Week15 - 10.17

java-sdk ~

```
    public static void pubsub() throws ConnectException {
        WebSocketService wsService = new WebSocketService("wss://test.confluxrpc.com/ws",
false);
        wsService.connect();
        Cfx cfx = Cfx.create(wsService);

        // add the filter
        BigInteger cur = cfx.getEpochNumber().sendAndGet();
        // construct the filter parameter
        LogFilter filter = new LogFilter();

        // filter details
//        filter.setFromEpoch(Epoch.numberOf(cur));
//        filter.setToEpoch(Epoch.numberOf(cur.add(new BigInteger("20"))));
//        // To filter address
//        List<Address> toFilterAddress = new ArrayList<Address>();
//        toFilterAddress.add(new
Address("cfxtest:aajb342mw5kzad6pjjkdz0wxx0tr54nfwpbu6yaj49"));
//        filter.setAddress(toFilterAddress);

        // subscribe epoch events
//             Flowable<EpochNotification> events1 = cfx.subscribeEpochs();
//             Disposable disposable1 = events1.subscribe(event -> {
//                // You can get the detail through getters
//                  System.out.println(event.getParams().getResult().getEpochNumber());
//                 System.out.println("epoch");
//            });
//
//            disposable1.dispose();
```

```
        // subscribe newheads events
        cfx.subscribeNewHeads().subscribe(event -> {
            // You can get the detail through getters
            System.out.println(event.getParams().getResult().getEpochNumber());
        }, error -> {
            error.printStackTrace();
        });

        // subscribe log events
//        cfx.subscribeLogs(filter).subscribe(event -> {
//            // You can get the detail through getters
//            System.out.println(event.getParams().getResult().getLogIndex());
//        }, error -> {
//            error.printStackTrace();
//        });
    }
```

conflux-java sdk web3j batch request Batch RPC tip

```
    public static void test() throws Exception{
        Cfx t = Cfx.create("https://test.confluxrpc.com");
        Web3j client = Web3j.build(new HttpService("https://test.confluxrpc.com"));
        Account acc = Account.create(t, "fjkaldsdjfasxjvzlkxjczxlkjfas"); //replace with your
private key or to export the account from the keystore.
        Account.Option option = new Account.Option();
        RawTransaction tx = option.buildTx(t, new
Address("cfxtest:aajb342mw5kzad6pjjkdz0wxx0tr54nfwpbu6yaj49"), acc.getPoolNonce(), new
Address("cfxtest:aar9up0wsbgtw7f0g5tyc4hbwb2wa5wf7emmk94znd"), null);
        RawTransaction tx1 = option.buildTx(t, new
Address("cfxtest:aajb342mw5kzad6pjjkdz0wxx0tr54nfwpbu6yaj49"),
acc.getPoolNonce().add(BigInteger.ONE), new
Address("cfxtest:aar9up0wsbgtw7f0g5tyc4hbwb2wa5wf7emmk94znd"), null);
        RawTransaction tx2 = option.buildTx(t, new
Address("cfxtest:aajb342mw5kzad6pjjkdz0wxx0tr54nfwpbu6yaj49"),
acc.getPoolNonce().add(BigInteger.TWO), new
Address("cfxtest:aar9up0wsbgtw7f0g5tyc4hbwb2wa5wf7emmk94znd"), null);
```

```
        RawTransaction tx3 = option.buildTx(t, new
Address("cfxtest:aajb342mw5kzad6pjjkdz0wxx0tr54nfwpbu6yaj49"),
acc.getPoolNonce().add(BigInteger.valueOf(3)), new
Address("cfxtest:aar9up0wsbgtw7f0g5tyc4hbwb2wa5wf7emmk94znd"), null);

        tx.setValue(BigInteger.valueOf(100));
        String signedTx = acc.sign(tx);
        String signedTx1 = acc.sign(tx1);
        String signedTx2 = acc.sign(tx2);
        String signedTx3 = acc.sign(tx3);

        BatchResponse resp = client.newBatch()
                .add(t.sendRawTransaction(signedTx))
                .add(t.sendRawTransaction(signedTx1))
                .add(t.sendRawTransaction(signedTx2))
                .add(t.sendRawTransaction(signedTx3))
                .send();

        System.out.println(resp.getResponses().get(0).getResult());
    }
```

