PoSRegister

PoSRegister PoW PoS PoW PoS

1. conflux PoW PoS PoW
2. CFX 1000
3. 1 PoS = 1000 CFX
4. vote power
5. `voteLock()` ng

- `staking`
- 1000CFX

register

PoW PoS , PoW 1000CFX PoS PoW 1000CFX 1 vote power

- PoS
- `conflux rpc local pos register --power 1`, `register`

- indentifier: PoS
- votePower
- blsPubKey: PoS blsPubKey
- vrfPubKey PoS vrfPubKey
- blsPubKeyProof: PoS blsPubKeyProof

increaseStake

PoW PoS , , CFX

- PoW PoS
- CFX 1000CFX

- votePower:

retire

PoW PoS , PoW PoS

- PoW PoS

- votePower:

getVotes

token votes token votes

- PoW PoS

- identifier: PoS

- totalStakedVotes token votes
- totalUnlockedVotes token votes

identifierToAddress

PoS PoW

- PoW PoS

- identifier: PoS

- address PoW

addressToIdentifier

PoW PoS

- PoW PoS

- address: PoW

- identifier PoS

```
// SPDX-License-Identifier: MIT
pragma solidity >=0.5.0;

interface PoSRegister {
    /**
     * @dev Register PoS account
     * @param indentifier - PoS account address to register
     * @param votePower - votes count
     * @param blsPubKey - BLS public key
     * @param vrfPubKey - VRF public key
     * @param blsPubKeyProof - BLS public key's proof of legality, used to against some
attack, generated by conflux-rust fullnode
     */
    function register(
        bytes32 indentifier,
        uint64 votePower,
        bytes calldata blsPubKey,
        bytes calldata vrfPubKey,
        bytes[2] calldata blsPubKeyProof
    ) external;

    /**
     * @dev Increase specified number votes for msg.sender
     * @param votePower - count of votes to increase
     */
    function increaseStake(uint64 votePower) external;

    /**
     * @dev Retire specified number votes for msg.sender
     * @param votePower - count of votes to retire
     */
    function retire(uint64 votePower) external;
```

```
    /**
     * @dev Query PoS account's lock info. Include "totalStakedVotes" and "totalUnlockedVotes"
     * @param identifier - PoS address
     */
    function getVotes(bytes32 identifier) external view returns (uint256, uint256);

    /**
     * @dev Query the PoW address binding with specified PoS address
     * @param identifier - PoS address
     */
    function identifierToAddress(bytes32 identifier) external view returns (address);

    /**
     * @dev Query the PoS address binding with specified PoW address
     * @param addr - PoW address
     */
    function addressToIdentifier(address addr) external view returns (bytes32);

    /**
     * @dev Emitted when register method executed successfully
     */
    event Register(bytes32 indexed identifier, bytes blsPubKey, bytes vrfPubKey);

    /**
     * @dev Emitted when increaseStake method executed successfully
     */
    event IncreaseStake(bytes32 indexed identifier, uint64 votePower);

    /**
     * @dev Emitted when retire method executed successfully
     */
    event Retire(bytes32 indexed identifier, uint64 votePower);
}
```

Revision #7
Created 14 June 2022 00:38:33 by Pana
Updated 28 June 2022 01:57:41 by Pana