Decentralized **U**niversal **N**etwork for **E**nterprises

Technical Whitepaper: The Dune Network

Dune Foundation & Origin Labs SAS, Paris, France
contact@{dune.network,origin-labs.com} https://dune.network

Revision b7fad8e
September 06, 2019

Abstract

The Dune Network is a novel platform for distributed applications over a blockchain. Dune Network is not a hard-fork of the Tezos ledger. It started with its own genesis block on June 24, 2019, but uses an extended version of Tezos' open-source software as a basis to create a new platform with a focus on the easy development of distributed applications, thanks to multiple programming languages and communication technologies with other blockchains. Dune Network is, both, a public ledger that uses a working Liquid Delegated Proof-of-Stake (LDPOS) with its Dune (DUN) token, and a free software platform that can be used to instantiate private blockchains, with a focus on security, correctness and ergonomy.

Introduction

Blockchain technologies are a huge opportunity for businesses, yet, a dangerous one. The short history of blockchains is sprinkled with stories of huge amounts of tokens being either stolen, or just lost, because of security or safety issues. It's all the more unfortunate that modern technologies can be used to prevent most of these problems, through strong static typing, program verification, or code certification.

Very few blockchains have taken security issues seriously. Here, we gather both, security and safety, under the more generic security term. The Tezos ledger is among them. It uses OCaml, a powerful, yet, very secure programming language as its building language, while exposing a simple and semantically sound low level language for smart contracts called Michelson. Also, it uses proof-of-stake (POS) instead of the energy consuming proof-of-work (POW), in a variant called Liquid Delegated Proof-of-Stake, for its consensus algorithm, and an on-chain governance for software amendments. Tezos is also known as one of the biggest ICOs of 2017.

Unfortunately, security technologies are often difficult to use: very few blockchain developers are able to develop smart contracts in a language designed for security and safety; it becomes a real issue for mass adoption. Within an ecosystem where, literally, hundreds of blockchains are competing for market share, this adoption barrier favors

safety-negligent technologies. The small number of delivered projects on the Tezos blockchain, compared to the number of announcements, and the general inertia of the whole Tezos ecosystem are evidence of this problem.

We have started developing the **Dune Network** on top of the Tezos software, as an independent blockchain, with a focus on **security**, but also, on **accessibility** for developers and **attractivity** for users, with an ambition to deliver the first fully distributed, universal, academic-level **business-oriented** blockchain platform.

All of our work on the Dune Network benefits from the vast experience our technical team gathered during the recent years, working on the prototype of Tezos, its ICO platform, the Tezos node before launch, TzScan (the most popular block explorer for Tezos), Liquidity (a smart-contract language for Tezos over Michelson), and Ironmin, an improved storage layer for Tezos.

Dune Network is a **multi-Language** blockchain: in addition to high-level languages on top of Michelson, it provides the ability to develop smart contracts in several other programming languages, ranging from low to high safety, aiding the developers progressively raise the safety and correctness standards of their contracts as they get acquainted with Dune's technology. Dune's initial ecosystem has been built to onboard, train, and proactively support developers targetting the Dune Network.

The Dune Network also features **on-chain project governance** by the community, via multiple mechanisms:

1. the election of a Dune Council at the Dune Foundation. (This council is responsible for leading the project from a software and business development point of view.),
2. a Treasury for grants,
3. and a Veto-system for software amendments.

The Dune Network consists of a complete ecosystem, with block explorers, software and hardware wallets, and numerous promising dApps, for notarization or financial assets for example. Our prime efforts will focus on building a rich ecosystem of applications, accessible across all modern media.

Contents

1 Dune Security and Safety

Focus: security and safety

In traditional financial entities, drastic measures and protocols are in place (in particular on computer and software infrastructure) to manage security risks. Surprisingly, while competing with these entities for the future of financial markets, few blockchains have taken security seriously. Stories of fraud or fraudulent loss are numerous, sometimes for tens or hundreds of millions of dollars[1,2].

Dune Network considers security as a critical concern. It is addressed through several means, some of which are already deployed and functional. Among them, is the choice of statically- and strongly-typed programming languages for the node implementation and the for Smart Contracts. Other means include the development of formal-methods tools for the verification and the certification of code, both, within the implementation, and the Smart Contracts

[1] `https://discover.ledger.com/hackstimeline`
[2] `https://github.com/kcchu/awesome-smart-contract-bugs`

1.1 Security through Programming Languages

All programming languages are not equivalent: some display a smooth learning curve, whilst others focus on productivity through expressiveness, or on robustness and safety. All languages make a compromise between such qualities.

The heart of Dune Network, the code of the node, is written in OCaml. OCaml is a 30-year old multi-paradigm programming language, with a prime focus on *expressiveness* and *correctness*.

Expressiveness comes from the use of a core language with semantics close to a mathematical specification. OCaml makes expressing complex ideas as easy and short as writing a mathematical formula.

Correctness comes from the use of strong static typing that enforces a strong discipline on the use of values and functions, ensuring that all function calls are verified at compile time and guaranteed to never fail for type errors.

The combination of these two qualities makes OCaml a language that is as concise and expressive as Python, yet, significantly safer without writing as many tests. OCaml users are usually recognizable by their faith in their code as soon as the compiler accepts it.

