state compaction to fit as much data as possible into a single storage slot (e.g. bit packing, limiting balances to 96 bits, ...)
- Storage reads and writes are reduced to the absolute minimum by, for example, deferring certain state updates (i.e. total supply when something gets burned/minted)
- Optimized inter-contract communication between Purpose, Dubi, and Hodl to reduce overhead (i.e. hardcoded addresses)

Another enhancement on top of the ERC20 standard is the addition of a Burned event. Traditionally, the ERC20 token standard does not specify how tokens should be taken out of circulation. The most common way is to transfer tokens to the zero address (0x0000000000000000000000000000000000000000, also called genesis) which acts as a burner wallet and only works as long as nobody knows the corresponding private key.

Purpose and Dubi take this a step further and have been designed from the ground-up to provide a sound way of burning tokens without relying on a burner wallet.

Anyone can burn tokens by calling `burn(uint256, bytes)` which additionally takes an optional data payload that gets included in the emitted Burned event. This optional payload enables individuals to publish arbitrary immutable data to the blockchain when burning tokens which can then be used by third-party developers (i.e. the Clash of Streamers crypto shop).

Purpose (PRPS)

The Purpose contract internally differentiates between locked and unlocked PRPS of a wallet. Whenever PRPS is locked using the Hodl contract, the unlocked PRPS balance decreases and the locked PRPS balance increases in return for minted DUBI. Once the PRPS gets released, the locked and unlocked balances return to their original values.

The `balanceOf` function of the Purpose contract returns the unlocked balance only. However, the Purpose contract provides an additional function called `hodlBalanceOf`. To get the total amount of PRPS owned by a wallet it is therefore necessary to take the sum of both function calls.

While it is not possible to transfer locked PRPS, it is possible to burn locked PRPS. For this, the burn function first tries to burn from the unlocked balance and any remainder from the locked balance.

Since PRPS can be locked to mint DUBI, it would be economically unwise to burn unlocked PRPS instead of locked PRPS when the latter yields DUBI.

To prevent these economic mis-decisions, PRPS automatically locks unlocked PRPS for an equivalent of 365 days to mint 4% DUBI before burning which effectively means only locked PRPS is ever burned. When burning already locked PRPS, DUBI is minted prorated based on the time passed and duration. Ultimately, whether burning locked or unlocked PRPS, the resulting DUBI is always the same.

Decentralized Universal Basic Income (DUBI)

The DUBI tokens in circulation are minted by PRPS token holders.

By locking up PRPS in a contract called Hodl, DUBI is minted proportionally to the lock duration with a minimum of 1 day and a maximum of 365 days. Locking for the maximum duration yields 4% of the locked PRPS worth of DUBI and is transferred to the beneficiary immediately.

After the locking period is over the locked PRPS can be released again by interacting with the Hodl contract which then transfers the tokens to the beneficiary. Afterwards, a beneficiary can decide to re-lock the tokens to mint more DUBI.

The theoretical maximum inflation of DUBI is therefore 4% per year of the total PRPS supply.

However, since its inception, the amount of PRPS getting locked is only close to half of the total supply resulting in an effective inflation of around 2-2.5% per year without taking into account further diminishing factors such as token holders who forgot about their locked tokens. This inflation rate is further offset due to constant burns found in PRPS & DUBI utility applications such as the Clash of Streamers cryptocurrency shop.

Hodl

The Hodl contract allows PRPS token holders to lock up their PRPS in return for DUBI. The theoretical inflation is a maximum of 4% of the total PRPS supply.

The main functions of the Hodl contract are:

- hodl
- release
- withdraw

The `hodl` function is used to lock PRPS. It takes a set of parameters which defines the amount of PRPS to lock, the duration, the beneficiary of the minted DUBI, and the beneficiary of the PRPS when unlocking again.

Internally, it distinguishes between the creator of a `hodl`, the DUBI beneficiary and the PRPS beneficiary.

The creator is equal to `msg.sender` in Solidity terms and provides the PRPS to lock. The DUBI beneficiary receives the minted DUBI, and the PRPS beneficiary receives the unlocked PRPS at the end of the locking period.

It is common for all three to be the same account where the creator receives the minted DUBI and the unlocked PRPS .

By allowing different beneficiaries, the Hodl functionality can be leveraged in a variety of ways such as pilot programs for Decentralized Universal Basic Income (DUBI).

Finite Locking

The minimum finite lock duration is 1 day, and the maximum finite lock duration is 365 days.

At the end of the lock duration the now expired PRPS can be unlocked by calling the `release` function. Any account may call this function to release the expired PRPS (e.g. to pay gas fees), but the PRPS beneficiary always ends up receiving all of it.

