Summary

This document describes a smart contract which implements a ledger that maps identities to balances. This ledger implements token transfer operations, as well as approvals for spending tokens from other accounts.

Approvable Ledger Interface

A contract which implements approvable ledger must have the following entrypoints:

- `(address :from, (address :to, nat :value)) %transfer`
- `(address :spender, nat :value) %approve`
- `(view (address :owner, address :spender) nat) %getAllowance`
- `(view (address :owner) nat) %getBalance`
- `(view unit nat) %getTotalSupply`

`%getBalance` and `%getTotalSupply` entrypoints have the same semantics as they do in FA1. This standard specifies additional authorization checks for `%transfer` entrypoint, as explicitly allowed by FA1.

See also:

- Syntax sugar explanation.
- Explanation of `view`.

Metadata

This document does not specify contract metadata. For a metadata specification, see TZIP-012

Errors

This document definines additional error types while following the approach for signalling errors described in FA1. The error details are annotated in this document for the sake of clarity, the annotations in errors are not required by this standard.

Entrypoints

transfer

As specified by FA1, this entrypoint credits the account of the address passed in the `"to"` parameter, while debiting the account corresponding to the `"from"` parameter.

This standard requires additional authorization checks to be performed prior to transfer:

- When called with `"from"` account equal to the transaction sender, we assume that the user transfers their own money and this does not require approval.
- Otherwise, the transaction sender must be previously authorized to transfer at least the requested number of tokens from the `"from"` account using the `approve` entrypoint. In this case current number of tokens that sender is allowed to withdraw from the `"from"` address is decreased by the number of transferred tokens.

In addition to `NotEnoughBalance` error specified by FA1, this entrypoint can fail with:

- `NotEnoughAllowance` - a given account has no permission to withdraw a given amount of funds. The error will contain a `(nat :required, nat :present)` pair, where `required` is the requested amount of tokens, `present` is the current allowance.

approve

This entrypoint called with `(address :spender, nat :value)` parameters allows `spender` account to withdraw from the sender, multiple times, up to the `value` amount. Each call of `transfer` entrypoint decreases the allowance amount on the transferred amount of tokens unless `transfer` is called with `from` account equal to sender.

If this entrypoint is called again, it overwrites the current allowance with `value`.

Changing allowance value from a non-zero value to a non-zero value is forbidden to prevent the corresponding attack vector, however this is not enough on its own to guarantee a safe allowance change.

▶ **How to safely change the allowance**

A token holder that intends to safely change the allowance for `X` to `K` token must:

1. read the current allowance `M` for `X` from the latest transaction `S`.
2. send a transaction `T` that sets the allowance to `0`.
3. wait for the blockchain to confirm that `T` is included.
4. scan all transactions between `S` and `T`.
5. calculate the allowance `N <= M` spent by `X` in those transactions.
6. set the allowance to `K - N` iff `N < K`.

This entrypoint can fail with the following errors:

- `UnsafeAllowanceChange` - attempt to change approval value from non-zero to non-zero was performed. The error will contain `nat :previous` value, where `previous` stands for the allowance value upon the contract call.

getAllowance

This view returns the approval value between two given addresses.

Token Metadata

Token metadata is intended for off-chain, user-facing contexts (e.g. wallets, explorers, marketplaces).

Token metadata for TZIP-7 should be presented in the same form as TZIP-12's token metadata and conforms to the same semantic rules. The `token-id` used for FA1.2 tokens must be `0`.

Token balance updates

One can make token balances indexable by storing them in a standardized way:

```
big_map %ledger address (pair nat (map address nat))
```

where key is the owner's address and value is the pair [amount, map of allowances].

Related work

ERC-20 is a standard used in Ethereum for implementing tokens. It also describes transfer and approval operations. The interface we propose here differs from ERC-20. Specifically, we have `transfer` and `transferFrom` analogies merged into a single entrypoint. Also, ERC-20 is known to suffer from some vulnerabilities, and we took them into account when implementing our interface.

TZIP-012 defines `FA2`: a contract standard that generalizes and extends `FA1.2`.