OCaml has been used for the development of the Dune Network node, client, baker and endorser. OCaml makes these tools robust, with significantly less bugs than for other blockchains, while still very easy to maintain and extend.

1.2 Formal Methods: Verification and Certification

Ethereum introduced the possibility to write Turing-complete Smart Contracts. This power comes with additional vulnerabilities, especially with programming languages having loose semantics, such as Solidity. Now, there are numerous examples of huge losses on Ethereum, starting with the USD 50 million loss of The DAO in June, 2016, and many others such as the Polkadot loss of USD 90 million in November, 2017. Most of these losses could have been avoided with the use of *formal methods*.

There are usually two methods of addressing such vulnerabilities:

- *Verification* attempts to prove that a program written in some programming language meets a set of properties. Model checkers or verification frameworks with intermediate languages, like Why3, combined with SMT solvers, such as Alt-Ergo, can be used to perform such a function.
- *Certification* attempts to directly write the programs in a formal language that guarantees that a set of properties are met. The most famous framework is Coq, a proof assistant that can be used to extract programs from mathematical proofs.

The Dune Network development team includes experts in software verification and formal methods. They will work on the design of a verification platform for Smart Contracts written in the Liquidity 2.0 language, enabling developers to specify and verify their own contracts with state-of-the-art verification tools.

2 Dune On-Chain Governance

Focus: governance

The on-chain governance in the Dune Network is divided in three parts:

1. Elections of the Dune Council
2. Treasury for Token Grants
3. A Referendum System

Although a few blockchains have adopted on-chain governance to amend the software of the blockchain itself, we think this governance model is a misunderstanding of the principles of software development, and open-source software in particular, for several reasons:

- Some software modifications require quick processes, *e.g.* when fixing bugs or pushing new important features for example. On-chain governance is too slow[3].
- Elections for software modifications require voters to understand what they are voting for. However, few of them have the skills and the will to analyze in-depth software modifications. Voters tend to blindly follow a few experts.
- Many decisions are taken at other levels, outside of this governance model, and often before software modifications are proposed. Voters may only have the choice between accepting a huge set of modifications, or none of them; however they are not able to choose among them, independently[4].

For such reasons (and many smaller ones), the Dune Network uses on-chain governance to manage not the *software*, but the project. This project is led by a group of people, called the *Dune Council*, taking the decisions on the development of the network. This group is part of the *Dune Foundation* and is elected by stake-holders, akin to a representative democracy.

2.1 The Dune Foundation

The Dune Network is managed by the Dune Foundation. The decisions at the Dune Foundation are taken by a representative group, the Dune Council. The Dune Council is elected on-chain by stake-holders of the Dune Network.

The Dune Council is in charge of:

- Formulating a roadmap for the Dune Network development,
- Access rights for developers to the Dune Network's software repository,
- Upgrades to the protocol,
- Deciding which projects in the ecosystem should be funded by direct grants (an on-chain *treasury* is also in place).

The Dune Foundation was initially funded by investors (Venture Capital). After the initial launch, the Dune Foundation will hold around 4% of the tokens. These tokens will be used to fund the development of the Dune Network and of its ecosystem.

Resultantly, there will be a need to provide a continuous funding of the Foundation the development of the network. Every transaction on the Dune Network will contribute 0.07% of the transaction amount, in the form of a *transaction reward*, to the

[3] Adopting an amendment in Tezos takes about two months... if accepted.

[4] In Tezos, there is no other choice for the Babylon amendment, and there was only a small meaningless difference between the 2 Athens amendments.

Dune Foundation account. This reward is formed by creating new tokens (like,*e.g.*, block rewards), which means it is not a transaction fee. This transaction rewards model makes the development cost of the network scale proportionnally with its use.

At any point in time, restrictions will limit the Dune Foundation to not own more than 25,000,000 tokens (less than 5% of the total supply). No transaction rewards will be credited to the Foundation if this threshold is reached.

The ratio of the transaction rewards may be adjusted by the Dune Council depending on the observed behavior, either, by increasing rewards with a view to source funds needed by the ecosystem, or to decrease to manage inflation.

2.2 On-Chain Elections for the Dune Council

The Dune Council will be composed of nine (9) members from the community, divided into five (5) groups:

- **Two** in the business group
- **Two** in the technical group
- **Two** in the marketing group
- **Two** in the community group
- **One** in the academic group

The above structure is subject to change in the future, to better adapt to the evolution of Dune Network.

An on-chain governance mechanism will be used to elect members of the Dune Council: two (2) members will be changed every six (6) months. Any member of the community will be able to candidate officially, and the two candidates with the most votes will be elected, as long as a minimal quorum (initially 20%) is reached. Votes are proportional to the number of rolls of tokens, with the same delegation mechanism explained in the Proof-of-Stake section.

A stand-in council will be in place at the beginning of the network, for a period of one year until the first election, expected for September 2020.

2.3 Treasury of Token Grants

The Dune Counil will be in charge of deciding which projects are to receive direct grants (funds) from the Dune Foundation.

Additionnally, an on-chain "Community Treasury" will be used to provide token grants to projects. Any project will be able to apply for such a grant; projects garnering the most votes will be chosen.

The Treasury will be formed, in part, by tokens resulting of transaction rewards received by the Dune Foundation and by the tokens directly allocated to the Treasury through the Dune Council.

