Vesting

Purpose of golos.vesting smart contract

The `golos.vesting` smart contract provides binding of the vesting to a token created by the `cyber.token` smart contract.

Overview

The `golos.vesting` smart contract supports the following operations:

- creation/«burning» of the vesting by a user;
- delegation of funds in the form of the vesting to another user;
- accrual of the vesting to authors and curators from the reward pool;
- replenishment of the pool with funds received from the `golos.emit` smart contract.

Creating (buying) vesting
The user can store her/his funds in the form of tokens and in the form of vesting. The user can exchange part of tokens for vesting at the `golos.vesting` smart contract at the current price. In the smart contract, there is a table in the table that stores data on the availability of funds in the form of vesting on the balance sheet of each user. When purchasing the vesting, the transfer of funds (transaction) is performed directly in tokens. The user transfers a certain number of tokens from her/his balance to the `golos.vesting` balance. The user's balance is received amount of vesting at the current price as soon as the transaction is completed.

Vesting withdrawal
The user can withdraw the existing vesting on her/his balance into tokens at the current rate. The transfer of funds (transaction) occurs directly in tokens. The amount of vesting is removed from the user's balance. Amendments to a certain amount of ongoing tests. Depending on the settings of the smart contract.

Delegation of funds in the form of vesting
User (delegator) can delegate a part of vesting to another user. Direct transfer of funds between

balances does not occur. In the `golos.vesting` table, an appropriate entry is made that only the recipient can use the delegated part of vesting. Data on the distribution of funds in the table are adjusted.

Return of delegated funds Upon expiration of the delegation term, the right to use delegated vesting is revoked from the recipient of the vesting. The delegated part of vesting goes into a «frozen» state (blocked). The delegator can use this part of vesting only after it is unlocked. Data about rights between users to use the vesting is modified in the `golos.vesting` table. If the operation to return the delegated vesting is created before the end of delegation, the delegator can not use the returned vesting before the delegated period deadline.

Parameters set in golos.vesting smart contract

Smart contract parameters are set using the `setparams` call. Each `vesting_param` parameter is a «variant» type containing a structure of one of the following types:

```
name: vesting_param,
    types: [
        struct vesting_withdraw,
        struct vesting_amount,
        struct vesting_delegation
]
```

When calling the `setparams` action for the first time, all three types of parameters are transitioned to the smart contract. When performing repeated calls (to change the settings) not all the parameters are allowed to be transferred.

vesting_withdraw
The `vesting_withdraw` parameter is used to set intervals to convert vesting into tokens. It is the structure:

```
struct vesting_withdraw : parameter {
    uint32_t intervals;
    uint32_t interval_seconds;
}
```

- `intervals` — number of intervals. The value to be set must be greater than zero.
- `interval_seconds` — the duration of the interval (measured in seconds, in the first release the duration of the interval will be one week). The value to be set must be greater than zero.

vesting_amount

The `vesting_amount` parameter is used to set minimum number of vesting to be converted. It is the structure:

```
struct vesting_amount : parameter {
    uint64_t min_amount;
};
```

- `min_amount` — minimum allowable (threshold) amount of vesting, below which it is not converted to tokens.

vesting_delegation

The `vesting_delegation` parameter is used to set the delegation parameters. The value is a structure of the following form:

```
struct vesting_delegation : parameter {
    uint64_t min_amount;
    uint64_t min_remainder;
    uint32_t return_time;
    uint32_t min_time;
}
```

- `min_amount` — the minimum allowable amount of vesting for delegation / return delegated. Specifying the number of vesting below this value is not allowed.
- `min_remainder` — minimum balance delegated. It is not allowed to delegate less than this value. As a result of the return of the delegated, the balance must be at least this value, or it must be zero.
- `return_time` — the time it takes to return the delegated funds (in seconds). The time is counted from the moment of withdrawing funds from the account to which they have been delegated, until the moment they are receipted on the delegators' balance.
- `min_time` — minimum delegation time (in seconds). The withdrawal of delegated funds is possible not earlier than this value.

Actions used in golos.vesting smart-contract

The `golos.vesting` smart contract contains the following actions: setparams, validateprms, create, retire, unlocklimit, withdraw, stopwithdraw, delegate, undelegate, timeoutrdel, timeoutconv, timeout, open, close and paydelegator.

---

setparams

The `setparams` action is used to configure the `golos.vesting` smart contract. The signature of this action is:

```
void vesting::setparams(
    symbol symbol,
    std::vector<vesting_param> params
)
```

Parameters:

- `symbol` — identifier of the vesting for which the settings are made (there can be several types of vesting in the smart contract. For each type, a separate table is created containing the data on the distribution of this type of vesting).
- `params` — array vector comprising of several parameters. Each parameter is bound to one of the characters of vesting and is a structure vesting_param .

> **Please note:**
> The order of the parameters in *params* is important: the parameters must be transmitted in the same order as described in the variant, and not all parameters are allowed to be passed. This *setparams* rule is valid in all the contracts.

