Week11-9.13

conflux `ConfluxContext`, `PoSRegister`, `CrossSpaceCall`, `ParamsControl`

Day1 ConfluxContext

`epochNumber` , `posHeight` `0x0888000000000000000000000000000000000004`.

abi :

```
// SPDX-License-Identifier: MIT
pragma solidity >=0.4.15;

interface ConfluxContext {
    /*** Query Functions ***/
    /**
     * @dev get the current epoch number
     * @return the current epoch number
     */
    function epochNumber() external view returns (uint256);
    /**
     * @dev get the height of the referred PoS block in the last epoch
`    * @return the current PoS block height
     */
    function posHeight() external view returns (uint256);
    /**
     * @dev get the epoch number of the finalized pivot block.
     * @return the finalized epoch number
     */
    function finalizedEpochNumber() external view returns (uint256);
}
```

java sdk

```
    public static void test(String contractAddr, String caller) throws Exception {
        Cfx cfx = Cfx.create("https://test.confluxrpc.com");
        Account acc = Account.create(cfx, caller);
        Address address = new Address(contractAddr);
        String hash = acc.call(address, "epochNumber");
        cfx.waitForReceipt(hash);
        Optional<Receipt> receipt = cfx.getTransactionReceipt(hash).sendAndGet();
        if (receipt.isPresent()) {
□□□......
        } else {
           .......
        }

    }
```

Day2 PoSRegister

PoSRegister PoS

- `register` - pos pos
- `increaseStake` - pos
- `retire` - pos
- `getVotes` - vote
- `identifierToAddress` - pos pow
- `addressToIdentifier` - pow pos

`0x0888000000000000000000000000000000000005`

abi

```
// 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);
}
```

posRegister

java-examples

Day3 CrossSpaceCall

Conflux core space esapce core spac`CrossSpaceCall` space

- `createEVM` - espace
- `transferEVM` - espace
- `callEVM` - espace
- `staticCallEVM` - espace
- `withdrawFromMapped` - espace token
- `mappedBalance` -
- `mappedNonce` - nonce

`0x0888000000000000000000000000000000000006`

- abi

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

interface CrossSpaceCall {

    event Call(bytes20 indexed sender, bytes20 indexed receiver, uint256 value, uint256 nonce,
bytes data);

    event Create(bytes20 indexed sender, bytes20 indexed contract_address, uint256 value,
uint256 nonce, bytes init);

    event Withdraw(bytes20 indexed sender, address indexed receiver, uint256 value, uint256
nonce);

    event Outcome(bool success);

    function createEVM(bytes calldata init) external payable returns (bytes20);

    function transferEVM(bytes20 to) external payable returns (bytes memory output);

    function callEVM(bytes20 to, bytes calldata data) external payable returns (bytes memory
output);

    function staticCallEVM(bytes20 to, bytes calldata data) external view returns (bytes
memory output);

    function withdrawFromMapped(uint256 value) external;

    function mappedBalance(address addr) external view returns (uint256);

    function mappedNonce(address addr) external view returns (uint256);
}
```

CrossSpaceCall

java-examples

core space espace java-sdk

```
    public String getMappedEVMSpaceAddress() {
        String hexAddr =  this.getHexAddress();
        hexAddr = hexAddr.substring(2, hexAddr.length());
        byte[] t = Hash.sha3(Numeric.hexStringToByteArray(hexAddr));

        byte[] mappedBuf = new byte[20];
□□    System.arraycopy(t, t.length - 20, mappedBuf, 0, 20);

        return Keys.toChecksumAddress("0x" + BaseEncoding.base16().encode(mappedBuf));
    }
```

Day4 ParamsControl

`ParamsControl` CIP-94

- `readVote` -
- `castVote` -
- `readVote` - espace token
- `currentRound` -
- `totalVotes` -

`0x0888000000000000000000000000000000000006`

abi

```
// SPDX-License-Identifier: MIT

pragma solidity >=0.8.0;

interface ParamsControl {
    struct Vote {
        uint16 topic_index;
        uint256[3] votes;
    }

    /*** Query Functions ***/
    /**
     * @dev cast vote for parameters
     * @param vote_round The round to vote for
     * @param vote_data The list of votes to cast
     */
    function castVote(uint64 vote_round, Vote[] calldata vote_data) external;

    /**
     * @dev read the vote data of an account
     * @param addr The address of the account to read
     */
    function readVote(address addr) external view returns (Vote[] memory);

    /**
     * @dev Current vote round
     */
    function currentRound() external view returns (uint64);

    /**
     * @dev read the total votes of given round
     * @param vote_round The vote number
     */
    function totalVotes(uint64 vote_round) external view returns (Vote[] memory);

    event CastVote(uint64 indexed vote_round, address indexed addr, uint16 indexed
topic_index, uint256[3] votes);
    event RevokeVote(uint64 indexed vote_round, address indexed addr, uint16 indexed
topic_index, uint256[3] votes);
}
```

JS

v2 ~ V1

Revision #9
Created 13 September 2022 07:38:40 by Xianqi
Updated 16 September 2022 08:35:02 by Xianqi