2.4 Referendum System

Protocol changes in the Dune Network software are decided by the Dune Council.

The Dune Network provides an on-chain mechanism for token owners to express their opinion on the many decisions that have to be taken in the broad ecosystem that Dune Network is going to become.

A referendum system will be developed to allow Dune's users to submit proposals for consideration. Initially, proposals will be informal (short questions), but we anticipate for them to become more structured with adoption. With time, we anticipate an ecosystem to appear around this referendum system, either to help discuss the proposals in depth or as consistent parties to vote for them.

We also foresee the need at some point to decouple delegates for staking power of validation of blocks in proof-of-stake (and its reward system) from the delegates for staking power of voting, since financial incentives and general opinions may not be both aligned with a single delegate for many people. A token holder may seek to delegate its stake to a particularly-efficient baker for rewards, whilst seeking a different delegate for their vision of the platform's evolution.

3 Dune Smart Contract Languages

Focus: accessibility, attractivity, safety

The Dune Network aims to become a primary platform for blockchain-based businesses to build on. Applications targeting such a platform usually include one or more smart contracts on the blockchain. The programming languages used to write such contracts suffer a trade-off between expressivity, the ease of development, and safety (and security).

While most blockchains choose of a particular language, or a virtual machine (VM) only few languages can target, the Dune Network removes this trade off by proposing several programming languages by way of built-in support at the protocol level in the blockchain. The goal here is to attract new developers through expressive, yet, less secure Smart Contract languages, while allowing them, with experience, to switch progressively to safer alternatives.

Here, we present Liquidity 2.0, an expressive, yet, safe language, Love, its VM with long-term support in mind, DunePy, an easier target for new developers on Dune, and all legacy languages from the Tezos ecosystem.

3.1 Liquidity 2.0

Liquidity 2.0 is an improved and updated version of the Liquidity language, intended to be more powerful, like its ancestry language, OCaml, while still retaining the spirit of its predecessor. Being a strongly typed language, it is type-safe, while retaining conciseness and expressivenes thanks to a custom, static-type inference system. Similar to the Liquidity language, both OCaml and ReasonML (created by Facebook) syntaxes will be available. It adopts most of OCaml's features, as well as Dune-specific extensions, in particular, but not limited to:

- basic datatypes: integers, booleans, strings, amounts, keys, addresses, *etc.*
- composite datatypes: tuple, records, sums
- collections: lists, sets, maps
- iterators on collections: iter, fold, map
- recursive functions
- multiple entry points

- implicit storage through references
- storage accessors
- “library” modules

```
let [@storage] x = ref 0
let [@entry] incr () = x := !x + 1
let [@view] getCount () = !x
```

Figure 1: Simple Liquidity 2.0 program

Liquidity 2.0 facilitates the writing of contracts with multiple entry points. In addition, it enables the implementation of custom libraries that can be used within other contracts. The Liquidity 2.0 language is designed to be a living language; it will continuously evolve by tracking the most recent state-of-the-art Smart Contracts technologies. A Liquidity 2.0 contract consists of two components: a source code defining the methods of the contract and the signature, specifying the type of each entry point, and each public element.

3.2 Love VM — Long Term Support

Love (LOw-level language for Verification and Efficiency) is a coreML language designed to be Dune’s main Smart Contract target language. Contracts written in Liquidity 2.0 are stored on the blockchain in this format, interpreted by the Dune nodes. It is intended for Love to become a target language for many other higher-level languages.

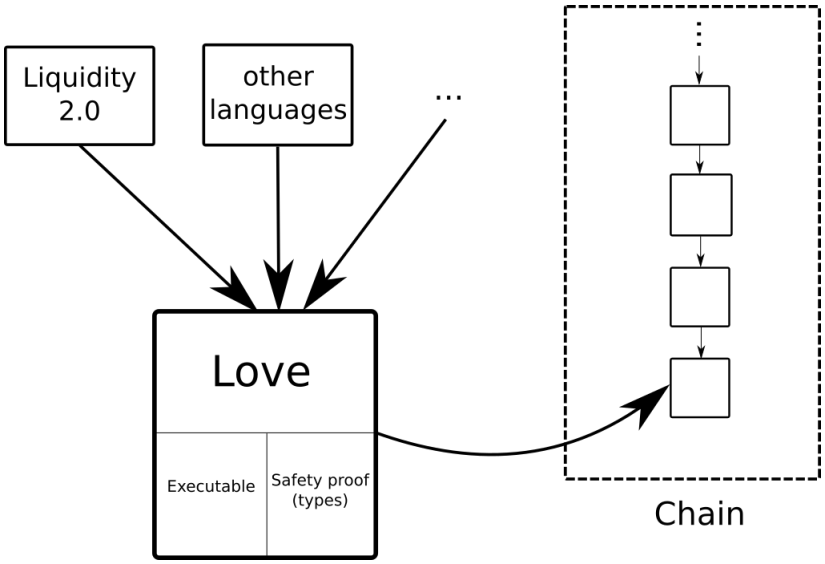


Figure 2: Love Integration as an intermediate language for multiple, type-safe, Smart Contract languages

Love defines an easily understandable yet complete language from which any high-level language (Solidity-like, Clarity, etc.) can be compiled to. In particular, it supports all of Yul's instructions, along with many OCaml-specific features. It provides several notable features:

- Safety: Love contracts are guaranteed to be free of many classical bugs, thanks to its strong static typing,
- Performance: Love contracts are executed efficiently, thanks to its high-level interpreter,
- Long Term Support: Love contracts will find unlimited (in time) support through the Dune Network. Other languages on the Dune Network may not always be backward compatible, but Love will always be.