---

validateprms

The `validateprms` action checks all the parameters for validity and, respectively, controls the appearance of errors in them. The signature of this action is:

```
void vesting::validateprms(
    symbol symbol,
    std::vector<vesting_param> params
)
```

Parameters:

- `symbol` — vesting identificator.
- `params` — array of several parameters.

---

create

The `create` action is used to create and launch tokens in the vesting form. This action looks as:

```
void vesting::create(
    symbol symbol,
    name notify_acc
)
```

Parameters:

- `symbol` — identifier of the created vesting, symbol of the associated token located in the `golos.token` smart contract.
- `notify_acc` — the name of the account (`golos.ctrl` smart contract) to which notifications will be sent when the balance changes. Each identifier of the corresponding account corresponds to a `notify_acc` account, which is a smart management contract. Each time the balance changes, the weights of the «leaders» change; for this, a smart management contract receives a notification.
  The successful completion of `create` action requires a two-step authentication:
- the right of account, which is the creator of the token with the same name as the vesting;
- presence of the signature of the token creator's account.
  Additionally, the presence of a previously created vesting with the same symbol is checked. Re-creation of the same name of vesting is unacceptable.

---

retire

The `retire` action is used for taking the «burnt» vesting out of circulation. The signature of this action is:

```
void vesting::retire(
    asset quantity,
    name user
)
```

Parameters:

- `quantity` — the number and type of vesting to be «burnt». The parameter comes in the form of a structure containing fields:
  - the quantity of vesting tokens to be «burnt»;
  - type (symbol) of the vesting.
- `user` — user, performing the action.
  The character of the token and its creator are determined by the type of the `asset` value. The amount (i.e. `quantity`) of vesting withdrawn from the system must be greater than zero, but not greater than the amount on the user's balance sheet, as well as the amount that is in the unlocked state.
  The `retire` and `create` actions are similar to the actions of `golos.token` smart contract. The difference is that the vesting is the token for shares and is not used as a cryptocurrency.
  To perform this operation, the rights of the account that issued this vesting are required. Only publisher can initiate withdrawal from the turnover of token tokens.

---

unlocklimit

The `unlocklimit` action is used to limit the «burnt» by the smart contract part of the vesting from the account balance. The user can allow the smart contract to withdraw (i.e.«burn») a certain part of the vesting from its balance sheet (for example, when «burning» the vesting for purchasing bandwidth resources). The user-selected part of the vesting for «burning» must be previously unlocked.
The signature of this action is:

```
void vesting::unlock_limit(
    name owner,
    asset quantity
)
```

Parameters:

- `owner` — the account name that unlocks and gives permission to «burn» vesting from its balance.
- `quantity` — specific vesting amount, safeguarded for «burning» and being in unlocked state. The specified quantity must be greater than zero or equal to zero. Zero number means no burning will be performed.
  To perform this action it is required the signature of the account `owner`.

---

withdraw

The withdraw action is used to convert the vesting into tokens. The signature of this action is:

```
void vesting::withdraw(
    name from,
    name to,
    asset quantity
)
```

Parameters:

- `from` — the account name that started the withdrawal process. In the process of withdrawing from the balance of this account, the amount of funds withdrawn in vesting.
- `to` — the account name on whose balance the amount of token funds is transferred.
- `quantity` — amount of funds allocated for withdrawal (i.e. conversion).

To perform this action it is required the signature of the account `from`.

Conversion can be performed both between different accounts, and between the same account (the names of `from` and `to` match). Conversion is performed not once, but for a certain

number of steps. The duration of one step is set by the action `setparams` (by default, it is set to one week, and the duration of the entire process (all steps) of conversion — 13 weeks). Converting vesting into tokens is performed in equal shares. Conversion by different parts is not allowed.

If the action `convert` is called again, the results of the previous conversion are canceled and new shares and a new balance are set in accordance with the current rate of vesting and token. In the process of conversion, internal actions are periodically called to verify the start of the payment of the next share and the execution of the current payment.

The conversion may imply the following restrictions:

- only undelegated part of the funds can be converted. It means that:
  - the vesting previously credited to the balance from the delegation operation cannot be involved in the conversion, since these funds must be returned upon completion of the delegation;
  - when vesting is marked as turned into the process of delegation, it cannot be converted until it is actually returned.
- the minimum allowed value to convert the vesting is `min_amount`; when it's not met, the vesting is never converted into tokens.

---

stopwithdraw

The `stopwithdraw` action is used for cancelling the «in progress» process of converting vesting to tokens.The signature of this action is:

```
void vesting::stopwithdraw(
    name owner,
    symbol symbol
)
```

Parameters:

- `owner` — the account name that launched the conversion process.
- `symbol` — a type of vesting for which the conversion process is canceled (conversion can be performed for each type of vesting separately and therefore a separate `stopwithdraw`

call can cancel only one separate process).

To perform this action it is required the signature of the account `owner`. As soon as `stopwithdraw` starts, payments for conversion are immediately terminated.