web3j client -> rawtransaction -> -> web3jclient batch ->

java-sdk nonce

web3j client -> rawtransaction -> -> web3jclient batch

ERC20

```
    public static void batchTx() throws Exception {
        String addr = "cfxtest:acffj2hwbrwbsxuk56jne9913xvmwj5g4u7syhbfr2";
        Cfx cfx = Cfx.create("https://test.confluxrpc.com");
        Web3j client = Web3j.build(new HttpService("https://test.confluxrpc.com"));
        Account acc = Account.create(cfx, "fjkaldsdjfasxjvzlkxjczxlkjfas"); //replace with
your own private key.
        BigInteger amount = BigInteger.valueOf(100);
```

```
        String data = call(new Address(addr), "transfer", new
Address("cfxtest:aar9up0wsbgtw7f0g5tyc4hbwb2wa5wf7emmk94znd").getABIAddress(), new
Uint256(amount));

        Account.Option option = new Account.Option();
        RawTransaction tx = option.buildTx(cfx, new
Address("cfxtest:aajb342mw5kzad6pjjkdz0wxx0tr54nfwpbu6yaj49"), acc.getPoolNonce(), new
Address(addr), data);
        RawTransaction tx1 = option.buildTx(cfx, new
Address("cfxtest:aajb342mw5kzad6pjjkdz0wxx0tr54nfwpbu6yaj49"),
acc.getPoolNonce().add(BigInteger.ONE), new Address(addr), data);
        RawTransaction tx2 = option.buildTx(cfx, new
Address("cfxtest:aajb342mw5kzad6pjjkdz0wxx0tr54nfwpbu6yaj49"),
acc.getPoolNonce().add(BigInteger.TWO), new Address(addr), data);

        String signedTx = acc.sign(tx);
        String signedTx1 = acc.sign(tx1);
        String signedTx2 = acc.sign(tx2);



        BatchResponse resp = client.newBatch()
        .add(cfx.sendRawTransaction(signedTx))
        .add(cfx.sendRawTransaction(signedTx1))
        .add(cfx.sendRawTransaction(signedTx2))
        .send();

        System.out.println(resp.getResponses().get(0).getResult());

        }

    public static String call(String method, Type<?>... inputs) throws Exception {

            Function function = new Function(method, Arrays.asList(inputs),
Collections.emptyList());
            String data = FunctionEncoder.encode(function);



        return data ;
    }
```

&

call reque`DecodeUtil.decode`tch rawdata

```
    public static void queryInfo() throws Exception{
        Cfx t = Cfx.create("https://test.confluxrpc.com");
        Web3j client = Web3j.build(new HttpService("https://test.confluxrpc.com"));

        BatchResponse resp = client.newBatch()
                .add(t.getBestBlockHash())
                .add(t.getBalance(new
Address("cfxtest:aajb342mw5kzad6pjjkdz0wxx0tr54nfwpbu6yaj49")))
                .add(t.getEpochNumber())
                .send();


System.out.println(Numeric.decodeQuantity(resp.getResponses().get(2).getResult().toString()));
        System.out.println(resp.getResponses().get(0).getResult().toString());

System.out.println(Numeric.decodeQuantity(resp.getResponses().get(1).getResult().toString()));
    }

//  ERC20
    public static void batchQueryContract() throws Exception {
        String addr = "cfxtest:acffj2hwbrwbsxuk56jne9913xvmwj5g4u7syhbfr2";
        Cfx cfx = Cfx.create("https://test.confluxrpc.com");
        Web3j client = Web3j.build(new HttpService("https://test.confluxrpc.com"));
        ContractCall call = new ContractCall(cfx, new Address(addr));


        ContractCall call1 = new ContractCall(cfx, new Address(addr));
        ContractCall call2 = new ContractCall(cfx, new Address(addr));
        ContractCall call3 = new ContractCall(cfx, new Address(addr));

        BatchResponse resp = client.newBatch()
                .add(call.call("name"))
                .add(call1.call("symbol"))
```

```
                .add(call2.call("totalSupply"))
                .add(call3.call("decimals"))
                .send();
        String name = DecodeUtil.decode(resp.getResponses().get(0).getResult().toString(),
Utf8String.class);
        String symbol = DecodeUtil.decode(resp.getResponses().get(1).getResult().toString(),
Utf8String.class);
        BigInteger decimals =
DecodeUtil.decode(resp.getResponses().get(3).getResult().toString(), Uint8.class);
        BigInteger totalSupply =
DecodeUtil.decode(resp.getResponses().get(2).getResult().toString(), Uint256.class);

        System.out.println(name);
        System.out.println(symbol);
        System.out.println(decimals);
        System.out.println(totalSupply);
        }
```

ERC4907

ERC4907 ERC721 ERC721 NFT

NFT ERC4907

ERC4907

ERC4907

User owner owner NFERC490 owner user

ERC49(user N expire

1. A A NFT
2. A NFT
3. B B NFT A NFT B
4. A NFT

```
interface IERC4907 {

    // Logged when the user of an NFT is changed or expires is changed
    /// @notice Emitted when the `user` of an NFT or the `expires` of the `user` is changed
    /// The zero address for user indicates that there is no user address
    event UpdateUser(uint256 indexed tokenId, address indexed user, uint64 expires);

    /// @notice set the user and expires of an NFT
    /// @dev The zero address indicates there is no user
    /// Throws if `tokenId` is not valid NFT
    /// @param user  The new user of the NFT
    /// @param expires  UNIX timestamp, The new user could use the NFT before expires
    function setUser(uint256 tokenId, address user, uint64 expires) external;

    /// @notice Get the user address of an NFT
    /// @dev The zero address indicates that there is no user or the user is expired
    /// @param tokenId The NFT to get the user address for
    /// @return The user address for this NFT
    function userOf(uint256 tokenId) external view returns(address);

    /// @notice Get the user expires of an NFT
    /// @dev The zero value indicates that there is no user
    /// @param tokenId The NFT to get the user expires for
    /// @return The user expires for this NFT
    function userExpires(uint256 tokenId) external view returns(uint256);
}
```

The userOf(uint256 tokenId) function MAY be implemented as pure or view.

The userExpires(uint256 tokenId) function MAY be implemented as pure or view.

The setUser(uint256 tokenId, address user, uint64 expires) function MAY be implemented as public or external.

The UpdateUser event MUST be emitted when a user address is changed or the user expires is changed.

The supportsInterface method MUST return true when called with 0xad092b5c.

EIP-4907

Revision #6
Created 13 September 2022 07:37:22 by Xianqi
Updated 21 October 2022 03:25:12 by Xianqi