Liquidity 2.0 is coupled with a compiler for Love.

3.3 DunePy

DunePy is a Python interpreter embedded in Dune which allows Smart Contracts to be written in Python and to be executed directly on the blockchain. Smart contracts written in DunePy are as simple and readable as standard Python programs. The DunePy environment comes with a library containing specific values and functions for Dune (*i.e.* cryptography, timestamps, DUN and contract operations, *etc.*). A simple example of a DunePy program is given in Figure 3. To assist programmers in developing safe and secure Python-based Smart Contracts, DunePy offers a programming environment for testing and simulating smart contracts whilst offline.

```
import Dune
class SimpleAuction:
    def __init__(self, _beneficiary, _biddingTime):
        self.beneficiary = _beneficiary
        self.End = Dune.now + _biddingTime
        self.hiBid = 0      # highest Bid
        self.PR = {}        # Pending Returns

    def bid(self, msg):
        assert Dune.now < self.End, 'Auction has ended'
        assert msg.value > self.hiBid, 'Bid higher'
        assert msg.sender != hiBidder

        self.PR[self.hiBidder] = self.hiBid
        self.hiBidder = msg.sender
        self.hiBid = msg.value

    def withdraw(self, msg):
        assert msg.sender != hiBidder
        assert self.PR[msg.sender] > 0

        amount = self.PR[msg.sender]
        self.PR[msg.sender] = 0
        if not Dune.send(msg.sender, amount):
           self.PR[msg.sender] = amount;
           return False
        return True
```

Figure 3: Simple auction in DunePy

3.4 Legacy Languages

Dune will retain the Michelson interpreter from Tezos, through which, it will remain compatible with all languages that target Michelson, in particular:

Michelson. It is a strongly typed stack language with S-exp syntax. Tezos' Smart Contracts are directly stored in such a format on the blockchain. Although Michelson features well-defined semantics, it is difficult to use and expensive to execute.

Liquidity 1.0. Liquidity 1.0 is the first high-level language developed on top of Michelson. Available with both, OCaml and ReasonML syntaxes, it features a decompiler to help auditing Michelson programs.

Ligo and SmartPy. These two recent, high-level languages are being developed within the Tezos ecosystem. Ligo has a Pascal-like syntax and one that is close to Liquidity 1.0, while SmartPy adopts a Python-like syntax; both of which compile to Michelson.

However, developers are encouraged to switch to Love-based languages, so as to take advantage of its expressiveness, efficiency and long-term support.

Tooling for formal verification of Love programs are on the medium-term roadmap of the Dune Network.

4 Dune DUN Token and Tezos Airdrop

Focus: accessibility

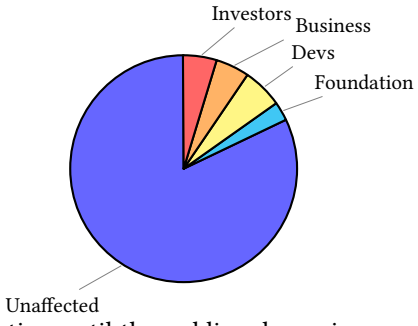
Dune Network's mainnet uses a token called DUN.

4.1 Initial Supply

An initial supply of 850 million DUNs has been pre-allocated when the network was created, on June 24, 2019.

The initial supply is divided into the following categories:

- 23,169,200 DUNs: Dune Foundation (2.7%)
- 48,360,000 DUNs: Development Team (5.7%)
- 40,300,000 DUNs: Business Team (4.8%)
- 40,300,000 DUNs: Investors (4.8%)
- 697,871,000 DUNs not affected (82.1%)



The network configuration halts reward generation until the public release, in September 2019.The tokens allocated to the Development and Business teams are contractually locked until September 2020.

4.2 Tezos Airdrop

At the beginning of September, Dune will airdrop DUN tokens to Tezos-holding addresses, to create an extended set of potential users, given the expressed interest of the Tezos community in Dune's technology.

The airdrop will be performed as follows:

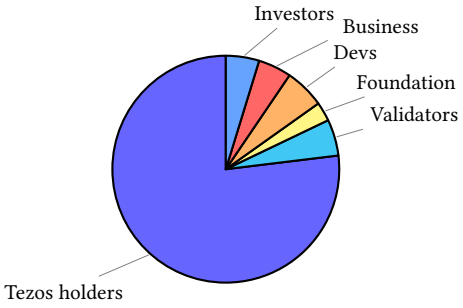
- A snapshot of all Tezos accounts will be taken at block number 600,000 of the Tezos mainnet. This block is expected to be baked around Sep 7-8, 2019.
- A curation will be executed on the snapshot:
  - All tokens held in originated accounts (KT-) will be moved to their associated manager account (tz-);
  - All tokens held in accounts with a balance below 100 XTZ will be burnt;
  - All tokens detained by the Tezos Foundation and associated parties will be burnt, to prevent malicious use[5].
- In the days after the snapshot, the accounts of the curated snapshot will be credited on Dune Network. The commitments (Tezos ICO accounts not yet activated) will be injected, too;

[5]We reserve the right to remove any account from the airdrop that threatens to cause disruptions to the network.

