Control

Overview

The Control (or `golos.ctrl`) smart contract implements logic for election of witnesses, including the following:

- a registering procedure for an account as a candidate for a witness;
- a voting procedure for election of a witness;
- determining a list of the most rated witnesses.

The `golos.ctrl` smart contract contains settings that apply to the Golos application as a whole. These settings can be used to the change parameters of any subsystems (for example, emission distribution between pools). Other Golos application smart contracts can access the `golos.ctrl` smart contract and get these settings, percentage ratios of the funds distributed by pools, limits on battery resources. Also, the smart contracts can obtain a list of the most rated witnesses with corresponding authority values. This makes it possible to verify the authenticity of actions certified by witnesses.

Parameters set in the Control smart contract

The `golos.ctrl` smart contract contains a kit of the parameters. To configure the contract, it needs to apply the action `setparams` action. The list of available parameters used by the contract is as follows:

```
ctrl_param, types: [
    symbol_code ctrl_token,
    name multisig_acc,
    uint16_t max_witnesses,
    struct multisig_perms {
        uint16_t super_majority,
        uint16_t majority,
        uint16_t minority
    },
    uint16_t max_witness_votes,
    struct update_auth {
        uint32_t period
    }
]
```

Parameters:

- `ctrl_token` — a token symbol that uniquely identifies a specific token:
    - token name. It should consist of a set of capital letters;
    - token rate. The token cost accuracy is set as a number of decimal digits.
- `multisig_acc` — an account name that controls a multisignature authorisation.
- `max_witnesses` — a maximum number of witnesses that can take a decision (for example, sign transactions) on behalf of the application. The `multisig_acc` parameter contains a list of such witnesses.
- `multisig_perms` — a required number of signatures from the most rated witnesses that have a permission to perform actions on behalf of the application. To change a parameter value it requires an appropriate level of permission. Different parameters require a different level of permission that is connected to their importance. There are three permission levels:
    - `super_majority` — a high level of permission. To get such level it needs to pick up at least «two-thirds plus one» votes of the most rated witnesses (if this parameter is "0", it takes default value — "2/3+1");
    - `majority` — an average level of permission. To get such level it needs to pick up at least «half plus one» votes of the most rated witnesses (if this parameter is "0", it takes default value — "1/2+1");
    - `minor_majority` — a low level of permission. To get such level it needs to pick up at least «one-third plus one» votes of the the most rated witnesses (if this parameter is "0", it takes default value — "1/3+1").
- `max_witness_votes` — a maximum number of witnesses for which a user can vote.
- `update_auth` — a parameter that specifies frequency of updating the authorization for the `multisig_acc` account:
    - `period` — an update period (in seconds). Re-authorization change for account is not performed if specified period has not passed since the last update.

Actions used in the Control smart contract

The `golos.publication` smart contract supports the following actions: setparams, validateprms, regwitness, unregwitness, stopwitness, startwitness, votewitness, unvotewitn, changevest

The setparams action

The `setparams` action is used to configure the `golos.ctrl` smart contract parameters. The action has the following form:

```
void control::setparams(vector<ctrl_param> params)
```

`params` — a value in the form of a structure containing fields with the setting parameters.

The validateprms action

The `validateprms` action checks parameters for validity and controls if there are errors or not. The `validateprms` action is called by the smart contract. It has the following form:

```
void control::validateprms(vector<ctrl_param> params)
```

`params` — a value in the form of a structure containing the parameters to be checked.

The regwitness action

The `regwitness` action is used to register candidates for witnesses. The action has the following form:

```
void control::regwitness(
    name witness,
    string url
)
```

Parameters:

- `witness` — a name of a candidate for witnesses.
- `url` — a website address from where information about the candidate can be obtained, including the reasons for her/his desire to become a witness. The address string must not exceed 256 characters.

Performing the `regwitness` action requires a signature of the witness candidate.

The unregwitness action

The `unregwitness` action is used to withdraw a user's candidacy from among the registered candidates to the witnesses. The action has the following form:

```
void control::unregwitness(name witness)
```

`witness` — the user name to be removed from the list of witnesses registered as candidates.

The `unregwitness` action can be called either by the candidate (in case of a withdrawal) or by a witness who found a discrepancy of the witness capabilities to desirable witness requirements, and mismatched data published on the web site with her/his relevant data.

Conditions for performing the `unregwitness` action:

- no votes for this witness candidate. Votes of all users who voted for this witness candidate should be removed;
- the transaction must be signed by the `witness` candidate himself.

The stopwitness action

The `stopwitness` action is used to temporarily suspend active actions of a witness (or a witness candidate). The action has the following form:

```
void stopwitness(name witness)
```

`witness` — account name of a witness (or a witness candidate) whose activity is temporarily suspended.

Conditions for performing the `stopwitness` action:

- the `witness` account should be active;
- a transaction must be signed by the `witness` account.

The `witness` account activity can be continued in case it has performed the `startwitness` action.

The startwitness action

The `startwitness` action is used to resume suspended witness activity (or a witness candidate activity). The action has the following form:

```
void startwitness(name witness)
```

`witness` — account name of a witness (or a witness candidate), whose activity is resumed.

Conditions for performing the `startfitness` action:

- the `witness` account activity should be suspended, that is, the operation `stopwitness` should be performed previously;
- a transaction must be signed by the `witness` account.

The votewitness action

The `votewitness` action is used to vote for a witness candidate. The action has the following form:

```
void control::votewitness(
    name voter,
    name witness
)
```

Parameters:

- `voter` — an account name that is voting for the witness candidate.
- `witness` — a name of the witness candidate for whom the vote is cast.

Doing the `votewitness` action requires signing the `voter` account.

Restrictions:

- the `witness` candidate name must first be registered through a call to `regwitness`;
- total number of votes cast by the `voter` account for all candidates should not exceed the `max_witness_votes` parameter value;
- it is not allowed to vote for a witness candidate whose activity is suspended (after the candidate has completed the `stopwitness` action).

The unvotewitn action

The action `unvotewitn` is used to withdraw a previously cast vote for a witness candidate.

The action has the following form:

```
void control::unvotewitn(
    name voter,
    name witness
)
```

Parameters:

- `voter` — an account name that intends to withdraw her/his vote which was previously cast for the witness candidate.
- `witness` — the witness candidate name from whom the vote is withdrawn.

It is allowed to withdraw a vote cast for a witness candidate whose activity is suspended (after the candidate has completed the `stopwitness` action).

Doing the `unvotewitn` action requires signing the `voter` account.

The changevest action

The `changevest` is an internal and unavailable to the user action. It is used by `golos.vesting` smart contract to notify the `golos.ctrl` smart contract about a change of the vesting amount on the user's balance. The `changevest` action has the following form:

```
void control::changevest(
    name who,
    asset diff
)
```

Parameters:

- `who` — an account name whose vesting amount has been changed.
- `diff` — a relative change of vesting amount.

The `changevest` action is called automatically each time in case the vesting amount is changed on a user's balance. The `golos.vesting` smart contract informs the `golos.ctrl` smart contract about this change. Because changing the vesting amount on the user's balance changes the weight of this user's vote, the witness rating that the user voted for will also be changed. The `golos.ctrl` smart contract corrects the rating of each witness based on received information about the change. The data of the smart contract table is not modified.

In case a list of the most rated witnesses is changed, `multisig_acc` authorization will automatically be changed too.