Infinite Locking

It is possible to lock PRPS forever without being able to release it again. By calling hodl with a duration of 0 days, the contract will treat the locked PRPS differently than normal finitely locked PRPS.

Infinite locking is equal to locking for 365 days and thus immediately mints 4% worth of DUBI, except that once 365 days passed it cannot be released again. Instead, withdraw can be called to mint DUBI equal to the time passed since the last withdrawal or initial time of the infinite lock, whichever is greater.

The total amount of minted DUBI from finitely locked PRPS and infinitely locked PRPS over the same timeframe is equal, however withdrawing DUBI from infinitely locked PRPS can happen gradually. Furthermore, the withdraw function can exceed the yearly maximum of 4% DUBI if the last withdrawal is older than 365 days.

For example, Bob infinitely locks 10,000 PRPS on 01.01.2021 and immediately receives 400 DUBI. Alice finitely locks 10,000 PRPS on 01.01.2021 and immediately receives 400 DUBI.

A year later on 01.01.2022, Alice releases the PRPS and re-locks it for another 400 DUBI.

Half a year later on 01.07.2022, Bob decides to withdraw.

18 months passed since the last withdrawal, thus the amount of minted DUBI is:

(18 / 12) * 4 = 6% = 600 DUBI

3 months later on 01.10.22 Bob decides to withdraw again and he receives

(3 / 12) * 4 = 1% = 100 DUBI

Another 3 months later on 01.01.23 Bob decides to withdraw again and he receives

(3 / 12) * 4 = 1% = 100 DUBI

Exactly 2 years later both received 800 DUBI.

Gasless Transactions

Unpredictable gas-fees and long confirmation times are not desired when a good user experience is crucial.

Traditional Ethereum contracts require everything to happen on-chain; also commonly referred to as Layer 1 (L1).

However, Layer 1 in the current form is unsuited for low gas-fees and fast confirmation times without compromising decentralization or security, simply because it cannot scale sufficiently due to technical limitations. Advancements in cryptography (e.g. ZK-Proofs) and base layer evolution (e.g. ETH 2.0) show promising results when looking at the future. At present time however, it is still a challenge to develop a low-cost and instant experience for end-users which often leads to a bad experience, or even poses an impossible barrier of entry.

OptIn

To overcome the present shortcomings of Layer 1 a different approach is necessary. For this reason, Purpose, Dubi and Hodl have been engineered from the ground-up with off chain - also called Layer 2 (L2) - compatibility in mind. The yet to be utilized foundation allows for instant and gas-less transactions without compromising security or decentralization.

The contract that makes all of this possible is called OptIn.

The OptIn contract is essentially a switch and only the contract owner can flip the switch. This happens irreversibly, once flipped the ownership is renounced leaving OptIn without an owner.

From there on, all contracts that are built with OptIn integration begin to function differently to enable support for instant and gas-less transactions. Naturally, Purpose, Dubi and Hodl are fully designed with OptIn integration.

Booster

OptIn by default is inactive and does nothing until the contract owner irreversibly flips the switch. There are no concrete plans to do this anytime soon.

Every wallet is then by default opted-in to gas-less and instant transactions. However, at any point in time a wallet can decide to opt-out with a grace period of up to 24 hours which can be compared to withdrawal periods of Layer 2 rollups.

Every opted-in wallet that wants to leverage instant and gas-less transactions is required to do this through a so-called Booster. A Booster is also just an externally owned account (EOA) and acts as a proxy to the contracts that support OptIn.

The default Booster address is defined by the OptIn contract owner during deployment, but a user can always decide to opt-out or change to a different Booster wallet if desired.

External services will provide the necessary tools for users to enable a frictionless interaction with Booster.

Boosted Functions

In essence, Booster uses digital signatures provided by users to perform actions on their behalf. A signature is verified on-chain and must be signed by a user and be addressed to the Booster they are opted-in to. Furthermore, signatures expire and are only valid as long as a user is opted-in. Built-in nonces guard signatures from being reused, also called replay-protection.

The contracts that can be called by Booster on behalf of a user are called Boostable.

Every boostable contract provides a set of functions which mirrors the opted-out counterpart of all state-altering functions.

While opted-in a user can still use non-boosted functions normally without going through Booster. But at the cost of the transaction not completing immediately to provide certain security guarantees for all participants involved.