- All DUN tokens not allocated during the snapshot will be moved to specific contracts for the validator program. Such contracts will inhibit token movement, rendering the tokens as burnt with respect to the circulating supply. These tokens will be staked via blockchain validators chosen by the Dune Foundation, to improve the stability and resilience of the network;
- Once the airdrop has been completed, the platform will launch to the community via a public announcement, with a view to deter blind speculation of Tezos' XTZ.

Expected supply after the airdrop: 806 millions DUNs

- 23,169,200 DUNs: Dune Foundation (4.5%)
- 48,360,000 DUNs: Development Team (6%)
- 40,300,000 DUNs: Business Team (4.8%)
- 40,300,000 DUNs: Investors (4.8%)
- 653,870,800 DUNs: Tezos accounts (79.5%)
- 44,000,200 DUNs: Validators Program



Early stakers and validators will hold approximately 25% of the network's baking power during the first months. This kernel, Dune anticipates, will be sufficient to keep the network stable and live, with a maximal block time of 4 minutes between blocks assuming a pessimistic outlook, and 1 minute with a more realistic and optimistic view.

5 Dune Performance and Management

The Dune Network aims to become one of the most distributed blockchains. Distribution is a cornerstone of censorship resilience. We expect to have several hundreds of nodes running across the globe.

Dune Network currently implements three storage strategies for nodes:

- *Light nodes* or *Rolling nodes* are nodes that store only the most recent blocks, to be able to reply to queries on the current state and validate new blocks limiting resource utilization;
- *Full nodes* are nodes that store all of the blocks, but keep only the most recent computed states. Such nodes only reply to queries on the current state, whilst holding a complete copy of the blockchain;
- *Archive nodes* are nodes that store all of the blocks and all states. They are able to reply to queries at any level of the blockchain's history. These nodes are typically used by block explorers and other data analysis websites.

Although such storage strategies can be effective at reducing storage for most nodes at the beginning, Dune Network will also implement a new storage format, called *Ironmin*, that reduces storage by several orders of magnitude compared to the initial Tezos software, and improve performance of computation at similar levels.

Dune Network will also assist validators in securing their infrastructure. In particular, *signing proxies* will allow bakers to replicate their keys on multiple hardware wallets, making their infrastructure more resilient to network and electrical failures.

6 Dune Subtokens and Standard Contracts

Focus: accessibility, attractivity

Dune Network aims at being used to incorporate other (sub-)tokens in addition to DUN. Such (sub-)tokens will require some level of standardization within the ecosystem, in particular, the interface and storage of their Smart Contracts. Ideally, such contracts should be easy to interact with, from within the blockchain (from other Smart Contracts) and from outside (block explorers, dApps and other tools).

Dune Network will work on the standardization of these Smart Contracts from the beginning, providing, both, standard interfaces and open-source implementations for these contracts in a public repository.

Dune Network will offer avenues to ease the deployment and use of such sub-tokens. In particular, Dune Network will enable interactions and transactions of sub-tokens without any need to hold DUN tokens – the Smart Contract can pay for fees with their own supply – something that is impossible in most other blockchains.

6.1 Preliminary Proposed Standard

The node and protocol of the Dune Network already supports Michelson as a Smart Contract programming language, and Liquidity as a high-level, human readable, language[6].

The Dune Network does not consider other tokens as first class citizens of the blockchain; the platform and its ecosystem will have full support from the core developers, so that anyone can seamlessly create and use their own tokens.

We have already designed and reviewed a proposed standard for creating *fungible* tokens on the Dune Network. This standard is similar, in many ways, to the ERC20 standard[7] on Ethereum.

The standard below describes an interface (or *contract signature* in Liquidity), composed of entry points that must be implemented by any contract that follows the standard.

```
contract type Token = sig

  type storage

  val%entry transfer : address * nat -> _
  val%entry approve : address * nat * nat -> _
  val%entry transferFrom : address * address * nat -> _

  (* ------------- Storage access from outside ------------- *)
  contract type BalanceForwarder = sig
    type storage
    val%entry main : nat -> _
  end
  val%entry balanceOf : address * BalanceForwarder.instance -> _

  contract type AllowanceForwarder = sig
    type storage
    val%entry main : nat * nat -> _
```

[6] http://liquidity-lang.org

[7] https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20.md

```
  end
  let%entry allowance : address * address * AllowanceForwarder.instance -> _

end
```

`transfer`

A call to `transfer (dest, amount)` transfers `amount` of tokens from the account of the sender (*i.e.* `Current.sender ()`) to `dest`. This call must fail if the sender's account does not hold enough tokens to transfer.

`approve`

A call to `approve (spender, old_amount, new_amount)` allows `spender` to spend up to `new_amount` of tokens from the account of the sender (*i.e.* `Current.sender ()`). This call must fail if the `new_amount` is non zero and the `old_amount` is different from the previous allowance (this is made to prevent double spending from allowance).