---

delegate

The `delegate` is used to delegate funds in the form of vesting. The signature of this action is:

```
void vesting::delegate(
    name from,
    name to,
    asset quantity,
    uint16_t interest_rate,
    uint8_t payout_strategy
)
```

Parameters:

- `from` — the account name that launched the process of delegating the vesting. Specified amount of this account's vesting will be marked as delegated.
- `to` — the account name to which funds are delegated. The balance of this account will be credited with the amount of funds in vesting. This amount of funds is absolutely refundable and will be returned to the `from` account upon completion of the delegation.
- `quantity` — amount of funds allocated for delegation.
- `interest_rate` — the percentage of payment to the account `from` that is deducted from curator fee of the account `to`.
- `payout_strategy` — identifier of delegation strategy (delegation options):
  - `to_delegator` — delegation, in which reward from curation (curator fee) are charged to the account `from`;
  - `to_delegated_vesting` — delegation, in which reward from curation (curator fee) is added to delegated funds (with compound interest).

The `interest_rate` parameter accepts one of two values: «0» — `to_delegator` or «1» — `to_delegated_vesting`.

To perform this action it is required the signature of the account `from`.

It may imply the following restrictions:

- account names in the `from` and `to` parameters must be different (if the user has two or more accounts, then delegation between them is allowed);
- the amount of funds allocated for delegation must not be less than the minimum value set in the parameters of the smart contract when calling `setparams`;
- `interest_rate` interest payment should not exceed the maximum delegation payment value set in the smart contract parameters when calling `setparams`;
- when the `delegate` action is called again for the same `from` and `to` pair, with the same vesting symbol, it is not allowed:
  - change the delegation strategy;
  - change interest payment `interest_rate`.
- tokens allocated for the conversion of vesting remain blocked and cannot be allocated for delegation.

---

undelegate

The `undelegate` action is used for partial or full refund of the delegated funds. The signature of this action is:

```
void vesting::undelegate(
    name from,
    name to,
    asset quantity
)
```

Parameters:

- `from` — the account name that started the process of vesting delegation.
- `to` — the account name to which funds were delegated.
- `quantity` — the amount of funds received from the delegation. Value must be greater than zero.

The `undelegate` action requires meeting of the following rules:

- transaction must be signed by `from` account.
- the presence of the actual body (object) of delegation, that is, the fact of the previous delegation between the sender and the recipient. Account balance `to` must contain funds delegated by account `from`.
- the amount of funds allocated for delegation must be specified in a relative value (how much it will decrease relative to the previous one). At the same time, the balance of funds delegated by the account `from` to the account `to`, should not be less than `delegation.min_remainder`.
- funds cannot be returned before the date set on the account when delegating.
- the amount of funds returned to the account's balance must not exceed the amount of the funds delegated to them.

Funds to the account's balance are not returned immediately, but only after the end of the set period (by default, this period is seven days). This period can be changed and set in the parameters of the smart contract when calling `setparams`.

---

timeoutrdel, timeoutconv and timeout

The `timeoutrdel`, `timeoutconv` and `timeout` actions are internal operations-related and are called by the smart contract to check the following data:

- the moment of converting funds from vesting to tokens;
- the time of return of the delegated amount.

The signatures of these actions are:

```
void vesting::timeoutrdel()
void vesting::timeoutconv()
void vesting::timeout()
```

Calling of these operations is not available to the user.

---

open

The `open` action is used to create a record in the database to store the user's balance. The user can open the record both for himself and for another account. The signature of this action is:

```
void vesting::open(
    name owner,
    symbol symbol,
    name ram_payer
)
```

Parameters:

- `owner` — account name (signature owner).
- `symbol` — parameter that uniquely identifies the vesting type.
- `ram_payer` — account name that signs the transaction and pays for the use of RAM.

Transaction must be signed by `ram_payer` account.

---

close

The `close` action is used for memory brush up occupied by a record. The signature of this action is:

```
void vesting::close(
    name owner,
    symbol symbol
)
```

Parameters:

- `owner` — account name, record owner.
- `symbol` — parameter that uniquely identifies the vesting type.

Transaction must be signed by the `owner` account.

To perform the action, it is required that the balance of all vesting, including the delegated one, equals to zero.

---

paydelegator

The `paydelegator` system action is called by `golos.publication` smart contract. The publication smart contract cannot imply any changes to the tables of the `golos.vesting` smart contract. Therefore, when closing a post, the publication smart contract calls the `paydelegator` and through it makes payments to the witnesses.

The signature of this action is:

```
void vesting::paydelegator(
    name voter,
    asset reward,
    name delegator,
    uint8_t payout_strategy
)
```

Parameters:

- `voter` — curator name whose curatorial fees were paid to delegates.
- `reward` — reward amount.
- `delegator` — account name, recipient of the reward.
- `payout_strategy` — identifier of delegation strategy. Accepts one of two values — «0» or «1». («0» — reward in tokens; «1» — reward in vesting).