The boosted functions of Purpose and Dubi consist of:

non-boosted	**boosted**
transfer	boostedSend
transferFrom	boostedTransferFrom
burn	boostedBurn

Likewise, the boosted functions of Hodl:

non-boosted	**boosted**
hodl	boostedHodl
release	boostedRelease

withdraw	boostedWithdraw

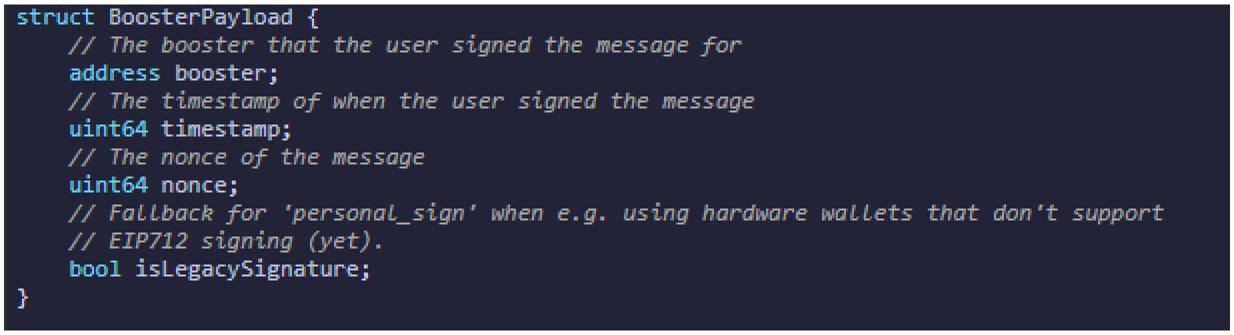
Furthermore, every boosted function also exists in a batched version, which works the same but allows to process multiple transactions at once to save gas costs for the Booster.

Booster Message

A Booster is required to provide a valid Booster Message when calling a boosted function or otherwise it is not permitted to perform an action on behalf of a user.

Every boosted function defines a specific message that a user is required to sign.

The so called BoosterPayload defines common fields that are mandatory and part of every boosted function message:



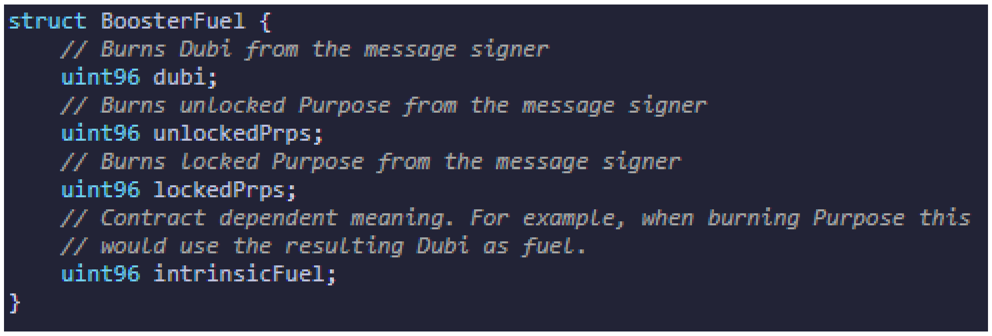
```
struct BoosterPayload {
    // The booster that the user signed the message for
    address booster;
    // The timestamp of when the user signed the message
    uint64 timestamp;
    // The nonce of the message
    uint64 nonce;
    // Fallback for 'personal_sign' when e.g. using hardware wallets that don't support
    // EIP712 signing (yet).
    bool isLegacySignature;
}
```

Booster Fuel

Due to high gas fees during times of congestion, it can be economically unviable for a Booster to submit transactions. Therefore, users can optionally incentivize a Booster by agreeing to burn fuel when the Booster submits their transaction.

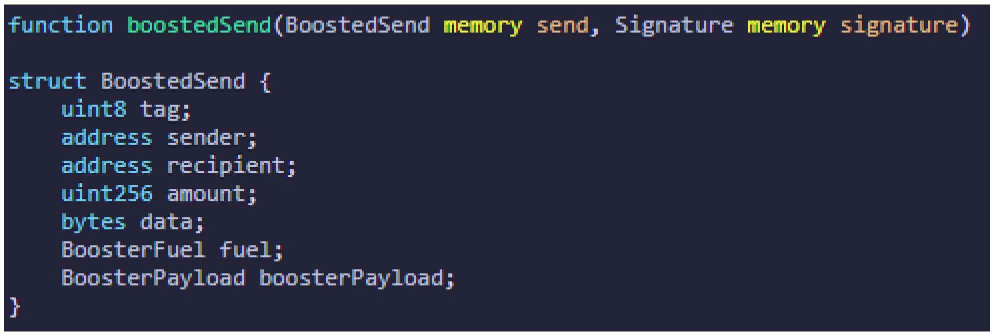
Booster fuel is always burned and never goes to the Booster. The interpretation of Booster fuel is up to the underlying contract; it may be Purpose or Dubi, but it can be virtually anything. The implementing contract decides what and how to burn fuel.