`transferFrom`

A call to `transfer (origin, dest, amount)` transfers `amount` of tokens from the account of `origin` to `dest`. This call must fail if the sender (*i.e.* `Current.sender ()`) is not allowed to spend from `origin` or if its allowance is less than `amount`.

`balanceOf` and `allowance`

These are getter entry-points that do not modify (within their call, but not the forwarder contract) the storage/state.

`balanceOf (addr, forwarder)` forwards the current balance of `addr` to the contract `forwarder`. Typically, a forwarder contract is deployed for each outside contract `C` that wants to interact with this token contract, and their only use is to forward a value to a specific entry point of `C`.

Similarly, `allowance (addr, origin, forwarder)` forwards the current allowance of `addr` on the `origin` account together with the balance of `origin`'s full account contract to the contract `forwarder`.

6.1.1 Future Token Standards

The above proposed token standard is already useful for implementing cryptocurrencies, tokenization of some assets, *etc.* We want to propose and implement other standards for non-fungible tokens, such as Ethereum's ERC-721 or generic interfaces for multiple tokens like ERC-1155. Such token standards will launch in a second phase with input from the community and developers.

6.1.2 Standard in Liquidity 2.0

The new Smart Contract language Liquidity 2.0, is more flexible, efficient and easier to use; it is simpler to implement a token contract with it. In particular, it does not suffer from the same limitations as Michelson, which Liquidity inherits.

One feature, which makes writing a token contract simpler, is the possibility to define and use pure functions. The following signature describes this standard in the syntax of Liquidity 2.0

```
contract type Token = sig

  val[@init] init : 'a -> unit

  val[@entry] transfer : address -> nat -> unit
  val[@entry] approve : address -> nat -> nat -> unit
  val[@entry] transferFrom : address -> address -> nat -> unit

  val[@view] balanceOf : address -> nat
  val[@view] allowance : address -> nat * nat

end
```

The Dune Network will ensuring compatibility of interfaces, for both, the Liquidity and the Liquidity 2.0 versions of the standard from a user perspective.

6.2 Modular Standards

In addition to sub-tokens, there are already some (but few) very common patterns and use cases, in decentralized computation platforms, that can benefit from standardization. We aim to make each of these standards self-contained and composable by restricting each interface to encapsulate features for a single functionality. Subtyping at the contract signature level means that implementation for several standards can be combined to form larger smart contracts.

We briefly describe, in this section, some of the standards that we envision for the decentralized application platform Dune.

Tokens

We have already described an interface for a fungible token standard. There are several kinds of tokens that can be implemented using Smart Contracts, each of which offering specific utility.

KYC

Recent regulatory advancements surrounding cryptocurrencies and blockchain technology have pushed increasingly more platforms and services to require their users to undergo a *Know Your Customer* (KYC) process.The regulatory measure consists of essentially in verifying identities, subsequently, tying such records to electronic records on/off-chain. Most platforms offer various levels of KYC, varying in layered documental criteria, limiting the user in access as per their respective KYC approval.

When these records are reflected on-chain, dApps and other Smart Contracts can have access to this information.

Bond Issuance and Fundraising

Fundraising has recently become realistically-viable, with recent successes of ICOs in the cryptocurrency space. Interestingly, fundraising, and more particularly

crowdfunding, is also a popular financial tool in a diverse array of communities (including technology, art, healthcare, charitable organizations, *etc.*). Blockchains provide ways to set up these kinds of initiatives in a completely decentralized fashion (although not everything necessarily needs or benefits from decentralization), so having standards can lower to barrier to entry for anyone who wishes to set-up or take part in a fundraising campaign.

The major difference between the issuance of bonds and fundraising is that bonds typically entails a reward through coupon payment or interests, granted periodically. Bonds are tradable; they can also be seen as tokens representing financial assets that render debt rights. Standardizing interfaces for such assets is exciting as it allows for smoother interoperability for exchanges and greater liquidity.

Voting

Be it fundraising platforms, asset tokens, or bonds, participants (*i.e.*, token holders) often need to reach consensus on certain aspects (evolution, governance, ...). Some of these consensuses can be achieved via voting.

An election generally follows the same pattern (vote proposition, voting phase, votes tallying, and finally, acting on the vote results). Some votes, have several rounds, where the action is starting a subsequent election. Not all elections can take place on a decentralized blockchain, for instance, situations where anonymity is important, or where voting power cannot be enforced by the chain. It is, nonetheless, a powerful tool which can be deployed in many cases.

An implementation of the Voting standard can usually be combined with a token contract, tying voting power to ownership of tokens. It is common procedure to first create a snapshot of the voting power which can in turn be represented with a temporary sub-token – this means that voting power can be transferred.

```
contract type Vote = sig
  type storage

  val%entry createVote :
    string (* name *) *
    timestamp (* start_date *) *
    timestamp (* end_date *) *
    nat (* min_quorum *) *
    (string, unit -> operation list) map (* actions *) -> _
  val%entry vote : string (* vote_name *) * string (* vote_choice *) -> _
  val%entry tallyVote : string (* vote_name *) -> _
end
```

`createVote`

A call to `createVote (name, start_date, end_date, min_quorum, actions)` initializes a new vote on the contract which happens between `start_date` and `end_date`. `actions` is a map from choices to lambdas (`i.e.` continuations).

`vote`

The `vote` entry point can be called by anyone, simply, specializing the *ongoing* vote in which the caller wants to take part and his/her choice. Depending on the voting mechanism in place, this call can fail if the person has already voted.

`tallyVote`

When `tallyVote` is called (with the name of the vote), the winning choice is decided and the corresponding action (the *lambda*) is executed. Depending on the voting mechanism in place, this call can fail if there is a draw. In all cases, the ongoing vote is removed.

6.3 Integration with Wallets, Block Explorers and dApps

The Dune Foundation will ensure smooth integration of token standards. Tokens that follow the predefined standard will be transferable using traditional Dune wallets (*i.e.*, these wallets will hold tokens). Block explorers will identify transfers of sub-tokens, balances, etc.

The Dune Wallet, as well as the official explorer, DunScan, will support all current and future token standards.

6.4 Smart Contract Fees

A major factor that hinders adoption of sub-tokens that are developed on top of other smart contract platforms, such as Tezos and Ethereum, is the need to hold the platform's native cryptocurrency to cover any transfer expenses (which are subject to price fluctuations).

There are various solutions to this problem. Some of which are unsatisfactory; for instance, paying transaction fees in amounts of sub-tokens is a sub-par solution as this reduces the utility of the base cryptocurrency token and thus compromises on the security of the Network. We propose for the token smart contract to pay for the fees, effectively rendering interactions with it free for all users. This is an attractive feature for dApps developers as they can be sure that Dune will not get in the way of their platform. The aforementioned mechanism is implemented programmatically by Smart Contract (the contract decides on the fees payable for a particular call); any such expensed fees can be recovered (in *e.g.*, sub-token amounts), should the developer facilitate the option via the contract.

7 Dune's Proof-of-Stake Consensus Algorithm

Focus: safety, governance

Dune Network uses the same consensus algorithm, Emmy+, as inherited from its Tezos codebase.

Emmy+ is a Liquid Delegated Proof-of-Stake consensus: in Emmy+, token owners are responsible for creating new blocks on the blockchain. This action is called *baking* in Tezos' terminology. They are also responsible for *endorsing* the blocks created by other bakers. We encompass both of these actions into the generic term of *validating* blocks on the network.

Precisely, for Dune Network, as for Tezos, at every block, a seed is used to generate an infinite list of bakers for the next block and a small set of endorsers (currently 32). The list of bakers gives an order on baking rights, called *priority*, that allows them to bake a new block after a short delay: the baker at priority 0 can bake a new block one minute after the preceding block; the baker at priority 1 will have to wait a short while longer, and so on.

The protocol takes *stake* into account to determine the bakers and endorsers: tokens are gathered in groups of 10,000 DUN, in what is called a *roll*. The more rolls a baker has, the more likely he/she is to be chosen for baking or endorsing.

Bakers and endorsers are rewarded if they perform well. A reward is granted for every baked block and for every endorsement that is added to the blockchain. Fees also are earned by bakers. However, they are punished for any wrong-doing: bakers are required to offer a collateral; should any rules be violated, namely, double-baking or double-endorsing, penalties will be applied to the deposited funds. Overall, the rewards create an annual inflation, currently, set at a rate of 5%, approximately, per year.

All of these mechanisms offer a high guarantee against malicious actors forking a long enough chain with a view to conduct malicious actions; all participants are incentivised to follow the chain with the most staking power behind it, represented by the *fitness* of the chain.

Finally, validating requires some infrastructure: validators need to run their own nodes in a secured location to protect their cryptographic keys. Stake-holders may not be interested in supporting such an infrastructure. Also, their token holdings may be insufficient to qualify for a full roll. In such cases, Emmy+ allows them to delegate their tokens to other validators. Such delegations are usually done under some off-chain contract, whereby, the delegate will proportionally-share staking rewards to the token holders in return for the additional staking power.

8 Dune Ecosystem

Focus: accessibility, attractivity, Safety

The Dune Network is based on Tezos' software: it is backward-compatible with Tezos; it expands on its predecessor's technology to offer superior features. Dune Network inherits ecosystem: hardware and software wallets, baking software, compilers and interoperability libraries.

The additionnal features primarily focus on accessibility: new powerful and easier to use smart contract languages, a simpler and more realistic governance model, fee-less Smart Contracts, *etc.*

8.1 Explorer and Wallets

Dune Network features a complete ecosystem around its blockchain, including:

- DunScan (`https://dunscan.io/`) is a complete block explorer, developed by OCamlPro for Dune Network. It provides an intuitive interface for validators to understand the protocol and the performance of their baker.
- Dune Documentation (`https://docs.dune.network/`) is a common hub to gather all relevant documentation on Dune Network, for both end users, validators and application developers.

- Dune wallet (`https://wallet.dune.network/`) is a web wallet that interacts with public and private nodes to perform transactions on the network.
- Dune Ledger App (`https://ledger-app.dune.network/`) is an application for the hardware wallet, Ledger Nano S. A hardware wallet is essential to protect cryptographic keys.
- Dune Metal (`https://metal.dune.network/`) is an extension of Google Chrome and Firefox that manages cryptographic keys and sign transactions for dApps on the Dune Network.
- Dune Testnet allows developers to test their applications on a full-featured test network and create Testnet accounts using a Faucet (`https://faucet.dune.network/`).