As a safety measure, the maximum fuel a user can burn per boosted transaction is capped to 10^18 WEI.



```
struct BoosterFuel {
    // Burns Dubi from the message signer
    uint96 dubi;
    // Burns unlocked Purpose from the message signer
    uint96 unlockedPrps;
    // Burns locked Purpose from the message signer
    uint96 lockedPrps;
    // Contract dependent meaning. For example, when burning Purpose this
    // would use the resulting Dubi as fuel.
    uint96 intrinsicFuel;
}
```

The booster message of the boostedSend function of Purpose and Dubi looks like the following:



```
function boostedSend(BoostedSend memory send, Signature memory signature)

struct BoostedSend {
    uint8 tag;
    address sender;
    address recipient;
    uint256 amount;
    bytes data;
    BoosterFuel fuel;
    BoosterPayload boosterPayload;
}
```

Ignoring Booster specific fields, the message contains the same input as the vanilla ERC20 transferFrom - namely, sender, recipient and amount.

Now, assuming that a signature is valid, a Booster can submit the message and execute a transferFrom on behalf of a user without the need for the user to pay gas or even own ETH.

EIP 712

Every boosted function uses the EIP712 standard for message signing to provide an improved user experience.

Booster messages can also be signed with hardware wallets such as Ledger by falling back to legacy signatures (personal_sign) when a signature cannot be verified from the EIP712 message hash, or when `isLegacySignature` is explicitly provided by the user.

Instant Transactions

So far only gasless transactions have been described. For instant transactions, it is necessary to build something on top of Layer 1. More concretely, a Layer 2 will enable instant transactions by replicating everything that is bound to happen on-chain the moment a user submits a transaction to the Booster.

Layer 2, yet to be developed, will presumably consist of a mirror chain that builds on top of the Layer 1 state of all boostable contracts.

The main idea is that Layer 1 and Layer 2 will be eventually consistent. This can be guaranteed by having certain on-chain security measures in-place that protect the Booster and user alike.

A Booster chooses when to sync with Layer 1 but it is expected to happen near real-time.

Security Measures

Layer 2 applies state changes before they are on-chain. It is therefore necessary to be certain that Layer 1 changes in a deterministic way.

This is crucial to prevent double-spends or conflicting state changes since every booster message must be applicable on-chain even if they aren't applied right away.

A user is not forced to use a Booster and can still interact with non-boostable version of the boostable contract. This leads to scenarios where it is possible that user A sends a message to a Booster and at the same time does something on-chain. To prevent incompatible state changes, the boostable contracts have safety measures that protect Booster from this scenario until it is safe to perform the user initiated on-chain transaction.

Any user initiated transaction while opted-in that alters the on-chain state without using a Booster is not finalized immediately. Instead, such transactions are considered pending until enough time passed or a booster approved it.

Pending Transactions

A pending transaction can be finalized or reverted. It can be reverted by a Booster if it conflicts with an already signed message of a user. This allows a Booster to prevent exploits and malicious actors from causing harm to other users.

By default pending transactions have a grace period of 24 hours which is equal to the opt-out period. A Booster may reduce the opt-out period, but can never increase it or may immediately finalize a transaction at it's own discretion. Otherwise, anyone is able to finalize a pending transaction after the grace period is over.

A pending transaction can be thought of as a partial transaction. For example, when a pending token transfer is executed, then the tokens of a user are moved to the contract. The contract acts as a middleman. The transfer is only completed, when a pending transaction gets finalized. Likewise, if it gets reverted the tokens are returned to the original owner.

When finalizing a pending transaction, the actual state change happens irreversibly and appears to a user like the result of a normal transaction. At this point the user is out of the picture and it is the responsibility of Booster to ensure Layer 2 handles this correctly.

If a user attempts to use a boosted transaction, while a pending transaction exists on-chain, Booster can use the signature from the boosted message to revert the pending transaction. This ensures that Booster can safely execute the boosted function or it would otherwise lead to a potentially conflicting state between Layer 1 and Layer 2.

A booster cannot revert a pending transaction if the signature is no longer valid. A signature is only considered valid if it has been signed by a user for the Booster and after the pending transaction happened.

Pending transactions can always be finalized without a grace period if the user is no longer opted-in.

Furthermore, all pending transactions of a user can only be finalized in first-in first-out order (FIFO). Likewise, reversal is only possible in last-in first-out order (LIFO).