8.2 Bakers and Dapps

Validators are an essential part of the Dune Network. To help them set up nodes and bakers, Dune Network provides several tools:

- Dune Baker Hub (`https://baker-hub.dune.network`) offers resources to support the set up of a fully-secured baker on the Dune Network.
- Dune Snapshots (`https://snapshots.dune.network`) provides snapshots to initiate fully functional nodes in minutes.

From the foundations of the projects architectural plan, our developers have built dApps to understand the challenges arising in doing so. Consequently, we already have a few of them deployed on the Dune Network:

- DrawIt (`https://dapps.dune.network/draw-it/`) is a very simple dApp that enables the user to draw a picture stored on the blockchain; it's displayed on `https://dunscan.io/`
- Notarize (`https://dapps.dune.network/notarize/`) is a notarization tool; it allows users to record their ownership of documents and scales to hundreds of notarization acts per second.
- Okkad (`https://dapps.dune.network/okkad/`) is a stable-coin contract that allows owners of the stable-coin to automatically convert their coins from and to fiat currencies.

8.3 Tezos Legacy

The Dune Network contains part of Tezos open-source software under the MIT license, with extensions and modifications under the GPLv3 license, the free-software license officially recommended by the Free Software Foundation. As a consequence, most of the tools developed for Tezos should work without modification to interact with the kernel of Dune Network. Only specific, identified, features of Dune Network (new smart contract languages, new RPCs, etc.) will not be directly compatible. In particular, all tooling developed around the Michelson languages, such as contracts certified with Mi-Cho-Coq can be used without alteration with Dune Network.

8.4 Dune Validator Program

Focus: accessibility

The Dune Foundation will launch a Validator Program to facilitate onboarding projects and bakers onto the Dune Network.

After the airdrop of September 2019, the DUN supply will be around 805 million DUNs. 850 million DUNs were initially allocated in June 2019, all of which won't contribute towards the actual supply. Tokens that are not allocated during the airdrop will be transferred to specific smart contracts, the Validator Contracts. Tokens on Validator Contracts are locked forever. They are, by definition, not part of the circulating supply. Instead, Validator Contracts can be delegated to Validators, as per the Validator Program. Each Validator Contract has a balance of 900,000 DUNs.

Within the Validator Program, a project or a baker can apply for a Validator Contracts delegation. If granted, the grantee will commit to keep 100,000 DUNs on a baker account for at least the duration of the grant (one year, subject to change). For the duration of the grant, the grantee will be delegated 900,000 DUNs. As a consequence, the grantee can expect a baking reward of about 50,000 DUNs following a year.

The validator program will be used to support several kinds of grantees:

- Public Delegators: the Validator Program guarantees a minimal delegation to bootstrap their activity;
- Community Projects: the Validator Program expects such projects to install a baker, offering the corresponding rewards allowing for both supporting the project and giving it voting power with respect to the evolution of the Dune Network;
- Companies: the Validator Program offers companies a means of connecting with the Dune ecosystem, a first step towards understanding how the Dune Network functions, and some visibility (on DunScan for example);

9 Dune Development Team

Dune's development team is a group of nine highly-skilled software developers, most of whom, hold PhDs in Computer Science, heavily-experienced with OCaml - the language in which Dune's node is developed.

The team has a long experience in the sphere of blockchain technology, and in particular the Tezos project, on which Dune is based. As former staff of OCamlPro, they were previously involved in the development of Tezos' prototype. Since 2017, the team has participated in improving the node for the launch of Tezos' mainnet in 2018, and in the development of Irontez, a dedicated version of Tezos for private deployments.

The team has been deeply involved in the Tezos ecosystem, developing the TzScan project, a popular Tezos block explorer on which DunScan is based, and the Liquidity project, the most popular Smart Contract language on top of Michelson. Furthermore the team has also worked on several projects targeting Tezos as a platform for their dApps. Many improvements in Dune are born out of this experience.

10 Roadmap and Calendar

The Dune Network was started on June 24, 2019 for early investors only. It will become public around September 10, 2019, after the airdrop of tokens from a Tezos snapshot.

This roadmap positions our milestones in three periods: the *Short Term* targets the public availability of Dune Network; the *Medium Term* targets the first year of existence of Dune Network; finally, the *Long Term* targets the following years.

10.1 Short Term (Public Release)

- Languages: Liquidity 2.0 on Love VM
- Web Wallet
- Complete Block Explorer, DunScan
- ERC20 Smart Contracts
- Validators' Smart Contracts
- *Metal* Browser Extension
- Application for Hardware Wallet (Ledger Nano)

10.2 Medium Term (First Year)

- Languages: DunePy on Love VM
- Ecosystem contributed by the community with Dune Foundation's support
- On-chain Governance for the Dune Council
- Community Treasury
- Tooling for dApps (on-chain authentication system, distributed storage, etc.)

10.3 Long Term

Rewriting the Dune node:

- Smart Contract Verification Platform
- Tendermint or other BFT consensus algorithms, *e.g.*, Algorand or HotStuff
- zk-SNARKs or other privacy layer
- New implementation in OCaml, with long term maintainability
- Implementation in other languages, for interoperability